We Rate Dog

Data-wrangling project

Tien Duong
08-Aug-2020

Table of Contents

Introduction

Nothing more interesting than to analyze human's longest best companion. By using datas col from a twitter page `WeRateDog` twitter's archives, this data handed down as a starting point to for the current project. The aim for this project is to demonstrate my skill in data wrangling. structure procecure in this project will contain of asessing, storing, analyzing, and visualizing th data.

* `weratedog` is a twitter page with millions of followers. The page share and rate video and picture of dogs. The files collected from this page are `twitter_archive_enhanced.csv` , `image_predictions.tsv` , and `tweet_json.txt` . The property of this dataset consist 5000+ tweets but not all are dogs rating, some of the tweet are retweets. One unique about th rating of this `weratedog` page is the rating system which where the numerators are greater the denominators.

In [1]:
```
# Installing the needed package
import pandas as pd
import numpy as np
import requests
import tweepy
import json
import re
import matplotlib.pyplot as plt
import warnings
import seaborn as sns
%matplotlib inline
```

Gathering Data

In [2]:
```
# Using open and request to open the image-predictions file in tsv
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/59
with open('image-predictions.tsv', 'wb') as file:
    image = requests.get(url)
    file.write(image.content)
# Read the tsv file into pandas dataframe
image_prediction = pd.read_csv('image-predictions.tsv', delim_whit
```

In [3]:
```
# API accessing using tweepy API

import tweepy

consumer_key = 'hOmOpuXwks9L8zWJCZFmHR1eX'
consumer_secret = 'Ys9NxQqHe8wLwe5YoN5lkk8AO8eeZAUGSv9lFCz1rMzNEE8
access_token = '1291601114379374592-gtw5uV0g9ydTO8WiYVAugGAgcRyMGv
access_secret = 'b6IaLdZLPC4urs5WHQwLU7nm15OpLiOtzpuz4g6Hkffor'

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

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

In [5]:
```
# Reading in the twitter archives file into pandas dataframe
twitter_archive = pd.read_csv('twitter-archive-enhanced copy.csv')
```

In [25]:
```
# Using twitter_archive tweet id to query line by line into a dict
Id = list(twitter_archive['tweet_id'])

data = {}

for tweets in Id:
    try:
        tweet_status = api.get_status(tweets, waitwait_on_rate_lir
        data [str(tweets)] = tweet_status._json
    except:
        print('Error:', tweets)
```

```
Error: 844704788403113984
Error: 842892208864923648
Error: 837366284874571778
Error: 837012587749474308
Error: 829374341691346946
Error: 827228250799742977
Error: 812747805718642688
Error: 802247111496568832
Error: 779123168116150273
Error: 775096608509886464
Error: 771004394259247104
Error: 770743923962707968
Error: 759566828574212096

Rate limit reached. Sleeping for: 742

Error: 754011816964026368
Error: 680055455951884288

Rate limit reached. Sleeping for: 776
```

In [26]:
```
# Create tweet_json text file with the new dictionary
with open('tweet_json.txt', 'w') as f:
    json.dump(data, f)
```

In [6]:
```
# Extracting the columns of interest and file it into a list
with open('tweet_json.txt') as file:
    data = json.load(file)
df_list = []

for ID_tweet in data.keys():
    retweet = data[ID_tweet]['retweet_count']
    favorite = data[ID_tweet]['favorite_count']
    df_list.append({'ID': ID_tweet,
                    'retweet': retweet,
                   'favorites ': favorite})
```

In [7]:
```
# Using the newly made list and insert it into pandas DataFrame
retweet_favorite = pd.DataFrame(df_list)
```

Assess the Data

After extraction of the `image_prediction` , `twitter_archive` , and `retweet_favori` and stored stored into pandas dataframe for the next step in this data analysis. The data assessment will consist of two main elements of focus,looking for `quality` and `tidiness` inside each dataset. Quality and tidiness data assessment is to look through data contents us either visual or programtic.

Quality: Make observation of data content for missing , confusing data description and etc...

Tidiness: Look for through the structulize and properties of the dataset for correction

Visual assessment

In [8]:
```
# Image prediction
image_prediction.sample(10)
```

Out[8]:

	tweet_id	jpg_url	img_num	
1079	717841801130979328	https://pbs.twimg.com/media/CfZJTphWAAAl5Ys.jpg	1	
992	708109389455101952	https://pbs.twimg.com/media/CdO1u9vWAAApj2V.jpg	1	S
1671	813096984823349248	https://pbs.twimg.com/media/C0izZULWgAAKD-F.jpg	1	
738	687124485711986689	https://pbs.twimg.com/media/CYkoE10WEAAWqxm.jpg	1	
818	692901601640583168	https://pbs.twimg.com/media/CZ2uU37UcAANzmK.jpg	1	c
352	672538107540070400	https://pbs.twimg.com/media/CVVV1wJWoAEcOyk.jpg	1	
1011	709449600415961088	https://pbs.twimg.com/media/Cdh4pgAW0AEKJ_a.jpg	2	
988	707776935007539200	https://pbs.twimg.com/media/CdKHWimWoAABs08.jpg	1	
534	676936541936185344	https://pbs.twimg.com/media/CWT2MUgWIAECWig.jpg	1	Ches
1680	813800681631023104	https://pbs.twimg.com/media/C0szZh_XUAAm9je.jpg	1	

In [9]:
```
# twitter archive
twitter_archive.sample(10)
```

Out[9]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
643	793195938047070209	NaN	NaN	2016-10-31 21:00:23 +0000	href="http://tw
2153	669661792646373376	NaN	NaN	2015-11-25 23:39:47 +0000	href="http://tw
1828	676263575653122048	NaN	NaN	2015-12-14 04:52:55 +0000	href="http://tw
1602	685973236358713344	NaN	NaN	2016-01-09 23:55:38 +0000	href="http://tw
405	823939628516474880	NaN	NaN	2017-01-24 17:04:50 +0000	href="http://tw
527	808733504066486276	NaN	NaN	2016-12-13 18:01:07 +0000	href="http://tw
1714	680440374763077632	NaN	NaN	2015-12-25 17:30:01 +0000	href="http://tw
1180	719339463458033665	NaN	NaN	2016-04-11 01:41:07 +0000	href="http://tw
1555	688894073864884227	NaN	NaN	2016-01-18 01:22:00 +0000	href="http://tw
466	817171292965273600	NaN	NaN	2017-01-06 00:49:53 +0000	href="http://tw

In [10]:
```
# retweet favorite
retweet_favorite.sample(20)
```

Out[10]:

	ID	retweet	favorites
1793	676593408224403456	2081	4411
504	809920764300447744	3972	15680
1363	700505138482569216	557	2256
742	778286810187399168	3300	10398
2085	670444955656130560	1875	6396
2270	667119796878725120	122	313
1942	673342308415348736	541	1211
432	819006400881917954	19013	45462
691	785170936622350336	4868	12116
1575	686034024800862208	1125	3039
1860	674793399141146624	1041	2400
2027	671486386088865792	183	549
1643	682662431982772225	1058	3009
214	849412302885593088	3055	15501
1443	694329668942569472	494	1965
441	818145370475810820	2576	12326
1439	694669722378485760	14110	35271
469	814578408554463233	5879	0
302	834931633769889797	1621	10729
1921	673705679337693185	390	1206

Programmatic assessment

In [11]:
```
1 image_prediction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   int64  
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [12]:
```
1 image_prediction.describe()
```

Out[12]:

	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 [13]:
```
1 twitter_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [14]:
```
1 twitter_archive.describe()
```

Out[14]:

	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 [15]:
```
1 retweet_favorite.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2331 entries, 0 to 2330
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   ID          2331 non-null   object
 1   retweet     2331 non-null   int64 
 2   favorites   2331 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 54.8+ KB
```

In [16]:
```
1 retweet_favorite.describe()
```

Out[16]:

	retweet	favorites
count	2331.000000	2331.000000
mean	2665.983698	7478.497211
std	4508.528533	11610.380504
min	1.000000	0.000000
25%	540.500000	1300.500000
50%	1246.000000	3249.000000
75%	3099.000000	9156.000000
max	76636.000000	154350.000000

In [17]:
```
1 twitter_archive.sample(20)
```

Out[17]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
1009	747512671126323200	NaN	NaN	2016-06-27 19:31:23 +0000	href="http://tw
360	829861396166877184	NaN	NaN	2017-02-10 01:15:49 +0000	href="http://tw
26	886983233522544640	NaN	NaN	2017-07-17 16:17:36 +0000	href="http://tw
1954	673656262056419329	NaN	NaN	2015-12-07 00:12:23 +0000	href="http://tw

In [18]:
```
# Checking the names under name column
twitter_archive.name.sort_values()
```

Out[18]:
```
1035     Abby
1021     Abby
938       Ace
1933     Acro
1327    Adele
        ...
1031     very
773      very
1097     very
819      very
1385     very
Name: name, Length: 2356, dtype: object
```

In [19]:
```
# lowercase under name column
twitter_archive.loc[(twitter_archive['name'].str.islower())]
```

Out[19]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
22	887517139158093824	NaN	NaN	2017-07-19 03:39:09 +0000	href="http://tw
56	881536004380872706	NaN	NaN	2017-07-02 15:32:16 +0000	href="http://tw
118	869988702071779329	NaN	NaN	2017-05-31 18:47:24 +0000	href="http://tw
				2017-05-02	

In [20]:
```
# Inconsistent rating numbers
twitter_archive[(twitter_archive['rating_numerator'] > 20) & (twitt
```

Out[20]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
433	820690176645140481	NaN	NaN	2017-01-15 17:52:40 +0000	href="http://tw
902	758467244762497024	NaN	NaN	2016-07-28 01:00:57 +0000	href="http://tw
1120	731156023742988288	NaN	NaN	2016-05-13 16:15:54 +0000	href="http://tw
1202	716439118184652801	NaN	NaN	2016-04-03 01:36:11 +0000	href="http://tw
1228	713900603437621249	NaN	NaN	2016-03-27 01:29:02 +0000	href="http://tw
1254	710658690886586372	NaN	NaN	2016-03-18 02:46:49 +0000	href="http://tw
1274	709198395643068416	NaN	NaN	2016-03-14 02:04:08 +0000	href="http://tw
1351	704054845121142784	NaN	NaN	2016-02-28 21:25:30 +0000	href="http://tw
1433	697463031882764288	NaN	NaN	2016-02-10 16:51:59 +0000	href="http://tw
1634	684225744407494656	6.842229e+17	4.196984e+09	2016-01-05 04:11:44 +0000	href="http://tw
1635	684222868335505415	NaN	NaN	2016-01-05 04:00:18 +0000	href="http://tw
1779	677716515794329600	NaN	NaN	2015-12-18 05:06:23 +0000	href="http://tw

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
1843	675853064436391936	NaN	NaN	2015-12-13 01:41:41 +0000	href="http://tw

Issues

Tidiness

- Float in reply status_id (archive)
- Float in reply user_id (archive)
- Float in retweeted_status_user_id (archive)
- Float in retweeted_status_id (archive)
- Timestamp column in string (archive)
- Retweet stat timestamp in string (archvie)
- floofer, doggo, puppo, pupper columns belong in one column (archve)
- Tweet_image and retweet and favorites share dataframe (favorites

Quality

- Upper and lower in p1, p2, and p3 columns (image)
 - column name for p1,p2 not clear (image)
- Retweet status_id in decimal (archive)
- Retweet status_user_id in decimal (archive)
- Rating numerator max is with index 979 (archive)
- Missing data in in_reply_to_status_id and in_reply_to_user_id (achive)
- Missing data in retweet status id, user_id, timestamp (archive)
- Missing data in expanded urls (archive)
- Under name column, names lowercase is not actual name (archive)
- Rating denominator and numerator are inconsistent (archive)
- Source code difficult to comprehend

In [21]:
```
# Make copy before cleaning
clean_archive = twitter_archive.copy()
clean_image = image_prediction.copy()
clean_favorites = retweet_favorite.copy()
```

Cleaning

Tidiness

```
#### Define:
        The missing data are at large in these columns,
in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id,
retweeted_status_user_id, and retweeted_status_timestamp. Therefor
we will drop these columns since we dont need them to complete our
analysis because our focus of this project is dog's breed rating.
```

Code

In [22]:
```
clean_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [23]:
```
# Dropping the columns
clean_archive.drop(columns=['in_reply_to_status_id',
                            'in_reply_to_user_id',
                            'retweeted_status_id',
                            'retweeted_status_user_id',
                           'retweeted_status_timestamp'], inplace=
```

In [24]:
```
# Dropping the missing rows with missing data in expanded_urls co
clean_archive.dropna(inplace=True)
```

Test

In [25]:
```
clean_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2297 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   tweet_id            2297 non-null   int64 
 1   timestamp           2297 non-null   object
 2   source              2297 non-null   object
 3   text                2297 non-null   object
 4   expanded_urls       2297 non-null   object
 5   rating_numerator    2297 non-null   int64 
 6   rating_denominator  2297 non-null   int64 
 7   name                2297 non-null   object
 8   doggo               2297 non-null   object
 9   floofer             2297 non-null   object
 10  pupper              2297 non-null   object
 11  puppo               2297 non-null   object
dtypes: int64(3), object(9)
memory usage: 233.3+ KB
```

Define

Timestamp data type is in string. We will need to convert the data type to timestamp.

Code

In [26]:
```
clean_archive.timestamp = pd.to_datetime(clean_archive.timestamp)
```

In [27]:
```
clean_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2297 entries, 0 to 2355
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2297 non-null   int64              
 1   timestamp           2297 non-null   datetime64[ns, UTC]
 2   source              2297 non-null   object             
 3   text                2297 non-null   object             
 4   expanded_urls       2297 non-null   object             
 5   rating_numerator    2297 non-null   int64              
 6   rating_denominator  2297 non-null   int64              
 7   name                2297 non-null   object             
 8   doggo               2297 non-null   object             
 9   floofer             2297 non-null   object             
 10  pupper              2297 non-null   object             
 11  puppo               2297 non-null   object             
dtypes: datetime64[ns, UTC](1), int64(3), object(8)
memory usage: 233.3+ KB
```

Define

The dog's stage columns need to organize to under one single colum by using melt function. Also we willing filling out the None value inside the column.

Code

In [28]:
```
dog = ['doggo','pupper', 'floofer', 'puppo' ]
for stage in dog:
        clean_archive[stage] = clean_archive[stage].replace('None'
```

In [29]:
```
clean_archive = pd.melt(clean_archive, id_vars=['tweet_id', 'timestam
                                    'rating_numerator', 'rating_de
                                    'name'], var_name ='stage')
clean_archive.drop(columns='value', axis=1, inplace=True)

```

Test

In [30]:
```
clean_archive.sample(5)
```

Out[30]:

	tweet_id	timestamp	source	text
7774	757354760399941633	2016-07-24 23:20:20+00:00	<a href="http://twitter.com/download/iphone" r...	This is Devón (pronounced "Eric"). He forgot h...
7312	819238181065359361	2017-01-11 17:42:57+00:00	<a href="http://twitter.com" rel="nofollow">Tw...	Some happy pupper news to share. 10/10 for eve...
5585	746369468511756288	2016-06-24 15:48:42+00:00	<a href="http://twitter.com/download/iphone" r...	This is an Iraqi Speed Kangaroo. It is not a d...
7263	825876512159186944	2017-01-30 01:21:19+00:00	<a href="http://twitter.com/download/iphone" r...	This is Mo. No one will push him around in the...
6063	691416866452082688	2016-01-25 00:26:41+00:00	<a href="http://twitter.com/download/iphone" r...	I present to you... Dog Jesus. 13/10 (he could...

Define

Tidiness and quality cleaning

Combinding the clean_archive, clean_favorites, and clean_image dat frame all together by using merge.

In [31]:
```
# Changing the name of Id column to tweet_id
# Changing the Datatype for the ID column
clean_image.rename(columns={'ID': 'tweet_id'}, inplace=True)
clean_image.tweet_id = clean_image.tweet_id.astype(str)
clean_favorites.rename(columns={'ID': 'tweet_id'}, inplace=True)
clean_favorites.tweet_id = clean_favorites.tweet_id.astype(str)
clean_archive.tweet_id = clean_archive.tweet_id.astype(str)
```

In [32]:
```
# Merging all 3 dataframe together fall under tweet_id
clean_tweets = pd.merge(clean_archive, clean_favorites, how='inner
clean_tweets = pd.merge(clean_tweets, clean_image, how='inner', on
```

Test

In [33]:
```
clean_tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8236 entries, 0 to 8235
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype              
---  ------               --------------  -----              
 0   tweet_id             8236 non-null   object             
 1   timestamp            8236 non-null   datetime64[ns, UTC]
 2   source               8236 non-null   object             
 3   text                 8236 non-null   object             
 4   expanded_urls        8236 non-null   object             
 5   rating_numerator     8236 non-null   int64              
 6   rating_denominator   8236 non-null   int64              
 7   name                 8236 non-null   object             
 8   stage                8236 non-null   object             
 9   retweet              8236 non-null   int64              
 10  favorites            8236 non-null   int64              
 11  jpg_url              8236 non-null   object             
 12  img_num              8236 non-null   int64              
 13  p1                   8236 non-null   object             
 14  p1_conf              8236 non-null   float64            
 15  p1_dog               8236 non-null   bool               
 16  p2                   8236 non-null   object             
 17  p2_conf              8236 non-null   float64            
 18  p2_dog               8236 non-null   bool               
 19  p3                   8236 non-null   object             
 20  p3_conf              8236 non-null   float64            
 21  p3_dog               8236 non-null   bool               
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(5), object(
memory usage: 1.3+ MB
```

In [34]:
```
clean_tweets.sample(20)
```

Out[34]:

	tweet_id	timestamp	source	tex
113	886366144734445568	2017-07-15 23:25:31+00:00	<a href="http://twitter.com/download/iphone" r...	This is Roscoe Anothe pupper faller victim t..
3939	717047459982213120	2016-04-04 17:53:31+00:00	<a href="http://twitter.com/download/iphone" r...	This is Flávio (pronounced Baxter). He' a Ben..
2776	761745352076779520	2016-08-06 02:06:59+00:00	<a href="http://twitter.com/download/iphone" r...	Guys.. we only rate dogs. Pls don't send any m..
1087	831309418084069378	2017-02-14 01:09:44+00:00	<a href="http://twitter.com/download/iphone" r...	This is Scooter and his son Montoya Scooter ..
6814	672594978741354496	2015-12-04 01:55:13+00:00	<a href="http://twitter.com/download/iphone" r...	Meet Scott Just trying to catch his train to ..
2961	756288534030475264	2016-07-22 00:43:32+00:00	<a href="http://twitter.com/download/iphone" r...	Here's a heartwarming scene of a single father..
5470	684222868335505415	2016-01-05 04:00:18+00:00	<a href="http://twitter.com/download/iphone" r...	Someone help the girl is being mugged Several..
4446	704819833553219584	2016-03-02 00:05:17+00:00	<a href="http://twitter.com/download/iphone" r...	This is Chesterson He's a Bolivian Scoop Dog...
5666	681523177663676416	2015-12-28 17:12:42+00:00	<a href="http://twitter.com/download/iphone" r...	This is Carly She' actually 2 dogs fused tog..
2093	790987426131050500	2016-10-25 18:44:32+00:00	<a href="http://twitter.com/download/iphone" r...	This is Misty She has a cowboy hat on her nos..
777	845677943972139009	2017-03-25 16:45:08+00:00	<a href="http://twitter.com/download/iphone" r...	C'mon guys Please only send in dogs We only ..

	tweet_id	timestamp	source	tex
6351	675135153782571009	2015-12-11 02:08:58+00:00	<a href="http://twitter.com/download/iphone" r...	This is Steven. He got locked outside Damn it..
1788	802323869084381190	2016-11-26 01:31:31+00:00	<a href="http://twitter.com/download/iphone" r...	This is Severus He's here to fix you cable. ..
6712	673295268553605120	2015-12-06 00:17:55+00:00	<a href="http://twitter.com/download/iphone" r...	Meet Eve She's a raging alcoholic 8/10 (would..
5170	689275259254616065	2016-01-19 02:36:42+00:00	<a href="http://twitter.com/download/iphone" r...	Meet Lucky He was showing his friends an extr..
2206	786363235746385920	2016-10-13 00:29:39+00:00	<a href="http://twitter.com/download/iphone" r...	This is Rizzo He has many talents. A true ren..
3021	753294487569522689	2016-07-13 18:26:16+00:00	<a href="http://twitter.com/download/iphone" r...	This is Ace He's a window washer. One of the ..
1380	819227688460238848	2017-01-11 17:01:16+00:00	<a href="http://twitter.com/download/iphone" r...	This is Finn He's wondering if you come here ..
5137	689659372465688576	2016-01-20 04:03:02+00:00	<a href="http://twitter.com/download/iphone" r...	This is Ricky He's being escorted out of the ..
2907	757741869644341248	2016-07-26 00:58:34+00:00	<a href="http://twitter.com/download/iphone" r...	This is Leonard. He hides in bushes to escape ..

20 rows × 22 columns

Quality Cleaning

Define

Convert all letters in p1,p2,p3 columns into capitalize dog's name for consistency. Also, will change the column name for readability.

Code

In [35]:
```
# Changing columns names for p1, p2, and p3 including the feature
clean_tweets.rename(columns={'p1': 'prediction_1', 'p2':'predictio
                             'p1_conf': 'confidence_prediction_1',
                             'p1_dog': 'dog_1', 'p2_dog': 'dog_2',
                             'favorites ' : 'favorites'}, inplace=T
```

In [36]:
```
# Run a for loop into prediction columns to apply capitalize the r
mask = ['prediction_1','prediction_2', 'prediction_3']
for pred in mask:
        clean_tweets[pred] = clean_tweets[pred].str.capitalize()
```

Test

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8236 entries, 0 to 8235
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   tweet_id                8236 non-null   object
 1   timestamp               8236 non-null   datetime64[ns, UTC]
 2   source                  8236 non-null   object
 3   text                    8236 non-null   object
 4   expanded_urls           8236 non-null   object
 5   rating_numerator        8236 non-null   int64
 6   rating_denominator      8236 non-null   int64
 7   name                    8236 non-null   object
 8   stage                   8236 non-null   object
 9   retweet                 8236 non-null   int64
 10  favorites               8236 non-null   int64
 11  jpg_url                 8236 non-null   object
 12  img_num                 8236 non-null   int64
 13  prediction_1            8236 non-null   object
```

Define

Inconsistence rating scale of numerator and denominator, will trac down and removes these inconsistency.

Code

In [38]:
```
# Removes all rows that over 20 rating point scale both in denomin
clean_tweets.drop(clean_tweets.loc[(clean_tweets['rating_numerator
.index, inplace=True)
```

Test

In [39]:
```
clean_tweets.query("rating_numerator >= 20 and rating_denominator
```

Out[39]:

tweet_id	timestamp	source	text	expanded_urls	rating_numerator	rating_denominator	na

0 rows × 22 columns

Define

Replace out all lowercase names under name column. The names that re lowercase under name column are not improper name or human entr errors.

code

In [40]:
```
# Create a mask which contain the rows with lowercase names
# Convert the list of name into a list and use the name to interat
mask = clean_tweets.loc[(clean_tweets['name'].str.islower())]
mask = list(mask['name'])
for entry in mask:
    clean_tweets['name'].replace(entry, 'None', inplace=True)
```

Test

In [41]:
```
clean_tweets.loc[(clean_tweets['name'].str.islower())]
```

Out[41]:

tweet_id	timestamp	source	text	expanded_urls	rating_numerator	rating_denominator	na

0 rows × 22 columns

Define

Use option.display to change format display for to 2 decimal for a

Code

In [42]:
```
# Set default to all float number display 2 decimal using option.d
pd.options.display.float_format = "{:,.2f}".format
clean_tweets[['confidence_prediction_2', 'confidence_prediction_3'
# Change jpg_url and img_num column name
clean_tweets.rename(columns={'jpg_url': 'image_url', 'img_num':'im
```

test

In [43]:
```
# check random sample and expand display for columns and width
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
clean_tweets.sample(10)
```

Out[43]:

	tweet_id	timestamp	source	text
763	846514051647705089	2017-03-28 00:07:32+00:00	<a href="http://twitter.com/download/iphone" r...	This is Barney. He's an elder doggo. Hitches a...
4372	706291001778950144	2016-03-06 01:31:11+00:00	<a href="http://twitter.com/download/iphone" r...	When you're just relaxin and having a swell ti...
8057	666781792255496192	2015-11-18 00:55:42+00:00	<a href="http://twitter.com/download/iphone" r...	This is a purebred Bacardi named Octaviath. Ca...
1504	815390420867969024	2017-01-01 02:53:20+00:00	<a href="http://twitter.com/download/iphone" r...	Happy New Year from the squad! 13/10 for all h...
6166	676440007570247681	2015-12-14 16:34:00+00:00	<a href="http://twitter.com/download/iphone" r...	Hope your Monday isn't too awful. Here's two b...
7310	670411370698022913	2015-11-28 01:18:21+00:00	<a href="http://twitter.com/download/iphone" r...	Meet Scooter. He's ready for his first day of ...
6120	676776431406465024	2015-12-15 14:50:49+00:00	<a href="http://twitter.com/download/iphone" r...	When someone yells "cops!" at a party and you ...
3923	717841801130979328	2016-04-06 22:29:56+00:00	<a href="http://twitter.com/download/iphone" r...	This is Barclay. His father was a banana. 11/1...
5525	683498322573824003	2016-01-03 04:01:13+00:00	<a href="http://twitter.com/download/iphone" r...	This is Griffin. He's desperate for both a phy...
7876	667534815156183040	2015-11-20 02:47:56+00:00	<a href="http://twitter.com" rel="nofollow">Tw...	This is Frank (pronounced "Fronq"). Too many b...

In [44]:
```
clean_tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8184 entries, 0 to 8235
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   tweet_id                  8184 non-null   object
 1   timestamp                 8184 non-null   datetime64[ns, UTC]
 2   source                    8184 non-null   object
 3   text                      8184 non-null   object
 4   expanded_urls             8184 non-null   object
 5   rating_numerator          8184 non-null   int64
 6   rating_denominator        8184 non-null   int64
 7   name                      8184 non-null   object
 8   stage                     8184 non-null   object
 9   retweet                   8184 non-null   int64
 10  favorites                 8184 non-null   int64
 11  image_url                 8184 non-null   object
 12  image_number              8184 non-null   int64
 13  prediction_1              8184 non-null   object
 14  confidence_prediction_1   8184 non-null   float64
 15  dog_1                     8184 non-null   bool
 16  prediction_2              8184 non-null   object
 17  confidence_prediction_2   8184 non-null   float64
 18  dog_2                     8184 non-null   bool
 19  prediction_3              8184 non-null   object
 20  confidence_prediction_3   8184 non-null   float64
 21  dog_3                     8184 non-null   bool
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(5), object(
memory usage: 1.3+ MB
```

Define

Drop duplicated rows using duplicated method and keep the first oc
urence

Code

In [45]:
```
# Dropping duplicated rows using duplicated method
clean_tweets = clean_tweets[~clean_tweets.tweet_id.duplicated(keep
```

Test

In [46]:
```
# Test if any duplicated rows in dataframe, If True, there is dup
clean_tweets.duplicated().value_counts()
```

Out[46]:
```
False    2046
dtype: int64
```

Define

Replace values in the source columns to which ever devices the twe
ter's user sent out.

Code

In [47]:
```
place to transcript sources code into more reader friendly
eets['source'] =  clean_tweets['source'].str.replace('<a href="http://
eets['source'] = clean_tweets['source'].str.replace('<a href="http://tw
eets['source'] = clean_tweets['source'].str.replace('<a href="https://a
```

Test

In [48]:
```
# Use value_counts to test for values under source column
clean_tweets.source.value_counts()
```

Out[48]:
```
Twitter for iphone    2006
Twitter Web Client      30
TweetDeck               10
Name: source, dtype: int64
```

Store

In [49]:
```
# Save clean data to a master csv
clean_tweets.to_csv('twitter_archive_master.csv')
```

Analysis

- Numbers of correct prediction for dog breed?
- Distribution of the dog stages throughout DataFrame
- Test for correlation between retweet_count and favorites_count c
lumns
- Analysis on the classification of dog's breed result

In [50]:
```
tweet_analysis = pd.read_csv('twitter_archive_master.csv')
```

In [51]:
```
1 tweet_analysis.sample(10)
```

			03:00:47+00:00		iphone	own feet. 12/10 w...	
206	824	842535590457499648	2017-03-17 00:38:32+00:00	Twitter for iphone	This is Winnie. She lost her body saving a chi...	https://twitter.com/dc
18	72	888554962724278272	2017-07-22 00:23:06+00:00	Twitter for iphone	This is Ralphus. He's powering up. Attempting ...	https://twitter.com/dc
1516	6112	676819651066732545	2015-12-15 17:42:34+00:00	Twitter for iphone	Watch out Airbud. This pupper is also good	https://twitter.com/dc

1. Numbers of correct prediction for dogs breed.

In [52]:
```
1 # Create a new DataFrame which contained that columns tracksm cor
2 correct_breed = tweet_analysis[['dog_1','dog_2','dog_3']].apply(pc
3 correct_breed
```

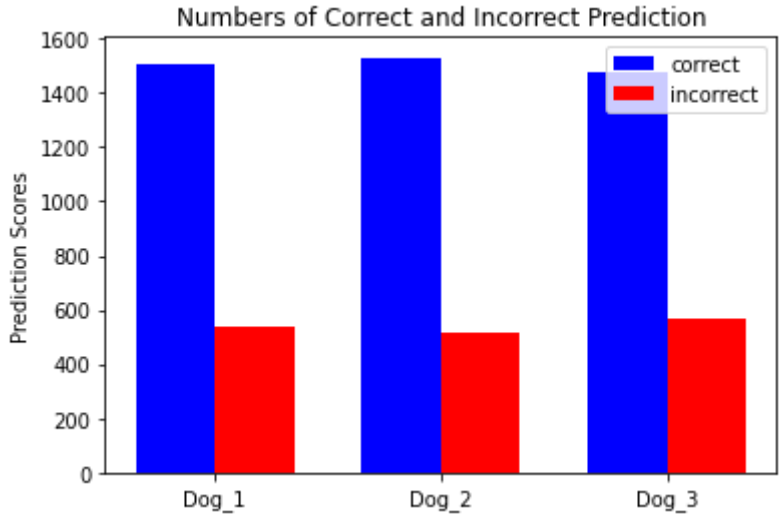
Out[52]:

	dog_1	dog_2	dog_3
True	1509	1529	1476
False	537	517	570

In [53]:
```
width = 0.35
labels = ['Dog_1', 'Dog_2', 'Dog_3']
correct = correct_breed.iloc[0].tolist()
incorrect = correct_breed.iloc[1].tolist()
x = np.arange(len(labels))

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, correct, width, label='correct',colo
rects2 = ax.bar(x + width/2, incorrect, width, label='incorrect',

ax.set_ylabel('Prediction Scores')
ax.set_title('Numbers of Correct and Incorrect Prediction')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend();
```



In [54]:
```
# Sum up the total number of correct and incorrect prediction and
correct_breed['total'] = correct_breed.sum(axis=1)
correct_breed
```

Out[54]:

	dog_1	dog_2	dog_3	total
True	1509	1529	1476	4514
False	537	517	570	1624

In [55]:
```
# Percentage of correction
correct_breed.total[1]/ sum(correct_breed.total)
```

Out[55]: 0.7354187031606386

The percentage of total correct prediction is 0.735, which is bases on the total amount of `Tru` under all 3 dog columns divided by the combined numbers of `True` and `False`. This mean there is a 3/4 chances of the image is correctly predicted by the Neural network.

2. Which dog breed appeared the most according the prediction?

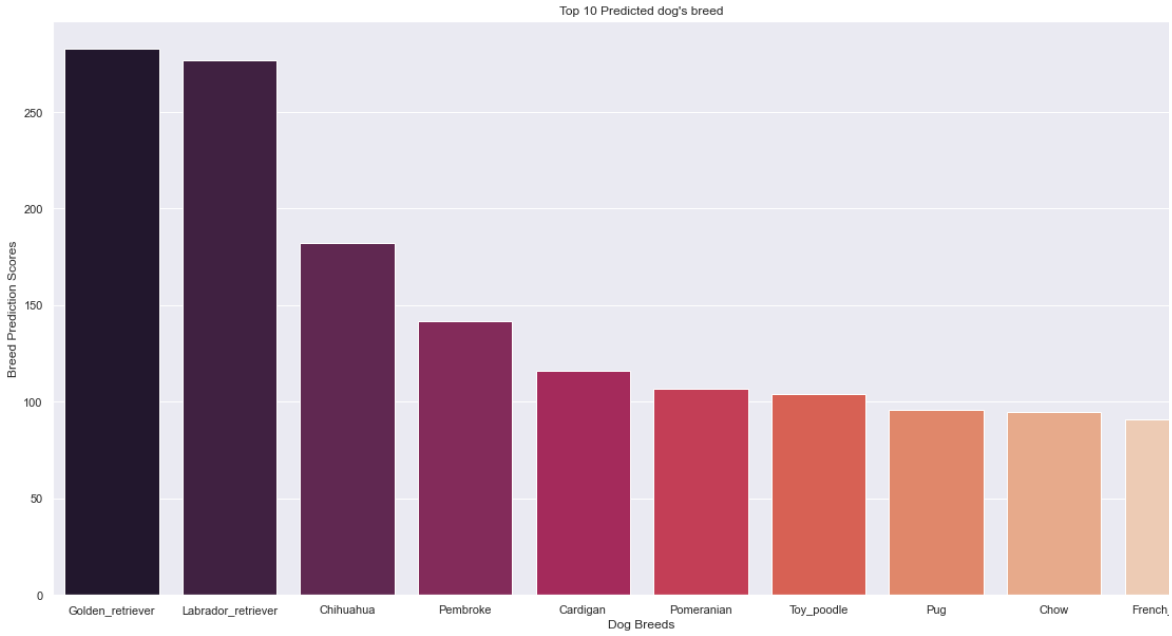
In [91]:
```
# Extract and combined the prediction columns into a new DataFrame
breed = tweet_analysis[['prediction_1', 'prediction_2', 'predictio
breed['total'] = breed.sum(axis=1)
breed.sort_values('total', ascending=False, inplace=True)
breed = breed.head(10)
breed
```

Out[91]:

	prediction_1	prediction_2	prediction_3	total
Golden_retriever	148.00	88.00	47.00	283.00
Labrador_retriever	94.00	104.00	79.00	277.00
Chihuahua	81.00	43.00	58.00	182.00
Pembroke	88.00	27.00	27.00	142.00
Cardigan	19.00	74.00	23.00	116.00
Pomeranian	38.00	41.00	28.00	107.00
Toy_poodle	38.00	37.00	29.00	104.00
Pug	57.00	17.00	22.00	96.00
Chow	44.00	20.00	31.00	95.00
French_bulldog	25.00	40.00	26.00	91.00

In [57]:
```
# Visualize bar chart top 10 predicted dog breed
x = breed.index.tolist()
y = breed.total.tolist()
sns.set(style='darkgrid')
plt.figure(figsize=(20,10));
sns.barplot(x=x, y=y, palette='rocket');
plt.xlabel('Dog Breeds')
plt.ylabel('Breed Prediction Scores')
plt.title("Top 10 Predicted dog's breed")
```

Out[57]:
```
Text(0.5, 1.0, "Top 10 Predicted dog's breed")
```



Summary

According to the prediction, the most predicted type of dog breed is Golden retriever. This ma sense because Golden retrivever is very high energy and friendly type of breed, they are easy train and very well behave, and their most distintive trait is their golden long fur coat.

In [58]:
```
1 tweet_analysis.head()
```

Out[58]:

	Unnamed: 0	tweet_id	timestamp	source	text	
0	0	892420643555336193	2017-08-01 16:23:56+00:00	Twitter for iphone	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_
1	4	892177421306343426	2017-08-01 00:17:27+00:00	Twitter for iphone	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_
2	8	891815181378084864	2017-07-31 00:18:03+00:00	Twitter for iphone	This is Archie. He is a rare Norwegian Pouncin...	https://twitter.com/dog_
3	12	891689557279858688	2017-07-30 15:58:51+00:00	Twitter for iphone	This is Darla. She commenced a snooze mid meal...	https://twitter.com/dog_
4	16	891327558926688256	2017-07-29 16:00:24+00:00	Twitter for iphone	This is Franklin. He would like you to stop ca...	https://twitter.com/dog_

3. What are the relationship between retweet and favorites?

In [59]:
```
1 tweet_analysis['favorites'].describe()
```

Out[59]:
```
count      2,046.00
mean       7,924.41
std       11,962.84
min            0.00
25%        1,495.00
50%        3,490.50
75%        9,901.50
max      154,350.00
Name: favorites, dtype: float64
```

In [60]:
```
1 tweet_analysis['retweet'].describe()
```

Out[60]:
```
count    2,046.00
mean     2,558.13
std      4,464.57
min         11.00
25%        545.00
50%      1,206.00
75%      2,943.00
max     76,636.00
Name: retweet, dtype: float64
```

In [61]:
```
1 data = tweet_analysis[['retweet', 'favorites']]
2 correlation = data.corr(method='pearson')
3 correlation
```

Out[61]:

	retweet	favorites
retweet	1.00	0.86
favorites	0.86	1.00

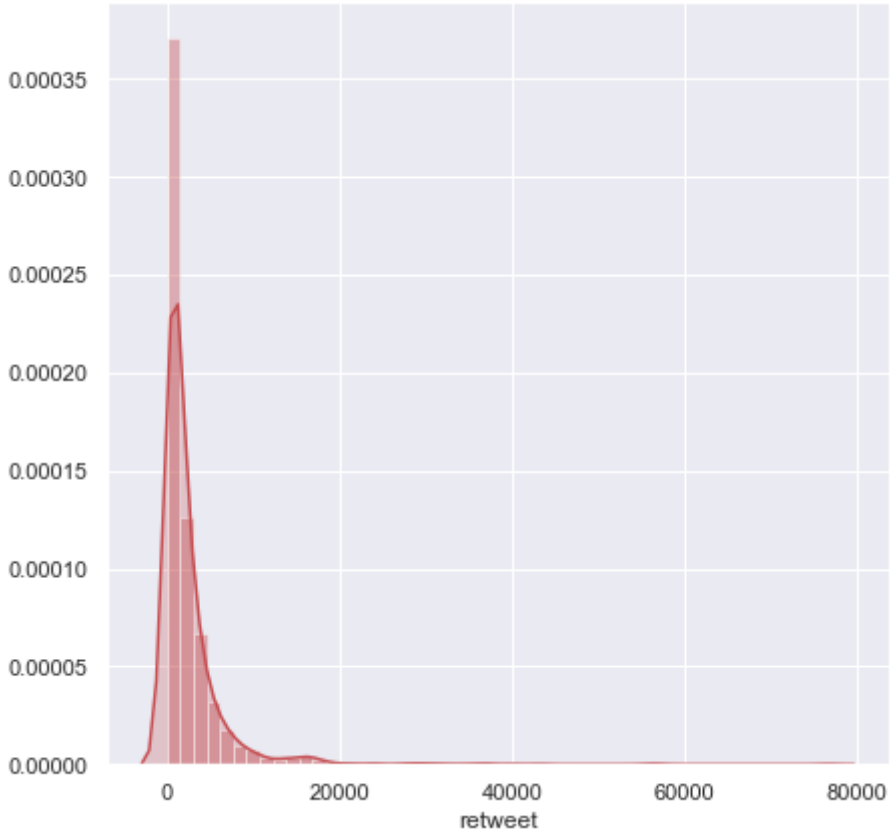
In [62]:
```
1 tweet_analysis.query("favorites == 0")
```

Out[62]:

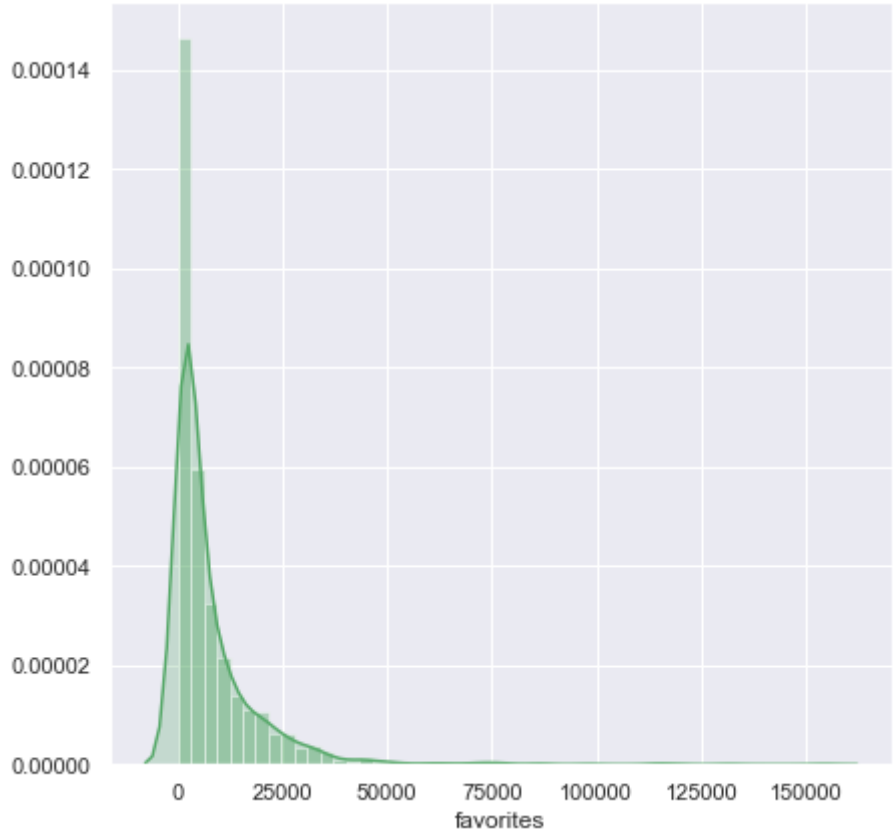
anded_urls	rating_numerator	rating_denominator	name	stage	retweet	favorites	
30583320...	13	10	Lilly	doggo	16643	0	https://pbs.t
87685077...	14	10	None	doggo	72	0	https://pbs.t
tus/86501...	13	10	None	doggo	110	0	https://pbs.
60914485...	13	10	None	doggo	744	0	https://pbs.
/8482893...	10	10	None	doggo	19	0	https://pbs.t
...	...	...	...	...	...	...	
79158373...	11	10	Rubio	doggo	7833	0	https://pbs.t
75354435...	13	10	None	doggo	16096	0	https://pb
99827977...	12	10	None	doggo	125	0	https://pbs.t
tatus/667...	12	10	None	doggo	31	0	https://pbs.tv
tatus/667...	5	10	None	doggo	31	0	https://pbs.t

The correlation coefficient between retweet and favorites is 0.86. And there are 72 rows with z favorites with high number of share. It's look like the favorites or like button was disabled.

In [63]:
```
1 plt.subplots( figsize=(7, 7), sharex=True)
2 sns.distplot(tweet_analysis.retweet, kde_kws={"shade":True}, colo
```



In [64]:
```
plt.subplots( figsize=(7, 7), sharex=True)
sns.distplot(tweet_analysis.favorites, kde_kws={"shade":True}, col
```



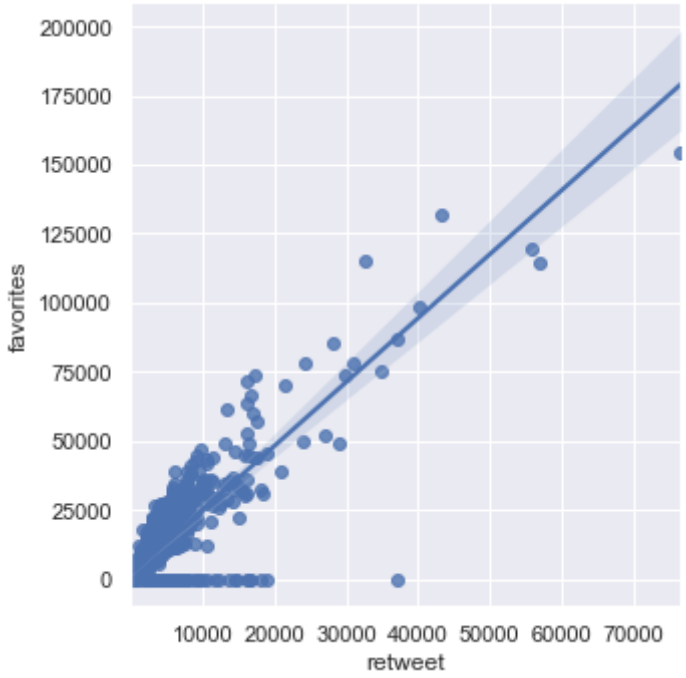
Both of the graph above are skewed to the right, which mean both show relation with each ot

In [92]:
```
tweet_analysis.head()
```

Out[92]:

	Unnamed: 0	tweet_id	timestamp	source	text	
0	0	892420643555336193	2017-08-01 16:23:56+00:00	Twitter for iphone	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_
1	4	892177421306343426	2017-08-01 00:17:27+00:00	Twitter for iphone	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_
2	8	891815181378084864	2017-07-31 00:18:03+00:00	Twitter for iphone	This is Archie. He is a rare Norwegian Pouncin...	https://twitter.com/dog_

In [101]:
```
# Scatter plot and regression line of favorites and retweet
import seaborn as sns
sns.lmplot(x='retweet',y='favorites',data=tweet_analysis,fit_reg=T
```



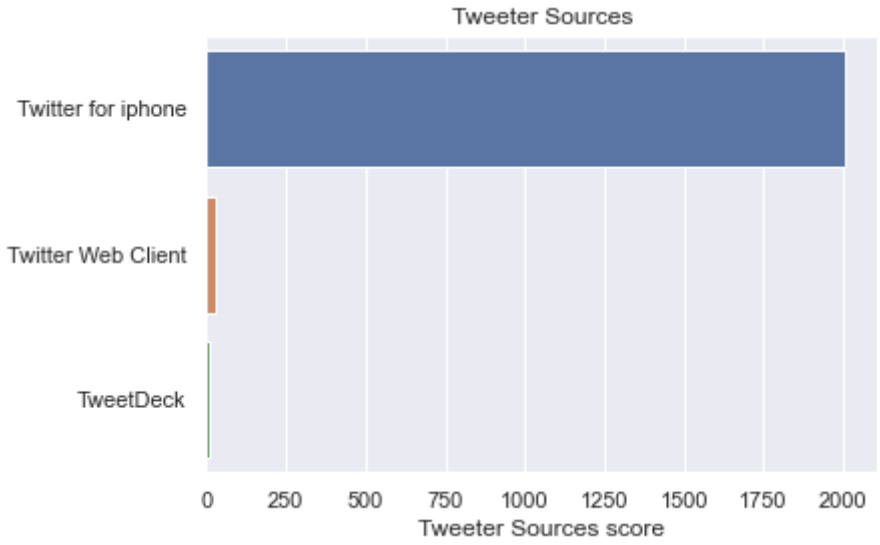
4. Which sources was used the most for tweeting?

In [65]:
```
# Using Value_counts to find values under source column and convei
sources = pd.DataFrame(tweet_analysis.source.value_counts())
sources
```

Out[65]:

	source
Twitter for iphone	2006
Twitter Web Client	30
TweetDeck	10

In [66]:
```
# plotting the sources score using barplot
sns.barplot(data=sources, x=sources['source'], y=sources.index);
plt.xlabel('Tweeter Sources score');
plt.title('Tweeter Sources');
```



Base on the barplot above, the most popular sources to sent tweets from is by using twitter ip
app. This make sense because mobile phone is much more convienece for usage to check ne

Conclusion

Base on the dataset, after cleaning, wrangling, analyzing, and visualizing. The important keys
away from this dataset are golden retreiver is either very popular dog breed or more accuratel
predict by the neural network. The image prediction rates is approximately 73% correct. And
with high favorites rating is also high have high number of retweets.

In [ ]:
```
1
```