wrangle_act

October 12, 2018

1 Project: "We Rate Dogs" Data Analysis

1.1 Table of Contents

Introduction

Twitter account 'WeRateDogs (@dog_rates)" is famous for its unique tweets which evaluate dog pictures which are posted from readers. In this analysis, we get the tweets data from Twitter API and the additional data about pictures of those tweets and assess them. Specifically, we focus on 2 topics below

- How the ratings are related with stages and types of dogs.
- Relationship between favorite counts and the ratings.

Data Wrangling

1.2 Gather

```
In [188]: #import packages

          import pandas as pd
          import numpy as np
          import requests
          import tweepy
          import json
          import timeit
          import datetime
          import matplotlib.pyplot as plt
          import seaborn as sns
          import statsmodels.api as sm
          %matplotlib inline
```

1.2.1 Aquire Twitter archieve data

```
In [189]: # read "twitter-archive-enhanced.csv" file
          df_archive = pd.read_csv("twitter-archive-enhanced.csv")
          df_archive.columns

Out[189]: Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
                 'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
                 'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
                 'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
                dtype='object')
```

1.2.2 Acquire image-predictions data

```
In [190]: # download image prediction tsv file as "image_predictions.tsv" in the local directo
          url = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predi
          response = requests.get(url)

          with open("image_predictions.tsv", mode="wb") as file:
              file.write(response.content)

In [191]: # import the file into this notebook as a dataframe
          df_image = pd.read_csv("image_predictions.tsv", sep="\t")
          df_image.columns

Out[191]: Index(['tweet_id', 'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2',
                 'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
                dtype='object')
```

1.2.3 Acquire tweet json-file (from TwitterAPI)

```
In [5]: # get api using tweepy
        consumer_key = "###"
        consumer_secret = "###"
        access_token = "###"
        access_token_secret = "###"

        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_token_secret)
        api = tweepy.API(auth)

In [11]: # select tweets we need in api json and get "retweet_count"  and "favorite_count" inf
         # store them in the list"data_list"
         start = timeit.timeit()

         data_list = []
         for tweet_id in df_archive["tweet_id"]:
             try:
                 api_status = api.get_status(tweet_id, tweet_mode='extended', wait_on_rate_limi
```

```
                tweet_data = {}
                tweet_data["tweet_id"] = api_status._json["id"]
                tweet_data["ret"] = api_status._json["retweet_count"]
                tweet_data["fav"] = api_status._json["favorite_count"]
                data_list.append(tweet_data)
            except Exception as e:
                print(str(tweet_id) + " : " + str(e))

        end = timeit.timeit()
        print(end - start)
```

```
888202515573088257 : [{'code': 144, 'message': 'No status found with that ID.'}]
873697596434513921 : [{'code': 144, 'message': 'No status found with that ID.'}]
872668790621863937 : [{'code': 144, 'message': 'No status found with that ID.'}]
869988702071779329 : [{'code': 144, 'message': 'No status found with that ID.'}]
866816280283807744 : [{'code': 144, 'message': 'No status found with that ID.'}]


Rate limit reached. Sleeping for: 648


861769973181624320 : [{'code': 144, 'message': 'No status found with that ID.'}]
845459076796616705 : [{'code': 144, 'message': 'No status found with that ID.'}]
842892208864923648 : [{'code': 144, 'message': 'No status found with that ID.'}]
837012587749474308 : [{'code': 144, 'message': 'No status found with that ID.'}]
827228250799742977 : [{'code': 144, 'message': 'No status found with that ID.'}]
802247111496568832 : [{'code': 144, 'message': 'No status found with that ID.'}]
775096608509886464 : [{'code': 144, 'message': 'No status found with that ID.'}]
771004394259247104 : [{'code': 179, 'message': 'Sorry, you are not authorized to see this statu
770743923962707968 : [{'code': 144, 'message': 'No status found with that ID.'}]
754011816964026368 : [{'code': 144, 'message': 'No status found with that ID.'}]


Rate limit reached. Sleeping for: 647
Rate limit reached. Sleeping for: 563


0.002290688003995456
```

```
In [59]: #  write the list in 'tweet_json.txt' (each element of tweet on each line in the file
         with open('tweet_json.txt', 'w') as file:
             for line in data_list:
                 json.dump(line, file)
                 file.write("\n")
```

```
In [192]: # load 'tweet_json.txt' and store it in the list "json_list" (each element of this l
          # this list should be the same as the list"data_list"
          json_list = []
```

```
with open('tweet_json.txt','r') as file:
    for line in file:
        json_list.append(json.loads(line))
```

In [193]:
```
# construct dataframe from the list "json_list" and sort columns
df_api = pd.DataFrame(json_list)
df_api = df_api.iloc[:,[1,0,2]]
df_api.columns
```

Out[193]:
```
Index(['ret', 'fav', 'tweet_id'], dtype='object')
```

1.3 Assess

In [194]:
```
# visual assessment
df_archive.head(5)
```

Out[194]:
```
             tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
0  892420643555336193                    NaN                  NaN
1  892177421306343426                    NaN                  NaN
2  891815181378084864                    NaN                  NaN
3  891689557279858688                    NaN                  NaN
4  891327558926688256                    NaN                  NaN

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

                                                                           sourc
0  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a
1  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a
2  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a
3  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a
4  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a


0                                         This is Phineas. He's a mystical boy. Onl
1  This is Tilly. She's just checking pup on you. Hopes you're doing ok. If not, she
2      This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall gras
3                                                    This is Darla. She commenced a snoo
4  This is Franklin. He would like you to stop calling him "cute." He is a very fier

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

```
4                  NaN                       NaN                         NaN


0                                                                  https://twitter.co
1                                                                  https://twitter.co
2                                                                  https://twitter.co
3                                                                  https://twitter.co
4  https://twitter.com/dog_rates/status/891327558926688256/photo/1,https://twitter.co

   rating_numerator  rating_denominator      name doggo floofer pupper puppo
0                13                  10   Phineas  None    None   None  None
1                13                  10     Tilly  None    None   None  None
2                12                  10    Archie  None    None   None  None
3                13                  10     Darla  None    None   None  None
4                12                  10  Franklin  None    None   None  None
```

In [195]:
```
df_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [196]:
```
# visual assessment
df_api.sample(5)
```

Out[196]:

	ret	fav	tweet_id
366	184	2336	828361771580813312
459	5205	0	816829038950027264
2072	246	752	670797304698376195

```
905   3941  11093  756526248105566208
2015  3624   7237  671789708968640512
```

In [197]: df_api.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2341 entries, 0 to 2340
Data columns (total 3 columns):
ret         2341 non-null int64
fav         2341 non-null int64
tweet_id    2341 non-null int64
dtypes: int64(3)
memory usage: 54.9 KB
```

In [198]:
```
# visual assessment
df_image.sample(5)
```

Out[198]:

	tweet_id	jpg_url
1819	834209720923721728	https://pbs.twimg.com/media/C5OlUAaWIAAMBMd.jpg
2024	881666595344535552	https://pbs.twimg.com/media/DDxPFwbWAAEbVVR.jpg
2020	880872448815771648	https://pbs.twimg.com/media/DDl8zzJW0AAisCJ.jpg
472	675145476954566656	https://pbs.twimg.com/media/CV6ZOPqWsAA2OUj.jpg
1320	756288534030475264	https://pbs.twimg.com/media/Cn7gaHrWIAAZJMt.jpg

	img_num	p1	p1_conf	p1_dog	p2
1819	1	golden_retriever	0.754799	True	Pekinese
2024	1	Saluki	0.529012	True	Afghan_hound
2020	1	Pembroke	0.791416	True	Norwich_terrier
472	1	Labrador_retriever	0.458746	True	Great_Dane
1320	3	conch	0.925621	False	French_bulldog

	p2_conf	p2_dog	p3	p3_conf	p3_dog
1819	0.197861	True	Labrador_retriever	0.008654	True
2024	0.250003	True	golden_retriever	0.160739	True
2020	0.061393	True	Chihuahua	0.033726	True
472	0.235504	True	Staffordshire_bullterrier	0.116864	True
1320	0.032492	True	tiger_cat	0.006679	False

In [199]: df_image.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
```

```
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

1.3.1 Quality

df_archive table

- "timestamp" is a string not a datetime.
- Rows of tweets which are later than 08/01 2017 should be removed.
- Tweets which are original ratings should be extracted.
- "name" column are unreliable, thus it should be delete.
- 'rating_numerator' and 'rating_denominator' columns are not necessarily correctly extracted.
- 'rating' column which represents ('rating_numerator / rating_denominator) should be created.
- Stage columns ('doggo', 'floofer','pupper', 'puppo') are not necessarily correctly extracted.

df_image table

- Some pictures are predicted not as dogs. If a picture is not predicted as dog till the 3rd prediction, delete that row.

1.3.2 Tidiness

- "df_api" and "df_image" should be merged to "df_archive"
- Dog stages in the archive data should be in 1 column.
- We need only the most primary confident prediction of dog types from pictures, so make the column "predicted dog type" in place of p1~p3 predictions.

1.4 Clean

```
In [200]: # copy the dataframes for the following cleanness
          archive_clean = df_archive.copy()
          image_clean = df_image.copy()
          api_clean = df_api.copy()
```

1.4.1 Tweets which are original should be extracted from archive table.

```
In [201]: # use .query method to extract rows whose both "in_reply_to_status_id" and "retweete
          archive_clean = archive_clean[archive_clean.in_reply_to_status_id.isnull() & archive_
          archive_clean.info()
```

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

```
In [202]: # remove columns related to "in_reply_status" or "retweeted_status" which is no long
          archive_clean = archive_clean.drop(columns=["in_reply_to_status_id","in_reply_to_use
                                                      "retweeted_status_id","retweeted_status_user_id","retwee
```

```
In [203]: # check columns
          archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2097 entries, 0 to 2355
Data columns (total 12 columns):
tweet_id              2097 non-null int64
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(3), object(9)
memory usage: 213.0+ KB
```

In [204]: archive_clean.columns

Out[204]:
```
Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls',
       'rating_numerator', 'rating_denominator', 'name', 'doggo', 'floofer',
       'pupper', 'puppo'],
      dtype='object')
```

1.4.2 "name" column in archive data is not necessarily in this analysis so delete this.

In [205]:
```python
# use drop() method
archive_clean = archive_clean.drop(columns="name")
```

In [206]:
```python
# confirm that there is no column named "name" now.
assert "name" not in archive_clean.columns
```

1.4.3 "timestamp" in archive data is a string not a datetime.

In [207]:
```python
# use pd.to_datetime() function to convert the data type.
archive_clean.timestamp = pd.to_datetime(archive_clean.timestamp)
```

In [208]:
```python
# confirm it was converted properly and the term in it is possible
archive_clean.timestamp.describe()
```

Out[208]:
```
count                    2097
unique                   2097
top       2016-09-12 15:10:21
freq                        1
first     2015-11-15 22:32:08
last      2017-08-01 16:23:56
Name: timestamp, dtype: object
```

1.4.4 Rows of tweets which are later than 08/01 2017 should be removed from archive data.

In [209]:
```python
# use query() method to extract rows
archive_clean = archive_clean.query('timestamp < datetime.date(2017, 8, 1)')
```

In [210]:
```python
# confirm the latest tweet information in archive data is earlier than  08/01 2017
archive_clean.timestamp.describe()
```

Out[210]:
```
count                    2095
unique                   2095
top       2016-09-12 15:10:21
freq                        1
first     2015-11-15 22:32:08
last      2017-07-31 00:18:03
Name: timestamp, dtype: object
```

1.4.5 'rating_numerator' and 'rating_denominator' columns in archive data are not necessarily correctly extracted.

Definition

We take a look at "text" column in the archieve data. We can assume that ratings are in the form of fractions, so extract the fractions as ratings.

However, we will assess texts of tweets visually in the case where the texts have 2 or more fractions.

When we find that there are 2 or more different dogs' ratings in 1 tweet in this process, delete the tweet because it can be misguiding.

```
In [211]: # Firstly, extract the first fraction which appears in the "text" and confirm that w
          fractions = archive_clean.text.str.extract(r'([0-9]+[.]*[0-9]*[/][0-9]+)')
          fractions.isnull().sum()

Out[211]: 0    0
          dtype: int64

In [212]: # split the fractions into numerators and denominators
          f_nume, f_deno = fractions.iloc[:,0].str.split("/").str
          archive_clean.rating_numerator = f_nume.astype(float)
          archive_clean.rating_denominator = f_deno.astype(float)
          archive_clean.rating_denominator.dtype

Out[212]: dtype('float64')

In [213]: # extract fractions from "text" column using .str.extractall().
          # If there are 2 or more than "text" column, assess each row one by one.
          pd.options.display.max_colwidth = 150

          ratings = archive_clean.text.str.extractall(r'([0-9]+[.]*[0-9]*[/][0-9]+)')

          list_duplicated = []     # list of index whose rows have 2 or more fractions  the "te
          for multi_index in ratings.index:
              if multi_index[1] != 0:
                  list_duplicated.append(multi_index[0])

          print(list_duplicated)
          archive_clean.info()
```

```
[766, 1007, 1068, 1165, 1202, 1222, 1359, 1459, 1465, 1508, 1525, 1538, 1662, 1795, 1832, 1897
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2095 entries, 2 to 2355
Data columns (total 11 columns):
tweet_id              2095 non-null int64
timestamp             2095 non-null datetime64[ns]
source                2095 non-null object
text                  2095 non-null object
```

```
expanded_urls         2092 non-null object
rating_numerator      2095 non-null float64
rating_denominator    2095 non-null float64
doggo                 2095 non-null object
floofer               2095 non-null object
pupper                2095 non-null object
puppo                 2095 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(7)
memory usage: 196.4+ KB
```

In [214]:
```python
# assess each row one by one
archive_clean.text[list_duplicated]
```

Out[214]:
```
766                                     "Yep... just as I suspected. You're not flos
1007                             This is Bookstore and Seaweed. Bookstore is tired and Sea
1068                     After so many requests, this is Bretagne. She was the last surviving
1165
1202                                        This is Bluebert. He just saw that both #Final
1222                 Meet Travis and Flurp. Travis is pretty chill but Flurp can't lie down pr
1359                           This is Socks. That water pup w the super legs just splashed hi
1459                        This may be the greatest video I've ever been sent. 4/10 for Charle
1465           Meet Oliviér. He takes killer selfies. Has a dog of his own. It leaps at ran
1508                 When bae says they can't go out but you see them with someone else that s
1525                   This is Eriq. His friend just reminded him of last year's super bowl. N
1538           Meet Fynn &amp; Taco. Fynn is an all-powerful leaf lord and Taco is in the w
1662                       This is Darrel. He just robbed a 7/11 and is in a high speed police
1795              Meet Tassy &amp; Bee. Tassy is pretty chill, but Bee is convinced the Ruff
1832                         These two pups just met and have instantly bonded. Spectacular sce
1897                 Meet Rufio. He is unaware of the pink legless pupper wrapped around him.
1901                    Two gorgeous dogs here. Little waddling dog is a rebel. Refuses to loo
1970                     Meet Eve. She's a raging alcoholic 8/10 (would b 11/10 but pupper alc
2010                                                                     10/10 for dog. 7/10 for
2010                                                                     10/10 for dog. 7/10 for
2064                  Meet Holly. She's trying to teach small human-like pup about blocks but
2113                              Meet Hank and Sully. Hank is very proud of the pumpkin they foun
2177                    Here we have Pancho and Peaches. Pancho is a Condoleezza Gryffindor, a
2216                 This is Spark. He's nervous. Other dog hasn't moved in a while. Won't com
2263                       This is Kial. Kial is either wearing a cape, which would be rad, or
2272                        Two dogs in this one. Both are rare Jujitsu Pythagoreans. One sligh
2306                 These are Peruvian Feldspars. Their names are Cupit and Prencer. Both res
2335                            This is an Albanian 3 1/2 legged  Episcopalian. Loves well-polish
Name: text, dtype: object
```

In [215]:
```python
def put_ratings(index, numerator, denomenator):
    archive_clean.ix[index,"rating_denominator"] = denomenator
    archive_clean.ix[index,"rating_numerator"] = numerator
```

```
indexes_to_be_deleted = [766, 1007, 1165, 1222, 1459, 1525,1538,1795,1832,1901,2113,2
archive_clean = archive_clean.drop(index = indexes_to_be_deleted)

put_ratings(1068,14,10)
put_ratings(1202,11,10)
put_ratings(1359,9,10)
put_ratings(1465,10,10)
put_ratings(1508,10,10)
put_ratings(1662,10,10)
put_ratings(1897,10,10)
put_ratings(1970,8,10)
put_ratings(2010,10,10)
put_ratings(2064,11,10)
put_ratings(2263,10,10)
put_ratings(2335, 9,10)

archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2080 entries, 2 to 2355
Data columns (total 11 columns):
tweet_id              2080 non-null int64
timestamp             2080 non-null datetime64[ns]
source                2080 non-null object
text                  2080 non-null object
expanded_urls         2077 non-null object
rating_numerator      2080 non-null float64
rating_denominator    2080 non-null float64
doggo                 2080 non-null object
floofer               2080 non-null object
pupper                2080 non-null object
puppo                 2080 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(7)
memory usage: 275.0+ KB


/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  from ipykernel import kernelapp as app
/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:3: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing
```

```
See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  app.launch_new_instance()
```

1.4.6 'rating' column which represents ('rating_numerator / rating_denominator) should be created.

```
In [216]: # create a new column called "rating" and drop 'rating_numerator' and 'rating_denomi
          archive_clean["rating"] = archive_clean.rating_numerator / archive_clean.rating_denor
          archive_clean = archive_clean.drop(columns=['rating_numerator','rating_denominator']
          archive_clean.columns
```

```
Out[216]: Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls', 'doggo',
                 'floofer', 'pupper', 'puppo', 'rating'],
                dtype='object')
```

```
In [217]: # Also, I will check "rating" extremely high or low and remove unappropriate ones.
          archive_clean[(archive_clean.rating < 0.1) | (archive_clean.rating > 2.)].text
```

```
Out[217]: 315                                         When you're so blinded by your systemati
          516     Meet Sam. She smiles 24/7 &amp; secretly aspires to be a reindeer. \nKeep Sa
          979                                                                          This 
          2074                                                                      After so m
          Name: text, dtype: object
```

```
In [218]: # We delete tweet whose picture are not dogs.
          archive_clean = archive_clean.drop([315, 516, 2074])
          archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2077 entries, 2 to 2355
Data columns (total 10 columns):
tweet_id         2077 non-null int64
timestamp        2077 non-null datetime64[ns]
source           2077 non-null object
text             2077 non-null object
expanded_urls    2074 non-null object
doggo            2077 non-null object
floofer          2077 non-null object
pupper           2077 non-null object
puppo            2077 non-null object
rating           2077 non-null float64
dtypes: datetime64[ns](1), float64(1), int64(1), object(7)
memory usage: 178.5+ KB
```

1.4.7 Stage columns ('doggo', 'floofer','pupper', 'puppo') in archive are not necessarily correctly extracted.

In [219]:
```
# First, delete these columns
archive_clean = archive_clean.drop(columns = ['doggo', 'floofer','pupper', 'puppo'])
archive_clean.columns
```

Out[219]:
```
Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls', 'rating'], dtype=
```

In [220]:
```
# add the new column of each stage (boolean)
archive_clean["doggo"] = archive_clean.text.str.contains("doggo")
archive_clean["pupper"] = archive_clean.text.str.contains("pupper")
archive_clean["puppo"] = archive_clean.text.str.contains("puppo")
archive_clean["floof"] = archive_clean.text.str.contains("floof")
archive_clean["snoot"] = archive_clean.text.str.contains("snoot")
archive_clean["blep"] = archive_clean.text.str.contains("blep")

archive_clean[['doggo','pupper', 'puppo','floof','snoot', 'blep']].sum()
```

Out[220]:
```
doggo      84
pupper    245
puppo      29
floof      23
snoot       0
blep        1
dtype: int64
```

In [221]:
```
# assess the rows which has 2 or more stage name in the "text" column.

archive_clean.text[archive_clean[['doggo','pupper', 'puppo','floof','snoot', 'blep']]
```

Out[221]:
```
172     I have stumbled puppon a doggo painting party. They're looking to be the next
191     Here's a puppo participating in the #ScienceMarch. Cleverly disguising her ow
531          Here we have Burke (pupper) and Dexter (doggo). Pupper wants to be exact
575          This is Bones. He's being haunted by another doggo of roughly the same s
705     This is Pinot. He's a sophisticated doggo. You can tell by the hat. Also poin
889          Meet Maggie &amp; Lila. Maggie is the doggo, Lila is the pupper. They ar
956                  Please stop sending it pictures that don't even have a doggo or
1063                                                              This is just downri
1113                                                                              Li
Name: text, dtype: object
```

In [222]:
```
# read each text and choose the most appropriate stage.
# If we find that there are 2 dogs in the picture and thus there are 2 stages, delete
# This is because these rows can mis leading when we consider them with image predic

def cancel_stage(index, stage):
    archive_clean.ix[index, stage] = False
```

```
indexes_to_be_deleted2 = [531, 889, 1062, 1113]
archive_clean = archive_clean.drop(index = indexes_to_be_deleted2)

cancel_stage(172, "puppo")
cancel_stage(191, "doggo")
cancel_stage(575, "doggo")
cancel_stage(705, "pupper")
cancel_stage(956, "doggo")
cancel_stage(1063, "pupper")
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:6: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
```

In [223]:
```
# confirm that there is no row which has 2 or more stages
assert len(archive_clean[archive_clean[['doggo','pupper', 'puppo','floof','snoot', 'l
```

1.4.8 Dog stages in the archive data should be in 1 column.

First, we separate the rows into those which is given stages(df"have_stage_tweets") if dog and those which are not(df"no_stage_tweets").

- For df"no_stage_tweets", we give a "stage" colunn which stores "NaN".
- For df"have_stage_tweets", we give a "stage" colunn which stores the name of stage.

Finally, we concatabnate these dataframes to get the final dataframe.

In [224]:
```
# extract the rows which are not assignes dos stages as "no_stage_tweets".
# construct new column named "stage" which is stored NaN and delete unnecessary colu

no_stage_tweets = archive_clean[archive_clean[['doggo','pupper', 'puppo','floof','sn
no_stage_tweets = no_stage_tweets.drop(columns = ['doggo','pupper', 'puppo','floof',
no_stage_tweets["stage"] = np.nan
no_stage_tweets.columns
```

Out[224]:
```
Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls', 'rating',
       'stage'],
      dtype='object')
```

In [225]:
```
# extract the rows which are assignes any dos stages as "have_stage_tweets".
# construct new column named "stage"  and delete unnecessary columns (using .melt() 
```

```
have_stage_tweets = archive_clean[archive_clean[['doggo','pupper', 'puppo','floof','

## consirm that all rows are in df"no_stage_tweets" or df"have_stage_tweets".
assert len(no_stage_tweets) + len(have_stage_tweets) == len(archive_clean)

have_stage_tweets = have_stage_tweets.melt(id_vars=['tweet_id', 'timestamp', 'source
have_stage_tweets = have_stage_tweets[have_stage_tweets.value == True]
have_stage_tweets = have_stage_tweets.drop(columns = 'value')
have_stage_tweets

## confirm that df"no_stage_tweets" or df"have_stage_tweets" and same columns.
no_stage_tweets.columns == have_stage_tweets.columns
```

Out[225]:
```
array([ True,  True,  True,  True,  True,  True,  True])
```

In [226]:
```
# concatanate df"no_stage_tweets" or df"have_stage_tweets"
archive_clean2 = pd.concat([no_stage_tweets, have_stage_tweets]).sort_index().reset_i
assert len(archive_clean2) == len(archive_clean)

pd.options.display.max_colwidth = 30
archive_clean2.head()

## reset index
archive_clean2 = archive_clean2.reset_index(drop=True)
archive_clean2.head()
```

Out[226]:

	tweet_id	timestamp	source
0	890240255349198849	2017-07-26 15:59:51	<a href="http://twitter.co...
1	891815181378084864	2017-07-31 00:18:03	<a href="http://twitter.co...
2	891689557279858688	2017-07-30 15:58:51	<a href="http://twitter.co...
3	891327558926688256	2017-07-29 16:00:24	<a href="http://twitter.co...
4	884162670584377345	2017-07-09 21:29:42	<a href="http://twitter.co...

	text	expanded_urls	rating	stage
0	This is Cassie. She is a c...	https://twitter.com/dog_ra...	1.4	doggo
1	This is Archie. He is a ra...	https://twitter.com/dog_ra...	1.2	NaN
2	This is Darla. She commenc...	https://twitter.com/dog_ra...	1.3	NaN
3	This is Franklin. He would...	https://twitter.com/dog_ra...	1.2	NaN
4	Meet Yogi. He doesn't have...	https://twitter.com/dog_ra...	1.2	doggo

1.4.9 Some pictures are predicted not as dogs. If a picture is not predicted as dog till the 3rd prediction, delete that row in prediction image data.

In [227]:
```
# use .query() method to extract rows of tweets which are predicted as dogs
image_clean = image_clean.query(' p1_dog == True | p2_dog == True | p3_dog == True ')
```

In [228]:
```
# confirm there is no row which is not predicted as dogs either in p1~p3 predictions
assert ( (image_clean.p1_dog | image_clean.p2_dog | image_clean.p3_dog)==0 ).sum() ==
image_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1751 entries, 0 to 2073
Data columns (total 12 columns):
tweet_id    1751 non-null int64
jpg_url     1751 non-null object
img_num     1751 non-null int64
p1          1751 non-null object
p1_conf     1751 non-null float64
p1_dog      1751 non-null bool
p2          1751 non-null object
p2_conf     1751 non-null float64
p2_dog      1751 non-null bool
p3          1751 non-null object
p3_conf     1751 non-null float64
p3_dog      1751 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 141.9+ KB
```

1.4.10 We need only the most primary confident prediction of dog types from pictures, so make the column "predicted dog type" in place of p1~p3 predictions.

```
In [229]: # create a new list of dog types according to p1 ~p3.
          # add new column "dog_type" in df"image_clean" using it.


          image_clean = image_clean.reset_index(drop=True )
          dog_type_list = []

          for index in image_clean.index:
              if image_clean.p1_dog.iloc[index] == True:
                  dog_type_list.append(image_clean.p1.iloc[index])
              elif image_clean.p2_dog.iloc[index] == True:
                  dog_type_list.append(image_clean.p2.iloc[index])
              elif image_clean.p3_dog.iloc[index] == True:
                  dog_type_list.append(image_clean.p3.iloc[index])
              else:
                  raise   # throw an error when there is no dog type predicted in p1~p3


          image_clean["dog_type"] = dog_type_list
```

```
In [230]: # remove p1~p3 columns because the are no longer needed.
          image_clean = image_clean.drop(columns = ["p1","p1_conf","p1_dog","p2","p2_conf","p2_
```

```
In [231]: # confirm the columns
          image_clean.head()
```

```
Out[231]:               tweet_id                          jpg_url  img_num  \
          0  666020888022790149  https://pbs.twimg.com/medi...        1
          1  666029285002620928  https://pbs.twimg.com/medi...        1
          2  666033412701032449  https://pbs.twimg.com/medi...        1
          3  666044226329800704  https://pbs.twimg.com/medi...        1
          4  666049248165822465  https://pbs.twimg.com/medi...        1

                            dog_type
          0  Welsh_springer_spaniel
          1                redbone
          2         German_shepherd
          3     Rhodesian_ridgeback
          4      miniature_pinscher
```

1.4.11 "df_api" and "df_image" should be merged to "df_archive"

```
In [232]: # use inner merge to merge 3 dataframes to make final dataframe "df"

          df = archive_clean2.merge(api_clean)
          df = df.merge(image_clean)
          df.head()
```

```
Out[232]:               tweet_id           timestamp                          source  \
          0  890240255349198849 2017-07-26 15:59:51  <a href="http://twitter.co...
          1  891815181378084864 2017-07-31 00:18:03  <a href="http://twitter.co...
          2  891689557279858688 2017-07-30 15:58:51  <a href="http://twitter.co...
          3  891327558926688256 2017-07-29 16:00:24  <a href="http://twitter.co...
          4  884162670584377345 2017-07-09 21:29:42  <a href="http://twitter.co...

                                        text                 expanded_urls  rating  \
          0  This is Cassie. She is a c...  https://twitter.com/dog_ra...     1.4
          1  This is Archie. He is a ra...  https://twitter.com/dog_ra...     1.2
          2  This is Darla. She commenc...  https://twitter.com/dog_ra...     1.3
          3  This is Franklin. He would...  https://twitter.com/dog_ra...     1.2
          4  Meet Yogi. He doesn't have...  https://twitter.com/dog_ra...     1.2

             stage   ret    fav                       jpg_url  img_num  \
          0  doggo  7300  31555  https://pbs.twimg.com/medi...        1
          1    NaN  4102  24746  https://pbs.twimg.com/medi...        1
          2    NaN  8540  41677  https://pbs.twimg.com/medi...        1
          3    NaN  9265  39842  https://pbs.twimg.com/medi...        2
          4  doggo  2952  20137  https://pbs.twimg.com/medi...        1

                       dog_type
          0            Pembroke
          1           Chihuahua
          2  Labrador_retriever
          3              basset
```

```
        4     German_shepherd
```

```
In [233]: # confirm final dataframe
          df.info()
```

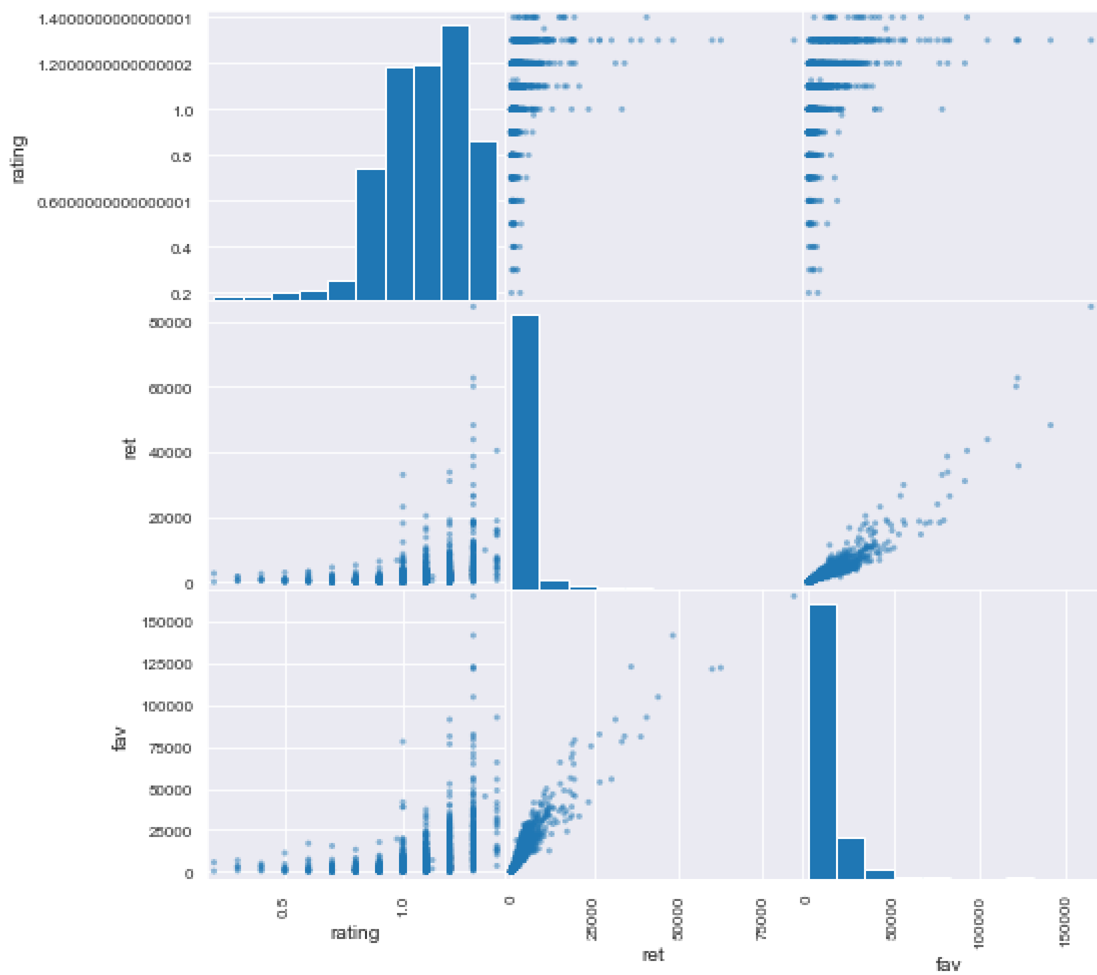
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1645 entries, 0 to 1644
Data columns (total 12 columns):
tweet_id         1645 non-null int64
timestamp        1645 non-null datetime64[ns]
source           1645 non-null object
text             1645 non-null object
expanded_urls    1645 non-null object
rating           1645 non-null float64
stage            287 non-null object
ret              1645 non-null int64
fav              1645 non-null int64
jpg_url          1645 non-null object
img_num          1645 non-null int64
dog_type         1645 non-null object
dtypes: datetime64[ns](1), float64(1), int64(4), object(6)
memory usage: 167.1+ KB
```

1.5 Store

```
In [234]: # store the final dataframe "df" in the csv file.
          df.to_csv("twitter_archieve_master.csv", index=False)
```

Exploratory Data Analysis

```
In [235]: pd.plotting.scatter_matrix(df.iloc[:,5:9], figsize=(8,8));
```



1.5.1 Topic1. How the ratings are related with stages and types of dogs

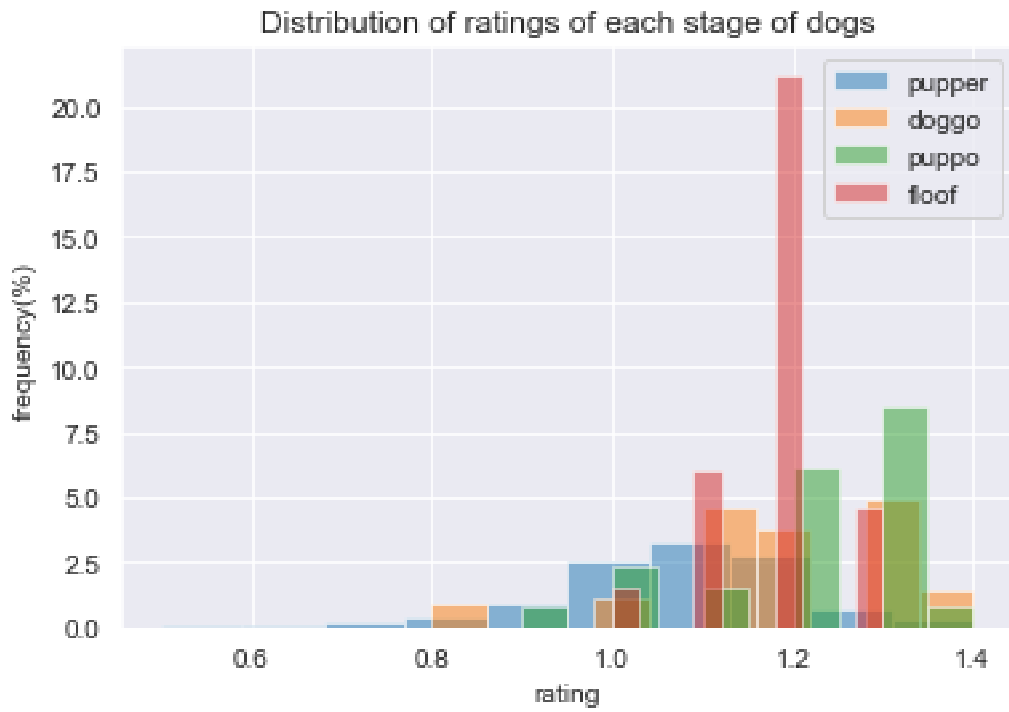
It can be possible that a master of the account has personal preference of dog stages or dog types and the rating is somehow biased. We take a look at the realtionship between dog ratings, stages, and types.

First, we visualize the relationship between dog stages and ratings. The dog stage "blep" has only 1 tweet, so we ignore this stage.

In [236]:
```
sns.set_style("darkgrid")

# visualize each stage's distribution
df[df.stage == "pupper"]["rating"].hist(density=1, alpha=0.5, label="pupper")
df[df.stage == "doggo"]["rating"].hist(density=1,alpha=0.5, label="doggo")
df[df.stage == "puppo"]["rating"].hist(density=1, alpha=0.5, label="puppo")
df[df.stage == "floof"]["rating"].hist(density=1, alpha=0.5, label="floof")
plt.title("Distribution of ratings of each stage of dogs")
```

```
plt.xlabel("rating")
plt.ylabel("frequency(%)")
plt.legend();
```



What we have found is that there seems to be a little effect on the ratings depending on the stage of dogs. The graph suggests that the stage "puppo" tend to have the highest rating, followed by "floof", "doggo", and "pupper".

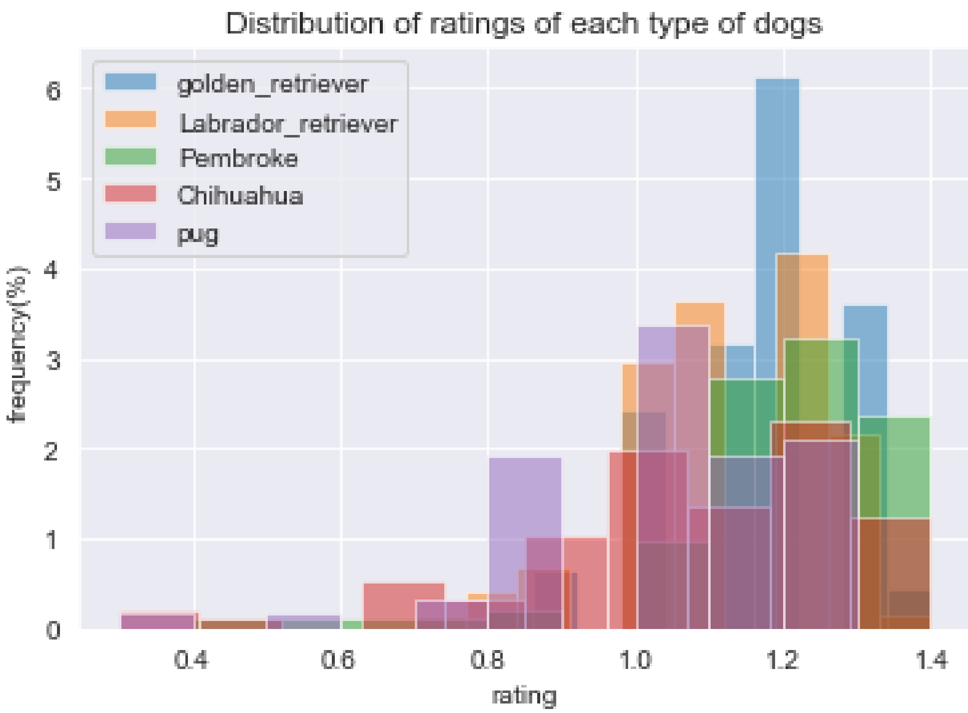
Next, we visualize the relationship between dog types and ratings. We begin by knowing dog type categpry names.

In [237]:
```
# get to know the dog type categories
df.dog_type.value_counts().sort_values(ascending=False).head(10)
```

Out[237]:
```
golden_retriever      152
Labrador_retriever    106
Pembroke               93
Chihuahua              87
pug                    62
toy_poodle             50
chow                   48
Samoyed                42
Pomeranian             41
malamute               33
Name: dog_type, dtype: int64
```

There are too many types of dogs, so we focus only on the 5 most types which are "golden_retriever", "Labrador_retriever", "Pembroke", "Chihuahua", and "pug".

```
In [238]: # visualize each type's distribution
          df[df.dog_type == "golden_retriever"]["rating"].hist(density=1, alpha=0.5, label="gol
          df[df.dog_type == "Labrador_retriever"]["rating"].hist(density=1,alpha=0.5, label="La
          df[df.dog_type == "Pembroke"]["rating"].hist(density=1, alpha=0.5, label="Pembroke")
          df[df.dog_type == "Chihuahua"]["rating"].hist(density=1, alpha=0.5, label="Chihuahua"
          df[df.dog_type == "pug"]["rating"].hist(density=1, alpha=0.5, label="pug")
          plt.title("Distribution of ratings of each type of dogs")
          plt.xlabel("rating")
          plt.ylabel("frequency(%)")
          plt.legend();
```



Appearent gaps have not found between types, but "pug" seems to tend to get rather low ratings.

Now we use multiple linear model for the relationship between ratings and the combination of stages and columns. (In this project, let me skip to check if this really has no problem in terms of multicollinearity .etc.) We set "pupper" and "pug" as baselines and see if the differences of ratings from other each stage or types are statistically significant or not.

```
In [239]: # extract rows which are given both specific stage and type
          df2 = df.copy()
```

```
df2 = df2[df2.stage.isin(["puppo","pupper","doggo","floof"])]
df2 = df2[df2.dog_type.isin(["golden_retriever","Labrador_retriever","Pembroke","Chi

# add intercept and dummy columns of dog stages and types
df2["intercept"] = 1.
df2[["doggo","floof","pupper","puppo"]] = pd.get_dummies(df2.stage)
df2[["Chihuahua","Labrador_retriever","Pembroke","golden_retriever","pug"]] = pd.get_
df2.sample(3)
```

Out[239]:

```
                tweet_id           timestamp                          source  \
0    890240255349198849 2017-07-26 15:59:51  <a href="http://twitter.co...
458  693262851218264065 2016-01-30 02:41:58  <a href="http://twitter.co...
445  699434518667751424 2016-02-16 03:25:58  <a href="http://twitter.co...

                             text                 expanded_urls  rating  \
0    This is Cassie. She is a c...  https://twitter.com/dog_ra...     1.4
458  I hope you guys enjoy this...  https://twitter.com/dog_ra...     1.1
445  I know this is a tad late ...  https://twitter.com/dog_ra...     1.2

      stage   ret    fav                         jpg_url ...   intercept doggo  \
0     doggo  7300  31555  https://pbs.twimg.com/medi... ...         1.0     1
458  pupper   539   2358  https://pbs.twimg.com/medi... ...         1.0     0
445  pupper   547   2287  https://pbs.twimg.com/medi... ...         1.0     0

     floof  pupper  puppo  Chihuahua  Labrador_retriever  Pembroke  \
0        0       0      0          0                   0         1
458      0       1      0          0                   0         0
445      0       1      0          0                   0         0

     golden_retriever  pug
0                   0    0
458                 1    0
445                 1    0

[3 rows x 22 columns]
```

In [240]:

```
# apply multiple linear regression model
lm = sm.OLS(df2.rating, df2[["intercept","doggo","floof","puppo","Chihuahua","Labrado
results = lm.fit()
results.summary()
```

Out[240]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 rating   R-squared:                       0.274
Model:                            OLS   Adj. R-squared:                  0.212
Method:                 Least Squares   F-statistic:                     4.425
```

```
Date:                  Fri, 12 Oct 2018   Prob (F-statistic):           0.000326
Time:                          01:22:52   Log-Likelihood:                 75.298
No. Observations:                    90   AIC:                            -134.6
Df Residuals:                        82   BIC:                            -114.6
Df Model:                             7
Covariance Type:            nonrobust
=================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975
---------------------------------------------------------------------------------
intercept              1.0182      0.033     30.753      0.000       0.952       1.08
doggo                  0.0900      0.029      3.053      0.003       0.031       0.14
floof                  0.0287      0.049      0.589      0.558      -0.068       0.12
puppo                  0.0622      0.045      1.371      0.174      -0.028       0.15
Chihuahua              0.1174      0.050      2.358      0.021       0.018       0.21
Labrador_retriever     0.1015      0.044      2.329      0.022       0.015       0.18
Pembroke               0.1618      0.045      3.631      0.000       0.073       0.25
golden_retriever       0.1237      0.040      3.098      0.003       0.044       0.20
==============================================================================
Omnibus:                        4.612   Durbin-Watson:                   1.713
Prob(Omnibus):                  0.100   Jarque-Bera (JB):                3.882
Skew:                          -0.463   Prob(JB):                        0.144
Kurtosis:                       3.423   Cond. No.                         8.12
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
"""
```

This results suggests that (in terms of dog stages) "doggo" has statistically significant difference from "pupper" and that (in terms of dog types) each of ""Chihuahua","Labrador_retriever","Pembroke","golden_retriever" has statistically significant difference from "pug".

Also, it can be said that 27.4% of rating evaluation can be "explained" by these 2 factors.

1.5.2 Answer1. Relationship between stages of dogs and the ratings.

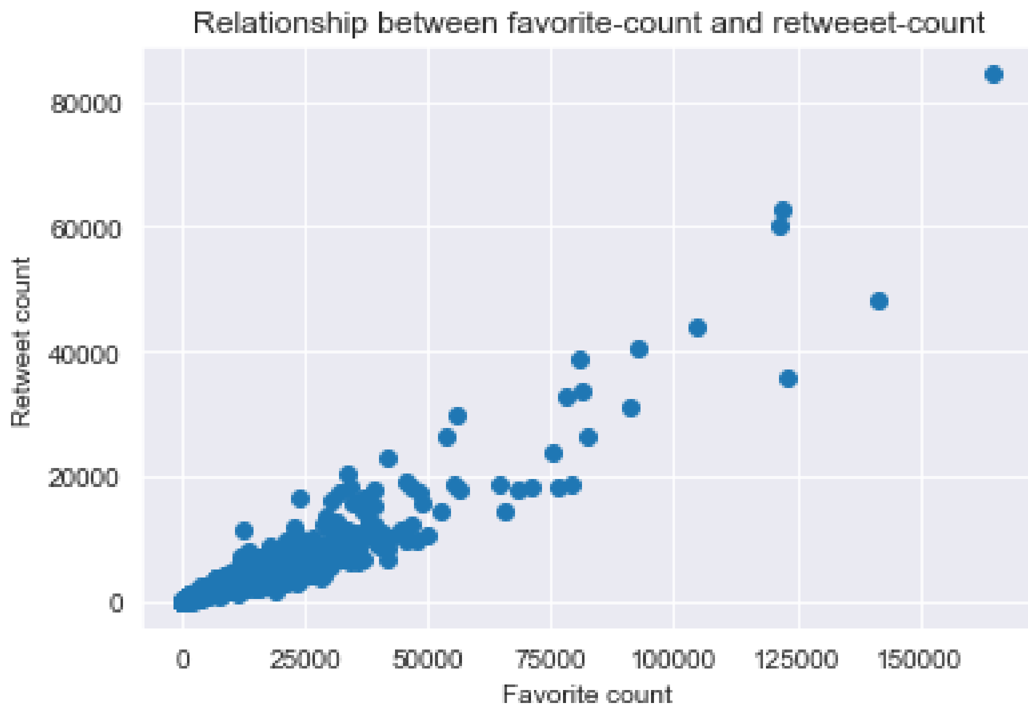
- Regarding dog stages, it is statsistically significant that "doggo" tend to get better ratings than "pupper".
- Regarding dog types, it is statsistically significant that "Chihuahua", "Labrador_retriever", "Pembroke", and "golden_retriever" tend to get better ratings than "pupper".

- 27.4% of rating evaluation juege depends on these 2 factors.

1.5.3 Topic2. Relationship between favorite counts and the ratings

In this data, we have 2 indicators that show reader's preference on tweets, which are "favorite" counts and "retweet" counts. It is easily predictable that tweets with high ratings are favored and also retweeted more and also that there is a strong positive correlationship between "favorite" counts and "retweet" counts. First, this should be checked.

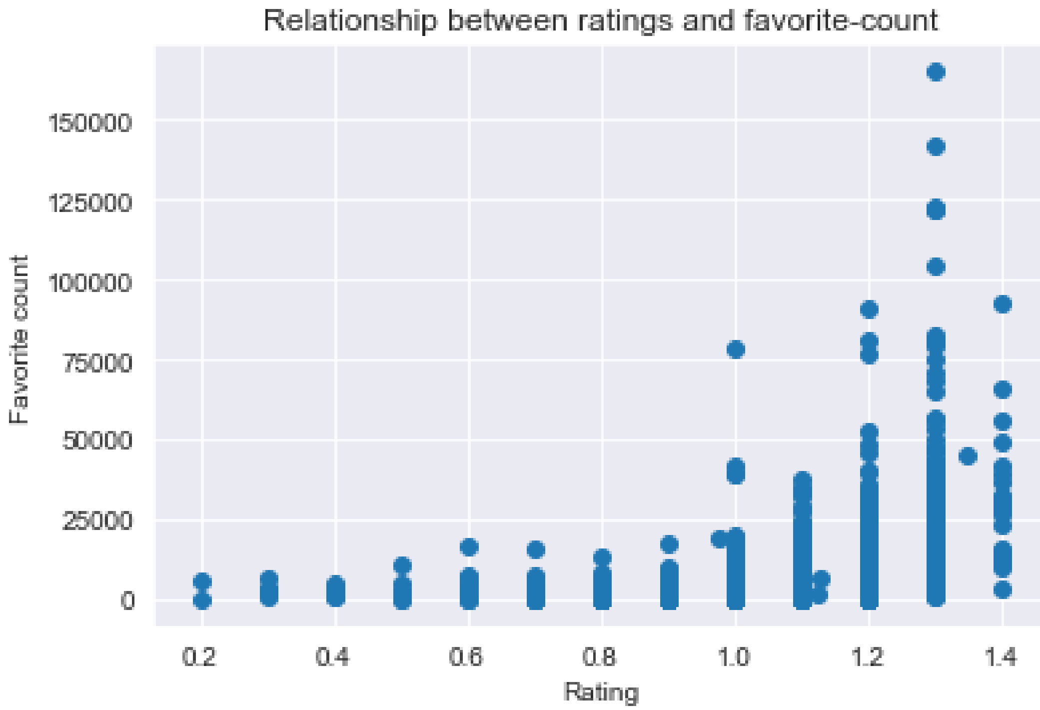
```
In [241]: plt.scatter(df.fav, df.ret)
          plt.title("Relationship between favorite-count and retweeet-count" )
          plt.xlabel("Favorite count")
          plt.ylabel("Retweet count");
```



As we predicted, these counts are correlated. This suggests that we can deem favorite counts as a reliable indicator of tweets' popularity. In this analysis, we focus on "favorite" counts. (And also I have noticed that there is a tweet which had by far the most favs and retweets from the graph. This would be assessed visually later.)

We take a look at the relationship between ratings from the account and the favorite counts.

```
In [242]: plt.scatter(df.rating, df.fav)
          plt.title("Relationship between ratings and favorite-count")
          plt.ylabel("Favorite count")
          plt.xlabel("Rating");
```



We use simple linear regression model to interprete this result (even though it may not be the best way.)

In [243]:
```
df["intercept"] = 1.
lm = sm.OLS(df.fav, df[["intercept","rating"]])
results = lm.fit()
results.summary()
```

Out[243]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    fav   R-squared:                       0.178
Model:                            OLS   Adj. R-squared:                  0.177
Method:                 Least Squares   F-statistic:                     355.2
Date:                Fri, 12 Oct 2018   Prob (F-statistic):           7.00e-72
Time:                        01:22:55   Log-Likelihood:                -17804.
No. Observations:                1645   AIC:                         3.561e+04
Df Residuals:                    1643   BIC:                         3.562e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
intercept   -2.589e+04   1885.466    -13.729      0.000    -2.96e+04    -2.22e+04
rating       3.236e+04   1716.991     18.846      0.000      2.9e+04     3.57e+04
==============================================================================
Omnibus:                     1648.536   Durbin-Watson:                   1.691
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           101955.703
Skew:                           4.749   Prob(JB):                         0.00
Kurtosis:                      40.381   Cond. No.                         12.6
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
"""
```

Even though this suggests that there is a positive correlationship between rating result and favorite accounts, R-squared value(0.178) tells us that there may be a better way to understand this realtionship.

1.5.4 The best tweet which is extermely popular

We take a look at the tweet which had extermely popularity

In [244]:
```
best_tweet = df[df.fav == df.fav.max()]
best_tweet
```

Out[244]:
```
                tweet_id           timestamp                          source  \
139  744234799360020481 2016-06-18 18:26:18  <a href="http://twitter.co...

                                  text                  expanded_urls  rating  \
139  Here's a doggo realizing y...  https://twitter.com/dog_ra...     1.3

     stage    ret     fav                          jpg_url  img_num  \
139  doggo  84426  165057  https://pbs.twimg.com/ext_...        1

               dog_type  intercept
139  Labrador_retriever        1.0
```

In [245]:
```
pd.options.display.max_colwidth = 150
best_tweet.text
```

Out[245]:
```
139    Here's a doggo realizing you can stand in a pool. 13/10 enlightened af (vid by
Name: text, dtype: object
```

I checked the tweet visulally by jumping to the link, but what I have found was just a movie in which a dog is swimming in the pool. It may be said that an ordinary tweet sometimes attracts enormous attention unexpectedly.

1.5.5 Answer2. Relationship between favorite counts and the ratings

- There is a positive correlationship between rating result and favorite accounts.

- This relationship is not completely explained by a linear model.

Conclusions

In this analysis, we focused on the tweet data from a dog-rating twitter account. We began from getting data from api. Then we cleaned and finally analyzed the data.

We found that some some dog stages and types get higher ratings from the account. Favorite count, which is a great indicater of the popularity of tweets, tend to be more for the tweets which are more highly evaluated. However, it is not sure whether reader favored because they found that the rating better, or because pictures themselves are wonderful worth great rating.

Reference

- tweet data of "WeRateDogs (@dog_rates)" from https://twitter.com/dog_rates
- 'image_predictions.tsv' file from "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad predictions/image-predictions.tsv"