wrangle_act

March 24, 2021

1 Gathering

In [77]:
```
import pandas as pd
import requests
import json
#The tweet archive
archive = pd.read_csv('twitter-archive-enhanced.csv')

#Image predictions
url = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predicti
image_get = requests.get(url)
name = url.split('/')[-1]
with open(name, 'wb') as image_file:
    image_file.write(image_get.content)

#final df
image = pd.read_csv(name, sep = '\t')

#Additional data from json file
dict_list = []
with open('tweet-json.txt', 'r') as file_json:
    for line in file_json:
        dic = json.loads(line.strip())
        tweet_id = dic['id']
        retweet = dic['retweet_count']
        favorite = dic['favorite_count']
        dict_list.append({'retweet_count': retweet,
                          'favorite_count': favorite,
                          'tweet_id': tweet_id,})
#final df
api = pd.DataFrame(dict_list, columns = ['tweet_id', 'retweet_count', 'favorite_count']
```

In [78]:
```
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
```

2 Assessing

2.1 Quality

2.1.1 Archive

- 55 names have 'a' as name and 7 as 'an', 8 as 'the': visually by examining the df

```
In [79]: # programmatically
         archive.name.value_counts()
```

```
Out[79]: None          745
         a              55
         Charlie        12
         Oliver         11
         Lucy           11
         Cooper         11
         Tucker         10
         Penny          10
         Lola           10
         Winston         9
         Bo              9
         the             8
         Sadie           8
         Daisy           7
         an              7
         Toby            7
```

```
Bailey          7
Buddy           7
Koda            6
Milo            6
Dave            6
Jax             6
Leo             6
Bella           6
Rusty           6
Jack            6
Scout           6
Oscar           6
Stanley         6
Phil            5
               ...
Harnold         1
Shikha          1
Kane            1
Batdog          1
Julio           1
Chloe           1
Brandy          1
Livvie          1
Buckley         1
Rodney          1
Geoff           1
Holly           1
Dante           1
Sid             1
Ginger          1
Huxley          1
Robin           1
Noosh           1
space           1
Pinot           1
Darla           1
Poppy           1
Ralphson        1
Reptar          1
Bloop           1
Jim             1
Finnegus        1
Franq           1
Chesterson      1
Geno            1
Name: name, Length: 957, dtype: int64
```

- missing names in name column: visually by examining df

```
In [80]: # programmatically
         archive.name.value_counts()
```

```
Out[80]: None          745
         a              55
         Charlie        12
         Oliver         11
         Lucy           11
         Cooper         11
         Tucker         10
         Penny          10
         Lola           10
         Winston         9
         Bo              9
         the             8
         Sadie           8
         Daisy           7
         an              7
         Toby            7
         Bailey          7
         Buddy           7
         Koda            6
         Milo            6
         Dave            6
         Jax             6
         Leo             6
         Bella           6
         Rusty           6
         Jack            6
         Scout           6
         Oscar           6
         Stanley         6
         Phil            5
                        ...
         Harnold         1
         Shikha          1
         Kane            1
         Batdog          1
         Julio           1
         Chloe           1
         Brandy          1
         Livvie          1
         Buckley         1
         Rodney          1
         Geoff           1
         Holly           1
         Dante           1
         Sid             1
```

```
Ginger          1
Huxley          1
Robin           1
Noosh           1
space           1
Pinot           1
Darla           1
Poppy           1
Ralphson        1
Reptar          1
Bloop           1
Jim             1
Finnegus        1
Franq           1
Chesterson      1
Geno            1
Name: name, Length: 957, dtype: int64
```

- some values in rating_denominator are not 10: visually by examining the df column

```
In [81]: # programmatically
         archive.rating_denominator.value_counts()
```

```
Out[81]: 10     2333
         11        3
         50        3
         80        2
         20        2
         2         1
         16        1
         40        1
         70        1
         15        1
         90        1
         110       1
         120       1
         130       1
         150       1
         170       1
         7         1
         0         1
         Name: rating_denominator, dtype: int64
```

- existence of replies and retweets: visually by finding non-null values in the columns: in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, and retweeted_status_timestamp

```
In [82]: # programmatically
         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
```

- existence of tweets without image predictions: only programmatically

```
In [83]: # programmatically
         len(archive) - len(image) > 181 + 78
```

```
Out[83]: True
```

- timestamps are objects instead of datetime: only programmatically

```
In [84]: # programmatically
         archive.timestamp.dtype
```

```
Out[84]: dtype('O')
```

- tweets containing photos of non-dogs: visually only by checking one of the tweets and finding it is not a dog, could be confirmed by image predictions since there are predictions for non-dogs

```
In [85]: # programmatically
         sum(image.p1_dog == False)
```

```
Out[85]: 543
```

2.1.2 Image

- some predictions for unoriginal tweets might exist: only programmatically, the difference between their lenghts is more than the number of retweets and replies.

```
In [86]: # programmatically
         len(archive) - len(image) - (181 + 78)
```

```
Out[86]: 22
```

- three predictions per one image: visually by examining df columns

```
In [87]: # programmatically
         image.columns
```

```
Out[87]: Index(['tweet_id', 'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2',
                'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
               dtype='object')
```

- some predictions show the images don't belong to dogs: visually by examining prediction columns

```
In [88]: # programmatically
         sum(image.p1_dog == False)
```

```
Out[88]: 543
```

2.2 Tidiness

2.2.1 Archive

- doggo, floofer, pupper, puppo should be only one column determining the stage of dog: mostly, three of these columns are empty and one is filled or all are empty as confirmed visually

```
In [89]: # programmatically
         archive.sample(20)
```

```
Out[89]:                tweet_id  in_reply_to_status_id  in_reply_to_user_id  \
         2150  669683899023405056                    NaN                  NaN
         1440  696886256886657024                    NaN                  NaN
         545   805826884734976000                    NaN                  NaN
         2314  666701168228331520                    NaN                  NaN
         2104  670668383499735048                    NaN                  NaN
         816   770787852854652928                    NaN                  NaN
         753   778774459159379968                    NaN                  NaN
         2205  668633411083464705                    NaN                  NaN
         957   751538714308972544                    NaN                  NaN
         2254  667782464991965184                    NaN                  NaN
         871   761599872357261312                    NaN                  NaN
         1058  741793263812808706                    NaN                  NaN
```

```
115   870374049280663552                   NaN                  NaN
1931  674036086168010753                   NaN                  NaN
1748  679062614270468097                   NaN                  NaN
1029  745712589599014916                   NaN                  NaN
663   790946055508652032                   NaN                  NaN
2304  666983947667116034                   NaN                  NaN
2048  671511350426865664                   NaN                  NaN
2229  668248472370458624                   NaN                  NaN

                      timestamp  \
2150  2015-11-26 01:07:38 +0000
1440  2016-02-09 02:40:05 +0000
545   2016-12-05 17:31:15 +0000
2314  2015-11-17 19:35:19 +0000
2104  2015-11-28 18:19:37 +0000
816   2016-08-31 00:58:39 +0000
753   2016-09-22 01:54:34 +0000
2205  2015-11-23 03:33:22 +0000
957   2016-07-08 22:09:27 +0000
2254  2015-11-20 19:12:01 +0000
871   2016-08-05 16:28:54 +0000
1058  2016-06-12 00:44:30 +0000
115   2017-06-01 20:18:38 +0000
1931  2015-12-08 01:21:40 +0000
1748  2015-12-21 22:15:18 +0000
1029  2016-06-22 20:18:30 +0000
663   2016-10-25 16:00:09 +0000
2304  2015-11-18 14:18:59 +0000
2048  2015-12-01 02:09:16 +0000
2229  2015-11-22 02:03:45 +0000

                                                 source  \
2150  <a href="http://twitter.com/download/iphone" r...
1440  <a href="http://twitter.com/download/iphone" r...
545   <a href="http://twitter.com/download/iphone" r...
2314  <a href="http://twitter.com/download/iphone" r...
2104  <a href="http://twitter.com/download/iphone" r...
816   <a href="http://twitter.com/download/iphone" r...
753   <a href="http://twitter.com/download/iphone" r...
2205  <a href="http://twitter.com/download/iphone" r...
957   <a href="http://twitter.com/download/iphone" r...
2254  <a href="http://twitter.com/download/iphone" r...
871   <a href="http://twitter.com/download/iphone" r...
1058  <a href="http://twitter.com/download/iphone" r...
115   <a href="http://twitter.com/download/iphone" r...
1931  <a href="http://twitter.com/download/iphone" r...
1748  <a href="http://twitter.com/download/iphone" r...
1029  <a href="http://twitter.com/download/iphone" r...
```

```
663   <a href="http://twitter.com/download/iphone" r...
2304  <a href="http://twitter.com/download/iphone" r...
2048  <a href="http://twitter.com/download/iphone" r...
2229  <a href="http://twitter.com/download/iphone" r...

                                                  text  retweeted_status_id  \
2150  This is Kloey. Her mother was a unicorn. 10/10...                   NaN
1440  Guys I found the dog from Up. 12/10 https://t...                  NaN
545   This is Duke. He is not a fan of the pupporazz...                   NaN
2314  This is a golden Buckminsterfullerene named Jo...                   NaN
2104  This is Phineas. He's a magical dog. Only appe...                   NaN
816   This is Winston. His tongue has gone rogue. Do...                   NaN
753   RT @dog_rates: In case you haven't seen the mo...         7.580996e+17
2205  This is Churlie. He likes bagels. 10/10 https:...                   NaN
957   This is Max. She has one ear that's always sli...                   NaN
2254  Super rare dog. Endangered (?). Thinks it's fu...                   NaN
871   This is Sephie. According to this picture, she...                   NaN
1058  When your crush won't pay attention to you. Bo...                   NaN
115   This is Zoey. She really likes the planet. Wou...                   NaN
1931  Meet Daisy. She has no eyes &amp;amp; her face has...                   NaN
1748  This is Chompsky. He lives up to his name. 11/...                   NaN
1029  This is Percy. He fell asleep at the wheel. Ir...                   NaN
663   This is Betty. She's assisting with the dishes...                   NaN
2304  This is a curly Ticonderoga named Pepe. No fee...                   NaN
2048  Say hello to Hammond. He's just a wee lil pup...                  NaN
2229  Say hello to Bisquick. He is a Brown Douglass ...                   NaN

      retweeted_status_user_id retweeted_status_timestamp  \
2150                       NaN                        NaN
1440                       NaN                        NaN
545                        NaN                        NaN
2314                       NaN                        NaN
2104                       NaN                        NaN
816                        NaN                        NaN
753               4.196984e+09  2016-07-27 00:40:12 +0000
2205                       NaN                        NaN
957                        NaN                        NaN
2254                       NaN                        NaN
871                        NaN                        NaN
1058                       NaN                        NaN
115                        NaN                        NaN
1931                       NaN                        NaN
1748                       NaN                        NaN
1029                       NaN                        NaN
663                        NaN                        NaN
2304                       NaN                        NaN
2048                       NaN                        NaN
2229                       NaN                        NaN
```

	expanded_urls	rating_numerator	\
2150	https://twitter.com/dog_rates/status/669683899...	10	
1440	https://twitter.com/dog_rates/status/696886256...	12	
545	https://twitter.com/dog_rates/status/805826884...	12	
2314	https://twitter.com/dog_rates/status/666701168...	8	
2104	https://twitter.com/dog_rates/status/670668383...	10	
816	https://twitter.com/dog_rates/status/770787852...	10	
753	https://vine.co/v/hQJbaj1VpIz,https://vine.co/...	13	
2205	https://twitter.com/dog_rates/status/668633411...	10	
957	https://twitter.com/dog_rates/status/751538714...	10	
2254	https://twitter.com/dog_rates/status/667782464...	9	
871	https://twitter.com/dog_rates/status/761599872...	11	
1058	https://twitter.com/dog_rates/status/741793263...	10	
115	https://twitter.com/dog_rates/status/870374049...	13	
1931	https://twitter.com/dog_rates/status/674036086...	9	
1748	https://twitter.com/dog_rates/status/679062614...	11	
1029	https://twitter.com/dog_rates/status/745712589...	7	
663	https://twitter.com/dog_rates/status/790946055...	12	
2304	https://twitter.com/dog_rates/status/666983947...	11	
2048	https://twitter.com/dog_rates/status/671511350...	8	
2229	https://twitter.com/dog_rates/status/668248472...	8	

	rating_denominator	name	doggo	floofer	pupper	puppo
2150	10	Kloey	None	None	None	None
1440	10	None	None	None	None	None
545	10	Duke	None	None	None	None
2314	10	a	None	None	None	None
2104	10	Phineas	None	None	None	None
816	10	Winston	None	None	None	None
753	10	None	None	None	None	None
2205	10	Churlie	None	None	None	None
957	10	Max	None	None	None	None
2254	10	None	None	None	None	None
871	10	Sephie	None	None	None	None
1058	10	None	None	None	None	None
115	10	Zoey	None	None	None	None
1931	10	Daisy	None	None	None	None
1748	10	Chompsky	None	None	None	None
1029	10	Percy	None	None	None	None
663	10	Betty	None	None	None	puppo
2304	10	a	None	None	None	None
2048	10	Hammond	None	None	None	None
2229	10	Bisquick	None	None	None	None

2.3 All dfs

- the 3 dfs contain observation of a shared subject, but are separated into 3 dfs: visually by finding shared tweet ids between the 3 dfs

```
In [90]: # programmatically
         tweets = list(image.tweet_id.unique())
         sum(archive.tweet_id.isin(tweets))
```

```
Out[90]: 2075
```

```
In [91]: sum(api.tweet_id.isin(tweets))
```

```
Out[91]: 2073
```

3 Clean

3.1 Quality

3.1.1 Archive

3.1.2 Define

- convert all 'a's, 'an's and 'the's into 'None'
- missing names cannot be filled, will be left only as 'None'

3.1.3 Code

```
In [92]: archive_clean = archive.copy()
         archive_clean.name = archive_clean.name.replace('a', 'None')
         archive_clean.name = archive_clean.name.replace('an', 'None')
         archive_clean.name = archive_clean.name.replace('the', 'None')
```

3.1.4 Test

```
In [93]: sum(archive_clean.name == 'a')
```

```
Out[93]: 0
```

```
In [94]: sum(archive_clean.name == 'an')
```

```
Out[94]: 0
```

```
In [95]: sum(archive_clean.name == 'the')
```

```
Out[95]: 0
```

```
In [96]: sum(archive.name == 'None'), sum(archive_clean.name == 'None')
```

```
Out[96]: (745, 815)
```

```
In [97]: sum(archive_clean.name == 'None') - sum(archive.name == 'None')

Out[97]: 70

In [98]: #The sum of all occurrences = 70
         sum(archive.name == 'a') + sum(archive.name == 'an') + sum(archive.name == 'the')

Out[98]: 70

In [ ]:
```

3.1.5 Define

- convert all denominators indivisible by 10 into 10

3.1.6 Code

```
In [99]: for row in range(len(archive_clean)):
             if archive_clean.loc[row]['rating_denominator'] %10 != 0 or archive_clean.loc[row][
                 archive_clean.loc[row]= 10
```

3.1.7 Test

```
In [100]: archive_clean.rating_denominator.value_counts()

Out[100]: 10     2341
          50        3
          80        2
          20        2
          170       1
          150       1
          130       1
          120       1
          110       1
          90        1
          70        1
          40        1
          Name: rating_denominator, dtype: int64
```

3.1.8 Define

- create dog_counts column using the denominator column

3.1.9 Code

```
In [101]: archive_clean['dog_counts'] = archive_clean['rating_denominator']/10
```

3.1.10 Test

```
In [102]: archive_clean.dog_counts.value_counts()
```

```
Out[102]: 1.0     2341
          5.0        3
          8.0        2
          2.0        2
          12.0       1
          11.0       1
          13.0       1
          4.0        1
          9.0        1
          17.0       1
          15.0       1
          7.0        1
          Name: dog_counts, dtype: int64
```

```
In [ ]:
```

3.1.11 Define

- divide all ratings in numerator by dog counts

3.1.12 Code

```
In [103]: archive_clean['rating_numerator'] /= archive_clean['dog_counts']
```

3.1.13 Test

```
In [104]: archive_clean['rating_numerator'].value_counts()
```

```
Out[104]: 12.0      562
          10.0      471
          11.0      469
          13.0      351
          9.0       157
          8.0       102
          7.0        54
          14.0       54
          5.0        37
          6.0        32
          3.0        19
          4.0        15
          2.0        11
          1.0         8
          0.0         2
          75.0        2
          15.0        2
```

```
420.0       2
666.0       1
26.0        1
17.0        1
1776.0      1
27.0        1
182.0       1
Name: rating_numerator, dtype: int64
```

```
In [ ]:
```

3.1.14 Define

- drop all replies and retweets to leave only original tweets

3.1.15 Code

```
In [105]: import numpy as np
          # looping over a list of the two columns for retweets and replies
          # to exclude all non-null values from archive
          unoriginal = ['in_reply_to_status_id', 'retweeted_status_id']
          for column in unoriginal:
              unoriginal_archive = archive_clean[column].notnull()
              # dropping retweets and replies from archive
              archive_clean = archive_clean[~unoriginal_archive]

          # a list of all unneeded columns
          unoriginal_full = ['in_reply_to_status_id', 'in_reply_to_user_id',
                             'retweeted_status_id', 'retweeted_status_user_id',
                             'retweeted_status_timestamp']
          # dropping all columns
          archive_clean.drop(columns = unoriginal_full, inplace = True)
```

3.1.16 Test

```
In [106]: archive_clean.columns
```

```
Out[106]: Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls',
                 'rating_numerator', 'rating_denominator', 'name', 'doggo', 'floofer',
                 'pupper', 'puppo', 'dog_counts'],
                dtype='object')
```

```
In [ ]:
```

3.1.17 Define

- remove tweets with no images

3.1.18 Code

```
In [107]: # a list of tweet_ids with images
          tweets_image = list(image.tweet_id.unique())
          # remove all tweets with no images
          archive_clean = archive_clean[archive_clean.tweet_id.isin(tweets_image)]
```

3.1.19 Test

```
In [108]: archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1967 entries, 0 to 2355
Data columns (total 13 columns):
tweet_id              1967 non-null int64
timestamp             1967 non-null object
source                1967 non-null object
text                  1967 non-null object
expanded_urls         1967 non-null object
rating_numerator      1967 non-null float64
rating_denominator    1967 non-null int64
name                  1967 non-null object
doggo                 1967 non-null object
floofer               1967 non-null object
pupper                1967 non-null object
puppo                 1967 non-null object
dog_counts            1967 non-null float64
dtypes: float64(2), int64(2), object(9)
memory usage: 215.1+ KB
```

3.1.20 Define

- make timestamp column into datetime

3.1.21 Code

```
In [109]: archive_clean['timestamp'] = pd.to_datetime(archive_clean['timestamp'])
```

3.1.22 Test

```
In [110]: archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1967 entries, 0 to 2355
Data columns (total 13 columns):
tweet_id              1967 non-null int64
timestamp             1967 non-null datetime64[ns]
source                1967 non-null object
```

```
text                  1967 non-null object
expanded_urls         1967 non-null object
rating_numerator      1967 non-null float64
rating_denominator    1967 non-null int64
name                  1967 non-null object
doggo                 1967 non-null object
floofer               1967 non-null object
pupper                1967 non-null object
puppo                 1967 non-null object
dog_counts            1967 non-null float64
dtypes: datetime64[ns](1), float64(2), int64(2), object(8)
memory usage: 215.1+ KB
```

In [ ]:

3.1.23 Image

3.1.24 Define

- remove predictions for unoriginals

3.1.25 Code

In [111]:
```
image_clean = image.copy()
np.logical_not(image_clean.tweet_id.isin(list(archive_clean.tweet_id)))
image_clean[~np.logical_not(image_clean.tweet_id.isin(list(archive_clean.tweet_id)))]
```

Out[111]:

	tweet_id	jpg_url
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg
5	666050758794694657	https://pbs.twimg.com/media/CT5Jof1WUAEuVxN.jpg
6	666051853826850816	https://pbs.twimg.com/media/CT5KoJ1WoAAJash.jpg
7	666055525042405380	https://pbs.twimg.com/media/CT5N9tpXIAAifs1.jpg
8	666057090499244032	https://pbs.twimg.com/media/CT5PY90WoAAQGLo.jpg
9	666058600524156928	https://pbs.twimg.com/media/CT5Qw94XAAA_2dP.jpg
10	666063827256086533	https://pbs.twimg.com/media/CT5Vg_wXIAAXfnj.jpg
11	666071193221509120	https://pbs.twimg.com/media/CT5cN_3WEAAl0oZ.jpg
12	666073100786774016	https://pbs.twimg.com/media/CT5d9DZXAAALcwe.jpg
13	666082916733198337	https://pbs.twimg.com/media/CT5m4VGWEAAtKc8.jpg
14	666094000022159362	https://pbs.twimg.com/media/CT5w9gUW4AAsBNN.jpg
15	666099513787052032	https://pbs.twimg.com/media/CT51-JJUEAA6hV8.jpg
16	666102155909144576	https://pbs.twimg.com/media/CT54YGiWUAEZnoK.jpg
17	666104133288665088	https://pbs.twimg.com/media/CT56LSZWoAAlJj2.jpg
18	666268910803644416	https://pbs.twimg.com/media/CT8QCd1WEAADXws.jpg
19	666273097616637952	https://pbs.twimg.com/media/CT8T1mtUwAA3aqm.jpg

```
21    666293911632134144    https://pbs.twimg.com/media/CT8mx7KW4AEQu8N.jpg
22    666337882303524864    https://pbs.twimg.com/media/CT9OwFIWEAMuRje.jpg
23    666345417576210432    https://pbs.twimg.com/media/CT9Vn7PWoAA_ZCM.jpg
24    666353288456101888    https://pbs.twimg.com/media/CT9cx0tUEAAhNN_.jpg
25    666362758909284353    https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg
26    666373753744588802    https://pbs.twimg.com/media/CT9vZEYWUAAlZO5.jpg
27    666396247373291520    https://pbs.twimg.com/media/CT-D2ZHWIAA3gK1.jpg
28    666407126856765440    https://pbs.twimg.com/media/CT-NvwmW4AAugGZ.jpg
29    666411507551481857    https://pbs.twimg.com/media/CT-RugiWIAELEaq.jpg
30    666418789513326592    https://pbs.twimg.com/media/CT-YWb7U8AA7QnN.jpg
...                  ...                                               ...
2044  886258384151887873    https://pbs.twimg.com/media/DEyfTG4UMAE4aE9.jpg
2045  886366144734445568    https://pbs.twimg.com/media/DEOBTnQUwAApKEH.jpg
2046  886680336477933568    https://pbs.twimg.com/media/DE4fEDzWAAAyHMM.jpg
2047  886736880519319552    https://pbs.twimg.com/media/DE5Se8FXcAAJFx4.jpg
2048  886983233522544640    https://pbs.twimg.com/media/DE8yicJW0AAAvBJ.jpg
2049  887101392804085760    https://pbs.twimg.com/media/DE-eAq6UwAA-jaE.jpg
2050  887343217045368832  https://pbs.twimg.com/ext_tw_video_thumb/88734...
2051  887473957103951883    https://pbs.twimg.com/media/DFDw2tyUQAAAFke.jpg
2052  887517139158093824  https://pbs.twimg.com/ext_tw_video_thumb/88751...
2053  887705289381826560    https://pbs.twimg.com/media/DFHDQBbXgAEqY7t.jpg
2054  888078434458587136    https://pbs.twimg.com/media/DFMWn56WsAAkA7B.jpg
2056  888554962724278272    https://pbs.twimg.com/media/DFTH_O-UQAACu20.jpg
2057  888804989199671297    https://pbs.twimg.com/media/DFWra-3VYAA2piG.jpg
2058  888917238123831296    https://pbs.twimg.com/media/DFYRgsOUQAARGhO.jpg
2059  889278841981685760  https://pbs.twimg.com/ext_tw_video_thumb/88927...
2060  889531135344209921    https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg
2061  889638837579907072    https://pbs.twimg.com/media/DFihzFfXsAYGDPR.jpg
2062  889665388333682689    https://pbs.twimg.com/media/DFi579UWsAAatzw.jpg
2063  889880896479866881    https://pbs.twimg.com/media/DFl99B1WsAITKsg.jpg
2064  890006608113172480    https://pbs.twimg.com/media/DFnwSY4WAAAMliS.jpg
2065  890240255349198849    https://pbs.twimg.com/media/DFrEyVuW0AAO3t9.jpg
2066  890609185150312448    https://pbs.twimg.com/media/DFwUU__XcAEpyXI.jpg
2067  890729181411237888    https://pbs.twimg.com/media/DFyBahAVwAAhUTd.jpg
2068  890971913173991426    https://pbs.twimg.com/media/DF1eOmZXUAALUcq.jpg
2069  891087950875897856    https://pbs.twimg.com/media/DF3HwyEWsAABqE6.jpg
2070  891327558926688256    https://pbs.twimg.com/media/DF6hr6BUMAAzZgT.jpg
2071  891689557279858688    https://pbs.twimg.com/media/DF_q7IAWsAEuuN8.jpg
2072  891815181378084864    https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg
2073  892177421306343426    https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg
2074  892420643555336193    https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg

      img_num                          p1   p1_conf  p1_dog  \
0           1       Welsh_springer_spaniel  0.465074    True
1           1                     redbone  0.506826    True
2           1              German_shepherd  0.596461    True
3           1          Rhodesian_ridgeback  0.408143    True
4           1           miniature_pinscher  0.560311    True
```

5	1	Bernese_mountain_dog	0.651137	True
6	1	box_turtle	0.933012	False
7	1	chow	0.692517	True
8	1	shopping_cart	0.962465	False
9	1	miniature_poodle	0.201493	True
10	1	golden_retriever	0.775930	True
11	1	Gordon_setter	0.503672	True
12	1	Walker_hound	0.260857	True
13	1	pug	0.489814	True
14	1	bloodhound	0.195217	True
15	1	Lhasa	0.582330	True
16	1	English_setter	0.298617	True
17	1	hen	0.965932	False
18	1	desktop_computer	0.086502	False
19	1	Italian_greyhound	0.176053	True
21	1	three-toed_sloth	0.914671	False
22	1	ox	0.416669	False
23	1	golden_retriever	0.858744	True
24	1	malamute	0.336874	True
25	1	guinea_pig	0.996496	False
26	1	soft-coated_wheaten_terrier	0.326467	True
27	1	Chihuahua	0.978108	True
28	1	black-and-tan_coonhound	0.529139	True
29	1	coho	0.404640	False
30	1	toy_terrier	0.149680	True
...	...	...	...	...
2044	1	pug	0.943575	True
2045	1	French_bulldog	0.999201	True
2046	1	convertible	0.738995	False
2047	1	kuvasz	0.309706	True
2048	2	Chihuahua	0.793469	True
2049	1	Samoyed	0.733942	True
2050	1	Mexican_hairless	0.330741	True
2051	2	Pembroke	0.809197	True
2052	1	limousine	0.130432	False
2053	1	basset	0.821664	True
2054	1	French_bulldog	0.995026	True
2056	3	Siberian_husky	0.700377	True
2057	1	golden_retriever	0.469760	True
2058	1	golden_retriever	0.714719	True
2059	1	whippet	0.626152	True
2060	1	golden_retriever	0.953442	True
2061	1	French_bulldog	0.991650	True
2062	1	Pembroke	0.966327	True
2063	1	French_bulldog	0.377417	True
2064	1	Samoyed	0.957979	True
2065	1	Pembroke	0.511319	True
2066	1	Irish_terrier	0.487574	True

```
2067        2                  Pomeranian  0.566142    True
2068        1                 Appenzeller  0.341703    True
2069        1     Chesapeake_Bay_retriever  0.425595    True
2070        2                      basset  0.555712    True
2071        1                 paper_towel  0.170278   False
2072        1                   Chihuahua  0.716012    True
2073        1                   Chihuahua  0.323581    True
2074        1                      orange  0.097049   False

                          p2   p2_conf  p2_dog                          p3  \
0                     collie  0.156665    True            Shetland_sheepdog
1           miniature_pinscher  0.074192    True          Rhodesian_ridgeback
2                    malinois  0.138584    True                   bloodhound
3                     redbone  0.360687    True           miniature_pinscher
4                   Rottweiler  0.243682    True                     Doberman
5             English_springer  0.263788    True   Greater_Swiss_Mountain_dog
6                   mud_turtle  0.045885   False                      terrapin
7               Tibetan_mastiff  0.058279    True                     fur_coat
8               shopping_basket  0.014594   False             golden_retriever
9                     komondor  0.192305    True  soft-coated_wheaten_terrier
10              Tibetan_mastiff  0.093718    True           Labrador_retriever
11           Yorkshire_terrier  0.174201    True                      Pekinese
12            English_foxhound  0.175382    True                  Ibizan_hound
13                bull_mastiff  0.404722    True               French_bulldog
14             German_shepherd  0.078260    True                     malinois
15                    Shih-Tzu  0.166192    True               Dandie_Dinmont
16                Newfoundland  0.149842    True                       borzoi
17                        cock  0.033919   False                    partridge
18                        desk  0.085547   False                     bookcase
19                toy_terrier  0.111884    True                      basenji
21                      otter  0.015250   False               great_grey_owl
22                Newfoundland  0.278407    True                  groenendael
23    Chesapeake_Bay_retriever  0.054787    True           Labrador_retriever
24              Siberian_husky  0.147655    True                    Eskimo_dog
25                        skunk  0.002402   False                       hamster
26                Afghan_hound  0.259551    True                       briard
27                toy_terrier  0.009397    True                      papillon
28                  bloodhound  0.244220    True         flat-coated_retriever
29                  barracouta  0.271485   False                           gar
30                    papillon  0.148258    True                     Chihuahua
...                        ...       ...     ...                           ...
2044                shower_cap  0.025286   False                  Siamese_cat
2045                 Chihuahua  0.000361    True                  Boston_bull
2046                sports_car  0.139952   False                    car_wheel
2047             Great_Pyrenees  0.186136    True               Dandie_Dinmont
2048                toy_terrier  0.143528    True                   can_opener
2049                Eskimo_dog  0.035029    True    Staffordshire_bullterrier
2050                  sea_lion  0.275645   False                    Weimaraner
```

2051	Rhodesian_ridgeback	0.054950	True	beagle
2052	tow_truck	0.029175	False	shopping_cart
2053	redbone	0.087582	True	Weimaraner
2054	pug	0.000932	True	bull_mastiff
2056	Eskimo_dog	0.166511	True	malamute
2057	Labrador_retriever	0.184172	True	English_setter
2058	Tibetan_mastiff	0.120184	True	Labrador_retriever
2059	borzoi	0.194742	True	Saluki
2060	Labrador_retriever	0.013834	True	redbone
2061	boxer	0.002129	True	Staffordshire_bullterrier
2062	Cardigan	0.027356	True	basenji
2063	Labrador_retriever	0.151317	True	muzzle
2064	Pomeranian	0.013884	True	chow
2065	Cardigan	0.451038	True	Chihuahua
2066	Irish_setter	0.193054	True	Chesapeake_Bay_retriever
2067	Eskimo_dog	0.178406	True	Pembroke
2068	Border_collie	0.199287	True	ice_lolly
2069	Irish_terrier	0.116317	True	Indian_elephant
2070	English_springer	0.225770	True	German_short-haired_pointer
2071	Labrador_retriever	0.168086	True	spatula
2072	malamute	0.078253	True	kelpie
2073	Pekinese	0.090647	True	papillon
2074	bagel	0.085851	False	banana

	p3_conf	p3_dog
0	0.061428	True
1	0.072010	True
2	0.116197	True
3	0.222752	True
4	0.154629	True
5	0.016199	True
6	0.017885	False
7	0.054449	False
8	0.007959	True
9	0.082086	True
10	0.072427	True
11	0.109454	True
12	0.097471	True
13	0.048960	True
14	0.075628	True
15	0.089688	True
16	0.133649	True
17	0.000052	False
18	0.079480	False
19	0.111152	True
21	0.013207	False
22	0.102643	True
23	0.014241	True

```
24    0.093412    True
25    0.000461   False
26    0.206803    True
27    0.004577    True
28    0.173810    True
29    0.189945   False
30    0.142860    True
...        ...     ...
2044  0.002849   False
2045  0.000076    True
2046  0.044173   False
2047  0.086346    True
2048  0.032253   False
2049  0.029705    True
2050  0.134203    True
2051  0.038915    True
2052  0.026321   False
2053  0.026236    True
2054  0.000903    True
2056  0.111411    True
2057  0.073482    True
2058  0.105506    True
2059  0.027351    True
2060  0.007958    True
2061  0.001498    True
2062  0.004633    True
2063  0.082981   False
2064  0.008167    True
2065  0.029248    True
2066  0.118184    True
2067  0.076507    True
2068  0.193548   False
2069  0.076902   False
2070  0.175219    True
2071  0.040836   False
2072  0.031379    True
2073  0.068957    True
2074  0.076110   False

[1967 rows x 12 columns]
```

3.1.26 Test

```
In [112]: image_clean.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
```

3.1.27 Define

- remove columns for prediction 2 and 3

3.1.28 Code

```
In [113]: # a list of all unneeded columns
          weak_predictions = ['p2', 'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog']
          # dropping all columns
          image_clean.drop(columns = weak_predictions, inplace = True)
          # renaming prediction columns
          image_clean.rename(columns = {'p1': 'p', 'p1_conf': 'p_conf'}, inplace = True)
```

3.1.29 Test

```
In [114]: image_clean.columns

Out[114]: Index(['tweet_id', 'jpg_url', 'img_num', 'p', 'p_conf', 'p1_dog'], dtype='object')
```

3.1.30 Define

- remove all non-dog images from image df

3.1.31 Code

```
In [115]: # non-dog series
          non_dog = image_clean['p1_dog'] == False
          # dropping non-dogs
          image_clean = image_clean[~non_dog]

In [116]: # dropping p1_dog column
          image_clean.drop(columns = ['p1_dog'], inplace = True)
```

3.1.32 Test

```
In [117]: len(non_dog) == len(image_clean)
```

```
Out[117]: False
```

```
In [ ]:
```

3.1.33 Archive continue

3.1.34 Define

- remove all tweets for non-dog images

3.1.35 Code

```
In [118]: # creating a list of tweet_ids for image df
          dog_image = list(image_clean.tweet_id.unique())

          archive_clean = archive_clean[archive_clean.tweet_id.isin(dog_image)]
```

3.1.36 Test

```
In [119]: len(archive_clean)
```

```
Out[119]: 1460
```

```
In [ ]:
```

4 Tidiness

4.1 Archive

4.1.1 Define

- create dog_stage column and remove the 4 unneeded columns

4.1.2 Code

```
In [120]: # looping over archive in each of the stage columns to search for stage
          stage  = []
          for row in range(len(archive_clean)):
              if archive_clean.iloc[row]['doggo'] != 'None':
                  stage.append(archive_clean.iloc[row]['doggo'])
              elif archive_clean.iloc[row]['floofer'] != 'None':
                  stage.append(archive_clean.iloc[row]['floofer'])
              elif archive_clean.iloc[row]['pupper'] != 'None':
                  stage.append(archive_clean.iloc[row]['pupper'])
              elif archive_clean.iloc[row]['puppo'] != 'None':
                  stage.append(archive_clean.iloc[row]['puppo'])
```

```
            else:
                # if stage not found in any, append 'None specified'
                stage.append('None specified')
        # create new stage column
        archive_clean['stage'] = stage
        # drop all 4 columns
        stage_column = ['doggo', 'floofer', 'pupper', 'puppo']
        archive_clean.drop(columns = stage_column , inplace = True)
```

4.1.3 Test

```
In [121]: archive_clean.columns
```

```
Out[121]: Index(['tweet_id', 'timestamp', 'source', 'text', 'expanded_urls',
                 'rating_numerator', 'rating_denominator', 'name', 'dog_counts',
                 'stage'],
                dtype='object')
```

```
In [122]: archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 1 to 2355
Data columns (total 10 columns):
tweet_id              1460 non-null int64
timestamp             1460 non-null datetime64[ns]
source                1460 non-null object
text                  1460 non-null object
expanded_urls         1460 non-null object
rating_numerator      1460 non-null float64
rating_denominator    1460 non-null int64
name                  1460 non-null object
dog_counts            1460 non-null float64
stage                 1460 non-null object
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 125.5+ KB
```

```
In [ ]:
```

4.2 Master df

4.2.1 Define

- align api_clean and image_clean according to archive_clean tweet_id, then merge the three into one df

4.2.2 Code

```
In [123]: api_clean = api.copy()
```

```
In [124]: # a list for tweet ids the final version of archive
          archive_final = list(archive_clean.tweet_id.unique())
          # removing all tweets not in archive from images
          image_clean = image_clean[image_clean.tweet_id.isin(archive_final)]
          # a list for tweet ids the final version of archive
          image_final = list(image_clean.tweet_id.unique())
          # cleaning api according to final image and archive
          api_clean = api_clean[api_clean.tweet_id.isin(archive_final)]
          api_clean = api_clean[api_clean.tweet_id.isin(image_final)]
```

4.2.3 Test

```
In [125]: len(api_clean) == len(image_clean) == len(archive_clean) == 1460
```

```
Out[125]: True
```

```
In [126]: # merging into a master df
          from functools import reduce
          dfs = [archive_clean, api_clean, image_clean]
          twitter_archive = reduce(lambda left,right: pd.merge(left,right,on=['tweet_id'],
                                                   how='outer'), dfs)
```

```
In [127]: twitter_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 0 to 1459
Data columns (total 16 columns):
tweet_id              1460 non-null int64
timestamp             1460 non-null datetime64[ns]
source                1460 non-null object
text                  1460 non-null object
expanded_urls         1460 non-null object
rating_numerator      1460 non-null float64
rating_denominator    1460 non-null int64
name                  1460 non-null object
dog_counts            1460 non-null float64
stage                 1460 non-null object
retweet_count         1460 non-null int64
favorite_count        1460 non-null int64
jpg_url               1460 non-null object
img_num               1460 non-null int64
p                     1460 non-null object
p_conf                1460 non-null float64
dtypes: datetime64[ns](1), float64(3), int64(5), object(7)
memory usage: 193.9+ KB
```

5 Saving into csv file

```
In [128]: twitter_archive.to_csv('twitter_archive_master.csv', index = False)
```

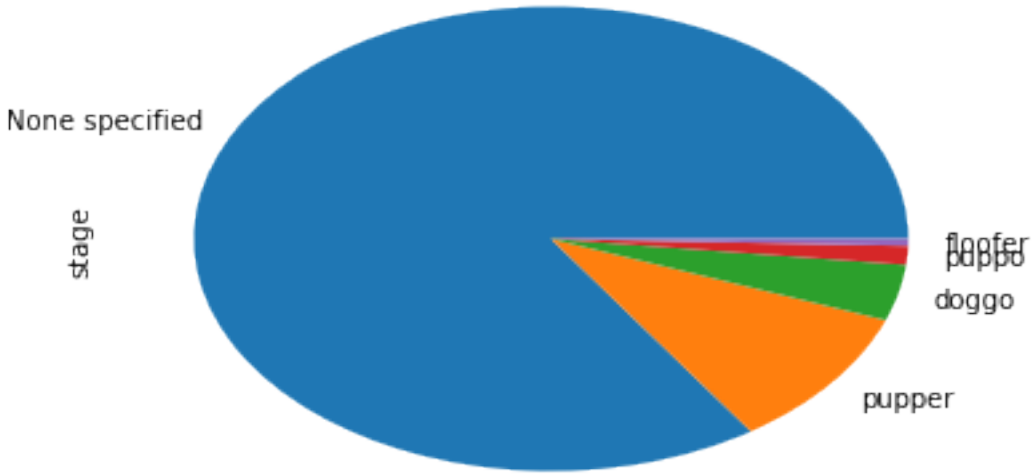
6 Visualize

```
In [129]: import matplotlib.pyplot as plt
          % matplotlib inline
          twitter_archive[twitter_archive['dog_counts'] != 1].dog_counts
```

```
Out[129]: 261      7.0
          528     15.0
          697      2.0
          725      5.0
          743      9.0
          757      8.0
          773      5.0
          823      5.0
          882      4.0
          1006    11.0
          1138     8.0
          Name: dog_counts, dtype: float64
```

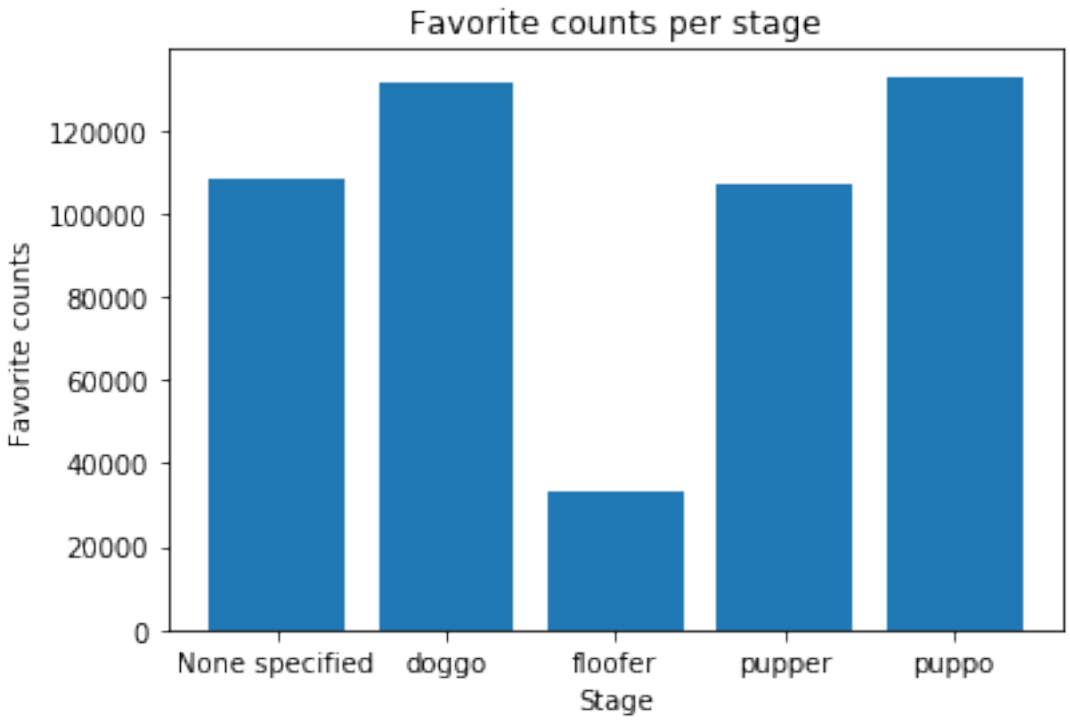
```
In [130]: # checking numbers of each stage
          twitter_archive.stage.value_counts().plot(kind = 'pie')
```

```
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x7f949fcddf60>
```



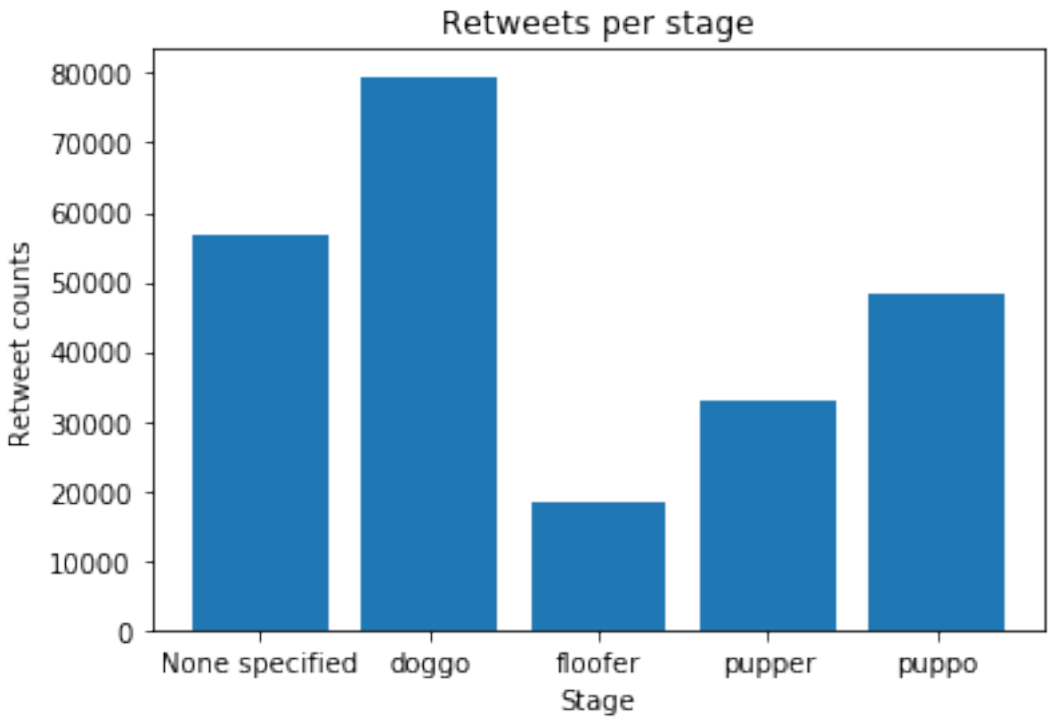

```
In [131]: # checking differences in favorite counts between stages
          plt.bar(twitter_archive.stage, twitter_archive.favorite_count)
          plt.title('Favorite counts per stage')
          plt.xlabel('Stage')
          plt.ylabel('Favorite counts')
```

Out[131]: Text(0,0.5,'Favorite counts')



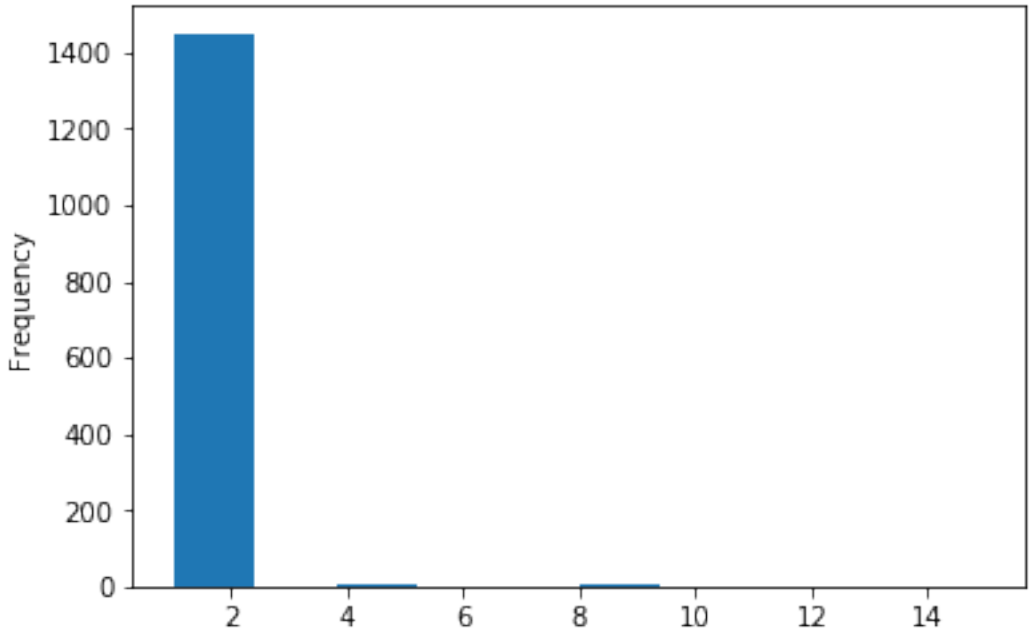
In [132]:
```
# checking differences in retweet counts between stages
plt.bar(twitter_archive.stage, twitter_archive.retweet_count)
plt.title('Retweets per stage')
plt.xlabel('Stage')
plt.ylabel('Retweet counts')
```

Out[132]: Text(0,0.5,'Retweet counts')



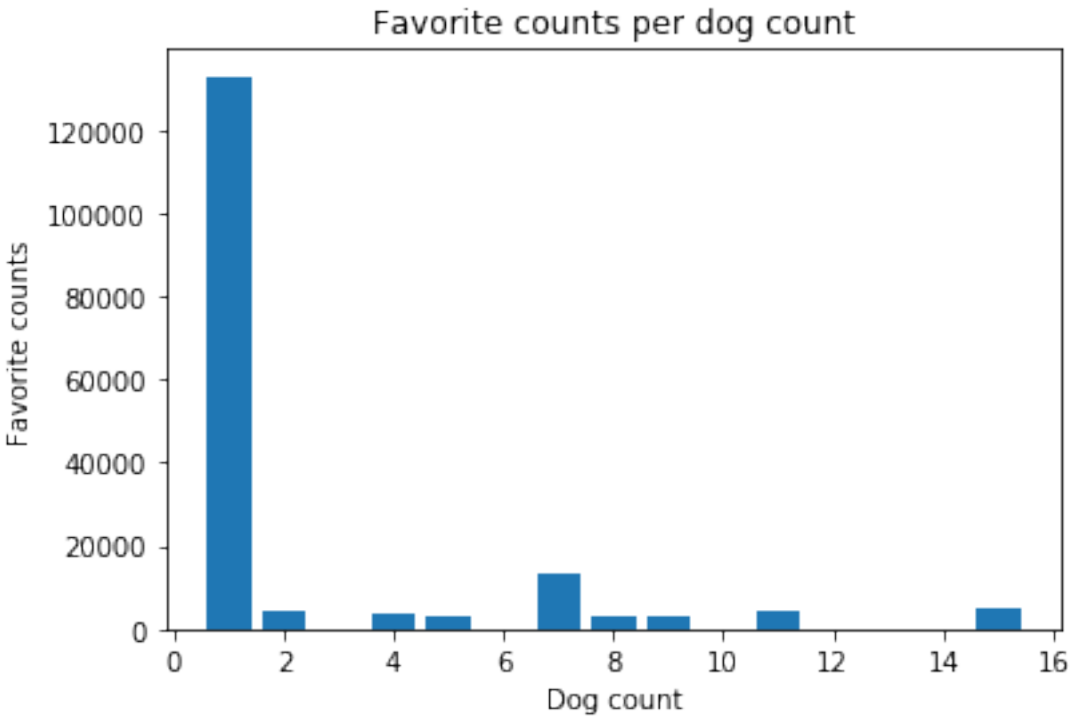

```
In [133]: # checking numbers of each count
          twitter_archive.dog_counts.plot(kind = 'hist')
```

```
Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x7f949f090b00>
```



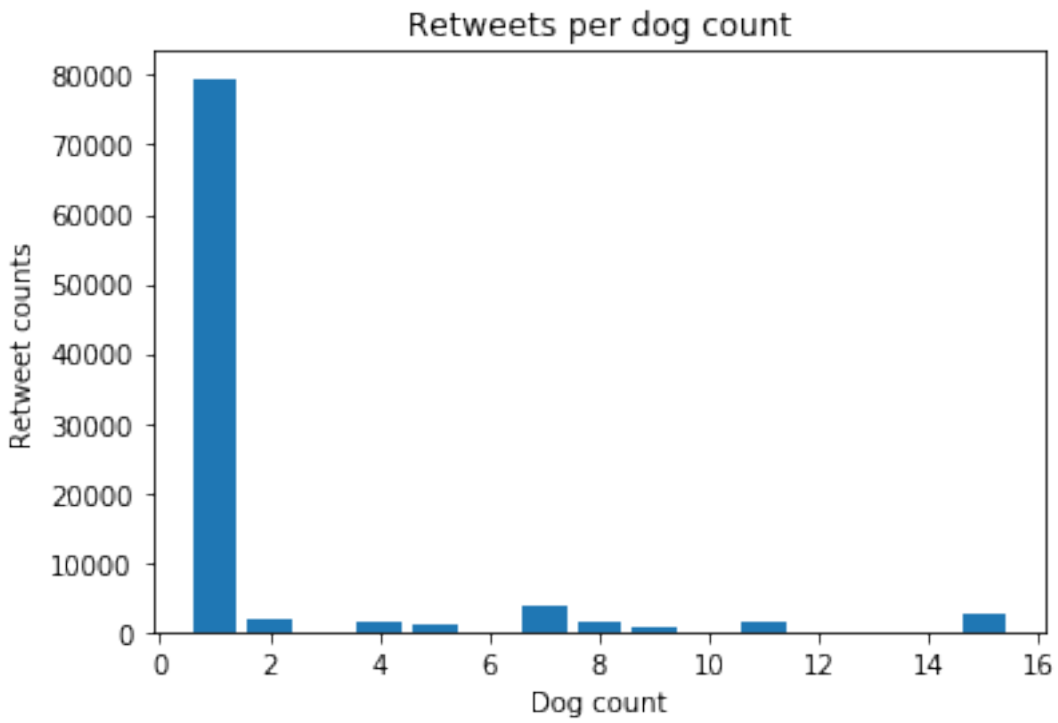
In [134]:
```
# checking  favorite counts for each count
plt.bar(twitter_archive.dog_counts, twitter_archive.favorite_count)
plt.title('Favorite counts per dog count')
plt.xlabel('Dog count')
plt.ylabel('Favorite counts')
```

Out[134]:
```
Text(0,0.5,'Favorite counts')
```

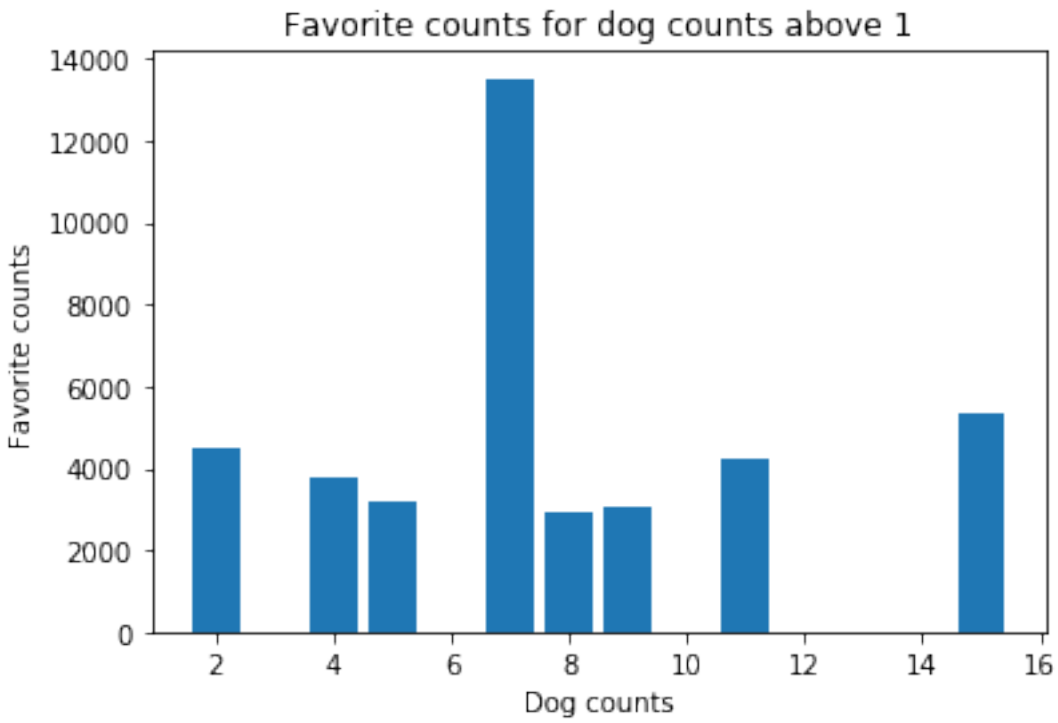


In [135]:
```
# checking  retweet counts for each count
plt.bar(twitter_archive.dog_counts, twitter_archive.retweet_count)
plt.title('Retweets per dog count')
plt.xlabel('Dog count')
plt.ylabel('Retweet counts')
```

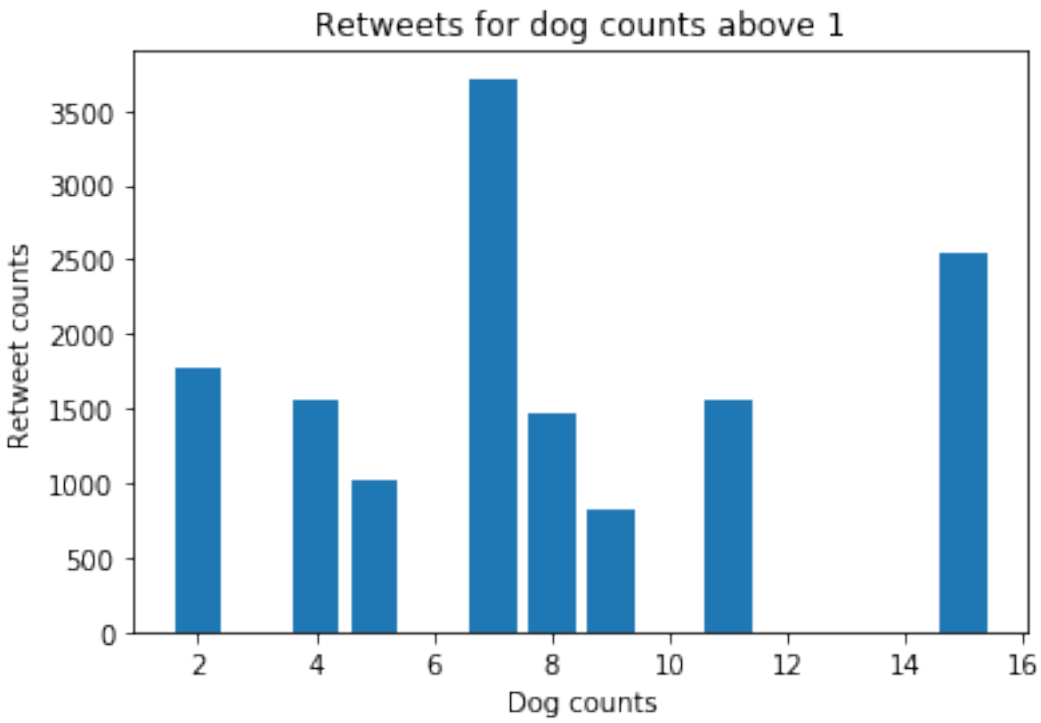
Out[135]:
```
Text(0,0.5,'Retweet counts')
```




```
In [136]: # checking  mean favorite counts for tweets with 1 dog
          twitter_archive[twitter_archive['dog_counts'] == 1].favorite_count.mean()

Out[136]: 9370.8737060041403

In [137]: # checking  mean favorite counts for tweets with 1 dog
          twitter_archive[twitter_archive['dog_counts'] == 1].retweet_count.mean()

Out[137]: 2811.4195997239476

In [138]: # a bar graph for favorite counts for tweets with more than 1 dog
          plt.bar(twitter_archive[twitter_archive['dog_counts'] != 1].dog_counts,
                  twitter_archive[twitter_archive['dog_counts'] != 1].favorite_count)
          plt.title('Favorite counts for dog counts above 1')
          plt.xlabel('Dog counts')
          plt.ylabel('Favorite counts')

Out[138]: Text(0,0.5,'Favorite counts')
```




```
In [139]: # a bar graph for retweet counts for tweets with more than 1 dog
          plt.bar(twitter_archive[twitter_archive['dog_counts'] != 1].dog_counts,
                  twitter_archive[twitter_archive['dog_counts'] != 1].retweet_count)
          plt.title('Retweets for dog counts above 1')
          plt.xlabel('Dog counts')
          plt.ylabel('Retweet counts')

Out[139]: Text(0,0.5,'Retweet counts')
```


Retweets for dog counts above 1
Retweet counts
Dog counts
0
500
1000
1500
2000
2500
3000
3500
2
4
6
8
10
12
14
16