Project: Wrangling and Analyze Data

In [1]:
```
#Importing Libraries
import requests #download data
import numpy as np #array functions
import pandas as pd #data handling
import tweepy #twitter api
import json #handle json data
import matplotlib.pyplot as plt #data visualization
import seaborn as sns #data visualization
import re #text processing
from tweepy import OAuthHandler
import json# open json file
from timeit import default_timer as timer # checking computing time
```

Data Gathering

In the cell below, gather **all** three pieces of data for this project and load them in the notebook. **Note:** the methods required to gather each data are different.

1. Directly download the WeRateDogs Twitter archive data (twitter_archive_enhanced.csv)

In [2]:
```
df_1 = pd.read_csv("twitter_archive_enhanced.csv")
df_1.head()
```

Out[2]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.com/down
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/down
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/down
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.com/down
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.com/down

1. Use the Requests library to download the tweet image prediction (image_predictions.tsv)

In [3]:
```
url = "https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictio

#get response
response = requests.get(url)

#write return to an image
with open("image_predictions.tsv", mode = "wb") as file:
    file.write(response.content)
df_im= pd.read_csv("image_predictions.tsv", sep='\t')
df_im.head(2)
```

Out[3]:

	tweet_id	jpg_url	img_num	p
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spanie
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbon

In [5]:
```
# Comparing with Downloaded data from udacity classrom
df_i= pd.read_csv("image_predictions.tsv", sep='\t')
df_i.head(2)
```

Out[5]:

	tweet_id	jpg_url	img_num	p
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spanie
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbon

3. Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

In [6]:
```
from timeit import default_timer as timer
```

In [7]:
```
# Query Twitter API for each tweet in the Twitter archive and save JSON in a text file
# These are hidden to comply with Twitter's API terms and conditions
consumer_key =
consumer_secret =
access_token =
access_secret =
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
```

In [8]:
```
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
# I just use pip install tweepy==3.10.0 to accept wait_on_rate_limit_notify=True
```

I just use pip install tweepy==3.10.0 to accept wait_on_rate_limit_notify=True

In [9]:
```
# api = tweepy.API(auth, wait_on_rate_limit = True, wait_on_rate_limit_notify = True)
df_1.columns
```

Out[9]:
```
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')
```

In [10]:
```
tweet_ids = list(df_1.tweet_id)
tweet_data = {}
fails_dict = {}
start = timer()
for tweet_id in tweet_ids:
    try:
        tweet_status = api.get_status(tweet_id, tweet_mode='extended')
```

```
        tweet_data[str(tweet_id)] = tweet_status._json
    except tweepy.TweepError as e:
        print("Error for: " +str(tweet_id))
        fails_dict[str(tweet_id)] = e

end = timer()
print(end - start)
print(fails_dict)
```

```
Error for: 888202515573088257
Error for: 877611172832227328
Error for: 873697596434513921
Error for: 872668790621863937
Error for: 872261713294495745
Error for: 869988702071779329
Error for: 866816280283807744
Error for: 861769973181624320
Error for: 856602993587888130
Error for: 856330835276025856
Error for: 851953902622658560
Error for: 851861385021730816
Error for: 845459076796616705
Error for: 844704788403113984
Error for: 842892208864923648
Error for: 839290600511926273
Error for: 837366284874571778
Error for: 837012587749474308
Error for: 829374341691346946
Error for: 827228250799742977
Error for: 812747805718642688
Error for: 802247111496568832
Error for: 779123168116150273
Error for: 775096608509886464
Error for: 771004394259247104
Error for: 770743923962707968
Error for: 766864461642756096
Error for: 759923798737051648
Error for: 759566828574212096
Rate limit reached. Sleeping for: 160
Error for: 754011816964026368
Error for: 717790033953034240
Error for: 680055455951884288
Error for: 676533798876651520
2234.414643084
{'888202515573088257': TweepError([{'code': 144, 'message': 'No status found with that I
D.'}]), '877611172832227328': TweepError([{'code': 179, 'message': 'Sorry, you are not a
uthorized to see this status.'}]), '873697596434513921': TweepError([{'code': 144, 'mess
age': 'No status found with that ID.'}]), '872668790621863937': TweepError([{'code': 14
4, 'message': 'No status found with that ID.'}]), '872261713294495745': TweepError([{'co
de': 144, 'message': 'No status found with that ID.'}]), '869988702071779329': TweepErro
r([{'code': 144, 'message': 'No status found with that ID.'}]), '866816280283807744': Tw
eepError([{'code': 144, 'message': 'No status found with that ID.'}]), '8617699731816243
20': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '856602993
587888130': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '85
6330835276025856': TweepError([{'code': 144, 'message': 'No status found with that I
D.'}]), '851953902622658560': TweepError([{'code': 144, 'message': 'No status found with
that ID.'}]), '851861385021730816': TweepError([{'code': 144, 'message': 'No status foun
d with that ID.'}]), '845459076796616705': TweepError([{'code': 144, 'message': 'No stat
us found with that ID.'}]), '844704788403113984': TweepError([{'code': 144, 'message':
'No status found with that ID.'}]), '842892208864923648': TweepError([{'code': 144, 'mes
sage': 'No status found with that ID.'}]), '839290600511926273': TweepError([{'code': 17
9, 'message': 'Sorry, you are not authorized to see this status.'}]), '83736628487457177
8': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '8370125877
49474308': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '829
374341691346946': TweepError([{'code': 144, 'message': 'No status found with that I
```

```
D.'}]), '827228250799742977': TweepError([{'code': 144, 'message': 'No status found with
that ID.'}]), '812747805718642688': TweepError([{'code': 144, 'message': 'No status foun
d with that ID.'}]), '802247111496568832': TweepError([{'code': 144, 'message': 'No stat
us found with that ID.'}]), '779123168116150273': TweepError([{'code': 144, 'message':
'No status found with that ID.'}]), '775096608509886464': TweepError([{'code': 144, 'mes
sage': 'No status found with that ID.'}]), '771004394259247104': TweepError([{'code': 17
9, 'message': 'Sorry, you are not authorized to see this status.'}]), '770743923962707968': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '766864461642756096': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '759923798737051648': TweepError([{'code': 144, 'message': 'No status found with that I
D.'}]), '759566828574212096': TweepError([{'code': 144, 'message': 'No status found with
that ID.'}]), '754011816964026368': TweepError([{'code': 144, 'message': 'No status foun
d with that ID.'}]), '717790033953034240': TweepError("Failed to send request: ('Connect
ion aborted.', ConnectionResetError(54, 'Connection reset by peer'))"), '680055455951884288': TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), '676533798876651520': TweepError("Failed to send request: HTTPSConnectionPool(host='api.twitter.com', port=443): Read timed out. (read timeout=60)")}
```

In [11]:
```
print("Number of tweet ids : %d"% len (fails_dict))
```

```
Number of tweet ids : 33
```

We can see that there are 31 tweet ids which are not present in the twitter website

In [12]:
```
#Writing the data into text file
with open('tweet_json.txt', 'w') as file:
    json.dump(tweet_data, file)
```

In [13]:
```
#Reading the text file in json format
with open('tweet_json.txt') as file:
    data = json.load(file)


tweets_info_list = []
for tweet_id in data.keys():
    retweets = data[tweet_id]['retweet_count']
    favourites = data[tweet_id]['favorite_count']
    followers = data[tweet_id]['user']['followers_count']
    friends = data[tweet_id]['user']['friends_count']
    tweets_info_list.append({'tweet_id': tweet_id, 'retweets' : retweets,
                             'favorites': favourites
                             ,
                            'followers' : followers
                             ,
                             'friends' : friends
                            })

mytweet_df = pd.DataFrame(tweets_info_list, columns = ['tweet_id', 'retweets', 'favorite
mytweet_df.sample(7)
```

Out[13]:

	tweet_id	retweets	favorites	followers	friends
1937	673295268553605120	2699	6579	9375727	20
979	747512671126323200	1452	5001	9375704	20
62	879862464715927552	2877	19155	9375693	20
1988	672082170312290304	317	821	9375728	20
1259	708130923141795840	769	3034	9375713	20
865	759197388317847553	1775	5586	9375702	20
1782	676617503762681856	843	2565	9375719	20

Data Gathered

df_1 - this is a dataset "twitter-archive-enhanced.csv" which was converted into a dataframe and gives information on basic tweet data.

mytweet_df- This dataset will contain information like tweet_id, no of retweets and no of favorites etc.,

df_i - This dataset will contain information about predictions about the image.

Assessing Data

In this section, detect and document at least **eight (8) quality issues and two (2) tidiness issue**. You must use **both** visual assessment programmatic assessement to assess the data.

Note: pay attention to the following key points when you access the data.

- You only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.
- Assessing and cleaning the entire dataset completely would require a lot of time, and is not necessary to practice and demonstrate your skills in data wrangling. Therefore, the requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned. This unique rating system is a big part of the popularity of WeRateDogs.
- You do not need to gather the tweets beyond August 1st, 2017. You can, but note that you won't be able to gather the image predictions for these tweets since you don't have access to the algorithm used.

Visual Assesment

In [14]:
```
# Visual Assesment of twitter-archive-enhanced data set
df_1.sample(6)
```

Out[14]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
535	807059379405148160	NaN	NaN	2016-12-09 03:08:45 +0000	href="http://twitter.com/d
653	792050063153438720	NaN	NaN	2016-10-28 17:07:05 +0000	href="http://twitter.com/d
2135	670061506722140161	NaN	NaN	2015-11-27 02:08:07 +0000	href="http://twitter.com/d
1945	673707060090052608	NaN	NaN	2015-12-07 03:34:14 +0000	href="http://twitter.com/d

380	827600520311402496	NaN	NaN	2017-02-03 19:31:54 +0000	href="http://twitter.com/d
694	786729988674449408	NaN	NaN	2016-10-14 00:47:00 +0000	href="http://twitter.com/d

1. We can see that there is missing values in 'in_reply_to_status_id', 'in_reply_to_user_id', retweeted_status_id retweeted_status_user_id,retweeted_status_timestamp columns of twitter-archive-enhanced data set
2. we can also see that some columns in twitter-archive-enhanced data set such as 'name', 'doggo', 'floofer', 'pupper' have missing values which have been replace with a value 'None' thus they will not appear as missing but as object.
3. We can see text column has mixed lower and upper case, while other rows have upper case only (eg index 943) of of twitter-archive-enhanced data set.
4. Special characters are also dominant in text column of twitter-archive-enhanced data set.

In [15]:
```
# Visual Assesment of image_predictions data set
df_i.sample(10)
```

Out[15]:

	tweet_id	jpg_url	img_num	p1
1775	828372645993398273	https://pbs.twimg.com/media/C374hb0WQAAIbQ-.jpg	1	malamute
1805	832273440279240704	https://pbs.twimg.com/ext_tw_video_thumb/83227...	1	Pembroke
1449	776201521193218049	https://pbs.twimg.com/media/CsWfKadWEAAtmlS.jpg	1	Rottweiler
1599	799422933579902976	https://pbs.twimg.com/media/Cxge6AdUQAAvXLB.jpg	1	miniature_pinscher
1310	754120377874386944	https://pbs.twimg.com/media/CncseIzWgAA4ghH.jpg	1	chow
1707	817777686764523521	https://pbs.twimg.com/ext_tw_video_thumb/81777...	1	curly-coated_retriever
540	676975532580409345	https://pbs.twimg.com/media/CWUZpydWcAAeipD.jpg	1	malamute
1183	738537504001953792	https://pbs.twimg.com/media/Cj_P7rSUgAAYQbz.jpg	1	chow
1366	761672994376806400	https://pbs.twimg.com/ext_tw_video_thumb/76167...	1	gondola
25	666362758909284353	https://pbs.twimg.com/media/CT9lXGsUcAAyUFt.jpg	1	guinea_pig

1. The columns of image_predictions data set such as p1,p2,p3 have mixed lower and upper cases, while other rows have lower cases only.

In [16]:
```
# Visual Assesment of tweet json file dataset
mytweet_df.sample(10)
```

Out[16]:

	tweet_id	retweets	favorites	followers	friends
2162	668872652652679168	260	465	9375730	20

4	891327558926688256	7616	34332	9375692	20
1184	715009755312439296	1081	3732	9375712	20
657	789137962068021249	2564	8992	9375699	20
587	797545162159308800	4554	13494	9375699	20
122	867774946302451713	6156	29507	9375694	20
109	870656317836468226	2160	10588	9375694	20
821	766069199026450432	760	3846	9375701	20
1069	735991953473572864	1028	3241	9376118	20
2059	670782429121134593	645	1317	9375728	20

Programatic Assessment

In [17]:
```
mytweet_df.shape,df_i.shape,df_1.shape
```

Out[17]:
```
((2323, 5), (2075, 12), (2356, 17))
```

Programatic Assessment of twitter-archive-enhanced data set

In [18]:
```
df_1.info()
```

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

1. The data type for Timestamp column in twitter-archive-enhanced dataset appears as object instead of datetime dtype
2. Some columns have fewer rows than others implying there is missing values
3. The data type of tweet_id is integer and should be object

Checking the completeness of twitter-archive-enhanced dataset

In [19]:
```
def missings_(df_1):
    miss      = df_1.isnull().sum()
```

```
    miss_pct  = 100 * df_1.isnull().sum()/len(df_1)

    miss_pct      = pd.concat([miss,miss_pct], axis=1)
    missings_cols = miss_pct.rename(columns = {0:'Missings values', 1: 'Missing percenta
    missings_cols = missings_cols[missings_cols.iloc[:,1]!=0].sort_values('Missing perce

    return missings_cols

missings = missings_(df_1)
missings
```

Out[19]:

	Missings values	Missing percentage
in_reply_to_status_id	2278	96.69
in_reply_to_user_id	2278	96.69
retweeted_status_id	2175	92.32
retweeted_status_user_id	2175	92.32
retweeted_status_timestamp	2175	92.32
expanded_urls	59	2.50

- The features with extremely high missing values in are 'in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id','retweeted_status_user_id', 'retweeted_status_timestamp'. These columns have above 92% missing values.
- The column 'expanded_urls' has 2.50% missing values. This percentage is small.

Checking the Duplicates of twitter-archive-enhanced dataset

In [20]:
```
print('The number of duplicates in twitter-archive-enhanced dataset arae', sum(df_1.dupl
```

```
The number of duplicates in twitter-archive-enhanced dataset arae 0
```

twitter-archive-enhanced dataset does not have duplicates

Checking the Consistency of twitter-archive-enhanced dataset

In [21]:
```
df_1.nunique()
```

Out[21]:
```
tweet_id                      2356
in_reply_to_status_id           77
in_reply_to_user_id             31
timestamp                     2356
source                           4
text                          2356
retweeted_status_id            181
retweeted_status_user_id        25
retweeted_status_timestamp     181
expanded_urls                 2218
rating_numerator                40
rating_denominator              18
name                           957
doggo                            2
floofer                          2
pupper                           2
puppo                            2
dtype: int64
```

In [22]:
```
df_1.text.value_counts()
```

Out[22]:
```
This is Phineas. He's a mystical boy. Only ever appears in the hole of a donut. 13/10 ht
tps://t.co/MgUWQ76dJU                                                               1
Army of water dogs here. None of them know where they're going. Have no real purpose. Ag
gressive barks. 5/10 for all https://t.co/A88x73TwMN                                1
This is Louis. He's a rollercoaster of emotions. Incalculably fluffy. 12/10 would pet fi
rmly https://t.co/17RGvOZO9P                                                        1
With great pupper comes great responsibility. 12/10 https://t.co/hK6xB042EP
                                                                                    1
Meet Trooper &amp;amp; Maya. Trooper protects Maya from bad things like dognappers and Comca
st. So touching. 11/10 for both https://t.co/c98k1IoZKy                             1


                                                                                   ..
This is Tucker. He would like a hug. 13/10 someone hug him https://t.co/wdgY9oHPrT
                                                                                    1
This is Finley. She's a Beneboop Cumbersplash. 12/10 I'd do unspeakable things for Finle
y https://t.co/dS8SCbNF9P                                                           1
This is Sprinkles. He's trapped in light jail. 10/10 would post bail for him https://t.c
o/4s5Xlijogu                                                                        1
I WAS SENT THE ACTUAL DOG IN THE PROFILE PIC BY HIS OWNER THIS IS SO WILD. 14/10 ULTIMAT
E LEGEND STATUS https://t.co/7oQ1wpfxIH                                             1
Here we have a Japanese Irish Setter. Lost eye in Vietnam (?). Big fan of relaxing on st
air. 8/10 would pet https://t.co/BLDqew2Ijj                                         1
Name: text, Length: 2356, dtype: int64
```

- Some lines in text column are written in upper case only while others are mixed(upper and lower cases) in twitter-archive-enhanced data set
- Some rows in text columns contain special characters (Punctuations) in twitter-archive-enhanced data set
- The text column contain multiple variables such as texts, numbers and url links within each single row in twitter-archive-enhanced data set

In [23]:
```
df_1.doggo.value_counts()
```

Out[23]:
```
None     2259
doggo      97
Name: doggo, dtype: int64
```

In [24]:
```
df_1.floofer.value_counts()
```

Out[24]:
```
None       2346
floofer      10
Name: floofer, dtype: int64
```

In [25]:
```
df_1.pupper.value_counts()
```

Out[25]:
```
None      2099
pupper     257
Name: pupper, dtype: int64
```

In [26]:
```
df_1.puppo.value_counts()
```

Out[26]:
```
None     2326
puppo      30
Name: puppo, dtype: int64
```

- We can see the types of dogs such as 'doggo', 'floofer', 'pupper', 'puppo' have more 'None' values implying they have many missing values
- Since 'doggo', 'floofer', 'pupper', 'puppo' are types of dogs they should in a same column name and not separate columns

In [27]:
```
df_1.name.value_counts().head(20)
```

Out[27]:
```
None       745
a           55
Charlie     12
Cooper      11
Lucy        11
Oliver      11
Tucker      10
Penny       10
Lola        10
Winston      9
Bo           9
Sadie        8
the          8
Daisy        7
Buddy        7
Toby         7
an           7
Bailey       7
Leo          6
Oscar        6
Name: name, dtype: int64
```

- The name column contain some uncommon values (dog names) such as a , an which neeed to be investigated well

In [28]:
```
df_1.source.value_counts()
```

Out[28]:
```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>     2
221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>
11
Name: source, dtype: int64
```

The source column is a bit dirty with HTML format with a and \a tags surrounding the text (should be cleaned to be more readable)

In [29]:
```
df_1.timestamp.value_counts()
```

Out[29]:
```
2017-08-01 16:23:56 +0000    1
2016-01-13 02:43:46 +0000    1
2016-01-15 02:41:12 +0000    1
2016-01-15 02:08:05 +0000    1
2016-01-15 01:25:33 +0000    1
                            ..
2016-09-11 21:34:30 +0000    1
2016-09-10 23:54:11 +0000    1
2016-09-10 16:03:16 +0000    1
2016-09-09 18:31:54 +0000    1
2015-11-15 22:32:08 +0000    1
Name: timestamp, Length: 2356, dtype: int64
```

In [30]:
```
df_1.retweeted_status_timestamp.value_counts()
```

Out[30]:
```
2017-07-19 00:47:34 +0000    1
2015-11-28 03:31:48 +0000    1
2015-11-19 03:29:07 +0000    1
2015-11-16 04:02:55 +0000    1
2016-09-02 18:03:10 +0000    1
                            ..
```

```
2016-11-06 01:33:58 +0000    1
2016-10-06 15:49:14 +0000    1
2017-01-20 00:50:15 +0000    1
2017-01-20 17:00:46 +0000    1
2015-11-20 03:41:59 +0000    1
Name: retweeted_status_timestamp, Length: 181, dtype: int64
```

retweeted_status_timestamp column in twitter_archive dataset depicts that the there 181 retweets which may not be neccessary for analysing dogs images

Programatic Assessment of image_predictions data set

In [31]:
```
df_i.info()
```

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

The datatype tweet_id is integer (should be object)

In [32]:
```
# Checking duplicates
sum(df_i.duplicated())
```

Out[32]:
```
0
```

In [33]:
```
df_i.select_dtypes(include='object').nunique()
```

Out[33]:
```
jpg_url    2009
p1          378
p2          405
p3          408
dtype: int64
```

In [34]:
```
df_i.jpg_url.value_counts()
```

Out[34]:
```
https://pbs.twimg.com/media/CZhn-QAWwAASQan.jpg
   2
https://pbs.twimg.com/media/Cq9guJ5WgAADfpF.jpg
   2
https://pbs.twimg.com/ext_tw_video_thumb/807106774843039744/pu/img/8XZg1xW35Xp2J6JW.jpg
   2
https://pbs.twimg.com/media/CU1zsMSUAAAS0qW.jpg
   2
https://pbs.twimg.com/media/CsrjryzWgAAZY00.jpg
   2

  ..
https://pbs.twimg.com/media/CXrmMSpUwAAdeRj.jpg
```

```
   1
https://pbs.twimg.com/media/CXrawAhWkAAWSxC.jpg
   1
https://pbs.twimg.com/media/CXrIntsUsAEkv0d.jpg
   1
https://pbs.twimg.com/media/CXqcOHCUQAAugTB.jpg
   1
https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg
   1
Name: jpg_url, Length: 2009, dtype: int64
```

In [35]:
```
df_i.p1.value_counts().head(10)
```

Out[35]:
```
golden_retriever      150
Labrador_retriever    100
Pembroke               89
Chihuahua              83
pug                    57
chow                   44
Samoyed                43
toy_poodle             39
Pomeranian             38
cocker_spaniel         30
Name: p1, dtype: int64
```

In [36]:
```
df_i.p2.value_counts().head(10)
```

Out[36]:
```
Labrador_retriever          104
golden_retriever             92
Cardigan                     73
Chihuahua                    44
Pomeranian                   42
Chesapeake_Bay_retriever     41
French_bulldog               41
toy_poodle                   37
cocker_spaniel               34
miniature_poodle             33
Name: p2, dtype: int64
```

In [37]:
```
df_i.p3.value_counts().head(10)
```

Out[37]:
```
Labrador_retriever           79
Chihuahua                    58
golden_retriever             48
Eskimo_dog                   38
kelpie                       35
kuvasz                       34
Staffordshire_bullterrier    32
chow                         32
beagle                       31
cocker_spaniel               31
Name: p3, dtype: int64
```

There is no duplicates in the image_predictions dataset

Tidiness

Checking common columns between image_predictions twitter_archive dataset

In [38]:
```
res = df_1.columns.intersection(df_i.columns)
res
```

Out[38]:
```
Index(['tweet_id'], dtype='object')
```

Since the image_predictions dataset has common column (tweet_id) with twitter_archive dataset thus the two tables need to be merged

Programatic Assessment of tweet json file dataset

In [39]:
```
mytweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2323 entries, 0 to 2322
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   tweet_id    2323 non-null   object
 1   retweets    2323 non-null   int64 
 2   favorites   2323 non-null   int64 
 3   followers   2323 non-null   int64 
 4   friends     2323 non-null   int64 
dtypes: int64(4), object(1)
memory usage: 90.9+ KB
```

tweet json file data has tweet_id column which is common in the other tables, thus there is need to merge with other tables

In [40]:
```
# Checking duplicates
sum(mytweet_df.duplicated())
```

Out[40]:
```
0
```

Quality Issues

1. There are features with extremely high missing values in are 'in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id','retweeted_status_user_id', 'retweeted_status_timestamp'. These columns have above 92% missing values twitter-archive-enhanced data set.

- Also The column 'expanded_urls' has 2.50% missing values in twitter-archive-enhanced data set

1. we can also see that the name, 'doggo', 'floofer', 'pupper', 'puppo' columns in twitter-archive-enhanced data set have object name 'None' (this is could be missing value which have been replace with a value None) thus they will not appear as missing but as object.

2. We can see text column has mixed lower and upper case, while other rows have upper case only and others lower case only in the twitter-archive-enhanced data set.

1. The data type for Timestamp column in twitter-archive-enhanced dataset appears as object instead of datetime dtype

2. The text column contain multiple variables such as html, url links and numbers within each (text) single row in twitter-archive-enhanced data set

3. The text column contain some white spaces of twitter-archive-enhanced data set.

4. The data type of tweet_id is integer and should be object in twitter-archive-enhanced dataset.

5. The name column contain some uncommon values (dog names) such as a , an which neeed to be investigated well(since they might be parts of strings from elsewhere) in twitter-archive-enhanced

dataset

6. The source column is a bit dirty with HTML format with a and \a tags surrounding the text (The column looks redundant)
7. retweeted_status_timestamp column in twitter_archive dataset depicts that the there 181 retweets which may not be neccessary for analysing dogs images

Tidiness issues

1. Since 'doggo', 'floofer', 'pupper', 'puppo' columns are dogs stage they should in a same column name (one variable) and not separate columns
2. Since the image_predictions dataset has common column (tweet_id) with twitter_archive dataset thus the two tables need to be merged

Cleaning Data

In this section, clean **all** of the issues you documented while assessing.

Note: Make a copy of the original data before cleaning. Cleaning includes merging individual pieces of data according to the rules of tidy data. The result should be a high-quality and tidy master pandas DataFrame (or DataFrames, if appropriate).

In [161…
```
# Make copies of original pieces of data
df_archive_clean=df_1.copy()
df_image_clean=df_i.copy()
jtweet_clean=mytweet_df.copy()
```

In [162…
```
#Check the number of columns before
df_archive_clean.shape
```

Out[162]:
```
(2356, 17)
```

In [ ]:
```

```

In [163…
```
# Test
df_archive_clean.shape
```

Out[163]:
```
(2356, 17)
```

The columns with large proprortion of missing values have been dropped,in addition some of these columns are redundant.

Issue #1: we can also see that the name, 'doggo', 'floofer', 'pupper', 'puppo' columns in twitter-archive-enhanced data set have object name 'None' (this is could be missing value which have been replace with a value None) thus they will not appear as missing but as object.

- These values should be be replaced and np.nan as the replacement value

In [164…
```
df_archive_clean = df_archive_clean.replace('None', value=np.nan)

# Test
df_archive_clean.nunique()
```

Out[164]:
```
tweet_id                    2356
in_reply_to_status_id         77
in_reply_to_user_id           31
timestamp                   2356
source                         4
text                        2356
retweeted_status_id          181
retweeted_status_user_id      25
retweeted_status_timestamp   181
expanded_urls               2218
rating_numerator              40
rating_denominator            18
name                         956
doggo                          1
floofer                        1
pupper                         1
puppo                          1
dtype: int64
```

We can see dog stages have one unique values since they were initially two, meaning None values have been replaced and NaN , thus missing values are now clear

In [166…
```
#Test name column
df_archive_clean.name.value_counts()
```

Out[166]:
```
a             55
Charlie       12
Oliver        11
Cooper        11
Lucy          11
              ..
Aqua           1
Chase          1
Meatball       1
Rorie          1
Christoper     1
Name: name, Length: 956, dtype: int64
```

We no longer see None value, meaning None values have been replaced and NaN , thus missing values are now clear

Issue #2: We can see text column has mixed lower and upper case, while other rows have upper case only and others lower case only in the twitter-archive-enhanced data set.

In [167…
```
df_archive_clean=df_archive_clean.applymap(lambda x: x.lower() if type(x)==str else x)
df_archive_clean.dtypes
```

Out[167]:
```
tweet_id                     int64
in_reply_to_status_id      float64
in_reply_to_user_id        float64
timestamp                   object
source                      object
text                        object
retweeted_status_id        float64
retweeted_status_user_id   float64
retweeted_status_timestamp  object
expanded_urls               object
rating_numerator             int64
rating_denominator           int64
name                        object
doggo                       object
```

```
floofer                         object
pupper                          object
puppo                           object
dtype: object
```

In [168…
```
df_archive_clean.head(3)
```

Out[168]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.com/dow
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/dow
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/dow

All strings of twitter-archive-enhanced dataset have been converted to lower cases to enhance uniformity and consistency

Issue #3: The data type for Timestamp column in twitter-archive-enhanced dataset appears as object instead of datetime dtype

In [169…
```
df_archive_clean.dtypes
```

Out[169]:
```
tweet_id                         int64
in_reply_to_status_id          float64
in_reply_to_user_id            float64
timestamp                       object
source                          object
text                            object
retweeted_status_id            float64
retweeted_status_user_id       float64
retweeted_status_timestamp      object
expanded_urls                   object
rating_numerator                 int64
rating_denominator               int64
name                            object
doggo                           object
floofer                         object
pupper                          object
puppo                           object
dtype: object
```

Code

In [170…
```
df_archive_clean['timestamp']=pd.DatetimeIndex(df_archive_clean['timestamp'])
df_archive_clean.dtypes
```

Out[170]:
```
tweet_id                                 int64
in_reply_to_status_id                  float64
in_reply_to_user_id                    float64
```

```
timestamp                       datetime64[ns, UTC]
source                                       object
text                                         object
retweeted_status_id                         float64
retweeted_status_user_id                    float64
retweeted_status_timestamp                   object
expanded_urls                                object
rating_numerator                              int64
rating_denominator                            int64
name                                         object
doggo                                        object
floofer                                      object
pupper                                       object
puppo                                        object
dtype: object
```

The datetime dtype has been parsed on timestamp which was an object

Issue #4: The text column contain multiple variables such as html, url links within each single row in twitter-archive-enhanced data set

define We can first remove (https) ending url links using replace and regex functions in clean twitter-archive-enhanced data We can then convert the html ampersand code ("&amp ;" to "&") using replace and regex functions in the text column of clean twitter-archive-enhanced data Finally the newline symbols can be removed the "/n " using replace and regex functions

In [171…
```
#Removing http url links
df_archive_clean['text'] = df_archive_clean['text'].str.replace(r"http\S+", "", regex=Tr
```

In [172…
```
#converting the html ampersand code to &
df_archive_clean['text'] = df_archive_clean['text'].str.replace("&amp;", "&", regex=True
```

In [173…
```
# Removing new lines and replacing with a space
df_archive_clean['text'] = df_archive_clean['text'].str.replace("\n", " ", regex=True)
```

In [174…
```
#Test
df_archive_clean['text'].value_counts()
```

Out[174]:
```
this is phineas. he's a mystical boy. only ever appears in the hole of a donut. 13/10
                                                                                  1
army of water dogs here. none of them know where they're going. have no real purpose. a
ggressive barks. 5/10 for all                                                     1
this is louis. he's a rollercoaster of emotions. incalculably fluffy. 12/10 would pet f
irmly                                                                             1
with great pupper comes great responsibility. 12/10
                                                                                  1
meet trooper & maya. trooper protects maya from bad things like dognappers and comcast.
so touching. 11/10 for both                                                       1

                                                                                 ..
this is tucker. he would like a hug. 13/10 someone hug him
                                                                                  1
this is finley. she's a beneboop cumbersplash. 12/10 i'd do unspeakable things for finl
ey                                                                                1
this is sprinkles. he's trapped in light jail. 10/10 would post bail for him
                                                                                  1
i was sent the actual dog in the profile pic by his owner this is so wild. 14/10 ultima
te legend status                                                                  1
here we have a japanese irish setter. lost eye in vietnam (?). big fan of relaxing on s
tair. 8/10 would pet                                                              1
Name: text, Length: 2356, dtype: int64
```

All the ending url links were removed the html ampersand code was replaced with & and the newline symbols can be removed using replace and regex functions in text column of cleaned twitter-archive-enhanced data.

Issue #5: The text column contain some white spaces of twitter-archive-enhanced data set.

define

The leading and trailing white spaces from a string can be removed using strip() function

In [175…
```
df_archive_clean['text'] = df_archive_clean['text'].str.strip()
#Test
df_archive_clean['text'].value_counts()
```

Out[175]:
```
this is phineas. he's a mystical boy. only ever appears in the hole of a donut. 13/10
                                                                                    1
army of water dogs here. none of them know where they're going. have no real purpose. a
ggressive barks. 5/10 for all                                                       1
this is louis. he's a rollercoaster of emotions. incalculably fluffy. 12/10 would pet f
irmly                                                                               1
with great pupper comes great responsibility. 12/10
                                                                                    1
meet trooper & maya. trooper protects maya from bad things like dognappers and comcast.
so touching. 11/10 for both                                                         1

                                                                                   ..
this is tucker. he would like a hug. 13/10 someone hug him
                                                                                    1
this is finley. she's a beneboop cumbersplash. 12/10 i'd do unspeakable things for finl
ey                                                                                  1
this is sprinkles. he's trapped in light jail. 10/10 would post bail for him
                                                                                    1
i was sent the actual dog in the profile pic by his owner this is so wild. 14/10 ultima
te legend status                                                                    1
here we have a japanese irish setter. lost eye in vietnam (?). big fan of relaxing on s
tair. 8/10 would pet                                                                1
Name: text, Length: 2356, dtype: int64
```

All the eading and trailing white spaces from a string have been removed

Issue #6: The data type of tweet_id is integer and should be object in twitter-archive-enhanced and image prediction dataset.

Define

This can be converted to strings using astype("str")

Code

In [176…
```
#Code
df_archive_clean['tweet_id']=df_archive_clean['tweet_id'].astype('str')
df_image_clean['tweet_id']=df_image_clean['tweet_id'].astype('str')
```

In [177…
```
#Assess
print(df_archive_clean.dtypes)
df_image_clean.dtypes
```

```
tweet_id                                    object
```

```
in_reply_to_status_id                      float64
in_reply_to_user_id                        float64
timestamp                    datetime64[ns, UTC]
source                                      object
text                                        object
retweeted_status_id                        float64
retweeted_status_user_id                   float64
retweeted_status_timestamp                  object
expanded_urls                               object
rating_numerator                             int64
rating_denominator                           int64
name                                        object
doggo                                       object
floofer                                     object
pupper                                      object
puppo                                       object
dtype: object
```

Out[177]:

```
tweet_id     object
jpg_url      object
img_num       int64
p1           object
p1_conf     float64
p1_dog         bool
p2           object
p2_conf     float64
p2_dog         bool
p3           object
p3_conf     float64
p3_dog         bool
dtype: object
```

We can see that the tweet_id column has been converted to string in cleaned twitter-archive-enhanced and cleaned image prediction dataset (both tables).

Issue #7: The name column contain some uncommon values (dog names) such as a , an which need to be investigated well(since they might be parts of strings from elsewhere) in twitter-archive-enhanced dataset

Define First we can check if there are other more weird dog We need to check the ideal name by looking at the text

In [178…

```
print(df_archive_clean.name.value_counts().tail(60))
df_archive_clean.name.value_counts().head(60)
```

```
theo            1
rumpole         1
fido            1
emma            1
spencer         1
lilli           1
boston          1
brandonald      1
corey           1
leonard         1
beckham         1
devón           1
gert            1
einstein        1
arya            1
marlee          1
dex             1
bookstore       1
jordy           1
```

```
coleman         1
bayley          1
remy            1
chadrick        1
kellogg         1
buckley         1
livvie          1
shikha          1
hermione        1
ralpher         1
aldrick         1
this            1
unacceptable    1
rooney          1
ziva            1
stefan          1
pupcasso        1
puff            1
flurpson        1
storkson        1
lili            1
burt            1
simba           1
shiloh          1
gustav          1
arlen           1
lenox           1
jersey          1
harvey          1
blanket         1
zooey           1
geno            1
stark           1
beya            1
kayla           1
edmund          1
aqua            1
chase           1
meatball        1
rorie           1
christoper      1
Name: name, dtype: int64
```

Out[178]:

```
a          55
charlie    12
oliver     11
cooper     11
lucy       11
lola       10
tucker     10
penny      10
bo          9
winston     9
the         8
sadie       8
daisy       7
buddy       7
bailey      7
toby        7
an          7
bella       6
jack        6
oscar       6
rusty       6
stanley     6
scout       6
jax         6
```

```
leo          6
milo         6
koda         6
dave         6
oakley       5
larry        5
sunny        5
louis        5
chester      5
george       5
alfie        5
finn         5
very         5
sammy        5
gus          5
phil         5
bentley      5
sampson      4
reginald     4
quite        4
boomer       4
reggie       4
loki         4
dexter       4
duke         4
luna         4
carl         4
ruby         4
brody        4
sophie       4
clark        4
just         4
riley        4
jerry        4
maggie       4
chip         4
Name: name, dtype: int64
```

We can see there are more weird dog names such as 'a', 'an', 'this', 'unacceptable', 'very','quite'

In [179…

```
import  warnings
warnings.filterwarnings('ignore')
nan_dogs= [ 'a', 'an', 'this', 'unacceptable', 'very','quite' ]

for doggy in nan_dogs:
    print(df_archive_clean.text[df_archive_clean['name'] == doggy])
```

```
56      here is a pupper approaching maximum borkdrive...
649     here is a perfect example of someone who has t...
801     guys this is getting so out of hand. we only r...
1002    this is a mighty rare blue-tailed hammer sherk...
1004    viewer discretion is advised. this is a terrib...
1017    this is a carrot. we only rate dogs. please on...
1049    this is a very rare great alaskan bush pupper....
1193    people please. this is a deadly mediterranean ...
1207    this is a taco. we only rate dogs. please only...
1340    here is a heartbreaking scene of an incredible...
1351    here is a whole flock of puppers.  60/50 i'll ...
1361    this is a butternut cumberfloof. it's not wind...
1368    this is a wild tuscan poofwiggle. careful not ...
1382    "pupper is a present to world. here is a bow f...
1499    this is a rare arctic wubberfloof. unamused by...
1737    guys this really needs to stop. we've been ove...
1785    this is a dog swinging. i really enjoyed it so...
1853    this is a sizzlin menorah spaniel from brookly...
```

```
1854    seriously guys?! only send in dogs. i only rat...
1877    c'mon guys. we've been over this. we only rate...
1878    this is a fluffy albino bacardi columbia mix. ...
1923    this is a sagitariot baklava mix. loves her ne...
1941    this is a heavily opinionated dog. loves walls...
1955    this is a lofted aphrodisiac terrier named kip...
1994    this is a baby rand paul. curls for days. 11/1...
2034    this is a tuscaloosa alcatraz named jacob (yac...
2066    this is a helvetica listerine named rufus. thi...
2116    this is a deciduous trimester mix named spork....
2125    this is a rich mahogany seltzer named cherokee...
2128    this is a speckled cauliflower yosemite named ...
2146    this is a spotted lipitor rumpelstiltskin name...
2153    this is a brave dog. excellent free climber. t...
2161    this is a coriander baton rouge named alfredo....
2191    this is a slovakian helter skelter feta named ...
2198    this is a wild toblerone from papua new guinea...
2211    here is a horned dog. much grace. can jump ove...
2218    this is a birmingham quagmire named chuk. love...
2222    here is a mother dog caring for her pups. snaz...
2235    this is a trans siberian kellogg named alfonso...
2249    this is a shotokon macadamia mix named cheryl....
2255    this is a rare hungarian pinot named jessiga. ...
2264    this is a southwest coriander named klint. hat...
2273    this is a northern wahoo named kohl. he runs t...
2287    this is a dasani kingfisher from maine. his na...
2304    this is a curly ticonderoga named pepe. no fee...
2311    this is a purebred bacardi named octaviath. ca...
2314    this is a golden buckminsterfullerene named jo...
2327    this is a southern vesuvius bumblegruff. can d...
2334    this is a funny dog. weird toes. won't come do...
2347    my oh my. this is a rare blond canadian terrie...
2348    here is a siberian heavily armored polar bear ...
2350    this is a truly beautiful english wilson staff...
2352    this is a purebred piers morgan. loves to netf...
2353    here is a very happy pup. big fan of well-main...
2354    this is a western brown mitsubishi terrier. up...
Name: text, dtype: object
759     rt @dog_rates: this is an east african chalupa...
1025    this is an iraqi speed kangaroo. it is not a d...
1362    this is an east african chalupa seal. we only ...
2204    this is an irish rigatoni terrier named berta....
2333    this is an extremely rare horned parthenon. no...
2335    this is an albanian 3 1/2 legged  episcopalian...
2349    this is an odd dog. hard on the outside but lo...
Name: text, dtype: object
1120    say hello to this unbelievably well behaved sq...
Name: text, dtype: object
1121    we only rate dogs. pls stop sending non-canine...
Name: text, dtype: object
773     rt @dog_rates: we only rate dogs. pls stop sen...
819     we only rate dogs. pls stop sending in non-can...
1031    we only rate dogs. pls stop sending in non-can...
1097    we only rate dogs. please stop sending in non-...
1385    we only rate dogs. pls stop sending in non-can...
Name: text, dtype: object
118     rt @dog_rates: we only rate dogs. this is quit...
169     we only rate dogs. this is quite clearly a smo...
193     guys, we only rate dogs. this is quite clearly...
2326    this is quite the dog. gets really excited whe...
Name: text, dtype: object
```

Code

In [180…

```
for doggy in nan_dogs:
```

```
    df_archive_clean.name[df_archive_clean['name'] == doggy] = 'None'
```

In [181…
```
for doggy in nan_dogs:
    print(df_archive_clean.text[df_archive_clean['name'] == doggy])
```

```
Series([], Name: text, dtype: object)
Series([], Name: text, dtype: object)
Series([], Name: text, dtype: object)
Series([], Name: text, dtype: object)
Series([], Name: text, dtype: object)
Series([], Name: text, dtype: object)
```

In [182…
```
#test
df_archive_clean.name.value_counts()
```

Out[182]:
```
None          73
charlie       12
oliver        11
cooper        11
lucy          11
              ..
edmund         1
aqua           1
chase          1
meatball       1
christoper     1
Name: name, Length: 951, dtype: int64
```

We can see that some of the weird dog names have been replaced with assigned value

Issue #8: The source column is a bit dirty with HTML format with a and \a tags surrounding the text (The column looks redundant)

Define

The source column is a bit messy and may not necessary for our annalysis and should be dropped using drop() function

Code

In [183…
```
df_archive_clean=df_archive_clean.drop('source',axis=1)
df_archive_clean.columns
```

Out[183]:
```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestamp',
       'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
      dtype='object')
```

We can no longer see the source column on df_archive_clean since it has been dropped

Issue #9: retweeted_status_timestamp column in twitter_archive dataset. There are 2323 retweets in twitter data which may not be neccessary for analysing dogs images

Define Drop retweets in jtweet_clean table retweeted_status_timestamp column is not necessary and should be dropped using drop() function, however had been taken care of since it had been dropped together with other features with missing values.

In [184…
```
len(jtweet_clean.retweets)
```

Out[184]:
```
2323
```

Removing Retweet columns

We need to remove the rows. These rows can be easily identified by looking at the values in retweeted_status_id column. If a value is present, then it's a retweet, else, it's original.

So first remove the rows and then the columns will be empty and can be deleted.

In [185…
```
df_archive_clean = df_archive_clean[pd.isnull(df_archive_clean.retweeted_status_id)]
df_archive_clean = df_archive_clean[pd.isnull(df_archive_clean.in_reply_to_status_id)]
```

In [187…
```
df_archive_clean.shape
```

Out[187]:
```
(2097, 16)
```

In [186…
```
#test
df_archive_clean[df_archive_clean.text=="@dog_rates"]
```

Out[186]:

tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	text	retweeted_status_id	retweeted_s

Issue #11: There are features with extremely high missing values in are 'in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id','retweeted_status_user_id', 'retweeted_status_timestamp'. These columns have above 92% missing values twitter-archive-enhanced data set.

- We shall drop the missing values with huge percentage of missing values those above 92%
- Also The column 'expanded_urls' has 2.50% missing values in twitter-archive-enhanced data this will be noted on limitation part since it will difficult to decide on imputing urls, dropping may also lead to data loss

In [188…
```
df_archive_clean.drop(['retweeted_status_id','retweeted_status_user_id',
                       'retweeted_status_timestamp','in_reply_to_user_id',
                       'in_reply_to_status_id'], axis=1, inplace=True)
```

In [189…
```
jtweet_clean.drop('retweets', axis=1, inplace=True)
```

In [190…
```
df_archive_clean.columns
```

Out[190]:
```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
      dtype='object')
```

We can no longer see the source column on df_archive_clean since it has been dropped

Tidiness issues

1. Since the image_predictions dataset has common column (tweet_id) with twitter_archive dataset thus the two tables need to be merged also the twitter json data has tweet_id. Generally the table should be maerged into one.

2. Since 'doggo', 'floofer', 'pupper', 'puppo' columns are dogs stage they should in a same column name (one variable) and not separate columns

Meging tidiness:

In [191…
```
df_archive_clean.columns
```

Out[191]:
```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
      dtype='object')
```

In [192…
```
df_image_clean.columns
```

Out[192]:
```
Index(['tweet_id', 'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2',
       'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
      dtype='object')
```

In [193…
```
jtweet_clean.columns
```

Out[193]:
```
Index(['tweet_id', 'favorites', 'followers', 'friends'], dtype='object')
```

In [ ]:
```
```

Tidiness issue #1: Since the image_predictions dataset has common column (tweet_id) with twitter_archive dataset thus the two tables need to be merged also the twitter json data has tweet_id. Generally the table should be maerged into one.

Define We shall merge first df_archive_clean and df_image_clean tables using merge() function from pandas then assign it as merge_twitter. Then and merge again with jtweet_clean into one table using the merge function and assign it as tweet_master using tweet_id which is common identifier.

Code

In [194…
```
#Merging df_archive_clean and df_image_clean
merge_twitter = pd.merge(df_archive_clean, df_image_clean,on = 'tweet_id')
merge_twitter.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1971 entries, 0 to 1970
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1971 non-null   object
 1   timestamp           1971 non-null   datetime64[ns, UTC]
 2   text                1971 non-null   object
 3   expanded_urls       1971 non-null   object
 4   rating_numerator    1971 non-null   int64
 5   rating_denominator  1971 non-null   int64
 6   name                1447 non-null   object
 7   doggo               73 non-null     object
 8   floofer             8 non-null      object
 9   pupper              209 non-null    object
 10  puppo               23 non-null     object
 11  jpg_url             1971 non-null   object
 12  img_num             1971 non-null   int64
 13  p1                  1971 non-null   object
 14  p1_conf             1971 non-null   float64
 15  p1_dog              1971 non-null   bool
```

```
 16  p2                  1971 non-null   object
 17  p2_conf             1971 non-null   float64
 18  p2_dog              1971 non-null   bool
 19  p3                  1971 non-null   object
 20  p3_conf             1971 non-null   float64
 21  p3_dog              1971 non-null   bool
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(3), object(12)
memory usage: 313.7+ KB
```

In [195…]
```
#Merging merge_twitter and jtweet_clean
tweet_master = pd.merge(merge_twitter, jtweet_clean,on = 'tweet_id')
tweet_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1961 entries, 0 to 1960
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            1961 non-null   object
 1   timestamp           1961 non-null   datetime64[ns, UTC]
 2   text                1961 non-null   object
 3   expanded_urls       1961 non-null   object
 4   rating_numerator    1961 non-null   int64
 5   rating_denominator  1961 non-null   int64
 6   name                1440 non-null   object
 7   doggo               72 non-null     object
 8   floofer             8 non-null      object
 9   pupper              208 non-null    object
 10  puppo               23 non-null     object
 11  jpg_url             1961 non-null   object
 12  img_num             1961 non-null   int64
 13  p1                  1961 non-null   object
 14  p1_conf             1961 non-null   float64
 15  p1_dog              1961 non-null   bool
 16  p2                  1961 non-null   object
 17  p2_conf             1961 non-null   float64
 18  p2_dog              1961 non-null   bool
 19  p3                  1961 non-null   object
 20  p3_conf             1961 non-null   float64
 21  p3_dog              1961 non-null   bool
 22  favorites           1961 non-null   int64
 23  followers           1961 non-null   int64
 24  friends             1961 non-null   int64
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(6), object(12)
memory usage: 358.1+ KB
```

In [196…]
```
tweet_master.columns
```

Out[196]:
```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'favorites', 'followers',
       'friends'],
      dtype='object')
```

In [197…]
```
tweet_master.nunique()
```

Out[197]:
```
tweet_id              1961
timestamp             1961
text                  1961
expanded_urls         1961
rating_numerator        33
rating_denominator      14
name                   927
doggo                    1
floofer                  1
```

```
pupper                  1
puppo                   1
jpg_url              1961
img_num                 4
p1                    373
p1_conf              1958
p1_dog                  2
p2                    396
p2_conf              1956
p2_dog                  2
p3                    402
p3_conf              1959
p3_dog                  2
favorites            1805
followers              61
friends                 1
dtype: int64
```

In [198…
```
tweet_master.columns
```

Out[198]:
```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'favorites', 'followers',
       'friends'],
      dtype='object')
```

Tidiness issue #1: Since 'doggo', 'floofer', 'pupper', 'puppo' columns are stages of dogs they should in a same column name (one variable) and not separate columns

Define Melt the 'doggo', 'floofer', 'pupper', 'puppo' columns to a *dog_stage* column. Drop the Immediate *dn_stages* column.

In [199…
```
t_clean = pd.melt(tweet_master, id_vars=['tweet_id', 'timestamp', 'text', 'expanded_urls
       'rating_denominator', 'name',
       'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2', 'p2_conf',
       'p2_dog', 'p3', 'p3_conf', 'p3_dog', 'favorites',
       'followers', 'friends'], value_name='dog_stage')
darchive_clean=t_clean.drop('variable', axis=1)

darchive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7844 entries, 0 to 7843
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            7844 non-null   object
 1   timestamp           7844 non-null   datetime64[ns, UTC]
 2   text                7844 non-null   object
 3   expanded_urls       7844 non-null   object
 4   rating_numerator    7844 non-null   int64
 5   rating_denominator  7844 non-null   int64
 6   name                5760 non-null   object
 7   jpg_url             7844 non-null   object
 8   img_num             7844 non-null   int64
 9   p1                  7844 non-null   object
 10  p1_conf             7844 non-null   float64
 11  p1_dog              7844 non-null   bool
 12  p2                  7844 non-null   object
 13  p2_conf             7844 non-null   float64
 14  p2_dog              7844 non-null   bool
 15  p3                  7844 non-null   object
```

```
 16  p3_conf             7844 non-null   float64
 17  p3_dog              7844 non-null   bool
 18  favorites           7844 non-null   int64
 19  followers           7844 non-null   int64
 20  friends             7844 non-null   int64
 21  dog_stage           311 non-null    object
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(6), object(9)
memory usage: 1.2+ MB
```

In [200…
```
darchive_clean.nunique()
```

Out[200]:
```
tweet_id              1961
timestamp             1961
text                  1961
expanded_urls         1961
rating_numerator        33
rating_denominator      14
name                   927
jpg_url               1961
img_num                  4
p1                     373
p1_conf               1958
p1_dog                   2
p2                     396
p2_conf               1956
p2_dog                   2
p3                     402
p3_conf               1959
p3_dog                   2
favorites             1805
followers               61
friends                  1
dog_stage                4
dtype: int64
```

We can now replace null values with None for both name and dog_stage column

In [201…
```
darchive_clean=darchive_clean.fillna('None')
darchive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7844 entries, 0 to 7843
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            7844 non-null   object
 1   timestamp           7844 non-null   datetime64[ns, UTC]
 2   text                7844 non-null   object
 3   expanded_urls       7844 non-null   object
 4   rating_numerator    7844 non-null   int64
 5   rating_denominator  7844 non-null   int64
 6   name                7844 non-null   object
 7   jpg_url             7844 non-null   object
 8   img_num             7844 non-null   int64
 9   p1                  7844 non-null   object
 10  p1_conf             7844 non-null   float64
 11  p1_dog              7844 non-null   bool
 12  p2                  7844 non-null   object
 13  p2_conf             7844 non-null   float64
 14  p2_dog              7844 non-null   bool
 15  p3                  7844 non-null   object
 16  p3_conf             7844 non-null   float64
 17  p3_dog              7844 non-null   bool
 18  favorites           7844 non-null   int64
```

```
 19  followers           7844 non-null   int64
 20  friends             7844 non-null   int64
 21  dog_stage           7844 non-null   object
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(6), object(9)
memory usage: 1.2+ MB
```

In [202…]
```
darchive_clean.nunique()
```

Out[202]:
```
tweet_id              1961
timestamp             1961
text                  1961
expanded_urls         1961
rating_numerator        33
rating_denominator      14
name                   927
jpg_url               1961
img_num                  4
p1                     373
p1_conf               1958
p1_dog                   2
p2                     396
p2_conf               1956
p2_dog                   2
p3                     402
p3_conf               1959
p3_dog                   2
favorites             1805
followers               61
friends                  1
dog_stage                5
dtype: int64
```

We can see that number of unique values for name column has not changed while dog stages has increased by one after replacing Null values with None

Checking and Handling the duplicates

In [203…]
```
print('There are', sum(darchive_clean.duplicated()), 'duplicates in the merged data')
```

```
There are 5572 duplicates in the merged data
```

In [204…]
```
darchive_clean.shape
```

Out[204]:
```
(7844, 22)
```

Code

In [205…]
```
#drop the duplicates
darchive_clean_master=darchive_clean.drop_duplicates()
darchive_clean_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2272 entries, 0 to 6683
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   tweet_id            2272 non-null   object
 1   timestamp           2272 non-null   datetime64[ns, UTC]
 2   text                2272 non-null   object
 3   expanded_urls       2272 non-null   object
 4   rating_numerator    2272 non-null   int64
 5   rating_denominator  2272 non-null   int64
 6   name                2272 non-null   object
```

```
 7   jpg_url             2272 non-null   object
 8   img_num             2272 non-null   int64
 9   p1                  2272 non-null   object
 10  p1_conf             2272 non-null   float64
 11  p1_dog              2272 non-null   bool
 12  p2                  2272 non-null   object
 13  p2_conf             2272 non-null   float64
 14  p2_dog              2272 non-null   bool
 15  p3                  2272 non-null   object
 16  p3_conf             2272 non-null   float64
 17  p3_dog              2272 non-null   bool
 18  favorites           2272 non-null   int64
 19  followers           2272 non-null   int64
 20  friends             2272 non-null   int64
 21  dog_stage           2272 non-null   object
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(6), object(9)
memory usage: 361.7+ KB
```

We can see that 5836 duplicates have been removed

Test

In [206…
```
# Checking if there is any duplicates in cleaned master data
sum(darchive_clean_master.duplicated())
```

Out[206]:
```
0
```

Droping columns with one unique values since they are not useful for analysis

In [207…
```
darchive_clean_master.loc[:,darchive_clean_master.nunique()==1].columns
```

Out[207]:
```
Index(['friends'], dtype='object')
```

In [208…
```
darchive_clean_master.drop('friends', axis=1, inplace=True)
```

friends column has been droppped since it has only one unique value and may not be relevant for our analysis

Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".

The data is now cleaned and can be stored in "twitter_archive_master.csv" as advised

In [220…
```
#Code
darchive_clean_master.to_csv('twitter_archive_master.csv',index=False)
```

Analyzing and Visualizing Data

In this section, analyze and visualize your wrangled data. You must produce at least **three (3) insights and one (1) visualization.**

In [210…
```
from dataprep.eda import *
```

In [211…
```
darchive_clean_master.columns
```

Out[211]:
```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'jpg_url', 'img_num', 'p1', 'p1_conf',
       'p1_dog', 'p2', 'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog',
       'favorites', 'followers', 'dog_stage'],
      dtype='object')
```

In [212…
```
plot(darchive_clean_master,'text')
```

```
  0%|                                                                | 0/76 [00:00<?, ?it/s]
```

Out[212]:

Stats | Bar Chart | Pie Chart | Word Cloud | Word Frequency | Word Length | Value Table

Overview

Approximate Distinct Count	1961
Approximate Unique (%)	86.3%
Missing	0
Missing (%)	0.0%
Memory Size	378206

Sample

1st row	this is phineas. h...
2nd row	this is tilly. she...
3rd row	this is archie. he...
4th row	this is darla. she...
5th row	this is franklin. ...

Length

Mean	98.1026
Standard Deviation	25.376
Median	108
Minimum	12
Maximum	140

Letter

Count	164907
Lowercase Letter	164907
Space Separator	37704
Uppercase Letter	0
Dash Punctuation	89
Decimal Number	8932

In [213…
```
from wordcloud import WordCloud
```

In [214…
```
# Select the text column
text = darchive_clean_master.text.tolist()

# Join the text elements into a single string
text = " ".join(text)

# Create a WordCloud object
wordcloud = WordCloud().generate(text)

# Display the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

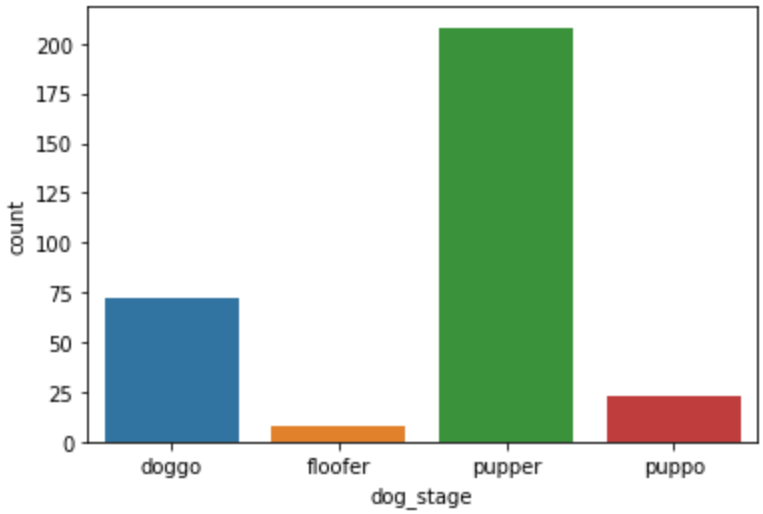
The word Cloud shows the most mentioned words are pupper,doggo, dog, rates, dog_rates, good. This means most of the users,followers are rating different stages of dogs. Also pupper seemed to be most mentioned.

In [215…
```
darchive_clean_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2272 entries, 0 to 6683
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            2272 non-null   object             
 1   timestamp           2272 non-null   datetime64[ns, UTC]
 2   text                2272 non-null   object             
 3   expanded_urls       2272 non-null   object             
 4   rating_numerator    2272 non-null   int64              
 5   rating_denominator  2272 non-null   int64              
 6   name                2272 non-null   object             
 7   jpg_url             2272 non-null   object             
 8   img_num             2272 non-null   int64              
 9   p1                  2272 non-null   object             
 10  p1_conf             2272 non-null   float64            
 11  p1_dog              2272 non-null   bool               
 12  p2                  2272 non-null   object             
 13  p2_conf             2272 non-null   float64            
 14  p2_dog              2272 non-null   bool               
 15  p3                  2272 non-null   object             
 16  p3_conf             2272 non-null   float64            
 17  p3_dog              2272 non-null   bool               
 18  favorites           2272 non-null   int64              
 19  followers           2272 non-null   int64              
 20  dog_stage           2272 non-null   object             
dtypes: bool(3), datetime64[ns, UTC](1), float64(3), int64(5), object(9)
memory usage: 343.9+ KB
```

In [216…
```
### Checking distribution of dog stage using bar graph (countplot)
```

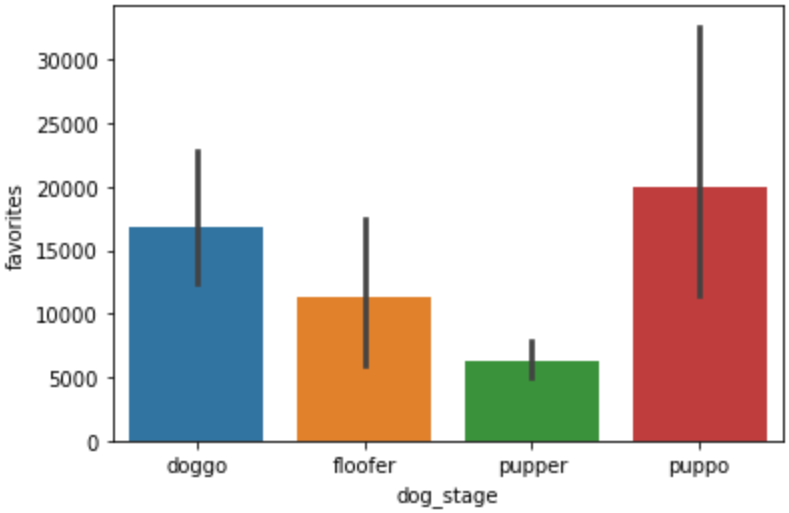
In [217…
```
filtered_df = darchive_clean_master[darchive_clean_master['dog_stage'] != 'None']
sns.countplot('dog_stage', data=filtered_df);
```



From the plot above we can see that pupper has highest frequency. Thus we can ascertain that pupper is most common dog stage in this dataset

Checking the favourite dogs stages

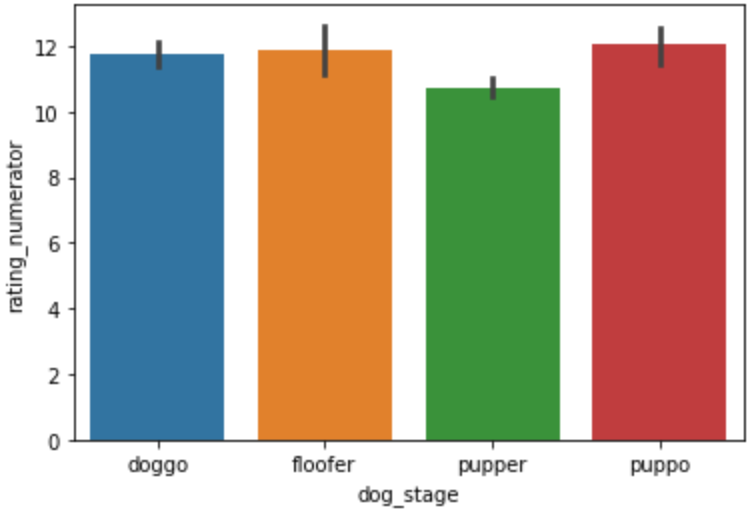
In [218…
```
sns.barplot(x='dog_stage', y='favorites', data=filtered_df);
```



The results showed that the most favourite dog stage is puppo followed by doggo, pupper seemed to favor few followers

Visualizing dog_stages ratings using rating_numerator

In [219…
```
sns.barplot(x='dog_stage', y='rating_numerator', data=filtered_df);
```



We can see that puppo has the highest rating_numerator, followed by floofer the doggo

Insights:

1. The text column was visualized using the word Cloud. The word cloud depicted the most mentioned words are pupper,doggo, dog, rates, dog_rates, good. This means most of the users,followers are rating different stages of dogs. Also pupper seemed to be most mentioned.

2. The distribution of dog stage was visualized using bar graph (countplot). From the plot above we can see that pupper has highest frequency. Thus we can ascertain that pupper is most common dog stage in this dataset

3. The favorites column was plotted against dog_stage. The results showed that the most favourite dog stage is puppo followed by doggo, pupper seemed to favor few followers

4. Visualizing dog_stages ratings using rating_numerator We can see that puppo has the highest rating_numerator, followed by floofer the doggo

Conclusions

The dataset was gathered from different sources. The data was assessed, and was found to be dirty and messy. After cleaning the dataset was merged in master dataset and was saved in CSV file named "twitter_archive_master.csv". The most popular genres over time are adventure and western movies After data wrangling the some data visualization was performed to gain some insights The text column was visualized using the word Cloud. The word cloud depicted the most mentioned words are pupper,doggo, dog, rates, dog_rates, good. This means most of the users,followers are rating different stages of dogs. Also pupper seemed to be most mentioned. The distribution of dog stage was visualized using bar graph (countplot). From the plot above we can see that pupper has highest frequency. Thus we can ascertain that pupper is most common dog stage in this dataset The favorites column was plotted against dog_stage. The results showed that the most favourite dog stage is puppo followed by doggo, pupper seemed to favor few followers

Limitation

There were features with small percentage of missing values such as expanded_urls' has 2.50% missing values in twitter-archive-enhanced data however this feature had some limitations on handling missing values since it was difficult to decide on imputing urls and dropping may also lead to data loss.

Refernces:

Github links

https://github.com/franciskip/Data-Cleaning-and-Data-Wrangling-Preprocessing-

https://github.com/PacktPublishing/Practical-Data-Wrangling

https://github.com/franciskip/Business-Success-prediction

https://github.com/franciskip/Data-Visualiaztion

https://github.com/shravankoninti?tab=repositories

Jiang, S., & Kahn, J. (2020). Data wrangling practices and collaborative interactions with aggregated data. International Journal of Computer-Supported Collaborative Learning, 15(3), 257-281.

Royston, P. (2004). Multiple imputation of missing values. The Stata Journal, 4(3), 227-241.

Chen, C. H., Härdle, W. K., & Unwin, A. (Eds.). (2007). Handbook of data visualization. Springer Science & Business Media.

In [ ]: