WRANGLE REPORT (WeRateDog) by Olaoluwa Idowu

WeRateDogs®
@dog_rates

This is Charlie. He has literal heart-eyes. 14/10 would be an honor to pet

This is a report on the wrangling process of WeRateDog twitter data analysis. The wrangling includes gathering, assessing and cleaning. The wrangling act was carried out in a single notebook, and all the processes were carefully documented.

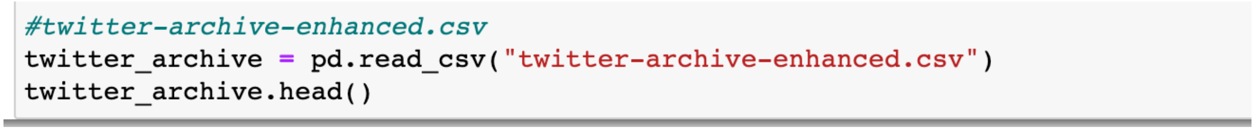
The wrangling steps include.

- Gathering
- Assessing
- Cleaning
- Storing

Gathering

The datasets were gathered from three sources. WeRateDog twitter archive (provide to Udacity), a tsv file for image prediction on Udacity (downloaded programmatically), and additional data downloaded via twitter api.

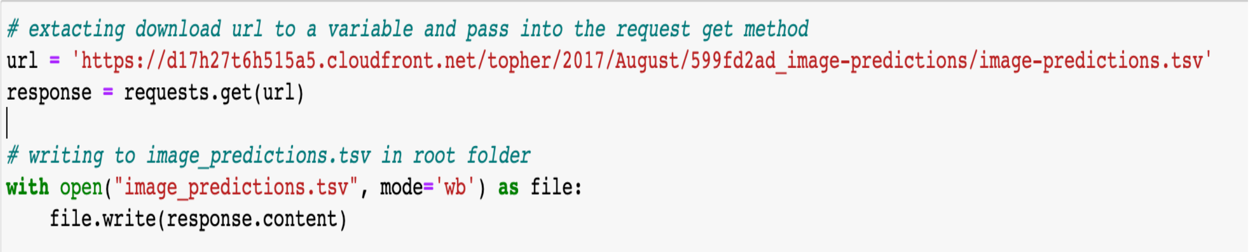
The WeRateDog twitter archive data was extracted using the pandas read_csv method into a data frame “twitter_archive”.



```
#twitter-archive-enhanced.csv
twitter_archive = pd.read_csv("twitter-archive-enhanced.csv")
twitter_archive.head()
```

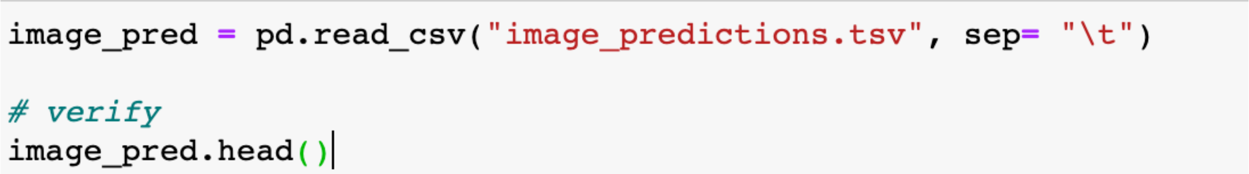
And the head() method was used to confirm the first 5 rows to ensure the data collection worked properly.

The Image prediction dataset download link which was provided by Udacity was downloaded programmatically using python content manager.



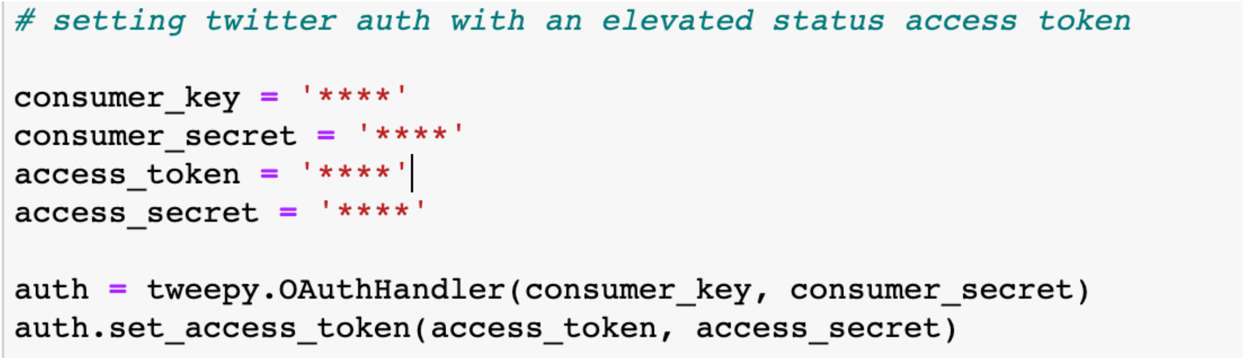
```
# extacting download url to a variable and pass into the request get method
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
response = requests.get(url)

# writing to image_predictions.tsv in root folder
with open("image_predictions.tsv", mode='wb') as file:
    file.write(response.content)
```

The coding used downloaded the file to the same folder of the notebook. While the read_cvs method was used to collect the data into pandas' data frame, this file is tab separated, so the "sep" parameter was used to indicate tab as delimiter. The was verified used the pandas dataframe head() method.



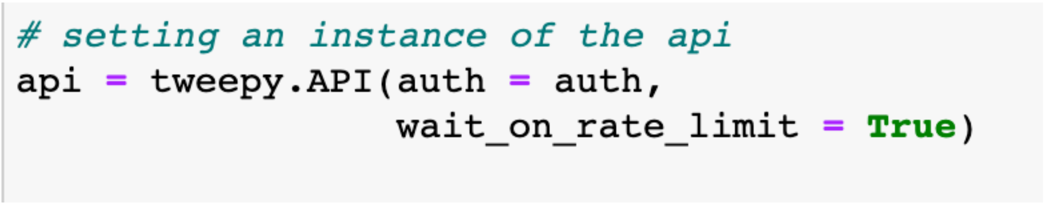
```
image_pred = pd.read_csv("image_predictions.tsv", sep= "\t")

# verify
image_pred.head()
```

Lastly, in the gathering process, additional data was gathered from tweeter api using the tweepy library.
The authentication param was set by saving tokens and keys in variables as setting authentication with the stored access keys and token.



```
# setting twitter auth with an elevated status access token

consumer_key = '****'
consumer_secret = '****'
access_token = '****'
access_secret = '****'

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

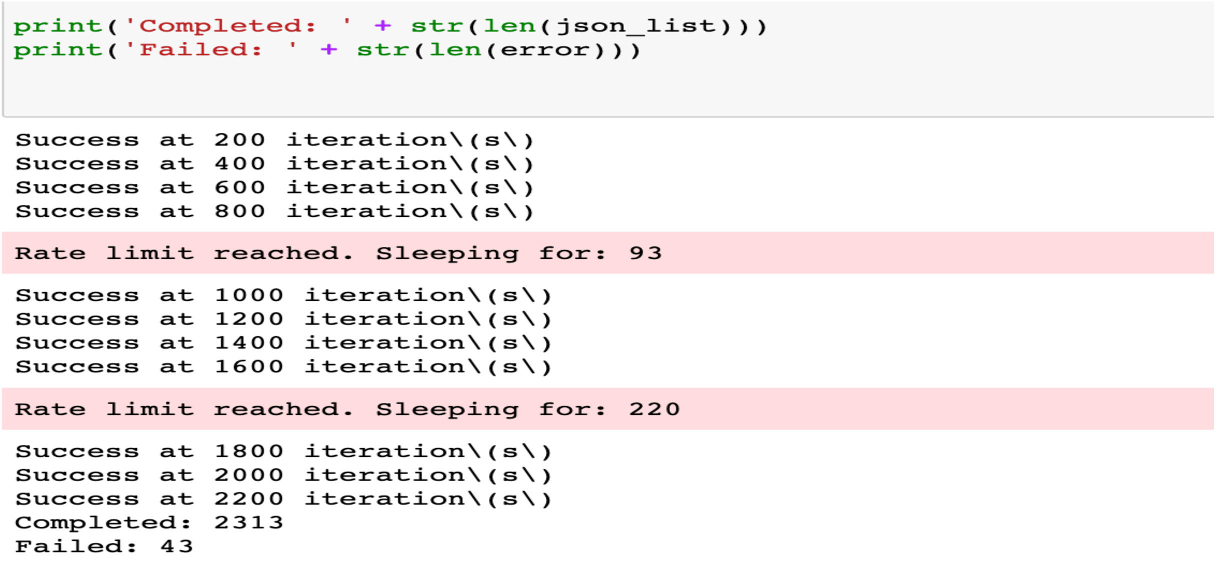
An instance of the api was created and 'wait_on_rate_limit' parameter was set to 'True' to prevent code break due to data rate limit.



```
# setting an instance of the api
api = tweepy.API(auth = auth,
                 wait_on_rate_limit = True)
```

The additional data was extracted from the api and appended to a list per tweet_id in the twitter_archive data.

Two print statements were used to track the progress and ascertain the result, and there were 2313 successes



```
print('Completed: ' + str(len(json_list)))
print('Failed: ' + str(len(error)))
```

```
Success at 200 iteration\(s\)
Success at 400 iteration\(s\)
Success at 600 iteration\(s\)
Success at 800 iteration\(s\)
Rate limit reached. Sleeping for: 93
Success at 1000 iteration\(s\)
Success at 1200 iteration\(s\)
Success at 1400 iteration\(s\)
Success at 1600 iteration\(s\)
Rate limit reached. Sleeping for: 220
Success at 1800 iteration\(s\)
Success at 2000 iteration\(s\)
Success at 2200 iteration\(s\)
Completed: 2313
Failed: 43
```

The json list was written into a txt file with python content manager as tweet_json.txt. And this txt file was extracted line by line to insert into three columns: tweet_id, retweet_count, and favorite_count.

Assessing

The three datasets were assessed one after the other.

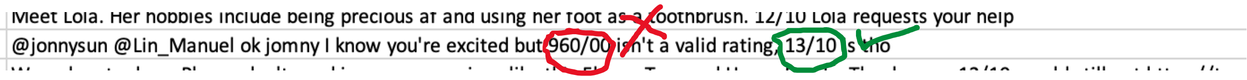
Starting from the twitter archive data:

from visual assessment there are some null values.

- Dog categories seems to be in different columns

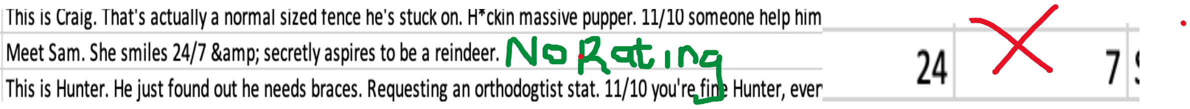
```
 13   doggo                       2356 non-null   object
 14   floofer                     2356 non-null   object
 15   pupper                      2356 non-null   object
 16   puppo                       2356 non-null   object
```

- Some dog names are missing, and some don't seem to make sense.
- The dog names seem to be gotten from the text column
- Visual assessment using excel showed extracted rating numerator and denumerator were 24 and 7, correct ones should be 13 and 10. And also a part extracted rating numerator and denumerator were 24 and 7 for an observation that hard no rating.



Meet Lola. Her hobbies include being precious af and using her foot as a toothbrush. 12/10 Lola requests your help

@jonnysun @Lin_Manuel ok jomny I know you're excited but 960/00 isn't a valid rating, 13/10 is tho



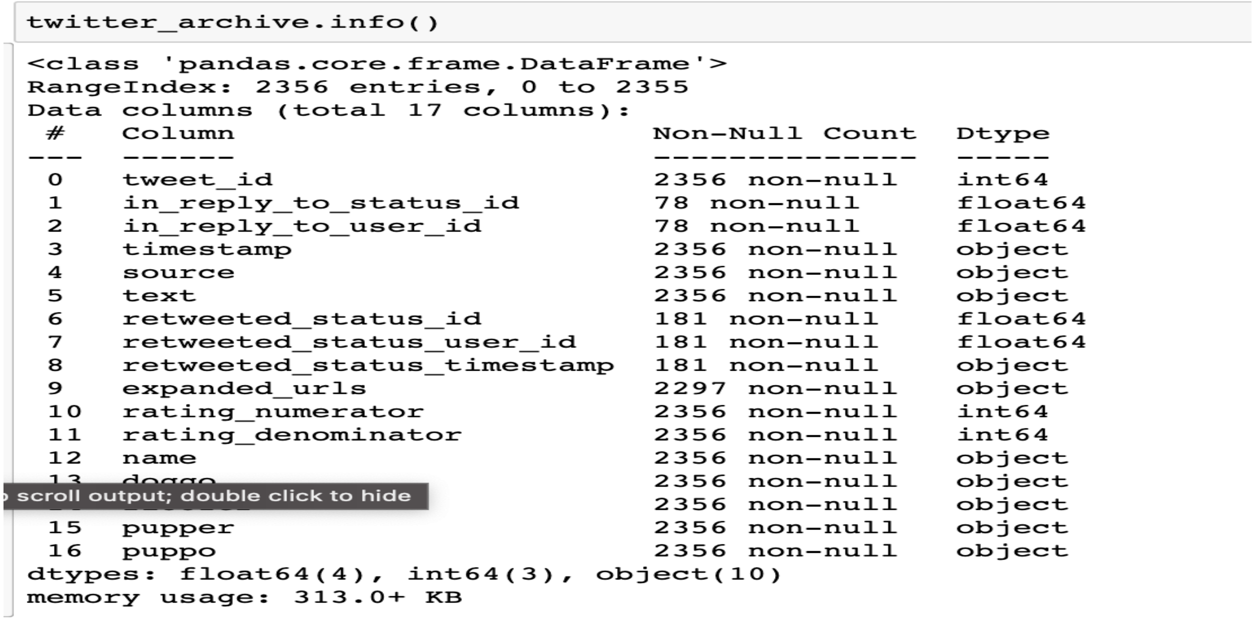
This is Craig. That's actually a normal sized fence he's stuck on. H*ckin massive pupper. 11/10 someone help him

Meet Sam. She smiles 24/7 &amp; secretly aspires to be a reindeer. No Rating

This is Hunter. He just found out he needs braces. Requesting an orthodogtist stat. 11/10 you're fine Hunter, ever

24 7

Programmatical assessment was used to check for duplicate tweet ids, duplicated values, datatypes etc.

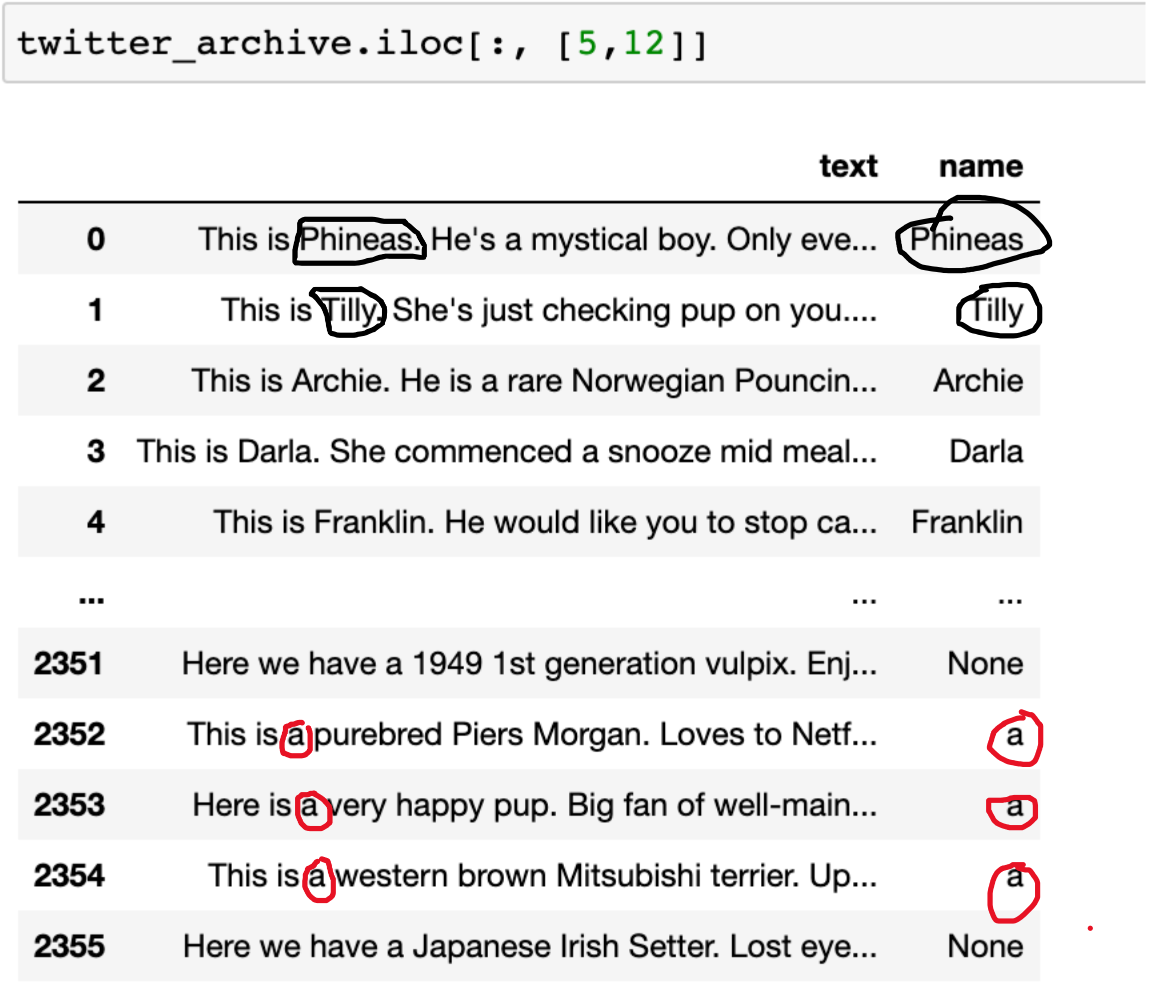


```
twitter_archive.info()
```

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

- tweet_id Datatype is int. It should be string not int.
- Some of the observations are retweets, we need to remove this rows using the retweeted_id, to avoid double observations

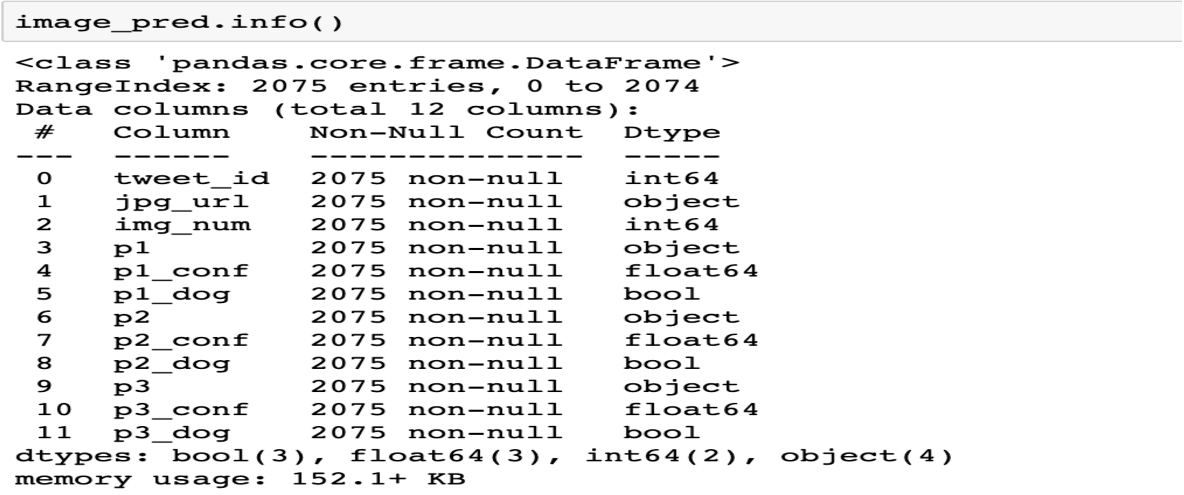
Further assessment of Dog names column confirmed the names were extracted from the text column. However, all names start with capital letters which gave an insight on how to clean the name column.



```
twitter_archive.iloc[:, [5,12]]
```

	text	name
0	This is Phineas. He's a mystical boy. Only eve...	Phineas
1	This is Tilly. She's just checking pup on you....	Tilly
2	This is Archie. He is a rare Norwegian Pouncin...	Archie
3	This is Darla. She commenced a snooze mid meal...	Darla
4	This is Franklin. He would like you to stop ca...	Franklin
...	...	...
2351	Here we have a 1949 1st generation vulpix. Enj...	None
2352	This is a purebred Piers Morgan. Loves to Netf...	a
2353	Here is a very happy pup. Big fan of well-main...	a
2354	This is a western brown Mitsubishi terrier. Up...	a
2355	Here we have a Japanese Irish Setter. Lost eye...	None

Assessing the image prediction data frame:

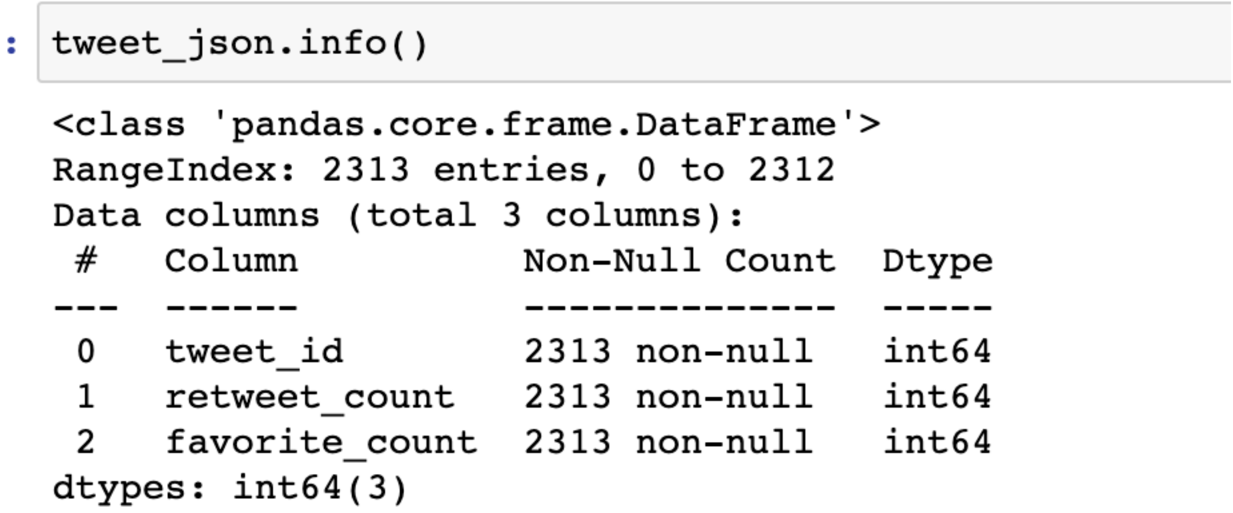


```
image_pred.info()
```

```
<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
```

- No null values
- There are different predictions, it would be nice to choose the best prediction
- Not every user's dog has prediction. The number of rows in our prediction data is lesser that number of observations in the archive DataFrame.
- tweet_id Datatype is int. It should be string not int.

Assessing the tweet_json api data:



```
: tweet_json.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2313 entries, 0 to 2312
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2313 non-null   int64
 1   retweet_count   2313 non-null   int64
 2   favorite_count  2313 non-null   int64
dtypes: int64(3)
```

- tweet_id Datatype is int. It should be string not int.

Cleaning

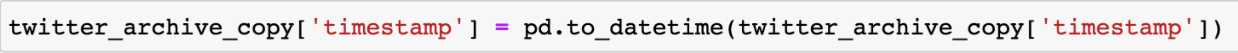
The cleaning process involves; define, code and test.

Quality issues fixed:

- tweet_ids fixed in all dataframes. tweet_id in the three dataframes was converted to strings

- The timestamp column in the twitter archive data was converted from strings to datetime.

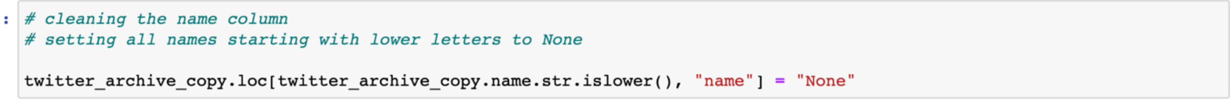
code



```
twitter_archive_copy['timestamp'] = pd.to_datetime(twitter_archive_copy['timestamp'])
```

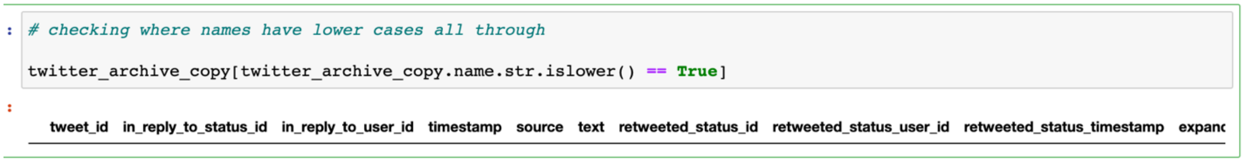
- Dog name column in twitter archive dataframe was fixed. Only names starting with capital letter was retained. All other text in the column are not names of dogs and dog owners didn't specify their dog names, so they were returned as None.

Code



```
# cleaning the name column
# setting all names starting with lower letters to None

twitter_archive_copy.loc[twitter_archive_copy.name.str.islower(), "name"] = "None"
```

Test



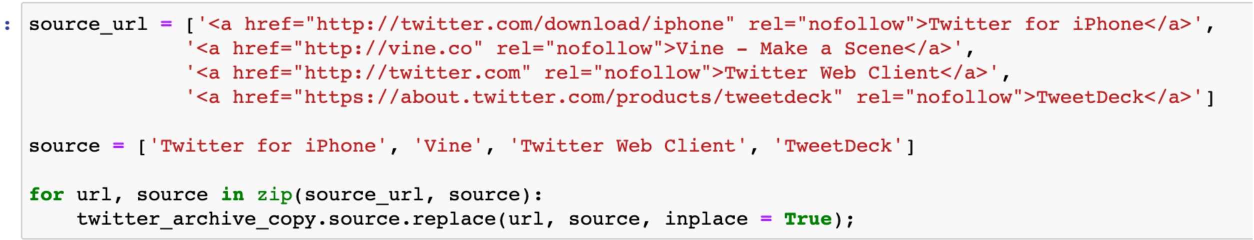
```
# checking where names have lower cases all through

twitter_archive_copy[twitter_archive_copy.name.str.islower() == True]
```

```
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  expanc
```

- After finding and dropping retweeted observation 3 columns (retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp,) were dropped in the twitter archive dataframe as there is very little information from these columns.

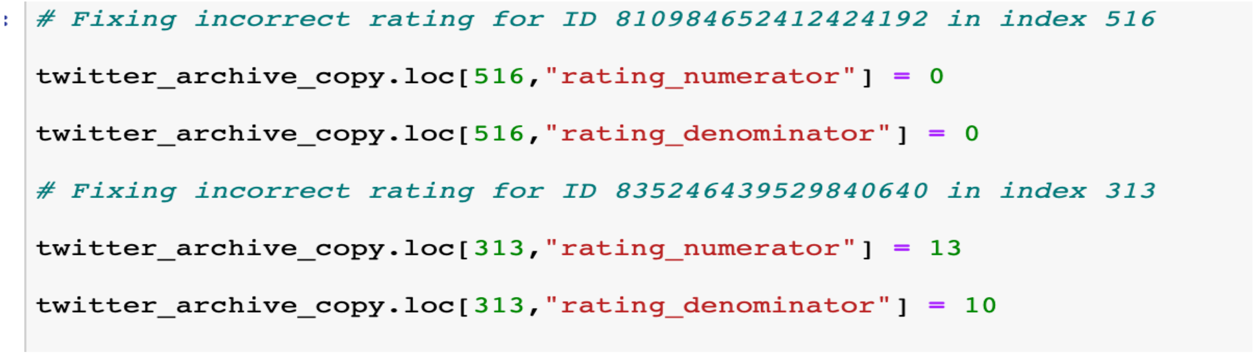
- Source url was replaced with the source name.

Code



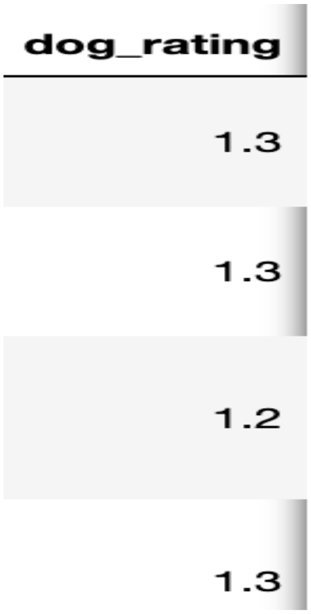
```
source_url = ['<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>',
             '<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>',
             '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
             '<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetDeck</a>']

source = ['Twitter for iPhone', 'Vine', 'Twitter Web Client', 'TweetDeck']

for url, source in zip(source_url, source):
    twitter_archive_copy.source.replace(url, source, inplace = True);
```

- Fixing incorrect ratings for ID 810984652412424192 and 835246439529840640. Setting ratings for ID 810984652412424192 to 0 and for 835246439529840640 to 13/10



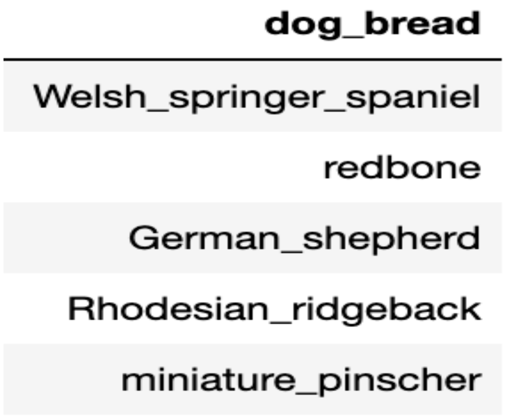
```
# Fixing incorrect rating for ID 810984652412424192 in index 516

twitter_archive_copy.loc[516,"rating_numerator"] = 0

twitter_archive_copy.loc[516,"rating_denominator"] = 0

# Fixing incorrect rating for ID 835246439529840640 in index 313

twitter_archive_copy.loc[313,"rating_numerator"] = 13

twitter_archive_copy.loc[313,"rating_denominator"] = 10
```

- Dog rating column was created, by dividing the rating numerator by the rating denominator.



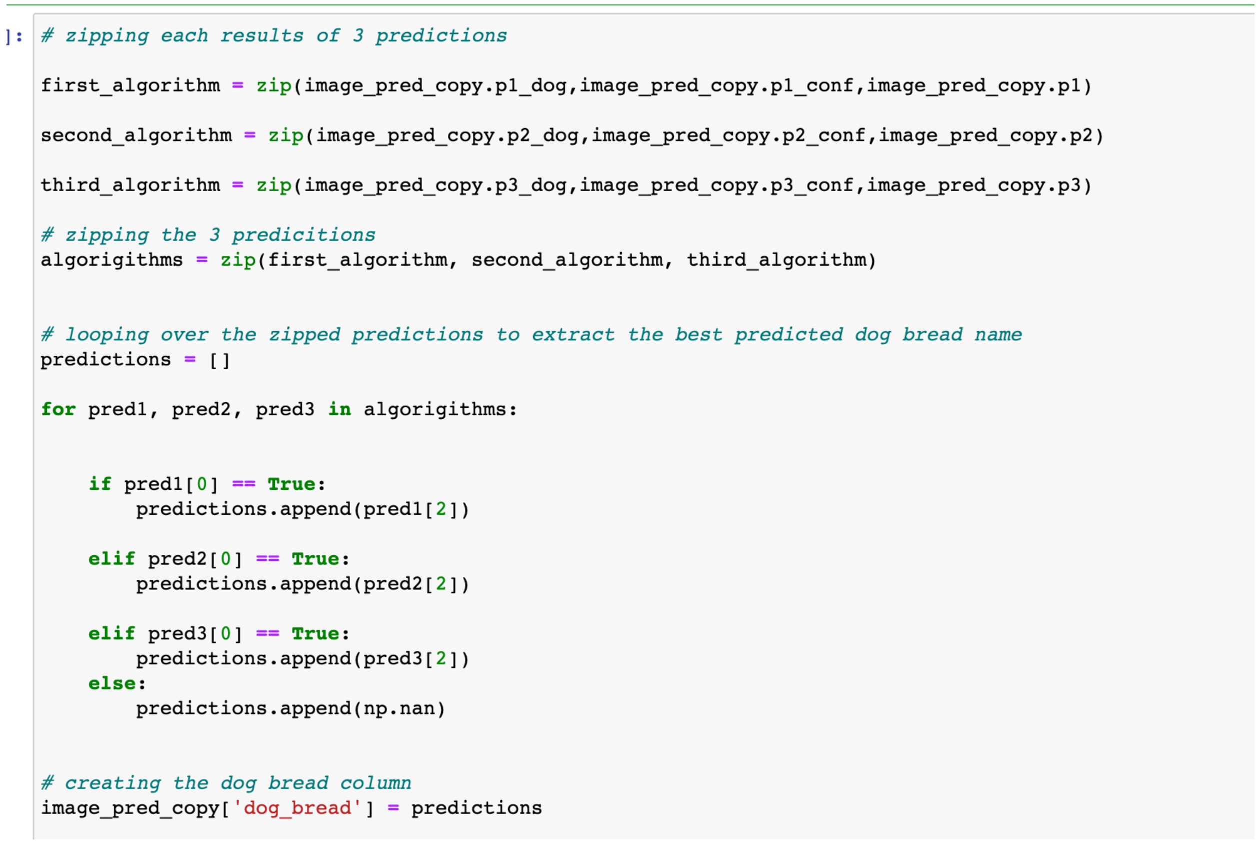
dog_rating
1.3
1.3
1.2
1.3

- The prediction columns set as a single column and named as Dog bread column. To ensure quality of data, dog bread was extracted from second and third predictions where first prediction is False



dog_bread
Welsh_springer_spaniel
redbone
German_shepherd
Rhodesian_ridgeback
miniature_pinscher

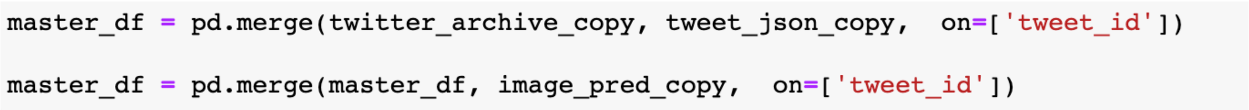
Code



```
# zipping each results of 3 predictions

first_algorithm = zip(image_pred_copy.p1_dog,image_pred_copy.p1_conf,image_pred_copy.p1)

second_algorithm = zip(image_pred_copy.p2_dog,image_pred_copy.p2_conf,image_pred_copy.p2)

third_algorithm = zip(image_pred_copy.p3_dog,image_pred_copy.p3_conf,image_pred_copy.p3)

# zipping the 3 predicitions
algorigithms = zip(first_algorithm, second_algorithm, third_algorithm)


# looping over the zipped predictions to extract the best predicted dog bread name
predictions = []

for pred1, pred2, pred3 in algorigithms:


    if pred1[0] == True:
        predictions.append(pred1[2])

    elif pred2[0] == True:
        predictions.append(pred2[2])

    elif pred3[0] == True:
        predictions.append(pred3[2])
    else:
        predictions.append(np.nan)


# creating the dog bread column
image_pred_copy['dog_bread'] = predictions
```

Tidiness issues fixed:

- Dog stages merged in one column from four columns.
- Merging the three datasets as one



```
master_df = pd.merge(twitter_archive_copy, tweet_json_copy,  on=['tweet_id'])

master_df = pd.merge(master_df, image_pred_copy,  on=['tweet_id'])
```

Limitations

- It seems some tweets have been deleted as some tweet id were not found and some were for forbidden. This was an issue while getting additional data from twitter api
- Twitter data limit rate is quite low which resulted in more time while trying to extract data. It was somehow difficult to rerun analysis and check for other ways to diagnose some tweeted fails.