We Rate Dogs Analysis

Udacity Nanodegree Project 2

Version: 0.1

Prepared by: Koketso Mangwale

Date: 06/09/2022

Wrangling Data – We Rate Dogs

This document describes the gathering, assessing and cleaning process on the We Rate Dogs data with tweets from August 15, 2015 to August 1, 2017.

In this project I am using the data from the WeRateDogs twitter archive, twitter API additional tweet data and an image predictions dataset.

Dataset Description and Data Gathering

Twitter archive sourced from a csv file – tweet basic data with 2356 rows and 17 columns

- tweet_id: Tweet ID
- reply columns: in_reply_to_status_id, in_reply_to_user_id
- timestamp: Date timestamp for the tweet
- source: Tweet source
- text: Tweet text
- retweets columns: retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp
- expanded_urls: Expanded urls
- rating_numerator: Rating numerator
- rating_denominator: Rating denominator
- name: Dog name
- dog stages: doggo, floofer, pupper, puppo

	531	1864
tweet_id	808106460588765185	675362609739206656
in_reply_to_status_id	NaN	NaN
in_reply_to_user_id	NaN	NaN
timestamp	2016-12-12 00:29:28 +0000	2015-12-11 17:12:48 +0000
source	<a href="http://twitter.com/download/iphone" r...	<a href="http://twitter.com/download/iphone" r...
text	Here we have Burke (pupper) and Dexter (doggo)...	This is Daisy. She loves that shoe. Still no s...
retweeted_status_id	NaN	NaN
retweeted_status_user_id	NaN	NaN
retweeted_status_timestamp	NaN	NaN
expanded_urls	https://twitter.com/dog_rates/status/808106460...	https://twitter.com/dog_rates/status/675362609...
rating_numerator	12	12
rating_denominator	10	10
name	NaN	Daisy
doggo	doggo	NaN
floofer	NaN	NaN
pupper	pupper	NaN
puppo	NaN	NaN

Image predictions sourced from the web – dog breed predictions with 2075 rows and 12 columns

- tweet_id: Tweet ID
- jpg_url: Image url
- img_num: Image number
- p1: First prediction breed image
- p1_conf: Confidence prediction

- p1_dog: Is the prediction a dog?
- p2: Second prediction breed image
- p2_conf: Confidence prediction
- p2_dog: Is the prediction a dog?
- p3: Third prediction breed image
- p3_conf: Confidence prediction
- p3_dog: Is the prediction a dog?

	0	1	
tweet_id	666020888022790149	666029285002620928	
jpg_url	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	http
img_num	1	1	
p1	Welsh_springer_spaniel	redbone	
p1_conf	0.465074	0.506826	
p1_dog	True	True	
p2	collie	miniature_pinscher	
p2_conf	0.156665	0.074192	
p2_dog	True	True	
p3	Shetland_sheepdog	Rhodesian_ridgeback	
p3_conf	0.061428	0.07201	
p3_dog	True	True	

Additional twitter archive sourced from tweet_json.txt with Twitter API – dog tweet favorite and retweet count

- favorite and retweet counts are extracted from the tweet json and stored a dataframe.

	0	1	2
2344	666058600524156928	115	61
2345	666057090499244032	304	146
2346	666055525042405380	448	261
2347	666051853826850816	1253	879
2348	666050758794694657	136	60
2349	666049248165822465	111	41
2350	666044226329800704	311	147
2351	666033412701032449	128	47
2352	666029285002620928	132	48
2353	666020888022790149	2535	532

Question(s) for Analysis

I will explore the following questions:

- Which dogs are the most rated? What dog types are they?
- Are the most rated dogs also the most popular/most liked?
- Is there a relationship between dog ratings, favorites and retweets?

Data Assessment

The twitter_ archive and image_predictions datasets were assessed using the pandas info(), duplicated() and isnull() methods to check for missing data stored as NaN or None.

```
twitter_archive.isnull().sum()
```

```
tweet_id                       0
in_reply_to_status_id       2278
in_reply_to_user_id         2278
timestamp                      0
source                         0
text                           0
retweeted_status_id         2175
retweeted_status_user_id    2175
retweeted_status_timestamp  2175
expanded_urls                 59
rating_numerator               0
rating_denominator             0
name                         745
doggo                       2259
floofer                     2346
pupper                      2099
puppo                       2326
dtype: int64
```

To inspect the tweet_json text files, I used Atom text editor and the online JSON editor to view the contents as JSON objects and try to find the favorite and retweets nodes.

The following data quality and tidiness issues were discovered:

- 2356 missing data on twitter archive
  - 2175 on retweets columns
  - 2278 missing data on in_reply_to_status_id and in_reply_to_user_id columns
- Messy data
  - twitter archive dog stages columns are not variables
  - number of prediction, confidence prediction, p1_dog, p2_dog and p3_dog columns in image predictions dataset are not variables
- Dirty data on twitter archive
  - some ratings are retweets
  - some tweets are replies
  - some dogs are missing names
  - some dogs are missing dog stages
  - timestamp stored as string
  - 21 tweets have an invalid denominator rating
- Dirty data on image predictions
  - some images are not dogs

Data Cleaning

First I am dealing with missing data then messy data and at the end I clean low quality issues so I end up with high quality data for analysis.

- In_reply_to_status_id and in_reply_to_user_id have nulls stored as NaN. These are not missing values, they are tweet replies and not all tweets have replies. Therefore the columns were removed as below:

```
#drop the tweet replies columns
twitter_archive_clean.drop(['in_reply_to_status_id','in_reply_to_user_id'], axis = 1, inplace = True)
```

- retweeted_status_id, retweeted_status_user_id and retweeted_status_timestamp have nulls stored as NaN. These are not missing values, they are tweets which are retweets and not all tweets are retweets. Therefore the columns were removed as below. That brings twitter archive data to 2175 rows and 14 columns.

```
#drop the retweet columns
twitter_archive_clean.drop(['retweeted_status_id','retweeted_status_user_id','retweeted_status_timestamp'],axis = 1,
                           inplace = True)
```

- name has nulls stored as NaN. Some are missing values while others may be puppies that are not named yet. Extract names from text column and then replace NaN names left with 'a'
- Dog stages: doggo, floofer, pupper, puppo have nulls stored as None. The column None values were read from csv as NaN. The columns are not variables, so they should be treated as observational units. Therefore I concatenated the dog stages columns into one column and removed the doggo, floofer, pupper, puppo columns form the dataframe. Thus bringing the twitter archive data to 2064 rows and 9 columns.

```
#replace nan with empty string using fillna
twitter_archive_clean['doggo'].fillna(' ', inplace = True)
twitter_archive_clean['floofer'].fillna(' ', inplace = True)
twitter_archive_clean['pupper'].fillna(' ', inplace = True)
twitter_archive_clean['puppo'].fillna(' ', inplace = True)
```

```
#Create a new column called dog_stages, cocatenate columns and strip spaces
twitter_archive_clean['dog_stages'] = twitter_archive_clean.apply(lambda r: (str(r['doggo'].strip()) +
                                                                             str(r['floofer'].strip()) +
                                                                             str(r['pupper'].strip()) +
                                                                             str(r['puppo'].strip())) , axis = 1)
```

- The new dog_stages column has missing values stored as an empty string and
- Columns in image prediction dataset are transformed to observational units using the pandas melt() function and combing(using concat method) the relevant transformed data as one clean data frame bringing the dataset to 6225 rows and 5 columns named 'tweet_id', 'prediction_num', 'breed_name', 'is_dog', and 'prediction_confidence'.
    - In addition, I cleaned the prediction_num column as below:

```
#strip the letter 'p' in p1, p2 and p3 and convert those columns to data type integer
image_predictions_clean.prediction_num = image_predictions_clean.prediction_num.str.strip('p').astype(int)
```

Finally, I filtered the image prediction clean data by only images that are dogs which left me working with 4584 rows.

```
image_predictions_clean.sample(10)
```

	tweet_id	prediction_num	breed_name	is_dog	prediction_confidence
5797	808733504066486276	3	golden_retriever	True	0.016972
2431	672604026190569472	2	miniature_poodle	True	0.178404
3144	716080869887381504	2	chow	True	0.254717
3569	783334639985389568	2	Shetland_sheepdog	True	0.130611
3032	705442520700944385	2	kuvasz	True	0.224556
370	672975131468300288	1	pug	True	0.836421
848	695446424020918272	1	basenji	True	0.748904
5103	705102439679201280	3	Pomeranian	True	0.076922
208	669970042633789440	1	miniature_pinscher	True	0.734744
4895	687494652870668288	3	Tibetan_mastiff	True	0.041692

```
print(image_predictions_clean.shape)
image_predictions_clean.info()
```

```
(4584, 5)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4584 entries, 0 to 6223
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   tweet_id              4584 non-null   int64
 1   prediction_num        4584 non-null   int32
 2   breed_name            4584 non-null   object
 3   is_dog                4584 non-null   bool
 4   prediction_confidence  4584 non-null   float64
dtypes: bool(1), float64(1), int32(1), int64(1), object(1)
memory usage: 165.6+ KB
```

Quality Issues:

- Converting timestamp from string to datetime data types
- Replacing invalid rating denominators with 10
- Renaming columns in the tweets count to meaningful names

At the end I merged the 3 dataset by tweet_id and saved the cleaned datasets in new csv files. The merged 4721 dataset now has rows and 15 columns

```
#merge twitter archive with favourite and retweet count
merged_df = twitter_archive_clean.merge(tweet_counts_clean,
                                        how='left',
                                        left_on = 'tweet_id',
                                        right_on = 'tweet_id',
                                        validate = '1:m',
                                        suffixes = ('x','y') )
merged_df.info()
```

```
#lastly merge with image_predictions
twitter_archive_master = merged_df.merge(image_predictions_clean,
                                         how='left',
                                         left_on = 'tweet_id',
                                         right_on = 'tweet_id',
                                         validate = '1:m',
                                         suffixes = ('x','y') )
```

```
# save the final merged dataset to csv
twitter_archive_master.to_csv('data/twitter_archive_master.csv')
```