wrangle_report

June 5, 2019

Investigation of Twitter archive of `WeRateDogs`

0.1 Table of Contents

Introduction
Questions imposed
Data Wrangling
Gathering Data
Accessing Data
Cleaning Data
End
Introduction
Home

0.1.1 About the Dataset

The dataset that is being wrangled (and analyzed and visualized) is the tweet archive of Twitter user @dog_rates, also known as WeRateDogs. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "they're good dogs Brent." WeRateDogs has over 4 million followers and has received international media coverage.

This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017. More on this soon.

0.1.2 Inspiration

Is it possible to find the best rated dogs based on the tweets

Questions - - Ratings of dogs based on type - Best dog according to rating - Tweets based on hour of the day - Tweets based on the days of the week - Tweets based on month - Most important factor which leads to better rating

Home

Data Wrangling

Home

Gathering Data Data has been gathered in 3 different formats from 3 different sources. Data files included are `twitter-archive-enhanced.csv`, `image-predictions.tsv`

from url https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv) and retweet and favorite counts from Twitter's API as `tweet-json.txt`

0.1.3 Accessing Data

Home #### General Properties

- The given dataframe has 2356 rows and 17 different columns
- Columns are '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'
- `in_reply_to_status_id` and `in_reply_to_user_id` have only 78 not-null values
- `retweeted_status_id`, `retweeted_status_user_id` and `retweeted_status_timestamp` have only 181 not-null values
- Data types are

Column Name	Count	Data Type
tweet_id	2356 non-null	int64
in_reply_to_status_id	78 non-null	float64
in_reply_to_user_id	78 non-null	float64
timestamp	2356 non-null	object
source	2356 non-null	object
text	2356 non-null	object
retweeted_status_id	181 non-null	float64
retweeted_status_user_id	181 non-null	float64
retweeted_status_timestamp	181 non-null	object
expanded_urls	2297 non-null	object
rating_numerator	2356 non-null	int64
rating_denominator	2356 non-null	int64
name	2356 non-null	object
doggo	2356 non-null	object
floofer	2356 non-null	object
pupper	2356 non-null	object
puppo	2356 non-null	object

-	rating_numerator	rating_denominator
count	2356.000000	2356.000000
mean	13.126486	10.455433
std	45.876648	6.745237
min	0.000000	0.000000

-	rating_numerator	rating_denominator
25%	10.000000	10.000000
50%	11.000000	10.000000
75%	12.000000	10.000000
max	1776.000000	170.000000

0.1.4 Data Quality Issues

1. Many null values
    - `in_reply_to_status_id`
    - `in_reply_to_user_id`
    - `retweeted_status_id`
    - `retweeted_status_user_id`
2. Incorrect data types
    - `tweet_id`
    - `in_reply_to_status_id`
    - `in_reply_to_user_id`
    - `retweeted_status_id`
    - `retweeted_status_user_id`
3. `rating_denominator` has minimum value as 0 which is not possible for denominators
4. `datetime` format for
    - `timestamp`
    - `retweeted_status_timestamp`
5. Retweets need to be removed to avoid duplication in our analysis. This may be done by removing rows that have non-empty `retweeted_status_id`, `retweeted_status_user_id`, and `retweeted_status_timestamp`
6. Add `rating` column as the ratio of numerator and denominator
7. Reorder the columns into similar ones close to each other after adding or removing some extra columns
8. Some numerators are wrongly entered. They are different as in the comments

0.1.5 Data Tidiness Issues

- `category` column can be created to store the type of dog instead of the last 4 columns named as `doggo`, `floofer`, `pupper`, `puppo`
- Information about one type of observational unit (tweets) is spread across three different dataframes. Therefore, these three dataframes should be merged as they are part of the same observational unit.

0.1.6 Data Cleaning

Home

Remove retweets Retweets need to be removed to avoid duplication in our analysis. This may be done by removing rows that have non-empty `retweeted_status_id`, `retweeted_status_user_id`, and `retweeted_status_timestamp`

Modify Data types Change data types for `tweet_id`, `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id` and `retweeted_status_user_id`

Syntax: `twitter_df['col_name'] = twitter_df['col_name'].astype('datatype')`

Format Date Change `timestamp` and `retweeted_status_timestamp` columns datatype to `datetime` format

Syntax: `df['col_name'] = pd.to_datetime(df['timestamp'])`

Drop record with 0 denominator Record having 0 as the denominator needs to be removed so that the value of rating is consistent

Syntax: `null_index = df[df['rating_denominator'] == 0].index df = df.drop(null_index)`

Wrongly entered Numerators Some numerators are wrongly entered. They are different as in the comments. These need to be corrected to get correct rating

Syntax: `twitter_df.text.str.extract('((?:\d+\.)?\d+)\/(\d+)', expand=True)`

Add 'rating' column Add `rating` column as the ratio of numerator and denominator

Syntax: `df['rating'] = df['rating_numerator']/df['rating_denominator']`

Add category column `category` column can be created to store the type of dog instead of the last 4 columns named as `doggo`, `floofer`, `pupper`, `puppo`

Syntax:

- Create a function to assign the category to the dog
- Use `apply` function to apply that function to all the rows of the dataset

```
df.apply (lambda row: label_category(row), axis=1).value_counts()
df['category'] = df.apply (lambda row: label_category(row), axis=1)
```

Check for duplicate values Duplicate records can be removed as they lead to data redundance

Syntax: `for _ in twitter_df.columns: print(_,sum(twitter_df[_].duplicated()))`

Check for unique values We can see if there is any impossible value as negative value for age or weight etc. We need to check for inly some particular columns

Syntax: `cols = ['doggo', 'floofer', 'pupper', 'puppo', 'rating', 'category'] for _ in cols: print(_,len(twitter_df[_].unique())) print((twitter_df[_].unique()),'\n')`

Adding some extra columns to enhance our data usability Adding `Month`, `Day`, `Hour`, `re-Month`, `re-Day` and `re-Hour` columns to increase our understanding of the time stamps for the tweets and retweets

Syantax: `twitter_df['col_name'] = twitter_df['existing_col_name'].dt.month_name()`

Adding time difference between tweets and retweets Adding the time difference will help us in understanding that what is the average time taken for a new reteeets to be done.

Syntax: `twitter_df['retweetTime'] = twitter_df['timestamp'] - twitter_df['retweeted_status_timestamp']`

Final columns are -

```
'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',
'rating', 'category', 'Month', 'Day', 'Hour', 're-Month', 're-Day',
're-Hour', 'retweetTime'
```

The End
Home