Data Wrangling Report_WeRateGogs

September 12, 2022

1 Introduction

This report first introduces the data collection process and then focuses on data assessment and data cleaning. It highlights the most severe quality and tideness issues and the corresponding cleaning steps, and also briefly summarizes minor issues.

2 Data Collection

Data were collected from multiple sources:

1. A dataset titled `twitter-archive-enhanced` (renamed as `df_tw`) which contains the information of each tweet (e.g., tweet id, creation time, tweet content), of each dog's rating, name, and dog stage. The orignal shape of `df_tw` is `2356 * 17`.
2. Tweepy API from which I collected data about number of retweets, number of favorites and creation time for each tweet in `twitter-archive-enhanced`. In the first round of collection, information of 28 tweets could not retrieved, therefore, I repeated the collection process for the 28 unretrievebale tweets and they were lucily retrieved in the second time. Then, I stored all collected data titled `df_tweepy`.The orignal shape `df_tweepy` is `2356 * 4`.
3. A dataset titled `image-prediction` (renamed as `df_img`) which contains the information of predicted breed of each dog in `twitter-archive-enhanced`. The orignal shape `df_img`is `2075 * 12`.

3 Data Assessment & Data Cleaning

3.1 Quality Issues & Solutions

3.1.1 1. ID fields were stored as inapporiate variable type in dataframes, accessed with `df.info()` & change these variables to correct type using `df.astype()`

1.1. Issue:

- `tweet_id` in `df_tw`, `df_img` and `df_tweepy` were stored as integar, which should be string
- `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id` and `retweeted_status_user_id` in `df_tw`were stored as floats which should be string

1.2. Solution:

- I first defined a function for changing these variables:
```
def ToString(df, val_list):
df[val_list] = df[val_list].astype('string')
return df[val_list].dtypes
```
- I then applied this function to variables need to be changed in the corresponding dataframe

3.1.2 2.timestamp variables were stored as inapporiate variable type in dataframes, accessed with `df.info()` & change these variables to correct type using `pd.to_datetime()`

2.1. Issue:

- timestamp and retweeted_status_timestamp in df_tw, and creation_time in df_tweepy were stored as objects which should be timestamp

2.2. Solution:

- I first defined a function for changing these variables:
```
def ToDateTime(df, val_name):
df[val_name] = pd.to_datetime(df[val_name])
return df[val_name].dtypes
```
- I then applied this function to variables need to be changed in the corresponding dataframe

3.1.3 3. some tweets had retweets , assessed with `df.info()` & dropped these retweets using `df.drop`

3.1. Issue:

- 181 tweets had retweets, which should be removed

3.2. Solution:

- df_tw_clean.drop(df_tw_clean[df_tw_clean['retweeted_status_id'] != '<NA>'].index, inplace = True) were used to drop these retweets

3.1.4 4. abnormal values in rating_denominator and rating_numerator in df_tw, assessed with`series.value_counts()` & corrected them in several steps

4.1. Issue:

- There were abnormal values in rating_denominator (e.g., 130, 110) and rating_numerator (e.g., 143, 121)

4.2. Solution:

- I first mannually checked the dataset to detect the patterns of anomalies in rating_denominator and rating_numerator.
- This the mannual checking showed that the common causes of the anomalies are the following: 1)the rating was not for a single dog but for a group of dogs (e.g., 13 became 130 for 10 dogs), 2) denominator had decimals but only the decimal part was retrieved (e.g., 10.9

became 9), and 3) more than two ratings were mentioned in the text and only one of them was retrieved.

- I then extracted all ratings from texts with `ratings = df_tw_clean['text'].apply(lambda x: re.findall(r'((\d+|\d+\.\d+)/(\d+))', x))`
- Next,I transversed all the texts to re-retrieve all the ratings correctly (i.e., `for rating in ratings`) based on four conditions:
  - if there is not rating (i.e.,`len(rating) == 0`), both denominator and numerator would be 'NA';
  - if there is only one rating (i.e.,`len(rating) == 1`), denominator and numerator would be number before and after '/' respectively;
  - if there are at least two ratings and two ratings with 10 as the denominator (i.e.,`len(rating) == 2 & rating[0][-1] == '10' & rating[1][-1] == '10'`), I calculated the average of the numerator with `rating_numerator_total += float(rating[i][-2])` and `rating_numerator_avg = rating_numerator_total / len(rating)`, and all denominator in this cases were 10.
  - if none of the above condition was met, both denominator and numerator would be 'error'.
- Finally, I went over the error rows and read through the texts and corrected the denominator and numerator following two steps:
  - defined a function for correcting errors:

```
def correct_values(correction_dict):
for key, value in correction_dict.items():
df_tw_clean.loc[key,['rating_numerator_corrected','rating_denominator_corrected']]
= value
```

  - created a dictionary of correct denominator and numerator and apply the function to the dictionary
- After all the above procedure, one row did not have any rating and was dropped (row# 516)

3.1.5 5. missing or irrelevant information in `expanded_urls` in `df_tw`, visually assessed & filled the missing information based on other information and store irrelevant information in another variable

5.1. Issue:

- There were missing information in `expanded_urls` (N = 58)
- There were irrevelant information `expanded_urls` (i.e., non-twitter links such as `https://gofundme.com/ydvmve-surgery-for-jax` because these tweets included these additional link in their content)

5.2. solution:

- To address the missing information, I filled the missing links based on `tweet_id` because all links follow this pattern `https://twitter.com/dog_rates/status/tweet_id/photo/1`
- To address the irrelevant information, I took the following steps:
  - First, I stored all urls for each tweet in a new list named `urls_list`.
  - Then, I trasversed all the urls in the new list and seperately stored twitter links and non-twitter links in two newly-created dictionaries.
  - Finnally, I updated `expanded_urls` with the twitter links dictionary and created a new variable–`additional_urls` for nontwitter links.

3.1.6 6. redudant information in `source` in `df_tw`, visually assessed & retrieved key information

6.1. Issue:

- There were redudant infomration in `source` (e.g. `<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>`)

6.2. Solution:

- I retrieved the key information (e.g., Twitter, iPhone) using `df_tw_clean['source'] = df_tw_clean['source'].apply(lambda x: re.findall(r '>(.*)<', x)[0])`.

3.1.7 7. duplicated values for 'jpg_url' in`df_img`, assessed with `series.duplicated().value_counts()` & dropped these tweets

7.1. Issue:

- There were duplicated values for 'jpg_url' (N = 66), indicating some tweets had identical images

7.2. Solution:

- I dropped these duplicated values, because we are only interested in original ratings with images.

3.1.8 8. missing images for some tweets in `df_img`, assessed with `df.info()` & dropped these tweets

8.1. Issue:

- There were missing values for some tweets (N of `df_img` < N of `df_tw`), indicating some tweets did not include pictures

8.2. Solution:

- I used inner merge when merging the datasets and creating the master dataset, because we are only interested in original ratings with images.

3.2 Tideness Issues & Solutions

3.2.1 9. multiple variables/columns for dog stage, visually assessed & restructured into one variable

9.1. Issue:

- There were multiple variables/columns (i.e.,`doggo`, `pupper`, `puppo`, `floofer`) for one variable – `dog_stage`

9.2. Solution:

- I first reated a new variable/column named `dog_stage` and filled each row based on the information given using `df_tw_clean.dog_stage = df_tw_clean.doggo + df_tw_clean.floofer + df_tw_clean.pupper + df_tw_clean.puppo`
- Meanwhile, I checked the dog stage information for all dogs using `df_tw_clean.groupby(["doggo", "floofer", "pupper", "puppo"]).size().reset_index().rename(columns={0: "count"})`
- This checking showed that some dogs were in multiple stages (N = 12), and thus, I mannually read the text for the 12 dogs and check if they were truly in multiple stages or it was due to error recording.
- The mannual checking on dogs with multiple stages show that 6 dogs were not truly in multiple stage, therefore, I corrected dog stage information for the 6 dog following a similiar procedure to 4.2.
- Finally, I dropped `doggo`, `pupper`, `puppo` and `floofer`

10. multiple predictions of dog breed, visually assessed & restructured into one variable

10.1. Issue:

- There were multiple predictions of dog breed (i.e, `p1`, `p2`, `p3`) ##### 10.2. Solution:
- I first created two new variables: `dog_breed` and `p_conf`
- I then chose the algorithm predicting dog correctly (since all images were indeed dogs) and with the highest accuracy. Note that since the accuracy ranking is always p1 > p2 >p3, I could only check whether each algorithm made the correct dog prediction in this order and assigned the predicted breed; if none of them was correct, I could assign NA value.
- By doing so, 318 dogs were not classfied in any breed, and there were 113 breeds in total.
- Finally, I dropped `p1`, `p2`, `p3`,`p1_cof`, `p2_cof`, `p3_cof`,`p1_dog`, `p2_dog`,`p3_dog`.

11.three seperate dataframes, visually assesed & merged into one master dataframe

11.1. Issue:

- There were the three seperate dataframes, which should be merged into one master dataframe for future analysis ##### 11.2. Solution:
- I first merged the three dataframes based on `tweet_id` with `df_master = df_tw_clean.merge(df_img_clean,how = 'inner', on = 'tweet_id').merge(df_tweepy_json_clean, how = 'inner', on = 'tweet_id')`.
- I then dropped irrelevant columns `['in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp']`.
- Next, I uniformed the expression of missing values to be 'NA'.
- Furthermore, reorderd the columns based on the importance `['tweet_id','dog_rating', 'dog_stage','dog_breed','retweets', 'favorites', 'img_num', 'creation_time','timestamp','p_conf', 'rating_numerator_corrected', 'rating_denominator_corrected','name','source','text','expanded_urls', 'jpg_url']`.

- Finally, to make the analysis more meaningful, I created a variable `dog_gender` with the following steps:
  - created two lists for words that refers male (i.e., `male = ['he', 'him', 'his', "he's", 'himself', 'boy']`) or female dogs (i.e., `female = ['she', 'her', 'hers',"she's", 'herself', 'girl']`)
  - transversed all text to locate the gender-related words and assigned the gender based on the gender-related words used in the text; if none gender-related words were found, I assigned 'NA'.
  - by doing so, there were 1028 male dogs, 338 female dogs and 627 dogs without gender information.

After final checks with `df_master`, I saved it as the master dataframe and conducted EDA and virsualization afterwards.