Wrangle-Report

1. Wrangling Report

Prepared by Akash Yadav

1.1 Gathering

In the gathering step, we are asked to gather 3 different tables using 3 different ways: 1. twitter-archive-enhanced.csv: this file is obtained by downloading it manually from the udacity site. 2. image-predictions.tsv: this file is obtained by downloading it programmatically using the python's requests library 3. tweet_json.txt: this file is obtained by calling the twitter API for each tweet, using the tweet id as parameter, which we obtained from the twitter-archive-enhanced.csv data. Instead of using the Requests library, we use Tweepy library which provides a higher level way to get the data from the API. After that, we make sure that all the data are available in the same directory as the wrangle-act.ipynb

1.2 Assessing

After gathering all the 3 files, their data is stored into a data frame for easier assessment and cleaning. In order to assess the data, I examined it visually and programmatically using python's pandas library. First, we printed out all the data frames entirely, used the info() function to assess the datatypes, used describe() function to summarize the quantitative variables in the datasets, etc. Then examined the data frames more specifically by examining each variable separately and found out the following issues.

Quality

tweet_api_data findings:

1. Have unrelated columns like user, favorited, retweeted which is based on the personal user information who obtain the data from API. This is found by looking through the columns inside the tweet_json.
2. Have a column in which all data is null: contributors, coordinates, geo, place, quoted status id, and quoted status id str This is found by calling the info() method, which indicates the number of non null data in each column.
3. Source column contains the whole html tag . This is obtained by looking through the data in each column.

4. possibly_sensitive and possibly_sensitive_appealable column is all zero. This is initially obtained by looking through several rows in the data, after which I started getting a little bit suspicious since it seems like all the data in those 2 columns are always 0. This suspicion is then confirmed by using the value_counts() function.

image_predictions table :
1. p1, p2, p3 whitespace usage is not standardized, some uses -, some uses _, some are capitalized, some are not. This is obtained by looking through the data.

twitter-archive-enhanced.csv findings:

1. the dog name may not be accurate, some names are a or an.
2. the dog name has None string that should be a null instead.
3. Column names doggo, floofer, puppo, and pupper have value either None or its column name.
4. Some ratings are wrong, it is marked by the denominator is not 10. One of the tweets text is the below: > "This is Darrel. He just robbed a 7/11 and is in a high speed police chase. Was just spotted by the helicopter 10/10".

However, the captured rating was 7/11 instead of 10/10 5. Some rating_numerator also captures wrong data, spotted by having the value less than 10. 6. the timestamp should be put as datetime instead of string

Tidiness :
•The dog stage columns in twitter_archive can be arranged into a single column.
• The image predictions could be condensed to show just the most confident dog breed prediction.
• All three data frames can be combined into one single dataframe

1.3 Cleaning

In this step we are trying to clean the data, however, due to time constraints and the project's recommendation, we will only clean a few of the issues we mentioned above, instead of all of them**.**

1.I could clean all the tables one by one, but all of them share some cleaning needs or are dependent on each other to do so (for example removing retweets or pictures not containing dogs). By merging them all together as the first step, I can save some coding time and avoid repetition.

2. Drop the replies, retweets and the corresponding columns and also drop the tweets without an image or with images which don't display doggos Now that this is done, we have to remove the replies, retweets and the tweets without an image displaying a dog, because we only want original tweets with images. Let's visualize our missing data first.

3. Clean the datatypes of the columns There are some columns in tweet_api_data and twitter_archives which are redundant as they are different columns who have similar meaning. To solve this, we can drop the timestamp, source, text, in_reply_to_status_id, in_reply_to_user_id in the twitter_archives

4. Clean the wrong numerators - the floats on the one hand, the ones with multiple occurence of the pattern on the other, tweet_api_data should be part of twitter_archive. This is a tidiness issue. We should merge the tweet_api_data table to the twitter_archive table, joining on tweet_id and id.

5. Extract the source from html code Right now the source column is not giving us any useful information while looking at it. Because the relevant information is always between two "> <", the information will be easy to extract.

6. Split the text range into two separate columns

7. Remove the "None" out of the doggo, floofer, pupper and puppo column and merge them into one column. We want to reduce the columns into one for easier analysis. For

that we have to remove the None with "" at first to concat the columns together and afterwards with np.nan, so we could easily exclude these rows from a specific analysis.

8.Remove the wrong names of name columns Some dogs have name a, an, and the in twitter_archives_clean table. This must be a mistake. In order to increase the quality of the data, we should replace 'a', 'an', 'the' to nan.

9. Reduce the prediction columns into two - breed and conf In the next step we want to reduce the prediction columns into two - breed and confidence. The columns are already sorted by confidence. We will take the most likely prediction for each row which is supposed to be a dog.
10. Clean the new breed column by replacing the "_" with a whitespace and make them all lowercase Now that we have our reduced column, we have to clean it for consistency.

Some rows have invalid ratings, with rating numerator less than 10 or denominator not equal to 10. To solve this, we can either develop a better function to extract the rating from the status or we can drop rows with such occurrences.

.

.