Report on WeRateDogs data wrangling

To wrangle WeRateDogs Twitter data, I went through 3 consecutive steps; gather, assess, and clean datasets.

Firstly, gathering 3 datasets from 3 difference resources:

1- Enhanced_Twitter_Archive (csv file), provided by Udacity (downloadable) and contains basic tweet data for 2356 records/tweets including information such as tweet's text, rating, name, stage (i.e. doggo, floofer, pupper, puppo).
2- Image_Predictions (tsv file), obtained by using 'requests' library and the provided url. The resultant table includes image predictions (the top three only) of each tweet id, image URL, and the image_num that corresponded to the most confident prediction.
3- Additional information were gathered from Twitter API as json_txt file using the authentication keys provided by Twitter API. df_api was filtered to include only id, retweet_count and favorite(like)_count columns.

The three datasets were converted into pandas dataframes for assessing/cleaning/analyzing/visualization.

Secondly, assessing the dataframes for quality and tidiness:

Now, we have 3 dataframes; df_archive, df_image_predictions, df_api

The dataframes have been explored manually and programmatically. The manual assessment was based on opening the files either in jupyter notebook web-browser or from saved files on my computer, then scrolling up and down to get familiar with the data structure. Regarding the programmatic assessment, that was performed by using different methods in pandas such as; .head(), tail(), .sample() , .shape to get an overview of the data structure. Also, .info() was used to check the data entries for each variable along with their data types, .duplicated() to check duplicated records , .isnull() to check null entries, and .value_counts() for specific variables to deepen in the data details. Though, I did not focus on .describe() since most numerical data are available in rating variables that have outliers (probably, as erroneous values or exaggeration from the dogs' lovers). So, I was happy with sorting out the rating score as a string.

Thirdly, cleaning the dataframes by using (Define-Code-Test) protocol:

Quality (create a copy of each dataframe to work with, so we still have the original data)

1- Quality_1 (validity/consistency): dtype of tweet_id in the three dataframes is integer, so convert into a string to avoid performing any mathematical calculation on that column. The dtype of timestamp column in df_archive is object, then convert into datetime (might be used in the future) (use as.type()).

2- Quality_2 (consistency): rename id column in df_api as tweet_id (parent key), for consistency with other dataframes (use .rename()).

3- Quality_3 (completeness/consistency): some data entries in df_archive don't have image_url, so filter records with incomplete information by using tweet_id in df_image_predictions as a guide (make a list of tweet_id in df_image, then clean using .isin()).

4- Quality_4 (accuracy): Remove retweets and replies from df_archive since we are interested in only the original tweets (assign the retweets and replies entries, each to a new variable then drop them from df_archive).

5- Quality_5 (consistency): filter df_image_prediction from retweet and replies records by matching against the cleaned df_archive using tweet_id as a guide (use isin()).

6-Quality_6: in df_archive, drop columns; in_reply_to_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id and retweeted_status_timestamp, since they are redundant, and useless (use .drop()).

7- Quality_7: replace 'None' values in the last 4 columns (dogs'_developmental _tages) with pandas NaN (first, replace 'None' with empty spaces .replace(), then fillna after merging).

8- Quality_8: replace erroneous values in name column by first creating a pattern from the tweet text, then a for loop to replace 'a', 'an' names with their correct values from the text. Then replace 'None' entries with NaNs.

9- Quality_9: replace erroneous entries in column name with correct values extracted from the text pattern (.str.extract().

10- Quality_10 (accuracy): extract source names from the HTML tag (.str.extract()).

Tidiness

1- Tidiness_1: In df_archive, combine 4 columns of dogs' stages in one single column (one variable) by creating a new column, dog_stage and add the data string from 4 columns into it, then fill in the empty spaces with NaN.

2- Tidiness_2: In df_image_predictions, rename columns (p, p_conf, and p_dog) with more descriptive names to be more understood (use .rename(columns={'oldname':'newname'})).

3- Tidiness_3: Each type of observational unit forms a separate table, i.e., df_image_predictions and df_api are displaying some extra attributes of the tweets in df_archive. So, after cleaning, combine the 3 dataframes into one pandas dataframe (use .merge() on tweet_id).

Storing data

After cleaning and merging the dataframes, save as (twitter_archive_master.csv) to act on.