WeRateDogs Twitter Wrangling Report

The first step was to gather the data. Data gathering entailed that three pieces of data were downloaded and imported from three different sources using three different methods. First, the file twitter-archive-enhanced.csv was provided in the project details. This file simply had to be uploaded to the Jupyter notebook and then read in as a pandas dataframe using pd.read_csv. The second file was the image-predictions.tsv file. This file was downloaded programmatically using requests and then read into a pandas dataframe using pd.read_csv. The final piece of data was the Twitter archive. For this final piece of data, Tweepy was used to access the Twitter API to download the JSON for each tweet as a line in a txt file, tweet_json.txt. Then, pd.read_json was used to read in each line of JSON data in the tweet_json.txt file as a line in a pandas dataframe.

After gathering the data, it needed to be assessed. Assessment of each dataframe was conducted using both manual and programmatic methods. To make assessment easier, the pandas display options were set to show a max of 32 columns (in order to display all columns of the tweet_data dataframe) and the max column width was set to 500 to show the entire contents of the larger cells. Commands such as .head(), .info(), .sample(), .describe(), and .value_ counts were used to display sections of the dataframes for manual assessment and to show information and details from the dataframes for programmatic assessment. During assessment, about 16 quality issues and 4 tidiness issues were discovered.

The quality and tidiness issues discovered during the assessment phase were then addressed in the cleaning stage. First, copies were made of all three dataframes to preserve the originals. The first issue addressed was a tidiness issue that was fixed by dropping all unnecessary columns from the tweet_data dataframe, leaving only the id, favorite_count, and retweet_count columns. A quality issue was then addressed by converting the id column of tweet_data to strings instead of integers, and then changing the name of the id column to tweet_id to maintain consistency with the other two dataframes. The next issue addressed was the tidiness issue of having stage classifications as four different columns. To fix this, first the “None” entries were converted to NaNs, then the .apply function was used to apply a lambda function to the stage columns to combine them into one column called “stage” while dropping NaNs. The doggo, floofer, puppo, and pupper columns were then dropped. This still left the issue of multiple stage classifications, which was then fixed by converting any instance of more than one stage term to “multiple”. Next, the columns in twitter_archive were converted to appropriate data types to solve a quality issue (change tweet_id to string, change stage to categorical, change timestamp to datetime). Then, retweets and replies were removed from twitter_archive. The next step was to tackle the image_predictions dataframe. First, any rows that did not have a legitimate dog breed as any of the top three predictions were dropped. Then, a “breed” column was made and filled with the greatest probability legitimate dog breed prediction for each row using a for loop with if, elif, and else clauses. The tweet_id column of image_predictions was also converted from integers to strings to address another quality issue. The main tidiness issue was then addressed by merging all three dataframes and then dropping the unnecessary columns to create the twitter_archive_master (tam_df) dataframe. The merge was performed using an inner join to merge the twitter_archive with image_predictions, then another inner join to merge tweet_data with the other two. After joining, all unnecessary columns were dropped leaving only the following: 'tweet_id', 'timestamp', 'text', 'name', 'breed', 'stage', 'rating_numerator', 'rating_denominator', 'favorite_count', 'retweet_count', 'expanded_urls', and 'jpg_url'. The next issue cleaned was to fix instances where the ratings were extracted incorrectly. This was done by first splitting the “text” column into two columns, “text” that contained only the tweet text, and “text_url” to contain the url at the end of the text. Regex was then used to find instances where multiple forward slashes occurred in the text column and these entries were examined and fixed as needed. A similar approach was used to fix instances with decimal rating numerators. The final quality issue addressed was to fix incorrect dog names. There was one instance where “O” was corrected to “O’Malley” manually. The remaining name issues were handled by first using loc and logic statements to make lists of entries where the name was lowercase or NaN, and where either of those conditions occurred as well as either “named” or “name is” appeared in the text. For loops were then used to replace lowercase names and NaNs with names if names were found, or with “None” if no name was found. The resulting clean tam_df dataframe was then stored using df.to_csv to save as twitter_archive_master.csv.