Wrangling Report

After assessing the WeRateDogs Twitter data, I identified the following wrangling tasks. This report contains a brief description of the wrangling work for each task.

Quality

WeRateDogs Twitter archive (df_tw_archive)

1. **Statuses where retweeted_status_id or in_reply_to_status_id are not null, are not original ratings.**

   One of the wrangling requirements was to only include statuses that are original ratings. I used the pandas "notna" method to remove statuses that were retweets and replies.

2. **Timestamp column should be datetime data type. Also, data in the floofer column should be boolean data type.**

   I used the pandas "to_datetime" method to cast the timestamp column as a datetime object.

   Values for the floofer column are "floofer" or "None." Since there are only two values, a boolean data type is appropriate. I used pandas ".loc" notation to define a new column called "is_floofer." Its value is either True for floofers, or False for non-floofers.

3. **When the status has other numbers separated by slashes, these are incorrectly captured as the rating. Tweet ID's where this happened: 740373189193256964, 682962037429899265, 666287406224695296, 810984652412424192.**

   Visual assessment showed unusual values for rating_numerator for these tweets. I checked the statuses and used pandas ".loc" notation to set the correct value for the rating.

4. **When the status uses a float for the rating, the rating_numerator is captured incorrectly. Tweet ID's where this happened: 778027034220126208, 680494726643068929, 786709082849828864.**

   Similar to the wrangling task above, visual assessment revealed incorrect ratings if the rating was a float. Pandas ".loc" notation was used to set the correct rating.

5. **Tweets with a rating_denominator greater than 10 are for statuses with multiple dogs.**

   I used the pandas "query" method to filter on statuses with a rating_denominator less than or equal to 10. This way, all remaining statuses are for a single dog. This will make it easier to do analysis.

Tweet image predictions (df_tw_img_preds)

6. **img_num column should be category data type.**

   Img_num only has 4 different values, so the category data type is appropriate here. I used the pandas "astype" method to cast this column as a category.

7. **If p1_dog, p2_dog, or p3_dog are False, the predicted dog breed is invalid for our analysis.**

I filtered on rows with invalid dog breed names using pandas ".loc" notation. Breed name was replaced with NaN here, using the numpy "nan" method.

8. **p1, p2, and p3 columns have inconsistent capitalization.**

   For easier readability, I replaced underscores with spaces with the pandas "str.replace" method. Then, I capitalized the first letter of each word using the "str.title" method.

Tidiness

WeRateDogs Twitter archive (df_tw_archive)

1. **Pupper, puppo, and doggo info in separate columns.**

   The original columns each indicate the stage of a dog's life. To meet the requirement for tidy data, I reorganized this data into one column called "dog_stage." At first I used the pandas "melt" method. However, this resulted in extra rows, as some dogs were classified as a doggo and either puppo or pupper. Instead, I combined the contents of doggo, pupper, and puppo using string concatenation. Then I removed the "None" part with pandas "str.replace."

Twitter API data (df_tw_api)

2. **The data in df_tw_api should be in df_tw_archive.**

   The Twitter API data contained favorite_count and retweet_count. This extends the data for each status from the WeRateDogs Twitter archive, so it made sense to combine these dataframes. I used the pandas "merge" method for this.

Image predictions (df_tw_img_preds)

3. **The data in df_tw_img_preds should also be in df_tw_archive.**

   The image predictions have a row for each tweet_id, so they are part of the same observational unit as the other dataframes. As with the task above, I used the pandas "merge" method to add the image prediction data to the master dataframe.