WeRateDogs Twitter Data Wrangle Process

By Ryan Gee

The necessary datasets for the WeRateDogs analysis were spread across 3 different sources:

1. Archived twitter data was given in the form of a csv file
2. Image prediction data was stored as a tsv on Udacity's server
3. Retweet and Favorite counts resided on Twitter and was extracted via an API

For the Data Gathering portion of this project, I wanted to practice writing object-oriented code. I started by creating a class called WeRateDogsDataPuller whose main function was to extract the data from the 3 different sources and combine them into one Pandas data frame. The "Archived twitter" data was very straight forward and just utilized the Pandas's read_csv() function to convert the given csv into a Pandas data frame. The "Image Prediction" data was downloaded from Udacity's servers by using Python's built-in requests library. Once downloaded, I used the read_csv() function to convert the tsv file into a data frame. The last dataset, "Retweet and Favorite Counts", was the most involved wrangling task. The first step was to create a developer's account with Twitter, so that I could access their API. Once I gained access to the API, I used the get_status() function in the Tweepy library to pull a JSON for each of the tweet_ids in the "Archived Twitter" dataset. This JSON included a ton of data, but the most important data was the retweet_count and favorite_count data. I created a new class called Tweet that parses and extracts the important data from these JSONs and then writes the data to a new line in a specified text file. Once the text file was fully populated, I created a Pandas data frame by reading each line of the text file. Finally, I used Pandas's merge function to "left join" each of the datasets together. I used a "left join" to keep all of the data even if there was some missing data in the "Image Prediction" or "Retweet and Favorite counts" data.

To clean and tidy the dataset I first split the dataset into a twitter dataset and an image predictions dataset, since they are two different observational units. Next, I used Panda's melt() function to fix the common tidy issue: "Column headers are values, not variable names". Essentially, instead of having a separate column for each prediction_rank (ex: p1, p2, p3), I used the melt() function to create a new row for each prediction_rank.

The twitter dataset also had some missing data specifically in the *in_reply_to_status_id* and *retweeted_status_id* columns. However, since this project was supposed to focus on just original tweets, I decided to only keep rows with NaN in those columns. Then I decided to remove those columns from the entire dataset, thereby keeping only original tweets and removing missing data.

Next, I fixed some accuracy issues that may have slightly skewed the analysis. For example, I found that the parsing logic used to extract the rating numerator was incorrect when it contained a decimal. There was one example where the rating was

9.75/10, but the logic parsed out a rating of 75/10. In order to fix this, I used regex and some functions like find() and extract() to handle these edge cases.

Finally, I fixed some consistency issues that were negatively affecting the readability of the dataset. For example, the dog breeds in the image predictions dataset had irregular capitalization (i.e some were capitalized and some were not). These consistency issues were fixed easily by using functions like apply() to change the values of each row in a column.

Overall, this was a comprehensive task that requires knowledge of the Pandas library, Request Library, and API's in order to complete.