Wrangle Report

Data Gathering

I did the Data Wrangling Project as the following order: Gathering, Assessing, Cleaning.

First, I did the Gathering. I used 'requests' library , downloaded the contents of image prediction csv, and saved it in the local directory as file named 'imagePredict.tsv'.

Since 'Twitter-Archive-enhanced.csv' was given for free, I saved it and loaded with 'pd.read_csv' function.

Lastly, since I got rejected to receive tweeter-development account, I manually obtained the 'tweet_json.txt' and loaded with Pandas DataFrame.

Data Assessing

I used Excel to have a quick look the file structure and contents. I spotted that some errors like **'rating_denominator'** and '**rating_numerator'** values are way above or below the average (or even decimal points)

Some names were not correctly extracted. I spotted some names like 'a', 'not' etc.

Some column headers are values, not variable names (doggo, floofer pupper, puppo instead of 'stage')

Some unnecessary columns were also found.

Secondly, I used .info() function to look for any null values and data types.

I found out that 'timestamp' column can be converted to 'datetime64' instead of just 'obeject=string'.

Also, I thought 'in_reply_to_status_id' and 'retweeted_status_id' columns can be used to distinguish the original tweet from 'retweet' and 'replytweet'

Data Cleaning

Data Tidiness

I tried to approach this problem with using 'pd.melt' function, but then since some rows have multiple values, I could not find effective way to solve this issue.

Then I came up with another idea, which was to add the whole 4 types (doggo, floofer, pupper, and puppo) and manually replace the values.

.replace('doggofloofer', 'doggo/floofer')

.replace('doggopupper', 'doggo/pupper')

.replace('doggopuppo', 'doggo/puppo')

.replace('flooferpupper', 'floofer/pupper')

.replace('pupperpoppo', 'pupper/puppo')

I stored this new string in the new column names 'stage' and dropped the 4 columns (doggo, floofer, pupper, and puppo)

Then I merged the three separate tables using ‘tweet_id’ as the key column.

Data Quality

----Original Tweets----

I sorted out the rows that are original tweet by putting conditions on the dataFrame:

cond1 = df_total['in_reply_to_status_id'].isnull()

cond2 = df_total['retweeted_status_id'].isnull()

df_total = df_total[cond1&cond2].copy()

Then I dropped unnecessary columns and came up with **['id', 'timestamp', 'text', 'rating_numerator', 'rating_denominator', 'name','stage','retweet_count','favorite_count','img_num','p1','p1_conf'],** which I would assume to be essential for our project.

----rating_numerator----

Then I worked on tracking outlier in 'rating_numerator' values

After I had a brief look on the 'rating_numerator' values and 'value_counts()' function, I found out that there are also numerator values with decimal pts that were not correctly extract.

So, embracing Udacity's reviewer's feedback, I used the following code to correctly extract the 'rating_numerator' values form the 'text'

df_total['text'].str.extract('((?:\d+.)?\d+)\/(\d+(.\d+)?)', expand=True)

Then I changed the datatype to 'float' for numerator values and correctly raplce them with the newly extracted values

----rating_denominator----

Same for 'rating_denominator' values, I spotted outliers with function '.value_counts()' and fixed the wrongly recorded denominator values by manually looking at the text. For the data that did not have any ratings, I replaced the value with 'np.nan' instead of 'None'. And I also changed its datatype to 'Int64' instead of 'int64' because 'np.nan' was regarded as float64.

----Fix the Wrong data in 'name' columne - Strange names like ['a', 'None', 'Not', 'actually', 'all']----

I used .value_counts() function to look for wrongly recorded names. And I found unreasonable names like 'a', 'None', 'Not', 'actually', 'all'. By manually looking at the text, it seemed like some tweet do not contain the names at all and some tweets have different starting sentences which caused the wrong extraction of the name. I found a pattern that WeRateDogs sometimes puts the name after '~ ~ ~named {dog name}'. So I did some python string manipulation to extract the names for these irregular cases