Catherine Taft 8/5/22

Udacity Data Analytics Nanodegree

Project 4 – Wrangle a Dataset: WeRateDogs

Wrangle Report

As described in the Udacity classroom, the wrangling process contains three parts: gathering, assessing, and cleaning. I will outline my process and thoughts with each of these parts below.

Gathering

The gathering process was fairly straightforward. We had one dataset that was given to us which I believe was sent to Udacity by the WeRateDogs account holder and then altered slightly for this project. Additionally we were given a file with some image predictions that was created by the Udacity instructors, where they ran a ML algorithm to try to predict the breed of dog in some of the photos. The third dataset had to be acquired through the Twitter API. I've used the Twitter API before, for a school project actually (I'm in college for data analytics), so I knew how to use that. The biggest problem was getting them to accept my application elevated status for this project. They didn't seem to understand what Udacity was and it was very frustrating. Once I eventually got that access I was able to pull what I wanted from the API, though.

Assessing

Having three datasets, two of them a bit large, made the assessment more challenging. I ended up coming up with more than the required eight quality issues and two tidiness issues, and could have kept going but I had to stop somewhere. I separated out the issues by dataset and they were as follows:

<u>QUALITY</u>

<u>dogs_master (master dataset from Udacity)</u>

1. The project specified that retweets should be dropped. This was done by just finding the columns with the retweet information, and subsetting the dataset to include only rows where the retweet columns had null values, and making that an inplace edit.
2. Several unnecessary columns needed to be dropped, including all the retweet information columns after the above was completed.

3. However the original dataset was parsed, it pulled some random words into the data thinking it was the name of the dog. Words like “an”, “actually”, etc. I called up all the unique names and looked down the list and noticed that other than “None,” which was the null placeholder when a dog’s name was not mentioned, all the non-names were in lowercase. So I wrote a for loop to go through the names in the dataframe and change any lowercase words to “None.”
4. The “timestamp” column was a string rather than datetime, which would be problematic if I decided to run any datetime operations, so I changed that to datetime and removed the time part because it was cluttery and unnecessary. I ended up having to convert it back later because using .date to truncate the timestamp converted it to a string.
5. I changed the tweet ID to a string from an int, because there will be no math operations performed on tweet IDs.
6. I renamed the timestamp column to tweet_date to match the dataset from the API, because I think tweet_date is a better name and since they’ll be joined, the names need to match.

dogs_master_API (the stuff I pulled from the API)

7. Similar to dogs_master, I needed to change the timestamp from a string to a datetime and remove the hours and minutes.

Image_predictions (the neural network’s prediction of dog breeds)

8. I wanted to drop some columns. I was only interested in the NN’s first prediction but it had made three, of decreasing confidence. I was only interested in the one it was most confident in.
9. Similar to dogs_master I needed to change the tweet ID to a string.
10. Change p1 column name to breed_prediction
11. Change p1_conf column name to prediction_conf

TIDINESS

First I had to merge the dog categories (reviewers keep calling them “stages” but I don’t like that) into one column. This is my third submission – the first time I used merge for all three steps of this which created a ton of duplicate columns which I was unable to drop because I was careless when I merged the pupper/woofer/floofer columns. In my second submission I chose not to merge the categories (it doesn’t say I HAVE to) and used join for the dfs, but was told that I did have to merge the categories. So this time I have been more careful about it (the first time I didn’t use any parameters in drop_duplicates) and I did NOT use merge to join the dataframes. So just to control this I did it in three steps and looked at the aftermath of each step instead of all at once.

1. So first I did use merge, and I immediately addressed dropping the duplicate rows, remembering the parameters, and I subset based on tweet ID.
2. Then I joined dogs_master to dogs_master_API to create the fierce sounding Ultimate Dogs Master. I did not use merge to avoid creating duplicate columns.
3. Then I merged the image predictions dataframe into Ultimate Dogs Master. Since the only common column was the join column of tweet ID, there were no duplicate columns and since I didn’t use merge there were no duplicate rows this time.