wrangle_report

June 23, 2022

0.1 Reporting: wrangle_report

0.1.1 Gathering:

- To start, I needed to import everything I would need to run my codes, such as: matplotlib, pandas, json, requests, and later on datetime.
- Next, I read the csv provided by WeRateDogs and read it as my first dataframe. To get my second dataframe, I used the request.get() function to read the image prediction tsv and then with a file handler wrote it to a file as a tsv. All that was left to do was to use pd.read_csv function with specifying the tab separator and read it as my second dataframe.
- I didn't get the access to Twitter's api, so I had to read the json file provided by Udacity. I used the requests.get() function and its content to download the json file; I then read it with panda's read_json() function to get my third dataframe.

0.1.2 Assessing:

When assessing, I was looking mainly for completeness issues since these files were gotten separately.

0.1.3 Cleaning:

The quality issues I discovered were too many to clean at first, but after thought, I figured that some I could just leave them unclean and it would harm no one. I went through the dataframes one by one documenting the issues I had found in the assessing stage.

- With the first dataframe, frame_og, I found out:

  It had the timestamp column as an object rather than a datetime; I later on cleaned it with pandas' to_datetime() function and read the result to the same column in my dataframe.

  It had another datetime column as an object called retweeted_time_stamp, same thing as before, I used pandas' to_datetime function to read it as the same column in the dataframe.

  The numerator column and its relative denominator column had huge gaps in their values that I had to normalize at least one of them. As it states on their website, WeRateDogs normally rate anything to 15 over 10, So I decided to normalize the denominator to 10 for the whole column.

- With the second dataframe, frame_image, I found out:

  It had a text_range column as an object which I needed to visualize later, so I transformed it to an integer by extracting its values array-wise and subtracting them.

  It had several columns that came from the machine learning algorithm about probabilities and a dog breed type associated with the highest probability. For me, I just needed to extract just that breed name for where the probability was higher, and I did that with several if conditions and read finally that name to its column called breed_type.

- With the third dataframe, frame_tweepy, I found out:

  It had a hashtag element I wanted to extract, I used the json's dictionary read-like method to read and assign it to a column in the dataframe.

  It had a retweet_status that signaled if a tweet was a retweet or not, and I used that column to know when a tweet was a retweet or not. If it was, the value wasn't empty; so, I cleaned all non-null values from that column. It had so many other non-needed columns that I dropped to save space and simple visibility.