wrangle_report

April 21, 2020

1 Introduction

In this document, I will briefly overview my approach to the data wrangling process. Before wrangling the data, I imported all relevant packages (warnings, requests, os, tweepy, json, numpy, and pandas). I also disabled the truncation of dataframe column text for easier analysis and suppressed warnings.

2 Gather

The data itself could be found in three sources – (1) a .csv file, provided by Udacity; (2) a hyperlink, and (3) Twitter:

1. Using the function pd.read_csv(), I saved the .csv file to a dataframe.
2. I then opened the hyperlink using python's requests package and saved the retrieved .tsv file to a dataframe by once again using pd.read_csv().
3. After creating a developer account on Twitter and accessing my consumer keys and access tokens, I was able to scrape the Twitter archives and collect the .json files for all tweet IDs present in the first .csv file.
    - I have since removed my credentials and to re-run the notebook, another user's credentials will need to be entered. Alternatively, the user can access the "extra_info.csv" and "errors.csv" files.

Unfortunately, not all of the tweets were accessible. Any tweet ID which raised an error was put into its own dataframe. I examined this dataframe and found that in 24 of the 25 cases, no tweet (referred to by Twitter as a "status") was found matching the tweet ID. It's possible that those tweets were deleted or the tweet ID was incorrectly recorded. In the last case, I was not authorized to see the tweet – perhaps because someone set their account to private. Either way, I was unable to work around these errors and unfortunately the 25 tweet IDs were later deleted from the final datasets.

3 Assess

Once all the data was collected, I used a combination of visual and programmatic tools to assess the dataframes. The majority of quality issues were found in the archive table, while most tidiness issues appeared in the extra_info table.

3.1 Quality

- archive table
  - NaNs not recognized (doggo, floofer, pupper and pupppo columns)
  - Erroneous datatypes (timestamp)
  - Missing data
  - Outliers (rating_numerator and rating_denominator columns)
  - Incorrect entries in name column (e.g. 'one' index #924)
  - Mix of tweets, retweets and replies
  - Not all rankings are for dogs
- images table
  - Mix of upper- and lowercase string values (p1, p2 and p3 columns)
  - extra_info table
  - Mix of tweets and retweets
  - Not all rankings are for dogs

3.2 Tidiness

- archive table
  - Type of dog variable split into four columns (doggo, floofer, pupper and puppo)
- extra_info table
  - Text column includes dog name and rating variables
  - Tweet_id, text columns duplicated in archive table
- other
  - Final merged dataframe of archive and extra_info tables contains messy data

4 Clean

After making copies of the three dataframes, I followed the programmatic data cleaning process as described in Lesson 4 Part 5. I first translated all of my observations regarding the datasets' quality and tidiness into actionable tasks ("Define"), then executed the tasks in Pythonic code ("Code") and finally employed a variety of techniques to check that the changes were correctly made ("Test"). The cleaned datasets were then split into two smaller, tidier tables. Finally, a combined master version and the two smaller versions were exported as .csv files.

5 Analysis

The remainder of my notebook is dedicated to reshaping and filtering my dataset for the purposes of analysis. I selected the following metrics: + Development of ratings over time + Average rating per dog stage + Average number of shares per post + Top ten posts