Data Wrangling Report

Introduction

This project is focused on wrangling and analyzing data of WeRateDogs. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. As of November 2020, WeRateDogs has currently nearly 8.8 million followers. WeRateDogs asks people to send photos of their dogs, then tweets selected photos rating and a humorous comment. Dogs are rated on a scale of one to ten, but are invariably given ratings in excess of the maximum, such as “13/10”. Why? Because "they're good dogs Brent."

This report contains all the steps I have taken in gathering, assessing, cleaning and saving the dataset in the data wrangling process. The original data from the twitter archive is messy and dirty and it should be handled properly without losing important information. The goal is to create clean master dataset that can be used for further analysis to gain “Wow!"-worthy insights about WeRateDogs.

Data Gathering

Data were collected from different sources in different file formats using different gathering methods. Each piece of data was imported into a separate pandas DataFrame.

1. Download data manually
   - WeRateDogs downloaded their Twitter archive and sent it to Udacity via email. Therefore, it can be downloaded manually as a csv file with the name *‘twitter-archive-enhanced.csv’*. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017.

2. Download data programmatically using Requests library
   - Every image in the WeRateDogs Twitter archive is ran through a neural network that can classify breeds of dogs. The results are full of image predictions alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction.
   - The file that contains the results is hosted on Udacity's servers. It was downloaded programmatically using the Requests library and the url given by Udacity to access to its servers. The response content was written into a file named *‘image-predictions.tsv’* and imported into a pandas DataFrame.

3. Gather additional data via the Twitter API
   - Twitter API requires users to be authorized to use it. Before running API querying code, own Twitter application needs to be set up.
   - Python's Tweepy library was used to query Twitter's API for each tweet's JSON data included in the WeRateDogs Twitter archive. Tweet data are stored in JSON format by Twitter. These data include retweet count, favorite count, etc.
   - A code timer was used for sanity reasons because it took more than 30 minutes to query all of the tweet IDs in the WeRateDogs Twitter archive.
   - Also, try-except block was used while querying. If the attempt fails and an error is encountered, the tweet_id and the error is printed out.
   - Each tweet's entire set of JSON data was written to its own line and stored in a file called *tweet_json.txt*. Next, the JSON objects in this file were appended line by line into a list of dictionaries. This list of dictionaries was then used to create a pandas DataFrame with tweet ID, retweet count, and favorite count.

Data Assessing

All three DataFrames were merged into one single DataFrame so that it can be assessed at once. The dataset was first assessed visually and then programmatically to identify quality and tidiness issues. In this project, only records with original ratings (no retweets) and with image were kept. In this dataset, not all are dog ratings and some are retweets.

1. Assess visually

Each piece of data was printed out in the Jupyter Notebook for visual assessment purposes. Three issues were found:

- Two variables (text and url) form `text` column
- One variable forms four columns (dog stage - doggo, floofer, pupper, puppo)
- Unnecessary HTML code in the `source` column

2. Assess programmatically

- A summary of datatypes and non-null values was displayed to identify erroneous datatypes and variables with missing values.
- The number of each unique values of `rating_numerator` and `rating_denominator` were first calculated to check for outliers. The outliers were then explored by looking at the `text` column to check whether the ratings were extracted correctly.
- `tweet_id` and `jpg_url` variables were checked for duplicates.
- In the `name` column, there were several values that are not dog names, like 'a', 'the', 'such', etc.

Data Assessing Summary
Quality - Completeness - Irrelevant records (retweets) and corresponding columns - Missing prediction results - Missing values in `retweet_count` and `favorite_count column`

Quality - Validity, Accuracy, Consistency
- Invalid names in the `name` column
- Some values of `rating_numerator` and `rating_denominator` are wrong
- Erroneous datatypes
  • The ID fields, like `tweet_id`, `in_reply_to_status_id` etc. are integer not string
  • `timestamp` is a string not datetime
  • `rating_numerator` and `rating_denominator` are integer not float
  • `retweet_count`, `favorite_count` and `img_num` are float not integer

Tidiness - Two variables (text and url) form `text` column - Multiple prediction columns (`p1`, `p2`, `p3`) - One variable forms four columns (dog "stage" - doggo, floofer, pupper, puppo) - Unnecessary HTML code in the `source` column

Data Cleaning

The data quality and tidiness issues mentioned above were handled using pandas. Each issue was cleaned step by step following Define, Code and Test sequence.

1. Remove retweets records
2. Remove records without predictions
3. Remove records without retweet and favorite counts
4. Combine the `doggo`, `floofer`, `pupper` and `puppo` columns to a `dog_stage` column, convert to category datatype and drop the corresponding columns
   - There were some records with two different dog stages. In order to determine the correct dog stage, the images of those records were checked one by one because sometimes the text was misleading.
5. Reduce the prediction columns into one `breed` and one `conf` column
   - There were three predictions for each records. The prediction with the highest confidence coefficient was chosen.
6. Extract all shortened URLs from `text` column using regular expression, create new `shortened_url` column and replace this url in the `text` column with empty string.
7. Extract the name of the source from the HTML code in the `source` column using regular expression
8. Replace the invalid names with *None*
9. Convert the erroneous datatypes to the correct datatypes
10. Replace the wrong ratings with correct ratings by extracting it from the `text` column using regular expression
11. Remove the records that have non 10 denominators
    - Even though having non 10 denominators is not an error, these records were removed for a fair comparison purpose later in the exploratory analysis.