We Rate Dogs: Wrangle Report

Tim Quan

***Udacity**: Data Analyst Nanodegree Program*
***Project 03**: Wrangle and Analyze Data*

March 2022

0.1 Introduction

In this stage of the project, we gathered/wrangled data from 3 sources:

- **twitter_archive_enhanced.csv**
- **image_predictions.tsv**
- Twitter API

0.2 Twitter Archive Enhanced (twitter_archive_enhanced.csv), df_archive_en

This is a file provided as a project resource. The file was manually downloaded from the repository and stored locally.

For use in this project the file was imported into a Pandas DataFrame, **df_archive_en**.

0.2.1 Assessment

The resulting dataframe contained 2356 rows and 16 columns/variables. Most variables are metadata gleaned from tweets.

Quality Issues, Cleaning & Tidying

1. As imported to dataframe, some columns were interpreted as inappropriate data types. This was resolved by setting columns to more appropriate data types as follows:

column	adjusted dtype
*tweet*_id	string
timestamp	datetime
source	category
retweeted_status_id	string
retweeted_status_user_id	string
retweeted_status_timestamp	datetime
rating_numerator	float
rating_denominator	float

2. *retweeted_status_id*'s can represent self-retweets. This means the retweeted records could be rows of duplicate data. We can identify the duplicate data by comparing the retweet_status_id column to the tweet_id column. To resolve this, we dropped the rows that contain tweet IDs which are also in the retweet_status_ids list.

3. The rating scale appears to be inconsistent. There are outliers in this data that need to be removed for analysis/visualization. There is some division by 0 in the denominators. Ratings that contained a decimal place were not parsed correctly.

4. There are tweets in **twitter_archive_enhanced.csv** that do not have corrosponding records in **image_.tsv** and vice versa. After merging the tables, these rows were identifiable by tweets that do not have a value for *jpg_url*. These rows were dropped.

5. The column *source* contains interesting data about devices but is not readable. We parsed the string data from the source column into something human readable and managable, then changed the datatype to category, and finally renamed the column to *source_device*.

6. Columns *doggo, floofer, pupper, puppo* are categorical values of the same variable. We have merged these values into one column of type category.

0.3 Image Prediction (image_predictions.tsv)

This remote file location was provided in the project instructions. It (the file) was downloaded programatically and stored locally.

The data was ingested into a Pandas DataFrame for assement and use.

0.3.1 Assessment

The resulting dataframe contained 2075 rows, 12 columns/variables. This delimited dataset contained both metadata and additional data added by the course authors.

The images from tweets in the dataset were run through a predictive image neural network; the results are the 3 most likely detected breed of dog (or, in some cases random object), and the p-values associated with the likelihood of correctness of the predictions.

Quality Issues, Cleaning & Tidying

1. As imported to dataframe, one column was interpreted as an inappropriate data type. This was resolved by setting the column to more appropriate data type as follows:

column	adjusted dtype
tweet_id	string

2. There appear to be 66 duplicate images/image predictions, these can be identified by *jpg_url*. By identifying duplicate *jpg_url*, a list of duplicate tweet IDs was gleaned and dropped.

3. There are 324 records where no dogs were detected whatsoever. These records were dropped.

4. There are 3 separate predictions for each dog photo. The dog prediction columns could use meaningful names. We took the most likely of p1, p2, p3, moved it to a new column, *predicted_breed*, and drop the others. Also we can drop *p1_dog* because know all are suspected to be dogs now.

Rename:

original column name	new column name
p1	*breed_prediction*
p_conf	*prediction_confidence*

0.4 Twitter API (json_dicts.txt)

As instructed in the project description, we used Tweetpy to access the twitter API.

The first step in this process was gaining Twitter API access. The first level of developer access was trivial to apply for and obtain; I was able to use the 'explanation' recommended in **Additional Resource: Twitter API**, verbatim, in the application and it was granted immediately. However, some of the methods tweepy uses to lookup tweets - namely tweepy.getstatus - requires elevated API access. For this level of access, it would appear there is a human making approvals. At first I attempted just resubmitting the same description, but over the course of a day and several attempts, it became evident that a fairly detailed project proposal was required. Because the stakes at the time appeared to include being able to complete the nanodegree, and the process resulted in my account being suspended for suspicious activity for several hours, it was surprisingly stressful process, but after a day of persistence we had success.

0.4.1 Gathering via the Twitter API

Roughly, these were the steps and some notable details of gathering tweet meta-data via API:

- Initialize a tweepy API instance using the new developer keys.

  Because these credentials are private, this project is using a basic yaml 'config' file to store/retrieve as required, which (the yaml) will be omited from git and submission. In the init, the wait_on_rate_limit_notify parameter was important in order to handle the API limits.

- Iterate over over all the *tweet_id*s in **(twitter_archive_enhanced.csv)**, run the get_status method against each. This takes about 20 minutes in total, including waiting for API limits.

- Write the list of dicts to a local file, **json_dicts.txt**. This gives us some flexibility in terms of being able to avoid the long wait of the previous iteration process when we need to run all cells.

- Re-read the data from **json_dicts.txt** into a dataframe using column/variables *id, retweet_count, favorite_count* only.

Quality Issues, Cleaning & Tidying

- While there were up to 30 records that failed to be imported by API, this data came out relatively clean. The issues of missing records we were able to mitigate the issue of missing records by using an inner join when creating **df_master**.

0.5 Master Dataset (twitter_archive_master.csv, df_master)

Quality Issues, Cleaning & Tidying

As a cleaning and tidying issue relating to all 3 datasets, we have merged them into one DataFrame; **df_master**. The records were joined on column *tweet_id*.

The DataFrame was then saved locally to **twitter_archive_master.csv**.

This (the creation of our new master dataset CSV) was the final step and concludes the wrangling and cleaning processes. We now have a clean and organized dataset stored in **twitter_archive_master.csv** which can be imported back into DataFrame for further analysis.