Noelia Guzman
December 18, 2022

Wrangle Report

I. Data Gathering

In the context of this project, we were required to gather three data sets: ‘twitter-archive-enhanced,’ ‘image-predictions,’ and ‘tweet-json.’ The steps to gathering each data set are indicated below:

1. **Twitter Archive Enhanced CSV File:** This file was able to be gathered programmatically using the Request library, or manually with the resources provided.

2. **Image Predictions TSV File:** This file was downloaded programmatically using the requests library. It was hosted and only made available through Udacity servers. Udacity provided the URL that was used to execute the programmatic download of the file.

3. **Tweet JSON File:** This file was able to be gathered programmatically using the Request library and API access through a Twitter developer account. Another option was to manually download the file through resources provided by Udacity.

II. Data Quality Issues and Steps for Data Cleaning

After gathering the 3 datasets, they were merged into one dataset and saved locally. There were various quality and tidiness issues with the dataset that needed to be addressed. Below are just some of the issues that were found and cleaned throughout this project.

➔ **<u>Quality Issue #1:</u> Certain column names are not descriptive enough: img_num, p1, p1_conf, p1_dog, p2, p2_conf, p2_dog, p3, p3_conf, p3_dog**

Some of the column names ('p1', 'p1_conf', etc.) are not descriptive. They needed to be renamed the non-descript columns using .rename() and ensure we overwrite the old column names with 'inplace=True'

These columns were renamed to be easily understood:

- img_num = image_number
- p1 = first_prediction
- p1_conf = first_prediction_confidence
- p1_dog = first_prediction_is_dog
- p2 = second_prediction
- p2_conf = second_prediction_confidence
- p2_dog = second_prediction_is_dog

➔ **Quality Issue #2: Some column values for the column 'name' are invalid: Each value should either be the dog's name, or 'None'.**

Upon inspecting the column called ‘name’, it was found that the data was inconsistently formatted. There were random words in place of dog names (‘a’, ‘the’, etc.). It was also observed using .value_counts() that the invalid values began with a lowercase letter. To fix this, I used a for loop to iterate through the column ‘name’, and all values with a lowercase letter were replaced with ‘None’, as I did not have the data to verify the correct dog names that belonged to those dogs/entries of data.

➔ **Quality Issue #3: There are rows of data that represent retweets. These should be removed.**

Several columns in the dataset contain data that indicates the row represents a retweet, rather than an original post. These columns are: in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id, retweeted_status_timestamp. It was best to remove these rows of data as to not skew any analysis performed on the data. To clean this quality issue, a for loop was created to iterate through the dataset, and wherever there were not NaN values in the column ‘retweeted_status_id’ (since that would mean the row of data represents a retweet) that row was dropped.

After dropping the rows, any columns that held retweet data were dropped as well using .drop().

➔ **Quality Issue #4: Some columns are not the correct data type.**

After using .info() to inspect the columns in the dataset, it became clear that some columns had inappropriate data types, and that they would need to be adjusted. To do this, the .astype() function was used, and a dictionary was passed through it that indicated the column name and the data type to be converted to. The timestamp was separately converted using .to_datetime(). The columns which were converted were the following:

- ID columns --> string datatype
- Categorical variables --> category data type
- Time-related columns --> datetime datatype
- Count-related columns --> integer datatype
- Numerator and denominator --> float data type

➔ **Quality Issue #5: The ‘timestamp’ column contains data that could be split into different columns for readability and analysis**

The ‘timestamp’ column contains the date and time that can be split into two columns ‘date’ and ‘time’. To do so, we can create a variable 'date_time' which is an index object containing the date values present in each of the entries of the DatetimeIndex object.* We can then split the ‘timestamp’ column to create 'date' and 'time' using the attributes .date and .time. As to not have redundancy, we can then drop 'timestamp' from the data.

**Source:* https://www.geeksforgeeks.org/python-pandas-datetimeindex-date/

➔ **Quality Issue #6: For the dog breed prediction columns ('first_prediction', 'second_prediction', 'third_prediction') values contained unwanted characters.**

The breed prediction columns have values that contain unwanted characters. These include underscores instead of spaces, and lowercase letters at the start of the dog breed names. This was resolved using .replace() to overwrite ‘_’ with ‘ ‘.

- **Quality Issue #7: The HTML tags are in the values for the column 'source'**

The 'source' column contains HTML tags in the values which affect readability. To remove the HTML tags from column 'source' we can slice the values to exclude the tags at the beginning and end of the string values using .str.slice(), and use this to make a new column. Then, we can replace the remaining unwanted characters using .replace(). Once the values display with more readability, we can drop the original 'source' column.

- **Quality #8: The column 'text' contains a url.**

The 'text' column contains a url at the end, affecting readability. We can remove the url using a regular expression:
- r'https://\S*' will match any characters (.) that are not whitespace (\S) after the string 'https://',
- the * indicates that the pattern can happen more than once
- regex=True argument tells the str.replace method to interpret the pattern as a regular expression.

- **Tidiness Issue #1: There are 3 separate dataframes and they can be combined into one**

I began the project with 3 separate dataframes. For ease of analysis, I combined the dataframes into 1 using .merge() with a left join.

- **Tidiness Issue #2: There are individual columns for categories of dogs: doggo, floofer, pupper, puppo**

In this dataset, there are 4 columns that indicate a "level" or "category" of dog: doggo, floofer, pupper, and puppo. To make the dataset tidier, we can combine these 4 columns into one column called 'dog_categories'. We can do so by first replacing all 'None' values with spaces (to avoid confusing strings when we concatenate), then combine the columns using the '+' operator. We then will check the values of the new column to ensure the values are readable. Any values that need to be cleaned can be done so by using .loc to find where the values occur.