Project: Wrangling and Analyze Data

Wrangle Report

Table of Contents

Introduction

This document summarizes the data wrangling efforts while analyzing the WeRateDogs® tweet dataset. This twitter accounts is famous for assigning ratings higher than 10 on a 10 points scale.

Sources

| **Source** | **Description** | |:---|:---| | `twitter-archive-enhanced.csv` | This standard CSV file provided by Udacity contains the the bulk of the bulk of the information. From here we can extract the tweet ids, texts, ratings, and stages. This file was downloaded to a local file (using `requests`) and then later read with `pandas.read_csv`. Before any data was read, the first few lines of the file were shown to check the format.| | `image-predictions.tsv` | This CSV file was provided by Udacity and contains the dog breed AI prediction for the images embedded in each tweet. This file was read directly into a dataframe using `pandas.read_csv` and `requests`, but alternative ways of ingestion were also shown. The file used tab characters `"\t"` as column separators. Before any data was fetched, the first few lines of the file were shown to check the format.| | Twitter API v1 | The file `twitter-archive-enhanced.csv` file was missing the `favorite_count` and `retweet_count` columns so this information needed to be fetched using Twitter's API with the `tweepy`. After the information was read, it was stored into a JSON lines formatted filed named `tweet_json.txt`. This file was kept throughout the project as fetching this information was quite slow. |

Summary of Problems

I think the main source of problems is the nature of the dataset. Although funny and wholesome, the WeRateDogs® rating system is anything but scientific or strict, which means that any analysis or insights we take from it will be just academic.

The data cleanlines issues ranged from the most common and simple like missing values for certain columns, unavailable tweets, which could be fixed quite easily to complex ones which required human analysis, like confusing certain strings for ratings (9/11, 7/11, etc.) or joke ratings.

The data tidiness provided more of a challenge. We had multiple columns representing a single variable more than once. Each of these issue had to be solved in a slightly different manner. There were also simpler tidyness problems like column type interpretations and table consolidations, but those were easily solved using `pandas`.

Summary of Solutions

Once the first round of assessment was done, it was clear which data could be interesting to look at; so all the cleaning steps were geared towards maximizing the quantity and quality of data useful for the path chosen.

Since I chose to focus mostly on the breed predictions and rating, any rows without that information was safe to drop. Cleaning up the breeds was easy, but the ratings required some work. Most of the problems were related to pattern recognition as it seems a simple "\d+/\d+" regexp was use to extract the numerator and denominator. This pattern matched things like "7/11" and "27/4", which were not ratings, and could match ratings with a decimal numerator. I fixed the pattern recognition only accepting "10" as a denominator and matching decimals as numerators. This in turn forced me to convert the `rating` column to `float`.

The solutions for the tidyness problems were much easier as the rules postulated by Hadley Wickham in his "Tidy Data" paper forced our hands once we chose certain variables. The most important cleaning tasks in this regards were:

- Combining the `rating_numerator` and `rating_denominator` into a single `rating` column and chaning its type to float to match the real world schema.
- Combining the `p1`, `p2`, `p3` into a single columng. Even though the three columns are not violating the rule as each one represents a separate variable (1st, 2nd and 3rd predictions), since I was only interested in successful dog breed predictions, I combined them into the `prediction` column chosing the first column that has a dog prediction using the `p1_dog`, `p2_dog` and `p3_dog` columns. I also used these columns to chose the `confidence` column out of `p1_conf`, `p2_conf` and `p3_conf`.
- The dog stages columns `doggo`, `pupper`, `puppo` and `fluffer` were consolidated into the `stage` column as they all represented a single variable.
- Finally, since I considered the tweet a single observational unit, I combined the three dataframes into a single one.

The final steps were just dropping the rows with missing data, and fixing simple data problems.

Outputs

During the wrangling process I generated files that were not inteded as an output, but were used to avoid fetching the same information multiple times from Udacity and other 3rd party services:

| **File** | **Description** | |:---|:---|| `twitter-archive-enhanced.csv` | One of the data sources. || `image-predictions.tsv` | One of the data sources. || `image_predictions_df.pkl` | One of the data sources dataframes saved as a `pickle` file. Since we're supposed to fetch this data straight from the URL without saving it locally first, I used this file to avoid hitting Udacity's CDN over and over again.|| `tweet_json.txt` | A JSON lines formatted file with the tweet data fetched using Twitter's API through `tweepy`. |

As I mentioned in the previous section, all the dataframes were consolidated into a single one called `twitter_archive_master_df`. As part of the project, I was supposed to store this information, which I did in the following formats:

| **File** | **Description** | |:---|:---| | `twitter_archive_master.csv` | CSV file | | `twitter_archive_master.feather` | Feather file. This is a portable format that allows us to keep the type information. | | `twitter_archive_master.sqlite` | SQLite database. Contains a table called `twitter_archive_master` with the contents of the dataframe. |

Conclusions

Even though this was a "toy" dataset not intended to be treated seriously, it required quite a lot of work to turn it into something appropriate for my data analysis purposes. Luckily `pandas` proved once again to be an incredibly powerful tool to simplify both the tedious and challenging tasks of the data wrangling process.