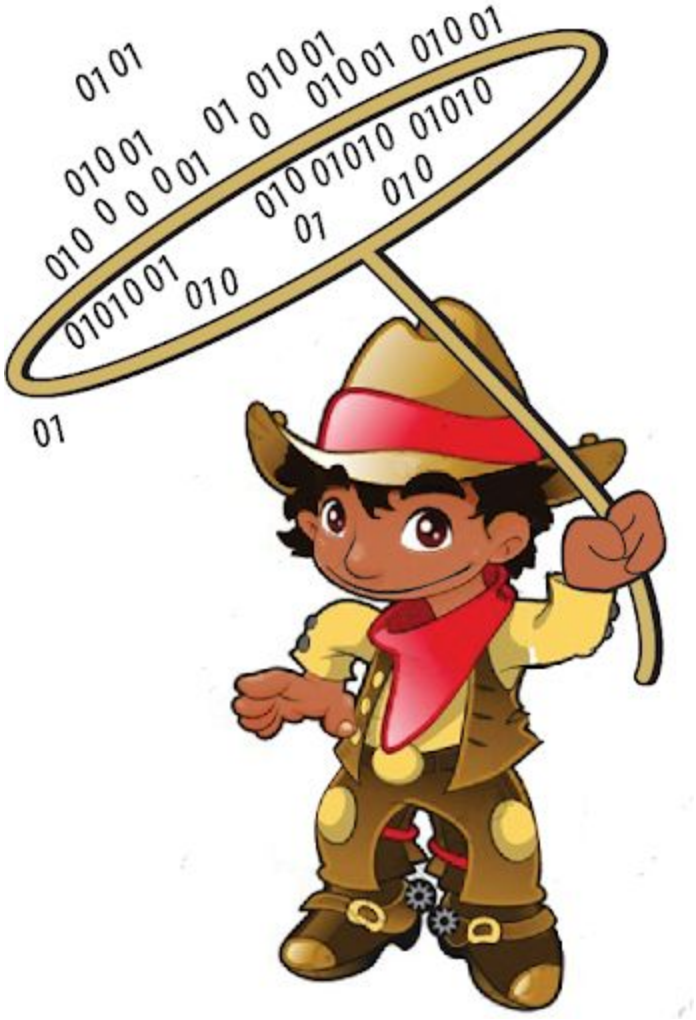
Data Wrangling Report



Dain Russell

Udacity's Data Analyst Nanodegree

INTRODUCTION

Real-world data rarely comes clean. In this project, I gathered, assessed and cleaned data then acted on it through analysis, and visualization. Using Python and its libraries, I gathered data from 3 different sources and in different formats. The dataset that was wrangled, analyzed and visualized is the tweet archive of Twitter user @dog_rates, also known as WeRateDogs. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog.

DATA WRANGLING

- Gathering data
- Assessing data
- Cleaning data

Gathering

The first step of the data wrangling process is to gather our data. I gathered three pieces of datasets

- The WeRateDogs **Twitter archive** `twitter_archive_enhanced.csv` was provided by Udacity and downloaded manually.
- The **tweet image predictions** `image_predictions.tsv` was also provided from Udacity and was downloaded from their servers programmatically using the ***Requests*** library and the following URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv

- I used the tweet IDs in the WeRateDogs Twitter archive to query the **Twitter API** for each tweet's **JSON** data using Python's Tweepy library and store each tweet's entire set of JSON data in a file called 'tweet_json.txt' file.

Assessing

1. **Visually** assessing the data simply involved visually assessing the data by printing each dataset different ways so that I could scan through the data and do an instant observation. I used the `.head()` , and `.sample()` functions to visually see attributes of each dataset.
2. **Programmatically** assessing the data involved using `.info()` for viewing datatypes and missing values; `.describe()` for viewing the summary statistics for each numerical column. I also used other functions including `.duplicated()` to check for duplicates.

Cleaning

The first thing I did was to make a copy of each table in order to keep things organized using `.copy()`. I then dropped all unnecessary columns. After dropping the columns I checked to see if all intended columns were dropped using the `.info()`. I did happen to notice , in the dictionary, that **floofer** is actually **floof** so that was renamed. **None** values in the column names, **doggo, floof, pupper, puppo** were replaced with np.nan (NaN). To keep things tidy , the column values of **doggo, floof, pupper and puppo** were combined into a single column called **dog_stage** and checked with the `.value_counts()` function.

Timestamp needed to be changed to a `datetime` datatype. Once that was done, **date** and **time** were extracted and added to new columns. **Timestamp** was then dropped to keep the table tidy. I also converted **tweet_id** from an integer to string. The **source** column was made more readable by replacing the urls in it by the source name and changing its datatype to category. Combining **rating_numerator** and **rating_denominator** columns

into a single **ratings** column then drop rating_numerator and rating_denominator columns.

Conclusion

In this project I was able to successfully gather, assess and clean three different types of data. The project did reinforce the purpose and need to clean data thoroughly. This allowed for insight to be gained and beautiful visualizations. It also reiterates the points that you made to support it throughout the paper. But instead of simply repeating the paper's arguments, summarize the ideas.

There are more areas of the dataset that could have been assessed and cleaned but for the purpose of this project there was no need to.

The define, code, and test steps of the cleaning process were clearly documented. Copies of the original pieces of data were made prior to cleaning. All issues identified in the assess phase were successfully cleaned using Python and pandas. A tidy master dataset with all pieces of gathered data was created.