Wrangling Report

The wrangling objectives of this project included: gathering, assessing and cleaning data. The objectives were carried out on various file formats programmatically. Programming the wrangling steps in Python is useful because the language provides various functionality through methods and libraries which can be used throughout and optimize the wrangling process.

During the data gathering phase I used Python's: Pandas, Requests and Tweepy libraries to load and explore data. For example Pandas' .load_csv method was used to load the first set of data which was a set of enhanced twitter information. This information was also loaded into a Google sheet for visual assessment. The Tweepy API was used to query Twitter information based on the keys from the first provided .csv. In addition to gathering API data and .csv data, the Requests library was used to download a tab delimited dataset containing a set of image data.

Assessing the data was done both programmatically and visually. Google sheets were used to get a rough idea of the shape of the data from the provided .csv. The data appeared to be in fairly good shape but needed some tidying and cleaning. Superfluous and duplicative columns were identified for removal and consolidation. Observations containing incomplete or incorrectly labeled data were noted for correction, completion or deletion. Using Pandas '.info' method was useful to see if columns were correctly formatted, incorrect formats such as the timestamp were noted for reformatting.

Using the notes from the assessment as a guide, made the cleaning process straightforward. Cleaning was done programmatically using Pandas' process, columns such as 'numerator' and 'denominator' were combined to form a single 'score' column, reducing the need for the separate columns. Observations were updated, dropping the unneeded values such as observations relating to re-tweets. Further cleaning was done to merge and update untidy aspects. The various stage columns (doggo, floofer, pupper and puppo) were integrated into a singular 'stage' column using string selection on the text observations.. Similarly regular expressions were used to fill in missing dog names. Cleaning concluded after converting the data's timestamp values into datetime objects.

Merging the data followed the cleaning process. To simplify merging, each datasets' index was set to its 'twitter_id' column. By doing so, it made it simple to use to Pandas' .merge function to join the datasets based on each observation's unique identifier - 'twitter_id'. After completing the merges, observations that failed to meet the criteria were removed, these included observations without image data.