Wrangle Report

The data wrangling process was divided into three main steps: gathering, assessing, and cleaning.

Gathering

The gathering part focused on obtaining and opening three different data sources into pandas data frames. The first part focused on opening a manually downloaded file in CSV format and reading it using pandas.read_csv() function.
For the second part, I downloaded a file programmatically using requests API and opened the downloaded TSV file with the same function as the first, using pandas.read_csv() function just using tabs as the delimiter.
Finally, for the third part, since I didn't get access to my age-old twitter account, and not getting feedback from the twitter team, I didn't use the tweepy API that was suggested for the course, to download the tweet_json file. But then after downloading it manually from the Udacity classroom, I opened the file and imported the necessary data from the file, in a for loop appending the items as a python dictionary before turning it into a pandas data frame.

Assessing

During the assessment the main thing to look out for are messy and dirty data. The major things to fix for respective dataframe were:

- In twitter archive, the numerators and denominators were not always consistent and a little messy. The dogs development/stage had four separate columns, that should be one single column, some names were words or none instead of names. Those had to be changed into NaNs.
- For image prediction, some predictions had false for all three predicitons, those had to be removed. It also had a few duplicates. Finally, I only kept the p column with the highest confidence interval for analysis
- Lastly, the tweet json basically just needed to drop columns that werenät going to be productive for analysis.

Cleaning

This is a list of the most immersive executions for the cleaning part:

- Manually changing the denominator and numerator for several tweets, using 'loc' function and querying the twitter_id to change the values (see figure 1 below.)
- Creating a new column, rating after cleaning up the denominator and numerator (see figure 2 below.)
- Creating a new column for each dog stage or 'dog_dev' as I called it. This was accomplished through creating an if, elif, else function iterating through each of the four columns: puppo, pupper, fluffer, and doggo. Returning the respective value in each column to the newly added one (see picture below for more details.)
- Creating a new column, dog_breed, which inside a function elected the column with the highest confidence interval and returned it into the column, using a similar method to the previous intervention: if, elif, and else to evaluate which value should be added to the new column.
- Other cleaning actions were:
    - Dropping columns.
    - Splitting strings using replace() method.
    - Changing data types of column values e.g. integers into strings

    - Dropping duplicates and other rows such as retweets.
    - And a few other minor interventions.
- Ultimately, the three cleaned data sets were merged into a combined dataframe before any analysis and/or being saved into a CSV file.

Figures:

```
#Summary of actions and executed:
# Dropping eight tweets

# Remove rows 342, 188, 189, 290, 516, 979, 1663, and 2074 Snoop Dogg is not a dog, but I see what you did there,
# ids: 832088576586297345, 855862651834028034, 855860136149123072, 838150277551247360, 810984652412424192,
# 749981277374128128, 682808988178739200, 670842764863651840

twitter_arch_clean = twitter_arch_clean.query('tweet_id != "832088576586297345" and tweet_id != "855862651834028034" an

# Changing values of four tweets
twitter_arch_clean.loc[twitter_arch_clean.tweet_id == '835246439529840640', 'rating_numerator'] = 13 # Change row 313 -
twitter_arch_clean.loc[twitter_arch_clean.tweet_id == '835246439529840640', 'rating_denominator'] = 10
twitter_arch_clean.loc[twitter_arch_clean.tweet_id == '666287406224695296', 'rating_numerator'] = 9 # Change row 2335 -
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '666287406224695296'), 'rating_denominator'] = 10
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '786709082849828864'), 'rating_numerator'] = 9.75 # Change row 6
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '786709082849828864'), 'rating_denominator'] = 10
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '680494726643068929'), 'rating_numerator'] = 11 # Change row 171
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '680494726643068929'), 'rating_denominator'] = 10
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '778027034220126208'), 'rating_numerator'] = 11 # Change row 763
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '778027034220126208'), 'rating_denominator'] = 10
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '881633300179243008'), 'rating_numerator'] = 11 # Change row int
twitter_arch_clean.loc[(twitter_arch_clean.tweet_id == '881633300179243008'), 'rating_denominator'] = 10
```

Figure 1. Manually changing Numerators and Denominators

```
# Creating a new rating column named 'rating' that takes the rating_numerator divided by rating_denominator
twitter_arch_clean['rating'] = twitter_arch_clean.rating_numerator / twitter_arch_clean.rating_denominator
twitter_arch_clean.head()
```

Figure 2. Creating a new Column

```
# 7. Turning columns, doggo, floofer, pupper and puppo into a single columns
# and the aforementioned columns are the values inside this new column.
# First replace all 'None' strings into NaN with str.replace method
twitter_arch_clean.doggo.replace('None', np.nan, inplace=True)
twitter_arch_clean.floofer.replace('None', np.nan, inplace=True)
twitter_arch_clean.pupper.replace('None', np.nan, inplace=True)
twitter_arch_clean.puppo.replace('None', np.nan, inplace=True)

# Then create a function that uses if,elif, and else to figure out which value in each column can be found

def dog_development_split(i):
    if i['doggo'] == 'doggo':
        return 'doggo'
    elif i['pupper'] == 'pupper':
        return 'pupper'
    elif i['puppo'] == 'puppo':
        return 'puppo'
    elif i['floofer'] == 'floofer':
        return 'floofer'
    else:
        pass
# Then use apply() method and lambda to execute the function above
twitter_arch_clean['dog_dev'] = twitter_arch_clean.apply(lambda i: dog_development_split(i), axis=1)
# Drop columns that aren't needed anymore
twitter_arch_clean.drop(columns=['doggo', 'floofer', 'pupper', 'puppo'], inplace=True)
```

Figure 3. Creating a new Column dog_dev from previous four columns describing the stages in an untidy manner

```
# Turn p1, p2 and p3 where the the confidence interval for each row is the highest to the corresponding breed
def dog_breed_find(i):
    if i['p1_conf'] > (i['p2_conf'] and i['p3_conf']):
        return i['p1']
    elif i['p2_conf'] > (i['p1_conf'] and i['p3_conf']):
        return i['p2']
    elif i['p3_conf'] > (i['p1_conf'] and i['p2_conf']):
        return i['p3']
    else:
        return "Error"

image_prediction_clean['dog_breed'] = image_prediction_clean.apply(lambda i: dog_breed_find(i), axis=1)
```