Wrangle Report

Intro:

In this project I gathered, assessed and analyzed sets of data from Udacity and 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. Software used include: Python(pandas, NumPy, requests, tweepy, json), Google Docs.

Files provided ahead include: 'twitter-archive-enhanced-2.csv', image-predictions.tsv, tweet-json.txt. The archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017. 'image-predictions' is a file contains prediction result of whether the object in the picture is a dog or not as well as dog breed. It was powered by a neural network that can classify breeds of dogs and ran through every image in the WeRateDogs Twitter. 'tweet-json' was provided as additional file that contains all information in every tweet.

Prep:

First step is to use Twitter API to scrap off json file and save them into a text file line by line. Then I created a for-loop to read tweet-json.txt line by line and gathered three pieces of information I need: tweet_id, retweet_count, favorite_count, saved them into a dictionary, append it to a list and converted it to a dataframe.

After loading and reading the necessary files, next step is to access data on hand. During this process, i discovered more than fifteen issues need to be fixed, include five tidiness issues and ten quality issues. Besides common issues like wrong datatype, typo, inconsistent formatting etc, there were ones required more effort and manipulation. I will briefly discuss couple of them.

In adf(twitter-archive) dataframe, there were multiple information contained in a single

column ‘text’. First i uncovered few quality issues, both ‘rating_numerator’ and ‘rating_denominator’ were not probably extracted from ‘text’, neither did difference dog stages. This could lead to big analytical problems as numerator, denominator and dog stages were potential metrics. So I re-extracted all the information changed data types and assigned them to appropriate columns. In order to do so, i need to create regex to capture the patterns: Codes as follow:

Define

re-extract rating_numerator and rating_denominatorvalue from `text`

Code

```
# match regex and slice off '/'
adf['rating_numerator'] = adf['text'].str.extract(r'(\d+(\.\d+)?/)', expand = True)[0]
adf['rating_numerator'] = adf['rating_numerator'].str[:-1]

adf['rating_denominator'] = adf['text'].str.extract(r'(/\d+(\.\d+)?)', expand = True)[0]
adf['rating_denominator']= adf['rating_denominator'].str[1:]

# change data type
adf[['rating_numerator','rating_denominator']] = adf[['rating_numerator','rating_denominator']].astype(float)
```

Define

re-extract different dog stages

code

```
adf['doggo'] = adf['text'].str.extract('([Dd]oggo)')
adf['floofer'] = adf['text'].str.extract('([Ff]loofer)')
adf['pupper'] = adf['text'].str.extract('([Pp]upper)')
adf['puppo'] = adf['text'].str.extract('([Pp]uppo)')
```

During the process, I decided to create a new metric ‘quotient’ and also extract image URL from ‘text’ so the column is more tidy:

Define

add `quotient` column, calculate as: rating_numerator/rating_denominator

Code

```
# insert new column 'quotient' after 'rating_denominator'
data = adf['rating_numerator']/adf['rating_denominator']
adf.insert(9, 'quotient', data)
```

Define

add new column to store url, then slice off url from text and drop column 'expanded_urls' becasue it provides same imformation as 'image'

Code

```
# add new column 'image'
adf['image'] =adf['text'].str.extract('([https]+://[\w(\.|/)]+)')
```

```
# use string.split(pattern, 1)[0] to slice off url
adf['text'] = adf['text'].apply(lambda x: x.split('https://',1)[0].strip())
```

```
# drop column 'expanded_urls'
adf.drop(['expanded_urls'], axis = 1, inplace = True)
```

Another big issue is to 'pivot' four different dog stage columns into a single column. To do so, i simply filled 'NaN' with empty space then added four columns' string together due to the fact a dog should only be in one kind of stage if was correctly recorded. Then i placed 'NaN' back to omit any statistical problem in the future:

Define

combine four columns indicate different dog stage to one single 'dog_stage' column

code

```
# concat four columns together, fill NaN with ''
adf['dog_stage'] = adf['doggo'].fillna('')+adf['floofer'].fillna('')+adf['puppo'].fillna('')+adf['pupper'].fillna('')
# place NaN back to replcae ''
adf['dog_stage'] = adf['dog_stage'].replace('',np.nan)
# drop four columns
adf.drop(adf.iloc[:,11:15], axis =1, inplace = True)
```

There were a lot of more issues in all three data frames. With all that set and fixed, I saved the clean dataframe as a csv file and ready to move on to analysis phase.