Name: Alaa Zarban

Introduction

The analysis done in this project is from three sources that we combined together in one dataframe. The dataset is from WeRateDogs twitter account that rates dogs on unique and strange scale of x over 10. Where x the dog rate which usualy is more than 10!.

Gathering

We started our analysis with gathering the data. The first source was from WeRateDogs archive provided. Second, the data given from image predictions. Third, tweets data that has been taken from text file provided. Unfortunately, I didn't use Twitter API due to a delay of receiving the required authentication IDs.

1- WeRateDogs archive data

We have 2356 entries in WeRateDogs Archive. 17 columns are provided as part of the dataset.

```
WeRateDogs_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

2- Image predictions data using request library through url provided

```
url=" https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv"
response = requests.get(url)

with open('image_predictions.tsv', 'wb') as file:
    file.write(response.content)

image_predictions = pd.read_csv('image_predictions.tsv', sep='\t')

image_predictions.head()
```

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_d
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074	True	collie	0.156665	Tr
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826	True	miniature_pinscher	0.074192	Tr
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.596461	True	malinois	0.138584	Tr
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridgeback	0.408143	True	redbone	0.360687	Tr
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pinscher	0.560311	True	Rottweiler	0.243682	Tr

3- Twitter data from text file provided. Note that JSON python library has been used to ease the conversion from JSON string to dataframe. JSON normalize function was a magical function that made my gathering at its best.

```
# Reading tweet-json.txt file, line by line and append it to "status" list
with open('tweet-json.txt') as file:
    status = []
    for line in file:
        status.append(json.loads(line))

#using json_normalize to convert json keys to columns
from pandas.io.json import json_normalize
df_json = pd.DataFrame.from_dict(json_normalize(status), orient='columns')

# Refining the the json dataframe to the columns we need for analysis
columns_needed = ['id' , 'retweet_count' , 'favorite_count']
df_json_cleaned = df_json[columns_needed]



df_json_cleaned.rename(columns = {'id':'tweet_id'} , inplace=True)

df_json_cleaned.head()
```

Assessing

I did the following in my assessment phase of data wrangling:

- Check tweet IDs for uniqueness.
- Check sources columns value counts.
- Visually, check the tweet text, rating system, dog stages, dog names.
- Visually and Programmatically, check how people call dog names in their tweets.

- Programmatically, check inapplicable and unreal names (less than 2 characters, lower-case)
- Show tweets with decimal ratings and cross check that with the actual rating.
- Check denominator rating scale consistency.
- Check tweets with no images.

As a result, the following has been addressed:

Quality issues:

- Tweets with wrong rating numerator because of decimal number in text.
- Dog names with less than 2 letter.
- Dog names with lower-case (Not real or wrong).
- Dog names with inappliacable names (e.g. "The" , "a").
- Missing dog stage values (All None).
- inconsistent denominator rating (150 , 120 , 130).
- inadequate datatypes (timestamp)
- Tweets with no expanded URLs.

Tidiness issues:

- Dogs stages in different columns, better to be unified in one columns called Dog_Stage.
- Merge all 3 dataframes into one.

Cleaning

We started by making a copy of all dataframes to avoid missing with the original data in case we needed them later. First, we started with the first tidiness issue of dog stages and merged all of the four stages under one column called **Dog_stage.** Second, we merged all the three dataframes into one to ease analysis, insights and visualizations later.

As a result of our assessments, we cleared all quality issues mentioned before and did proper testing to make sure all issues has been addressed.