Project: Wrangling and Analyze Data Report

This project is focus on data wrangling;

Data wrangling can be defined as the process of gathering,cleaning, organizing, and transforming raw data into the desired format for analysts to use for prompt decision-making.

First; imported the python libraries and packages that are going to be used for this analysis

```
1]: import tweepy
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
    import requests
    import os
    from tweepy import OAuthHandler
    import json
    from timeit import default_timer as timer
```

Gathering the Data

The data for this project came in three different formats:

1. Twitter Archive File from WeRateDogs: WeRateDogs downloaded their Twitter archive and sent it to Udacity via email exclusively for use in this project. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017.

1. Directly download the WeRateDogs Twitter archive data (twitter_archive_enhanced.csv)

```
n [2]: Twitter_archive_data = pd.read_csv('twitter-archive-enhanced.csv')# importing  WeRateDogs Twitter archive data
```

Udacity programmatically downloaded this file and provided it as a CSV file to be downloaded manually.

2. Image Prediction File: The images in the WeRateDogs Twitter archive (that is the first dataset above) were run through a neural network that can classify breeds of dogs. The image predictions were stored in this file. This file was hosted on Udacity's server in a TSV format and was downloaded programmatical using the URL the python Requests library.

2. Use the Requests library to download the tweet image prediction (image_predictions.tsv)

```
folder = 'image_predictions'
# Make directory if it doesn't already exist
if not os.path.exists(folder):
    os.makedirs(folder)

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



# Save HTML to file
with open(os.path.join(folder,url.split('/')[-1]), mode='wb') as file:
    file.write(response.content)
```

```
os.listdir(folder)
```

```
image=pd.read_csv('image_predictions/image-predictions.tsv',sep='\t')# reading the image_prediction tsv
image.sample(5)
```


Requests is a versatile HTTP library in python with various applications. One of its applications is to download or open a file from the web using the URL.

3. JSON File from Tweeter API: Using the tweet IDs in the WeRateDogs Twitter archive, I queried the Twitter API for each tweet's JSON data using Python's Tweepy library and stored each tweet's entire set of JSON data in a file called tweet_json.txt file. Each tweet's JSON data was written to its own line. Then I read the .txt file line by line into a pandas DataFrame.

3. Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

```
Query Twitter API for each tweet in the Twitter archive and save JSON in a text file
These are hidden to comply with Twitter's API terms and conditions

%run ./keys.ipynb
```

```
auth = OAuthHandler(API_Key, API_Secret)
auth.set_access_token(Access_Token, Access_Token_Secret)
```

```
api = tweepy.API(auth, wait_on_rate_limit=True)
```

```
tweet_ids = Twitter_archive_data.tweet_id.values
len(tweet_ids)
```

```
# Query Twitter's API for JSON data for each tweet ID in the Twitter archive
count = 0
fails_dict = {}
start = timer()
# Save each tweet's returned JSON as a new line in a .txt file
with open('tweet_json.txt', 'w') as outfile:
    # This loop will likely take 20-30 minutes to run because of Twitter's rate limit
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Success")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except  tweepy.errors.TweepyException as e:
            print("Fail")
            fails_dict[tweet_id] = e
            pass
end = timer()
print(end - start)
print(fails_dict)
```



```
df_list =[]

with open('tweet_json.txt','r') as file:
    for lines in file:
        tweets = json.loads(lines)
        tweet_id=tweets['id']
        retweet_count =tweets['retweet_count']
        favorite_count =tweets['favorite_count']
        df_list.append({'tweet_id':str(tweet_id),
                        'retweet_count':int(retweet_count),
                        'favorite_count': int(favorite_count)})

df_tweetdata = pd.DataFrame(df_list, columns=['tweet_id','retweet_count','favorite_count'])
```

```
df_tweetdata
```

Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python. You can find more details on setting up an app in Twitter and accessing Twitter API using Python.

Assessing the Data

The three data have been gathered and were properly assessed. With the assessment, I looked for quality and tidiness issues.

Quality: Low-quality data is commonly referred to as dirty data. Dirty data has issues with its content. The Data Quality Dimensions are Completeness, Validity, Accuracy, and Consistency.Quality issues worked on are:

Quality issues

1. Twitter_archive_data: Dog name are not consistence , some rows has dog name as ('a,an,the,my,very,such,his....')
2. Twitter_archive_data: tweet_id is integer,timestamp is object(wrong datatype)
3. Twitter_archive_data : removing the retweets and replies.
4. Twitter_archive_data:Extract the main_source of tweets from source column
5. master : Removing outliers (rating_numerator ,rating_denominator)
6. image : p1,p2,p3 has a mixture of uppercase and lowercase
7. image:p1,p2,p3 has a mixture of hypen and Underscore
8. image: Renaming columns

Tidiness: Untidy data is commonly referred to as “messy” data. Messy data has issues with its structure. Tidy issues worked on are stated below

Tidiness issues

1. image & df_tweetdata:These data are part of the same observation unit as the Twitter_archive_data ,(one table with information about dog rating)
2. There are multiple dog stages columns present e.g. doggo, pupper, etc. They should be merged into one column

Cleaning the Data

I cleaned all of the issues I documented that I documented during the assessing stage. It’s good to know that cleaning the data doesn’t mean changing the form of the data. It simply means improving the quality of the data so that one can work with it. The data is cleaned to improve its quality and tidiness.

I cleaned the data using the three cleaning processes which are Define, Code, and Test.

- Define: the issues that were gotten in the assessment are converted into cleaning tasks.
- Code: the cleaning task is converted into code and then run.
- Test: I used codes to test my cleaning efforts to make sure they worked.

Storing the Data

After cleaning the data, I combined the three cleaned datasets using a common attribute, which is the Twitter id, and then saved the master dataset in a CSV file named `twitter_archive_master.csv`.

Analysis and Visualisation

After the data was cleaned it was loaded into the master dataframe , i did some basic analysis like :

Insights:

1. Most used Twitter source
2. Dog Rating distribution
3. Retweeting and Favoriting trend over time
4. Top 6 common dog names
5. Most Popular Dog_stage
6. Dog stage with the higest ima