WRANGLE REPORT

Data wrangling is the process of gathering data, assessing its quality and tidiness, and cleaning it before analysis and visualization. I'll cover the data wrangling process of gathering, assessing and cleaning data of the tweet archive of 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.

I have used various python libraries in this project, which are;

- pandas
- NumPy
- Matplotlib
- requests
- tweepy
- json

[1]:
```
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import re
import requests
import tweepy as tw
import json
%matplotlib inline
```

Gathering the Data:

The data for this project was in *three* different formats:

1. Twitter Archive File — WeRateDogs: WeRateDogs downloaded their Twitter archive and shared it 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. Of the 5000+ tweets, 2356 were filtered to be used as they were the only tweets with ratings.

This file was extracted programmatically by Udacity and provided as a csv file to use. I used read_csv function. The read_csv () function in pandas helps to read comma-separated values (CSV) into a DataFrame

[2]:
```
# contents of the archive file
archive = pd.read_csv('twitter-archive-enhanced.csv')
archive.head(2)
```

2. Image Prediction File: This is a tweet image predictions file of the breed of dogs that are present in each tweet according to a neural network. The results of the prediction, which is a table full of image predictions (the top three only) alongside each tweet ID, image URL, and the image

number that corresponded to the most confident prediction (numbered 1 to 4 since tweets can have up to four images) is stored in this file. It was hosted on Udacity's servers in tsv (tab-separated values) format and had to be downloaded programmatically using the *Url* provided.

Python's *request* library was used to gather this data from the web. ***Requests*** is a versatile HTTP library in python with various applications. One of its applications is to download or open a file from web using the file URL.

```
[3]: # URL of the tsv file downloaded and put in the same directory folder in use

     url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
     resp = requests.get(url, stream = True)
     with open('image_predictions.tsv','wb') as f:  # Saving received content as a tsv file in binary format
        |
         # write the contents of the response (resp.content)
         # to a new file in binary mode.
         f.write(resp.content)

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

3. Twitter API — JSON File: retweet count and favorite count are two of the notable column omissions from the twitter archive file. Fortunately, this additional data can be gathered using Twitter's API. To do this I used the tweet IDs in the WeRateDogs Twitter archive, and queried the Twitter API for each tweet's JSON data using Python's tweepy library

Tweepy is an open source Python package that gives you a very convenient way to access the Twitter API with Python. To access the Twitter API, I needed 4 things from my Twitter App page which were defined as shown;

- consumer key
- consumer seceret key
- access token key
- access token secret key

You can find more details on *setting up an app in twitter* and *accessing Twitter API using Python*.

```
[5]: consumer_key= '' # API Key
     consumer_secret= '' #API Secret
     access_token= ''# Access Token
     access_token_secret= '' #Access Token Secret

     auth = tw.OAuthHandler(consumer_key, consumer_secret)
     auth.set_access_token(access_token, access_token_secret)
     api = tw.API(auth,
                  parser = tw.parsers.JSONParser(),
                  wait_on_rate_limit=True)
```

```
6]: # Fetch tweets from the twitter API using the following loop:
list_of_tweets = []
# Tweets that can't be found are saved in the list below:
cant_find_tweets_for_those_ids = []

for tweet_id in archive['tweet_id']:
    try:
        list_of_tweets.append(api.get_status(tweet_id))
    except Exception as e:
        cant_find_tweets_for_those_ids.append(tweet_id)
```

I isolated the json data of each tweepy status and added them to a list by querying each tweet ID. I used this list to write the Jason data to a text file called tweet_json.txt. I then saved the file as a csv for easy reading to a DataFrame.

```
In [8]: #Then in this code block we isolate the json part of each tweepy
#status object that we have downloaded and we add them all into a list

my_list_of_dicts = []
for each_json_tweet in list_of_tweets:
    my_list_of_dicts.append(each_json_tweet)
```

```
In [9]: # Write tweet data to json file
with open('tweet_json.txt', 'w') as file:
        file.write(json.dumps(my_list_of_dicts, indent=4))
```

```
n [10]: # saving json file as csv
api_tweets = pd.read_json("tweet_json.txt")
api_tweets.to_csv("tweet.csv", index=False)
```

```
n [11]: # load the csv file to confirm
api_data = pd.read_csv("tweet.csv")
api_data.head(2)
```

Assessing the Data

After gathering the data, the three tables were saved and assessed visually and programmatically. With both the assessments I looked for unclean data in all the three DataFrames, i.e. for Tidiness and quality issues.

Quality: Low quality data is commonly referred to as dirty data. Dirty data has issues with its content. The data Quality Dimensions are;

1. Completeness - The degree to which the data contains all desired components or measures
2. Validity - The degree to which the data conforms to a defined schema
3. Accuracy - The degree to which the data conforms to the actual entity being measured or described
4. Consistency - The degree to which the data is repeatable from different points of entry or collection

Tidiness: Untidy data is commonly referred to as “messy” data. Messy data has issues with its structure. Tidy data is where:
1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Both assessments are well documented in the python notebook.

Cleaning the Data

Cleaning means acting on the made assessments to **improve quality and tidiness**.

Improving quality doesn’t mean changing the data to make it say something different — that’s data fraud. Quality improvement means;

- Correcting when inaccurate
- Removing when irrelevant
- Replacing when missing.

Similarly, improving tidiness means transforming the dataset so that;

- each variable is a column,
- each observation is a row,
- each type of observational unit is a table.

The programmatic Data Cleaning process i.e. ***Define, Code and Test*** was followed.

- Coding - Translating these defined cleaning tasks to code and executing that code
- Defining - Creating a data cleaning plan in writing, where we turn our assessments into defined cleaning tasks
- Testing - Assessing our dataset, often using code, to make sure our cleaning operations worked

The cleaned dataset was stored in a csv file called twitter_archive_master.csv ready for analysis.