Project 2: Wrangle and Analyze Data

Data Wrangling Report

Prepared By: Suzan Hamza

Introduction

This report briefly describes the efforts used to wrangle *WeRateDogs* Twitter data to create interesting and trustworthy analyses and visualizations.

The following packages (libraries) were installed and imported at the beginning of the wrangle_act.ipynb notebook.

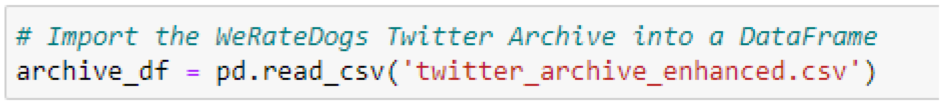
- *pandas*
- *NumPy*
- *requests*
- *tweepy*
- *json*
- *matplotlib*
- *seaborn*

Gathering Data for this Project

Each of the **three pieces of data** as described below were gathered in a Jupyter Notebook titled wrangle_act.ipynb:

1) The WeRateDogs Twitter archive.

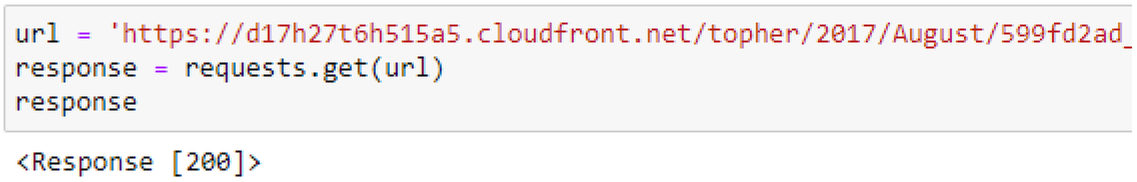
This file named twitter_archive_enhanced.csv was downloaded manually and read into a dataframe using pandas `read_csv()`.



```
# Import the WeRateDogs Twitter Archive into a DataFrame
archive_df = pd.read_csv('twitter_archive_enhanced.csv')
```

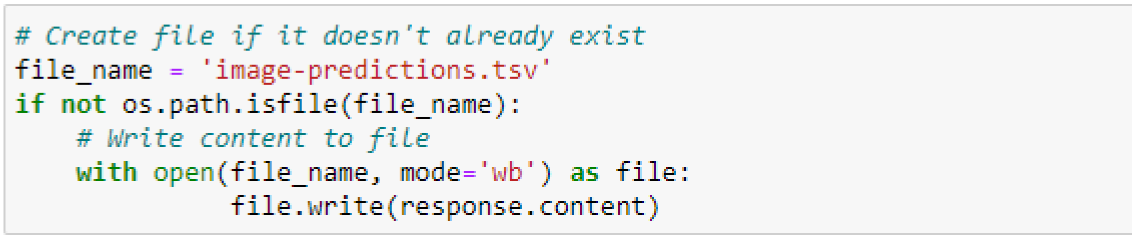
2) Tweet Image Predictions File

This file (image_predictions.tsv) is hosted on Udacity's servers and was *downloaded programmatically* using the **Requests** library and the following URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv



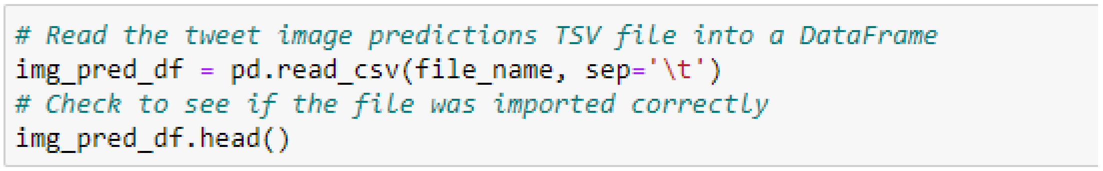
```
url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_
response = requests.get(url)
response
```

```
<Response [200]>
```



```
# Create file if it doesn't already exist
file_name = 'image-predictions.tsv'
if not os.path.isfile(file_name):
    # Write content to file
    with open(file_name, mode='wb') as file:
            file.write(response.content)
```

Then the file was read into a dataframe using pandas `read_csv()`.



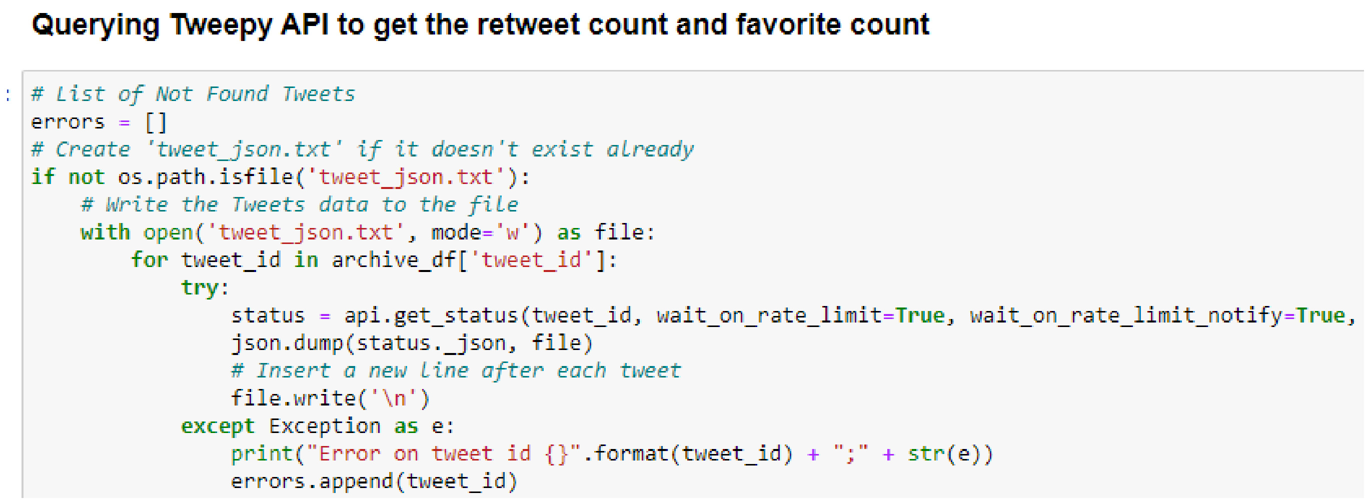
```
# Read the tweet image predictions TSV file into a DataFrame
img_pred_df = pd.read_csv(file_name, sep='\t')
# Check to see if the file was imported correctly
img_pred_df.head()
```

3) Additional Data via the Twitter API

First, a Twitter developer account was created, then it was used to generate the *Consumer* API keys, and the Access Token and Access Token Secret needed.

Those secret credentials were stored in a separate .env file, and were loaded using Python's `dotenv` library.

Each tweet's **retweet count** and **favorite** *("like")* **count** was gathered using the **tweet IDs** in the WeRateDogs Twitter archive, by querying the Twitter API for each tweet's JSON data using Python's **Tweepy** library.

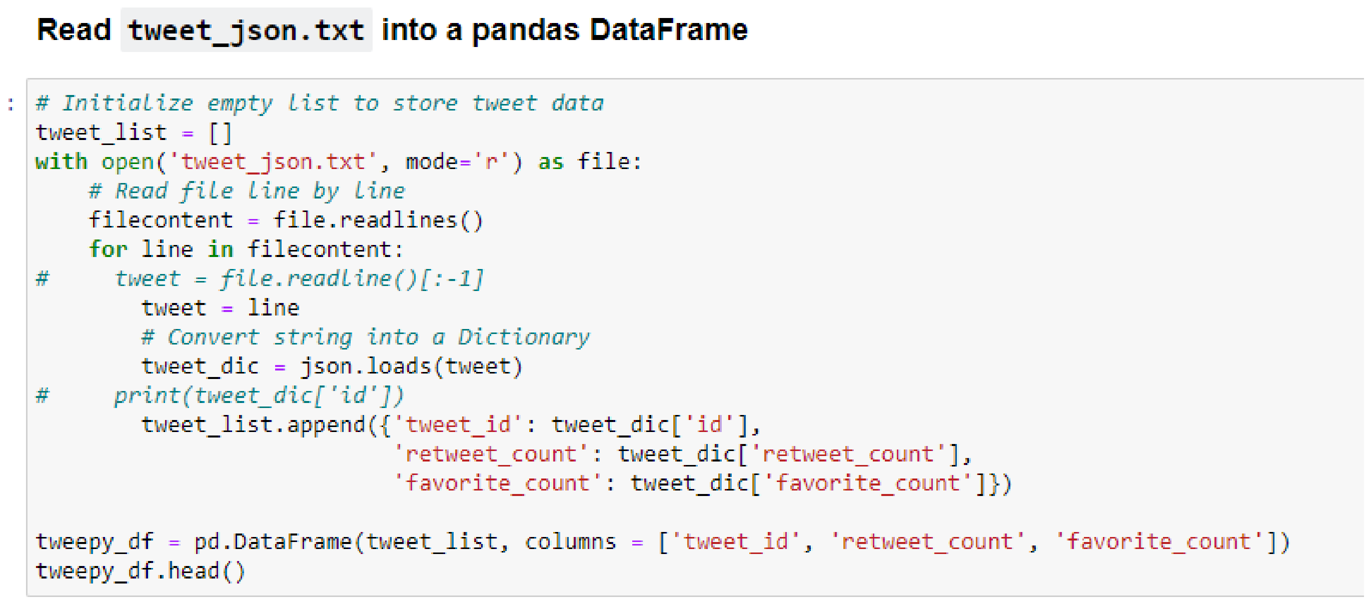


Querying Tweepy API to get the retweet count and favorite count

```
# List of Not Found Tweets
errors = []
# Create 'tweet_json.txt' if it doesn't exist already
if not os.path.isfile('tweet_json.txt'):
    # Write the Tweets data to the file
    with open('tweet_json.txt', mode='w') as file:
        for tweet_id in archive_df['tweet_id']:
            try:
                status = api.get_status(tweet_id, wait_on_rate_limit=True, wait_on_rate_limit_notify=True,
                json.dump(status._json, file)
                # Insert a new line after each tweet
                file.write('\n')
            except Exception as e:
                print("Error on tweet id {}".format(tweet_id) + ";" + str(e))
                errors.append(tweet_id)
```

Then each tweet's entire set of JSON data was stored in a file called tweet_json.txt file. Each tweet's JSON data was written to its own line.

Then this .txt file was read line by line into a pandas *DataFrame* with `tweet ID`, `retweet count`, and `favorite count`.



Read `tweet_json.txt` into a pandas DataFrame

```
# Initialize empty list to store tweet data
tweet_list = []
with open('tweet_json.txt', mode='r') as file:
    # Read file line by line
    filecontent = file.readlines()
    for line in filecontent:
#     tweet = file.readline()[:-1]
        tweet = line
        # Convert string into a Dictionary
        tweet_dic = json.loads(tweet)
#     print(tweet_dic['id'])
        tweet_list.append({'tweet_id': tweet_dic['id'],
                           'retweet_count': tweet_dic['retweet_count'],
                           'favorite_count': tweet_dic['favorite_count']})

tweepy_df = pd.DataFrame(tweet_list, columns = ['tweet_id', 'retweet_count', 'favorite_count'])
tweepy_df.head()
```

Assessing Data for this Project

After gathering each of the above pieces of data and loading them into a separate pandas dataframe, first they were assessed *visually* by displaying each dataframe in the notebook and investigating its contents for **quality** and **tidiness** issues.

Then a *programmatic assessment* was conducted using pandas methods, like `info`, `head`, `describe` and `value_counts`.

The following **quality** and **tidiness** issues were detected, including several issues that did not satisfy the Project Motivation:

Quality Issues

archive_df table

1. **78** entries are **replies** and not original tweets (`in_reply_to_status_id` and `in_reply_to_user_id` has values).
2. **181** entries are **retweets** and not original tweets (`retweeted_status_id`, `retweeted_status_user_id` and `retweeted_status_timestamp` has values).
3. `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id`, `retweeted_status_user_id` and `retweeted_status_timestamp` columns are not useful for analysis and should be removed.
4. **59** entries do not contain images (`expanded_urls` is null).
5. `tweet_id` is integer instead of string (object).
6. `timestamp` is a string and not datetime.
7. **55** dog `names` incorrectly extracted as `a`, because the tweet was in the format *'This is a ...'* instead of *'This is (Dog Name) ...'*.
8. **745** dog `names` incorrectly extracted as `None`, because the tweet was not in the format *'This is (Dog Name) ...'*.
9. `tweet_id = 810984652412424192` does not include a dog rating, the text= 'Meet Sam. She smiles 24/7 & secretly aspires to be a reindeer.' and the rating was incorrectly extracted as 24/7.
10. `tweet_id = 666287406224695296` rating was incorrectly extracted as 1/2 while it should be 9/10. text = 'This is an Albanian 3 1/2 legged Episcopalian. Loves well-polished hardwood flooring. 9/10'.

img_pred_df table

1. `tweet_id` is integer instead of string (object).

2. Only **2075** image predictions are available, which indicates that another **281** tweets have no images and should be excluded from the `archive_df` table.

tweepy_df table

1. `tweet_id` is integer instead of string (object).

Tidiness Issues

archive_df table

1. Dog Stages variables split into four columns (`doggo`, `floofer`, `pupper` and `puppo`) instead of one.
2. A single observational unit (Tweet information) is stored in multiple tables (`archive_df` and `tweepy_df`).

Cleaning Data for this Project

Each of the issues documented in the assessment phase were cleaned as follows:

1) Missing Data

Missing data or data that does not satisfy the Project Motivation was cleaned first.

- Removed all **replies** from archive table; entries where `in_reply_to_status_id` and `in_reply_to_user_id` are not null.
- Removed all **Retweets** from archive table; entries where `retweeted_status_id`, `retweeted_status_user_id` and `retweeted_status_timestamp` are not null.
- Dropped `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id`, `retweeted_status_user_id` and `retweeted_status_timestamp` columns from archive table, because they will not be useful to our analysis.
- Dropped entries where `expanded_urls` is null from archive table, because that means that these tweets do not contain any images.
- Removed entries from archive table with `tweet_id` that did not exist in img_pred table, because that means that these tweets do not contain any images.

2) Tidiness Issues

- Combined the four Dog Stages columns (`doggo`, `floofer`, `pupper` and `puppo`) into one column named `dog_stages`.
- Merged the `retweet_count` and `favorite_count` columns with the `archive` dataframe.

3) Remaining Quality Issues

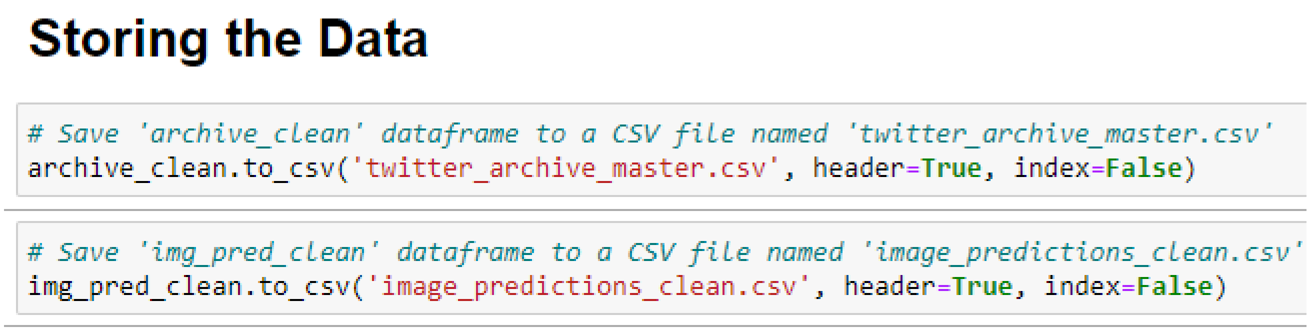
- Converted `tweet_id` data type from integer to string in both `archive` and `img_pred` tables.
- Converted `timestamp` from string data type to datetime in `archive` table.
- Corrected some of the dog names that were incorrectly extracted as `a` or `an` or `None` programmatically.
- Replaced the remaining `a`, `an` and `None` name values with `NaN`.
- Replaced the `None` values in the `dog_stages` column with `NaN`.
- Inspected the tweet's text for `tweet_id = 666287406224695296` and corrected the rating from 1/2 to 9/10 manually, since it is a one off occurrence.

Note: The rating for `tweet_id = 810984652412424192` was not found and could not be corrected, but that issue will not affect our analysis.

The result was stored in two high quality and tidy master pandas DataFrames, `archive_clean` and `img_pred_clean`.

Storing Data for this Project

Finally, the two clean DataFrame(s) were stored in a **CSV file** with the main one named twitter_archive_master.csv and the other named image_predictions_clean.csv for image predictions data.



Storing the Data

```
# Save 'archive_clean' dataframe to a CSV file named 'twitter_archive_master.csv'
archive_clean.to_csv('twitter_archive_master.csv', header=True, index=False)
```

```
# Save 'img_pred_clean' dataframe to a CSV file named 'image_predictions_clean.csv'
img_pred_clean.to_csv('image_predictions_clean.csv', header=True, index=False)
```