Wrangling Report

Introduction

Real-world data rarely comes clean. Thus, influences analysis which leads to poor analysis and wrong decision. With the aid of Python and its libraries, this challenge can be solved by a part of data analytics called ***data wrangling***.

Project overview

The dataset wrangled was 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. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because ***"they're good dogs Brent.?"***

WeRateDogs™ @dog_rates · Apr 16
Say hello to Alice. I'm told she enjoys car rides and smells good. 12/10 would give her everything she could ever want

WeRateDogs™ @dog_rates · Apr 16
A photographer took pictures before and after he told his bunny he's a good boy. Here are the results. 13/10

Software used for the project where;

- Jupyter Notebook
- Google chrome
- Microsoft Excel and Word

Metrology

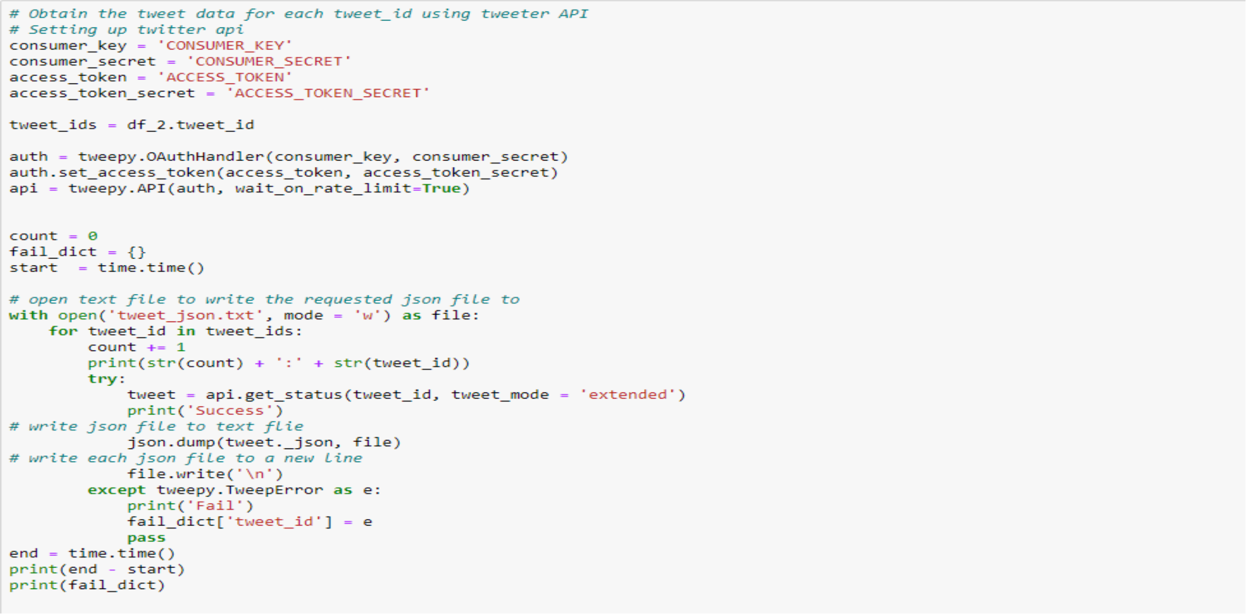
My wrangling process entailed;

- ### Data Gathering

Downloaded manually, a data sent to Udacity by WeRateDogs and imported it into my work space using pandas. This twitter archive contained basic tweet data for all 5000+ of their tweets, but not everything. One column of the archive contains though: each tweet's text, from which the rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) where extracted to make this Twitter archive "enhanced." Of the 5000+ tweets, it was filtered for tweets with ratings only (there are 2356).

Since the twitter archive didn’t contain every data required for the project, more data was gather programmatically via Twitter API. This involved;

- Creating a twitter developers account
- requesting and storing the JSON file of each tweet_id
- extracting the data, I found relevant
- finally, make a Data Frame for this data



```
# Obtain the tweet data for each tweet_id using tweeter API
# Setting up twitter api
consumer_key = 'CONSUMER_KEY'
consumer_secret = 'CONSUMER_SECRET'
access_token = 'ACCESS_TOKEN'
access_token_secret = 'ACCESS_TOKEN_SECRET'

tweet_ids = df_2.tweet_id

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)


count = 0
fail_dict = {}
start  = time.time()

# open text file to write the requested json file to
with open('tweet_json.txt', mode = 'w') as file:
    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')
# write json file to text flie
            json.dump(tweet._json, file)
# write each json file to a new line
            file.write('\n')
        except tweepy.TweepError as e:
            print('Fail')
            fail_dict['tweet_id'] = e
            pass
end = time.time()
print(end - start)
print(fail_dict)
```

Requesting and Storing the JSON File of Each Tweet_Id Using Twitter API

In addition, every image in the WeRateDogs Twitter archive was ran through a neural network that can classify breeds of dogs. The results: 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). This data was gathered programmatically from Udacity's servers using the Requests library and the following URL: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv and imported into my workspace,

- ## Data Accessing

While accessing the data, below were the observations made;

QUALITY

Twitter_a

- Several tweets were retweet
- Some tweets are replies. They are associated with missing expanded _urls and other data in the same row
- Tweet_id is an integral not a string
- timestamp is a string not a datetime
- Doggo, floofer, pupper and puppo column represent nulls as None
- Tweet_id 835246439529840640 rating denominator is 0
- Rating scale is inconsistent (rating denominator has 18 unique values)

Predictions

- Tweet_id is an integral not a string
- Several classifications are invalid

tweet_a

- Column containing tweet id is named as id_str instead of tweet_id
- The following columns: statuses_count, user_favourites_count, list_count, and friends_count; are not necessary since their values is unique to the user (WeRateDogs)

TIDENESS

- Similar information is contained in three tables
- The number of tweets_id for each table is different

- ### Data Cleaning

  In this phase of wrangling, the identified issues were fixed.

Discussion/Conclusion

Data Wrangling is an iterative rather than straight process. As I wrangled the data, I moved between the data assessing and cleaning steps. Although I was able to spot some issues in my data and fix them but not all issues were found fixable like the name of dog not present in the tweet text. Although all issue wasn't fixed but it was ensured those relevant to my analysis were fixed.