Report: act_report

Introduction

A data-driven data wrangler knows the importance of working with the right and complete data.

Just before analysis data, we need to gather data from the right sources and be able to put it in a useful form.

And that absolutely the key point of my Data Wrangling project at Udacity Data Analyst Nanodegree Program.

The project includes the best practice that I have learned which is gathering data from a different type of resources in a different of formats, assessing the data quality and tidiness, cleaning the data, and show some plot through analysis and visualizations part

****First step is Gathering the Data**

I've used three type of sources:.csv /request URL/ json.

So we start with the import of the python libraries and packages then reading the data from the three type of sources

```
import json
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import re
import seaborn as sns
import requests
from scipy import stats
```

then we import the Twitter Archive.csv 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.

```
# Contains tweets with extra info already parsed from text
df_twitter_archive = pd.read_csv('twitter-archive-enhanced.csv')
```

And the 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 programmatically using the URL the python Requests library.

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.

```
# Predictions of dog breeds
url_image_predictions = ('https://d17h27t6h515a5.cloudfront.net/topher/2017/August/'
                         '599fd2ad_image-predictions/image-predictions.tsv')
# Create dataframe from TSV
df_image_predictions = pd.read_csv(url_image_predictions, delimiter='\t')
df_image_predictions.head(5)
```

finally, the JSON File from Twitter 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.

Tweepy is an open-source Python package that gives you a very convenient way to access the Twitter API with Python.

```
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer
import time
```

```
# Twitter API for more data (save to local file)
#consumer_key = 'YOUR CONSUMER KEY'
#consumer_secret = 'YOUR CONSUMER SECRET'
#access_token = 'YOUR ACCESS TOKEN'
#access_secret = 'YOUR ACCESS SECRET'

consumer_key = 'lE0rhbCiSu1qoNZBtyVlAtbvp'
consumer_secret = 'DBKcFrHh8CxuyH4ut0P0H2XO6vkgShPWjtsoRr27P9jFJEtjRJ'
access_token = '1562078666301050882-OtCMdHIXU0o5aKbFNaQf5sj4ovWUA4'
access_secret = 'olaYX4cNksk2WfFXqLsssH7rXItgllIWv3HnE2bpl1uNq'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
```

```
# Query Twitter's API for the JSON data of each tweet ID in the Twitter archive
index = 0
# dictionary to catch the errors
error_dict = {}
start = time.time()

# Save each tweet's returned JSON as a new line in a .txt file
with open ('tweet_json.txt', 'w') as tweet_bk:
    # This will likely take 20 - 30 minutes to run because of Twitter's rate limit
    for tweet_id in tweet_ids:
        index += 1
        try:
            # Get the status data for each of the tweet IDs
            tweet = api.get_status(tweet_id, tweet_mode = 'extended')
            print(str(index) + ": " + "ID - " + str(tweet_id))
            # Convert each tweet status to JSON string and save it in the tweet_bk file
            json.dump(tweet._json, tweet_bk)
            # recognize \n as a break of text
            tweet_bk.write("\n")

        # Catching errors that might occur while accessing the tweet data or content
        except tweepy.TweepError  as error:
            print(str(index) + ": " + "ID - " + str(tweet_id) + " has an error:", error.response.text)
            # Appending the errors to the dictionary; error_dict
            error_dict[tweet_id] = error

end = time.time()
print(end - start)
```

```
# Save only certain tweet elements in dataframe
elements_to_save = ['id', 'favorite_count', 'retweet_count']
# Later convert list to dataframe
data = []

with open('tweet_json.txt', 'r') as readfile:
    # Read in JSON line and convert to dict
    tweet_json = readfile.readline()

    # Read line by line into DataFrame
    while tweet_json:
        tweet_dict = json.loads(tweet_json)
        # Create a smaller dict
        data_row = dict((k, tweet_dict[k]) for k in elements_to_save)
        data.append(data_row)

        # Read in JSON line and convert to dict
        tweet_json = readfile.readline()



df_tweet_json = pd.DataFrame.from_dict(data)
```

```
df_tweet_json
```

*****Second step Assessing the Data**

after gathering data from the three sources of data. we move to the assessment, we 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.

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.

After visually assessing the data in DataFrames and in excel spreadsheets, and programmatically assessing the three DataFrames individually, I found 8 quality issues and 2 tidiness issues in all, and all were documented in my python notebook.

*****Third step Cleaning the Data**

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

* Define: the issues that we found in the assessment part to clean it.

* Code: writing the line of code for cleaning the issue.

* Test: test codes to check results.

*****Fourth Storing the Data**

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

*****Finally, Analysis and Visualizations**

The data has been gathered, assessed, cleaned, and is now ready to be analyzed.

In other to get insights from the data, we asked the data some questions.

1: What is the most popular dog stage according to the neural network's image prediction?

Looking at the distribution of dog images, it shows that 'pupper' (a small doggo, usually younger) is the most popular dog stage, followed by 'doggo' and 'puppo'.

```
df_final_clean['dog_stage'].value_counts()
```

```
None             1831
pupper            224
doggo              75
puppo              24
doggo,pupper       10
floofer             9
doggo,puppo         1
doggo,floofer       1
Name: dog_stage, dtype: int64
```

Definition of each dog stages?

- Pupper: A pupper is a small doggo, usually younger. Can be equally if not more mature than some doggos.
- Doggo: A doggo is a big pupper, usually older. It appears to have its life in order. Probably understands taxes and whatnot.
- Puppo: A puppo is a transitional phase between pupper and doggo. Easily understood as the dog's equivalent of a teenager.

So our analysis is telling us that from all the image predictions, most of them were in. It could be due to the fact that a young dog is usually cute more than a mature dog and this is the reason that most people prefer it.

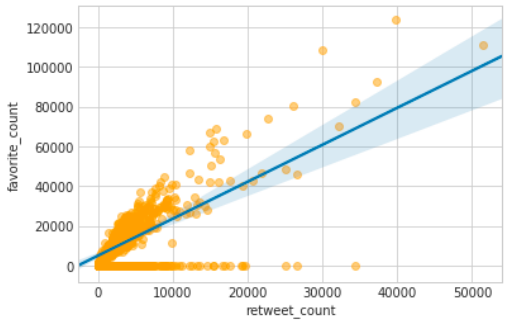
Pupper dog

But we must notice that there's a lot of missing data in the dog stage column of the master dataset, thus the distribution may not reflect the reality.

2.Does retweet count positively correlate with the favorite count?

```
sns.set_style('whitegrid')
sns.regplot(df_final_clean.retweet_count, df_final_clean.favorite_count, scatter_kws = {'color': 'orange', 'alpha': 0.5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f791a711860>
```



We see that there is a linear relationship between the two variables. This doesn't mean that increase in retweet_count causes an increase in favorite_count but when you compare both linearly, there is a strong positive linear relation between retweet_count and favorite_Count

Conclusion

This project was an interesting and challenging. It improves my skills as a good data wrangler.

Thanks for reading!