Report on Data Wrangling & Analysis of @WeRateDogs Twitter Data

Introduction

WeRateDogs is a rapidly growing Twitter account, dedicated to post, rate Twitter user's dog photos and make funny comments about them. Nowadays it has almost 6 million followers and tons of likes, favorites and comments.

Objective

The objective of this is to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "Wow!"-worthy analyses and visualizations.

Steps I took:

I started by creating a jupyter notebook named *wrangle_act.ipynb.,* where I imported all the necessary libraries needed for the wrangling, analysis and visualizations tasks.

Imported LIbraries:

```
import pandas as pd
import requests
import json
import os
import matplotlib.pyplot as plt
import numpy as np
```

- ## Data Gathering

  The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: eachtweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, In order to fix this, I needed to gather data from 2 others which sums the data sources to three.

  1. Twitter archive enhanced: This data (a csv file) was provided as a download from the Udacity Server. And uploaded into my jupyter notebook using pandas.

```
Tweet_archive_enhanced='~/Documents/Semicolon/Projects/wrangling_analysis_@dogrates_twiiterfeeds/datasets/twitter-archive-enhanced.csv'
try:
    df_twitter_archive = pd.read_csv(tweet_archive_enhanced)
except FileNotFoundError as e:
    print(e)
df_twitter_archive.head(1)
```

  2. Image_predictions: I gathered the image prediction data (a tsv file) from a provided link using python requests library and pandas read_csv function like the above code

```
img_predictions_link='https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
```

res = requests.get(img_predictions_link)

3. Tweets: This data (json format) was meant to be gathered using the twitter API. But my application for a twitter developer account was not approved in due time. So I had to use the Udacity provided json tweets data.

- ## Assessing Data

  After gathering the data from the 3 different sources, I needed to assess the data visually and programmatically for both tidiness and quality issues before it can be fir for analysis.
  These are key parts of the assessment summary plus a code excerpt

  Quality Issues

  Twitter Archives

  - Incorrect denominator and numerator rating values, values greater than 10 and some quite outrageous

  Image Predictions

  - The string values in p1, p2, and p3 columns _breed predictions_ by the neural network are not in uniform format

  Json Tweets

  - language column values are not meaningful

  Tidiness

  Twitter Archives

  - dog stages values in multiple columns

  Image Predictions

- image prediction datasets needs to be joined with twitter archive rather than in separate datasets

  Json Tweets

- multiple id columns in json_tweets da

Code Excerpts:

df_twitter_archive.info()
df_img_pred.isnull().any().sum()
Df_tweets.dtypes

- ## Cleaning Data

  After identifying the quality and tidiness issue from the assessment part, I proceeded to clean the identified tidiness and quality issues using the ***define, code, test format***. But before I started any cleaning effort, I created copies of the datasets. I will be using samples from the cleaning blocks in this report.

  Quality/TIdiness Issue:

  Dog Stages Columns with None Values and multiple columns

  - **Define**

    Firstly combine dog stages column into one column using pandas melt function Inspect text column to check if dog stages name was or was not properly extracted.

    If extraction is feasible, use the extract function and regex to get the dog stage type out and if not leave it as it is

  - **Code**

twitter_archive_copy.stage_check.value_counts()
None 9030
pupper 257
doggo 97
puppo 30
floofer 10
Name: stage_check, dtype: int64

```
twitter_archive_copy=
pd.melt(twitter_archive_copy,id_vars=id_vars_cols,value_vars=value_var_cols
,var_name='dog_stages', value_name='stage_check')

twitter_archive_copy.stage_check=
twitter_archive_copy.text.str.extract(r"(doggo|floofer|pupper|puppo)",
expand=False)
```

  - **Test**

twitter_archive_copy.stage_check.value_counts()
out:
pupper 1060
doggo 372
puppo 148
floofer 16
Name: stage_check, dtype: int64

From the above, we can see that our dog_stages column before the cleaning contained about 9030 None values and were not in a single columns, but after the cleaning, I was able to eliminate the None values with the correct dog stages and in a single column

Conclusion

I performed various tidy and quality cleaning on each of the datasets and when this was done completely, I had to merge the tree datasets into one dataset and convert to a csv file. This was later used as a dataset for analysis and visualizations.

Below are excerpts of the code used to merge the datasets and converted to csv file.

Code:

- ***df_master = pd.merge(left=twitter_archive_copy, right=img_pred_copy, on='tweet_id', how='left')***
- ***df_master = df_master.merge(tweets_copy, on=['tweet_id'], how='left')***
- ***master_file='~/Documents/Semicolon/Projects/wrangling_analysis_@dogrates_twiiterfeeds/datasets/cleaned_data/twitter_archive_master.csv'***
- ***df_master.to_csv(master_file, encoding='utf-8', index=False)***