Report on the Data Wrangling

The data wrangling process has three steps. First is the data gathering, second, is accessing and third is cleaning after which the clean data is stored in a csv file. I will discuss these steps according to what I did to get to my clean Masterdf which I used to draw insights from the data.

Data Gathering:

There are 3 data files needed in this project

1. The CSV file named twitter_archive_enhanced.csv which would also be referred to as the file in hand for this project. I downloaded this file manually by clicking on the link provided within the project, then read it in using “pd.read_csv” method. Within my workspace I referred to the dataframe as csv_file.
2. The TSV file, which is named image_predictions.tsv is hosted on a url provided in the project. I used the Requests library to download this file, then I read it in using pd.read_csv method. The only difference this time is that I used the argument “sep” with value “\t”. It contained predictions of the breeds of dog images in the tweets. Within my workspace, I named this dataframe tsv_file.
3. I got the third data from twitter using the tweepy library with reference to the tweet_ids in the csv_file. After fetching the data, I stored them in a txt file named tweet_json.txt. I then read this file in using json.dumps and created a dataframe with it which I named json_file with columns tweet_id, url, retweet_count, retweeted and favorite_count.
   Prior to this, I applied for a twitter developer account and got the API keys, secrets and tokens. I have also altered these in my submission.

   With all these files gotten, I went on to Assessing the Data.

Assessing Data:

First, I visually assessed the files. Visually Scanning through the twitter-archived-enhanced, I noticed that some rows had values that were not assigned to particular columns.

Though I could not get much information from them due to the large data sizes. Hence I resorted to the programmatic approach.

For each of the dataframes, I used the “.info”, “.describe”, “.nunique”, “.isduplicated” methods to accesses the dataframes programmatically.

I noticed the following

```
csv_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

```
csv_file.nunique()
```

```
tweet_id                    2356
in_reply_to_status_id         77
in_reply_to_user_id           31
timestamp                   2356
source                         4
text                        2356
retweeted_status_id          181
retweeted_status_user_id      25
retweeted_status_timestamp   181
expanded_urls               2218
rating_numerator              40
rating_denominator            18
name                         957
doggo                          2
floofer                        2
pupper                         2
puppo                          2
dtype: int64
```

```
tsv_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   int64  
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

```
tsv_file.nunique()
```

```
tweet_id    2075
jpg_url     2009
img_num        4
p1           378
p1_conf     2006
p1_dog         2
p2           405
p2_conf     2004
p2_dog         2
p3           408
p3_conf     2006
p3_dog         2
dtype: int64
```

```
json_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2057 entries, 0 to 2056
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   tweet_id        2057 non-null   int64 
 1   url             2057 non-null   object
 2   retweet_count   2057 non-null   int64 
 3   retweeted       2057 non-null   object
 4   favorite_count  2057 non-null   int64 
dtypes: int64(3), object(2)
memory usage: 80.5+ KB
```

```
json_file.nunique()
```

```
tweet_id          2057
url               1987
retweet_count     1527
retweeted            2
favorite_count    1827
dtype: int64
```

I noted these data quality and tidiness issues and more.

- Visually Scanning through the twitter-archived-enhanced, I noticed that some rows had values that were not assigned to particular columns. (Tidiness issue)
- Using Excel, the Json file cannot be visually scanned. Viewing the Json file programmatically as a dataframe, there was no quality issues seen.
- Tweet_id column in all the dataframes is wrongly typed to integers instead of objects (Quality issue)
- Somes rows in twitter-archived-enhanced has single characters for the name column and some None (Quality issue)
- Column "in_reply_to_status_id" in twitter-archived-enhanced have many null values. (Tidiness issue)
- Column "in_reply_to_user_id" in twitter-archived-enhanced have many null values. (Tidiness issue)
- Column "retweeted_status_id" in twitter-archived-enhanced have many null values. (Tidiness issue)
- Column "retweeted_status_user_id" in twitter-archived-enhanced have many null values. (Tidiness issue)
- Column "retweeted_status_user_timestamp" in twitter-archived-enhanced have many null values. (Tidiness issue)
    - Keeping these columns may bring in some bias to the analysis
- Column "timestamp" is wrongly typed (Quality issue)
- Create new column called rating from rating_numerator and rating_denominator (Quality issue)
- Create new column dog_type from text (Quality issue)
- Create new column dog_breed from image prediction data (Quality issue)
- Some text values have more than one dog_type in it (Quality issue)
- The data has retweeted tweets which are not useful for this analysis (Quality issue)
- Merge all available datasets to improve quality (Quality issue)

Data Cleaning:

To clean these data, I need to deal with all these issues one after the other following the define, code and test procedure.

First I changed the datatype of the tweet_id and timestap from int to string objects and from string object to datetime.

Following this, I merged three dataframes to form one master dataframe which I also accessed aain to ensure its cleanliness.

Then I extracted the dog type from the text column, creating a new column ‘type’. Following it I created new columns from existing ones and as well dropped columns that would not be useful to my analysis.

Finally, I stored the clean and ready to use data in a csv file called twitter_archive_master.csv