Wrangle report

WeRateDog analysis project.

What is data wrangling?

Data wrangling "is the process of cleaning and organized messy and complex data sets for easy access and analysis", the data wrangling consist of 3 sub-process which is gather, assess and clean. In this report, I will explain each sub-process and how I used it in my WeRateDog analysis project.

Gather the data:

In the gathering process, I'm required to gather data from different sources:

1- twitter_archive_enhanced.csv:

I download this dataset manually in Udacity classroom, the dataset is contained basic tweet data (tweet ID, timestamp, text, etc.), However, in this dataset I didn't face any obstacle it was already in the CSV format. I only used the pd.read_csv function to open it.

```
twitter_archive_df = pd.read_csv('twitter-archive-enhanced.csv')
```

2- image_predictions.tsv:

In this dataset I had to download it programmatically from the Udacity server via request library. This dataset is containing the image predictions alongside each tweet ID, image URL that the instructor ran the images in the WeRateDogs Twitter archive through a neural network that can classify breeds of dogs. However, I download this dataset in TSV format.

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

with open (url.split('/')[-1], mode='wb') as file:
    file.write(request.content)
image_predictions = pd.read_csv('image_predictions.tsv', sep='\t')
image_predictions.head()
```

3- tweet-json.txt:
As the twitter reject my request to access their API, I will download the data provided by Udacity. Which is in txt format, the data in the file is stored as JSON so I had to extract the JSON strings line by line and append it to an empty list then convert it to a dataframe. This file contains the tweet id, retweet count, and favorite count.

```python
list_tweet = []

with open('tweet-json.txt', mode='r') as json_file:
    for a in json_file:
        json_data = json.loads(a)
        tweet_id = json_data['id']
        favorite_count = json_data['favorite_count']
        retweet_count = json_data['retweet_count']

        list_tweet.append({'tweet_id': tweet_id,
                        'favorite_count': favorite_count,
                        'retweet_count': retweet_count})

#Create new df
json_df = pd.DataFrame(list_tweet, columns = ['tweet_id', 'favorite_count', 'retweet_count'])


#Save the df in file
json_df.to_csv('tweet-json2.txt', encoding = 'utf-8', index=False)
```

Assess the data

In the assess process I'm required to detect and document at least eight (8) quality issues and two (2) tidiness issue. Wherefore, I used both visual assessment and programmatic assessement to assess the data. I discover the data has many defects, but I'm only able to mention the following:

Quality issues:

1.Tweet ID should convert to strings (across the tables) + Timestamp should be datetime type

2.Rating are extracted incorrectly from text, some texts include rating with fractional part.

3.Source hard to read beacuse it inside HTML tags

4.Some name are is Inconsistent such as (a, an, very , the, not, such, quite, o, my, by)

5.Duplicate values in expanded_urls which means its a retweets, therfore must be deleted.

6.Column expanded_urls has missing values. Since the rating based on the image included in the tweet we must delete the null values.

7.inconsistent format for the dogs breeds (some lower and other are upper case) image_predication dataset.

8.The columns doggo,floofer,pupper,and puppo assign missing values as (None) instead of NaN.

9.Remove columns with missing data and unnecessary columns such as:

twitter_archive_df: in_reply_to_status_id in_reply_to_user_id retweeted_status_id retweeted_status_user_id retweeted_status_timestamp doggo,floofer,pupper,and puppo

image_predictions_clean: img_num

Note: I notice that the image prediction has false predictions such as (suit, seat belt, cardigan, computer, etc..) but fixing it will take so much time.

Tidiness issues:

1.twitter_archive_df: Replace the columns (doggo,floofer,pupper,and puppo) with a one column specify the type

2.Combine the dataframes into one dataframe because it related to each other.

Clean the data:

In the cleaning process I used the technique called Define,Code,Test to accomplish the cleaning effectively, First, before I start cleaning, I create copies of the datasets, then I start cleaning the issues below:

Quality issue:

1: convert the tweet_id columns to string in all dataframes

```
twitter_archive_df_clean["tweet_id"] = twitter_archive_df_clean["tweet_id"].astype(str)
```

```
image_predictions_clean["tweet_id"] = image_predictions_clean["tweet_id"].astype(str)
tweet_json_clean["tweet_id"] = tweet_json_clean["tweet_id"].astype(str)
```

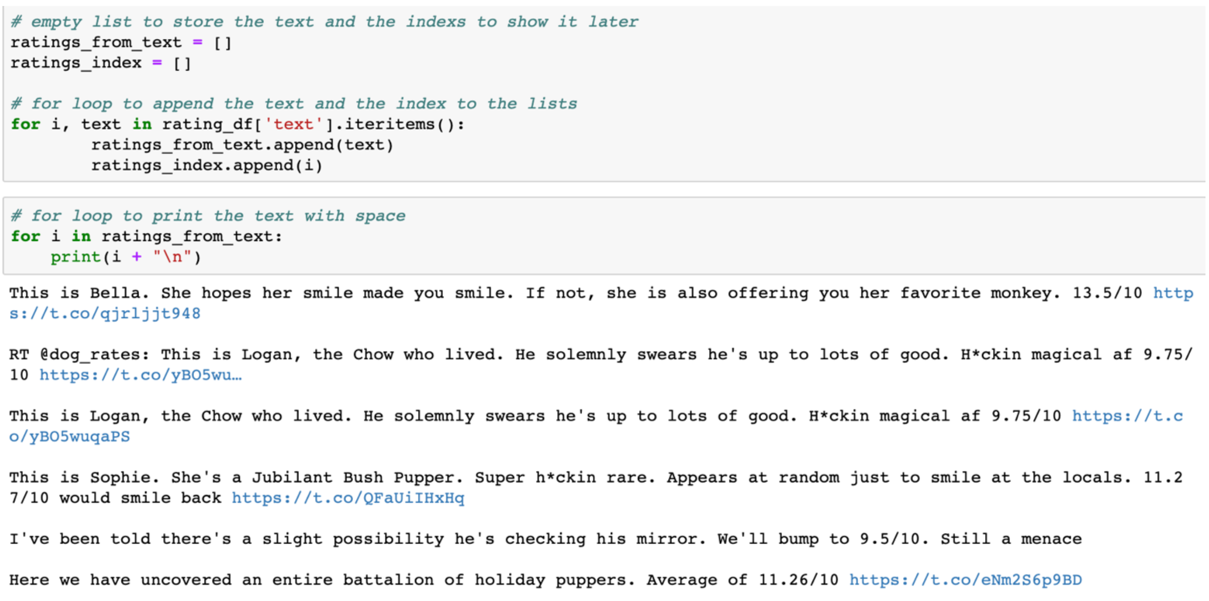
Convert the timestamp to datetime type

```
twitter_archive_df_clean['timestamp'] = pd.to_datetime(twitter_archive_df_clean['timestamp'])
```

2: Rating are extracted incorrectly from text, some texts include rating with fractional part.

```
# change the type to float
twitter_archive_df_clean["rating_numerator"] = twitter_archive_df_clean["rating_numerator"].astype(float)
```

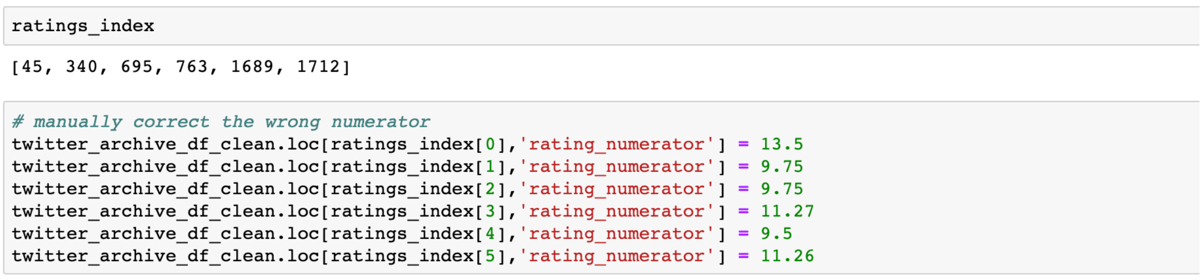
```
# extract text wanted
rating_df = twitter_archive_df_clean[twitter_archive_df_clean['text'].str.contains(r"(\d+\.\d*\/\d+)")]
```



```
# empty list to store the text and the indexs to show it later
ratings_from_text = []
ratings_index = []

# for loop to append the text and the index to the lists
for i, text in rating_df['text'].iteritems():
        ratings_from_text.append(text)
        ratings_index.append(i)
```

```
# for loop to print the text with space
for i in ratings_from_text:
    print(i + "\n")
```

```
This is Bella. She hopes her smile made you smile. If not, she is also offering you her favorite monkey. 13.5/10 http
s://t.co/qjrljjt948

RT @dog_rates: This is Logan, the Chow who lived. He solemnly swears he's up to lots of good. H*ckin magical af 9.75/
10 https://t.co/yBO5wu…

This is Logan, the Chow who lived. He solemnly swears he's up to lots of good. H*ckin magical af 9.75/10 https://t.c
o/yBO5wuqaPS

This is Sophie. She's a Jubilant Bush Pupper. Super h*ckin rare. Appears at random just to smile at the locals. 11.2
7/10 would smile back https://t.co/QFaUiIHxHq

I've been told there's a slight possibility he's checking his mirror. We'll bump to 9.5/10. Still a menace

Here we have uncovered an entire battalion of holiday puppers. Average of 11.26/10 https://t.co/eNm2S6p9BD
```

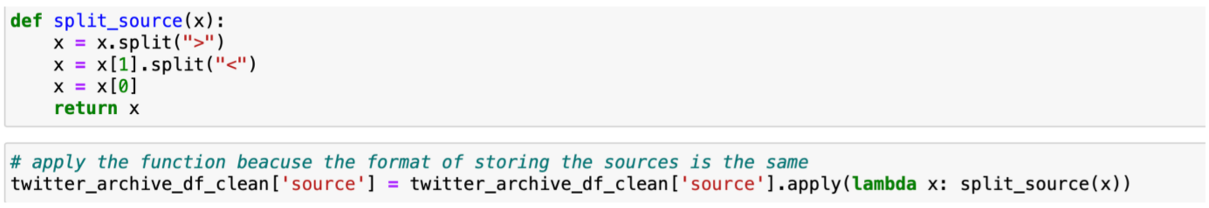


```
ratings_index
```

```
[45, 340, 695, 763, 1689, 1712]
```

```
# manually correct the wrong numerator
twitter_archive_df_clean.loc[ratings_index[0],'rating_numerator'] = 13.5
twitter_archive_df_clean.loc[ratings_index[1],'rating_numerator'] = 9.75
twitter_archive_df_clean.loc[ratings_index[2],'rating_numerator'] = 9.75
twitter_archive_df_clean.loc[ratings_index[3],'rating_numerator'] = 11.27
twitter_archive_df_clean.loc[ratings_index[4],'rating_numerator'] = 9.5
twitter_archive_df_clean.loc[ratings_index[5],'rating_numerator'] = 11.26
```

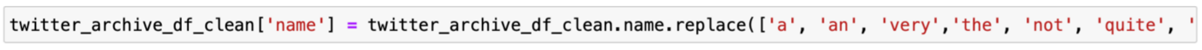
3: Extract the sourse from the tags by using custom function suit the format of the source



```
def split_source(x):
    x = x.split(">")
    x = x[1].split("<")
    x = x[0]
    return x
```

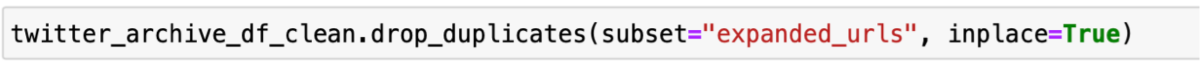
```
# apply the function beacuse the format of storing the sources is the same
twitter_archive_df_clean['source'] = twitter_archive_df_clean['source'].apply(lambda x: split_source(x))
```

4: Replace the wrong names with null



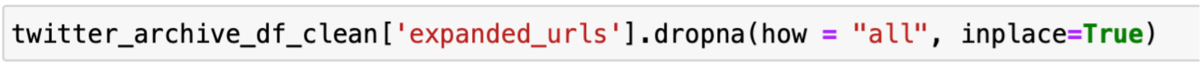
```
twitter_archive_df_clean['name'] = twitter_archive_df_clean.name.replace(['a', 'an', 'very','the', 'not', 'quite', '
```

5: Duplicate values in expanded_urls which means its retweets, therfore must be deleted.



```
twitter_archive_df_clean.drop_duplicates(subset="expanded_urls", inplace=True)
```

6: Column expanded_urls has missing values. Since the rating based on the image included in the tweet we must delete the null values.



```
twitter_archive_df_clean['expanded_urls'].dropna(how = "all", inplace=True)
```

7: Inconsistent format for the dogs breeds (some lower and other are upper case) also remove underscores for better analysis (image_predication dataframe).

```
image_predictions_clean['p1'] = image_predictions_clean['p1'].apply(lambda x: x.replace('_', ' '))
image_predictions_clean['p1'] = image_predictions_clean['p1'].apply(lambda x: x.capitalize())
```

```
image_predictions_clean['p2'] = image_predictions_clean['p2'].apply(lambda x: x.replace('_', ' '))
image_predictions_clean['p2'] = image_predictions_clean['p2'].apply(lambda x: x.capitalize())
```

```
image_predictions_clean['p3'] = image_predictions_clean['p3'].apply(lambda x: x.replace('_', ' '))
image_predictions_clean['p3'] = image_predictions_clean['p3'].apply(lambda x: x.capitalize())
```

8: The columns doggo,floofer,pupper,and puppo assign missing values as (None) instead of NaN.

```
twitter_archive_df_clean['doggo'].replace('None', np.nan, inplace = True)
twitter_archive_df_clean['floofer'].replace('None', np.nan, inplace = True
twitter_archive_df_clean['pupper'].replace('None', np.nan, inplace = True)
twitter_archive_df_clean['puppo'].replace('None', np.nan, inplace = True)
```

9: Drop the unnecessary column

```
twitter_archive_df_clean.drop(['in_reply_to_status_id', 'in_reply_to_user_id','retweeted_status_id','retweeted_statu
           'retweeted_status_timestamp', 'doggo', 'floofer', 'pupper', 'puppo'], axis=1, inplace=True)
```

```
image_predictions_clean.drop(['img_num'], axis=1, inplace=True)
```

Tidiness Issue:

1: Replace the columns (doggo,floofer,pupper,and puppo) with a one column specify the type. Note: I notice that some has 2 values so for a vaild analysis I will keep both.

```
# merge into column
twitter_archive_df_clean['dog_stage'] = twitter_archive_df_clean.doggo + twitter_archive_df_clean.floofer + twitter_arc

# handle multiple stages
twitter_archive_df_clean.loc[twitter_archive_df_clean.dog_stage == 'doggopupper', 'dog_stage'] = 'doggo, pupper'
twitter_archive_df_clean.loc[twitter_archive_df_clean.dog_stage == 'doggopuppo', 'dog_stage'] = 'doggo, puppo'
twitter_archive_df_clean.loc[twitter_archive_df_clean.dog_stage == 'doggofloofer', 'dog_stage'] = 'doggo, floofer'

# handle missing values
twitter_archive_df_clean.loc[twitter_archive_df_clean.dog_stage == '', 'dog_stage'] = np.nan
```

2: Merge the all the dataframes into one dataframe

```
# merge twitter_archive_clean_df with tweet_json_clean
full_cleaned_df = pd.merge(twitter_archive_df_clean, tweet_json_clean, how = 'inner', on = ['tweet_id']).copy()

# merge full_cleaned_df with image_predictions_clean
full_cleaned_df = pd.merge(full_cleaned_df, image_predictions_clean, how = 'inner', on = ['tweet_id']).copy()
```

Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv"

```
full_cleaned_df.to_csv('twitter_archive_master.csv', encoding = 'utf-8')
```