Wrangling And Analyze Data Project

Introduction:

This project is aims to practice wrangling data that is inaccurate, inconsistent, erroneous, and messy for analysis by using python libraries, to get precise results.

The wrangling data consists of three steps:

1- Gathering data.
2- Assessing data.
3- Cleaning data.

In addition to other task steps:

4- Storing data.
5- Analyzing, and visualizing data.
6- Reporting (the current step)

Note: I've used the Microsoft word to write this report and then convert it to ***.pdf*** format.

1- Gathering data.

The three pieces of data were gathered from different resources:

- **twitter_archive_enhanced.csv:** Downloaded and filtered by Udacity from WeRateDogs is twitter account that rates the dogs with a funny comment.
- **image_predictions.tsv:** Downloaded by using the Requests library.
- **tweet_json.txt:** Additional data (e.g., favorite_count and retweet_count) was collected from the WeRateDogs Twitter account, using the object API and Tweepy library and then saved (JSON data) to a text file.

2- Assessing data.

The three files were visually assessed using Microsoft Excel and programmatically using Python libraries in Jupyter Notebook for quality and tidiness issues.

Quality issues

Quality issues is the data content issues. The low-quality data is known as dirty data.

The features of data to be high quality are completeness, validity, accuracy, consistency.

We detected and documented more than (8) quality issues as following:

df_twit_arch table (data from twitter_archive_enhanced.csv)

- Missing values.
- Erroneous datatypes (tweet_id, timestamp, retweeted_status_timestamp).
- Inaccurate denominators- some of them don't equal 10 in (rating_denominator).
- All names that start with lowercase letters (such as: very, a, an, his, mad, just, ...) are incorrect, and most of them are "None" value.
- Using NaN for missing values in the columns (object type) instead of None. Despite they are different data types in Python, Pandas treat them similarly. So, there is no need to change NaN to None.
- Selecting the rows with only original ratings (no retweets) and have images.

image_predic table

- Missing records (2075 of 2356).
- Duplicated image url (jpg_url).
- Delete rows that have no (jpg_url).
- Non-descriptive column names (p1, p2, p3)

tweet_attributes table

- Erroneous datatypes (created_at).
- Missing records (2328 of 2356).

Tidiness issues

Tidiness issues is the structure of data issues. The untidy data is known as messy data.

We detected and documented more than two quality issues as following:

- Four variables in one column in df_twit_arch table (doggo, floofer, pupper, puppo).
- Tweet_attributes and image_predic should be part of df_twit_arch.
- Created_at in tweet_attributes table should be split into date and time. On the other hand, this column will be dropped because it will not be used for analysis.
- Drop unwanted columns such as (created_at, truncated, Text, ..., and the retweet attributes).
- Calculte the dog rating in new column by dividing the rating_numerator by rating_denominator.

3. Cleaning data

In this step, we fixed the quality and tidiness issues documented in the assessing step. For high quality of data, we first fixed missing values, then tidiness issues, and finally the quality issues.

The cleaning process is applied on a copied data and arranged in three steps: Define, Code, and Test. All the cleaning processes are defined and documented in wrangle_act.ipynb.

On the other hand, there are some processes that need further explanation such as:

- Merge four columns (doggo, floofer, pupper, puppo) in one column (dog_stage).

Before fixing this step, we tested the data, and the result was:

```
archive_clean.doggo.value_counts() ,'' , archive_clean.floofer.value_counts(), '' , archive_clean.pupper.value_counts(), '' , archive_clean.puppo.value_counts ().
```

```
(None 1920, doggo 74), ,(None 1986, floofer 8)' '(None 1782, pupper 212) ' ',
 (None 1971, puppo 23)
```

After fixing this issue, we get the following result:

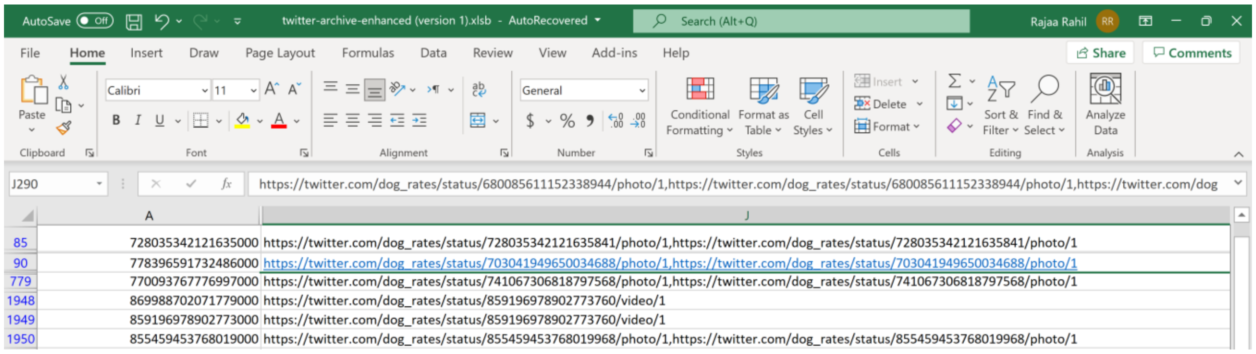
```
archive_clean.dog_stage.value_counts()
```

```
pupper     212
doggo       63
puppo       23
floofer      8
```

The difference in the number of doggo stage (74, 63) is due to a mistake that may have occurred during extracting data from the text and then two types of stages were assigned to the same dog in the original data.
The for loop in my code assigned only one stage type for each dog, so it will assign the last stage type if there is more than one stage.

This mistake may need to fix it manually or to re-extract the stages from the text, and due to time constraints, I didn't fix this issue.

- I downloaded the file twitter_archive_enhanced.csv twice, once when I invoked it in the wrangle_act project and re-uploaded it to check it visually in excel.



J290 | https://twitter.com/dog_rates/status/680085611152338944/photo/1,https://twitter.com/dog_rates/status/680085611152338944/photo/1,https://twitter.com/dog

	A	J
85	728035342121635000	https://twitter.com/dog_rates/status/728035342121635841/photo/1,https://twitter.com/dog_rates/status/728035342121635841/photo/1
90	778396591732486000	https://twitter.com/dog_rates/status/703041949650034688/photo/1,https://twitter.com/dog_rates/status/703041949650034688/photo/1
779	770093767776997000	https://twitter.com/dog_rates/status/741067306818797568/photo/1,https://twitter.com/dog_rates/status/741067306818797568/photo/1
1948	869988702071779000	https://twitter.com/dog_rates/status/859196978902773760/video/1
1949	859196978902773000	https://twitter.com/dog_rates/status/859196978902773760/video/1
1950	855459453768019000	https://twitter.com/dog_rates/status/855459453768019968/photo/1,https://twitter.com/dog_rates/status/855459453768019968/photo/1

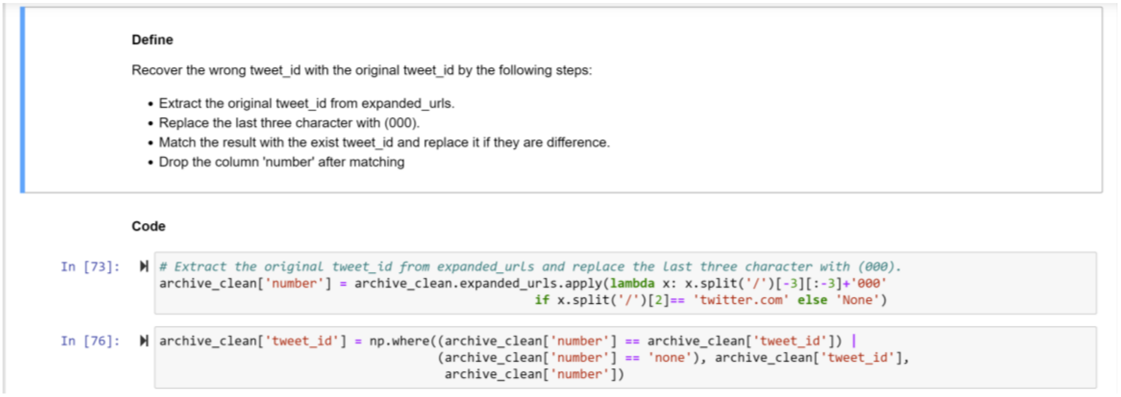
We can note that tweet_id consists of 18 digits and extracted from expanded_urls with replacing the last three-digit with (000), such as example :

tweet_id: 881536004380872000

expanded_urls: https://twitter.com/dog_rates/status/881536004380872706/video/1

Some of tweet_id doesn't match the number inside the expanded_urls.

I selected a sample (outside my workspace) from this file and tried to solve and document this issue:



Define

Recover the wrong tweet_id with the original tweet_id by the following steps:

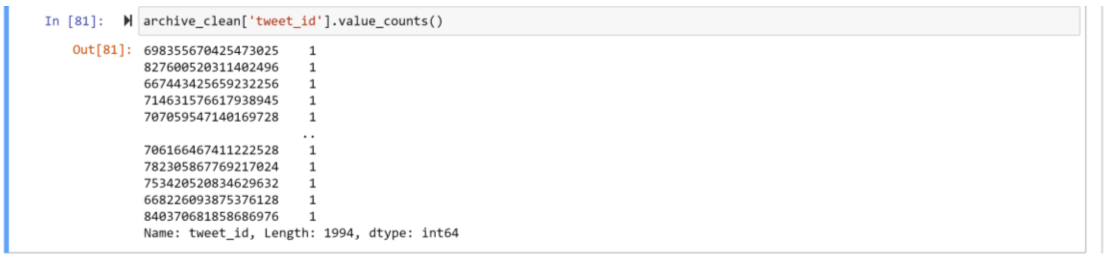
- Extract the original tweet_id from expanded_urls.
- Replace the last three character with (000).
- Match the result with the exist tweet_id and replace it if they are difference.
- Drop the column 'number' after matching

Code

```
In [73]:  # Extract the original tweet_id from expanded_urls and replace the last three character with (000).
          archive_clean['number'] = archive_clean.expanded_urls.apply(lambda x: x.split('/')[-3][:-3]+'000'
                                                                       if x.split('/')[2]== 'twitter.com' else 'None')
```

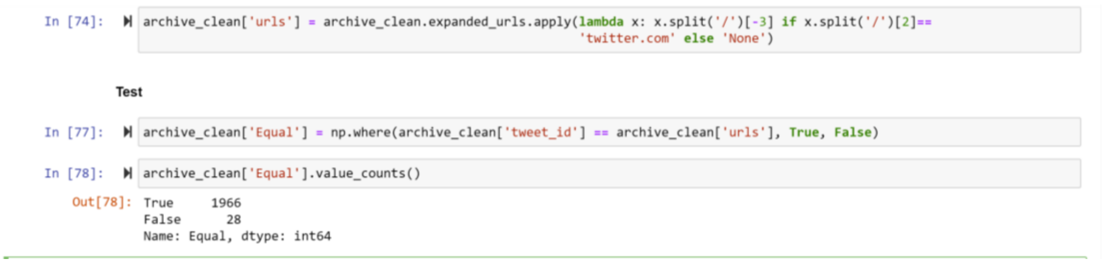
```
In [76]:  archive_clean['tweet_id'] = np.where((archive_clean['number'] == archive_clean['tweet_id']) |
                                              (archive_clean['number'] == 'none'), archive_clean['tweet_id'],
                                               archive_clean['number'])
```

After checking the data in the main workspace, I found that:



```
In [81]:  archive_clean['tweet_id'].value_counts()

Out[81]:  698355670425473025    1
          827600520311402496    1
          667443425659232256    1
          714631576617938945    1
          707059547140169728    1
                               ..
          706166467411222528    1
          782305867769217024    1
          753420520834629632    1
          668226093875376128    1
          840370681858686976    1
          Name: tweet_id, Length: 1994, dtype: int64
```



```
In [74]:  archive_clean['urls'] = archive_clean.expanded_urls.apply(lambda x: x.split('/')[-3] if x.split('/')[2]==
                                                                'twitter.com' else 'None')
```

Test

```
In [77]:  archive_clean['Equal'] = np.where(archive_clean['tweet_id'] == archive_clean['urls'], True, False)

In [78]:  archive_clean['Equal'].value_counts()

Out[78]:  True     1966
          False      28
          Name: Equal, dtype: int64
```

The 28 th False related to none value in ‘urls’ column. Therefore, I removed this issue from my project, but I preferred to share this solution with you.

4- Storing Data

We saved gathered, assessed, and cleaned master dataset to a CSV flat file that named "twitter_archive_master.csv".

Conclusion

In conclusion, Data wrangling is very important to obtain a high-quality analysis result without time-consuming. The wrangling process consists of three repeatable steps: gathering, assessing, and cleaning data. We practiced these steps to complete our project requirements. We defined, coded to fix, and tested each issue. Finally, we stored the cleaned data into a flat-file to analyze and visualize its data.