ALX-T UDACITY
DATA ANALYSIS NANO DEGREE

DATA WRANGLING REPORT

June 2022

PREPARED BY
James Idowu

PRESENTED TO
ALX-T UDACITY
Data Analyst
Nano Degree Program

BACKGROUND

The aim of this project is to wrangle twitter data of the Twitter user WeRateDogs.

WeRateDogs is a twitter account that rates user submitted dog pictures with humorous ratings and comments.

The account has been active since November 2015, but the dataset in our wrangling effort only covers the period from 2015 to 2017.

OBJECTIVE

Real-world data rarely comes clean and the dataset in this case is no exception.

The dataset needed to be wrangled come to us from three separate sources and in three different formats.

Using Python and its libraries, the objective is to gather all these data from their various sources and formats, assess their quality and tidiness and then clean them.

The final aim of the wrangling effort is to have a relatively clean dataset which would be conducive for getting meaningful insights from WeRateDogs' twitter data from 2015 - 2017

PROJECT DETAILS

The steps taken in this data wrangling process are as follows:

Gathering the data:

The data was spread across three different places. The aim was to get all these three datasets into our local system as efficiently as possible.

Two of the files(**image-predictions.tsv** and **twitter-archive-enhanced.csv**) were read into our system programmatically using python request library.

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

url_archive =
'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/59a4e958_twitter-archive-enhanced/twitter-archive-enhanced.csv'

r_image = requests.get(url_image)
r_archive = requests.get(url_archive)
```

The last file(**tweet-json.txt**) was downloaded from the link provided by Udacity. It was read line by line and read into jupyter using

```
with open('datasets/tweet-json.txt') as file:
  tweets = []

  for line in file:
    tw = json.loads(line)
    # Appending a dictionary of selected features into the 'tweets' list
    tweets.append({
        'tweet_id': tw['id'],
        'retweet_count':tw['retweet_count'],
        'favorite_count':tw['favorite_count']
    })
```

CONTD.

The files were read into a pandas dataframes for further wrangling processes

```
df_image = pd.read_csv('datasets/image_predictions.tsv', sep='\t')
df_archive = pd.read_csv('datasets/twitter-archive-enhanced.csv')
df_tweets= pd.DataFrame(tweets)
```

Assessment of the data:

the dataset were both visually and programmatically assessed to ascertain their quality and what cleaning processes would be required to get them clean enough for future analysis of the data.

```
# Visual assessment of image prediction dataframe

df_image.sample(10)
```

	tweet_id	jpg_url	img_num	p1	p1_c
676	683462770029932544	https://pbs.twimg.com/media/CXwlw9MWsAAc-JB.jpg	1	Italian_greyhound	0.399
1661	811744202451197953	https://pbs.twimg.com/media/C0PlCQjXAAA9TIh.jpg	1	Pekinese	0.386
1389	766423258543644672	https://pbs.twimg.com/media/CqLh4yJWcAAHomv.jpg	2	keeshond	0.995
54	667044094246576128	https://pbs.twimg.com/media/CUHREBXXAAE6A9b.jpg	1	golden_retriever	0.765
1369	761976711479193600	https://pbs.twimg.com/media/CpMVxoRXgAAh350.jpg	3	Labrador_retriever	0.475
784	690021994562220032	https://pbs.twimg.com/media/CZNzV6cW0AAsX7p.jpg	1	badger	0.289
1770	827600520311402496	https://pbs.twimg.com/media/C3w6RYbWQAAEQ25.jpg	1	Pembroke	0.325
1261	748932637671223296	https://pbs.twimg.com/media/CmS-QkQWAAAkUa-.jpg	1	borzoi	0.742
1945	862096992088072192	https://pbs.twimg.com/media/C_blo7QXYAAGfPu.jpg	2	chow	0.677
198	669625907762618368	https://pbs.twimg.com/media/CUr9NjgU8AEpf5w.jpg	1	seat_belt	0.874

Sample of Image prediction dataframe Visual Assessment

```
#checking for duplicated images in the image prediction dataset

nums_of_duplicates = df_image['jpg_url'].duplicated().sum()

print(f"There are ({nums_of_duplicates}) Duplicated images in Image Prediction Dataset")
```

```
There are (66) Duplicated images in Image Prediction Dataset
```

```
# Assessing the tweet counts & fovourite dataset

print(df_tweets.shape)

df_tweets.info()
```

```
(2354, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2354 entries, 0 to 2353
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2354 non-null   int64
 1   retweet_count   2354 non-null   int64
 2   favorite_count  2354 non-null   int64
dtypes: int64(3)
```

Sample of Image prediction and tweet-json dataframe Programatic Assessment

Contd.

```
# Assessing the twitter archive dataset

print(df_archive.shape)

df_archive.info()
```

```
(2356, 17)
<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     float
 2   in_reply_to_user_id         78 non-null     float
 3   timestamp                   2356 non-null   objec
 4   source                      2356 non-null   objec
 5   text                        2356 non-null   objec
 6   retweeted_status_id         181 non-null    float
 7   retweeted_status_user_id    181 non-null    float
 8   retweeted_status_timestamp  181 non-null    objec
 9   expanded_urls               2297 non-null   objec
 10  rating_numerator            2356 non-null   int64
 11  rating_denominator          2356 non-null   int64
 12  name                        2356 non-null   objec
 13  doggo                       2356 non-null   objec
 14  floofer                     2356 non-null   objec
 15  pupper                      2356 non-null   objec
 16  puppo                       2356 non-null   objec
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

Sample of Twitter Archive dataframe Programatic Assessment

Cleaning of the data:

Once the datasets were assessed both visually and programatically, the following quality and tidyness issues were identified for cleaning.

Quality Issues

1. We want only original tweets. No retweets and replies.
2. Change timestamp from string datatype to datetime datatype.
3. Have all the Dog stages in one column.
4. Strip the source table of its Html tags so that only the text is left.
5. Removing Duplicates image rows from the image prediction Dataframe.
6. Removing image predictions that are not dogs from the the image prediction dataframe.
7. Creating a column with the most appropriate dog image recognition.
8. Dropping all the unnecessary columns from the image recognition dataframe.

Tidyness issues

1. Remove shortened url from text column as it is redundant.
2. Merging the three dataframes into on dataframe for analysis.

Kindly note, the cleaning process was iterative and other quality tidyness issues were identified along the way and were taken care of. The above issues are just the one's that were identified at the beginning of the cleaning process.

Cleaning of the data Contd.

Before beginning the cleaning process, copies of each dataframe were made. This was done in case there is a need to compare changes between the cleaned data and the originals.

The cleaning process followed the **Define, Code** and **Test** methodology. Each quality or tidyness issue was defined, code to fix it was written and then it's result was tested to see if the issue was fixed.

Clean

Define

- Strip the source column in archive_clean dataframe of its html tags so that only the text is left.

Code

```
[ ]   archive_clean['source'] = archive_clean['source'].str.replace(r'<[^<>]*>', '', regex=True)
```

Test

```
[ ]   archive_clean['source'].value_counts()
```

```
Twitter for iPhone     1964
Vine - Make a Scene      91
Twitter Web Client       31
TweetDeck                11
Name: source, dtype: int64
```

Sample of the Define, Code, Test Methodology Used In Cleaning

Cleaning of the data Contd.

Before beginning the cleaning process, copies of each dataframe were made. This was done in case there is a need to compare changes between the cleaned data and the originals.

The cleaning process followed the **Define, Code** and **Test** methodology. Each quality or tidyness issue was defined, code to fix it was written and then it's result was tested to see if the issue was fixed.

Clean

Define

- Strip the source column in archive_clean dataframe of its html tags so that only the text is left.

Code

```
[ ]    archive_clean['source'] = archive_clean['source'].str.replace(r'<[^<>]*>', '', regex=True)
```

Test

```
[ ]    archive_clean['source'].value_counts()
```

```
Twitter for iPhone     1964
Vine - Make a Scene      91
Twitter Web Client       31
TweetDeck                11
Name: source, dtype: int64
```

Sample of the Define, Code, Test Methodology Used In Cleaning

Cleaning of the data Contd.

Once cleaning was done using various libraries from pandas built-in methods to using beautiful soup for parsing webscrapped Wikipedia data, the cleaned data was merged into a single dataframe and saved in the local system as a csv file for possible fure uses.

```
df = archive_clean.merge(image_clean[['dog_breed_p','jpg_url', 'tweet_id']])
```

```
df = df.merge(df_tweets[['retweet_count','favorite_count', 'tweet_id']])
```

Merging all three dataframes into one

```
[ ]  df.to_csv('datasets/weratedogs_dataset_merged.csv', index=False)
```

Saving the cleaned dataframe to file

CONCLUSION

The wrangling project was an interesting one and it afforded me the opportunity to practicalize what was taught during the course and also it exposed me to wide variety of methods in solving problems usually encountered during data wrangling.

The project and exposure it afforded has been very rewarding