Report on Data Wrangling Steps: Gather, Assess, and Clean

Step 1: Gathering Data

We were required to gather data from 3 sources.

The first source was the “Enhanced WeRateDogs Twitter” archive, which was provided as a .csv file "twitter-archive-enhanced.csv". We extracted this data into a dataframe using the pandas read_csv function.

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

We then made a copy of this dataframe and used the copy for further wrangling.

df_tweet = pd.DataFrame.copy(df_tweet_master)

The second source was the image predictions file obtained by running dog images through a neural network. The file was stored on the cloud (udacity server) as a .tsv file and the url to the file was provided to us. We used the "Requests" library to download and extract the contents of the file.

```
import requests

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
response = requests.get(url)
open(url.split('/')[-1], mode = 'wb').write(response.content)
```

The contents of the file were then be imported into a separate dataframe.

df_img_pred = pd.read_csv('image-predictions.tsv', sep='\t')

The third source was a text file with JSON data of each tweet's retweet count, favorite count, etc. Using the tweet IDs in the WeRateDogs Twitter archive extracted above, we queried the Twitter API for each tweet's JSON data and saved it as tweet_json.txt. We did this using Python's Tweepy library. Each tweet's JSON data was written to its own line, which we then wrote line by line into a pandas DataFrame.

```
import json

with open('tweet_json.txt', 'r') as json_data:
    df_list = []
    for line in json_data.readlines():
        data = json.loads(line)
        tweet_id = data['id']
        retweets = data['retweet_count']
        favorites = data['favorite_count']
        tweet_id, retweets, favorites
        df_list.append({'tweet_id': tweet_id,
                'retweet_count': retweets,
                'favorite_count': favorites})
```

```
df_retweet = pd.DataFrame(df_list, columns = ['tweet_id', 'retweet_count', 'favorite_count'])
```

Step 2: Assessing Data

- We were required to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.
- Cleaning included merging individual pieces of data according to the rules of tidy data.

Quality Issues Observed

1. Some entries are retweets. We only want the original ratings. The retweets need to be deleted.
2. The 'source' column has dirty data with a link to download twitter on a phone. We will drop this column. So can the "in_reply_to_status_id", "in_reply_to_user_id", "retweeted_status_id","retweeted_status_user_id" and "retweeted_status_timestamp" because they are not required anymore.
3. Some of the animals in the dataset are not dogs at all. @dog_rates only rates dogs.
4. The denominator for most ratings is 10. Entries with denominator greater than 10 seem like ratings for multiple dogs and are valid. But some have denominators other than 10 and are not ratings for multiple dogs.
5. Entries with no dog rating information need to be deleted.
6. There are several dog names which are wrong like "a", "the", "an", and "this". A lot of the dog names are missing and need to be filled in wherever possible.
7. Some dog stages do not look right. Some are missing and need filling in. Some have multiple dog stage entries.
8. There should be a single column for dog stage after consolidating information in the doggo, pupper, puppo and floofer columns. As it stands right now, the doggo, pupper, puppo and floofer columns cannot be used for any analysis.
9. The predictions from the neural network are not all accurate. It predicts dogs as laptops and toilet paper and desktop computers. While we are noting this down as an issue, we will not be fixing it for this project.

Tidyness Issues Observed

1. Having merged data from the doggo, floffer, pupper and puppo columns, we will no longer need these columns.
2. Dog breed Column needs to be added to the main dataframe. We need to update the column using data in the image predictions file, which we extracted to dataframe df_img_pred.
3. We are assuming that the most accurate breed prediction is in column "p1". Here, we are going to assume that the neural network is actually providing us with dog breeds and not paper towels and laptops. We need to also rename column "p1" to something more meaningful like "predicted_breed". We do not need the other columns and can drop them.
4. We need to merge the retweet_count and favorite_count fields into the main dataframe. We need to update the columns using data from the json file, which we extracted to dataframe df_retweets.

Step 3: Cleaning Data

- The quality and tidyness issues observed above will need to be cleaned so we only analyze clean data.
- For each of the issue observed, we split the cleaning process into
    - Define: We defined what needs to be done to fix the issue and clean the data.
    - Code: We wrote code that will help implement the fix described in the Define phase.
    - Test: We tested if the code correctly fixed the quality or tidyness issue.

Quality Issue 1:

- Some entries are retweets. We only want the original ratings. The retweets need to be deleted.

```
# Define:

# The dataframe consists of retweets as well, which need to be excluded. We are only interested ir
# Let us find out how many records are retweets

retweet_df = df_tweet.query('retweeted_status_id != "NaN"')
len(retweet_df)
```

```
181
```

```
# Code:

## Original tweets are the ones where the retweeted_status_id is null.

df_tweet = df_tweet.query('retweeted_status_id == "NaN"')
```

```
# Test:

## 181 entries are retweets. The original entries should be 2356 - 181 = 2175 if the code worked.

len(df_tweet)
```

```
2175
```

```
df_tweet.info()
```

Quality Issue 2:

- The df_tweet_e['source'] column has dirty data with a link to download twitter on a phone. Moreover, the same value repeats over a vast majority of the entries, rendering it useless. We will drop this column.
- So can the "in_reply_to_status_id", "in_reply_to_user_id", "retweeted_status_id","retweeted_status_user_id" and "retweeted_status_timestamp" because they are not required anymore.

```
# Define:

# The columns "source", in_reply_to_status_id", "in_reply_to_user_id", "retweeted_status_id","retweeted_status_user_id"
# and "retweeted_status_timestamp" need to be removed from the dataframe.

df_tweet.shape
```

```
(2175, 17)
```

```
# Code:

# We will use the drop function to remove these columns from the dataframe.

df_tweet.drop(columns = ['source','in_reply_to_status_id', 'in_reply_to_user_id', "retweeted_status_id",
                         "retweeted_status_user_id", "retweeted_status_timestamp"
                        ], inplace = True)
```

```
# Test:

# Let us make sure these columns are dropped.

df_tweet.head(2)
```

	tweet_id	timestamp	text	expanded_urls	rating_numerator	rating_denominator	name	doggo	floof

Quality Issue 3:

- Some of the animals in the dataset are not dogs at all. @dog_rates only rates dogs.
- @dog_rates only rates dogs. They do not rate other animals.
- There are several records for other animals like bear, lion, tyrannosaurus rex, tiger, lobster, etc.
- There are also things like carrot.
- A frustrated admin for the @dog_rates account repeated says "We only rate dogs".
- We will extract and remove all entries, where the tweet contains "only rate dogs" or "only. rate. dogs".

```
# Define:

# @dog_rates only rates dogs. They do not rate other anmimals.
# There are several records for other animals like bear, lion, tyrannosaurus rex, tiger, lobster, etc.
# There are also things like carrot.
# A frustrated admin for the @dog_rates account repeated says "We only rate dogs".
# We will extract and remove all entries, where the tweet contains "only rate dogs" or "only. rate. dogs".

df_not_dogs = df_tweet[df_tweet['text'].str.contains("only rate dogs", "only. rate. dogs")]
df_not_dogs
```

```
# Code:

# Looks like there are 55 records, where the animal is not a dog.
# We will now alter df_tweet to only include records for dogs.

df_tweet = df_tweet[~df_tweet['text'].str.contains("only rate dogs", "only. rate. dogs")]
df_tweet.shape
```

```
(2120, 11)
```

```
# Test:

#Let us ensure all the 2120 entries are for dogs only.

df_not_dogs = df_tweet[df_tweet['text'].str.contains("only rate dogs", "only. rate. dogs")]
df_not_dogs.shape
```

```
(0, 11)
```

Quality Issue 4:

- The denominator for most ratings is 10.
- But there are dog ratings with a denominator other than 10, that may need to be corrected.

We first extracted all rows with denominator not equal to 10 to observe the data and find some insights.

```
df_dnot10 = df_tweet.query('rating_denominator != 10')
```

- Some ratings with denominator greater than 10 seem like ratings for multiple dogs and are valid.
- The denominator in such cases is a multiple of 10.
- But some have denominators other than a multiple of 10 and do not seem to be valid ratings.
- We may need to correct these ratings whose denominator is not a multiple of 10.

```
d_dmod10 = df_tweet.query('rating_denominator%10 != 0 or rating_denominator == 0')
```

We found that there were 7 such records. We then tried to look for the correct rating in the tweet text.

```
d_dmod10.groupby('tweet_id').text.value_counts()
```

```
tweet_id            text
666287406224695296  This is an Albanian 3 1/2 legged  Episcopalian. Loves well-polished hardwood flooring. Penis
9/10 https://t.co/d9NcXFKwLv                                        1
682808988178739200  I'm aware that I could've said 20/16, but here at WeRateDogs we are very professional. An in
ng scale is simply irresponsible                                    1
682962037429899265  This is Darrel. He just robbed a 7/11 and is in a high speed police chase. Was just spotted
er 10/10 https://t.co/7EsP8LmSp5                                    1
740373189193256964  After so many requests, this is Bretagne. She was the last surviving 9/11 search dog, and ou
4/10. RIP https://t.co/XAVDNDaVgQ                                   1
810984652412424192  Meet Sam. She smiles 24/7 &amp;amp; secretly aspires to be a reindeer. \nKeep Sam smiling by cli
ng this link:\nhttps://t.co/98tB8y7y7t https://t.co/LouL5vdvxx      1
832088576586297345  @docmisterio account started on 11/15/15
1
832464395298406400  @jonnysun @Lin_Manuel ok jomny I know you're excited but 960/00 isn't a valid rating, 13/10
1
```

- There are only 4 records above, that have ratings information available.
- We will correct these ratings manually.

```
df_tweet.loc[df_tweet.tweet_id == 666287406224695296, ['rating_numerator', 'rating_denominator']] = 9, 10
df_tweet.loc[df_tweet.tweet_id == 682962037429899265, ['rating_numerator', 'rating_denominator']] = 10, 10
df_tweet.loc[df_tweet.tweet_id == 740373189193256964, ['rating_numerator', 'rating_denominator']] = 14, 10
df_tweet.loc[df_tweet.tweet_id == 832464395298406400, ['rating_numerator', 'rating_denominator']] = 13, 10
```

Quality Issue 5:

- There are 3 tweets left that do not have any rating information
- We will delete these entries.

```
#Define

# Tweets with tweet_id = 682808988178739200, 810984652412424192 and 832088576586297345
# We will drop them.
```

```
# Code:

df_tweet.drop(df_tweet.index[[342, 516, 1663]], inplace = True)
```

```
# Test

# Let us test if the code worked.

df_tweet.shape
```

```
(2117, 11)
```

The 3 entries with no rating are deleted. We only have 2117 instead of 2120 entries.

Quality Issue 6:

- A lot of the dog names are missing and need to be filled in wherever possible.
- Several other dog names like "a", "the", "an", and "this" are wrong and need to be corrected.

```
# Define:

# There are 644 records with no name. There are deveral others which are wrong.
# We need to try to extract the correct name from the tweet text.
# Once the correct names are extracted, the wrong names need to be replaced with the correct ones.

df_noname = df_tweet.query('name == "None" or name == "a" or name == "the" or name == "an" or name == "this"')
df_noname.shape
```

```
(705, 11)
```

```
# Code:

# There are 705 entries with no name, or a", "the", "an", and "this".
# We will split the text of the tweet to try and extract the name.

df_yesname = df_noname[df_noname['text'].str.contains("name", "named")]
df_yesname.drop(['timestamp','expanded_urls','rating_numerator','rating_denominator','doggo','floofer','pupper','puppo'],
                axis = 1, inplace = True)

df_yesname['name1'] = df_yesname['text'].str.split('named ', n=1, expand = True)[1].str.split(' ', n=1, expand = True)[0]
df_yesname['name2'] = df_yesname['text'].str.split('name is ', n=1, expand = True)[1].str.split(' ', n=1, expand = True)[0]
df_yesname['name_f'] = df_yesname['name1'].map(str) + df_yesname['name2'].map(str)
df_yesname['name_f'].replace('None', ' ', inplace = True)
df_yesname['name_f'].dropna()
df_yesname
```

We will now replace the missing name values in our original dataframe with the names in df_yesname, based on the tweet_id.

df_tweet.loc[df_tweet.tweet_id.isin(df_yesname.tweet_id), ['name']] = df_yesname[['name_f']]

Quality Issue 7:

- Some dog stages do not look right.
- Some are missing and need filling in.
- Some have multiple dog stage entries.
- We need to extract dog stage from the text of the tweet and aggregate values in a new column dog_stage

```
# Define

# We need to extract dog stage from the text of the tweet and aggregate values in a new column dog_stage
```

```
# Code

#Let us now change the values in the doggo, floffer, pupper and puppo columns if the tweet text has those values.


df_tweet['doggo'] = np.where(df_tweet['text'].str.contains('doggo|doggos', case=False, na=False), 'doggo', '')
```

```
df_tweet['floofer'] = np.where(df_tweet['text'].str.contains('floof|floofer|floofs', case=False, na=False), 'floofer', '')
```

```
df_tweet['pupper'] = np.where(df_tweet['text'].str.contains('pupper|puppers', case=False, na=False), 'pupper', '')
```

```
df_tweet['puppo'] = np.where(df_tweet['text'].str.contains('puppo|puppos', case=False, na=False), 'puppo', '')
```

Quality Issue 8:

- There should be a single column for dog stage after consolidating information in the doggo, pupper, puppo and floofer columns.
- As it stands right now, the doggo, pupper, puppo and floffer columns cannot be used for any analysis.

```
# Code

df_tweet['dog_stage'] = df_tweet['doggo'].map(str) + df_tweet['floofer'].map(str) +
df_tweet['pupper'].map(str) + df_tweet['puppo'].map(str)
```

- We were able to merge the values from 4 different columns into a single column.
- There are a few entries with more than one dog stage. It could be because there are different stages of dogs in those entries.
- There are still 1708 entries without any classification of dog stage.

Quality Issue 9:

10. The predictions from the neural network are not all accurate. It predicts dogs as laptops and toilet paper and desktop computers. While we are noting this down as an issue, we will not be fixing it for this project.

We will now fix the Tidyness issues with the data.

Tidyness Issue 1:

- Now that we have merged data from the doggo, floffer, pupper and puppo columns, we no longer need these columns.

```
# Define

# Drop columns doggo, floffer, pupper and puppo from the dataframe
```

```
# Code

df_tweet.drop(['doggo', 'floofer', 'pupper', 'puppo'], axis=1, inplace = True)
```

```
# Test

df_tweet.head(1)
```

Tidyness Issue 2:

- Dog breed Column needs to be added to the main dataframe.
- We need to update the column using data in the image predictions file, which we extracted to dataframe df_img_pred

```
# Define

# Dog breed Column needs to be added to the main dataframe.
# We need to update the column using data in the image predictions file, which w
```

```
# Code

df_tweet = pd.merge(df_tweet, df_img_pred, how='inner', on=['tweet_id'])
```

```
# Test

df_tweet.head(2)
```

Tidyness Issue 3:

- The most accurate breed prediction is in column "p1".
- We need to also rename column "p1" to something more meaningful like "predicted_breed"
- We do not need the other columns and can drop them.

```
# Define

# Drop all new columns merged, except "p1".
# Rename label "p1" to "predicted_breed".
```

```
# Code

df_tweet.drop(['jpg_url','img_num','p1_conf','p1_dog','p2','p2_conf','p2_dog','p3','p3_conf','p3_dog'], axis=1, inplace=True)
```

```
df_tweet.rename(columns={'p1':'predicted_breed'}, inplace=True)
```

```
# Test

df_tweet.sample(1)
```

Tidyness Issue 4:

- We need to merge the retweet_count and favorite_count fields into the main dataframe.
- We need to update the columns using data from the json file, which we extracted to dataframe df_retweet.

```
# Define

# Merge "retweet_count" and "favorite_count" from the df_retweet dataframe into the main dataframe.
```

```
### Code

df_tweet = pd.merge(df_tweet, df_retweet, how='inner', on=['tweet_id'])
```

```
# Test

df_tweet.head(2)
```

Looks like we were able to merge all entries except 1 (1936 instead of 1937).

Conclusions

- We cleaned dirty data, filled in missing data and deleted unwanted data from the main dataframe.
- We also merged useful data from other dataframes into the main dataframe.
- The dataframe now looks very clean and tidy.
- We are now ready to export this final dataframe to a .csv and/or .sql file.