Wrangling Report

This report summarises the wrangling and analysing of data, gathered from a variety of sources and in a variety of formats. The quality and tidiness was assessed, and then cleaned. Finally the wrangling results were show cased with insights and visualisations.

Data Gathering

There are 3 sources of data for this project:

1. The 'twitter_archive_enhanced.csv' was directly download and put into the Jupyter notebook local directory. WeRateDogs downloaded their Twitter archive and provided it to Udacity. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017

2. The Image Predictions File 'image_prediction.tsv' is the output from when Udacity ran every image in the WeRateDogs Twitter archive through a neural network that can classify breeds of dogs. The results were a table full of image predictions (the top three only) alongside each tweet ID, image URL, and the image number that corresponded to the most confident prediction (numbered 1 to 4 since tweets can have up to four images). This file had to be downloaded and written programmatically with python from

the Jupyter notebook.

3. The additional data 'tweet_json.txt' was gathered from Twitter's API. We have the WeRateDogs Twitter archive and specifically the tweet IDs within it, so can gather this data for all 5000+ tweets. The twiter api download was one complete json dumped tweet with all the fields from tweepy downloaded data into tweet_json.txt. The relevant fields needed to be parsed and saved to a data frame, without duplicating columns we already have in other tables:

   All 3 data sets have the column "tweet_id" which will be used as the primary key to match the data to a row in the twitter-archive-enhanced.csv and image_prediction.tsv.

   I needed to write a python script to parse the raw json data and just keep the following fields:

4. "tweet_id": needed to match the data to a row in the twitter-archive-enhanced.csv

5. "retweet_count"
6. "favorite_count":
7. "retweeted":
8. "display_text_range":

Assessing Data

With all 3 data sources downloaded,each needed to be analysed for quality and tidyness issues.

Quality Issues

Define Steps

1. The Datatype of the "tweet_id" column is integer and should be string.

2. The Datatype of the "timestamp" column is object and should be datetime.

3. Replace the value 'None' with NaN, the correct missing data value.

4. There are values some of the dog names are seem not to be correct such as:

- None
- 10
- a
- an
- by

1. Delete retweets As per project specification, we only want original dog ratings. So remove retweets (text column starts with RT @) as a user can retweet their on tweet.

2. Drop columns with high numbers of missing values from archive table:

- in_reply_to_status_id

- in_reply_to_user_id
- retweeted_status_id
- retweeted_status_user_id
- retweeted_status_timestamp'

1. The column "expanded URLs" has missing values and duplicates values.

- expanded_urls with null values = 59. Drop duplicates from archive table.

```
df_archive_clean.drop_duplicates(subset ="expanded_urls",
                    keep = False, inplace = True)
```

1. The wrong datatype in the 'img_num' Column is set.

```
df_predictions_clean['img_num'] =
df_predictions_clean['img_num'].astype(str)
```

Test the datatype has been changed.

2. There are 66 duplicates of the same image in the 'jpg_url' column. Drop duplicates in predictions

```
df_predictions_clean[df_predictions_clean['jpg_url'].duplicated()]
```

Test duplicates have been dropped.

Tidiness Issues

Define Steps

1. doggo, floofer, pupper, puppo these 4 variables should be combined into one categorical variable Dog Type.

2. Drop the 'retweeted' column from the df_tweepy_clean dataframe All the values say false, but this is untrue. We can see this in retweet_count that there were retweets.

3. Only keep the confidence variable with the highest confidence rate in the predictions table.

4. Merge the dataframe twitter_archive, dataframe image_predictions, and tweet_json dataframes.

Addressing the Issues Found in the Data

I cleaned and tidied each issue and tested each to verify the change. I wasn't sure whether to tidy first and then clean or visa versa. The first time I cleaned, I tidied each table then merged them all together into one data set and then tidied, but found the column formats changed and the data had become corrupted in ways, that prevented me from completing the tasks. I then cleaned each file , tidied and then merged and that worked out perfectly.

Quality Issues

1. I changed the data type of the 'tweet_id' column to string. Convert tweet_id in the 3 datasets, or the datasets won't merge later. the data is a string not a numerical value.

```
Test with print(<archive_name>['tweet_id'].dtypes)
```

2. The Datatype of the "timestamp" column is object and should be datetime. Change the string 'timestamp' column to a datetime format, which could be used to sort by time, or display time ranges later.

```
df_archive_clean['timestamp'] =
pd.to_datetime(df_archive_clean['timestamp'], format = "%Y-%m-%d
")
```

3. Replace the value 'None' with NaN, the corect missing data value. NaN is the correct missing data value to use in pandas. I tried to use numpy command like this: df_archive_clean = df_archive_clean.fillna(value=np.nan) This didn't work, but I found a way to convert 'None' to 'NaN' on a per column basis.

```
df_archive_clean.doggo.replace('None', 'NaN', inplace=True)
df_archive_clean.floofer.replace('None', 'NaN', inplace=True)
df_archive_clean.pupper.replace('None', 'NaN', inplace=True)
df_archive_clean.puppo.replace('None', 'NaN', inplace=True)
```

4. There are values some of the dog names are seem not to be correct such as:

- None
- 10
- a
- an
- by

I observed, that all of the invalid names are lower case,except None. I made a list of all of the lower case letters and convert them to 'NaN', by iterating each row of the 'name' column and comparing to the list values and converting the invalid names to 'NaN'.

```
df_archive_clean.name.replace('None', 'NaN', inplace=True)

for i in df_archive_clean[mask]:
    x = 0
    while x < len(invalid_names):
        if i == invalid_names[x]:
            i == 'NaN'
        x+=1
```

1. Delete retweets As per project specification, we only want original dog ratings. So remove retweets (text column starts with RT @) as a user can retweet their on tweet. To do this remove retweets is to select only rows that have null values in retweet related columns, using pandas isnull() function.

2. Drop columns with high numbers of missing values from archive table:

- in_reply_to_status_id
- in_reply_to_user_id
- retweeted_status_id
- retweeted_status_user_id

- retweeted_status_timestamp'

1. The column "expanded URLs" has missing values and duplicates values.

- expanded_urls with null values = 59. Drop duplicates from archive table. The in_reply_to_status_id column has 0.97 % data missing The in_reply_to_user_id column has 0.97 % data missing The retweeted_status_id column has 0.92 % data missing The retweeted_status_user_id column has 0.92 % data missing The retweeted_status_timestamp has 0.92 % data missing

```
df_archive_clean.drop(['in_reply_to_user_id','in_reply_to_status_id'
'retweeted_status_id','retweeted_status_user_id','retweeted_status_t
axis=1, inplace=True)
```

The other 2 tables have no missing data.

1. Drop columns without pictures

2. There are 66 duplicates of the same image in the 'jpg_url' column. Drop duplicates in predictions

Tidyness Issues

1. doggo, floofer, pupper, puppo these 4 columns should be combined into one column called Dog Type. This required some python enumeration of the cells and appending the values into a cell dog data, which represents all four cells. Having separate cells for this, is confusing and not good data presentation.

Create a list

```
dog_list =['doggo', 'floofer', 'pupper', 'puppo']
```

Check the values from dog_list match the 4 columns and if they do, add them to the new dog_type column, which will be appended to the dataframe and then the 4 columns can be dropped:

```
if x in df_archive_clean.loc[i,['doggo', 'floofer', 'pupper',
'puppo']].tolist():
                df_archive_clean.loc[i,'dog_type'].append(x)
```

1. A. Drop the 'retweeted' column from the df_tweepy_clean dataframe All the values say false, but this is untrue. We can see this in retweet_count that there were retweets. Counting the values of this column all rows have a flase value: ` df_tweepy_clean.retweeted.value_counts() False 2326

df_tweepy_clean.retweet_count.value_counts() The ouput shows there were large retweet counts.

```
3. Only keep the confidence variable with the highest
confidence rate in the predictions table.
```

```
Merge all of the neural network results into into 1 cell
called confidence.
```

drop columns

df_predictions_clean.drop(['p1_dog','p2','p2_conf','p2_dog','p3','p3_conf','p3_dog'], axis=1, inplace=True)

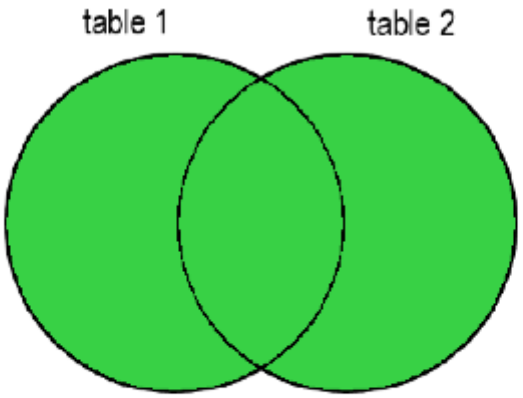
change column names

col_names = ['tweet_id', 'jpg_url', 'img_num', 'prediction', 'confidence']
df_predictions_clean.columns = col_names

```
Test that we have 1 column called 'confidence'.

4.  Merge the  dataframe twitter_archive, dataframe
image_predictions, and tweet_json dataframes.
Now that all of the cleaning and tidying tasks have been
completed, we can merge the 3 data sets into 1 with a full
outer join, using the pandas concat command:
```

df_copy = pd.concat([df_archive_clean, df_predictions_clean, df_tweepy_clean], join='outer', axis=1) `



Venn diagram representation for Full Outer Join