Reporting: wragle_report

- Create a **300-600 word written report** called "wrangle_report.pdf" or "wrangle_report.html" that briefly describes your wrangling efforts. This is to be framed as an internal document.

We Rate Dogs Tweet Analysis.

A look at dog rating tweets by `@weratedogs` on twitter.

Introduction

The data wrangling project was to gain insights into the possibility of patterns emergent in the we rate dogs twitter account's dog rating tweets.

Data Gathering

The data was gathered from 3 data sources, namely:

1. `Twitter_archive_enhanced.csv` a data archive tweets provided by Udacity. The data was first manually downloaded, then uploaded to the project workspace by using the manual upload functionality accessible under the jupyter homepage by clicking the `upload` button and selecting the archive.

2. `image_predictions.tsv` a data archive of dog breeds present in the tweet images, as inferred by a neural network. The data was also provided by Udacity and downloaded via the python `requests` library from: **https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv**. The data was downloaded by passing the url to `requests.get()` and writing the output to a data file by opening a file named `image_predictions.tsv` with write binary mode, and writing the response content to it.

3. `image_predictions.tsv` a data dump of json data downloaded from the Twitter API using the python `tweepy` library. The first step in this activity was opening a Twitter developer account, creating an app for this project activity, and then getting requisite API keys and tokens, and writing them to a json file named `settings.json` as json data, then uploading it to the project workspace; the next step was opening the json file in the notebook with read binary mode, and creating a dictionary by parsing the json data into a variable using the `Json` python package.

Next the settings were passed to the Oauth handler of tweepy to create an object named `auth`, and an access token was set on the object. A tweepy.API object was then created with `wait_on_rate_limit` and `wait_on_rate_limit_notify` set to True so a notification is made once the Twitter rate limit is reached and so the app continues to scrape once the timeout is over. A for loop was then made to traverse over all the tweet ids in the twitter archive dataset so the API object could download the tweets and save them into the file which was opened using a context processor (`with open(...):`) with write mode access. After each successful download a new line of json data would be written into a file, and failed access attempts would be saved into a fails dictionary which prints, together with the difference between start and end times at the end of the code's execution.

Data Assessment

The data was visually assessed by printing the datasets and scrolling to visually search for any and all anomalies in the dataframes.
The data was then assessed programmatically by utilising the following inbuilt methods:

- `info()` method to see a brief overview of the columns, their names, data types, and the total non-null entries in each column.
- `shape` member to see the number of rows and columns in the data frame.
- `describe()` method to view a statistical summary of the data in the columns

Upon assessment, the following issues were discovered:

Quality issues

1. twitter-archive-enhanced.csv data has missing values for reply tweets.
2. twitter-archive-enhanced.csv data has missing values for retweet tweets.
3. Tweet_json.txt data has missing values for geo.
4. Tweet_json.txt data has missing values for extended_entities.
5. Tweet_json.txt data has missing values for retweeted_status
6. twitter-archive-enhanced.csv data has missing values for expanded_urls
7. twitter-archive-enhanced.csv has a string for timestamp instead of a datetime.
8. tweet_id in all datasets should be a string not an int

Tidiness issues

1. arch_data needs a dog_genre variable to combine doggo, floofer, pupper, puppo binary columns.
2. id_str should be tweets_id in tweets_json.txt
3. The datasets need to be combined into one dataset.

Data Cleaning

The above discovered issues were then resolved by performing the following steps:

- Making a copy of all original datasets so as to have a reset point in case of mistakes during data cleaning
- Removing all columns where over 90% of the data was missing, This step resolved quality issues 1, 2, 3, 4, and 5.
- Removing all the data rows where data was missing as they only accounted for about 2.5% of all data. This resolved quality issue 6.
- Changing the data type of the `timestamp` column to `datetime`. This resolved quality issue 7.
- Changing the datatype of `tweet_id` in all datasets to `string`. This resolved quality issue 8.
- Combining the separate binary columns into a single column that lists the dog types separated by a comma where applicable, or as a single type where not. This resolved tidiness issue 1.
- Renaming `id_str` to `tweets_id` in `tweets_json.txt`. This resolved tidiness issue 2.
- Merging all 3 datasets on the `tweet_id` column with an inner join. This resolved tidiness issue 3.
- The data was checked to test the effectiveness of the data cleaning process at the end of each cleaning step.

Data Storage

After it was confirmed that the data was properly cleaned, it was then stored into a file named `twitter_archive_master.csv` for future reference.

Analysing and Visualising Data

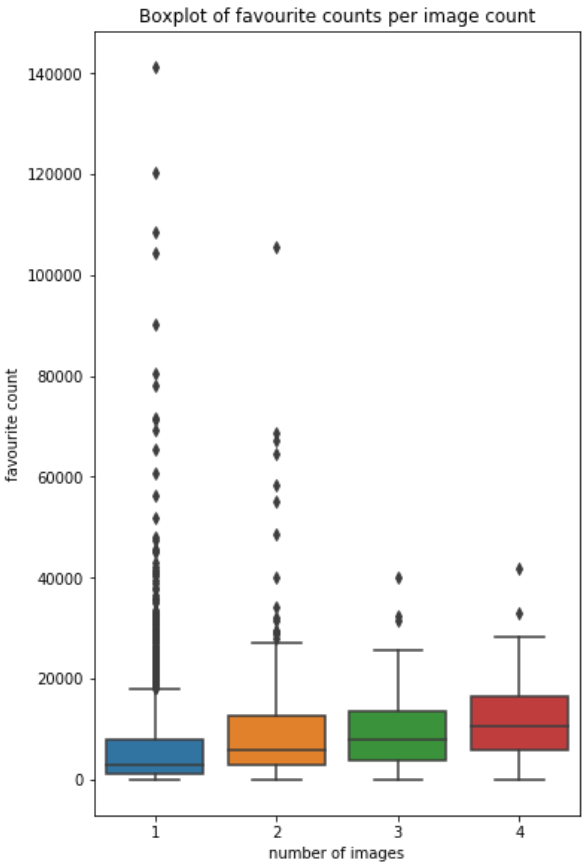
After the data was cleaned, it was analysed to gain insights and produce visualisations.

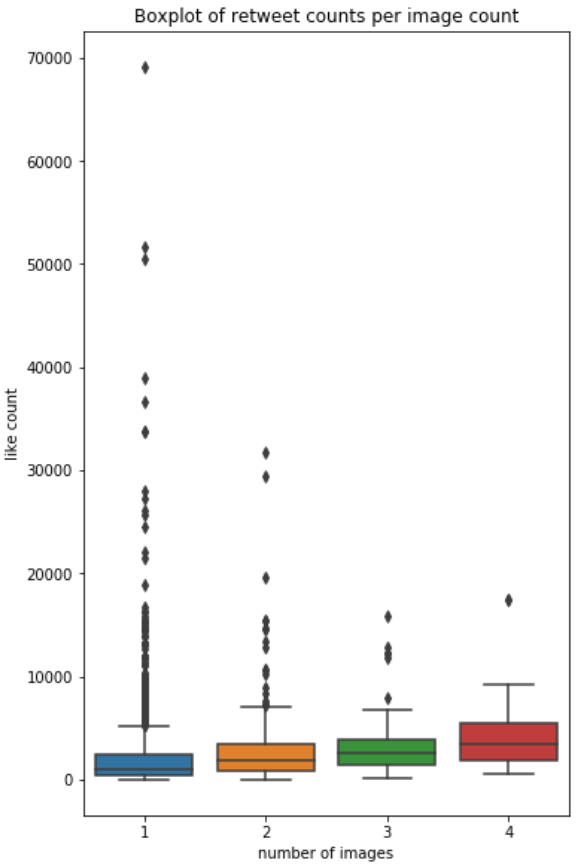
Insights:

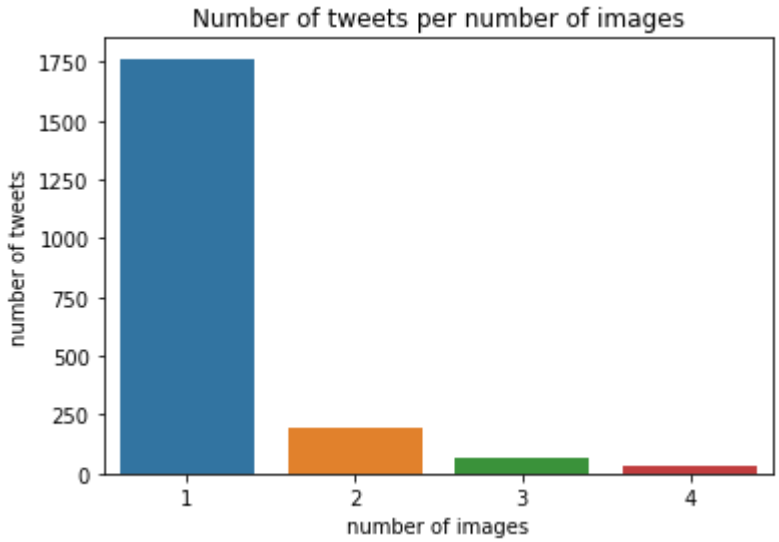
1. Most tweets only have one image attached to them.
2. Most images are predicted by the model to be golden retrievers.
3. Tweets with only one image have the highest like and retweet counts.
4. Golden Retrievers have the highest number of retweets and favourites.
5. Samoyed has more likes and retweets than chow and pug besides being in fewer tweet images.

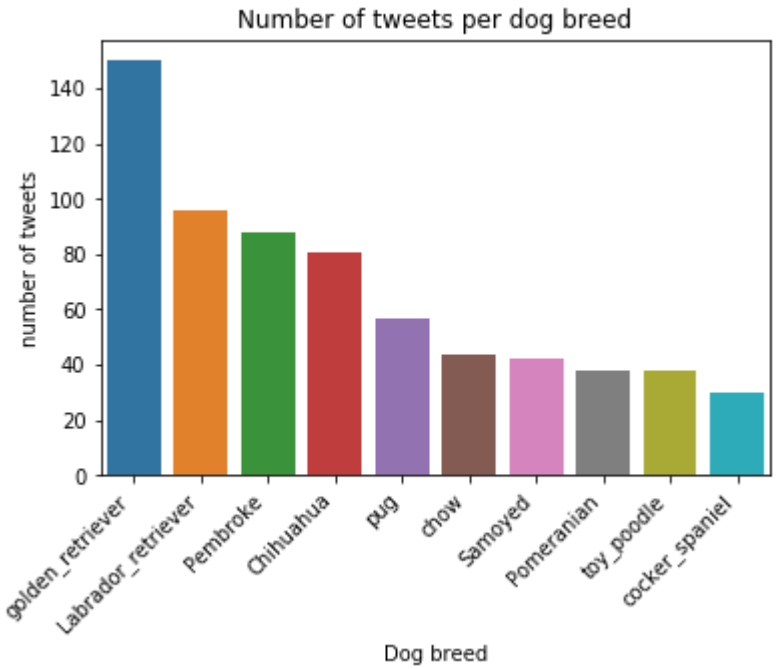
Visualisation

The following visualisations were produced:


Boxplot of favourite counts per image count
favourite count
number of images
0
20000
40000
60000
80000
100000
120000
140000
1
2
3
4


Boxplot of retweet counts per image count
like count
number of images
0
10000
20000
30000
40000
50000
60000
70000
1
2
3
4


Number of tweets per number of images
number of tweets
number of images
0
250
500
750
1000
1250
1500
1750
1
2
3
4


Number of tweets per dog breed
number of tweets
0
20
40
60
80
100
120
140
golden_retriever
Labrador_retriever
Pembroke
Chihuahua
pug
chow
Samoyed
Pomeranian
toy_poodle
cocker_spaniel
Dog breed