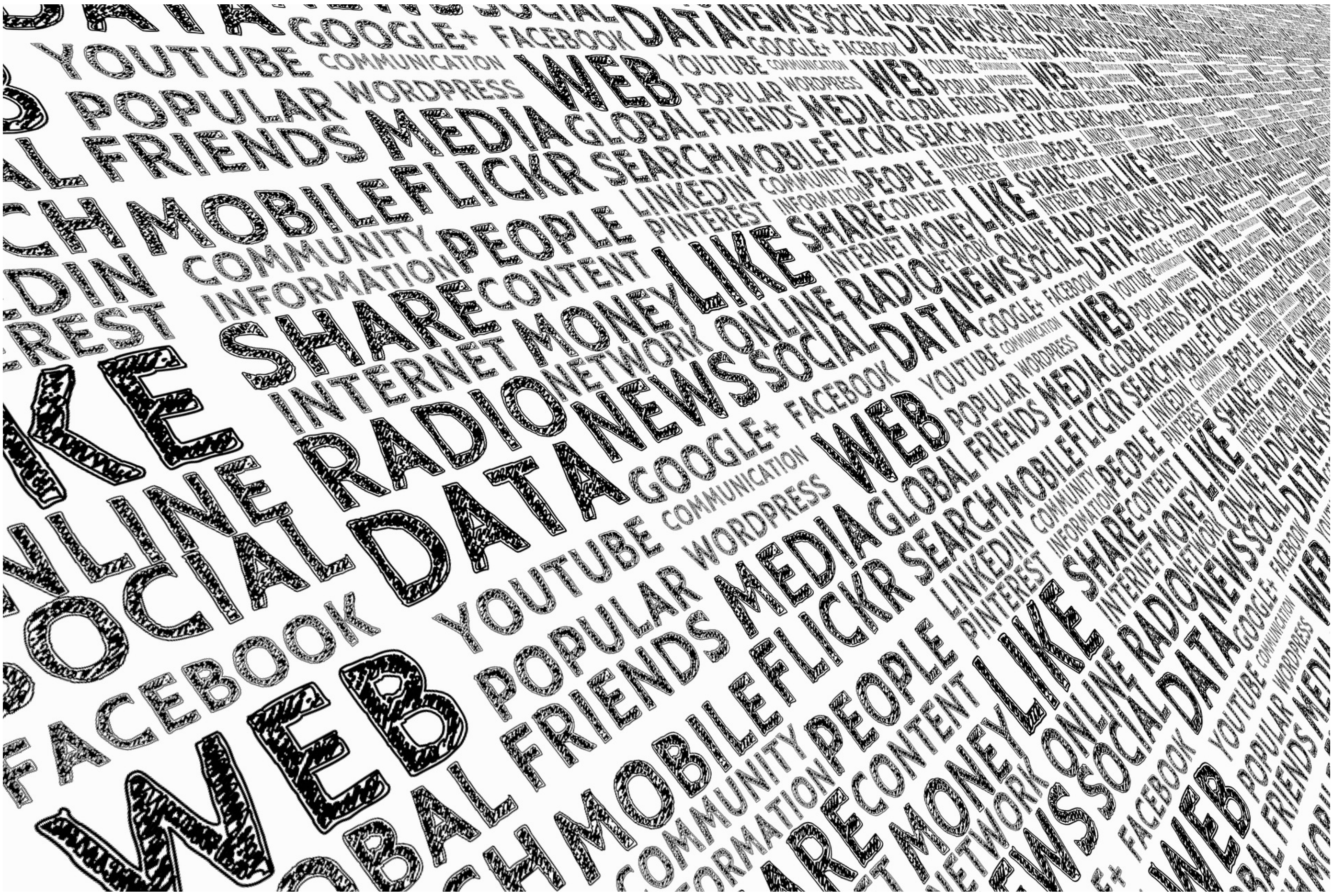

UDACITY Data Analysis Nanodegree

Project:- Wrangle & Analyze Data, Wrangle

Grant Patience, 26th August 2019

Table of Contents

Introduction

This project focused on wrangling data from the **WeRateDogs Twitter (https://twitter.com/dog_rates)** account using Python, documented in a Jupyter Notebook (wrangle_act.ipynb). This Twitter account rates dogs with humorous commentary. The rating denominator is usually 10, however, the numerators are usually greater than 10. They're Good Dogs Brent wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. WeRateDogs has over 4 million followers and has received international media coverage.

WeRateDogs downloaded their Twitter archive and sent it to Udacity via email exclusively for us to use in this project. This archive contains basic tweet data (tweet ID, timestamp, text, etc.) for all 5000+ of their tweets as they stood on August 1, 2017

---

1. Determine Objectives and Assess the Situation

For this project we will use the **CRISP-DM process (https://www.sv-europe.com/crisp-dm-methodology/)**. The first stage of the CRISP-DM process is to understand what you want to accomplish. The goal of this stage of the process is to uncover important factors that could influence the outcome of the project.

Project Details

Fully assessing and cleaning the entire dataset would require exceptional effort so only a subset of its issues (eight quality issues and two tidiness issues at minimum) needed to be assessed and cleaned.

The tasks for this project were:

- Data wrangling, which consists of:
  - Gathering data
  - Assessing data
  - Cleaning data
- Storing, analyzing, and visualizing our wrangled data
- Reporting on 1) our data wrangling efforts and 2) our data analyses and visualizations

Key Points

Key points to keep in mind when data wrangling for this project:

- We only want original ratings (no retweets) that have images. Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.

- Fully assessing and cleaning the entire dataset requires exceptional effort so only a subset of its issues (eight (8) quality issues and two (2) tidiness issues at minimum) need to be assessed and cleaned.
- Cleaning includes merging individual pieces of data according to the rules of tidy data.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned. This unique rating system is a big part of the popularity of WeRateDogs.
- We do not need to gather the tweets beyond August 1st, 2017. We can, but note that we won't be able to gather the image predictions for these tweets since we don't have access to the algorithm used.

1.1 Outline of Steps

- We state what resources are available to us and in this section we discuss what it is we wish to achieve,
- We decide which Questions we want to ask of the data
- We will Gather the Data that we need
- Import the data into Python to perform some initial Understanding of the data to help us understand the data, and Assess Data Quality and perform any resolve any Data Cleansing.
- Perform Exploratory Data Analysis where we will research the answers to our questions
- Create visualisations to aid exploration and research
- Draw our Conclusion based on the data and communicate our findings

1.2 What are the desired outputs of the project?

- Accurate project submission:

> - Ensure you meet specifications for all items in the Project Rubric. Your project "meets specifications" only if it meets specifications for all of the criteria.
> - Ensure you have not included your API keys, secrets, and tokens in your project files.
> - If you completed your project in the Project Workspace, ensure the following files are present in your workspace, then click "Submit Project" in the bottom righthand corner of the Project Workspace page:
> - wrangle_act.ipynb: code for gathering, assessing, cleaning, analyzing, and visualizing data
> - wrangle_report.pdf or wrangle_report.html: documentation for data wrangling steps: gather, assess, and clean
> - act_report.pdf or act_report.html: documentation of analysis and insights into final data
> - twitter_archive_enhanced.csv: file as given
> - image_predictions.tsv: file downloaded programmatically
> - tweet_json.txt: file constructed via API
> - twitter_archive_master.csv: combined and cleaned data
> - any additional files (e.g. files for additional pieces of gathered data or a database file for your stored clean data)

- Meet the Criteria of the Udacity Rubric:

Code Functionality

- The student's code is functional. All project code is contained in a Jupyter Notebook named wrangle_act.ipynb and runs without errors.
- The student's code is readable, i.e., uses good coding practices. The Jupyter Notebook has an intuitive, easy-to-follow logical structure. The code uses comments effectively and is interspersed with Jupyter Notebook Markdown cells. The steps of the data wrangling process (i.e. gather, assess, and clean) are clearly identified with comments or Markdown cells, as well.

Gathering Data

- The student is able to gather data from a variety of sources and file formats. Data is successfully gathered: -- From at least the three (3) different sources on the Project Details page. -- In at least the three (3) different file formats on the Project Details page. -- Each piece of data is imported into a separate pandas DataFrame at first.

Assessing Data

- The student is able to assess data visually and programmatically for quality and tidiness. Two types of assessment are used:

    > - Visual assessment: each piece of gathered data is displayed in the Jupyter Notebook for visual assessment purposes. Once displayed, data can additionally be assessed in an external application (e.g. Excel, text editor).
    > - Programmatic assessment: pandas' functions and/or methods are used to assess the data.

- The student is able to thoroughly assess a dataset. At least eight (8) data quality issues and two (2) tidiness issues are detected, and include the issues to clean to satisfy the Project Motivation. Each issue is documented in one to a few sentences each.

Cleaning Data

- The student uses the steps in the data cleaning process to guide their cleaning efforts. The define, code, and test steps of the cleaning process are clearly documented.
- The student is able to thoroughly clean a dataset programmatically. Copies of the original pieces of data are made prior to cleaning. All issues identified in the assess phase are successfully cleaned (if possible) using Python and pandas, and include the cleaning tasks required to satisfy the Project Motivation. A tidy master dataset (or datasets, if appropriate) with all pieces of gathered data is created.

Storing and Acting on Wrangled Data

- The student is able to store a gathered, assessed, and cleaned dataset. Students will save their gathered, assessed, and cleaned master dataset(s) to a CSV file or a SQLite database.
- The student is able to act on their wrangled data to produce insights (e.g. analyses, visualizations, and/or models). The master dataset is analyzed using pandas or SQL in the Jupyter Notebook and at least three (3) separate insights are produced. At least one (1) labeled visualization is produced in the Jupyter Notebook using Python's plotting libraries or in Tableau. Students must make it clear in their wrangling work that they assessed and cleaned (if necessary) the data upon which the analyses and visualizations are based.

Report

- The student is able to reflect upon and describe their data wrangling efforts. The student's wrangling efforts are briefly described. This document (wrangle_report.pdf or wrangle_report.html) is concise and approximately 300-600 words in length.
- The student is able to describe some insights found in their wrangled dataset. The three (3) or more insights the student found are communicated. At least one (1) visualization is included. This document (act_report.pdf or act_report.html) is at least 250 words in length.

Project Files

- Are all required files included in the student's submission?
- The following files (with identical filenames) are included:

  - wrangle_act.ipynb
  - wrangle_report.pdf or wrangle_report.html
  - act_report.pdf or act_report.html
  - All dataset files are included, including the stored master dataset(s), with filenames and extensions as specified on the Project Submission page.

1.3 What Resources are Available?

- UDACITY [Rubric (https://review.udacity.com/#!/rubrics/1136/view)](https://review.udacity.com/#!/rubrics/1136/view) for guidance on project submission
- Dataset supplied and gathered (Details in Section Data Description)
- Twitter API on Twitter's [Developer Portal (https://developer.twitter.com/en/docs/basics/developer-portal/overview)](https://developer.twitter.com/en/docs/basics/developer-portal/overview)
- Jupyter Python Notebook

1.4 What Questions Are We Trying To Answer?

2. Data Wrangling and Understanding

The second stage of the process is where we acquire the data listed in the project resources. Describe the methods used to acquire them and any problems encountered. We record problems you encountered and any resolutions achieved. Tis includes any data quality issues, and any resolution steps taken. This initial collection includes extraction details and source details, and subsequently loaded into Python and analysed in Jupyter notebook.

2.1 Data Description

Enhanced Twitter Archive

The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: each tweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) to make this Twitter archive "enhanced."

Additional Data via the Twitter API

Back to the basic-ness of Twitter archives: retweet count and favorite count are two of the notable column omissions. Fortunately, this additional data can be gathered by anyone from Twitter's API. Well, "anyone" who has access to data for the 3000 most recent tweets, at least. But we, because we have the WeRateDogs Twitter archive and specifically the tweet IDs within it, can gather this data for all 5000+. And guess what? We're going to query Twitter's API to gather this valuable data.

Image Predictions File

The tweet image predictions, i.e., what breed of dog (or other object, animal, etc.) is present in each tweet according to a neural network. This file (image_predictions.tsv) hosted on Udacity's servers and we downloaded it programmatically using python Requests library on the following (URL of the file: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv (https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv))

2.2 Data Gathering

Enhanced Twitter Archive

We manually downloaded this file manually by clicking the following link: **twitter_archive_enhanced.csv (https://d17h27t6h515a5.cloudfront.net/topher/2017/August/59a4e958_twitter-archive-enhanced/twitter-archive-enhanced.csv)**

Additional Data via the Twitter API

In this project, we'll be using Tweepy (http://www.tweepy.org/) to query Twitter's API for additional data beyond the data included in the WeRateDogs Twitter archive. This additional data will include retweet count and favorite count.

Some APIs are completely open, like MediaWiki (accessed via the wptools library). Others require authentication. The Twitter API is one that requires users to be authorized to use it. This means that before you can run your API querying code, we need to set up your own Twitter application. Here are the steps to do that on the Twitter site:

- First, you need to sign up for a Twitter account.
- Next, to set up a developer account, follow the directions on Twitter's Developer Portal, in the "How to Apply" section.
- You will be guided through the steps, and asked to describe in your own words what you are building
- Once you submit your application, you should soon receive an email from Twitter letting you know they have approved your new Twitter developer account. Follow the link in the email from Twitter to a page of directions to get started creating your app.
- If you are asked for an app name, it can be anything appropriate, and if you're asked for a Website URL, it can be anything in a standard URL format. You can do the same with other requested URLs, or perhaps leave them blank.
- If you're asked to explain how your app will be used, you could say something like "I'm creating this for a student Data Wrangling project with Udacity, where we need to query and analyze Twitter data from WeRateDogs."
- You should then be given a Success message, and a new developer page displayed to you where you can manage your app.
- You can then go to the Keys and Tokens tab on this page to find or generate the Consumer API keys, and the Access Token and Access Token Secret that you will need.

Image Predictions File

This file (image_predictions.tsv) hosted on Udacity's servers and we downloaded it programmatically using python Requests library on the following (URL of the file: https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv (https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv))

2.3 Describe Data's General Properties

In this section we describe the data that has been acquired including its format, its quantity (for example, the number of records and fields in each table), the identities of the fields and any other surface features which have been discovered. Evaluate whether the data acquired satisfies requirements.

In [793]:

```
# Import necessary libraries for initial data understanding, visualisations and exploratory
import numpy as np
import pandas as pd
import requests
import tweepy
import json
import time
import re

#For Visuals
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('darkgrid')
```

Enhanced Twitter Archive

In [794]:

```
# reads the data from the file - denotes as CSV, it has no header row, sets column headers

twitter_archive = pd.read_csv('./Data/twitter-archive-enhanced.csv')
```

```
Now let's take our first look at the data.
```

In [795]:

```
twitter_archive.head()
```

Out[795]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.c
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.c
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.c
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.c
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.c

In [796]:

```
twitter_archive.tail(3)
```

Out[796]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
2353	666033412701032449	NaN	NaN	2015-11-15 23:21:54 +0000	href="http://twitt
2354	666029285002620928	NaN	NaN	2015-11-15 23:05:30 +0000	href="http://twitt
2355	666020888022790149	NaN	NaN	2015-11-15 22:32:08 +0000	href="http://twitt

In [797]:

```
# Show me the shape of the data
twitter_archive.shape
```

Out[797]:

```
(2356, 17)
```

In [798]:

```
# Show me the complete column list
twitter_archive.columns
```

Out[798]:

```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestam
p',
       'source', 'text', 'retweeted_status_id', 'retweeted_status_user_id',
       'retweeted_status_timestamp', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'puppo'],
      dtype='object')
```

In [799]:

```
# Show me the info properties
twitter_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [800]:

```
#Show me the number of unique values
twitter_archive.nunique()
```

Out[800]:

```
tweet_id                      2356
in_reply_to_status_id           77
in_reply_to_user_id             31
timestamp                     2356
source                           4
text                          2356
retweeted_status_id            181
retweeted_status_user_id        25
retweeted_status_timestamp     181
expanded_urls                 2218
rating_numerator                40
rating_denominator              18
name                           957
doggo                            2
floofer                          2
pupper                           2
puppo                            2
dtype: int64
```

Looks sensible - we can see tweet_id, source, text, and other interesting features.

Image Predictions File

In [801]:

```
# Use requests to download tsv file programmatically
url="https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/im
response = requests.get(url)
with open('./Data/image_predictions.tsv', 'wb') as file:
    file.write(response.content)
image_predictions = pd.read_csv('./Data/image_predictions.tsv', sep='\t')
```

In [802]:

```
image_predictions.head()
```

Out[802]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_spring
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature

In [803]:

```
image_predictions.tail(3)
```

Out[803]:

	tweet_id	jpg_url	img_num	p
2072	891815181378084864	https://pbs.twimg.com/media/DGBdLU1WsAANxJ9.jpg	1	Chihuahu
2073	892177421306343426	https://pbs.twimg.com/media/DGGmoV4XsAAUL6n.jpg	1	Chihuahu
2074	892420643555336193	https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg	1	orang

In [804]:

```
image_predictions.shape
```

Out[804]:

```
(2075, 12)
```

In [805]:

```
image_predictions.columns
```

Out[805]:

```
Index(['tweet_id', 'jpg_url', 'img_num', 'p1', 'p1_conf', 'p1_dog', 'p2',
       'p2_conf', 'p2_dog', 'p3', 'p3_conf', 'p3_dog'],
      dtype='object')
```

In [806]:

```
image_predictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [807]:

```
image_predictions.nunique()
```

Out[807]:

```
tweet_id    2075
jpg_url     2009
img_num        4
p1           378
p1_conf     2006
p1_dog         2
p2           405
p2_conf     2004
p2_dog         2
p3           408
p3_conf     2006
p3_dog         2
dtype: int64
```

Looks sensible - we can see tweet_id, jpg url, and predictions.

Additional Data via the Twitter API

In [808]:

```
CONSUMER_KEY = ""
CONSUMER_SECRET = ""
OAUTH_TOKEN = ""
OAUTH_TOKEN_SECRET = ""
```

In [809]:

```
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
api = tweepy.API(auth)
```

In [810]:

```
# List of the errors
error_list = []
# List of the tweets
df_list = []
# Calculate the strat time of execution
start = time.time()
count = 0
fails_dict = {}
```

In [811]:

```
# For loop which will add each available tweet json to df_list
for tweet_id in df['tweet_id']:
    try:
        count += 1

        tweet = api.get_status(tweet_id, tweet_mode='extended',
                               wait_on_rate_limit = True, wait_on_rate_limit_notify = True)
        favorites = tweet['favorite_count'] # How many favorites the tweet had
        retweets = tweet['retweet_count'] # Count of the retweet
        date_time = tweet['created_at'] # The date and time of the creation

        df_list.append({'tweet_id': int(tweet_id),
                        'favorites': int(favorites),
                        'retweets': int(retweets),
                        'date_time': pd.to_datetime(date_time)})
    except Exception as e:
        fails_dict[tweet_id] = e
        print("Fail: " + str(tweet_id)+ " _ " + str(e))
        error_list.append(tweet_id)
# Calculate the duration of excution
end = time.time()
duration = end - start
print("Count: " + str(count) )
print("Duration: " + str(duration))
print(fails_dict)
```

```
Rate limit reached. Sleeping for: 630
Rate limit reached. Sleeping for: 654

Count: 1978
Duration: 1858.5161831378937
{}
```

In [812]:

```
# Number of results
print("Number of results: ", len(df_list))
# The tweet_id of the errors
print("Number of errors: ", len(error_list))
```

```
Number of results:  1978
Number of errors:  0
```

In [813]:

```
# Create DataFrames from list of dictionaries
json_tweets = pd.DataFrame(df_list, columns = ['tweet_id', 'favorites', 'retweets',
                                               'user_followers', 'user_favourites', 'date_t
# Save the dataFrame in file
json_tweets.to_csv('tweet_json.txt', encoding = 'utf-8', index=False)
```

In [814]:

```
# Read the saved tweet_json.txt file into a dataframe
tweet_data = pd.read_csv('tweet_json.txt', encoding = 'utf-8')
```

In [815]:

```
tweet_data.head()
```

Out[815]:

	tweet_id	favorites	retweets	user_followers	user_favourites	date_time
0	892420643555336193	37189	7957	NaN	NaN	2017-08-01 16:23:56+00:00
1	892177421306343426	31978	5907	NaN	NaN	2017-08-01 00:17:27+00:00
2	891815181378084864	24085	3904	NaN	NaN	2017-07-31 00:18:03+00:00
3	891689557279858688	40503	8110	NaN	NaN	2017-07-30 15:58:51+00:00
4	891327558926688256	38728	8790	NaN	NaN	2017-07-29 16:00:24+00:00

In [816]:

```
tweet_data.tail(3)
```

Out[816]:

	tweet_id	favorites	retweets	user_followers	user_favourites	date_time
1975	666033412701032449	120	43	NaN	NaN	2015-11-15 23:21:54+00:00
1976	666029285002620928	124	45	NaN	NaN	2015-11-15 23:05:30+00:00
1977	666020888022790149	2479	481	NaN	NaN	2015-11-15 22:32:08+00:00

In [817]:

```
tweet_data.shape
```

Out[817]:

```
(1978, 6)
```

In [818]:

```
tweet_data.columns
```

Out[818]:

```
Index(['tweet_id', 'favorites', 'retweets', 'user_followers',
       'user_favourites', 'date_time'],
      dtype='object')
```

In [819]:

```
tweet_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1978 entries, 0 to 1977
Data columns (total 6 columns):
tweet_id           1978 non-null int64
favorites          1978 non-null int64
retweets           1978 non-null int64
user_followers     0 non-null float64
user_favourites    0 non-null float64
date_time          1978 non-null object
dtypes: float64(2), int64(3), object(1)
memory usage: 92.8+ KB
```

In [820]:

```
tweet_data.nunique()
```

Out[820]:

```
tweet_id           1978
favorites          1828
retweets           1575
user_followers        0
user_favourites       0
date_time          1978
dtype: int64
```

2.4 Assess Data Quality

Examine the quality of the data, addressing questions such as:

- Is the data complete (does it cover all the cases required)?
- Is it correct, or does it contain errors and, if there are errors, how common are they?
- Are there missing values in the data? If so, how are they represented, where do they occur, and how common are they?

In [821]:

```
twitter_archive.sample(10)
```

Out[821]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
1044	743609206067040256	NaN	NaN	2016-06-17 01:00:24 +0000	href="http://twitt
765	777885040357281792	NaN	NaN	2016-09-19 15:00:20 +0000	href="http://twitt
1908	674436901579923456	NaN	NaN	2015-12-09 03:54:22 +0000	href="http://twitt
1275	709179584944730112	NaN	NaN	2016-03-14 00:49:23 +0000	
2299	667065535570550784	NaN	NaN	2015-11-18 19:43:11 +0000	href="http://twitt
1063	741067306818797568	NaN	NaN	2016-06-10 00:39:48 +0000	href="http://twitt
712	784431430411685888	NaN	NaN	2016-10-07 16:33:21 +0000	href="http://twitt
87	875144289856114688	NaN	NaN	2017-06-15 00:13:52 +0000	href="http://twitt
911	757597904299253760	NaN	NaN	2016-07-25 15:26:30 +0000	href="http://twitt
739	780601303617732608	NaN	NaN	2016-09-27 02:53:48 +0000	href="http://twitt

In [822]:

```
twitter_archive.info()
twitter_archive.nunique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

Out[822]:

```
tweet_id                      2356
in_reply_to_status_id           77
in_reply_to_user_id             31
timestamp                     2356
source                           4
text                          2356
retweeted_status_id            181
retweeted_status_user_id        25
retweeted_status_timestamp     181
expanded_urls                 2218
rating_numerator                40
rating_denominator              18
name                           957
doggo                            2
floofer                          2
pupper                           2
puppo                            2
dtype: int64
```

In [823]:

```
twitter_archive.describe()
```

Out[823]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweeted_s
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	
std	6.856705e+16	7.582492e+16	1.252797e+17	6.236928e+16	
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	

In [824]:

```
image_predictions.sample(10)
```

Out[824]:

	tweet_id	jpg_url	img_num	
2011	879008229531029506	https://pbs.twimg.com/media/DDLdUrqXYAMOVzY.jpg	1	
212	670037189829525505	https://pbs.twimg.com/media/CUxzQ-nWIAAgJUm.jpg	1	
676	683462770029932544	https://pbs.twimg.com/media/CXwlw9MWsAAc-JB.jpg	1	
541	677187300187611136	https://pbs.twimg.com/media/CWXaQMBWcAAATDi.jpg	1	
349	672482722825261057	https://pbs.twimg.com/media/CVUjd14W4AE8tvO.jpg	1	West_Hi
166	668981893510119424	https://pbs.twimg.com/media/CUize-0WEAAerAK.jpg	1	
1100	720775346191278080	https://pbs.twimg.com/media/CgC1WqMW4AI1_N0.jpg	1	
43	666776908487630848	https://pbs.twimg.com/media/CUDeDoWUYAAD-EM.jpg	1	
608	680070545539371008	https://pbs.twimg.com/media/CW-dU34WQAANBGy.jpg	1	
72	667211855547486208	https://pbs.twimg.com/media/CUJppKJWoAA75NP.jpg	1	

In [825]:

```
image_predictions.info()
image_predictions.nunique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

Out[825]:

```
tweet_id    2075
jpg_url     2009
img_num        4
p1           378
p1_conf     2006
p1_dog         2
p2           405
p2_conf     2004
p2_dog         2
p3           408
p3_conf     2006
p3_dog         2
dtype: int64
```

In [826]:

```
image_predictions.describe()
```

Out[826]:

	tweet_id	img_num	p1_conf	p2_conf	p3_conf
count	2.075000e+03	2075.000000	2075.000000	2.075000e+03	2.075000e+03
mean	7.384514e+17	1.203855	0.594548	1.345886e-01	6.032417e-02
std	6.785203e+16	0.561875	0.271174	1.006657e-01	5.090593e-02
min	6.660209e+17	1.000000	0.044333	1.011300e-08	1.740170e-10
25%	6.764835e+17	1.000000	0.364412	5.388625e-02	1.622240e-02
50%	7.119988e+17	1.000000	0.588230	1.181810e-01	4.944380e-02
75%	7.932034e+17	1.000000	0.843855	1.955655e-01	9.180755e-02
max	8.924206e+17	4.000000	1.000000	4.880140e-01	2.734190e-01

In [827]:

```
tweet_data.sample(10)
```

Out[827]:

	tweet_id	favorites	retweets	user_followers	user_favourites	date_time
277	828011680017821696	10700	2235	NaN	NaN	2017-02-04 22:45:42+00:00
1019	706265994973601792	2796	952	NaN	NaN	2016-03-05 23:51:49+00:00
1370	680130881361686529	2334	970	NaN	NaN	2015-12-24 21:00:12+00:00
78	874012996292530176	33283	9776	NaN	NaN	2017-06-11 21:18:31+00:00
534	780459368902959104	5513	1126	NaN	NaN	2016-09-26 17:29:48+00:00
915	716285507865542656	2830	1092	NaN	NaN	2016-04-02 15:25:47+00:00
617	765371061932261376	7344	2252	NaN	NaN	2016-08-16 02:14:15+00:00
315	821107785811234820	9959	2232	NaN	NaN	2017-01-16 21:32:06+00:00
1514	675015141583413248	2713	1190	NaN	NaN	2015-12-10 18:12:05+00:00
1464	676430933382295552	1410	347	NaN	NaN	2015-12-14 15:57:56+00:00

In [828]:

```
tweet_data.info()
tweet_data.nunique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1978 entries, 0 to 1977
Data columns (total 6 columns):
tweet_id           1978 non-null int64
favorites          1978 non-null int64
retweets           1978 non-null int64
user_followers     0 non-null float64
user_favourites    0 non-null float64
date_time          1978 non-null object
dtypes: float64(2), int64(3), object(1)
memory usage: 92.8+ KB
```

Out[828]:

```
tweet_id           1978
favorites          1828
retweets           1575
user_followers        0
user_favourites       0
date_time          1978
dtype: int64
```

In [829]:

```
tweet_data.describe()
```

Out[829]:

	tweet_id	favorites	retweets	user_followers	user_favourites
count	1.978000e+03	1978.000000	1978.000000	0.0	0.0
mean	7.356678e+17	8491.629929	2535.257331	NaN	NaN
std	6.745683e+16	12506.314781	4543.242986	NaN	NaN
min	6.660209e+17	75.000000	11.000000	NaN	NaN
25%	6.758041e+17	1824.000000	567.250000	NaN	NaN
50%	7.082494e+17	3807.000000	1218.000000	NaN	NaN
75%	7.876377e+17	10572.250000	2878.000000	NaN	NaN
max	8.924206e+17	160671.000000	80396.000000	NaN	NaN

2.4.1. Quality

Missing Data

In addition to incorrect datatypes, another common problem when dealing with real-world data is missing values. These can arise for many reasons and have to be either filled in or removed before we train a machine learning model. First, let’s get a sense of how many missing values are in each column

While we always want to be careful about removing information, if a column has a high percentage of missing values, then it probably will not be useful to our model. The threshold for removing columns should depend on the problem

In [830]:

```
# Function that will take an input table with aggregated values to columns, and then create
# two columns - the values and the percentage of total values in that column
def missing_values_table(df):
        mis_val = df.isnull().sum()
        mis_val_percent = 100 * df.isnull().sum() / len(df)
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")
        return mis_val_table_ren_columns
```

In [831]:

```
missing_values_table(twitter_archive)
```

```
Your selected dataframe has 17 columns.
There are 6 columns that have missing values.
```

Out[831]:

	Missing Values	% of Total Values
in_reply_to_status_id	2278	96.7
in_reply_to_user_id	2278	96.7
retweeted_status_id	2175	92.3
retweeted_status_user_id	2175	92.3
retweeted_status_timestamp	2175	92.3
expanded_urls	59	2.5

In [832]:

```
missing_values_table(image_predictions)
```

```
Your selected dataframe has 12 columns.
There are 0 columns that have missing values.
```

Out[832]:

Missing Values	% of Total Values

In [833]:

```
missing_values_table(tweet_data)
```

```
Your selected dataframe has 6 columns.
There are 2 columns that have missing values.
```

Out[833]:

	Missing Values	% of Total Values
user_followers	1978	100.0
user_favourites	1978	100.0

In [834]:

```
twitter_archive.retweeted_status_id.isnull()
```

Out[834]:

```
0        True
1        True
2        True
3        True
4        True
5        True
6        True
7        True
8        True
9        True
10       True
11       True
12       True
13       True
14       True
15       True
16       True
17       True
18       True
19      False
20       True
21       True
22       True
23       True
24       True
25       True
26       True
27       True
28       True
29       True
        ...
2326     True
2327     True
2328     True
2329     True
2330     True
2331     True
2332     True
2333     True
2334     True
2335     True
2336     True
2337     True
2338     True
2339     True
2340     True
2341     True
2342     True
2343     True
2344     True
2345     True
2346     True
2347     True
2348     True
2349     True
```

```
2350     True
2351     True
2352     True
2353     True
2354     True
2355     True
Name: retweeted_status_id, Length: 2356, dtype: bool
```

In [835]:

```
#Highlighting that not all tweets are original, some are retweets
twitter_archive[twitter_archive['retweeted_status_id'].isnull()]
```

2344	666071193221509120	NaN	NaN	2015-11-16 01:52:02 +0000	href="http://twitter.com/downloa
2345	666063827256086533	NaN	NaN	2015-11-16 01:22:45 +0000	href="http://twitter.com/downloa
2346	666058600524156928	NaN	NaN	2015-11-16 01:01:59 +0000	href="http://twitter.com/downloa
2347	666057090499244032	NaN	NaN	2015-11-16 00:55:59 +0000	href="http://twitter.com/downloa
2348	666055525042405380	NaN	NaN	2015-11-16 00:49:46 +0000	href="http://twitter.com/downloa
2349	666051853826850816	NaN	NaN	2015-11-16 00:35:11	href="http://twitter.com/downloa

Options

- We may want to remove null rows entirely from the dataset. To do so we would run something like the following

```
df.dropna()
```

- We may want to drop the columns if they appear to be predominantly NA. To do so we would run something like the following

```
# Get the columns with > 50% missing
missing_df = missing_values_table(df);
missing_columns = list(missing_df[missing_df['% of Total Values'] > 50].index)
print('We will remove %d columns.' % len(missing_columns))
df = df.drop(list(missing_columns))
```

- We may want to fill the missing values with the mean values from the dataset. To do so we would run something like the following

```
mean = df['x'].mean()
df['x'].fillna(mean, inplace=True)
```

Duplicates

There may be duplicates in the data. However, these may be legitimate new rows depending on the structure of the data. We need to discover them, then decide what to do with them

In [836]:

```
sum(twitter_archive.tweet_id.duplicated())
```

Out[836]:

```
0
```

In [837]:

```
sum(image_predictions.tweet_id.duplicated())
```

Out[837]:

```
0
```

In [838]:

```
sum(image_predictions.jpg_url.duplicated())
```

Out[838]:

```
66
```

In [839]:

```
sum(tweet_data.tweet_id.duplicated())
```

Out[839]:

```
0
```

In [840]:

```
tweet_data.tweet_id.value_counts()
```

Out[840]:

```
685532292383666176    1
667160273090932737    1
743545585370791937    1
671163268581498880    1
770655142660169732    1
762316489655476224    1
826598365270007810    1
687109925361856513    1
774314403806253056    1
814530161257443328    1
751937170840121344    1
750071704093859840    1
821886076407029760    1
891689557279858688    1
679527802031484928    1
703382836347330562    1
732585889486888962    1
734776360183431168    1
746131877086527488    1
683773439333797890    1
877316821321428993    1
677557565589463040    1
668268907921326080    1
785872687017132033    1
673355879178194945    1
669359674819481600    1
702598099714314240    1
666776908487630848    1
675798442703122432    1
680934982542561280    1
                     ..
736010884653420544    1
666099513787052032    1
843604394117681152    1
673342308415348736    1
794926597468000259    1
666817836334096384    1
748307329658011649    1
885984800019947520    1
773922284943896577    1
767500508068192258    1
837482249356513284    1
674291837063053312    1
693109034023534592    1
713175907180089344    1
670338931251150849    1
708109389455101952    1
828770345708580865    1
675898130735476737    1
791672322847637504    1
770069151037685760    1
685169283572338688    1
672538107540070400    1
760252756032651264    1
676496375194980353    1
```

```
692417313023332352    1
667119796878725120    1
688828561667567616    1
834931633769889797    1
836989968035819520    1
700151421916807169    1
Name: tweet_id, Length: 1978, dtype: int64
```

In [841]:

```
tweet_data[tweet_data.tweet_id == 853760880890318849]
```

Out[841]:

	tweet_id	favorites	retweets	user_followers	user_favourites	date_time
154	853760880890318849	28599	5741	NaN	NaN	2017-04-17 00:03:50+00:00

Options We may want to remove duplicate rows entirely from the dataset. To do so we would run the following

```
df.drop_duplicates(inplace=True)
```

Outliers

At this point, we may also want to remove outliers. These can be due to typos in data entry, mistakes in units, or they could be legitimate but extreme values. For this project, we will remove anomalies based on the definition of extreme outliers:

In [842]:

```
twitter_archive['name'].value_counts()
```

Out[842]:

```
None           745
a               55
Charlie         12
Lucy            11
Cooper          11
Oliver          11
Penny           10
Lola            10
Tucker          10
Bo               9
Winston          9
Sadie            8
the              8
Bailey           7
Toby             7
Daisy            7
Buddy            7
an               7
Dave             6
Oscar            6
Koda             6
Bella            6
Scout            6
Rusty            6
Jax              6
Milo             6
Leo              6
Jack             6
Stanley          6
Louis            5
              ...
Brat             1
Tuck             1
Comet            1
Einstein         1
Naphaniel        1
Bluebert         1
Howie            1
Stark            1
Jarvis           1
Lucia            1
Miguel           1
Swagger          1
Striker          1
Grizz            1
Hanz             1
Dallas           1
Ridley           1
Tug              1
Willy            1
Rolf             1
Ralphus          1
Cleopatricia     1
Maya             1
O                1
```

```
Bradley         1
Bobble          1
Jazz            1
Butters         1
Fynn            1
Arnold          1
Name: name, Length: 957, dtype: int64
```

In [843]:

```
# View rows where the value of 'name' is lowercase may not be an actual name
twitter_archive.loc[(twitter_archive['name'].str.islower())]
```

988	748977405889503236	NaN	NaN	20:31:43 +0000	href="http://twitter.com/downloa
992	748692773788876800	NaN	NaN	2016-07-01 01:40:41 +0000	href="http://twitter.com/downloa
993	748575535303884801	NaN	NaN	2016-06-30 17:54:50 +0000	href="http://twitter.com/downloa
1002	747885874273214464	NaN	NaN	2016-06-28 20:14:22 +0000	href="http://twitter.com/downloa
1004	747816857231626240	NaN	NaN	2016-06-28 15:40:07 +0000	href="http://twitter.com/downloa

In [844]:

```
#unusual names
twitter_archive[twitter_archive['name'].apply(len) < 3]
```

NaN	NaN	https://twitter.com/dog_rates/status/747885874...	8
NaN	NaN	https://twitter.com/dog_rates/status/747816857...	4
NaN	NaN	https://twitter.com/dog_rates/status/746872823...	11
NaN	NaN	https://twitter.com/dog_rates/status/746369468...	9

In [845]:

```
# View rows in twitter_archive which contain '&amp;' instead of '&' in 'text' column
twitter_archive[twitter_archive.text.str.contains('&amp;')]
```

NaN	NaN	https://twitter.com/dog_rates/status/750026558...	10
NaN	NaN	https://twitter.com/dog_rates/status/735137028...	9
NaN	NaN	https://twitter.com/dog_rates/status/719367763...	11

In [846]:

```
twitter_archive.source.value_counts()
```

Out[846]:

```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPho
ne</a>     2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck" rel="nofollow">TweetD
eck</a>      11
Name: source, dtype: int64
```

In [847]:

```
twitter_archive.rating_numerator.value_counts()
```

Out[847]:

```
12      558
11      464
10      461
13      351
9       158
8       102
7        55
14       54
5        37
6        32
3        19
4        17
1         9
2         9
420       2
0         2
15        2
75        2
80        1
20        1
24        1
26        1
44        1
50        1
60        1
165       1
84        1
88        1
144       1
182       1
143       1
666       1
960       1
1776      1
17        1
27        1
45        1
99        1
121       1
204       1
Name: rating_numerator, dtype: int64
```

In [848]:

```
twitter_archive.rating_denominator.value_counts()
```

Out[848]:

```
10     2333
11        3
50        3
80        2
20        2
2         1
16        1
40        1
70        1
15        1
90        1
110       1
120       1
130       1
150       1
170       1
7         1
0         1
Name: rating_denominator, dtype: int64
```

In [849]:

```
twitter_archive[twitter_archive['rating_numerator'].isnull()]
```

Out[849]:

tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_status

In [850]:

```
twitter_archive[twitter_archive['rating_denominator'].isnull()]
```

Out[850]:

tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_status

In [851]:

```
sum(twitter_archive['expanded_urls'].isnull())
```

Out[851]:

```
59
```

2.4.2. Tidiness

Investigate the dog "Stages" Columns

In [852]:

```
twitter_archive['doggo'].value_counts()
```

Out[852]:

```
None     2259
doggo      97
Name: doggo, dtype: int64
```

In [853]:

```
twitter_archive['floofer'].value_counts()
```

Out[853]:

```
None       2346
floofer      10
Name: floofer, dtype: int64
```

In [854]:

```
twitter_archive['pupper'].value_counts()
```

Out[854]:

```
None      2099
pupper     257
Name: pupper, dtype: int64
```

In [855]:

```
twitter_archive['puppo'].value_counts()
```

Out[855]:

```
None     2326
puppo     30
Name: puppo, dtype: int64
```

Data Summary Report

Category	Data set	Issue	Resolved
Quality	twitter_archive	Some tweets do not have images (expanded_urls)	Y
Quality	twitter_archive	Name column has invalid names e.g. 'O', 'a', 'an' and others less than 3 characters	Y
Quality	twitter_archive	Name column missing values showing as 'None' instead of NaN	Y
Quality	twitter_archive	Dog "Stage" columns missing values	Y
Quality	twitter_archive	Contains retweets	Y
Quality	twitter_archive	retweeted_status_timestamp, timestamp should be datetime instead of object (string).	Y
Quality	twitter_archive	in_reply_to_status_id, in_reply_to_user_id, retweeted_status_id, retweeted_status_user_id should be integers/strings instead of float.	Y
Quality	tweet_data	Duplicate values in tweet_id column	Y
Quality	image_predictions	The values p1, p2 and p3 are not very meaningful	Y
Quality	twitter_archive	Source column contains extraneous HTML content	Y

Category	Data set	Issue	Resolved
Quality	twitter_archive	Remove any all extreme values from the numertor and denuminator columns	Y
Tidiness	twitter_archive	Dog "stage" variable in uneccesary columns: doggo, floofer, pupper, puppo. It is better to create one column to contains the values.	Y
Tidiness	twitter_archive	Data unnecessarily split over three datasets Join 'tweet_info' and 'image_predictions' to 'twitter_archive'	Y
Tidiness	twitter_archive	Drop unnecessary columns 'retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp' since we no longer car about Retweets	Y

Data Cleansing

Cleasning our data is the final step in data wrangling. We will fix the quality and tidiness issues that we identified in the assess step.

In [856]:

```
#copy dataframes - always keep the originals intact
twitter_archive_clean = twitter_archive.copy()
image_predictions_clean= image_predictions.copy()
tweet_data_clean = tweet_data.copy()
```

In [857]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [858]:

```
image_predictions_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [859]:

```
tweet_data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1978 entries, 0 to 1977
Data columns (total 6 columns):
tweet_id           1978 non-null int64
favorites          1978 non-null int64
retweets           1978 non-null int64
user_followers     0 non-null float64
user_favourites    0 non-null float64
date_time          1978 non-null object
dtypes: float64(2), int64(3), object(1)
memory usage: 92.8+ KB
```

Define

Correct erroneous, mistaken, or incorrect dog names.

Code

In [860]:

```
twitter_archive_clean.name.value_counts()
```

Out[860]:

```
None           745
a               55
Charlie         12
Lucy            11
Cooper          11
Oliver          11
Penny           10
Lola            10
Tucker          10
Bo               9
Winston          9
Sadie            8
the              8
Bailey           7
Toby             7
Daisy            7
Buddy            7
an               7
Dave             6
Oscar            6
Koda             6
Bella            6
Scout            6
Rusty            6
Jax              6
Milo             6
Leo              6
Jack             6
Stanley          6
Louis            5
              ... 
Brat             1
Tuck             1
Comet            1
Einstein         1
Naphaniel        1
Bluebert         1
Howie            1
Stark            1
Jarvis           1
Lucia            1
Miguel           1
Swagger          1
Striker          1
Grizz            1
Hanz             1
Dallas           1
Ridley           1
Tug              1
Willy            1
Rolf             1
Ralphus          1
Cleopatricia     1
Maya             1
O                1
```

```
Bradley        1
Bobble         1
Jazz           1
Butters        1
Fynn           1
Arnold         1
Name: name, Length: 957, dtype: int64
```

In [861]:

```
# Save locations to a list where the 'name' column is either
# a. lowercase and 'text' column contains 'named'
# b.  lowercase and 'text' column contains 'name is'
# c.  all lowercase
a_replace = twitter_archive_clean.loc[(twitter_archive_clean['name'].str.islower()) & (twit
b_replace = twitter_archive_clean.loc[(twitter_archive_clean['name'].str.islower()) & (twit
c_replace = twitter_archive_clean.loc[(twitter_archive_clean['name'].str.islower())]

# Save these locations as lists
a_replace_list = a_replace['text'].tolist()
b_replace_list = b_replace['text'].tolist()
c_replace_list = c_replace['text'].tolist()

# Iterate through locations and set the 'name' to be the word that appears after 'named'
for entry in a_replace_list:
    mask = twitter_archive_clean.text == entry
    name_column = 'name'
    twitter_archive_clean.loc[mask, name_column] = re.findall(r"named\s(\w+)", entry)

# Iterate through locations and set the 'name' value to be the word that appears after 'nam
for entry in b_replace_list:
    mask = twitter_archive_clean.text == entry
    name_column = 'name'
    twitter_archive_clean.loc[mask, name_column] = re.findall(r"name is\s(\w+)", entry)

# Iterate through locations replace the name value with the word "None"
for entry in c_replace_list:
    mask = twitter_archive_clean.text == entry
    name_column = 'name'
    twitter_archive_clean.loc[mask, name_column] = "None"
```

In [862]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

Test

In [863]:

```
twitter_archive_clean.name.value_counts()
```

Out[863]:

```
None           854
Charlie         12
Oliver          11
Cooper          11
Lucy            11
Penny           10
Lola            10
Tucker          10
Winston          9
Bo               9
Sadie            8
Daisy            7
Bailey           7
Toby             7
Buddy            7
Scout            6
Jack             6
Bella            6
Stanley          6
Rusty            6
Koda             6
Dave             6
Oscar            6
Leo              6
Jax              6
Milo             6
Chester          5
Gus              5
Finn             5
Louis            5
              ...
Reptar           1
Jaycob           1
Stuart           1
Spanky           1
Emmie            1
Julius           1
Lipton           1
Ridley           1
Butters          1
Tug              1
Jarvis           1
Willy            1
Ralphus          1
Cleopatricia     1
O                1
Bradley          1
Bobble           1
Jazz             1
Fynn             1
Binky            1
Rolf             1
Stark            1
Howie            1
Bluebert         1
```

```
Mitch           1
Perry           1
Jomathan        1
Kingsley        1
Burt            1
Arnold          1
Name: name, Length: 932, dtype: int64
```

Define

Remove duplicated tweet ids in tweet_data

Code

In [864]:

```
sum(tweet_data_clean.tweet_id.duplicated())
```

Out[864]:

```
0
```

In [865]:

```
sum(twitter_archive_clean.tweet_id.duplicated())
```

Out[865]:

```
0
```

In [866]:

```
sum(image_predictions_clean.tweet_id.duplicated())
```

Out[866]:

```
0
```

In [867]:

```
tweet_data_clean.tweet_id.drop_duplicates(inplace=True)
```

In [868]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

Test

In [869]:

```
twitter_archive_clean.info()
```


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [870]:

```
sum(tweet_data_clean.tweet_id.duplicated())
```

Out[870]:

```
0
```

In [871]:

```
tweet_data_clean.tweet_id.value_counts()
```

Out[871]:

```
685532292383666176    1
667160273090932737    1
743545585370791937    1
671163268581498880    1
770655142660169732    1
762316489655476224    1
826598365270007810    1
687109925361856513    1
774314403806253056    1
814530161257443328    1
751937170840121344    1
750071704093859840    1
821886076407029760    1
891689557279858688    1
679527802031484928    1
703382836347330562    1
732585889486888962    1
734776360183431168    1
746131877086527488    1
683773439333797890    1
877316821321428993    1
677557565589463040    1
668268907921326080    1
785872687017132033    1
673355879178194945    1
669359674819481600    1
702598099714314240    1
666776908487630848    1
675798442703122432    1
680934982542561280    1
                     ..
736010884653420544    1
666099513787052032    1
843604394117681152    1
673342308415348736    1
794926597468000259    1
666817836334096384    1
748307329658011649    1
885984800019947520    1
773922284943896577    1
767500508068192258    1
837482249356513284    1
674291837063053312    1
693109034023534592    1
713175907180089344    1
670338931251150849    1
708109389455101952    1
828770345708580865    1
675898130735476737    1
791672322847637504    1
770069151037685760    1
685169283572338688    1
672538107540070400    1
760252756032651264    1
676496375194980353    1
```

```
692417313023332352    1
667119796878725120    1
688828561667567616    1
834931633769889797    1
836989968035819520    1
700151421916807169    1
Name: tweet_id, Length: 1978, dtype: int64
```

Define

Merge the uneccesary 'doggo', 'floofer', 'pupper' and 'puppo' columns into one column 'dog_stage'.

Code

In [872]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [873]:

```
twitter_archive_clean.loc[twitter_archive_clean['doggo'] == 'doggo', 'dog_class'] = 'doggo'
twitter_archive_clean.loc[twitter_archive_clean['floofer'] == 'floofer', 'dog_class'] = 'fl
twitter_archive_clean.loc[twitter_archive_clean['pupper'] == 'pupper', 'dog_class'] = 'pupp
twitter_archive_clean.loc[twitter_archive_clean['puppo'] == 'puppo', 'dog_class'] = 'puppo'
```

Test

In [874]:

```
twitter_archive_clean.sample(10)
```

Out[874]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
828	768970937022709760	NaN	NaN	2016-08-26 00:38:52 +0000	<
1326	706153300320784384	NaN	NaN	2016-03-05 16:24:01 +0000	
1713	680473011644985345	NaN	NaN	2015-12-25 19:39:43 +0000	href="http://twitt
16	888917238123831296	NaN	NaN	2017-07-23 00:22:39 +0000	href="http://twitt
1426	697881462549430272	NaN	NaN	2016-02-11 20:34:41 +0000	href="http://twitt
2156	669597912108789760	NaN	NaN	2015-11-25 19:25:57 +0000	href="http://twitt
831	768609597686943744	NaN	NaN	2016-08-25 00:43:02 +0000	href="http://twitt
910	757611664640446465	NaN	NaN	2016-07-25 16:21:11 +0000	href="http://twitt
2078	670832455012716544	NaN	NaN	2015-11-29 05:11:35 +0000	href="http://twitt
2321	666435652385423360	NaN	NaN	2015-11-17 02:00:15 +0000	href="http://twitt

In [875]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 18 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dog_class                     380 non-null object
dtypes: float64(4), int64(3), object(11)
memory usage: 331.4+ KB
```

In [876]:

```
# dropping unneded doggo, floofer, pupper or poppo columns
twitter_archive_clean = twitter_archive_clean.drop(['doggo', 'floofer', 'pupper', 'puppo'],
```

In [877]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 14 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
dog_class                     380 non-null object
dtypes: float64(4), int64(3), object(7)
memory usage: 257.8+ KB
```

Define

Merge tweet_info and image_predictions to twitter_archive table.

Code

In [878]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 14 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
dog_class                     380 non-null object
dtypes: float64(4), int64(3), object(7)
memory usage: 257.8+ KB
```

In [879]:

```
twitter_archive_clean = pd.merge(left=twitter_archive_clean,
                                 right=tweet_data_clean, left_on='tweet_id', right_on='twee
```

In [880]:

```
twitter_archive_clean = twitter_archive_clean.merge(image_predictions_clean, on='tweet_id',
```

In [ ]:

```

```

Test

In [881]:

```
sum(twitter_archive_clean.tweet_id.duplicated())
```

Out[881]:

```
0
```

In [882]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 30 columns):
tweet_id                      1978 non-null int64
in_reply_to_status_id         21 non-null float64
in_reply_to_user_id           21 non-null float64
timestamp                     1978 non-null object
source                        1978 non-null object
text                          1978 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 1978 non-null object
rating_numerator              1978 non-null int64
rating_denominator            1978 non-null int64
name                          1978 non-null object
dog_class                     305 non-null object
favorites                     1978 non-null int64
retweets                      1978 non-null int64
user_followers                0 non-null float64
user_favourites               0 non-null float64
date_time                     1978 non-null object
jpg_url                       1978 non-null object
img_num                       1978 non-null int64
p1                            1978 non-null object
p1_conf                       1978 non-null float64
p1_dog                        1978 non-null bool
p2                            1978 non-null object
p2_conf                       1978 non-null float64
p2_dog                        1978 non-null bool
p3                            1978 non-null object
p3_conf                       1978 non-null float64
p3_dog                        1978 non-null bool
dtypes: bool(3), float64(9), int64(6), object(12)
memory usage: 438.5+ KB
```

Define

Remove rows where there are no images (expanded_urls).

Code

In [883]:

```
twitter_archive_clean = twitter_archive_clean.dropna(subset=['expanded_urls'])
```

Test

In [884]:

```
sum(twitter_archive_clean['expanded_urls'].isnull())
```

Out[884]:

```
0
```

Define

Remove non original tweets (retweets).

Code

In [885]:

```
twitter_archive_clean = twitter_archive_clean[twitter_archive_clean['retweeted_status_id'].
```

Test

In [886]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 30 columns):
tweet_id                      1978 non-null int64
in_reply_to_status_id         21 non-null float64
in_reply_to_user_id           21 non-null float64
timestamp                     1978 non-null object
source                        1978 non-null object
text                          1978 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 1978 non-null object
rating_numerator              1978 non-null int64
rating_denominator            1978 non-null int64
name                          1978 non-null object
dog_class                     305 non-null object
favorites                     1978 non-null int64
retweets                      1978 non-null int64
user_followers                0 non-null float64
user_favourites               0 non-null float64
date_time                     1978 non-null object
jpg_url                       1978 non-null object
img_num                       1978 non-null int64
p1                            1978 non-null object
p1_conf                       1978 non-null float64
p1_dog                        1978 non-null bool
p2                            1978 non-null object
p2_conf                       1978 non-null float64
p2_dog                        1978 non-null bool
p3                            1978 non-null object
p3_conf                       1978 non-null float64
p3_dog                        1978 non-null bool
dtypes: bool(3), float64(9), int64(6), object(12)
memory usage: 438.5+ KB
```

Define

Drop unnecessary retween columns

Code

In [887]:

```
columns = ['retweeted_status_id', 'retweeted_status_user_id', 'retweeted_status_timestamp']
twitter_archive_clean = twitter_archive_clean.drop(columns, axis=1)
```

Test

In [888]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 27 columns):
tweet_id                 1978 non-null int64
in_reply_to_status_id    21 non-null float64
in_reply_to_user_id      21 non-null float64
timestamp                1978 non-null object
source                   1978 non-null object
text                     1978 non-null object
expanded_urls            1978 non-null object
rating_numerator         1978 non-null int64
rating_denominator       1978 non-null int64
name                     1978 non-null object
dog_class                305 non-null object
favorites                1978 non-null int64
retweets                 1978 non-null int64
user_followers           0 non-null float64
user_favourites          0 non-null float64
date_time                1978 non-null object
jpg_url                  1978 non-null object
img_num                  1978 non-null int64
p1                       1978 non-null object
p1_conf                  1978 non-null float64
p1_dog                   1978 non-null bool
p2                       1978 non-null object
p2_conf                  1978 non-null float64
p2_dog                   1978 non-null bool
p3                       1978 non-null object
p3_conf                  1978 non-null float64
p3_dog                   1978 non-null bool
dtypes: bool(3), float64(7), int64(6), object(11)
memory usage: 392.1+ KB
```

Define

Change source column to more readuable categories.

Code

In [889]:

```
# Remove url from sources
twitter_archive_clean['source'] = twitter_archive_clean['source'].str.replace('<a href="htt
twitter_archive_clean['source'] = twitter_archive_clean['source'].str.replace('<a href="htt
twitter_archive_clean['source'] = twitter_archive_clean['source'].str.replace('<a href="htt
twitter_archive_clean['source'] = twitter_archive_clean['source'].str.replace('<a href="htt
```

Test

In [890]:

```
twitter_archive_clean.source.value_counts()
```

Out[890]:

```
Twitter for iPhone    1941
Twitter Web Client      28
TweetDeck                9
Name: source, dtype: int64
```

Define

Change datatypes to datetime, dog_stage to categorical, and tweet_id, in_reply_to_status_id, and in_reply_to_user_id to strings.

Code

In [891]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 27 columns):
tweet_id                 1978 non-null int64
in_reply_to_status_id    21 non-null float64
in_reply_to_user_id      21 non-null float64
timestamp                1978 non-null object
source                   1978 non-null object
text                     1978 non-null object
expanded_urls            1978 non-null object
rating_numerator         1978 non-null int64
rating_denominator       1978 non-null int64
name                     1978 non-null object
dog_class                305 non-null object
favorites                1978 non-null int64
retweets                 1978 non-null int64
user_followers           0 non-null float64
user_favourites          0 non-null float64
date_time                1978 non-null object
jpg_url                  1978 non-null object
img_num                  1978 non-null int64
p1                       1978 non-null object
p1_conf                  1978 non-null float64
p1_dog                   1978 non-null bool
p2                       1978 non-null object
p2_conf                  1978 non-null float64
p2_dog                   1978 non-null bool
p3                       1978 non-null object
p3_conf                  1978 non-null float64
p3_dog                   1978 non-null bool
dtypes: bool(3), float64(7), int64(6), object(11)
memory usage: 392.1+ KB
```

In [892]:

```
twitter_archive_clean['dog_class'] = twitter_archive_clean['dog_class'].astype('category')
twitter_archive_clean['timestamp'] = pd.to_datetime(twitter_archive_clean['timestamp'])
twitter_archive_clean['in_reply_to_status_id'] = twitter_archive_clean['in_reply_to_status_
twitter_archive_clean['in_reply_to_user_id'] = twitter_archive_clean['in_reply_to_user_id']
```

Test

In [893]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 27 columns):
tweet_id                 1978 non-null int64
in_reply_to_status_id    1978 non-null object
in_reply_to_user_id      1978 non-null object
timestamp                1978 non-null datetime64[ns, UTC]
source                   1978 non-null object
text                     1978 non-null object
expanded_urls            1978 non-null object
rating_numerator         1978 non-null int64
rating_denominator       1978 non-null int64
name                     1978 non-null object
dog_class                305 non-null category
favorites                1978 non-null int64
retweets                 1978 non-null int64
user_followers           0 non-null float64
user_favourites          0 non-null float64
date_time                1978 non-null object
jpg_url                  1978 non-null object
img_num                  1978 non-null int64
p1                       1978 non-null object
p1_conf                  1978 non-null float64
p1_dog                   1978 non-null bool
p2                       1978 non-null object
p2_conf                  1978 non-null float64
p2_dog                   1978 non-null bool
p3                       1978 non-null object
p3_conf                  1978 non-null float64
p3_dog                   1978 non-null bool
dtypes: bool(3), category(1), datetime64[ns, UTC](1), float64(5), int64(6),
object(11)
memory usage: 378.8+ KB
```

Define

Change missing values in 'name' from 'None' to NaN (dog stages already covered).

Code

In [894]:

```
twitter_archive_clean['name'] = twitter_archive_clean['name'].replace('None', np.NaN)
```

Test

In [895]:

```
twitter_archive_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1978 entries, 0 to 1977
Data columns (total 27 columns):
tweet_id                 1978 non-null int64
in_reply_to_status_id    1978 non-null object
in_reply_to_user_id      1978 non-null object
timestamp                1978 non-null datetime64[ns, UTC]
source                   1978 non-null object
text                     1978 non-null object
expanded_urls            1978 non-null object
rating_numerator         1978 non-null int64
rating_denominator       1978 non-null int64
name                     1342 non-null object
dog_class                305 non-null category
favorites                1978 non-null int64
retweets                 1978 non-null int64
user_followers           0 non-null float64
user_favourites          0 non-null float64
date_time                1978 non-null object
jpg_url                  1978 non-null object
img_num                  1978 non-null int64
p1                       1978 non-null object
p1_conf                  1978 non-null float64
p1_dog                   1978 non-null bool
p2                       1978 non-null object
p2_conf                  1978 non-null float64
p2_dog                   1978 non-null bool
p3                       1978 non-null object
p3_conf                  1978 non-null float64
p3_dog                   1978 non-null bool
dtypes: bool(3), category(1), datetime64[ns, UTC](1), float64(5), int64(6),
object(11)
memory usage: 378.8+ KB
```

Define

Remove any all extreme values from the numertor and denuminator columns

Code

In [896]:

```
twitter_archive_clean = twitter_archive_clean[twitter_archive_clean['rating_numerator'] !=
twitter_archive_clean = twitter_archive_clean[twitter_archive_clean['rating_denominator'] <
twitter_archive_clean = twitter_archive_clean[twitter_archive_clean['rating_numerator'] <=
```

Test

In [897]:

```
len(twitter_archive_clean[twitter_archive_clean['rating_numerator'] > 100 ])
```

Out[897]:

```
0
```

In [898]:

```
len(twitter_archive_clean[twitter_archive_clean['rating_denominator'] > 100 ])
```

Out[898]:

```
0
```

Define

Columns p1, p2 and p3 in image_predictions are not very meaningful, change to something more understandable

Code

In [899]:

```
itter_archive_clean.rename(columns={'p1':'Breed_Probability1', 'p2':'Breed_Probability2', 'p
itter_archive_clean.rename(columns={'p1_conf':'Breed_Confidence1', 'p2_conf':'Breed_Confiden
itter_archive_clean.rename(columns={'p1_dog':'Dog_Flag_1', 'p2_dog':'Dog_Flag_2', 'p3_dog':'
```

Test

In [900]:

```
twitter_archive_clean.sample(5)
```

Out[900]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source
945	712092745624633345	nan	nan	2016-03-22 01:45:15+00:00	Twitter for iPhone
1243	687317306314240000	nan	nan	2016-01-13 16:56:30+00:00	Twitter for iPhone
1641	672245253877968896	nan	nan	2015-12-03 02:45:32+00:00	Twitter for iPhone
1138	695314793360662529	nan	nan	2016-02-04 18:35:39+00:00	Twitter for iPhone
1181	691483041324204033	nan	nan	2016-01-25 04:49:38+00:00	Twitter for iPhone

5 rows × 27 columns

Store

In [901]:

```
# Sotre the cleaned DataFrame to a csv file
twitter_archive_clean.drop(twitter_archive_clean.columns[twitter_archive_clean.columns.str.
twitter_archive_clean.to_csv('./Data/twitter_archive_master.csv', encoding = 'utf-8', index
```

In [902]:

```
twitter_archive_clean = pd.read_csv('./Data/twitter_archive_master.csv')
```

In [903]:

```
twitter_archive_clean.columns
```

Out[903]:

```
Index(['tweet_id', 'in_reply_to_status_id', 'in_reply_to_user_id', 'timestam
p',
       'source', 'text', 'expanded_urls', 'rating_numerator',
       'rating_denominator', 'name', 'dog_class', 'favorites', 'retweets',
       'user_followers', 'user_favourites', 'date_time', 'jpg_url', 'img_nu
m',
       'Breed_Probability1', 'Breed_Confidence1', 'Dog_Flag_1',
       'Breed_Probability2', 'Breed_Confidence2', 'Dog_Flag_2',
       'Breed_Probability3', 'Breed_Confidence3', 'Dog_Flag_3'],
      dtype='object')
```

In [904]:

```
twitter_archive_clean.head(3)
```

Out[904]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56+00:00	Twitter for iPhone	T Phi H my boy.
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27+00:00	Twitter for iPhone	T Tilly. che pı y
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03+00:00	Twitter for iPhone	T Archi is a Norw Pour

3 rows × 27 columns

Report & Analysis Steps continues in act_report.ipynb

References

- Title Image (https://pixabay.com/illustrations/social-media-media-board-networking-1989152/)
- Reading and Writing to a JSON (https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/)
- How to create a pandas dataframe using Tweepy? (https://stackoverflow.com/questions/47925828/how-to-create-a-pandas-dataframe-using-tweepy)

- Tweepy Documentation (https://tweepy.readthedocs.io/en/latest/getting_started.html)
- WeRateDogs Project by kdow (https://github.com/kdow/WeRateDogs/blob/master/wrangle_act.ipynb)

In [ ]:

```

```