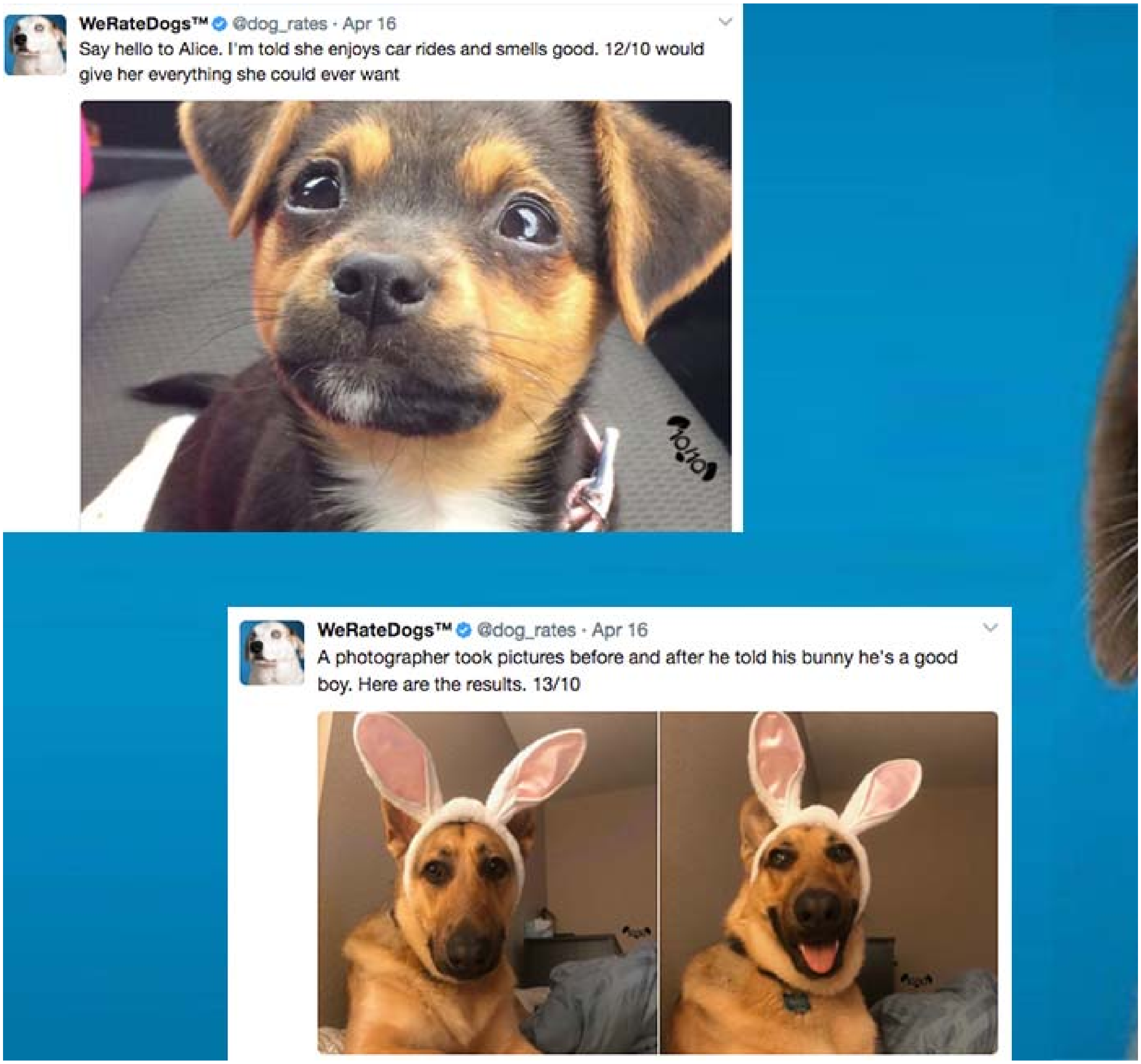
Introduction

Real-world data rarely comes clean. Using Python and its libraries, you will gather data from a variety of sources and in a variety of formats, assess its quality and tidiness, then clean it. This is called data wrangling. You will document your wrangling efforts in a Jupyter Notebook, plus showcase them through analyses and visualizations using Python (and its libraries) and/or SQL.

The dataset that you will be wrangling (and analyzing and visualizing) is the tweet archive of Twitter user **@dog_rates**, also known as **WeRateDogs**. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "**they're good dogs Brent**." 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 you 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. More on this soon.


WeRateDogs™ @dog_rates · Apr 16
Say hello to Alice. I'm told she enjoys car rides and smells good. 12/10 would give her everything she could ever want

WeRateDogs™ @dog_rates · Apr 16
A photographer took pictures before and after he told his bunny he's a good boy. Here are the results. 13/10

Image via Boston Magazine

What Software Do I Need?

The entirety of this project can be completed inside the Udacity classroom on the **Project Workspace** page using the Jupyter Notebook Workspace provided there.
If you want to work outside of the Udacity classroom, the following software requirements apply:

- You need to be able to work in a Jupyter Notebook on your computer. Please revisit our Jupyter Notebook and Anaconda tutorials earlier in the Nanodegree program for installation instructions.
- The following packages (i.e. libraries) need to be installed. You can install these packages via conda or pip. Please revisit our Anaconda tutorial earlier in the Nanodegree program for package installation instructions.

- pandas
- numpy
- requests
- tweepy
- json
- You need to be able to create written documents that contain images and you need to be able to export these documents as PDF files. This task can be done in a Jupyter Notebook, but you might prefer to use a word processor like **Google Docs**, which is free, or Microsoft Word.
- A text editor, like **Sublime**, which is free, will be useful but is not required.

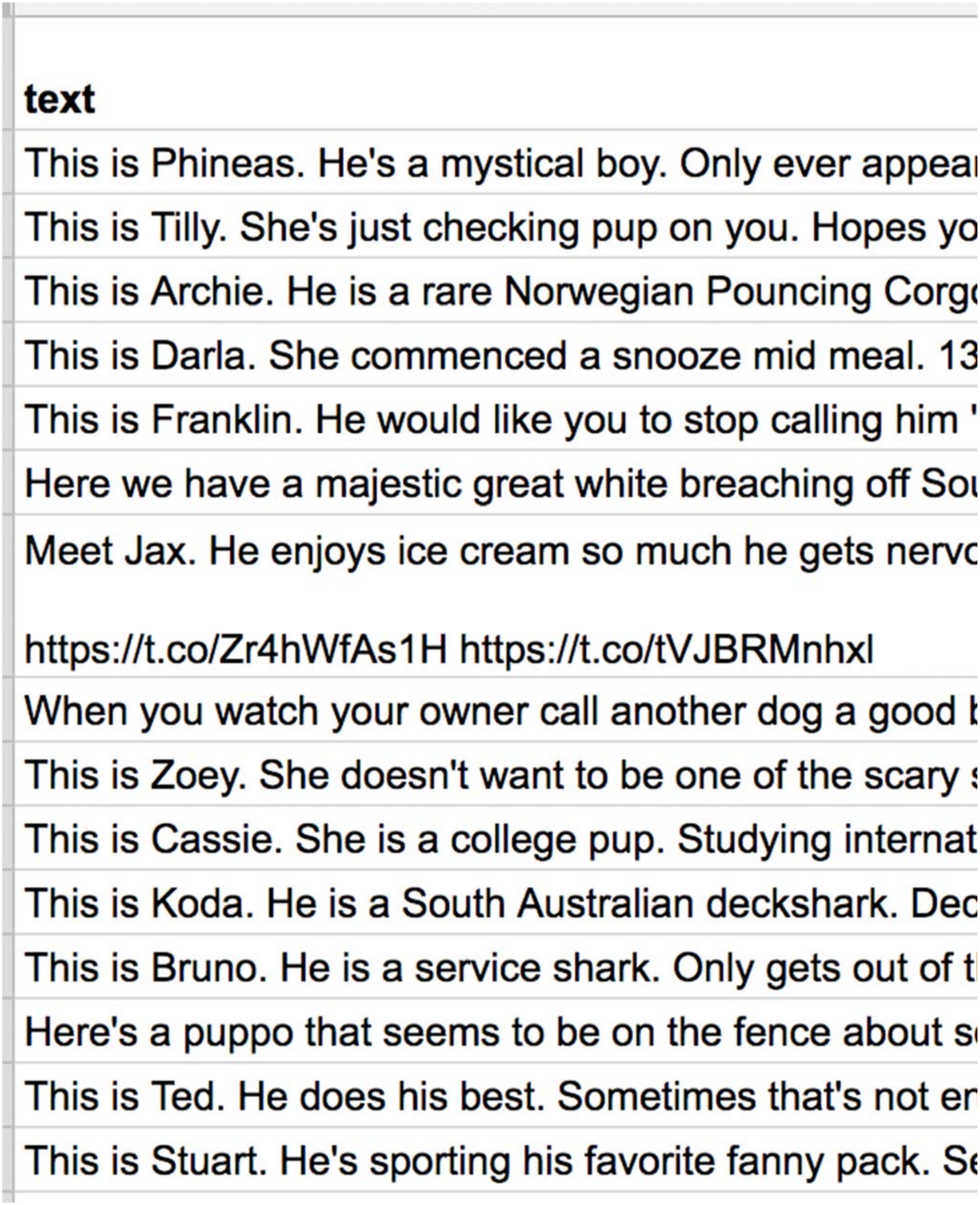
Project Motivation

Context

Your goal: wrangle WeRateDogs Twitter data to create interesting and trustworthy analyses and visualizations. The Twitter archive is great, but it only contains very basic tweet information. Additional gathering, then assessing and cleaning is required for "*Wow!*"-worthy analyses and visualizations.

The Data

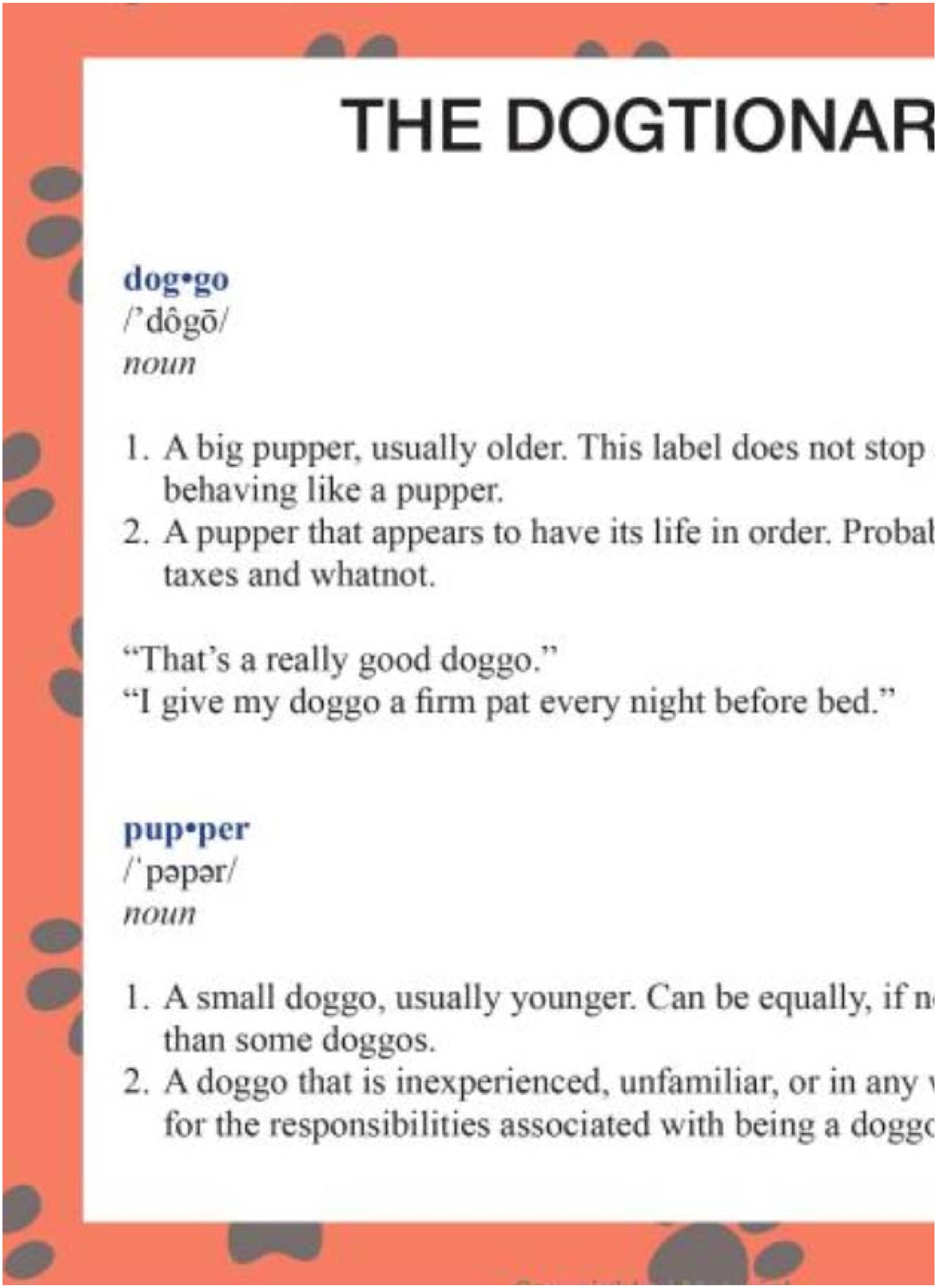
Enhanced Twitter Archive
The WeRateDogs Twitter archive contains basic tweet data for all 5000+ of their tweets, but not everything. One column the archive does contain though: each tweet's text, which I used to extract rating, dog name, and dog "stage" (i.e. doggo, floofer, pupper, and puppo) to make this Twitter archive "enhanced." Of the 5000+ tweets, I have filtered for tweets with ratings only (there are 2356).



text
This is Phineas. He's a mystical boy. Only ever appear
This is Tilly. She's just checking pup on you. Hopes yo
This is Archie. He is a rare Norwegian Pouncing Corg
This is Darla. She commenced a snooze mid meal. 13
This is Franklin. He would like you to stop calling him '
Here we have a majestic great white breaching off Sou
Meet Jax. He enjoys ice cream so much he gets nerv https://t.co/Zr4hWfAs1H https://t.co/tVJBRMnhxl
When you watch your owner call another dog a good b
This is Zoey. She doesn't want to be one of the scary s
This is Cassie. She is a college pup. Studying internat
This is Koda. He is a South Australian deckshark. Dec
This is Bruno. He is a service shark. Only gets out of t
Here's a puppo that seems to be on the fence about s
This is Ted. He does his best. Sometimes that's not en
This is Stuart. He's sporting his favorite fanny pack. Se

The extracted data from each tweet's text

I extracted this data programmatically, but I didn't do a very good job. The ratings probably aren't all correct. Same goes for the dog names and probably dog stages (see below for more information on these) too. You'll need to assess and clean these columns if you want to use them for analysis and visualization.



THE DOGTIONAR

dog•go
/'dôgō/
noun

1. A big pupper, usually older. This label does not stop behaving like a pupper.
2. A pupper that appears to have its life in order. Probal taxes and whatnot.

"That's a really good doggo."
"I give my doggo a firm pat every night before bed."

pup•per
/'pəpər/
noun

1. A small doggo, usually younger. Can be equally, if n than some doggos.
2. A doggo that is inexperienced, unfamiliar, or in any for the responsibilities associated with being a doggc

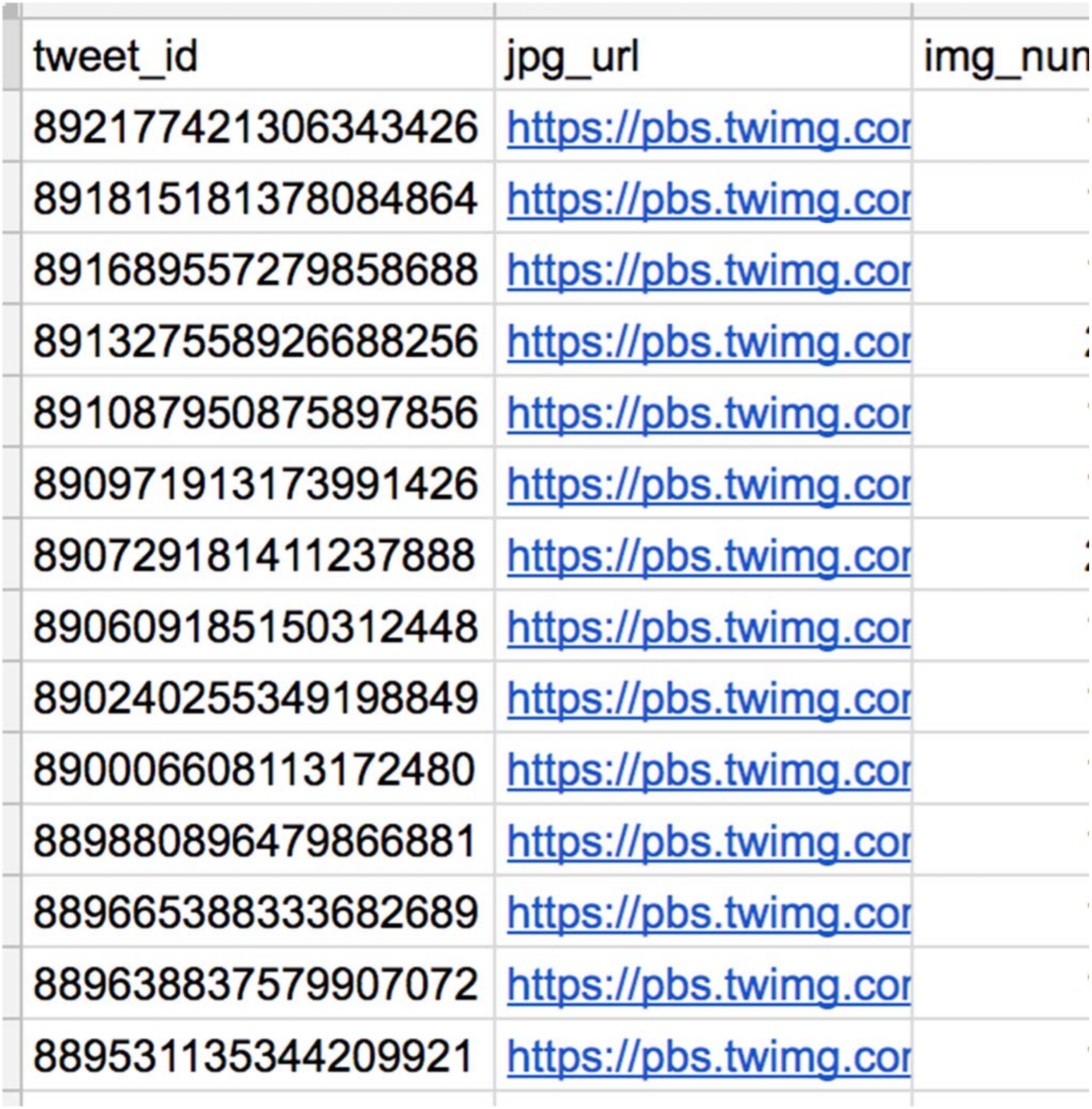
***The Dogtionary explains the various stages of dog: doggo, pupper, puppo, and floof(er) (via the [#WeRateDogs book](#)** on Amazon)*

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 you, because you have the WeRateDogs Twitter archive and specifically the tweet IDs within it, can gather this data for all 5000+. And guess what? You're going to query Twitter's API to gather this valuable data.

Image Predictions File

One more cool thing: I ran every image in the WeRateDogs Twitter archive through a **neural network** that can classify breeds of dogs*. The results: 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).



tweet_id	jpg_url	img_nun
892177421306343426	https://pbs.twimg.cor	
891815181378084864	https://pbs.twimg.cor	
891689557279858688	https://pbs.twimg.cor	
891327558926688256	https://pbs.twimg.cor	
891087950875897856	https://pbs.twimg.cor	
890971913173991426	https://pbs.twimg.cor	
890729181411237888	https://pbs.twimg.cor	
890609185150312448	https://pbs.twimg.cor	
890240255349198849	https://pbs.twimg.cor	
890006608113172480	https://pbs.twimg.cor	
889880896479866881	https://pbs.twimg.cor	
889665388333682689	https://pbs.twimg.cor	
889638837579907072	https://pbs.twimg.cor	
889531135344209921	https://pbs.twimg.cor	

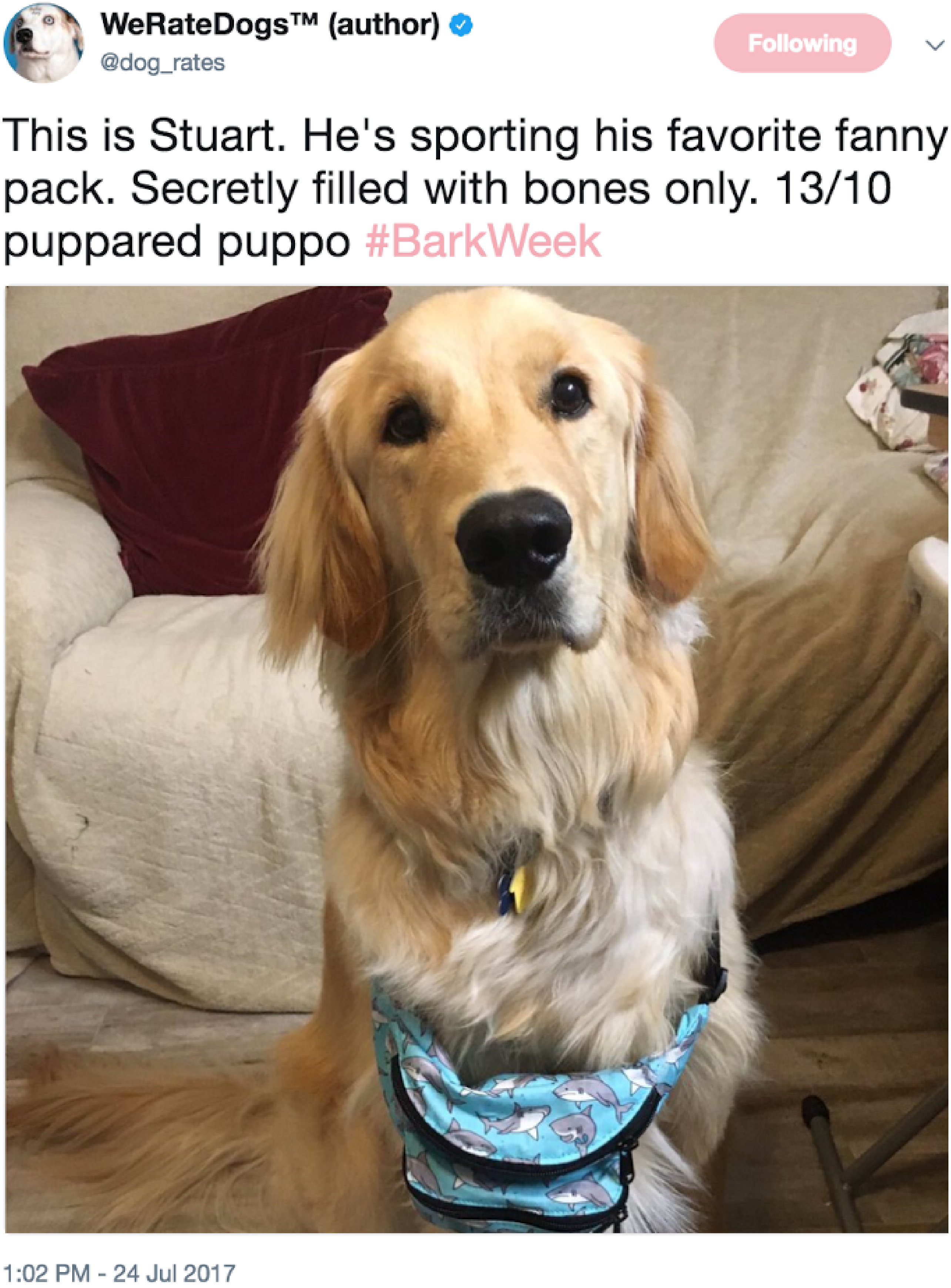
Tweet image prediction data

So for the last row in that table:

- tweet_id is the last part of the tweet URL after "*status/*" → **https://twitter.com/dog_rates/status/889531135344209921**
- p1 is the algorithm's #1 prediction for the image in the tweet → **golden retriever**
- p1_conf is how confident the algorithm is in its #1 prediction → **95%**
- p1_dog is whether or not the #1 prediction is a breed of dog → **TRUE**
- p2 is the algorithm's second most likely prediction → **Labrador retriever**
- p2_conf is how confident the algorithm is in its #2 prediction → **1%**

- p2_dog is whether or not the #2 prediction is a breed of dog → **TRUE**
- etc.

  And the #1 prediction for the image in that tweet was spot on:



WeRateDogs™ (author)
@dog_rates

This is Stuart. He's sporting his favorite fanny pack. Secretly filled with bones only. 13/10 puppared puppo #BarkWeek

1:02 PM - 24 Jul 2017

A golden retriever named Stuart

So that's all fun and good. But all of this additional data will need to be gathered, assessed, and cleaned. This is where you come in.

Key Points

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

- You 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.
- You do *not* need to gather the tweets beyond August 1st, 2017. You can, but note that you won't be able to gather the image predictions for these tweets since you don't have access to the algorithm used.
  Fun fact: creating this neural network is one of the projects in Udacity's **Machine Learning Engineer Nanodegree program and **Artificial Intelligence Nanodegree program**.*

Project Details

Your tasks in this project are as follows:

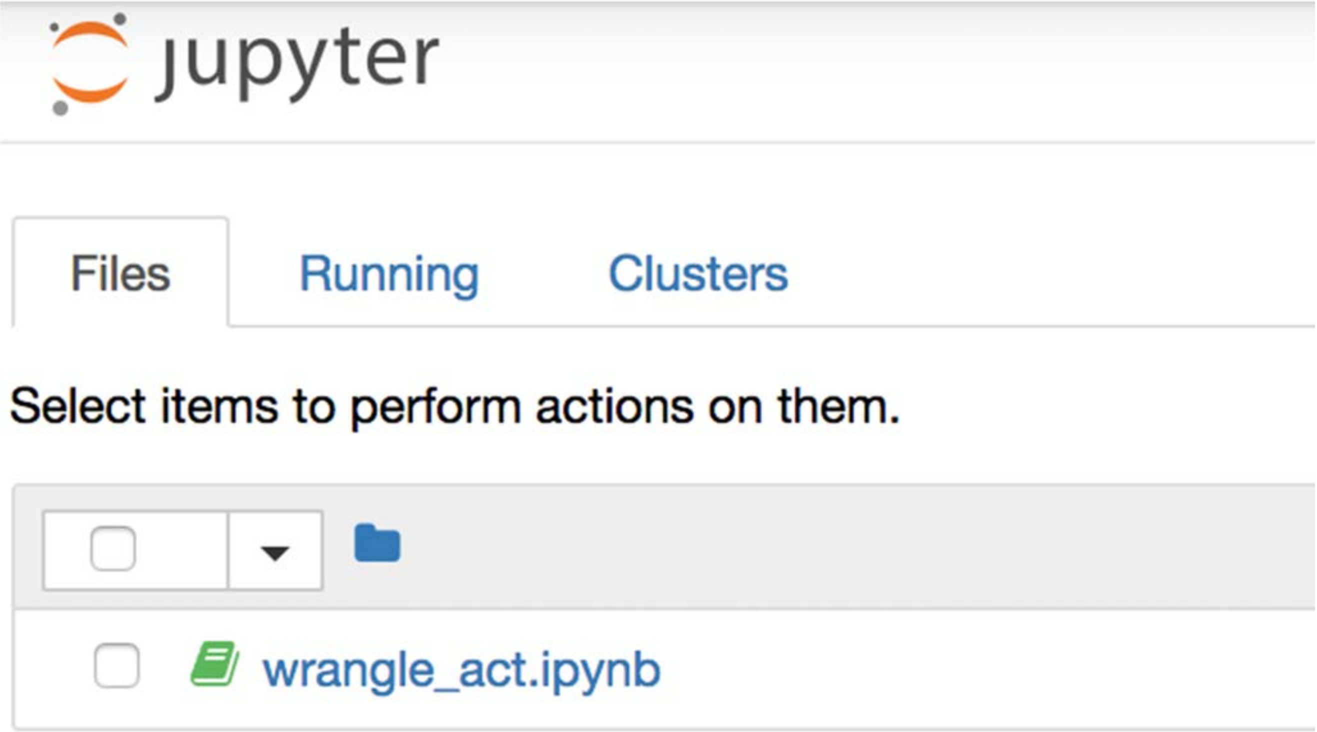
- Data wrangling, which consists of:
- Gathering data
- Assessing data
- Cleaning data
- Storing, analyzing, and visualizing your wrangled data
- Reporting on 1) your data wrangling efforts and 2) your data analyses and visualizations

Gathering Data for this Project

Gather each of the three pieces of data as described below in a Jupyter Notebook titled `wrangle_act.ipynb`:

1. The WeRateDogs Twitter archive. I am giving this file to you, so imagine it as a file on hand. Download this file manually by clicking the following link: **`twitter_archive_enhanced.csv`**
2. 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`) is hosted on Udacity's servers and should be downloaded programmatically using the **Requests** library and the following URL: **https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv**
3. Each tweet's retweet count and favorite ("like") count at minimum, and any additional data you find interesting. Using the tweet IDs in the WeRateDogs Twitter archive, query the

Twitter API for each tweet's JSON data using Python's **Tweepy** library and store each tweet's entire set of JSON data in a file called `tweet_json.txt` file. Each tweet's JSON data should be written to its own line. Then read this .txt file line by line into a pandas DataFrame with (at minimum) tweet ID, retweet count, and favorite count. *Note: do not include your Twitter API keys, secrets, and tokens in your project submission.*
If you decide to complete your project in the Project Workspace, note that you can upload files to the Jupyter Notebook Workspace by clicking the "Upload" button in the top righthand corner of the dashboard.



Select items to perform actions on them.

wrangle_act.ipynb

Jupyter Notebook dashboard with arrow pointing to the "Upload" button

Assessing Data for this Project

After gathering each of the above pieces of data, assess them visually and programmatically for quality and tidiness issues. Detect and document at least **eight (8) quality issues** and **two (2) tidiness issues** in your `wrangle_act.ipynb` Jupyter Notebook. To meet specifications, the issues that satisfy the Project Motivation (see the *Key Points* header on the previous page) must be assessed.

Cleaning Data for this Project

Clean each of the issues you documented while assessing. Perform this cleaning in `wrangle_act.ipynb` as well. The result should be a high quality and tidy master pandas DataFrame (or DataFrames, if appropriate). Again, the issues that satisfy the Project Motivation must be cleaned.

Storing, Analyzing, and Visualizing Data for this Project

Store the clean DataFrame(s) in a CSV file with the main one named `twitter_archive_master.csv`. If additional files exist because multiple tables are required for tidiness, name these files appropriately. Additionally, you may store the cleaned data in a SQLite database (which is to be submitted as well if you do).

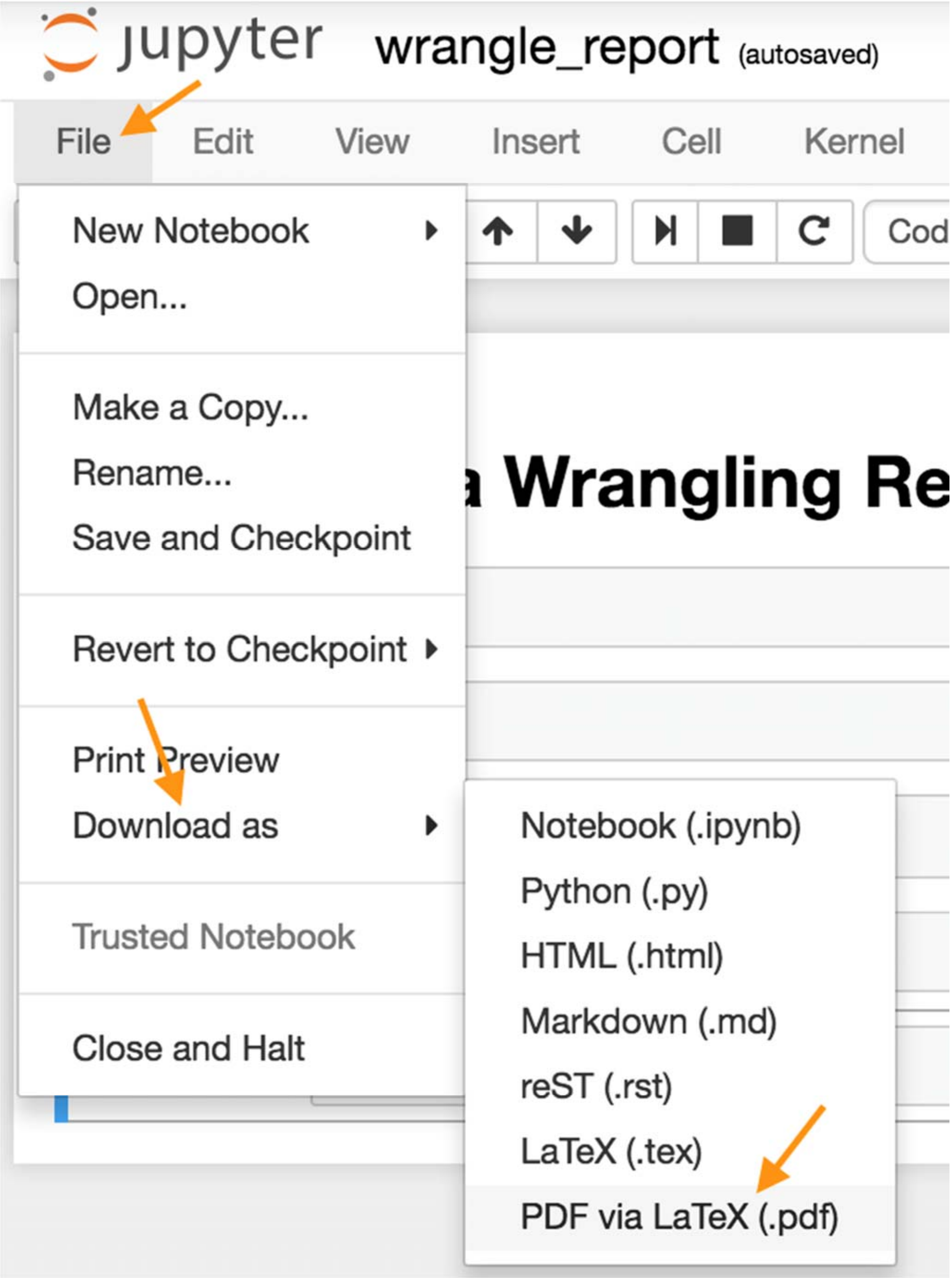
Analyze and visualize your wrangled data in your `wrangle_act.ipynb` Jupyter Notebook. At least **three (3) insights and one (1) visualization** must be produced.

Reporting for this Project

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.

Create a **250-word-minimum written report** called `act_report.pdf` or `act_report.html` that communicates the insights and displays the visualization(s) produced from your wrangled data. This is to be framed as an external document, like a blog post or magazine article, for example.

Both of these documents can be created in separate Jupyter Notebooks using the **Markdown functionality** of Jupyter Notebooks, then downloading those notebooks as PDF files or HTML files (see image below). You might prefer to use a word processor like Google Docs or Microsoft Word, however.


jupyter
wrangle_report (autosaved)
File
Edit
View
Insert
Cell
Kernel
Cod
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt
Notebook (.ipynb)
Python (.py)
HTML (.html)
Markdown (.md)
reST (.rst)
LaTeX (.tex)
PDF via LaTeX (.pdf)
a Wrangling Re

Download Jupyter Notebook in PDF form

How to Query Twitter Data

In this project, you'll be using Tweepy 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) in Lesson 2. 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, you need to set up your own Twitter application. And before that, you must sign up for a Twitter account. This guide describes the setup process well.* Once you have these set up, the following code, which is provided in the Getting started portion of the tweepy documentation, will create an API object that you can use to gather Twitter data.

```
import tweepy

consumer_key = 'YOUR CONSUMER KEY'
consumer_secret = 'YOUR CONSUMER SECRET'
access_token = 'YOUR ACCESS TOKEN'
access_secret = 'YOUR ACCESS SECRET'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
```

**Note: If you can't set up an application because of mobile verification issues, please email david.venturi@udacity.com.*

Tweet data is stored in JSON format by Twitter. Getting tweet JSON data via tweet ID using Tweepy is described well in this StackOverflow answer. Note that setting the `tweet_mode`parameter to `'extended'` in the `get_status` call, i.e., `api.get_status(tweet_id, tweet_mode='extended')`, can be useful.

Also, note that the tweets corresponding to a few tweet IDs in the archive may have been deleted. Try-except blocks may come in handy here.

Do Not Include Your API Keys, Secrets, and Tokens in Your Submission

Do **not** include your API keys, secrets, and tokens in your project submission. This is standard practice for APIs and public code.

Twitter's Rate Limit

Twitter's API has a rate limit. Rate limiting is used to control the rate of traffic sent or received by a server. As per Twitter's rate limiting info page:

Rate limits are divided into 15 minute intervals

To query all of the tweet IDs in the WeRateDogs Twitter archive, 20-30 minutes of running time can be expected. Printing out each tweet ID after it was queried and using a code timer were both helpful for sanity reasons. Setting the `wait_on_rate_limit` and `wait_on_rate_limit_notify`parameters to `True` in the `tweepy.api` class is useful as well.

Writing and Reading Twitter JSON

After querying each tweet ID, you will write its JSON data to the required `tweet_json.txt` file with each tweet's JSON data on its own line. You will then read this file, line by line, to create a pandas DataFrame that you will soon assess and clean. This **Reading and Writing JSON to a File in Python** article from Stack Abuse, will be useful.