Social Media, Bots and APIs

Sentiment Analysis

Disclaimer

- This notebook uses raw data collected from Twitter, which means you have to be prepared for the following:
  - Despite running this data through a profanity filter, it may still contain themes or language that is not school appropriate
  - Some tweets may address current events that you may be uncomfortable with (in particular, the COVID-19 pandemic)
- Our job as data scientists is to try to view this data through an academic lens whenever possible, but you have two options if you are not comfortable with this assignment:
  - Option 1: You can choose not to view the raw Twitter data (you can just analyze it in the aggregate)
  - Option 2: You can contact your teacher for an alternate assignment

Ethical Bot Usage

- The instructions you saw in the previous YouTube videos allow you to create an automated Twitter Bot
- While optional, we encourage you to experiment and try this out for yourself!
- But please remember the following guidelines:
  - Ethical guidelines for Bot creators:
    - Inform users that your account is a bot (do not impersonate a real person)
    - Inform users who controls your bot
    - Inform users who can access the data your bot collects
    - A bot should never be created for the purpose of harming its users (theft, fraud, harassment, spam, etc)
  - Fremd High School community guidelines:
    - Any tweets you send will be public (visible to your teachers, parents, classmates, and the community)
    - You are representing Fremd High School with any tweets you send
    - Profanity, harassment, spamming, or any other misbehavior will immediately result in a discipline referral
  - Anti-spam guidelines (even if you spam by accident):
    - Do not send *numerous* unsolicited tweets to the same account
    - Do not try to send the same tweet out over and over again
    - Do not tweet more than once per minute
    - To avoid accidental spamming, use *import time* (https://pythonspot.com/sleep/) and *time.sleep(60)* in any loops that send tweets

Now that this is done, let's learn about sentiment analysis:

Unstructured Data and Sentiment Analysis

Last chapter, you learned about structured, semi-structured, and unstructured data. Structured data from a database or semi-structured data from an XML file are easy to process in an automated fashion. However, most data in the world is unstructured. Twitter is a potential goldmine of data, but not all tweets follow the same structure.

For example, suppose you are a brand manager for Doritos chips. Your boss asks you to determine whether customers have a positive or negative perception of a new product, *Doritos Blaze Chips*. If you had a large team of employees, you could have the employees add every Tweet they could find about the chips to a spreadsheet while another team of employees rates the Tweets as "positive," "negative," or "neutral." This process is called **sentiment analysis**.

While using a great deal of human labor would be an effective way to complete this task, it is very resource intensive. It would be much more efficent to have a computer find and process these tweets automatically. But since this data is unstructured, we can't analyze it as easily as we could in previous activities (such as the XML notebook or the Open Data Project). For many decades, computer scientists didn't make much progress with programs that could perform sentiment analysis.

But in the past decade or so, computer scientists have made great strides with **sentiment analysis** using the tools of **natural language processing** and **machine learning**.

Let's start with some manual processing using human labor. You'll have to provide the labor:

Question #1: Read the following tweet and classify it as "positive," "negative," or "neutral." Briefly explain your reasoning (you should specifically discuss words/phrases/themes that led you to your conclusion).

"Doritos Blaze chips are the grossest chips I've ever eaten. They are so bad that I thought I was eating rancid croutons."

Your Answer: Negative: They sad words like gross/ bad/ rancid

Question #2: Read the following tweet and classify it as "positive," "negative," or "neutral." Briefly explain your reasoning (you should specifically discuss words/phrases/themes that led you to your conclusion).

"I love Doritos Blaze chips! Move over Cool Ranch, this is the tastiest flavor yet. Yum!"

Your Answer:

Positive: Uses words like love, tastiest, and yum

Question #3: Read the following tweet and classify it as "positive," "negative," or "neutral." Briefly explain your reasoning (you should specifically discuss words/phrases/themes that led you to your conclusion).

"Doritos Blaze chips are being sold for $2.99 at Mariano's (545 N Hicks Rd, Palatine, IL)"

Your Answer:

Neutral: Uses a neutral word set for the sentance

Using *Textblob* for Sentiment Analysis

After processing only three tweets, you are probably already getting the sense that this is tedious work to do by hand. Instead, we're going to use a Python module named [textblob (http://textblob.readthedocs.io/en/dev/quickstart.html)](http://textblob.readthedocs.io/en/dev/quickstart.html) to perform our sentiment analysis. This module is user-friendly for programmers who are experimenting with sentiment analysis for the first time. Let's look at the same three tweets from above:

In [1]:
```
from textblob import TextBlob

tweet1 = "Doritos Blaze chips are the grossest chips I've ever eaten. They are so

tbobject1 = TextBlob(tweet1)

tbobject1.sentiment.polarity
```

Out[1]:
```
-0.6999999999999998
```

This value of -0.6999999999999998 is the polarity of the tweet. This value can range from -1.0 to 1.0, with 0 being neutral. In this case, -0.6999999999999998 indicates a tweet with a very negative sentiment.

Now let's try another tweet from above:

In [2]:
```
tweet2 = "I love Doritos Blaze chips! Move over Cool Ranch, this is the tastiest

tbobject2 = TextBlob(tweet2)

tbobject2.sentiment.polarity
```

Out[2]:
```
0.53125
```

A polarity of 0.53125 indicates that this tweet is positive.

Now you try:

Question #4: Is the tweet below positive, negative, or neutral? Perform a sentiment analysis in the cell below that supports your answer.

"Doritos Blaze chips are being sold for $2.99 at Mariano's (545 N Hicks Rd, Palatine, IL)"

Your Answer:

In [3]:
```
# Your code here
from textblob import TextBlob

tweet3 = "Doritos Blaze chips are being sold for $2.99 at Mariano's (545 N Hicks


tbobject3 = TextBlob(tweet3)

tbobject3.sentiment.polarity
```

Out[3]: 0.0

Testing Out Sentiment Analysis

Our main focus of this notebook will be to perform a sentiment analysis on tweets we have collected, but first let's test out our sentiment analysis on a famous poem just to get an idea of whether this Python module can produce a classification that agrees with your own. The poem we are going to analyze is *Still I Rise* by Maya Angelou.

Task #1: Run the code below, and then add a line of code that prints out the full text of this poem.

In [4]:
```
file1 = open("rise.txt", "r", encoding="utf-8")

full_text = file1.read()

print(full_text)
```

```
Still I Rise
By Maya Angelou

You may write me down in history
With your bitter, twisted lies,
You may trod me in the very dirt
But still, like dust, I'll rise.

Does my sassiness upset you?
Why are you beset with gloom?
'Cause I walk like I've got oil wells
Pumping in my living room.

Just like moons and like suns,
With the certainty of tides,
Just like hopes springing high,
Still I'll rise.

Did you want to see me broken?
Bowed head and lowered eyes?
Shoulders falling down like teardrops,
Weakened by my soulful cries?

Does my haughtiness offend you?
Don't you take it awful hard
'Cause I laugh like I've got gold mines
Diggin' in my own backyard.

You may shoot me with your words,
You may cut me with your eyes,
You may kill me with your hatefulness,
But still, like air, I'll rise.

Does my sexiness upset you?
Does it come as a surprise
That I dance like I've got diamonds
At the meeting of my thighs?

Out of the huts of history's shame
I rise
Up from a past that's rooted in pain
I rise
I'm a black ocean, leaping and wide,
Welling and swelling I bear in the tide.

Leaving behind nights of terror and fear
I rise
Into a daybreak that's wondrously clear
I rise
Bringing the gifts that my ancestors gave,
I am the dream and the hope of the slave.
```

```
I rise
I rise
I rise.
```

Poetry is complicated, human emotions are complicated, the history behind Maya Angelou's iconic poem is complicated. It is a bit unrealistic to expect to learn very much about this poem by simply labeling its sentiment as "Positive," "Neutral," or "Negative." But for the sake of this example, you are being asked to do so anyway:

Question #5: Classify the sentiment in this poem as "Positive," "Neutral," or "Negative." There is no right or wrong answer here, but you do have to justify your reasoning just like you would in your English class (you can't just write a single word for your answer).

Your Answer: I think it is neutral, as it has both positive words within it and negative words, and they both balance out.

Note: If you would like to learn more about Maya Angelou or this poem, here's a source you might enjoy reading: https://www.biography.com/news/maya-angelou-still-i-rise (https://www.biography.com/news/maya-angelou-still-i-rise)

Let's compare your classification to a sentiment analysis performed by TextBlob:

Task #2: Write code in the cell below that performs a sentiment analysis on *full_text* (a variable that contains the text of Maya Angelou's poem).

In [5]:
```
# Your code here
from textblob import TextBlob

poem = full_text


tbobject4 = TextBlob(poem)

tbobject4.sentiment.polarity
```

Out[5]:
```
-0.13486928104575163
```

Question #6: Does your sentiment analysis agree or disagree with the analysis performed by TextBlob? How can you tell?

If you and TextBlob disagree, explain why your interpretation of the poem might have been different than the sentiment analysis performed by TextBlob. If your analyses agree, explain why it makes sense that both analyses produced the same result.

Your Answer: We disagreed, but only a little bit. I think this is because me a the tectBlob have diffrent values of words with their connotations, and the negative words had more meanign the textBlob code.

An Abstraction for Classifying Tweets

The Maya Angelou example helps to illustrate some of the risks of sentiment analysis. Out of context, the majority of the words in the poem convey a negative sentiment. Yet the poem's final two stanzas change the entire poem with what some consider to be one of the strongest messages of hope and empowerment in all of American poetry.

This gives us the perfect backdrop for introducing how we will use sentiment analysis on data from Twitter. If we only perform a sentiment analysis on a single sample of text (one poem, one tweet, etc), then there is a small but significant chance that the classifcation will be incorrect. But if we perform a sentiment analysis on thousands or even millions of tweets, then we expect that **(on average)** our analysis is likely to have produced meaningful results.

Since we're going to be performing a sentiment analysis on many tweets in this notebook, we should try to **use abstraction to manage the complexity of this program**:

Task #3: Define a function, *tweet_classifier(input_string)*, that returns the string "positive" if the tweet has a positive polarity, returns the string "negative" if the tweet has a negative polarity, and returns the string "neutral" if the tweet has a polarity of 0.

In [6]:
```
# Your code here
def tweet_classifier(input_string):
    from textblob import TextBlob
    value = ""
    tbobject5 = TextBlob(input_string)
    sentiment = tbobject5.sentiment.polarity
    if sentiment > 0:
        value = "positive"
    elif sentiment < 0:
        value = "negative"
    else:
        value = "neutral"
    print(value)
```

Now test out your function by running this cell and making sure that it behaves as expected on our Doritos tweets. Here are the tweets:

In [7]:
```
print(tweet1)
print(tweet2)
print(tweet3)
```

```
Doritos Blaze chips are the grossest chips I've ever eaten. They are so bad tha
t I thought I was eating rancid croutons.
I love Doritos Blaze chips! Move over Cool Ranch, this is the tastiest flavor y
et. Yum!
Doritos Blaze chips are being sold for $2.99 at Mariano's (545 N Hicks Rd, Pala
tine, IL)
```

And the results of a sentiment analysis:

In [8]:
```
print(tweet_classifier(tweet1))
print(tweet_classifier(tweet2))
print(tweet_classifier(tweet3))
```

```
negative
None
positive
None
neutral
None
```

Picking a Hashtag to Analyze

Normally, the best way to pick a hashtag to analyze is to look at what is currently trending on Twitter. However, since creating a Twitter developer account was optional for the e-learning version of this notebook, your teacher has pre-selected some hashtags that were trending on May 10, 2020.

Question #7: Which of the following hashtags would you like to analyze in this notebook?

- #covid19
- #mothersday
- #happymothersday
- #bitcoin
- #sundaythoughts

Your Answer: #covid19

Task #4: Each of the hashtags above has an associated *.txt* file. Modifying the code below to include the appropriate file name:

- *"covid19.txt"*
- *"mothersday.txt"*
- *"happymothersday.txt"*
- *"bitcoin.txt"*
- *"sundaythoughts.txt"*

In [9]:
```
#*********************************************************************************
filename = "covid19.txt" #IMPORTANT: INSERT FILE NAME HERE*********************
#*********************************************************************************

file1 = open(filename, "r", encoding="utf-8")

tweet_list = file1.read().splitlines()

file1.close()
```

The code above opens the *.txt* file and stores each line as a separate element of a list. This gives

us a list where each element is a separate tweet.

Task #5: Write code that (a) prints out the number of tweets total in this dataset and (b) prints out a single tweet at random.

Warning: Remember that printing a random tweet means that there is a chance that the tweet displayed may not be school-appropriate. If this happens, just run the code again until the tweet is school appropriate.

In [21]:
```
# Your code here
import random
t_list = tweet_list
length = len(t_list)
num = random.randint(0, length-1)
tweet = t_list[num]
print("The number of tweets tweeted is " + str(length))
print("A random tweet from the hashtag is: " + tweet)
```

```
The number of tweets tweeted is 3897
A random tweet from the hashtag is: Disappointed @BorisJohnson made no mention
of wearing face coverings in public or at work but particularly in crowded plac
es like banks, supermarkets &amp;amp; public transport. Our #Masks4All campaign wil
l go on as this simple measure makes a difference in breaking #COVID19 transmis
sion
```

Performing a Sentiment Analysis

We now have quite a few tweets stored in *tweet_list*. The next step is to do a sentiment analysis on each one and store the data in a new list, *sentiment_list*.

Task #6: Write code that does the following: For each tweet in *tweet_list,* perform a sentiment analysis on the tweet and append the result onto *sentiment_list*.

In [26]:
```
sentiment_list = [] # Get ready to store sentiment analysis results

# Your code here
for tweet in tweet_list:
    t_sentiment = TextBlob(tweet)
    sentiment = t_sentiment.sentiment.polarity
    sentiment_list.append(str(sentiment))
```

```
After you finish, you can test if your algorithm works by running the code
below. Re-run the code until (a) you are convinced that your sentiment analysis
is classifying tweets roughly as expected and (b) the randomly displayed tweet
is school-appropriate.
```

In [27]:
```
import random

total_tweets = len(tweet_list)
random_index = random.randint(0, total_tweets)

print('SENTIMENT = ' + sentiment_list[random_index])
print('TEXT = ' + tweet_list[random_index])
```

```
SENTIMENT = 0.15
TEXT = MEK Iran: No End in Sight for #COVID19 in #Iran with 40,200 Dead and Mor
e Cases Reported  https://t.co/x2AKa8yjin (https://t.co/x2AKa8yjin) #MEK #coron
avirus @USAdarFarsi
```

Analyze the Results

Now we have two lists, *tweet_list* and *sentiment_list* that are storing quite a bit of data:

In [28]:
```
print(len(tweet_list))
print(len(sentiment_list))
```

```
3897
3897
```

Depending on the hashtag you picked, the number above might be pretty large. You can always get even more data by setting up a stream yourself and letting it run for a longer period of time.

Here are the first three elements from these two lists:

In [29]:
```
print(tweet_list[:3])
print(sentiment_list[:3])
```

```
["@JCColtin @nahmias @NYC_DOI @CCRB_NYC @BilldeBlasio @SusanBEdelman How to hol
d court @BilldeBlasio &amp;amp; his #NYPD #criminal #mob's expense for $1,000 &amp;
attorney fees for smiles during #CoronaVirus #COVID19 #pandemic. #MothersDay ht
tps://t.co/cMvRZBEqNr", (https://t.co/cMvRZBEqNr",) "@PrisonPlanet @NoorBinLadi
n @CCPVirusDigest Australian police also don't want people to exercise their ri
ghts 😉  #FakePandemic #COVID19 #covidhoax #CovidNAZI #Australia https://t.co/t
vsh2lGfqP", (https://t.co/tvsh2lGfqP",) "I'm a signatory to this joint-union le
tter in today's @ObserverUK  The role of union health and safety reps represent
atives must be an integral part of the fight against this pandemic.  #COVID19
https://t.co/LqGPuX3lm3"] (https://t.co/LqGPuX3lm3"])
['-0.4', '0.0', '0.0']
```

Instead of compiling these results by hand, let Python do the heavy lifting:

In [30]:
```
from collections import Counter

sentiment_count = Counter(sentiment_list)

print(sentiment_count)
```

```
9': 8, '0.525': 8, '0.2333333333333333': 8, '0.13333333333333333': 7, '0.2666
6666666666666': 7, '-0.041666666666666664': 7, '0.18': 7, '0.75': 7, '-0.0555
5555555555556': 7, '-0.024999999999999994': 7, '0.325': 7, '-0.16590909090909
092': 7, '0.375': 7, '0.625': 7, '-0.19999999999999998': 7, '-0.175': 7, '0.0
625': 7, '0.041666666666666664': 7, '0.07575757575757576': 6, '0.045454545454
54545': 6, '-1.0': 6, '0.08522727272727272': 6, '0.3194444444444444': 6, '-0.
16666666666666666': 6, '-0.1638095238095238': 6, '0.08333333333333333': 6, '-
0.016666666666666666': 6, '0.04999999999999999': 6, '0.12159090909090908': 6,
'-0.024999999999999967': 6, '0.2681818181818182': 6, '0.24242424242424243':
6, '0.15833333333333333': 5, '-0.41785714285714287': 5, '-0.0666666666666666
7': 5, '0.3181818181818182': 5, '-0.07500000000000001': 5, '-0.01728395061728
3952': 5, '0.275': 5, '-0.20000000000000004': 5, '-0.05000000000000002': 5,
'0.05656565656565656': 5, '0.2340659340659341': 5, '0.65': 4, '-0.05555555555
555555': 4, '0.15000000000000002': 4, '-0.6': 4, '-0.15000000000000002': 4,
'0.225': 4, '-0.30000000000000004': 4, '0.08': 4, '-0.052083333333333336': 4,
'0.3454545454545454': 4, '-0.049999999999999996': 4, '0.037500000000000006':
4, '0.13506493506493505': 4, '0.06944444444444443': 4, '0.07777777777777777':
4, '0.46875': 4, '0.475': 4, '-0.10000000000000002': 4, '-0.0500000000000000
1': 4, '0.32285714285714284': 4, '0.0715909090909091': 4, '-0.259259259259259
24': 4, '0.21428571428571427': 4, '0.22888888888888886': 4, '0.43333333333333
```

Depending on your hashtag, you might see many more "positive" values than "negative" values (or vice-versa). You can get a sense of the overall (net) sentiment in your stream by subtracting the number of positive tweets from negative tweets:

In [31]:
```
net = sentiment_count['positive'] - sentiment_count['negative']

current_hashtag = '#' + filename.split(".")[0]

if net > 0:
    print(current_hashtag + ' was a net-positive hashtag during our streaming ses
elif net < 0:
    print(current_hashtag + ' is a net-negative hashtag during our streaming sess
else:
    print(current_hashtag + ' is a net-neutral hashtag during our streaming sessi
```

```
#covid19 is a net-neutral hashtag during our streaming session.
```

Question #7: You just analyzed overall sentiment of tweets using a particular hashtag. Were you surprised by your results or did your result line up with your expectations? Explain.

```
**Your Answer:**  I wasnt really surprised, as during covid19, there would be
negative messages about how bad the pandemic is, but there would be positive
encouragements too.
```

Sentiment Analysis and Machine Learning

You might have noticed some tweets in your data that were clearly "positive" but were incorrectly classified as "neutral." Or you might have seen a tweet that was classified as "positive," but it was written with a sarcastic tone and should have been classified as "negative."

As we highlighed with the Maya Angelou poem, the reality is that sentiment analysis is difficult and can depend heavily on context. Most data scientists will use **machine learning** algorithms to **train** their own **classifier**. All of this vocabulary might sound fancy, but all it means is that perform sentiment analysis by creating your own function that works well for your specific context. You just need a big *.csv* file containing tweets and where someone has manually performed a sentiment analysis. Unfortunately, this can be labor intensive and requires someone to classify thousands of tweets by hand. This *.csv* is your **training data**. Then the **machine learning algorithm** can identify patterns in this *.csv* to perform sentiment analysis on new tweets that it hasn't seen before. If we had more time, we could use this technique to improve TextBlob's results.

Here's an example of a blog post where the author walks the reader through this process: https://www.kaggle.com/ngyptr/python-nltk-sentiment-analysis (https://www.kaggle.com/ngyptr/python-nltk-sentiment-analysis)

Twitter: A Source for Input, a Place for Output

For most people, Twitter is a source of entertainment as well as a place to find relevant/current information, conversation, and debate. That's what social media is all about.

As a programmer, you are probably beginning to also see Twitter as a source of input. But it's also a great place to share your output. We took information from Twitter (as input), so if you created a Twitter developer account this will be your chance to give something back (as output).

First, let's use the data in *sentiment_count* as the labels and sizes for a pie chart:

In [32]:
```
# Data to plot in a pie chart
labels = list(sentiment_count.keys())
sizes = list(sentiment_count.values())

print(labels)
print(sizes)
```

```
['-0.4', '0.0', '0.10416666666666667', '0.1', '0.35', '0.5', '0.15', '-0.7999
999999999999', '-0.02121212121212122', '0.05', '0.11666666666666665', '-0.216
66666666666667', '-0.5', '-0.125', '0.09999999999999999', '-0.760416666666666
6', '0.08125', '0.13333333333333333', '0.8', '1.0', '-0.1875', '0.13999999999
999999', '0.13636363636363635', '0.09333333333333335', '-0.1', '0.125', '0.26
666666666666666', '0.16666666666666666', '0.2572727272727272', '0.3', '0.1500
0000000000008', '-0.5666666666666667', '0.65', '0.25', '0.35714285714285715',
'-0.08333333333333333', '-0.04999999999999999', '-0.0875', '-0.05625000000000
001', '0.18409090909090908', '0.016666666666666673', '-0.022655122655122664',
'0.1125', '-0.05', '0.4666666666666666', '0.19375000000000003', '-0.044444444
44444443', '0.1340909090909091', '0.07575757575757576', '0.0454545454545454
5', '-0.05555555555555555', '0.06666666666666667', '0.03333333333333333', '-
0.11666666666666667', '0.022222222222222213', '-1.0', '0.15000000000000002',
'0.0125', '0.2', '0.08522727272727272', '0.6', '0.4333333333333333', '0.15833
333333333333', '0.16818181818181818', '-0.09999999999999998', '-0.46666666666
66666', '-0.041666666666666664', '0.13727272727272727', '0.3194444444444444',
'0.047979797979797977', '-0.35000000000000003', '-0.29', '0.18', '-0.025',
'0.12045454545454547', '0.05000000000000001', '-0.3', '0.04444444444444445',
'0.41363636363636364', '0.12878787878787878', '0.4166666666666667', '-0.13333
```

Now plot a pie chart. Look for a line of code below that looks unfamiliar and try to determine what it does.

In [33]:
```
import matplotlib.pyplot as plt

# Reminder: plt.pie(data array, labels array, percent, shadow, starting angle)
plt.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Tweet Sentiment for ' + current_hashtag)
plt.axis('equal')
plt.show()
plt.savefig('pie.png') # Notice this line of code! What do you think it does?
                       # Hint: Check your Twitter Bot Part 4 folder :)
```

```
<Figure size 640x480 with 1 Axes>
```

Question #8: What does the *plt.savefig('pie.png')* do? Be specific.

```
**Your Answer:** It takes the generated pie chart, and saves it as a png file
```

If you decided to set up a a Twitter developer account, the code below would send out this pie chart as a tweet. If you did not set this up, the code will instead throw an error:

In [34]:
```
my_picture = 'pie.png'

my_status = "Sentiment analysis for the hashtag " + current_hashtag

# If you set up a Twitter account, you could remove the comment below and run th
#api.update_with_media(my_picture,my_status)
```

Optional Task #7: Create another sentiment analysis pie chart for one of the other hashtags listed above.

- While this task is 100% optional, it is highly recommended.
- If you analyze a second hashtag, this will force you to learn what code is necessary to process, analyze and visualize a hashtag.

In [ ]:
```
# Your code here
```

Brainstorm

Question #9: Describe how you might incorporate sentiment analysis into a project that you could work on over the summer. This could be a personal project that you could pursue just for fun, a project that you could complete as part of a job/internship, or a project that you could complete in order to launch your own business.

Your Answer: i COULD USE THIS TO SEE SPECIFIC PEOPLES MOODS, ADN POST IT ONLINE SO POEPLE CAN KNOW THE OVERALL MOOD OF PEOPLE, AND REACT ACCORDINGLY

Final Thoughts

Thank you all for the strength, resolve, and dedication you showed during the e-learning phase of this course. I'd like to ask that you finish this course by reflecting on just how much you learned this year. You've learned to write code in JavaScript and Python. You've learned about the Internet, cybersecurity, and encryption. You've learned about data, metadata, data science, web scraping, APIs, and more.

You've learned to be a creative and ethical. You've learned to become a problem solver. **You are a programmer.**

Here's what I would encourage you to do now that you've reached the end of this course:

- Back up all of your Python notebooks to your *personal* Google Drive
  - You would be shocked by the number of emails I receive from former students who want copies of their old notebooks to help them prepare for a college course, an internship, or even a new job
  - It is difficult to honor these requests, especially when I get emails from students who took the course many years ago
  - But if you back up your work now, you'll have access to these materials if (when) you need them again in the future!

- This summer, I would encourage you to create a project of your own:
  - Use the programming skills and cutting-edge tools you learned in this course
  - The **best way to improve as a programmer is to create your own projects!**
  - There may have been code in some of the Jupyter notebooks that you used but did not understand
    - Once you create a project of your own, you'll begin to understand any of that code that you didn't before!

In [ ]: