DS3002 – Data Project 2
25 points Due 5/10/2021

The goal of the second data project, building upon the first project, is to further demonstrate (1) an understanding of and (2) competence creating and implementing basic data science systems such as pipelines, scripts, data transformations, containers, APIs, databases and cloud services. Submit your project in the "quizzes" section of our Collab site.

Data Projects 1 and 4 must be done individually. Projects 2 and 3 can be done in pairs or individually.

Select a project:

1. Data ingestion & analysis – individual project
2. Twitter bot – individual or pairs
3. Discord bot – individual or pairs
4. Raspberry Pi sensor – individual project

Data Ingestion & Analysis

Deliverable: Write and deploy a process that executes *exactly* once every minute, retrieving data from a remote API (provided for you) and write all retrieved values to a database for 60 minutes. Using code-based data analysis techniques against the database, try to (i) describe any patterns or changes in the data over time; and (ii) explain the logic of these changes.

The remote data API can be found here:
https://4feaquhyai.execute-api.us-east-1.amazonaws.com/api/pi

Benchmarks:

1. Your solution must execute precisely once per minute at the same time each minute. (You should not use a sleep 60 command to simply "wait" between executions or other methods that will drift over time.) Therefore, your solution must be designed carefully.
2. Your solution must run for exactly one hour, starting at 00 minutes and finishing at 59 minutes. The API is available 24/7 for such testing.
3. Your solution will retrieve all data fields from an API and write them to the database of your choice and design – relational or NoSQL.
4. Submit all code in a standalone GitHub repository in your account.
5. Your analysis should look at the relationship between all data fields and their changes over time. In a brief statement, describe any changes or patterns you observe, and propose an explanation for them. Include this in your GitHub repository.
6. You should also provide text output or a screenshot of your database table verifying consistent execution of your code each minute. Include this in your Github repository.

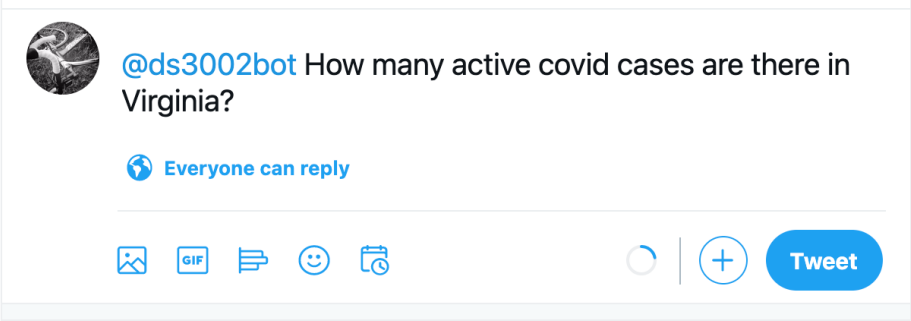
Grading:

☐ Successful deployment – 10 points.
☐ Functionality that meets all benchmarks – 12 points.
☐ Documentation – Describe your process, code, deployment strategy – 3 points.

Twitter Bot

Deliverable: Write an interactive, Twitter-based bot account that can receive and parse Tweets sent to it and returns a data-driven reply. This will involve creating a Twitter Developer account (free), and setting up your own Python-based API that is connected to that account. While responses or actions can be triggered by a variety of Twitter actions (DMs, replies, etc.) this project requires a reply only to basic tweets sent to your bot. Replies should relate to the message sent to your bot, and should integrate at least one external data source (an API or database). The API backing your bot could be a container running in Lightsail, on EC2, the API Gateway + Lambda, or another solution of your choice.

Your bot does not have to be "serious" – for instance it could play a game or quiz, but the seriousness should be found in how your application handles data, performs remote data retrievals and processes user requests.



Blog posts and documentation can help you get started:

- https://realpython.com/twitter-bot-python-tweepy/
- https://developer.twitter.com/

Machine learning and NLP would be employed in a deeper bot model, but those are not required here. But explore those options if you like!

Benchmarks:

1. Your bot should reply promptly with an intelligent response or an informative error message.
2. Your bot should recognize a "help" or "info" message and return user instructions.
3. Your bot should integrate with at least one external data source that you can document and describe. This could be a database system or API.
4. Submit all code in a standalone GitHub repository in your account.
5. Submit the twitter account handle for testing/grading in your repo.

Grading:

☐ Successful deployment of your API to Lightsail, an EC2 instance, or AWS Lambda + API Gateway – 10 points
☐ Functionality that meets all benchmarks – 10 points
☐ Creativity / Innovation / Quality – 2 points
☐ Documentation – Describe how to use the bot and the elements that make it operational. – 3 points.

Discord Bot

Deliverable: Create and publish an interactive Discord bot in a test server (provided for you by the instructor). Unlike a Twitter bot that simply parses and replies to messages sent to it, Discord bots can have much more functionality. For instance, they can have a wide variety of commands that users can invoke in channels as well as in direct messages with the bot.

Like a Twitter bot, a Discord bot must be backed by a Python-backed application that runs actively with an open connection to the Discord activity API and is invoked by bot commands and messages. You must write and publish this app, and it should connect to at least one external data source such as an external API or a database for some of its possible interactions.

Unlike a Discord "webhook" (from Data Project #1), bots require further authentication, and must be added to a server to implement.

Your bot does not have to be "serious" – for instance it could play a game or quiz, but the seriousness will be found in how your application handles data, performs remote retrieval and processing of requests and replies. See the "ChampBot" in our server for examples.

If you choose this option, join this server [https://discord.gg/cDrvtEh4T6] and you will be granted administrator access for setup/testing, etc.

There are many good resources to show you how to set up a Discord bot:

- https://realpython.com/how-to-make-a-discord-bot-python/
- https://www.freecodecamp.org/news/create-a-discord-bot-with-python/
- https://discord.com/developers/docs/intro
- https://top.gg/
- Complete multi-part video series: https://www.youtube.com/playlist?list=PLW3GfRiBCHOhfVoiDZpSz8SM_HybXRPzZ

Benchmarks:

1. Your bot should recognize a "help" message and reply with user instructions.
2. Your bot should provide at least three commands or functions.
3. Your bot should reply promptly with an intelligent response or an informative error message.
4. Your bot should integrate with at least one external data source that you can document and describe. This could be a database system or API.
5. Submit all code in a standalone GitHub repository in your account.

Grading:

☐ Successful build of the bot and API solution using Lightsail or EC2 – 10 points
☐ Functionality that meets all benchmarks – 10 points
☐ Creativity / Innovation / Quality – 2 points
☐ Documentation – Describes how to use the bot and the elements that make it operational – 3 points

Raspberry Pi Sensor

Deliverable: If you are up for a hardware-software challenge, this project requires that you purchase and build a "server" of your own using a Raspberry Pi and sensor. There are plenty of options, listed below, but I would recommend a Pi 3 or 4 for beginners, since they accept standard HDMI monitor cables and standard USB keyboards/mice. Be sure that your Pi can connect via WiFi, and you can buy either a blank memory card and image it yourself, or many starter kits come with the OS flashed to the card for you. If you have always wanted to have a Pi and have some cable adapters, the newer Pi Zero is extremely small.

As for sensors, there is a wide variety available: for light, motion, humidity, temperature, GPS, and so on. Many sensors require a pinboard and cables to attach – unless you're very handy you do not want to choose the options that require soldering!

IMPORTANT NOTE: If you select this project and order your equipment from Amazon or Adafruit it can arrive in 2-4 days. But you will need to order QUICKLY!

The project goal is that you deploy a Pi with a sensor connected, and make that data available as historical, compiled log data in a database or object storage [i.e. MongoDB, DynamoDB, or S3].

Raspberry Pi resources:

- https://www.adafruit.com/
- https://www.amazon.com/s?k=raspberrry+pi
- https://tutorials-raspberrypi.com/
- https://www.initialstate.com/

Benchmarks:

1. Set up and deploy a Pi – this includes setting up the OS, root user account, WiFi, etc.
2. Install your sensor and any software dependencies.
3. Set up a cron task that executes a Python3 script at regular intervals (every minute, or hour, etc.). This script should obtain the current sensor data and publish it to a database/storage of your choice, including the datetime.
4. Present a snapshot of your collected data as part of your final submission. Include any source code with that in a GitHub repository.

Grading:

☐ Successful build of the solution – 10 points
☐ Functionality that meets all benchmarks – 10 points
☐ Creativity / Innovation / Quality – 2 points
☐ Documentation – Describes your process as well as any obstacles, challenges and solutions – 3 points.

This is a great project choice but, unfortunately, is not one that can easily be supported by the instructor or TA during COVID. Therefore, you are on your own to work out the hardware, sensor, and software. However, there are volumes of great resources online to help with this – so go for it if you're feeling

brave, it's fun and gratifying. Initial State is a great company for integrating Raspberry Pi data with pre-made dashboards and visualizations – they also have free student accounts available.

Publicly-available datasets:

- https://www.kaggle.com/datasets
- https://data.world/
- https://www.data.gov/
- https://opendata.charlottesville.org/

Publicly-available APIs:

- https://docs.github.com/en/rest
- https://developer.twitter.com/en/docs/twitter-api
- HUGE LIST: https://github.com/public-apis/public-apis