

SCHOOL *of* BUSINESS AND TECHNOLOGY

Department of Engineering and Aviation Sciences

**Timothy Riley Jr**

Advisor: Dr. Zhang

April, 5 , 2020

Satellite image analyzing app for plant health

By

Timothy Riley Jr

Submitted to the Department of Engineering and Aviation Sciences in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the

UNIVERSITY OF MARYLAND EASTERN SHORE

Date

The author hereby grants the University of Maryland Eastern Shore permission to reproduce and distribute publicly, paper and or electronic copies of this document in whole or in parts.

Authors Timothy Riley Jr

Signature

Date

Department of Engineering and Aviation Sciences

Certified by

**List of Contents**

[**List of Contents 2**](#_30j0zll)

[**List of Figures 4**](#_1fob9te)

[**List of Tables 5**](#_3znysh7)

[**Abstract 6**](#_tyjcwt)

[**1.**](#_3dy6vkm) **Introduction 7**

[**1.1**](#_1t3h5sf) **Background/Motivation 7**

[**1.2**](#_4d34og8) **Objective 7**

[**1.3**](#_2s8eyo1) **Design Requirements 7**

[**1.4**](#_17dp8vu) **Design Constraints 7**

[**1.5**](#_3rdcrjn) **Design Methods 7**

[**2.**](#_26in1rg) **Project Description 9**

[**2.1**](#_lnxbz9) **System Description 9**

[**2.2**](#_35nkun2) **System Diagram 9**

[**2.3**](#_44sinio) **System Functions 9**

[**3.**](#_2jxsxqh) **Implementation Plan 11**

[**3.1**](#_z337ya) **Tasks 11**

[**3.2**](#_3j2qqm3) **Team Organization 12**

[3.2.1.](#_41mghml) Responsibility of Team Member 1. 12

[3.2.2.](#_2grqrue) Responsibility of Team Member 2. 12

[**3.3**](#_1y810tw) **Timeline/Milestones/Delivery Plan 12**

[**4.**](#_2xcytpi) **Implementation 13**

[**4.1**](#_1ci93xb) **Implementation of Task 1. 13**

[4.1.1.](#_3whwml4) Implementation of Subtask 1.1 13

[**4.2**](#_2bn6wsx) **Implementation of Task 1. 13**

[**5.**](#_qsh70q) **Conclusion (Discussion and Future Plans) 14**

[**Acknowledgment 15**](#_3as4poj)

[**Appendix 16**](#_1pxezwc)

[**A.**](#_49x2ik5) **Component Specs 16**

[1.](#_2p2csry) Specs of 16

[2.](#_147n2zr) Specs of 16

[**B.**](#_3o7alnk) **Source Code. 16**

[1.](#_23ckvvd) Source Code of 16

[2.](#_ihv636) Source Code of 16

[**REFERENCES 17**](#_32hioqz)

**List of Figures**

Figure 1[. System Diagram 9](#_4i7ojhp)

Figure 2.

**List of Tables**

Table 1[. System Diagram](#_4i7ojhp) 7

Table 2.

**Abstract**

By the end of the project, summarize the project into short text and put here.

1. **Introduction**

This project will be the development of a web application in Google Earth Engine that performs plant health predictions on satellite images. The web application will implement a trained machine learning algorithm that predicts the biomass and percent of nitrogen from a satellite image of plant life. The algorithms predictions will also be converted into a raster map by the application and presented to the user interactively.

## Background/Motivation

Modern agriculture today operates far differently than that of decades before due to the various advancements in technology like sensors, devices, machines, and information technology. Today’s agriculture routinely uses sophisticated technologies such as robots, temperature and moisture sensors, aerial images, and GPS technology. These advanced devices and precision agriculture and robotic systems allow businesses to be more profitable, efficient, safer, and more environmentally friendly.

With the rise in the application of artificial intelligence technologies in multiple industries globally, it is beginning to make its mark in the agriculture industry as well. Farms produce hundreds of thousands of data points on the ground daily. With the help of AI, farmers can now analyze a variety of things in real time such as weather conditions, temperature, water usage or soil conditions collected from their farm to better inform their decisions. For example, AI technologies help farmers optimize planning to generate more bountiful yields by determining crop choices, the best hybrid seed choices and resource utilization. AI systems are also helping to improve harvest quality and accuracy -- what is known as precision agriculture. Precision agriculture uses AI technology to aid in detecting diseases in plants, pests, and poor plant nutrition on farms. AI sensors can detect and target weeds and then decide which herbicides to apply within the right buffer zone. This helps to prevent over application of herbicides and excessive toxins that find their way in our food. All approaches to AI in agriculture have the same overarching goal, to quantify and predict crop health for the farmers benefit. Plant health can be quantified in different ways like, lack of diseases, crop growth over time, or color. A very common way to determine a crops health is to look at its biomass content and its percentage of nitrogen in the soil…...

[ways AI is influencing agriculture]

[influence of cover crops in primary crop production]

[impact of biomass and nitrogen in plant life ]

[scientific impact of estimating nitrogen and biomass of cover crops with satellite imaging]

Biomass and nitrogen content are essential factors to the growth and sustainability of plant life. Biomass is the weight of living plant material contained above and below a unit of ground surface area. Nitrogen is a key component of chlorophyll and directly affects photosynthesis. Monitoring and assessing the content of both factors is critical in ensuring the wellbeing of the crop. There is data supporting that plant health can be measured from advanced imaging such as near-infrared reflectivity.

With the use of satellite imaging...

Computer vision is defined as how a machine/computer can be made to gain high-level understanding from digital images or videos. This is used in a variety of devices for more complex tasks from tracking a ball to powering self driving cars. The implementation of this technology into various industries is growing exponentially. Not only in transportation or manufacturing but in agriculture as well.

## Objective

To deploy a web application through Google Earth Engines (GEE) API that analyzes satellite images and produces raster mapping of plant health, in the form of biomass content and percent nitrogen predictions.

## Design Requirements

1. Perform pixel classification of biomass & nitrogen content to obtain predictions with at least 90% accuracy
2. Access satellite images from coordinate inputs
3. Display results of biomass and nitrogen content as a raster map layer on top in the Earth Engine API

## Design Constraints

1. Web application must be written in Google earth engine.
2. Accessible data is determined by images taken by the constellation of Sentinel satellites.
3. Pixel resolution of sentinel images is accurate down to 10 m by 10 m area.

## Design Method (Approach)

The first step to accomplish design requirements is to analyze and organize the data from satellite images and excel spreadsheets. The Second step is to preprocess the data into training and validation datasets in Tensorflow. The third step is to build and train a machine learning model in Tensorflow with the dataset. The fourth step is to improve the accuracy of the model. The fifth step is to write the source code for an Alpha version of the Google Earth Engine app. The sixth step is to add backend improvements to the application. The seventh step is to improve the User interface code of the application.

## Standards

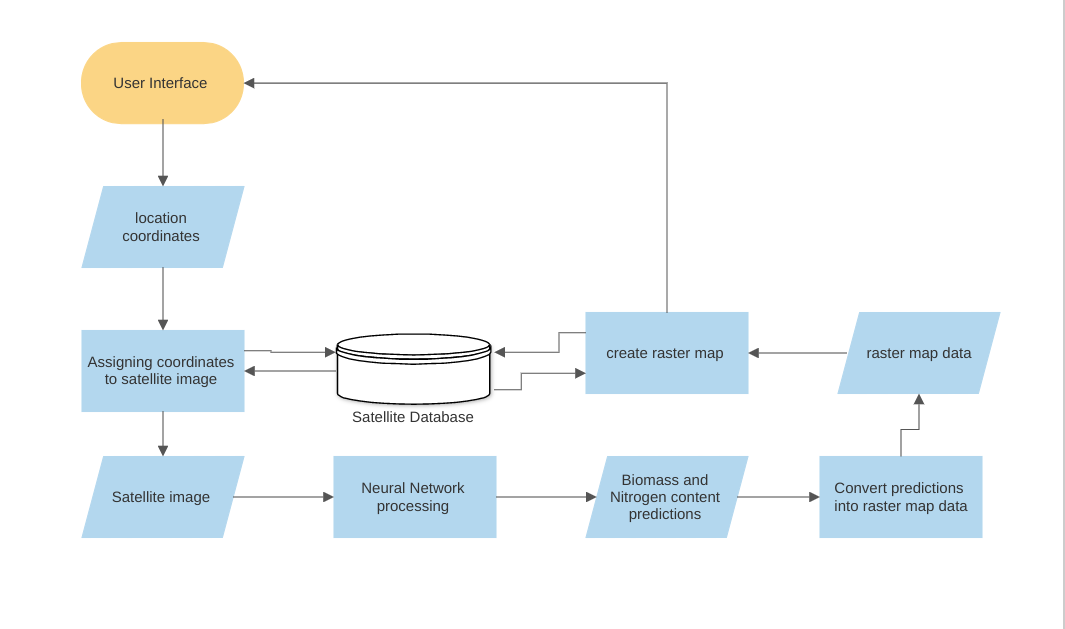
N/A

1. **Project Description**

## System Description

The system is a Web application accessible via web browser and served through Google’s Cloud Computing services. The backend of the application including its neural network will be written in python and the front end User Interface (UI) will be written in Javascript. The User Interface for the application is accessed through Google Earth Engine website. The User interface will show the most recent imaging from the Sentinel II satellite and will have a prompt asking for a location. Once requested the application will capture an image of that location then utilize a neural network to process the image to predict the Biomass content and nitrogen % of every pixel of that image. That vector data is transferred to raster mapping and presented to the UI for the user to evaluate. Figure [1] shows the system diagram for the flow of the application.

## System Diagram



1. System Diagram

## System Functions

1. When the user goes to the link of the Google Earth Engine app, satellite imaging is shown and can be interacted with to view different areas of the world.
2. Application prompt is displayed on the user interface (UI) asking for location coordinates
3. Application prompts the user to specify the dates of images intended to view
4. When coordinates are given to the prompt the application accesses the Sentinel-II satellite image database and directs the UI to show the corresponding coordinates submitted.
5. ‘run’ button is used to begin prediction processing.
6. Application processes the satellite images of the location displayed then utilizes the machine learning algorithm embedded in the code to produce predictions on plant health.
   1. Application breaks down the image pixel by pixel and performs prediction analysis on each pixel to obtain values of biomass and percent nitrogen.
7. The Application takes those prediction values and converts them into mappable raster data
8. The raster map data is associated with the pixel locations of the original image and a raster map layer is produced on the output map
9. The UI is now updated and prompting the raster map data and legend where the areas of good plant health and bad plant health were predicted to be.
10. The UI also shows options to save/download dataset images of the original satellite image and the raster map image.
11. **Implementation Plan**

## Tasks

* Task 1. Preprocessing the Dataset
  + Subtask 1. isolate necessary information from satellite
  + Subtask 2. isolate vital information from field site data
  + Subtask 3. Correlate quadrat data to pixel location in satellite images
  + Subtask 4. Build Dataset input pipeline
* Task 2. Produce Machine learning Network
  + Subtask 1. Build Network structure & begin initial training.
  + Subtask 2. Collect results, revise and retrain the network.
* Task 3. Develop Alpha-Application in GEE
  + Subtask 1. Build UI that accesses Sentinel - II satellite images
  + Subtask 2. Add coordinate location functionality
  + Subtask 3. Add image capture functionality
* Task 4. Backend Development
  + Subtask 1. Preprocess the captured image for compatibility to network input
  + Subtask 2. Integrate the trained neural net model into the GEE API
  + Subtask 3. Write code to associate prediction output values with satellite image pixel locations
* Task 5. Frontend
  + Subtask 1. Convert Biomass and Nitrogen % predictions to Raster Map data values
  + Subtask 2. Write code to Draw Raster map over satellite image
  + Subtask 3. Build User Interface for interaction with the Raster map
* Task 6. Testing and Refinement
  + Subtask 1. Test the operability of the application and its ease of use
  + Subtask 2. Refine the Apps use and workflow

## Team Organization

Timothy Riley is the sole team member responsible for all tasks and subtasks in completing this project.

## Timeline/Milestones/Delivery Plan

1. **Project Timeline and Delivery Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Task | Comments | Responsibility |
| week 1 | Start task 1.1 | read and extract satellite images of field locations |  |
| week 2 | Start task 1.2 | organize biomass and percentage nitrogen from field level data |  |
| week 3 | Start 1.3 | correlate field level data to pixel locations in satellite image |  |
| week 4 | Finish 1.2 |  |  |
| week 5 | Complete 1.3 | Build Dataset input pipeline |  |
| week 6 | Complete 2.1 | Build ML algorithm structure & begin training |  |
| week 7 | Start 2.2 | Collect results from initial training then start revision process [5 weeks needed] |  |
| week 8 | Continue 2.2 | revise structure or parameters of the algorithm |  |
| week 9 | Continue 2.2 | initiate training again and repeat process |  |
| week 10 | Continue 2.2 |  |  |
| week 11 | Continue 2.2 | achieved accuracy of 90% is ideal |  |
| week 12 | Start 5.3  Complete 3.1 | write Google Earth Engine app that can access the Sentinel II satellite database interactively |  |
| week 13 | Start 3.2, 3.3 | add functionality to capture image of a location when coordinates are given |  |
| week 14 | Complete 3.2, 3.3 |  |  |
| week 15 | Start 4.1, 4.3, 4.2 | write the code that breaks down the captured image into a pixel array |  |
| week 16 | Finish 4.1, 4.3 | Associate each pixel with a value representing its location |  |
| week 17 | Continue 4.2 | integrate Machine Learning algorithm into the App code (takes pixel as input and outputs prediction values) |  |
| week 18 | Finish 4.2  start 5.1, 5.2 | Convert prediction values into mappable raster data |  |
| week 19 | continue 5.2 | use pixel location values and raster mp data to draw raster map over the captured satellite image |  |
| week 20 | Continue 5.3 | add more user interface features to view and download the raster map image and data results from the algorithm |  |
| week 21 | Finish 5.3 | perform final user testing then deploy application to Google Earth Engine platform, finalize system and documentation |  |
| week 22 | Task 6 | Analyze application performance and conduct final redesigns |  |

1. **Implementation**

The pages that follow will highlight the processes completed, tools used, and skills gained for proper execution of the tasks at hand defined in this project's scope. The major services and tools used and referred to in the task implementations as as follows:

* Google Earth Engine: a cloud-based platform for planetary-scale environmental data analysis. [<https://developers.google.com/earth-engine>]
* Google Collaboratory:
* Microsoft Excel:
* Tensorflow:
* Python
* Javascript:

Completion of the Earth Engine plant health prediction app is dependent upon 5 major technical areas working together; the satellite image processing interpretation, the field level data processing interpretation, the machine learning algorithm, the Earth Engine application User Interface (UI) and the Earth Engine javascript back end code. Work on the first 4 areas is being progressed separately and will be added to the 5th area systematically.

## Implementation of Task 1.

Task 1 is vital to the training of the machine learning model because it provides the algorithm with a clear and informative dataset for the model to learn from. There are two parts to the dataset used for training, the satellite images and the plant health metrics (quantifying Biomass content and percentage of nitrogen). The dataset information was obtained from USDA operatives conducting research on quantifying cover crop abundance in various fields. The data was sent to me having this initial structure:

plant health metric data:

5 field sites

3 treatments per site

8 quadrats per treatment

-data sampled twice for each quadrat

-each quadrat is 0.5 m by 0.5 m

- data sampled is of biomass content and % nitrogen taken from soil

satellite imagery data:

5 field sites

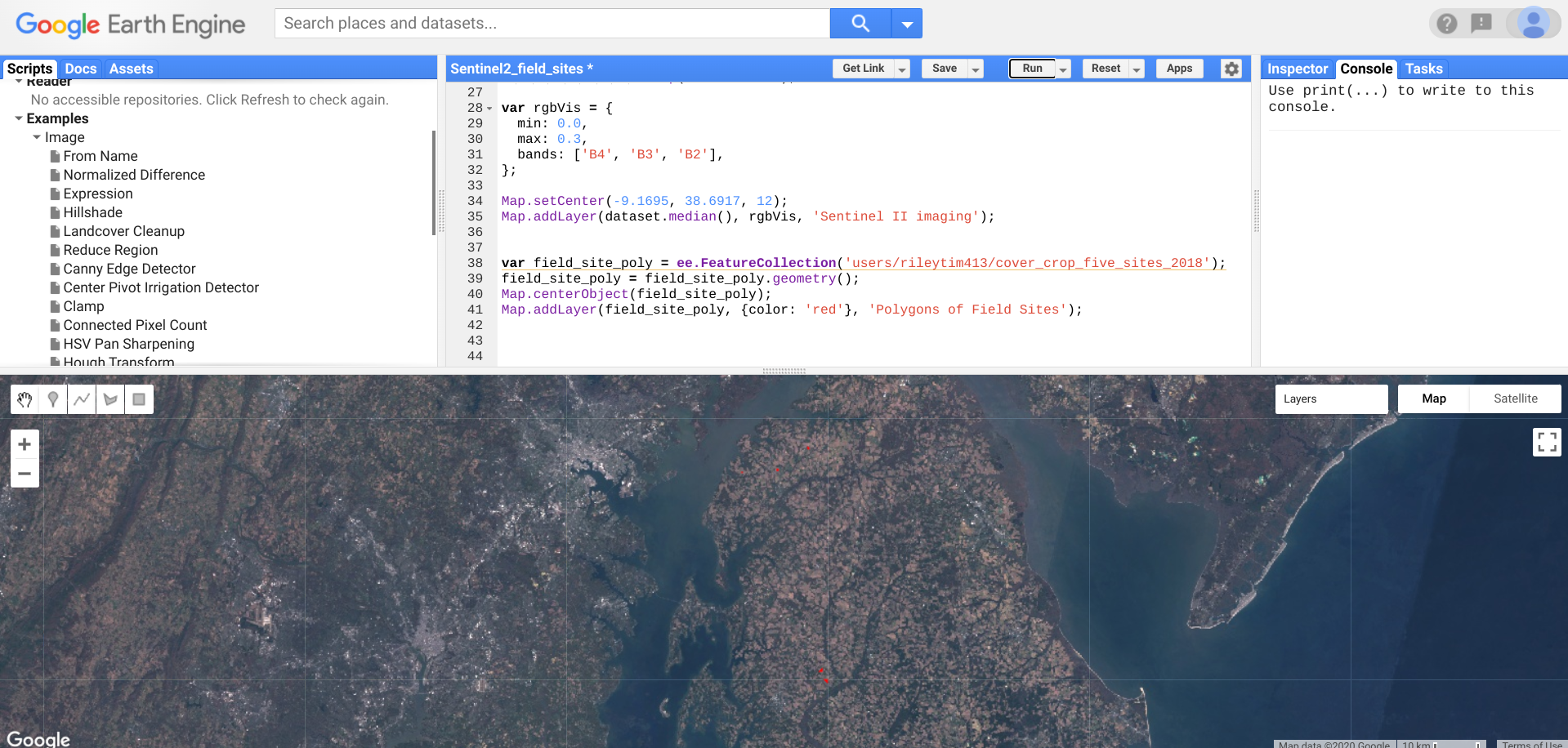
7 field geometric outlines

pixels per polygon to be determined

The data is taken from multiple field sites in Maryland. The satellite images will be analyzed in Google Earth Engines API which has access to the Sentinel -2 surface reflectance image database. The sentinel -2 satellite is

### Implementation of Subtask 1.1

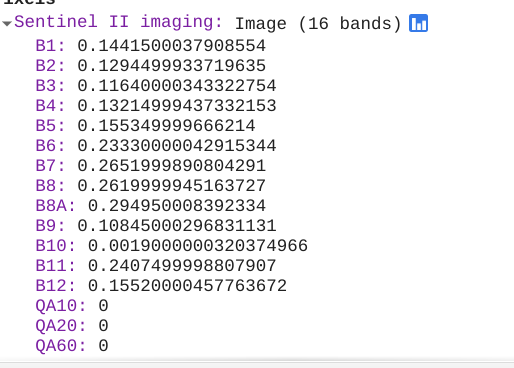
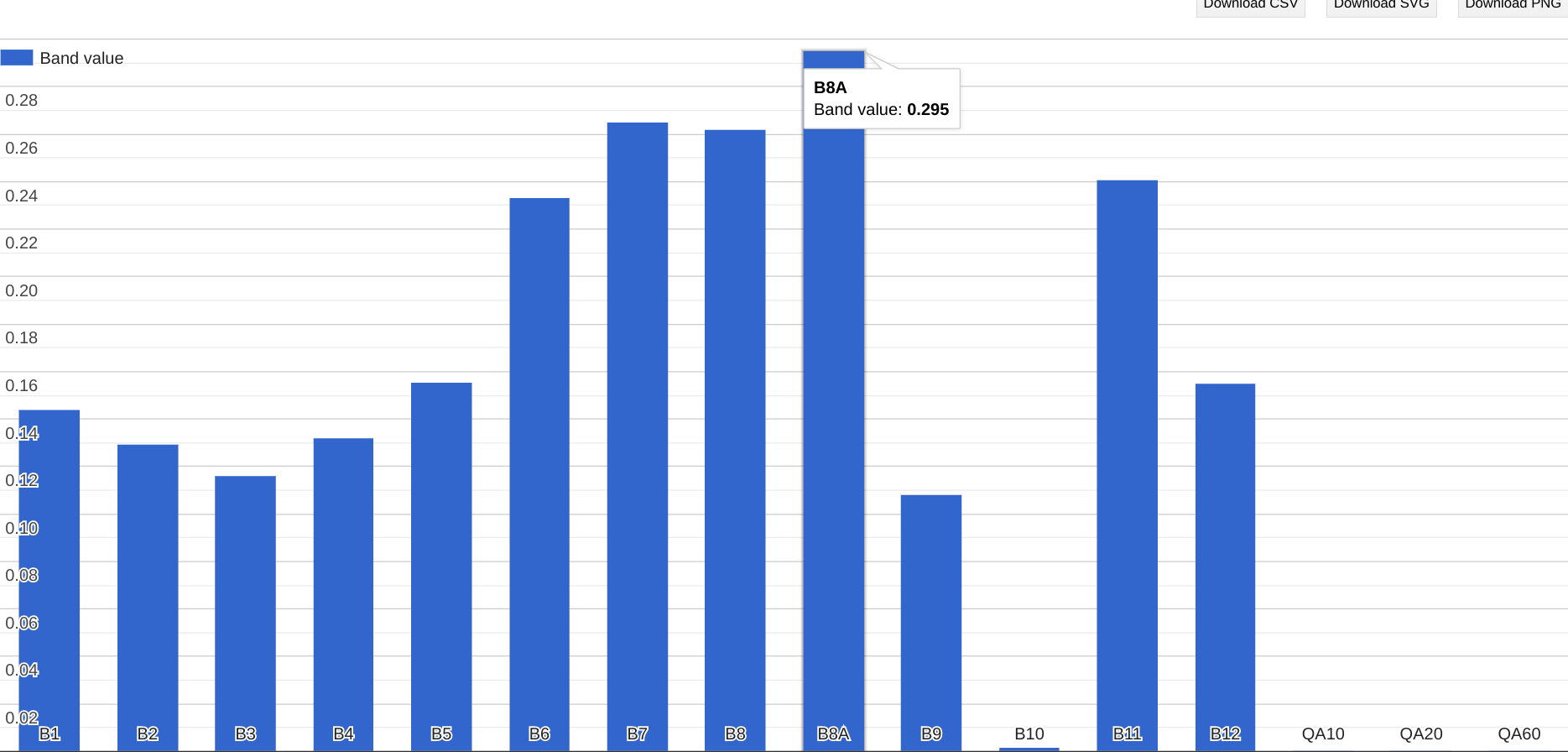
the open source coding platform Earth Engine allows for free extensive use and manipulation of satellite imagery from petabytes of satellite image datasets all accessible from their remote databases. The API for the online code editor is in Javascript programming language. The obtained dataset for the satellite imagery is in the form of a featurecollection which holds the boundary coordinates of 7 polygons representing the location and area of 7 fields from 5 research sites ( some sites have multiple bounded areas). In order to view the satellite images specific to multiple field sites, a file of assets that hold polygons corresponding to the field site coordinates was uploaded to the Earth Engine console using the following code:



The result produced is zoomed in on the Kennyville site in the map below. The red polygons represent the geometry of the field sites where the data for the biomass and percentage of nitrogen was taken from quadrats on the fields:



data in Earth Engine is represented in two main ways, features or images. A collection of feature data was used to draw polygons of the field site locations as a layer in the map. Images in Earth Engine are a stack of Georeferenced bands each with their own mask projection and resolution. the images are represented in arrays of pixels that each hold the band values for the location the pixel represents. The Sentinel-2 surface reflectance database consists of 16 band images. shown below is an example of the band values for a specific pixel manually inspected in the Earth Engine code editor (in both bar graph and list format ):



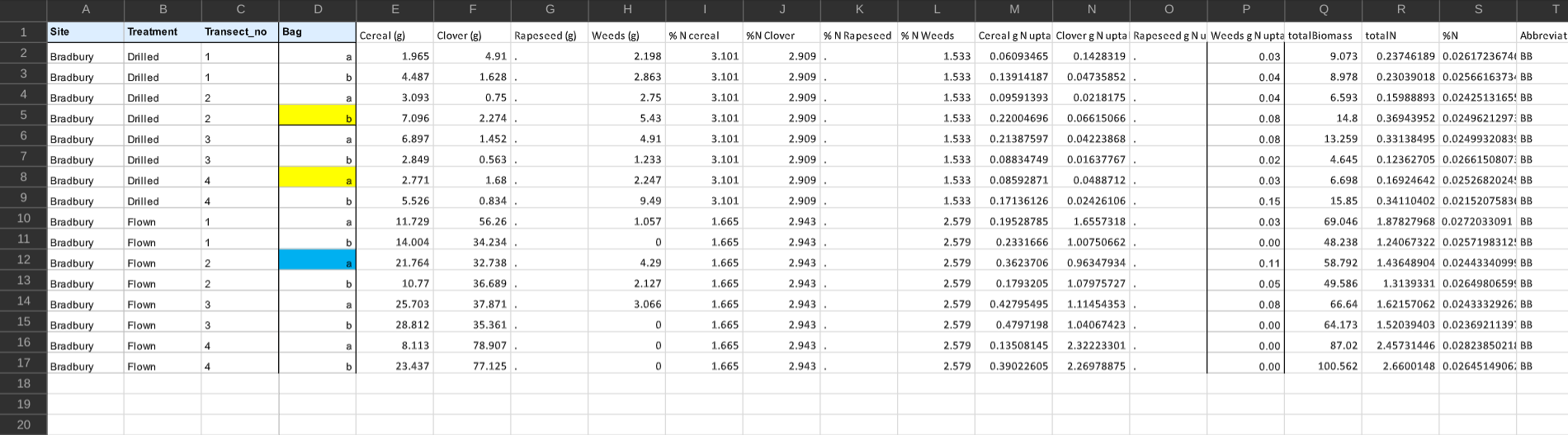
In this project the image pixels that are located within the polygonal boundaries were extracted into a collection so that the band value at each pixel is collected. The function image.clip is able to produce an image from feature (boundary) data. When applied the results are as expected, a third layer on the map that holds Image band values only within the locations of the 7 polygons.

### 4.1.2 Implementation of Subtask 1.2

### 

The second half of the data needed for training a machine learning model is the label data or in this case the values of the biomass and % of nitrogen taken at ground level at the specified field sites. the original data was obtained from the field in the form of quadrats as shown in figure[\_\_] The quadrats dimensions are 1m by 1m

The data from the quadrats were recorded into excel files and organized like in the sample [figure\_\_\_\_].



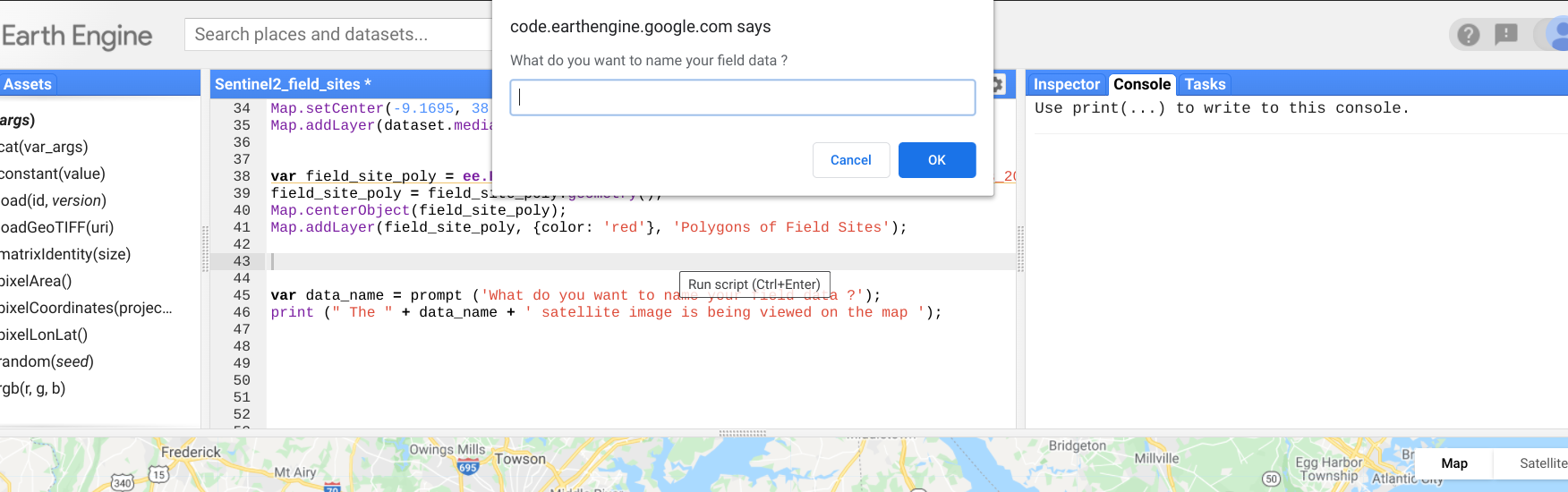
Figure[\_\_\_] shows that there were many values obtained from the analyzed quadrats but most of which were focused on seeding rates of the different cover crop species. The main pieces of data vital to this task are the ‘totalBiomass’ , ‘totalN’ , and ‘%N’. Data in excel format can be extracted and interpreted many ways but an efficient tool discovered was the **openpyxl**  library which manipulated excel data in python which is ideal when trying to use this data with a tensorflow model. The issue that arose in this task was that the quadrat data wasnt scalable enough to correlate to the much larger satellite image. The sentinel-2 images obtain band data down to 10m x 10m pixels which still allows for a large room of misinterpretation of field data. The researchers at USDA are currently in the process of upscaling this field level data to correlate and will be completed within a couple months.

## Implementation of Task 3.

Task 3 is the beginning of writing the code for the app functionality. The app will be written and published within the Earth Engine Javascript API. The most basic functionality of the application are the objectives of task 3, which will be the prompts from the app to the user. First asking for a location coordinate and to use that input to display imaging from the Sentinel-2 surface reflectance database of the location. Then asking for confirmation to use the location displayed to be captured and analyzed to produce plant health predictions.

***4.2.1*** ***Implementation of Subtask 3.1***

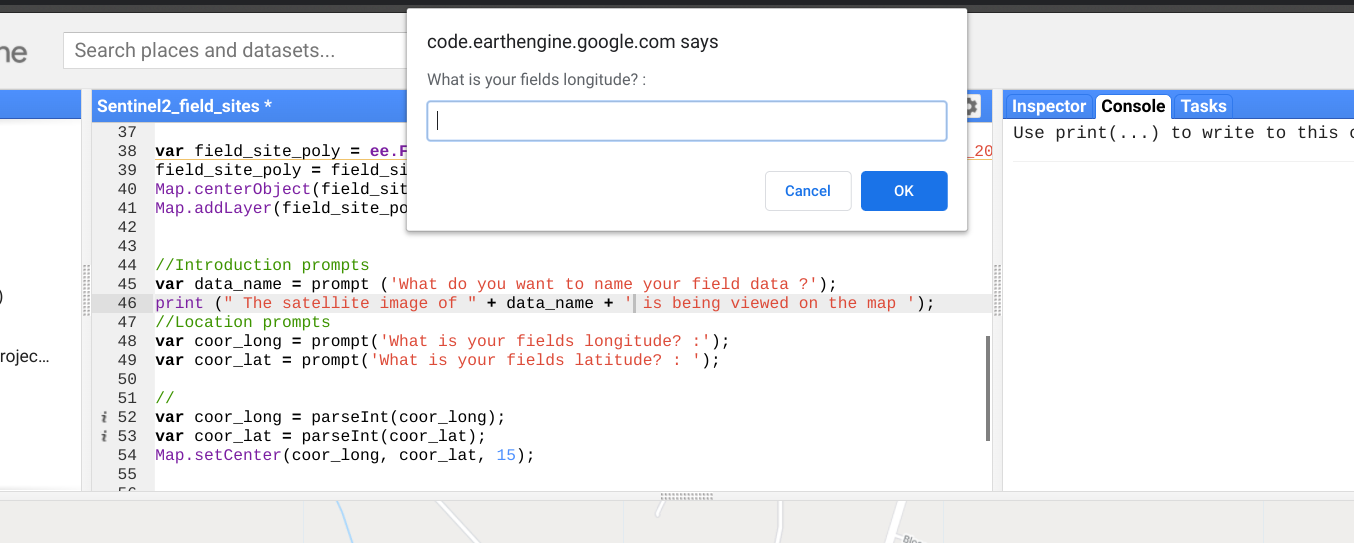
The earth engine javascript API comes equipped with libraries of functions that perform native Javascript actions as well as functions to analyze, interpret and process geospatial raster and vector data. To accomplish the objective of task 3.1 (initiating the applications user interface) the native Javascript libraries were at play along with a specialized library built towards providing App style interaction (ee.ui). The original satellite map of the code editor is actually a default output of the ui.() library and when the functions are called upon to perform further interface related actions they are printed on top of the map area of the code editor. To begin the task of connecting the user to the application an action for the script to ask the user for an input was developed by using the prompt function was written. When used in the code editor the result is shown in figure[\_\_\_]



This is just the basis for the prompts that the application will display to the user. Other prompts produced will be for coordinates, algorithm activation, and raster map exporting. The ee.ui library of functions was useful in adding another element to the cohesion of the map output. Using the maplinker() function and some other initializations outlined in figure [\_\_\_\_] the result are 4 maps in 4 quadrants that zoom and pan in synchronization, shown in figure [\_\_\_\_]

***4.2.2*** ***Implementation of Subtask 3.2***

The first functionality of the application would be for the user to specify the location that they would like to be shown on the map. When provided with longitude and latitude coordinates the script uses the MapCenter function to show the field location in the console map.



The next step was to add this functionality as a UI widget where latitude and longitude have seperate input areas but perform as before and set the map to center on that coordinate. The script and result can be seen be

***4.2.3*** ***Implementation of Subtask 3.3***

Instead of doing predictions/processing on still images and outputting

1. **Conclusion (Discussion and Future Plans)**

By the end of the project, conclude the project and your learning experience.

**Acknowledgment**

If you get help or support from someone else (besides the team member and the advisor) and want to show your appreciation, put here (**do not include the advisor**).

**Appendix**

You can put reference info here, including: i) specs of components used in the system, ii) source code (must be here but not in the body text), iii) CAD figures, etc.

1. **Component Specs**
2. ***Specs of Arduino Due***

...

1. ***Specs of Raspberry Pi***

…

1. **Source Code.**
2. ***Source Code of Graphic User Interface***

…

1. ***Source Code of Robotic Arm***

…

**REFERENCES**

[1] <https://nifa.usda.gov/topic/agriculture-technology>

<https://www.forbes.com/sites/cognitiveworld/2019/07/05/how-ai-is-transforming-agriculture/#7af8c74a4ad1>

<https://www.sciencedirect.com/topics/engineering/plant-biomass>