10 Academy Batch 4 - Weekly Challenge: Week 6

AgriTech - USGS LIDAR Challenge

Overview

Business Need

At AgriTech, we are very interested in how water flows through a maize farm field. This knowledge will help us improve our research on new agricultural products being tested on farms.

How much maize a field produces is very spatially variable. Even if the same farming practices, seeds and fertilizer are applied exactly the same by machinery over a field, there can be a very large harvest at one corner and a low harvest at another corner. We would like to be able to better understand which parts of the farm are likely to produce more or less maize, so that if we try a new fertilizer on part of this farm, we have more confidence that any differences in the maize harvest 9are due mostly to the new fertilizer changes, and not just random effects due to other environmental factors.

Water is very important for crop growth and health. We can better predict maize harvest if we better understand how water flows through a field, and which parts are likely to be flooded or too dry. One important ingredient to understanding water flow in a field is by measuring the elevation of the field at many points. The USGS recently released high resolution elevation data as a lidar point cloud called **USGS 3DEP** in a public dataset on Amazon. This dataset is essential to build models of water flow and predict plant health and maize harvest.

You work at an AgriTech, which has a mix of domain experts, data scientists, data engineers. As part of the data engineering team, you are tasked to produce an easy to use, reliable and well designed python module that domain experts and data scientists can use to fetch, visualise, and transform publicly available satellite and LIDAR data. In particular, your code should interface with **USGS 3DEP** and fetch data using their API.

You may search for open source python packages and adapt them to your needs, or you may choose to write everything from scratch. In the former case, please be clear about **attributing where the work and code came from - this is essential**.

The quality of your python package is judged by

- How easy it is to install and use
- How much abstraction it exposes - from high level - a few function calls to do all routine work, to low level - gives a high degree of control to users.
- CPU and RAM usage
- Implementation of parallelisation to speed up fetching, loading, transforming, and visualization.

You may not have time to implement all these elements this week, but you may write your code thinking about future implementation of these and other useful features. As per previous weeks, you may start your work by writing issues in your git repository, and completing them one by one throughout the week.

Expected Outcomes

Any industry working with satellite, or agriculture would likely be impressed to see a project like this on a portfolio.

Skills:

- Working with satellite imagery as well as geographical data files
- Exposure to building API that interacts with satellite imagery
- Code packaging and modularity
- Building data pipelines and orchestrations workflows

Knowledge:

- Satellite and geographical Image processing
- Functional and Modular Coding
- API access to Big Data

Competency Mapping

The tasks you will carry out in this week's challenge will contribute differently to the 17 competencies 10 Academy identified as essential for job preparedness in the field of data science, and Machine Learning engineering. The mapping below shows the change (lift) one can obtain through delivering the highest performance in these tasks.

MC0: Marginal contribution - causes no significant change
MC1: Minor contribution - recognized for routine performance gain
MC2: Measurable contribution - will contribute a value towards portfolio and job readiness metric
MC3: Major contribution - the best performance of these types of tasks at least three times within our training leads one to attain a job-ready level along that competency dimension.

Competency	**Value**	**Potential contributions from this week**

Business Understanding	MC3	Understanding and reasoning the business context. Thinking about suitable analysis that matches the business need. Thinking about clients and their interests.
Data Engineering	MC3	Thinking about how to store data for easy analysis, and what format to use to build responsive dashboards.
Data Understanding	MC3	Understanding the data provided and extract insight. Exploring different techniques, algorithms, statistical distributions, sampling, and visualization techniques to gain insight.
Dashboard & Visualization	MC2	Building a dashboard to explore data as well as to communicate insight. Advanced use of modules such as plotly, seaborn, matplotlib etc. to build descriptive visualizations. Reading through the modules documentation to expand0 your skill set.
Mathematics and Statistics ni ni	MC0	Thinking about statistical distributions, sampling, bias, overfitting, correlations.
MLOps & Continuous Delivery	MC2	Using Github for code development, thinking about feature store, planning analysis pipeline, using MLOps tools for code, data, model, artifact versioning, setting up docker containers for automated microservice deployment.
Modeling and evaluation	MC0	Comparing multiple Deep learning techniques; training and validating DL

		models; choosing appropriate architecture, loss function, and regularisers; hyperparameter tuning; choosing suitable evaluation metrics.
Python programming	MC3	Advanced object-oriented python programming. Python package building.
SQL programming	MC1	Building feature stores using SQL or NoSQL databases.
Fluency in the Scientific Method	MC1	Thinking about evidence. Generating hypothesis, testing hypothesis. Thinking about different types of errors.
Ethics	MC1	data privacy, data security, ethical use of data. The 8 principles of responsible machine learning
Statistical & Critical Thinking	MC1	Thinking about the difference between causal vs chance correlation. Giving reasonable recommendations. Thinking about uncertainties.
Software Engineering & Dev Environment	MC3	Reading articles on software project planning. Unit testing.
Impact & Lifelong learning	MC3	Learning new concepts, ideas, and skills fast, and applying them to the problem at hand.
Professional Culture & Communication	MC2	Writing a well-formatted presentation with no mistakes, formatted nicely.
Social Intelligence & Mentorship	MC2	Asking for help early, providing help for those who need it, avoiding being stuck.

Career Thinking	MC1	Working within groups in a successful way

Team

Instructors: Yabebal, Abubakar, Mahlet, Kevin

Group Work Policy

This submission is to be done individually. Collaborative learning is encouraged, but each person must have his or her own submissions.

Key Dates

- **Discussion on the case** - 11:30 UTC time on Monday 16 August 2021. Use #all-week5 to ask questions.
- **Interim Submission** - 8:00PM UTC time on Wednesday 18 August 2021.
- **Final Submission** - 8:00PM UTC time on Saturday 21 August 2021

Leaderboard for the week

There are 100 points available for the week.

20 points - community growth and peer support.

13 points - technical public and group-based RC channels

- Total number of messages (5)
- Total number of Mentions (3)
- Total number of DM connections (5)

7 points - community activities

- Number of messages in non-technical channels (4)
- On-time presence in Gmeet sessions (3)

30 points - presentation and reporting.

15 points - interim submission. PDF slide or report format. Evaluated for:

- Overview of LiDAR and Satellite data formats (3)
- Discussion of tools used to access and load LiDAR data (4)
- Code flow diagram and report of what is completed. (4)

15 points for the final submission. Evaluated as follows

- PDF of a presentation slide demonstrating what your code package does (5).
- Link or PDF of your code documentation generated by Sphinx or similar tool (5 points)
- Well written Readme (5)

50 points - Technical content

20 points - Interim submission

1. Github link submission (20)
    - Object-oriented programming (5)
    - Jupyter notebook illustrating the inputs and outputs (5)
    - Git issues or project that shows your work plan (5)
    - Successful LiDAR data fetching (5)

30 points - Final submission

- Github Link submission (20)
    - Pip installable python package that contains an implementation for data fetching, loading, transforming, and visualization. (10)
    - Jupyter notebook that uses the developed package and shows the visualization of the data (10)
- Package Documentation (10)
    - Documentation that details your package including how to use it, and which part of the code to call for what (5)
    - Well written Readme (5)

Badges

Each week, one user will be awarded one of the badges below for the best performance in the category below.

In addition to being the badge holder for that badge, each badge winner will get +20 points to the overall score.

Visualization - the quality of visualizations, understandability, skimmability, choice of visualization

Quality of code - reliability, maintainability, efficiency, commenting - in the future this will be CICD/CML

An innovative approach to analysis -using latest algorithms, adding in research paper content and other innovative approaches

Writing and presentation - clarity of written outputs, clarity of slides, overall production value

Most supportive in the community - helping others, adding links, tutoring those struggling

The goal of this approach is to support and reward expertise in different parts of the Machine learning engineering toolbox.

Late Submission Policy

Our goal is to prepare successful learners for the work and submitting late when given enough notice, shouldn't be necessary.

For interim submissions, those submitted 1-6 hours late will receive a maximum of 50% of the total possible grade. Those submitted >6 hours late may receive feedback, but will not receive a grade.

For final submissions, those submitted 1-24 hours late, will receive a maximum of 50% of the total possible grade. Those submitted >24 hours late may receive feedback, but will not receive a grade.

When calculating the leaderboard score:

- From week 8 onwards, your two lowest weeks' scores will not be considered.

Instructions

The fundamental tasks in this week's challenge are the following

1. Interact with a public API, know how to use it;
2. Create tools that can be used to interact with the API in a more user friendly and effective manner;
3. Visualize the geospatial data returned, if possible
4. Ensure the package is documented in a way that allows usage and understanding

The workflow for this week's challenge is as follows

- Read instructions and understand the business needs, the type of data available, the data engineering process(es) that needs to be carried out, the Workflow requirements, and the resources required/available to complete the project
- Plan your work and set up development environment to assist in completing the project
- Explore a sample of the dataset, understand it structure, the information stored within and develop intuition on how it can be used
- Set up a github repo, integrate unit testing and CICD for proper code package test and deployment
- Build a codebase that communicates with the provided data source and extract needed information based on the parameters passed

Task 1 - Data Fetching and Loading

LIDAR high definition elevation data - USGS 3DEP - The USGS recently released high resolution elevation data as a lidar point cloud in a public dataset on Amazon. This dataset is complicated to understand and use, and it would be useful to have an easy way to access and use it in order to build models of water flow and predict plant health and maize harvest.

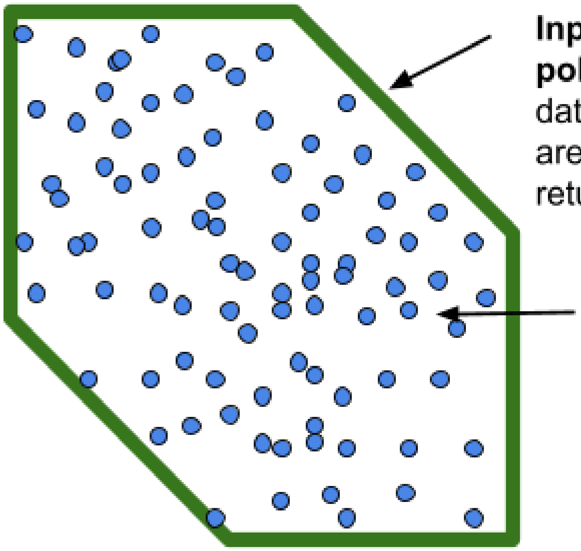
Your task is to write a modular python code/package to connect to the API, query the data model to select with a specified input and get a desired output. For example, submit a boundary (GPS coordinates polygon) and receive back a raster of the height of the terrain within the boundary. The expected inputs and outputs are

Inputs:

- Field boundary polygon in geopandas dataframe
  - All CRS's (coordinate reference systems) should be accepted
- Desired output CRS

Outputs:

- **Python dictionary** with all years of data available and geopandas grid point file with elevations encoded in the requested CRS.



Input boundary polygon. Geopandas dataframe defines what area of elevation points to return.

Output elevation point data. Each GPS point in the geopandas dataframe contains absolute elevation in meters, for example, 898.21 meters.

Returned dictionary structure example:

```
{
        2010: <geopandas dataframe>,
        2013: <geopandas dataframe>,
        2019: <geopandas dataframe>
}
```

Output geopandas dataframe example

	elevation_m	Geometry

0	299.02	POINT (978820.198 1954075.104)
1	298.78	POINT (978820.198 1954075.104)
2	298.82	POINT (514632.161 1532825.301)
...	...	...

Task 2 - Terrain Visualization

Include an option to graphically display the returned elevation files as either a 3D render plot or as a heatmap. The following is an example visualisation.

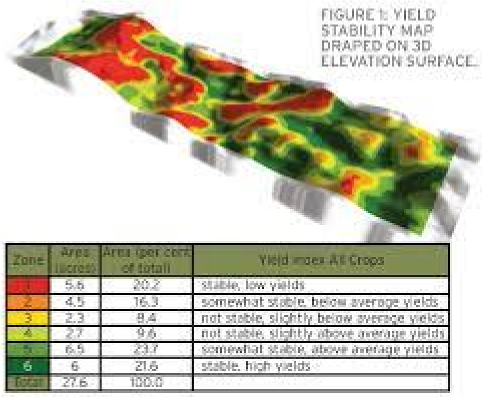


FIGURE 1: YIELD STABILITY MAP DRAPED ON 3D ELEVATION SURFACE.

Zone	Area (acres)	Area (per cent of total)	Yield index All Crops
1	5.6	20.2	stable, low yields
2	4.5	16.3	somewhat stable, below average yields
3	2.3	8.4	not stable, slightly below average yields
4	2.7	9.6	not stable, slightly above average yields
5	6.5	23.7	somewhat stable, above average yields
6	6	21.6	stable, high yields
Total	27.6	100.0	

Task 3 - Data Transformation

1. **Topographic wetness index (TWI)** - as an additional column returned with geopandas dataframe

	elevation_m	geometry	TWI
0	299.02	POINT (978820.198 1954075.104)	-3.5
1	298.78	POINT (978820.198 1954075.104)	-0.2

2	298.82	POINT (514632.161 1532825.301)	1.1
...	...	...	...

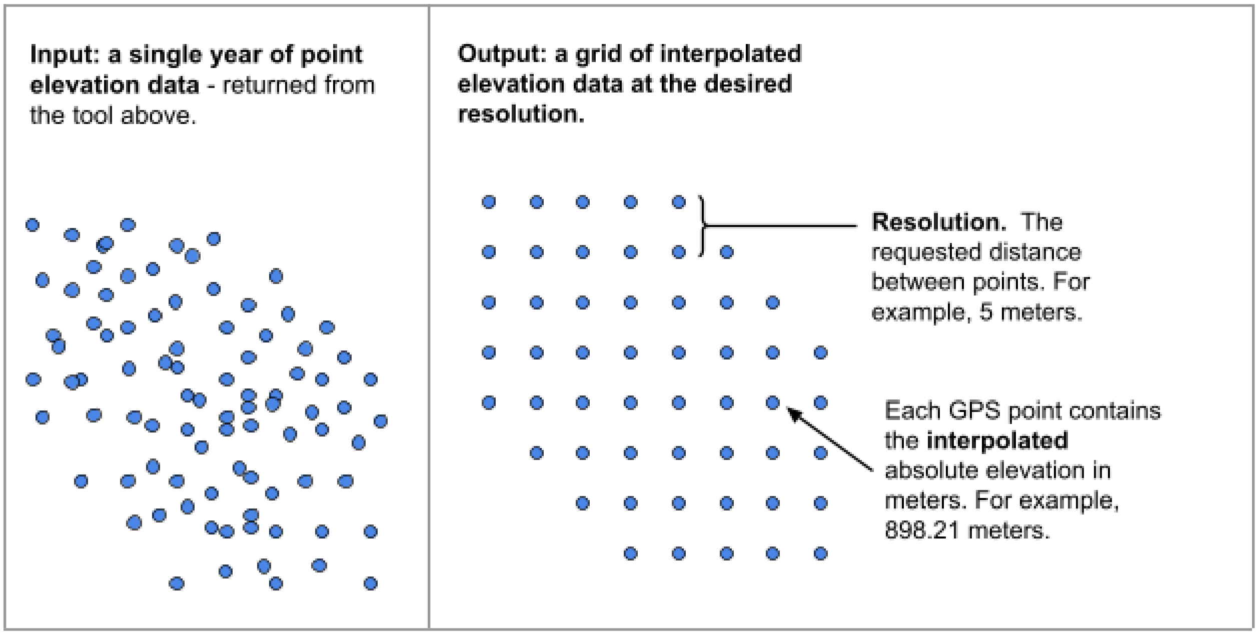
2. **Standardized grid** - A python code that takes elevation points output from the USGS LIDAR tool and interpolates them to a grid.

   Inputs:

   - A single year geopandas elevation point dataframe returned from the tool above.
   - Desired output resolution (in meters)

   Outputs:

   - An interpolated grid of points with interpolated elevation information
   - An option to visualize the output grid as a 3D render or heatmap to visually compare to the original, un-interpolated elevation data.



Input: a single year of point elevation data - returned from the tool above.

Output: a grid of interpolated elevation data at the desired resolution.

Resolution. The requested distance between points. For example, 5 meters.

Each GPS point contains the **interpolated** absolute elevation in meters. For example, 898.21 meters.

Bonus Tasks

Write a python code to query and fetch the following data sets, and visualise them together with the LIDAR data

1. **USGS soil data SSURGO** - write a code to interact with. There's more notes about this API below.
    a. For example, submit a boundary (GPS coordinates polygon) and receive back a raster or a geojson object with the shapes of the mukeys (soil types) within that boundary.

2. **Satellite data** - write a code to interact with a Sentinel public API dataset.
    a. Submit a boundary (GPS coordinates polygon) and receive the dates with imagery available within the boundary . Then submit date(s) for download and visualization of the satellite data.

3. **Climate data** - retrieve and visualize long-term averages for different types of weather data. Some example data sources are:
    - **worldclim** - climate averages
    - prism - oregon state - averages and time series
    - CRUTS - climate research unit time series
    - Projections - climate GCM models based on those inputs

4. **Quickstart guide for your package**- provide code snippets to help users use your package

Notes on USDA SSURGO soils data

https://sdmdataaccess.sc.egov.usda.gov/documents/TableColumnDescriptionsReport.pdf

Things to know

- The different layers in the soil (soil horizons) get depth weighted to give a single value for an attribute by cokey which gets weighted by the percentage of the map unit (mukey) it makes up.
- A map unit is a polygon area (on the soils surface) that has an associated set of soils and soil characteristics.
- Each map unit has a unique 'mukey' (or map unit key) that relates the associated set of soils and soil characteristics to the polygon area.
- The Component Key (cokey) links soil components to a mukey
- How the components change with the horizon (or depth) is uniquely identified with a horizon key (chkey)
- We depth average soil attributes (uniquely identified by chkey) for a given cokey and then weight them by the percentage of the mapunit made up from that cokey to get a single weighted average of the soil characteristic for a given mukey.
- The USDA SSURGO data typically has 3 values for each soil component for example: **sandtotal_l, sandtotal_r, sandtotal_h**. Here **_l** denotes the low value, **_r** denotes the representative value, **_h** denotes the high value.
- Texture is the dominant texture class of the largest soil component in the mukey

https://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/nrcs142p2_050900.pdf

Submission

Interim: Due Wednesday 18.08 8pm UTC

1. A pdf file that shows updates on the status of your work (3 Pages). This should include:
    a. Overview of the data source and formats
    b. Python packages you plan to build upon
    c. Planned schedule for completion of your work, including the entire week
    d. Progress vs. the schedule and explanation of any schedule variations
    e. Any blockers and your plan to overcome them
    f. Potential areas of risk that could impact an on-time delivery
    g. Any updates that may be relevant including
        i. Overlooked or understated complexities
        ii. Project scope validity
        iii. new learnings relevant to other team members.
2. Github link submission that demonstrates
    a. Object-oriented programming
    b. Well written Readme
    c. Git issues or project that shows your work plan

Final: Due Saturday 21.08 8pm UTC

1. Code Documentation (PDF or link if it is deployed as HTML pages somewhere)
2. Link to your Github code that includes your Jupyter notebook.
    a. Python package that contains an implementation for data fetching, loading, transforming, and visualization. (15)
    b. Jupyter notebook that shows the visualization of the data (10)

Feedback

You will receive comments/feedback in addition to a grade.

References:

Conceptual

- Get to know Lidar (Light Detection and Ranging) Point Cloud Data - Active Remote Sensing | Earth Data Science - Earth Lab
- 3D Point Cloud processing tutorial by F. Poux | Towards Data Science
- Create a Farm Map with Soil and Elevation Data Using QGIS | Towards Data Science

Existing Python Packages

Reference for both code and documentation

- LP DAAC - Getting Started with GEDI L2A Data in Python (usgs.gov)
- LidarVegMetrics/HAG311.py at master · jyoungUSGS/LidarVegMetrics (github.com)
- riverscapes-tools/3dep_testing.ipynb at 3dep · Riverscapes/riverscapes-tools (github.com)
- giswqs/lidar: A Python package for delineating nested surface depressions from digital elevation data. (github.com)
- Reading data from EPT — pdal.io
- cheginit/py3dep: A part of HyRiver software stack for getting topography data within the US through 3D Elevation Program (3DEP) (github.com)
- sentinel-hub/field-delineation: Field delineation with Sentinel-2 data from Sentinel-Hub and a ResUnet-a architecture. (github.com)
- ubarsc/pylidar: A set of Python modules which makes it easy to write lidar processing code in Python (github.com)
- Joffreybvn/lidario: High-level python library to manipulate LIDAR raster and point cloud. (github.com)
- jakarto3d/jakteristics: Compute point cloud geometric features from python (github.com)
- Ekan5h/LIDARtoolkit: Simplifying LIDAR point cloud processing and rapid prototyping (github.com)

General

- PDAL Tutorial - Basic LiDAR Data Handling (paulojraposo.github.io)
- Extracting buildings and roads from AWS Open Data using Amazon SageMaker | AWS Machine Learning Blog
- https://www.earthdatascience.org/courses/use-data-open-source-python/intro-vector-data-python/spatial-data-vector-shapefiles/intro-to-coordinate-reference-systems-python/
- https://pdal.io/tutorial/iowa-entwine.html

Data

- USGS 3DEP LiDAR Point Clouds - Registry of Open Data on AWS
- usgs-lidar/lambda at master · hobu/usgs-lidar (github.com)
- https://scihub.copernicus.eu/dhus/#/home

Tools:

- LAStools: converting, filtering, viewing, processing, and compressing LIDAR data in LAS format (unc.edu)
- rapidlasso GmbH | fast tools to catch reality
- PDAL/lambda: AWS Lambda Layer for PDAL (github.com)
- https://jblindsay.github.io/wbt_book/tutorials/lidar.html
- https://sentinelsat.readthedocs.io/en/master/api_overview.html
- farmOS.org
- FarmBuild - Farm Data

Some typical point cloud processing challenges and complimentary software include:

- Compression – https://laszip.org
- Organization – https://entwine.io
- Translation – https://pdal.io
- Exploitation – http://lastools.org, https://pdal.io, https://grass.osgeo.org/
- Visualization – http://potree.org/
- CloudCompare - http://plas.io

Code documentation

- https://docs.python-guide.org/writing/documentation/
- https://www.sphinx-doc.org/en/master/

Reference of References

- https://pythonrepo.com/tag/lidar-point-clouds