Data Scientist Take Home Assignment
Rubicon

Setup
This technical assessment focuses on Geospatial Data Processing and Computer Vision Modeling.

Requirements:

- All work must be completed in Python.
- Deliverables should include:

  -> All source code files (.py and/or .ipynb).
  -> A clear and reproducible workflow.
  -> Explanations, justifications, and answers to provided questions, either as markdown cells within notebooks or compiled in a separate PDF document.

You will be provided with:

- A GeoJSON file defining an Area of Interest (AOI) for the geospatial preprocessing section (AOI_Rubicon.geojson).

Recommendations:

- For the deep learning section, it is recommended to leverage GPU-accelerated environments such as Google Colab or Kaggle for faster training and inference.
- Although you may work either in notebooks or standalone Python scripts, please structure your code in a way that would easily scale to production environments, as this is one of the core skills we are looking for.

Questions

Section One: GeoSpatial Data Processing
In this section, you will implement a pipeline to generate a time series of vegetation indices for a given Area of Interest (AOI). The vegetation indices we will be looking at are the NDWI and the MSI.

Objective:

Given:

- An AOI (given as GeoJSON).
- A vegetation index.
- A start and end date.

You must produce a **GeoTIFF file**, where:
- Each band corresponds to the selected vegetation index computed at a specific acquisition date within the requested date range.
- The output GeoTIFF should have the same CRS as the input GeoJSON.

Requirements:
- We advise using Sentinel-2, but Landsat will also be accepted. **Important:** Clearly explain your choice of data **preprocessing level** (e.g., Level-1C, Level-2A for Sentinel-2, or L1/L2 for Landsat).
- You are free to choose the API, library, or provider you want. **Important:** Explain your choice of API/provider and discuss its advantages and drawbacks.
- You are not expected to submit the generated GeoTIFFs. We will test your code by running it against different AOIs and date ranges.

Section Two: Deep Learning - Semantic Segmentation on Sentinel-2 Imagery

In this section, you will implement a pipeline that performs semantic segmentation on Sentinel-2 imagery using a pretrained model. At this stage, we want to perform basic segmentation on 3 or so classes: Water, Vegetation and Buildings.

Objective:

Given:
- An AOI (given as GeoJSON).
- A target date (You may select any date within the last 5 years)

Your task is to:
- Download the Sentinel-2 data for the target date for that AOI. If data is unavailable for the target date, retrieve data for the closest available date.
- Select a pretrained model suited to this task. **Important:** Explain why you picked this model. Don't hesitate to go into detail here, on the training dataset used, the performance metrics, tradeoffs etc. We intentionally did not preselect a model, as we want to assess and understand your model selection process. This is a key skill we will be looking at.
- Process the data to the right format for the model
- Output predictions for our AOI and target date using the selected model. We do not expect a specific output type, just the one outputted by the model (csv, GeoTIFF, png). You **are** expected to submit the output along with your code.

Notes:
- You may use either PyTorch or Tensorflow.
- If the AOI generates too large of a GeoTIFF for reasonable inference times, you are allowed to reduce the area accordingly.
- In this section, **you are not expected to fine-tune the model**, but use it as is.

Section Three: Deep Learning - Model Fine-Tuning for Crop Mapping

In this section, you will implement a pipeline that fine-tunes a pretrained segmentation model to perform crop mapping using Sentinel-2 imagery.

Objective:

Your task is to:

- Fine-tune a pretrained segmentation model. You can use the same model from section 2, or pick another one. Some potential options include U-Net or DeepLabV3+.
- The model should classify crop type (e.g., Wheat, Rice, Maize, Vegetation, Bare Soil).
- Evaluate the model's performance on a validation set and report on its effectiveness in mapping different crop types.

Notes:

- **Dataset Choice**: Choose a dataset suitable for crop mapping tasks. This could be a simpler one like Munich480, or more extensive ones available on Google Earth Engine or Kaggle.
- Provide any files that you think are useful for the evaluation of your model, learning curves, metrics and such. This should include your code, the model weights or checkpoints, and any visuals and explanations you think are relevant.
- Given the allotted time, focus on having clean, reproducible code. The goal of this last section is to understand how you could approach a fine-tuning task.