# 4-Day CopPhil Training – Day 1: EO Data, AI/ML Fundamentals & Geospatial Python

## 1. Lecture Materials

**Introduction to Copernicus in the Philippines:** The Copernicus Capacity Support Programme for the Philippines (CopPhil) is a flagship initiative under the EU’s Global Gateway. It aims to increase the uptake of free and open Copernicus Earth observation data in the Philippines, strengthening the country’s ability to address **disaster risk reduction (DRR)**, **climate change adaptation (CCA)**, and *natural resource management (NRM). This involves establishing a local Copernicus data mirror site and co-developing pilot services, in partnership with the Philippine Space Agency (PhilSA) and DOST (Department of Science and Technology). Slide content:* Key points should include the EU-Philippines space cooperation context, CopPhil goals (DRR, CCA, NRM), and the roles of PhilSA and DOST in the programme.

**Copernicus Sentinel-1 & Sentinel-2 Overview:** Sentinel-1 and Sentinel-2 are core satellites of the EU Copernicus program, providing free high-resolution imagery. Sentinel-1 is a radar (SAR) mission with **C-band synthetic aperture radar** enabling all-weather, day/night imaging. With two satellites (1A & 1B, now 1C) orbiting 180° apart, Sentinel-1 achieves a ~6-day revisit cycle globally. In its common Interferometric Wide (IW) swath mode, Sentinel-1 has ~**5 m by 20 m** spatial resolution (range × azimuth) and a 250 km swath. It provides dual-polarization data (VV+VH or HH+HV) useful for mapping floods, land deformation (InSAR), forest biomass, and maritime surveillance. Typical products are Level-1 **GRD (Ground Range Detected)** images (multi-look intensity) and SLC (Single Look Complex) images, which are all openly accessible. Sentinel-2 is an optical mission carrying a **MultiSpectral Instrument (MSI)** with **13 spectral bands** (visible, near-infrared, and shortwave-infrared) at **10 m, 20 m, and 60 m** resolutions. Twin satellites (2A & 2B) in sun-synchronous orbit provide a **5-day revisit** for any location. Each Sentinel-2 scene covers a 100×100 km tile. Important band characteristics: 10 m for RGB and NIR bands, 20 m for red-edge and SWIR bands, 60 m for atmospheric bands. Slides should list Sentinel-2’s **Level-1C (Top-of-Atmosphere)** and **Level-2A (Surface Reflectance)** products, and Sentinel-1’s Level-1 GRD and SLC products, along with data access methods (the new Copernicus Data Space Ecosystem replacing the legacy Open Access Hub, and platforms like Google Earth Engine). *Slide content:* comparisons of Sentinel-1 vs Sentinel-2 (sensor type, spectral bands, spatial/temporal resolution, example uses). Mention that Sentinel data can be retrieved via Copernicus hubs (now the Data Space Ecosystem) or via cloud platforms and APIs (e.g., ASF for Sentinel-1, and GEE).

*Example Sentinel-2 imagery of Mayon Volcano in the Philippines (false-color composite highlighting the 2018 lava flows). Sentinel-2 provides multi-spectral optical data at 10–60 m resolution across 13 bands. With a 5-day revisit, it supports monitoring of dynamic environmental events. (Contains modified Copernicus Sentinel data [2018] processed by Pierre Markuse*[*[1]*](https://commons.wikimedia.org/wiki/File:Mount_Mayon_Sentinel-2_L1C_from_2018-01-30_(28280091769).jpg#:~:text=)*)*

**Philippine EO Data Ecosystem:** The Philippines has its own geospatial data platforms that complement Copernicus data. Slides should introduce the key agencies and tools:

* **PhilSA Space Data Dashboard (Space+ SDD):** PhilSA’s online platform for **satellite data access and visualization**, built with open-source tools (TerriaJS, OpenDataCube, etc.). It provides government and citizens with easy access to satellite-derived datasets, including direct downloads of imagery for uses in disaster management, urban planning, and environmental monitoring. *Include:* a note that SDD democratizes EO data access in the country, aligning with sustainable development goals.
* **NAMRIA Geoportal:** Managed by the National Mapping and Resource Information Authority, this geoportal provides national **basemaps, topographic maps, land cover data, and hazard maps**. It allows users to view and download layers like the official 1:50k base maps, administrative boundaries, and thematic maps (e.g., 2020 land cover) for the Philippines. Emphasize its role in DRR (hazard map access) and in providing authoritative geospatial data.
* **DOST-ASTI Projects:** The Advanced Science and Technology Institute of DOST leads several EO and AI initiatives:
* **DATOS (Remote Sensing and Data Science Help Desk):** A project applying AI, machine learning, GIS and remote sensing for disaster mapping and other applications. For example, DATOS developed methods to automatically map floods from satellite images and to identify crops (like mapping sugarcane via temporal radar signatures). It essentially serves as a rapid analytics service during disasters.
* **SkAI-Pinas (Sky Artificial Intelligence Program):** A DOST flagship R&D program to **democratize AI in the Philippines** by building AI capacity in remote sensing. It focuses on making AI “part of daily decision-making and national progress”. SkAI-Pinas supports the development of AI models and tools, including **DIMER** (Democratized Intelligent Model Exchange Repository) – a repository for sharing pre-trained AI models, and **AIPI** (AI Processing Interface) – a platform to streamline large-scale AI processing of geospatial data. These tools enable Filipino researchers and agencies to apply AI without needing big compute resources, by reusing models and an accessible processing interface.
* **Philippine Earth Data Resource Observation (PEDRO) Center:** (Related infrastructure, not in bullet but may be worth noting) – DOST-ASTI’s satellite ground receiving station that acquires Diwata microsatellite data and other imagery, contributing local high-resolution data.
* **PAGASA and Other Data Sources:** The national meteorological agency PAGASA provides climate and weather data (e.g., typhoon tracks, rainfall) that can be integrated with satellite data for climate adaptation analysis. Also mention that the CopPhil program is setting up a **Copernicus Mirror Site** in-country to locally host Sentinel data for faster access, ensuring sustainable data availability for Philippine users.

*Slide content:* a diagram or list of the Philippine EO ecosystem, showing how PhilSA, NAMRIA, DOST-ASTI (DATOS, SkAI-Pinas, DIMER, AIPI), and PAGASA contribute data or tools. Highlight that these local datasets (e.g. national base maps, hazard maps, local AI models) **complement Sentinel imagery** to create richer insights. For example, during a flood event, Sentinel-1 SAR can detect water extent, while NAMRIA flood hazard maps and PAGASA rain gauges provide context – together supporting better DRR decisions.

**AI/ML Concepts in Earth Observation:** This lecture segment introduces how artificial intelligence and machine learning are applied to EO data, covering key concepts and workflows:

* **AI/ML Workflow for EO:** A typical **machine learning workflow** in EO involves several stages. *Slides should illustrate:*
* **Problem Definition:** e.g., identifying what environmental question to answer (flood mapping? land cover classification? yield prediction).
* **Data Acquisition:** gathering relevant EO data (Sentinel images, ground truth labels, ancillary data).
* **Data Pre-processing:** crucial for EO (geometric corrections, cloud masking, normalization, etc.) before feeding data to models.
* **Feature Engineering:** deriving informative features (spectral indices like NDVI, textures, DEM derivatives) from raw data.
* **Model Selection & Training:** choosing an algorithm (e.g., Random Forest, CNN) and training it on labeled examples.
* **Validation & Evaluation:** using separate test data to assess accuracy (confusion matrix, error metrics).
* **Deployment:** applying the model to new data or integrating into workflows for operational use. This end-to-end process ensures that participants see the “big picture” of how AI/ML projects are executed for EO applications.
* **Types of ML – Supervised vs Unsupervised:** Define the two main paradigms with EO examples. In **supervised learning**, the model learns from labeled data. EO examples: land cover classification (labels = classes like water, urban, forest for each pixel) and regression tasks (predicting a continuous value such as soil moisture or air pollutant concentration from satellite data). In **unsupervised learning**, the algorithm finds patterns without explicit labels. Example: clustering multispectral images to discover land cover groups or anomalies (useful for exploratory analysis or change detection). *Slides:* could show an illustration of labeled training pixels on a satellite image for supervised learning, versus an image segmented into clusters for unsupervised.
* **Neural Networks & Deep Learning Basics:** Briefly introduce that **deep learning** is a subset of ML using neural networks with many layers (“deep” networks) that excel at learning complex patterns. A slide can depict a simple **Artificial Neural Network** with neurons organized in layers (input layer → hidden layers → output). Explain key concepts in simple terms: neurons apply activation functions to weighted sums of inputs (introducing non-linearity), the network is trained by adjusting weights to minimize a loss function (error) using algorithms like gradient descent (optimizers). Emphasize how **Convolutional Neural Networks (CNNs)** are specialized for images: using convolutional layers to automatically extract spatial features (edges, textures, objects) – this will foreshadow Day 2 content. The point is to demystify terms like “layers”, “activation”, “training” for participants new to AI.
* **Data-Centric AI in EO:** “Data-centric AI” is the philosophy that improving your **data** (quality, quantity, diversity) is as important as model tuning. This is **especially critical in EO** where challenges like sensor noise, cloud cover, class imbalance, and label uncertainty can derail an AI project. Slides should stress that model performance in EO is often limited by the dataset: having well-annotated, representative training data (e.g. good ground truth for all land cover types, across different seasons and regions) is crucial. Mention strategies like augmenting training data, cleaning labels, and incorporating expert knowledge. The goal is to encourage participants to focus on creating or curating high-quality datasets for their projects, not just on choosing fancy algorithms. This aligns with CopPhil’s capacity building – ensuring participants can develop robust EO applications by paying attention to data suitability.

**Intro to Google Earth Engine & Geospatial Pre-processing:** Google Earth Engine (GEE) is a cloud-based platform highly useful for EO data handling and was introduced as a tool for the training. Key concepts to cover:

* **GEE Data Structures:** Explain that in Earth Engine, satellite imagery is handled as **Image** objects (single raster image) and **ImageCollection** (a time-series or stack of images). Vector data are handled as **Feature** (with geometry and attributes) and **FeatureCollection** (set of features). These abstractions let users manage large datasets (e.g., an ImageCollection of all Sentinel-2 images for a year over the Philippines) with simple filter operations.
* **Common GEE Operations:** Introduce **Filters** (to subset ImageCollections by date, location, metadata) and **Reducers** (to aggregate or summarize data, e.g., taking a median across images or computing statistics over a region). For example, filtering a Sentinel-2 ImageCollection to a date range and AOI, then using a reducer to make a cloud-free composite.
* **Cloud Masking in Optical Imagery:** As a specific pre-processing task, mention that Sentinel-2 Level-2A data comes with QA bands (QA60) that indicate cloud/cloud-shadow pixels which can be masked out. Simpler approach: use the QA60 bitmask to remove cloudy pixels. More advanced: use the **s2cloudless** cloud probability images provided in GEE to mask clouds with a custom threshold. Cloud masking is critical for producing clean composites and reliable inputs to AI models.
* **Composites and Mosaics:** Demonstrate the idea of creating a **temporal composite** – e.g., taking the median reflectance per pixel over a 3-month period to get a cloud-free Sentinel-2 image. Such composites reduce noise and cloud effects, useful for mapping (e.g., a 2023 annual land cover composite). Also mention mosaicking images (spatially) to cover larger areas.
* **AOI Clipping:** Explain that in GEE one can define an Area of Interest (as a geometry or FeatureCollection) and clip or mask images to that boundary – for instance, clipping a satellite image to the province of Palawan.
* **Data Access in GEE:** Note that GEE has a petabyte-scale data catalog including Sentinel-1 and Sentinel-2 collections available on demand. Users can **search** the catalog by dataset name or keywords, and apply filters (e.g., cloud cover < 20%, date range, bounds) to find scenes. For example, searching for Sentinel-1 GRD images over Manila in July 2025.
* **Exporting from GEE:** Briefly foreshadow (to be done in hands-on) that GEE allows export of images or tables to Google Drive or Cloud Storage for use in local analysis. This is important when transitioning from the prototyping in GEE to training a custom model in Python.

*Slide content:* likely a schematic of Earth Engine’s structure (images & collections, with filters/reducers), and bullet points with examples: e.g., “Use median reducer to create cloud-free image composite; Use filterBounds + filterDate to get Sentinel-1 scenes for a flood event.” Emphasize how GEE simplifies pre-processing workflows that would be tedious locally. This sets the stage for the Day 1 hands-on where they actually use GEE.

## 2. Instructor Notes (Speaker’s Script)

**Introduction (CopPhil and Course Goals):** *Speaker notes:* Begin by welcoming participants and framing the training’s purpose. Explain that **CopPhil** is an EU-funded program to build the Philippines’ capacity in using Copernicus Earth observation data. The instructor should mention that this is part of a broader EU–Philippines partnership (the Global Gateway initiative) and highlight the ultimate goals: improving disaster resilience, climate adaptation, and resource management through satellite data. For context, note that the Philippines is one of the first countries in Asia to collaborate on Copernicus, demonstrating the country’s pioneering role in applying European satellite data for local needs. If a video message from the EU Ambassador or officials (PhilSA/DOST) is included, the instructor should introduce it and tie it to these themes (e.g., the Ambassador might speak about EU’s commitment to technology cooperation). After the video, briefly acknowledge key organizers and remind participants of the 4-day structure. Outline Day 1’s topics: Copernicus program, Sentinel-1/2, the local EO landscape, AI/ML fundamentals, and introductory coding sessions. Emphasize how each session today lays groundwork: by end of Day 1, they will have both conceptual understanding and practical skills to start working with satellite data.

**Session 1 – Copernicus Program Overview & PH EO Landscape:** In this lecture, break it into two modules.

* *Copernicus & Sentinels:* Explain what Copernicus is: a European Union Earth observation program with a constellation of satellites (Sentinels) providing free data. Mention the Sentinel family (1 through 6), but focus on Sentinel-1 and 2 as the most relevant for this training. **Sentinel-1**: Describe it as a radar imaging mission – the instructor can explain in simple terms that radar satellites send microwave signals and measure the backscatter, which enables seeing the Earth’s surface **even through clouds or at night**. This is very useful in a tropical country like the Philippines with frequent cloud cover and during floods/typhoons when optical satellites might be blinded. Note the 6-day revisit with two satellites (which was reduced when one satellite went offline, but Sentinel-1C launched in 2024 to restore coverage). Give examples the audience can relate to: “Sentinel-1 can monitor floods in near real-time, detect ground deformation for earthquakes or volcanoes, and even map rice fields and deforestation, because it can frequently image the same area regardless of weather.” **Sentinel-2**: Explain it as the “eyes” in visible and infrared – akin to a very powerful camera that captures images in 13 different wavelengths. Highlight the 10 m resolution detail: “At 10-meter resolution, we can see large buildings, city blocks, fields, and coral reef extents. It’s like Google Maps satellite view but constantly updated every 5 days.” Ensure the instructor clarifies the difference: Sentinel-2 sees actual reflected light (so it’s great for **land cover, vegetation, coasts**), while Sentinel-1 senses surface roughness and moisture (great for **water, floods, soil moisture, ships at sea**). Mention how the two together are complementary – e.g., after a typhoon, Sentinel-1 can map flood waters under clouds, and Sentinel-2 can assess vegetation damage where it’s clear.

The instructor should enumerate Sentinel-2’s bands and resolutions in an accessible way: perhaps list a few key bands like Red, Green, Blue (10 m), Near-IR (10 m) which is used for NDVI (vegetation index), and SWIR (20 m) which is sensitive to moisture and burn scars. The concept of Level-1C vs Level-2A can be explained briefly: Level-2A is the atmospherically corrected product (surface reflectance) which is usually preferred for analysis; Copernicus provides it operationally (maybe mention since 2018). For Sentinel-1 products, note that the **GRD product** is most commonly used for analysis (it’s multi-look, detected imagery, which has reduced speckle and is terrain-corrected in the GRD (GRD is essentially ready-to-use backscatter)).

When covering **data access**, instructor notes should update participants on the new **Copernicus Data Space Ecosystem** – “Previously we had SciHub; now it’s a new portal (Data Space) where you can search and download Sentinel data. Don’t worry, you will practice accessing data via easier methods (like GEE and Python APIs) so you won’t necessarily need to manually download huge files today.” If appropriate, mention the local **CopPhil Mirror Site** in development, which will eventually host a copy of the data in-country – this will reduce dependence on internet bandwidth in the long run.

*Instructor tip:* Perhaps prepare a visual comparing a Sentinel-2 image and a Sentinel-1 image of the same area (for example, a flood or a volcano, showing how Sentinel-1 penetrates clouds). This can spark interest and questions.

* *Philippine EO Landscape:* Now transition to local context. The instructor should convey that *“Beyond the European satellites, we have Philippine agencies and platforms that provide crucial data and support.”* Start with **PhilSA** – since it’s new (est. 2019) some in the audience may not be fully aware of its programs. Note that PhilSA’s mandate includes making space data useful for Filipino society. The **Space Data Dashboard (Space+)** is an example of this: *“It’s a web portal where you can browse and download satellite data relevant to the Philippines. For instance, it has base maps, land cover layers, and some satellite imagery, all in one place for easy access.”* The instructor can mention that the dashboard is built on modern tech (TerriaJS, etc.) but focus on user benefits: no programming needed to get data, it’s aimed at local governments, researchers, and even students to empower them with EO information. Encourage participants to explore it after the session – “we will provide the link in the handout.”

Next, **NAMRIA Geoportal**: Many participants might know NAMRIA (as it’s the national mapping agency). Emphasize its geoportal as the source of authoritative Philippine spatial data. The instructor might say, *“If you need the official Philippine boundaries, topographic maps, or something like the latest national land cover map, NAMRIA Geoportal is the go-to.”* It’s worth explaining that the geoportal includes a map viewer and a database of layers from various agencies. For example, NAMRIA produced a 1:50,000 scale national land cover dataset (latest 2020) that you can download region by region. It also hosts hazard maps (e.g., flood susceptibility maps from DENR-MGB, which are very useful for DRR planning). This underscores how local datasets can complement satellite imagery: you might use Sentinel-2 to map current forests, and compare with NAMRIA’s official land cover for consistency or change detection.

**DOST-ASTI projects (DATOS, SkAI-Pinas, etc.):** The instructor should explain these as part of building local technological capacity. For **DATOS**, highlight some achievements or use cases – e.g., “During past typhoons, the DATOS team used satellite images to produce flood maps and give them to disaster response agencies within hours. They also worked on crop mapping and detecting features like road networks or even individual trees from high-res images using AI.” This shows participants what kinds of AI applications are already happening in the Philippines. For **SkAI-Pinas**, it might be a new concept for many; describe it as a program to make AI accessible: *“SkAI-Pinas is creating tools so that even if you’re not a data scientist, you can use pre-built AI models or easily label data.”* Mention **DIMER**: an online repository of AI models (imagine a library of models for common tasks – e.g., a model to classify land cover from Sentinel-2, or detect clouds, etc.). If someone has a new use case, they could check DIMER if a suitable model exists instead of starting from scratch. Mention **AIPI**: an interface that likely allows users to run large computations on ASTI’s servers without heavy coding (maybe through a web interface or API). The instructor can give an example: *“Suppose you have a hundred satellite images and you want to apply an AI model to all – AIPI would let you do that job on a server, rather than your laptop.”* These initiatives align with data-centric thinking – providing infrastructure and models to ease the burden on data practitioners.

Finally, **PAGASA and others:** Briefly note that meteorological and climate data (e.g., historical rainfall, ENSO forecasts) from PAGASA can be combined with EO for climate studies. Perhaps mention collaborations like Project NOAH (before PAGASA took over) which used satellite data for flood forecasting. Also, if time permits, mention universities and other agencies (DENR, etc.) have their own datasets – but those mentioned above are the main ones.

*Key message for this part:* The Philippines is developing a rich EO ecosystem – one goal of CopPhil is to ensure participants know about these resources so they can leverage **both international (Sentinel) and national data** together. Encourage any participants who belong to these agencies to briefly comment or invite others to explore their platforms, fostering a sense of community.

**Session 2 – Core AI/ML Concepts:** Now the instructor shifts to a more conceptual lecture. The tone should reassure participants that you don’t need to be a math expert to grasp the basics, and these concepts will be applied in later hands-on sessions.

* *What is AI/ML & the EO workflow:* Start by clarifying terminology: **Artificial Intelligence** is a broad field of making machines “smart” and **Machine Learning** is a subset of AI focused on algorithms that learn from data. In EO, most “AI” applications are actually machine learning models that learn patterns from satellite data to make predictions (like classification maps). Show the typical workflow (perhaps refer to a slide diagram). For each step, the instructor provides an EO-specific example:
* *Problem Definition:* e.g., “We want to automatically map where rice paddies are, from satellite images.”
* *Data Acquisition:* “We’ll gather Sentinel-2 images for the growing season, and get ground truth GPS points of rice fields from the Department of Agriculture.”
* *Pre-processing:* “We have to mask clouds, maybe compute NDVI from Sentinel-2 bands, stack multi-date images.”
* *Feature Engineering:* “Perhaps compute the temporal profile of NDVI for each field, or add elevation as a feature to help the model.”
* *Model Training:* “Use a supervised algorithm – say a Random Forest or a Neural Network – feed it the satellite-derived features and the known rice/non-rice labels to learn a classification rule.”
* *Validation:* “Check the model’s accuracy on some held-out locations – if it’s 90% accurate in identifying rice, that’s good; if it’s 50%, we have a problem (maybe more training data needed or a different approach).”
* *Deployment:* “Integrate the model into a workflow that maybe automatically updates a rice area map every month and sends it to policymakers.”

As the instructor goes through this, they should stress the iterative nature – sometimes you go back and forth (e.g., model does poorly, so you gather more data or engineer new features). This manages expectations that real-world projects require refining. It also sets up the knowledge that later in Day 2 and Day 3, they will actually do many of these steps (we will train a model, validate, etc.).

* *Supervised vs Unsupervised:* The instructor can draw an analogy: **Supervised learning** is like teaching a child with flashcards – you show an image and tell the answer (this is water, this is forest), so the algorithm can later recognize them. **Unsupervised learning** is like giving the child a stack of photos with no labels and asking them to sort them – they might group them by similar appearance (e.g., all bright green images vs all dark blue ones) but those groupings are inherent patterns, not predefined classes. In EO, highlight that **supervised classification** is very common for creating thematic maps (land cover maps, hazard maps from satellite, etc.) and requires labeled examples (which could come from field surveys, existing maps, or photo-interpretation). Mention common supervised algorithms: decision trees, random forests, support vector machines, neural networks – they will hear about some of these in the course. For **unsupervised**, mention **K-means clustering** or similar algorithms that are used for tasks like distinguishing different spectral clusters. A practical EO example: *“We could run an unsupervised clustering on a satellite image and it might separate pixels into, say, 5 clusters that we later interpret as water, urban, vegetation, bare soil, clouds – even though we didn’t tell the algorithm those labels upfront.”* However, caution that unsupervised results need interpretation and sometimes manual labeling afterwards (hence the term “unsupervised classification” in remote sensing yields a cluster map that an analyst then labels). This part should make participants comfortable with these terms and understand why the course focuses a lot on **supervised** methods (because they are powerful when you have reference data).

The instructor can reference that a lot of what they see on Google (like image recognition) is supervised learning at work – relating it to everyday AI they might know (face recognition, etc., are trained on labeled images).

* *Neural Networks introduction:* Given time, this is a very high-level intro, so focus on conceptual understanding. Possibly use a simple diagram on the slide. The instructor might say: *“Think of a neural network as a series of filtering and combination steps. The first layer might look at the input data (e.g., pixel values of different bands) and apply some transformations, then pass those to the next layer, and so on, until an output layer that makes a prediction (e.g., 0 = not rice, 1 = rice).”* Explain the terminology: each **neuron** computes a weighted sum of inputs and then applies an **activation function** (like a threshold or a non-linear function). **Layers** are just groups of neurons. Training a neural network means adjusting all those weights so that the outputs match the known targets for your training data. You can mention that deep networks have many layers and can capture very complex relationships – *“they can automatically learn the best features for the task, which is why deep learning is so powerful, albeit data-hungry.”*

It might be useful to mention an example in EO where deep learning outperforms older methods: e.g., *“In flood mapping, a simple threshold on radar backscatter might misclassify some dark surfaces as water; a deep neural network could learn more nuanced patterns (including context from neighboring pixels) to better distinguish water.”* However, also point out that neural nets need lots of training data and computation. This naturally leads into the next point: data-centric AI.

* *Data-Centric AI:* The instructor should echo Andrew Ng’s message (if familiar) that improving the dataset often can boost model performance more than tweaking algorithms. In EO, issues like mislabelled training points (maybe a point marked “forest” that is actually shrubland on the ground) or imbalanced samples (90% of your training pixels are “no change” and 10% “change” – the model will be biased) are common. Give a concrete story: *“One project tried to map coral reefs with ML, but the model struggled until they realized the training data for ‘reef’ was mostly from clear water areas. Once they added examples of reefs in turbid water, the accuracy improved – the data was the key.”* Another example: *“If your satellite images have clouds, no fancy model will predict land cover right through clouds – you either need to mask them or use cloud-penetrating data like SAR. That’s a data preprocessing issue.”*

In practice, advise participants: **whenever your model isn’t performing, first examine your data.** Are the input features appropriate? Are the labels reliable? Is the training set representative of all conditions? Encourage them to think critically about data quality and to document data sources carefully (since in government projects, knowing the lineage of data is important). This mindset will pay off in later days when they prepare their own training sets for the hands-on exercises.

**Session 3 – Hands-on Python for Geospatial (Colab):** Here the instructor transitions from theory to practice. Likely before diving into the live notebook, they will give some context/slides:

* *Colab Setup:* Explain that Google Colab is essentially a free Jupyter notebook environment in the cloud. Ensure everyone has the link to the Day 1 Colab notebook (perhaps shared via chat or a Learning Management System). Walk through the interface: where to write code vs text, how to run a cell (Shift+Enter), and how to reset if needed. Mention that Colab provides some resources like a small amount of Google Drive storage (when mounted) and internet access to install libraries or fetch data. It’s important to show how to **mount Google Drive** (so participants can save outputs or upload data). In the notes, the instructor should say: “We’ll now mount your Google Drive. This will ask for authentication – please follow the prompt to permit Colab to access your Drive. Once mounted, you can read/write files as if it’s a local disk.” This avoids confusion when they have to read a shapefile or save a result.
* *Installing Packages:* Colab usually comes with many scientific packages pre-installed, but specialized ones like geopandas or rasterio might need installation. The instructor notes: “If you run the cell I provided, it does !pip install geopandas rasterio. This will take a minute to run. You might see a message to restart runtime – if so, go to Runtime > Restart and then continue (this is needed when new packages are installed).” Ensure everyone does this successfully before moving on.
* *Python Basics Recap:* Given varying Python experience, quickly review what a pandas DataFrame is vs a GeoDataFrame, what a numpy array is (for raster data), etc. Possibly mention basic Python types (lists, dicts) if needed, but likely the group has some background. The instructor can reassure: “Don’t worry if you’re not a Python expert – we will provide template code. Focus on understanding the steps and being able to modify them for your own data later.”
* *GeoPandas (Vector data) hands-on:* Now lead the group through reading a vector dataset. The sample provided (for example, *Philippine administrative boundaries shapefile* for provinces or regions) can be used. In the notes, explain each step:
* “We use GeoPandas which extends Pandas to handle spatial data. When we do gpd.read\_file('philippines\_provinces.shp'), it will load the shapefile into a GeoDataFrame.”
* After loading, show gdf.head() and gdf.crs (to discuss coordinate reference systems briefly, e.g., it might be WGS84 lat-long which is typical). If needed, demonstrate re-projecting with to\_crs if planning to do area calculations.
* Then, “Let’s visualize the vector data.” Show how gdf.plot() works for a quick map. The instructor should mention that in a notebook, plots appear inline. Possibly discuss how to change the color or add a column-based coloring (e.g., color provinces by region by passing column='Region' to plot).
* If relevant, demonstrate filtering the GeoDataFrame: e.g., gdf[gdf['Province']=='Palawan'] to get one province, and then plotting that. This ties into clipping operations later.
* Encourage participants to try simple tasks like identify how many features (provinces) are in the data (len(gdf)) and to check attribute names (gdf.columns).

As they do this, the instructor walks around (or virtually checks) to make sure everyone is getting output maps.

* *Rasterio (Raster data) hands-on:* Next, the instructor note introduces Rasterio for reading raster files (like GeoTIFF). The sample could be a small Sentinel-2 image tile (or a subset). Explain that raster data = pixels in a grid with georeferencing. Using rasterio.open('image.tif') provides a dataset object. Show how to read metadata: e.g., src.profile or src.count (number of bands), src.width, src.height, src.crs, src.transform (affine transform tying pixel coords to real world). Then read data: e.g., band1 = src.read(1) to get the first band as a numpy array. If the image has multiple bands, maybe read a few and display. This is a good place to show an actual image: perhaps use matplotlib to display a RGB composite. The instructor can include a code snippet using plt.imshow() with a combination of bands (taking care to stretch or normalize if needed). For example:

import numpy as np  
rgb = np.dstack([src.read(4), src.read(3), src.read(2)]) # Sentinel-2 bands 4-3-2 as true color  
plt.imshow(np.clip(rgb \* 3, 0, 255).astype('uint8')) # naive stretch  
plt.title("Sentinel-2 True Color")

Adjust as necessary for actual data values. If the sample is reflectance (0-1), scale to 0-255 for display.

The instructor should narrate what they’re doing: *“Here I’m stacking bands 4-3-2 which correspond to red, green, blue. I apply a simple scaling just to make it visible. The result is an image that looks like a photograph.”* If the network is good, perhaps even overlay the vector boundary on the image (though that might be advanced for now – could skip or mention it as possible with contextily or plotting libraries).

Show basic raster ops: like cropping. To “crop to an AOI”, one can use Rasterio’s mask function with a GeoJSON geometry (e.g., geometry of a province from the GeoDataFrame). Demonstrate that if possible: retrieve a province polygon from GeoPandas (geom = gdf.loc[gdf['Province']=='Palawan', 'geometry'].iloc[0]), then use rasterio.mask.mask(src, [geom], crop=True) to get the image subset. This ties back to the concept of AOI clipping from the lecture.

Another operation: resampling – e.g., if you want to resample the image to a coarser resolution, show how to use rasterio.warp.resize or read with out\_shape parameter. But due to time, might skip detailed resampling and just mention it.

Throughout, reinforce why these operations are important: *“In many projects, you won't use entire global images; you’ll subset to your area. Or you might need to align raster resolutions, say Sentinel-2 (10 m) with another dataset at 30 m.”*

Ensure participants run these steps and see output. If any error (like CRS mismatch warnings), explain those.

Ultimately, by the end of this hands-on, participants should feel “I can load a shapefile and a GeoTIFF in Python, inspect them, and do a simple plot.” This is foundational for the upcoming days.

* *Importance of Python skills:* As a closing note for Session 3, the instructor should stress: *“We just covered a lot of technical steps, but these form the bedrock of more complex workflows. If you’re comfortable loading and examining data like this, you’ll be able to preprocess inputs for AI models or analyze outputs.”* Encourage questions if anyone struggled. Possibly provide troubleshooting tips (e.g., if a file won’t read, check path; if a plot is blank, check if you closed the dataset or if the array values need scaling).

**Session 4 – Intro to GEE for Data Access:** Now the last part of Day 1, which may be partly lecture/demo and partly interactive.

* *Using the GEE Code Editor vs Python API:* The instructor might briefly show the Earth Engine Code Editor web interface (if participants have GEE accounts enabled). However, since the question suggests using the Python API in Colab, focus on that. Explain that Earth Engine’s Python API allows you to run Earth Engine commands from a Colab notebook – effectively sending tasks to Google’s servers which hold the satellite data.
* *Authentication:* In the notes, clearly instruct: “We need to authenticate Earth Engine. When you run ee.Authenticate(), it will give a URL – click it, log in with your Google account (the one with GEE access if needed), get the code, paste it back. Then run ee.Initialize(). After this, we can call Earth Engine functions.” This process might trip some up, so ensure everyone completes it.
* *Searching and Filtering ImageCollections:* Demonstrate with Sentinel-2:
* import ee  
  ee.Initialize()  
  s2 = ee.ImageCollection('COPERNICUS/S2\_SR') \  
   .filterDate('2021-01-01', '2021-12-31') \  
   .filterBounds(ee.Geometry.Point(120.9842, 14.5995)) \  
   .filterMetadata('CLOUDY\_PIXEL\_PERCENTAGE', 'less\_than', 20)  
  print(s2.size().getInfo())
* This code example would filter the Sentinel-2 surface reflectance collection for year 2021 over Manila city (just an example point) with <20% cloud cover. The instructor should explain each part: *“filterDate for time range, filterBounds for location (using a point or we could use ee.Geometry.Polygon for an AOI), filterMetadata for metadata like cloud percentage.”* Check how many images (s2.size().getInfo() returns a number).

Then show how to get one image or a composite: e.g., image = s2.median() to take median. Or use .first() just to grab the first image in the filtered collection. Also, show how to **add new derived bands** if desired (maybe not now, but mention it’s possible to map over collections and add bands).

* *Cloud Masking Example in GEE:* Provide a short function for Sentinel-2 cloud masking using the QA60 band (for simplicity):
* def maskS2clouds(image):  
   qa = image.select('QA60')  
   # Bits 10 and 11 are clouds and cirrus  
   mask = qa.bitwiseAnd(int('010000000000',2)).eq(0) \  
   .And(qa.bitwiseAnd(int('100000000000',2)).eq(0))  
   return image.updateMask(mask).copyProperties(image, ["system:time\_start"])
* Explain that this creates a mask where cloud bits are 0 (meaning clear). Then apply: s2\_clean = s2.map(maskS2clouds). Now if you do median composite on s2\_clean, it should ignore clouds. The instructor should clarify: *“We’re using Earth Engine’s ability to handle bitmasks on the QA60 band to filter out cloudy pixels. This is one approach; a more flexible one uses the s2cloudless probability as mentioned earlier, but QA60 is quick and works ok for basic needs.”*
* *Creating a Composite & Visualization:* With s2\_clean, do something like:
* composite = s2\_clean.median().clip(ee.Geometry.Point(120.9842,14.5995).buffer(50000))  
  url = composite.getThumbURL({'min':0,'max':3000,'bands':'B4,B3,B2'})  
  display(Image(url=url))
* (Note: getThumbURL is one way to get a quick preview; or use geemap library to display on folium map, but that might be too much). If using geemap, the instructor can show how to quickly visualize in a notebook.

Show that the composite has much less cloud (maybe none if done over enough time). Also mention you can do composite.reduceRegion(ee.Reducer.mean(), geometry=..., scale=10) to get average values, etc., just to hint at analytical capabilities.

* *Sentinel-1 in GEE:* Also demonstrate a Sentinel-1 query:
* s1 = ee.ImageCollection('COPERNICUS/S1\_GRD') \  
   .filterDate('2021-07-01', '2021-07-31') \  
   .filterBounds(some\_geometry) \  
   .filter(ee.Filter.eq('instrumentMode', 'IW')) \  
   .filter(ee.Filter.eq('orbitProperties\_pass', 'DESCENDING')) \  
   .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))  
  s1\_med = s1.median().clip(some\_geometry)
* Explain: *“We filtered Sentinel-1 to a specific month and area, chose IW mode and descending orbit (just as an example, maybe for a specific pass), and only VV polarization. Then we take a median composite (can reduce speckle).”* If possible, display it similarly (though interpreting SAR requires some stretch; maybe apply 10\*log10 and a color map, but okay to just mention it).
* *Clipping and Exporting:* The above examples used .clip(geometry) which restricts to AOI. Note to participants that clipping in GEE is a **remapping of pixel values to null outside the area, not reducing processing cost** per se, but it’s good for export. To **export**, illustrate:
* task = ee.batch.Export.image.toDrive(image=composite,  
   description='comp\_export',  
   folder='EarthEngineExports',  
   fileNamePrefix='ManilaComposite',  
   scale=10, region=geometry)  
  task.start()
* Explain that this will export the image to their Google Drive (folder EarthEngineExports) and they can download it later. They should monitor tasks in the GEE Code Editor or via task.status(). Since this might be slow, perhaps just show how it’s set up but not actually wait for completion in class.
* *Limitations and Next Steps:* Conclude by reminding participants that while GEE is powerful for data prep and certain analyses, there are times you need to download data and use other tools (like training a custom PyTorch model, which GEE can’t do on its servers). That’s why learning both GEE and Python is valuable – use each for what it’s best at. Also, mention that tomorrow they will dive into actually training a classifier (Random Forest) using either GEE or scikit-learn, bridging these skills.
* *Questions and common issues:* The instructor should anticipate some may have issues like “Earth Engine says my user is not whitelisted” (meaning they didn’t sign up – hopefully sorted beforehand) or “Memory limit exceeded” if they tried a huge export. Advise using reasonable AOIs and time ranges.

By the end of Day 1, the instructor reiterates the key takeaways: *Participants have learned where to get satellite data (Copernicus, local sources), how AI/ML can extract information from these data, and have practiced basic data handling both on their own machine (GeoPandas/Rasterio) and in the cloud (Earth Engine).* This sets a strong foundation for the more advanced modeling in the coming days.

## 3. Hands-on Google Colab Notebooks

To reinforce the lectures, Day 1 includes two interactive Colab notebooks that participants will run:

### Notebook 1: **Python Geospatial Data Handling** (GeoPandas & Rasterio)

**Objective:** Introduce participants to using Python for basic geospatial data manipulation – loading, inspecting, and visualizing vector and raster data – using libraries GeoPandas and Rasterio within Google Colab.

**Contents & Steps:**

* **Environment Setup:** The notebook begins with installing required libraries (e.g., geopandas, rasterio, maybe matplotlib for plotting). It also shows how to mount Google Drive. For example, a code cell:
* !pip install geopandas rasterio  
  from google.colab import drive  
  drive.mount('/content/drive')
* This ensures participants have the tools and data access. The notebook then navigates (if needed) to the data directory (e.g., a shared Drive folder containing sample data).
* **Loading Vector Data with GeoPandas:** The first data example is a **Philippine administrative boundaries** shapefile (small enough to handle, e.g., boundaries of regions or provinces). The notebook demonstrates:
* import geopandas as gpd  
  gdf = gpd.read\_file('Philippines\_Provinces.shp')  
  gdf.head()  
  print(gdf.crs)  
  print("Number of provinces:", len(gdf))
* This will output a table of the first few features and the coordinate reference system. Participants see that GeoPandas stores geometry and attributes. The notebook then instructs plotting:
* gdf.plot(figsize=(6,6), column='Region', legend=True)  
  plt.title("Philippines Provinces by Region")
* This produces a colored map of provinces by region. The notebook explains how the column parameter was used to color by an attribute, and how GeoPandas auto-selects a color map and adds a legend.

**Exercise:** The notebook might include a small exercise for learners, like *“Try changing the column to 'Island\_Group' or adjust the cmap.”* This encourages interactivity.

* **GeoDataFrame Operations:** Next, the notebook shows how to filter spatial data. For example:
* mindanao = gdf[gdf['Island\_Group']=="Mindanao"]  
  mindanao.plot(figsize=(5,5), color='orange')  
  plt.title("Mindanao Island Group Provinces")
* And similarly, how to get a single province geometry:
* davao = gdf[gdf['Province']=="Davao del Sur"]  
  print(davao.geometry.iloc[0]) # Print the polygon coordinates
* It might print a Polygon or MultiPolygon coordinates. The explanation emphasizes that we can treat the GeoDataFrame like a pandas DataFrame for filtering, and access geometries via the .geometry column or each row’s geometry attribute.
* **Loading Raster Data with Rasterio:** The notebook then moves to raster. It uses a **Sentinel-2 image tile (or subset)** covering a sample AOI in the Philippines (kept small, e.g., a 100 km² area, to reduce file size, perhaps stored as a Cloud-Optimized GeoTIFF or a pre-cropped TIFF). Example:
* import rasterio  
  src = rasterio.open('sample\_S2.tif')  
  print(src.crs, src.width, src.height, src.count, src.transform)
* This displays the projection (e.g., EPSG:32651 for UTM zone if in Philippines), raster dimensions, number of bands, and affine transform. Then:
* band1 = src.read(1)  
  print(band1.shape, band1.dtype, band1.min(), band1.max())
* This reads the first band (say Blue band) as a numpy array and prints shape and value range. The notebook explains that read(1) gives band 1; if the image has 3 bands, read(3) might be SWIR in Sentinel-2, etc.

**Visualization:** To plot a single band:

import matplotlib.pyplot as plt  
plt.imshow(band1, cmap='gray')  
plt.colorbar(label='Reflectance')  
plt.title("Sentinel-2 Band1 (Coastal Aerosol)")

Or to plot an RGB:

rgb = np.dstack([src.read(4), src.read(3), src.read(2)]) # B4,B3,B2 = RGB  
plt.imshow(np.clip(rgb \* 0.0001, 0, 1)) # assuming reflectances in 0-10000 scale, scale to 0-1  
plt.title("True-color Composite")

The notebook would clarify any scaling applied (Sentinel-2 L2A DN values need scaling by 1e-4, etc.). The output image should show a reasonably colored patch of land.

* **Spatial Referencing and Plot Overlays:** Show how to overlay vector boundaries on the raster for context. Since matplotlib can plot both, example:
* fig, ax = plt.subplots()  
  plt.imshow(np.clip(rgb \* 0.0001, 0, 1), extent=src.bounds, origin='upper')  
  mindanao.boundary.plot(ax=ax, edgecolor='yellow') # plot outlines of Mindanao provinces  
  plt.title("Sentinel-2 with province boundaries")
* The extent=src.bounds and origin='upper' ensure the image is placed in correct coordinates. This illustrates combining data sources.
* **Raster Cropping and Masking:** The notebook then demonstrates cropping the raster to a vector AOI using rasterio’s mask:
* from rasterio.mask import mask  
  geom = davao.geometry.iloc[0] # polygon of Davao del Sur  
  out\_image, out\_transform = mask(src, [geom], crop=True)  
  print(out\_image.shape) # should be (bands, new\_height, new\_width)
* Now out\_image contains the pixel values just for that province. They can plot this subset similarly. The notebook notes that mask sets values outside the polygon to nodata and returns a smaller window covering the polygon. It also likely retrieves src.meta and updates it for the new transform and dimensions if it were to save the cropped image:
* out\_meta = src.meta.copy()  
  out\_meta.update({"height": out\_image.shape[1],  
   "width": out\_image.shape[2],  
   "transform": out\_transform})  
  rasterio.open('davao.tif', 'w', \*\*out\_meta).write(out\_image)
* (Though writing to file in Colab is optional, it shows how to save results.)
* **Basic Raster Calculation:** If time/space permits, include a simple calculation, e.g., compute NDVI from bands:
* nir = src.read(8) # Band 8 is NIR for Sentinel-2  
  red = src.read(4) # Band 4 is Red  
  ndvi = (nir.astype(float) - red.astype(float)) / (nir + red)  
  plt.imshow(ndvi, cmap='RdYlGn')  
  plt.colorbar(label='NDVI')
* Show an NDVI image where green indicates vegetation. This ties back to AI – using band math to create features.

Throughout Notebook 1, markdown cells explain what each step is doing and why. The tone is tutorial-like: assume the user is following along and encourage them to inspect outputs. By the end, they have performed end-to-end steps: from reading raw data to making a simple map or analysis, all within Python.

**Outputs/Deliverables:** The notebook doesn’t produce a formal report, but participants will have generated maps and arrays. Key deliverables are the experience and code snippets which they can reuse. The notebook will be provided to them (via GitHub or Drive) so they can refer back. Instructors should ensure that any path or data needed is provided (or use geopandas to directly fetch data from a URL if possible).

### Notebook 2: **Google Earth Engine Python API for EO Data**

**Objective:** Teach participants how to use the Earth Engine Python API in Colab to find, filter, process, and download satellite images (Sentinel-1 and Sentinel-2) with basic pre-processing like cloud masking and compositing.

**Contents & Steps:**

* **Earth Engine Initialization:** The notebook starts with:
* !pip install earthengine-api  
  import ee  
  ee.Authenticate()  
  ee.Initialize()
* With instructions (in markdown) about the authentication process (click link, get code, paste). After initialization, they are ready to call EE functions.
* **Defining an Area of Interest (AOI):** The notebook provides a sample AOI, for example a rectangle over a part of Luzon or a specific city. Possibly:
* aoi = ee.Geometry.Rectangle([120.9, 14.5, 121.1, 14.7]) # bounding box around Metro Manila
* or use a FeatureCollection like ee.FeatureCollection("USDOS/LSIB\_SIMPLE/2017") filtered for Philippines for a country boundary – but a smaller AOI is better for quick results.
* **Searching Sentinel-2 ImageCollection:** They then create a Sentinel-2 ImageCollection query:
* s2\_col = ee.ImageCollection('COPERNICUS/S2\_SR') \  
   .filterBounds(aoi) \  
   .filterDate('2021-06-01', '2021-08-31') \  
   .filter(ee.Filter.lt('CLOUDY\_PIXEL\_PERCENTAGE', 50))  
  print("Images count:", s2\_col.size().getInfo())
* The markdown explains each filter. They then select only relevant bands (maybe all 12 reflectance bands except QA, or just the visible/NIR for simplicity) using .select([...]).
* **Cloud Mask Function:** Introduce a function to mask clouds in Sentinel-2:
* def mask\_clouds(image):  
   qa = image.select('QA60')  
   # Bits 10 and 11 are clouds and cirrus  
   cloud\_bit\_mask = (1 << 10) | (1 << 11)  
   mask = qa.bitwiseAnd(cloud\_bit\_mask).eq(0)  
   return image.updateMask(mask)
* Then apply: s2\_clean = s2\_col.map(mask\_clouds). A short explanation: *“The QA60 band’s bits 10 and 11 indicate cloudy pixels; this mask retains only pixels where those bits are 0 (no cloud).”*
* **Create a Median Composite:**
* median\_img = s2\_clean.median().clip(aoi)
* The notebook notes this takes the per-pixel median across the collection date range, yielding one cloud-free image. They then visualize or download this composite:
* **Visualization in Colab:** Perhaps use folium via geemap (if introduced) or get a thumbnail:
* url = median\_img.getThumbURL({'region': aoi, 'min':0, 'max':3000, 'bands':['B4','B3','B2']})  
  from IPython.display import Image  
  Image(url=url)
* This should display a small true-color thumbnail of the composite. The markdown might show the output or instruct users to open it.
* **Downloading the Composite:** Show how to export the image to Google Drive:
* export\_task = ee.batch.Export.image.toDrive(\*\*{  
   'image': median\_img,  
   'description': 'Sentinel2\_composite\_JunAug2021',  
   'folder': 'CopPhilTraining',  
   'fileNamePrefix': 'S2Composite\_Manila\_2021',  
   'region': aoi.getInfo()['coordinates'],  
   'scale': 10,  
   'crs': 'EPSG:4326'  
  })  
  export\_task.start()
* Explain that this will save the image in their Drive (the user will have to manually download from Drive). Mention that tasks take a few minutes; they can check status with export\_task.status() or in the GEE web UI. (Since it’s a small region and median, it might finish quickly).
* **Sentinel-1 Example:** The notebook then goes through a similar flow for Sentinel-1:
* s1\_col = ee.ImageCollection('COPERNICUS/S1\_GRD') \  
   .filterBounds(aoi) \  
   .filterDate('2021-06-01', '2021-06-30') \  
   .filter(ee.Filter.eq('instrumentMode', 'IW')) \  
   .filter(ee.Filter.eq('orbitProperties\_pass', 'DESCENDING')) \  
   .filter(ee.Filter.eq('resolution\_meters', 10)) \  
   .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))  
  print("Sentinel-1 count:", s1\_col.size().getInfo())
* This filters June 2021, IW mode, descending orbit, VV polarization only. The notebook explains these filters (mention dual polarization is often VV+VH; here we take only VV for demonstration). They then take a mean or median:
* s1\_img = s1\_col.mean().clip(aoi)
* Because SAR has speckle, averaging multiple scenes can reduce noise. They then visualize:
* url = s1\_img.getThumbURL({'region': aoi, 'min': -20, 'max': 0})  
  Image(url=url)
* where -20 to 0 dB might be typical backscatter range. The resulting grayscale image shows SAR backscatter over the area.

If time permits, show how to apply a simple threshold or edge detection on SAR (not necessary, but for fun: e.g., water has low VV backscatter, so threshold at -15 dB could highlight water – but perhaps beyond scope).

* **Downloading Sentinel-1 data:** They could demonstrate how to use ee.batch.Export.image.toDrive similarly for the SAR composite. Or mention using the ASF DAAC for raw data but that’s outside of GEE.
* **Using Earth Engine Data Catalog:** Show that one can search for datasets programmatically. Perhaps:
* ee\_catalog = ee.data.getList({'id': 'COPERNICUS'}) # fetch info on Copernicus datasets  
  for entry in ee\_catalog:  
   print(entry['id'])
* This might print available dataset IDs. Or encourage them to use the Docs tab in the Code Editor or the online catalog. The idea is to make participants aware of the huge variety of data (Landsat, MODIS, etc.) beyond Sentinel.
* **Additional GEE functionalities:** Briefly mention that GEE can do a lot more:
* Compute statistics: e.g., median\_img.reduceRegion(ee.Reducer.mean(), aoi, scale=10) to get mean reflectance over AOI.
* Time series: one can chart NDVI over time by iterating or using reduceRegions.
* Machine learning: Earth Engine also offers some built-in ML (Cart, SVM, randomForest via ee.Classifier) which they might use on Day 2. The notebook might not delve deep into these, but perhaps includes one example of a reduceRegion or adding NDVI band:
* ndvi\_img = median\_img.normalizedDifference(['B8','B4']).rename('NDVI')
* and visualizing NDVI.

**Guidance Provided:** The notebook is well-commented, and each section has markdown explaining what’s happening. It might also include cautionary notes like *“Be mindful of the size of your AOI and date range; too large might be slow or time out.”* It encourages exploration – maybe tasks like *“Change the date range to a different season and see how the Sentinel-2 composite colors change (indicative of phenology).”* Or *“Try increasing cloud percentage filter to get more images – does the composite improve or degrade?”*

**Outcome:** After Notebook 2, participants will have: - Queried and filtered satellite image collections in Earth Engine. - Applied a cloud mask on-the-fly for Sentinel-2. - Created a cloud-free composite. - Exported an image to their drive for offline use. - Done a similar process for Sentinel-1 (without cloud issues, but with other filters). They should appreciate how quickly they got a useful result (like a 3-month cloud-free mosaic) that would be painful to do manually downloading dozens of images. This hands-on directly supports use cases in DRR and NRM: e.g., generating baseline imagery for an area, which is a common first step.

**Note:** The notebooks will be made available via GitHub or the course site, with all required data either included or linked. The sample data chosen (e.g., a small Sentinel-2 TIFF and a shapefile) will also be provided so that even outside of Colab, participants can run the notebooks (if they set up locally with the same libraries).

## 4. Datasets for Day 1

To facilitate the hands-on exercises and case examples, a set of **sample datasets** (small and readily downloadable) will be provided:

* **Philippines Administrative Boundaries:** A shapefile (or GeoPackage) of Philippine administrative boundaries, suitable for vector exercises. We will use the official level-2 (province) or level-3 (municipality) boundaries from PhilGIS or HDX (Humanitarian Data Exchange) which are derived from PSA/NAMRIA datasets. For example, *Philippines\_Provinces.shp* containing all provinces with attributes (name, region, etc.). This dataset is only a few MB. *Source:* HDX provides admin level 0-4 boundaries openly.
* **NAMRIA Basemap (Excerpt):** Instead of the entire basemap (which is huge), we’ll supply either a small raster extract or demonstrate use of NAMRIA’s Web Map Service. The **Geoportal Basemap** is accessible via WMS/WMTS; a guide is available on how to consume it in QGIS. For the training, we might not need to download it, but we will include info in the handouts on how to connect to it. If needed, a static image (JPEG/PNG) of the basemap for a region can be given just for reference.
* **Sentinel-2 Sample Tile:** A cloud-free Sentinel-2 image or composite covering a Philippine area of interest. To keep size small, we might use:
* A single 100 km² tile in GeoTIFF (10 m resolution, 3 bands). For instance, a tile covering Metro Manila or Mayon Volcano. We can use an official L2A product and crop it. Alternatively, use a lower-resolution product like a Level-3 mosaic or the ESA WorldCover base map for demonstration.
* Example: **“Sentinel2\_L2A\_Luzon\_sample.tif”** – ~50 MB file with bands 2,3,4 (visible) and 8 (NIR) included. This is provided so participants can practice Rasterio on it.
* Alternatively, a pre-generated cloud-free mosaic of the Philippines (e.g., NextGIS provides mosaics) could be used, but those might be large. We prefer something small, even if it’s just a portion.
* **Sentinel-1 Sample:** To illustrate SAR, we might include a small Sentinel-1 GRD snippet. Possibly a 20x20 km GeoTIFF of VV backscatter over Laguna de Bay or Cagayan River. This could be ~10 MB. If not providing as file, we show how to get it from GEE (as we did in Notebook 2). Given time constraints, an actual file may not be needed if GEE is used directly.
* **Land Cover Map (for reference):** The 2020 NAMRIA land cover map (raster or vector) at national scale is large (~ hundreds of MB). Instead, we could include a simplified version (e.g., one region or a generalized raster). However, since it’s mostly for context, we might skip including the file and just point participants to the NAMRIA geoportal for later use. In the exercise, when we discuss classification, we might compare against known land cover in an area qualitatively.
* **Google Earth Engine data (online):** No need to download GEE datasets, but ensure participants have access. We will list some asset IDs in the course materials, e.g.,
* COPERNICUS/S2\_SR (Sentinel-2 surface reflectance collection),
* COPERNICUS/S1\_GRD (Sentinel-1 GRD collection),
* maybe USGS/SRTMGL1\_003 (SRTM 30m DEM) if elevation is used later,
* etc. These IDs and any filters used are documented so participants can reuse them.

All dataset links will be provided via a GitHub repository or cloud storage: - Shapefiles and raster samples in a ZIP on GitHub or Google Drive (to be downloaded in notebooks). - A text file with WMS service URLs for NAMRIA basemap and other Philippine geoservices. - Pointers to Copernicus Open Access Hub / Data Space for downloading Sentinel data manually (for those interested).

For example, in the handout we might include: - **Link:** HDX Philippine Admin Boundaries (levels 0-3). - **Link:** Wikimedia Commons image of Sentinel-2 over Mayon (for visualization) – as used above. - **Link:** Soar.Earth sample imagery of Mayon Volcano (false-color), which is a free sample tile. - **Link:** NextGIS cloud-free mosaic for Bicol Region (if they want a nice ready image).

These small datasets ensure that even if internet is limited, participants have something to work with offline or in the Colab environment without long downloads. They illustrate the kinds of data (administrative, optical imagery, radar imagery) that will be used throughout the training.

## 5. Participant Handouts (DOCX/Markdown)

To reinforce learning and for easy future reference, several handouts will be provided. These will be in both Word (DOCX) and Markdown (for easy viewing on the course site). Key handouts for Day 1:

* **Day 1 Schedule and Learning Objectives:** A one-page outline of the day. This includes a timed agenda:
* *9:00–9:30:* Introduction (Course overview, objectives, intro remarks)
* *9:30–11:30:* Session 1 – Copernicus Sentinel Data & PH EO Landscape
* *11:30–11:45:* **Break**
* *11:45–1:15:* Session 2 – AI/ML Fundamentals for EO
* *1:15–2:15:* **Lunch Break**
* *2:15–4:15:* Session 3 – Hands-on Python Geospatial Basics (Colab)
* *4:15–4:30:* **Break**
* *4:30–5:30:* Session 4 – Intro to Google Earth Engine (hands-on)
* *5:30–5:45:* Q&A and Wrap-up.

(Times can be adjusted as needed, but this gives structure.)  
Below the schedule, **learning objectives** are listed in bullet form: - *Understand* the goals of the Copernicus program in the Philippines and identify the main Sentinel satellites and their characteristics. - *Recognize* key Philippine agencies/platforms for Earth observation data (PhilSA, NAMRIA, DOST-ASTI projects) and how they can complement satellite data. - *Explain* fundamental AI/ML concepts (supervised vs unsupervised learning, neural networks basics) and the typical workflow to develop an EO application using ML. - *Perform* basic tasks in Python for geospatial data: reading and visualizing a map layer and satellite image. - *Use* Google Earth Engine to retrieve satellite imagery and apply simple preprocessing (filtering dates, cloud masking, composites).

This handout sets expectations and is something participants can quickly glance at to recall what was covered.

* **Summary of AI/ML Workflows for EO:** A 1-2 page summary that reiterates the workflow and key concepts from Session 2 in a concise form. It might have a simple flowchart graphic (if possible) and bullets for each step (Problem definition → Data collection → Preprocessing → Feature engineering → Training → Validation → Deployment). It will also define the types of ML and give an example in one sentence each:
* *Supervised Learning:* “learn from labeled examples to make predictions on new data (e.g., a classifier trained on known land cover pixels to map unknown areas).”
* *Unsupervised Learning:* “discover patterns or groups in unlabeled data (e.g., clustering an image into spectrally similar regions).”
* *Neural Network:* a diagram or description of input layer, hidden layers, output; mention of activation and training by adjusting weights.
* *Data-centric tips:* a checklist for data quality (e.g., check label accuracy, ensure variety in training data, normalize inputs, etc.).

Essentially a cheat-sheet of concepts introduced, so participants can review later without combing through slides.

* **Python Libraries Cheat Sheet:** Likely a two-part cheat sheet for GeoPandas and Rasterio (since those are new to many):
* **GeoPandas Cheatsheet:** covering common commands:
  + Reading data: gpd.read\_file('file.shp').
  + Viewing data: gdf.head() and gdf.plot() (and how to specify column, legend).
  + CRS: gdf.crs and gdf.to\_crs(epsg=4326).
  + Basic spatial ops: intersect, within, buffer (maybe just mention, not covered deeply yet).
  + Filtering: examples of attribute filter (gdf[gdf['FIELD']=='value']).
  + Merging dataframes (maybe not for Day 1, could be Day 3 topic, but could list).
  + *This cheat sheet* could be adapted from existing ones, simplified for our context.
* **Rasterio Cheatsheet:** common patterns:
  + Opening a file: rasterio.open('file.tif') and reading meta (.crs, .count, .shape).
  + Reading arrays: src.read(1) etc.
  + Displaying image with matplotlib.imshow (remembering to set extent or use plt.imshow directly for small array).
  + Masking: rasterio.mask.mask(src, [geom], crop=True).
  + Writing: using with rasterio.open(newfile, 'w', \*\*profile) as dst: dst.write(array).
  + Perhaps a note on data types and scaling (e.g., Sentinel-2 DN to reflectance). This cheat sheet helps participants recall syntax without searching docs.
* **Google Earth Engine Cheat Sheet:** A reference for GEE (in Python API). Could include:
* How to initialize (auth + ee.Initialize()).
* Geometry creation: ee.Geometry.Point([lon,lat]), ee.Geometry.Rectangle([...]).
* ImageCollections: ee.ImageCollection('ID'), filtering methods (filterDate, filterBounds, filterMetadata).
* Image: band selection (image.select('B4')), math operations (image.normalizedDifference([B8,B4]) for NDVI).
* Common reducers: reduceRegion with ee.Reducer.mean() etc., imageCollection.mean(), median(), max().
* Masking: image.updateMask(mask) (with example of mask creation from QA band).
* Export: ee.batch.Export.image.toDrive with required parameters. Since Google’s own documentation has cheat sheets and a “Beginner’s Cookbook”, we will condense relevant parts. Possibly include a link to the official cheat sheet or cookbook for further reading.

We’ll also highlight a couple of Earth Engine **tips**: - All operations are lazy/evaluated in the cloud, you need to .getInfo() or export to bring data locally. - Use .getInfo() sparingly (small objects) because it can hang if the object is large (like an image). - Earth Engine has a lot of datasets – link to the Data Catalog.

* **Day 1 Q&A/Key Points:** Possibly a handout summarizing key questions discussed or common pitfalls. This might be compiled after the session to address any confusion that arose. For instance:
* *Q: What’s the difference between Level-1C and Level-2A Sentinel-2 data?* – A: (Explain briefly).
* *Q: Can we use Google Earth Engine without coding?* – A: yes, with the Code Editor’s GUI and existing scripts, but this training focuses on coding to unlock more flexibility.
* *Q: Is there a way to get historical data beyond satellites?* – mention that aside from Copernicus, there’s Landsat etc. – possibly note these for curiosity.

All handouts will be written in a clear, concise manner with bullet points, short paragraphs, and possibly small graphics or tables. They serve as both in-class aids and post-class reference materials.

## 6. Web Hosting Solution for Course Materials

To publish all materials (notes, slides, notebooks, handouts) as an interactive course site, we recommend using **Jupyter Book or Quarto**, hosted via GitHub Pages, with embedded links to Colab and downloadable content. We also consider **Docusaurus** as an alternative. Here’s an evaluation of each:

* **Jupyter Book (Executable Books):** This platform is designed for publishing collections of content like lecture notes, notebooks, and markdown as a coherent static website. We can organize the Day 1 to Day 4 content into chapters. Jupyter Book supports **direct integration of Jupyter Notebooks** – you can include notebooks that will be executed or display outputs. It also conveniently adds interactive buttons like “Run in Colab” or “Run in Binder” on pages with notebooks. This means participants browsing the course site can immediately open a notebook in Google Colab by clicking a button, which is exactly our use case. Jupyter Book uses a Markdown/Markdown+MyST format, so our content (written in Markdown) can be incorporated with minimal friction. It also supports embedding images and search functionality. Technical setup: it’s based on Sphinx, and GitHub Pages can deploy it easily. One advantage is that **executable code** can be kept up-to-date; for example, we could allow the site to execute notebooks periodically or allow readers to execute code on Binder. In short, Jupyter Book excels for educational textbooks and courses with code.

*Trade-offs:* Theming is somewhat basic (but sufficient for our needs), and writing content requires learning MyST Markdown for advanced features (though basic Markdown works). Since our audience might use the site primarily to access materials and launch notebooks, Jupyter Book’s features align well. It also allows insertion of slides (we can embed Google Slides via iframe or provide link). Overall, for a technical training, Jupyter Book offers a good balance of ease and functionality.

* **Quarto:** Quarto is an open-source publishing system that can create blogs, books, or documentation sites from Markdown, Jupyter notebooks (.ipynb), or Quarto Markdown (.qmd). It supports output to HTML, PDF, and even slides. Using Quarto, we could maintain our content in notebooks/markdown and render to a polished website. Quarto has native support for Jupyter notebooks – it can execute them and embed outputs in the site. It also allows embedding interactive visualizations. A big plus: Quarto can generate **Reveal.js slides** from markdown, which means we could even convert our Google Slides content to markdown and have Quarto publish them as an slideshow on the site (though we might simply link the Google Slides if that’s easier). Quarto’s integration with Jupyter is quite smooth and it’s language-agnostic (works with Python, R, etc.). The learning curve for Quarto is not steep for someone familiar with Jupyter or RMarkdown. Quarto can be deployed to GitHub Pages similarly. Also, Quarto can leverage Jupyter Book-like features (in fact Quarto can be seen as a next-gen of RMarkdown/Bookdown that also covers Jupyter use cases).

*Trade-offs:* Quarto is relatively new (as of 2022/2023) but rapidly gaining adoption. It might require the course maintainer to install Quarto CLI to build the site. The site generation is straightforward but if customization beyond the default theme is needed, one might have to dig into configurations. That said, Quarto’s default look is clean, and it supports features like latex, code folding, etc., out-of-the-box. Quarto vs Jupyter Book: both would meet our needs, but Quarto might allow a bit more flexibility (like seamlessly including slides and notebooks together). If the team is comfortable with it, Quarto could be chosen for its modern approach.

* **Docusaurus:** This is a React-based static site generator commonly used for documentation websites. It supports content written in Markdown (and MDX, which allows embedding React components in Markdown). For our course, we could set up a Docusaurus site with sections for each day. Markdown pages can contain our lecture text or handouts. For Jupyter notebooks, Docusaurus doesn’t directly render .ipynb. We’d have to convert notebooks to Markdown/MDX (possibly using tools like nbconvert or Quarto integration) and then include them. There are community plugins and approaches (like using nbdev or nbdoc) that generate MDX from notebooks and embed outputs, which then Docusaurus can host[[2]](https://outerbounds.com/blog/technical-docs-with-docusaurus-and-notebooks#:~:text=and%20tested%20for%20many%20scenarios%3A,needed%20the%20following%20additional%20capabilities). This means it’s feasible to include our code outputs (as static images or interactive via some custom React components). Docusaurus has excellent theming and a fancy look; it also has a search and multi-language support, versioning (useful if the course is repeated and updated).

We can link to Colab easily by adding a button or link on pages (just HTML/JS in MDX). Also embedding an iframe of a slide deck or a map is doable in MDX.

*Trade-offs:* Docusaurus is not specifically made for executing or rendering Jupyter content, so it’s more manual. We’d basically treat our content as documentation. The setup requires Node.js and familiarity with React/JS if we want to heavily customize. For a small team, maintaining a Docusaurus site might be overkill compared to Jupyter Book/Quarto which are more writer-friendly for technical content. However, if we want a highly polished site or integration with a larger documentation system, Docusaurus could shine. It’s also worth noting that Docusaurus can handle assets and linking well, so providing downloads (like our datasets) through the site is straightforward.

In summary, **Jupyter Book** is likely the most straightforward solution for an interactive course site given our content. It allows us to keep everything in markdown/notebooks and generates a cohesive site with navigation, search, and interactive notebook launchers. **Quarto** is a strong alternative, offering similar features and possibly easier slide integration. **Docusaurus** offers superior web design but at the cost of extra complexity in notebook integration.

**Recommendation:** Use **Jupyter Book** to publish the course materials on GitHub Pages. Organize by Day (with Day 1 as a section containing subpages: lecture notes, instructor notes, labs, handouts). Embed **“Open in Colab”** links next to each notebook so participants can easily launch them. Include our slide decks as either embedded iframes or downloadable PDF. Jupyter Book will handle the Markdown and notebook content well, and the site can be kept private or public as needed via GitHub.

We will document in the README how to contribute (if needed) or how the site is built. This approach ensures the course is accessible during the training and afterward, contributing to the **Digital Space Campus** initiative mentioned (so participants can revisit materials and new learners can self-study). With everything on GitHub, it’s also easy to update for future iterations of the training.

*Sources:* This write-up drew on the CopPhil programme description, training agenda document, as well as external references for technical specifics like Sentinel missions and tools (GeoPandas, GEE) to ensure accuracy and currency. The guidance on web publishing platforms references official documentation and community experiences with Jupyter Book, Quarto, and Docusaurus to weigh their suitability for our needs. All materials have been tailored to the Philippine context and the CopPhil training goals of empowering local users in DRR, CCA, and NRM through AI/EO.

[[1]](https://commons.wikimedia.org/wiki/File:Mount_Mayon_Sentinel-2_L1C_from_2018-01-30_(28280091769).jpg#:~:text=) File:Mount Mayon Sentinel-2 L1C from 2018-01-30 (28280091769).jpg - Wikimedia Commons

<https://commons.wikimedia.org/wiki/File:Mount_Mayon_Sentinel-2_L1C_from_2018-01-30_(28280091769).jpg>

[[2]](https://outerbounds.com/blog/technical-docs-with-docusaurus-and-notebooks#:~:text=and%20tested%20for%20many%20scenarios%3A,needed%20the%20following%20additional%20capabilities) Testable Technical Documentation with Notebooks, nbdev, and Docusaurus | Outerbounds

<https://outerbounds.com/blog/technical-docs-with-docusaurus-and-notebooks>