Epistem-X Technical Documentation

IIASA

2026-06-09

Table of contents

About the project

The Epistem initiative aims to develop an open source landscape monitoring technology to empower actors involved in efforts to avoid deforestation, forest/landscape restoration, and other nature-based solutions (NbS). These actors cover a wide range of institutions whose needs and usage for landscape monitoring data represent diverse needs to address landscape degradation and restoration. They typically suffer from the lack of accessible data such as land cover map, change map, as well as ground-truthing/reference datasets that are fundamental for planning, mobilizing funds, monitoring and evaluating accountability of NbS implementation. Furthermore, even when monitoring tools exist, making sure they integrate into real-world NbS implementations and address the specific operational hurdles faced by users on the ground remains a substantial challenge. In response to these pressing NbS data needs, we are developing Epistem-X, a landscape monitoring technology designed with the following key characteristics.

Epistem-X

Introduction

This document is a technical specification for the Epistem-X workflow. It is intended for back-end engineers, remote sensing scientists, data engineers, and DevOps to implement a production-ready mapping pipeline.

System Overview

The envisioned landscape monitoring system is designed to comprise three core components, as illustrated in Figure 1. These components work in an integrated manner to provide a robust, scalable, and flexible solution for generating high-quality Land Use and Land Cover (LULC) maps. By combining advanced algorithms, structured reference data, and user-friendly interfaces, the system will support diverse stakeholders in monitoring, managing, and making informed decisions about landscapes.

LULC Classification Algorithm

[insert brief summary of all the modules]

Reference Data Repository

[insert brief summary of Epistem-Y]

User Interface

[insert UI description]

Epistem-X Development Phases

Epistem-X will progress through a structured development lifecycle consisting of three phases. Phase 1 is targeted for completion by December 2025, followed by Phase 2, which is planned for delivery by June 2026.

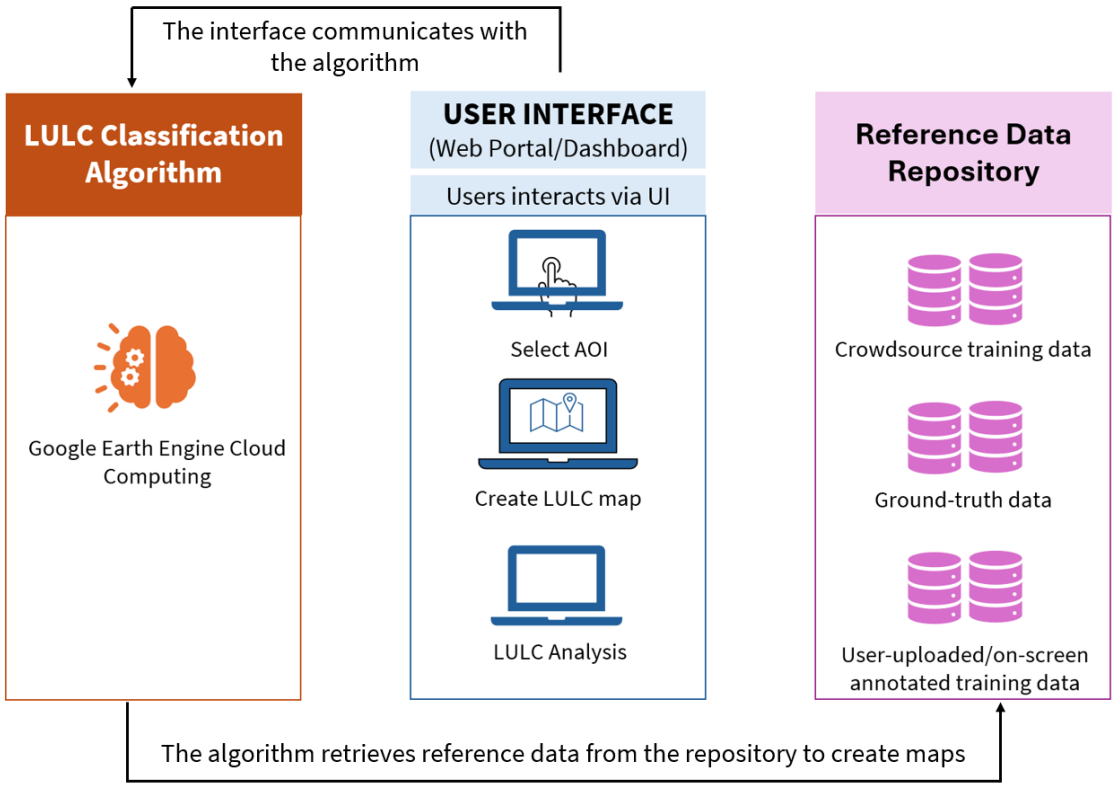


Figure 1: System architecture and core components of the LULC monitoring system

Phase 1: Express Map

> Phase 1 User's Focus
>
> Users seeking to create their first land cover map quickly, easily, and with high-quality results.

Phase 1 serves as the primary entry point for all Epistem users. The platform provides a guided workflow designed to produce standardized, high-quality maps with minimal complexity. Users follow a core set of steps to achieve results without navigating complex options.

[in this development phase, what can the user do with this tool? what are the inputs/outputs?]

Phase 2: Map Studio

> Phase 2 User's Focus
>
> Users with foundational knowledge seeking more complex mapping, experimenting with additional datasets, and refining map details according to specific needs.

Phase 3: Map Lab

> Phase 3 User's Focus
>
> Users requiring in-depth analysis, collaborative workflows, and the ability to learn from other Epistem users.

Modules Overview

The workflow for generating LULC maps consists of eight sequential modules:

Module 1: Generating cloud-free satellite images

Module 2: Determining the LULC classification schema and classes

Module 3: Obtaining representative training data

Module 4: Performing separability analysis and selecting covariates

Module 5: Incorporating non-image covariates to improve model accuracy

Module 6: Generating the preliminary LULC map

Module 7: Assessing thematic accuracy using independent validation data

Module 8: Performing post-classification analysis and data housekeeping

Each module builds on the previous one, ensuring a systematic and reproducible workflow from raw imagery to a validated, fully documented LULC map.

1 Module 1: Acquisition of Near-Cloud-Free Satellite Imagery

Module Overview

The output of this module is high-quality, near-cloud-free satellite imagery, which is essential for producing accurate and reliable land use and land cover (LULC) datasets. This module focuses on acquiring satellite imagery that has been pre-processed and corrected, including procedures such as cloud masking and shadow removal.

Input

User's Input

1. Area of interest (AOI).
2. Selection of single or multiple time-series imagery.
3. Target map year (YYYY).
4. Desired spatial resolution (meters).
5. Maximum allowable cloud cover (%).

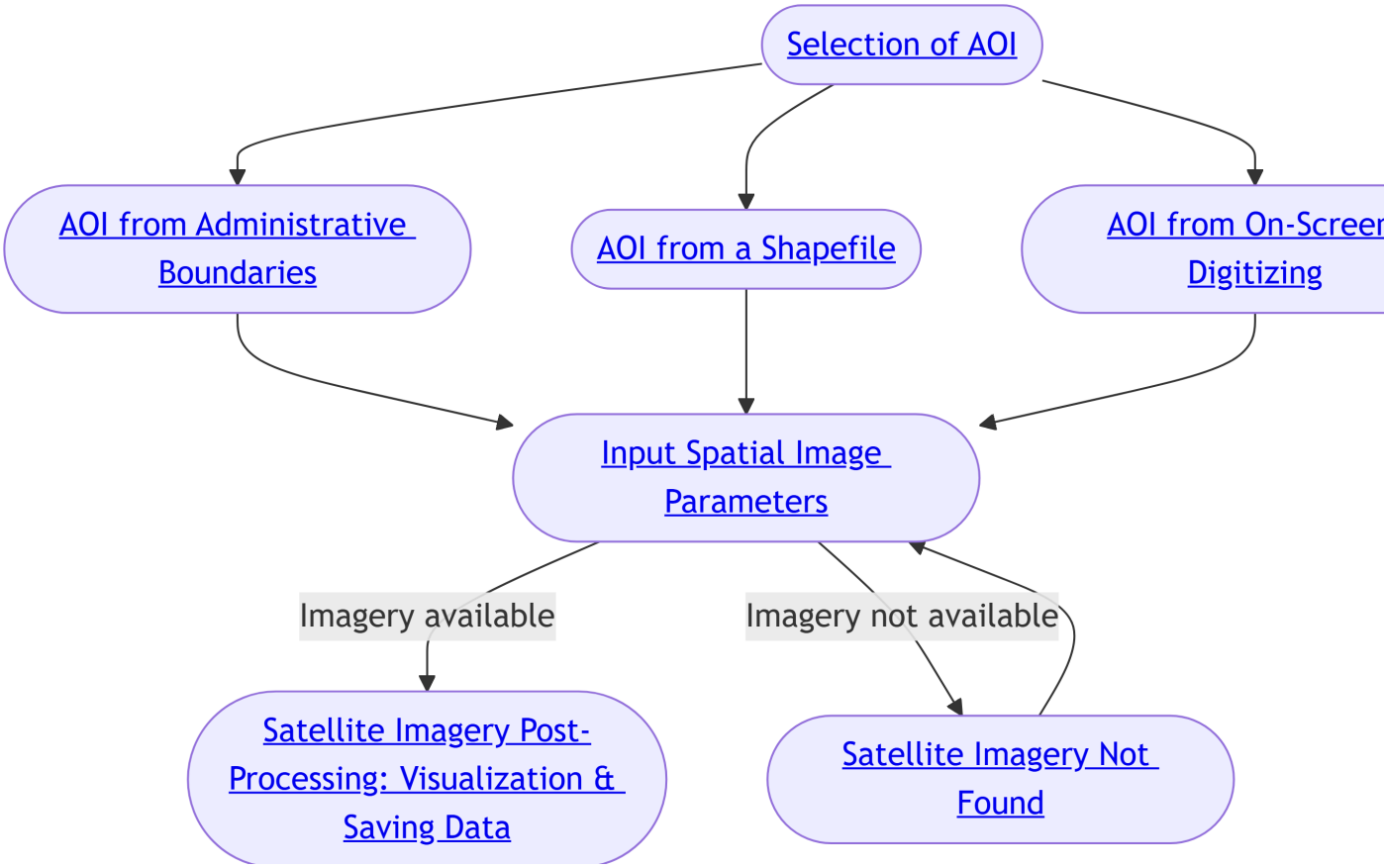
Automatic System Input

1. Raw satellite imagery.
2. Administrative boundaries.

Output

1. Selected AOI.
2. Near-cloud-free satellite imagery.

Process



1.1 Selection of AOI

> **Note**
>
> Section 1.1.1, Section 1.1.2, Section 1.1.3 are options.

1.1.1 AOI from Administrative Boundaries

Front-end

The user selects an administration boundary.

Back-end

The system queries the FAO Global Administrative Unit Layers (GAUL) 2015 datasets, which provide standardized information on country and sub-national boundaries, and saves the AOI as a variable

> Related function
>
> `from epistemx.helpers import get_aoi_from_gaul`

1.1.2 AOI from Shapefile

Front-end

1. The user can upload their `.SHP` file. in `.ZIP` format that includes all necessary files (`.SHP`, `.SHX`, `.DBF`, `.PRJ`).
2. The user will get a feedback such as "AOI conversion completed!", "Failed to convert AOI to Google Earth Engine Format", "Geometry Validation Failed", "Error Reading Shapefile".

Back-end

1. The system currently supports only shapefile in `.ZIP` format.
2. An Upload button is provided, allowing user to uploaded the file. It is then extracted to a temporary directory.
3. Next, the system validates and repairs the shapefile geometries (if any errors are found) in a GeoDataFrame (`.GDF`), converts the cleaned geometries into an Earth Engine objects, and converts the coordinate system into WGS 1984.
4. If an error occurs still, the system displays an error message and suggests performing a shapefile simplification process using GIS software.
5. The system displays a small preview map centered on the AOI.

1.1.3 AOI from On-Screen Digitizing

Front-end

The user is presented with an interactive on-screen digitization process.

Back-end

[not developed yet]

1.2 Selecting and Preparing Satellite Imagery

1.2.1 Input Spatial Image Parameters

Front-end

The user is prompted to fill in these parameters:

1. Spatial resolution (meters): Specify the desired spatial resolution.
2. Input year: Choose either single-date or multi-year imagery.
3. Maximum cloud cover: Set a preferred cloud cover threshold, or use the default value of 30%.

Back-end

1. The system currently supports Landsat 1-4 at sensor radiance and Landsat 5-9 collection 2 surface reflectance Analysis Ready Data (ARD), excluding the thermal bands. Landsat mission availability is as follows:

    - Landsat 1 Multispectral Scanner/MSS (1972 - 1978)
    - Landsat 2 Multispectral Scanner/MSS (1978 - 1982)
    - Landsat 3 Multispectral Scanner/MSS (1978 - 1983)
    - Landsat 4 Thematic Mapper/TM (1982 - 1993)
    - Landsat 5 Thematic Mapper/TM (1984 - 2012)
    - Landsat 7 Enhanced Thematic Mapper Plus/ETM+ (1999 - 2021)
    - Landsat 8 Operational Land Imager/OLI (2013 - present)
    - Landsat 9 Operational Land Imager-2/OLI-2 (2021 - present)

2. The system retrieves images that match user's selected criteria, including spatial resolution, year, and cloud cover. For single-year imagery, the system selects Landsat images from December 31st of year T or the lowest cloud cover scene available that year. For multi-year imagery, the system retrieves the lowest cloud cover image for each year in the specified range.

3. The system crops and mosaics the selected satellite imagery to match the user's AOI.

4. The system performs cloud masking in three steps: cloud detection, mask refinement, and image compositing. The QA_PIXEL band encodes pixel conditions using specific bit flags. `Bitwise AND` operations are applied to isolate these conditions, focusing on clouds (bit 4, value 16) and cloud shadows (bit 3, value 8). Pixels matching these flags are excluded to produce a cleaner, near cloud-free image.

> Related function
>
> `def mask_landsat_sr(image, cloud_conf_thresh=2, shadow_conf_thresh=2, cirrus_conf_thresh=2):`

5. The system evaluates whether the retrieved imagery meets the required spatial resolution, input year, and cloud cover thresholds.
6. If there is no satellite imagery available for the user's input parameters, proceed to Section 1.2.2

> Related function
>
> `from epistemx.module_1 import Reflectance_Data`

1.2.2 Satellite Imagery Not Found Based on User's Parameters

Front-end

The user is prompted to adjust parameters, such as increasing the cloud cover limit, selecting a different time period, or choosing a different spatial resolution.The user will then get a feedback: "No images found for the selected criteria, increase cloud cover threshold or change the date range".

Back-end

1. Before prompting the user to insert new parameters, the system searches imagery from year T to T-2 to replace cloudy areas via image compositing.
2. If requirements remain unmet, Section 1.2.1 will re-run based on new user parameters.

1.3 Satellite Imagery Post-Processing

1.3.1 Visualization and Saving Data

Front-end

1. The user can visualize the near cloud-free imagery within the web portal and download it as a GeoTIFF (`.TIF`)
2. Prior to initiating the image download, the user inputs technical parameters of the image that includes: file name, coordinate system, and spatial resolution.
3. If the user chooses not to download it, they may proceed to Module 2.
4. A summary report containing the metadata is provided as a `.TXT` file that includes:
    - Total images found
    - Data range of images
    - Unique WRS tiles
    - Path row tiles
    - Scene IDs
    - Image acquisition dates
    - Average scene cloud cover
    - Date range
    - Cloud cover range

Back-end

1. The system produces near-cloud-free Landsat imagery for the defined AOI, saves it as a variable, and visualizes the processed datasets.
2. A summary report of metadata is generated, including acquisition details, cloud cover statistics, and a summary of processing steps applied.

Related function

```
from epistemx.module_1 import Reflectance_Stats
def get_collection_statistics(collection, compute_stats=True,
print_report=False):
```

2 Module 2: Determining LULC Classification Schema and Classes

Module Overview

This module offers a flexible framework for generating Land Use/Land Cover (LULC) maps tailored to various applications, from broad assessments like deforestation monitoring to detailed analyses reflecting specific management practices or prioritized conservation areas. In the initial phase, only non-hierarchical schemas will be implemented. All classes are treated independently, making it suitable for single-level applications such as deforestation mapping or identifying priority conservation areas.

The output of this module is a well-defined list of LULC classes. The module also manages the selection, input, validation, and storage of LULC classification schemes. Users can select a predefined system schema, upload their own classification file, or enter the classification manually. The module ensures data integrity and provides a downloadable classification table for downstream analysis.

The module incorporates RESTORE+ LULC classifications from Indonesia to support restoration planning, guiding identification of degraded areas and management priorities. These classes are offered as a predefined data classification system.

Input

User's Input

1. Default LULC class list.
2. Manually entered LULC class list.
3. Uploaded LULC class list.

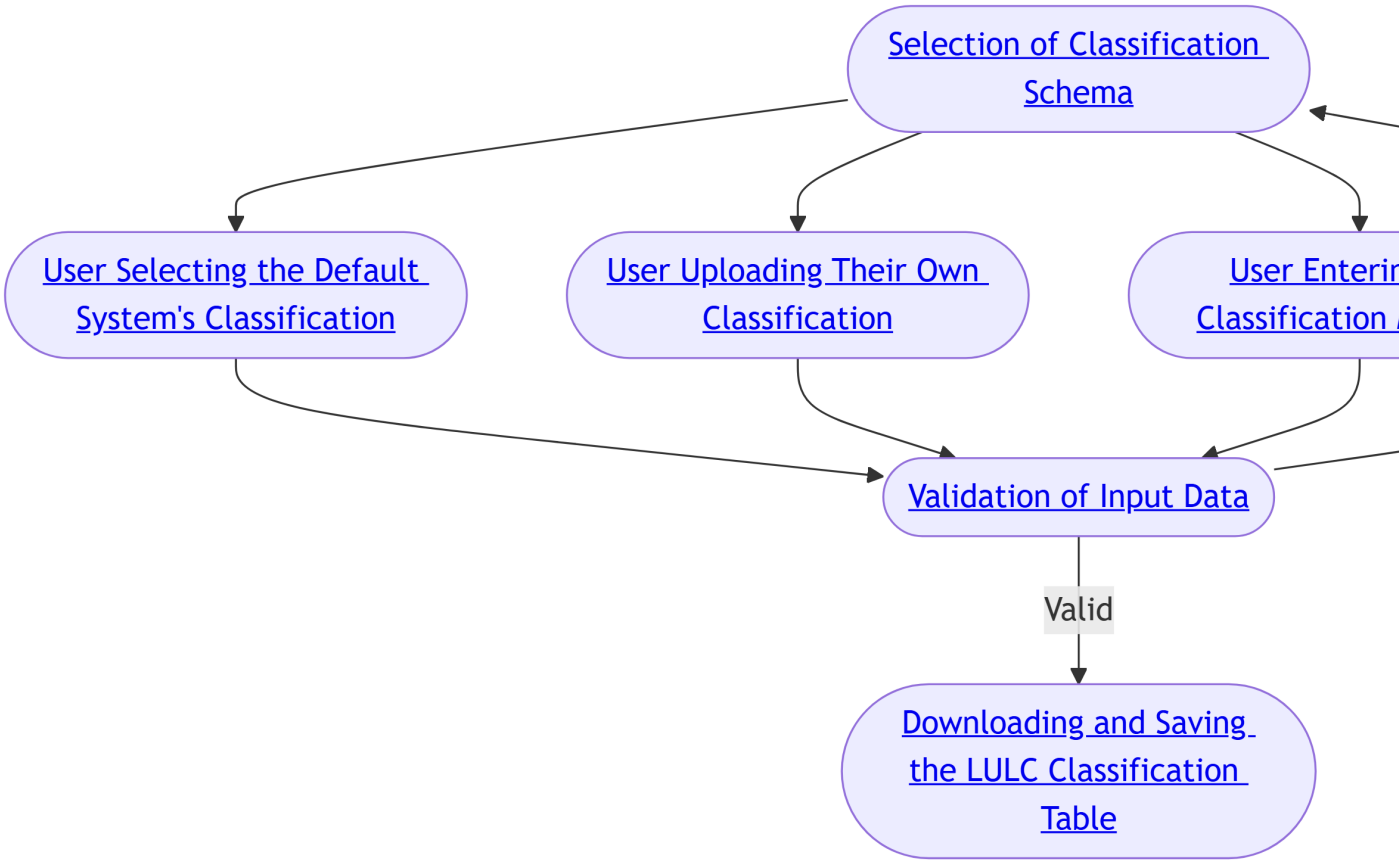
Automatic System Input

1. FAO LULC classes.
2. RESTORE+ LULC classes.

Output

LULC classification table

3 Process



3.1 Selection of Classification Schema

Front-end

The user has the option to use the system's default classification schema, upload a file containing their own classification scheme, or enter the classification manually.

Back-end

1. If the user chooses the default schema, the system proceeds to Section 3.1.1
2. If the user chooses to upload their own schema, the system proceeds to Section 3.1.2
3. If the user chooses to enter the classification manually, the system proceeds to Section 3.1.3

3.1.1 User Selecting the Default System's Classification

Front-end

If the default classification schema is selected, the user chooses a list of LULC classes from either the RESTORE+ or LCCS dataset.

Back-end

1. The system loads the default N classes from the database and displays the classification in tabular format with an N-class template containing the attributes: ID, Class, and Color Code.
2. The system then records the user's selection in its internal database table.

3.1.2 User Uploading Their Own Classification

Front-end

1. The user uploads the `.XLSX`, `.XLS`, or `.CSV` file.
2. If there is an error, the user receives a warning and is prompted to re-upload the corrected file or format.

Back-end

1. The system displays an Upload button.
2. The system validates the uploaded file, checking:
   - The file is not corrupt.
   - The format is valid.
   - The file size is within the allowed limit.
   - The file is not password-protected.
3. The system then validates and extracts data from sheets, cells, rows, and columns, converting it into a Pandas DataFrame. It checks whether the data is present or empty.
4. If the validation is successful, the system displays a preview of the data in a table and provides a verification result notification.

3.1.3 User Entering the Classification Manually

Front-end

1. The user fills out the input form containing ID, Class, and Color Code fields, and optionally the Description column.
2. The system warns the user if the form is incomplete or data conversion fails.

Back-end

1. The system displays the input form and checks that all required columns are filled.
2. Once the input form is converted into a Pandas DataFrame, the data are previewed in a table with a verification status.

3.2 Downloading and Saving the LULC Classification Table

3.2.1 Downloading and Saving the LULC Classification Table

Front-end

1. The user can download the table in Excel file format and save it in their local drive.
2. The system confirms the download; the user then proceed to Module 3.

Back-end

1. The system provides a button to download the result file.
2. The classification table is saved as a variable for use in subsequent modules.

4 Module 3: Training Data Generation

Module Overview

In supervised classification, the quality of training data directly determines the accuracy of the resulting model. The output of Module 3 is to produce georeferenced point vector datasets for each LULC class defined by the user in the preceding step, Module 2, accompanied by quantitative and visual summaries that facilitate rigorous evaluation, quality control, and optimization for subsequent classification. The module supports two workflows for data preparation: (1) importing user-defined datasets, or (2) interactively generating samples using system-guided tools. Upon completion, the user obtains a structured datasets for each LULC class, together with statistical diagnostics to assess the quality and representativeness of the data prior to classification.

Input

User's Input

1. User-specified options for generating training datasets.
2. User selection of the sampling method.
3. User selection of the approach for collecting training points.
4. User-specified the data splitting percentage.

Automatic System Input

1. Training datasets from the RESTORE+ project.
2. Publicly available datasets from the Epistem-X repository system.

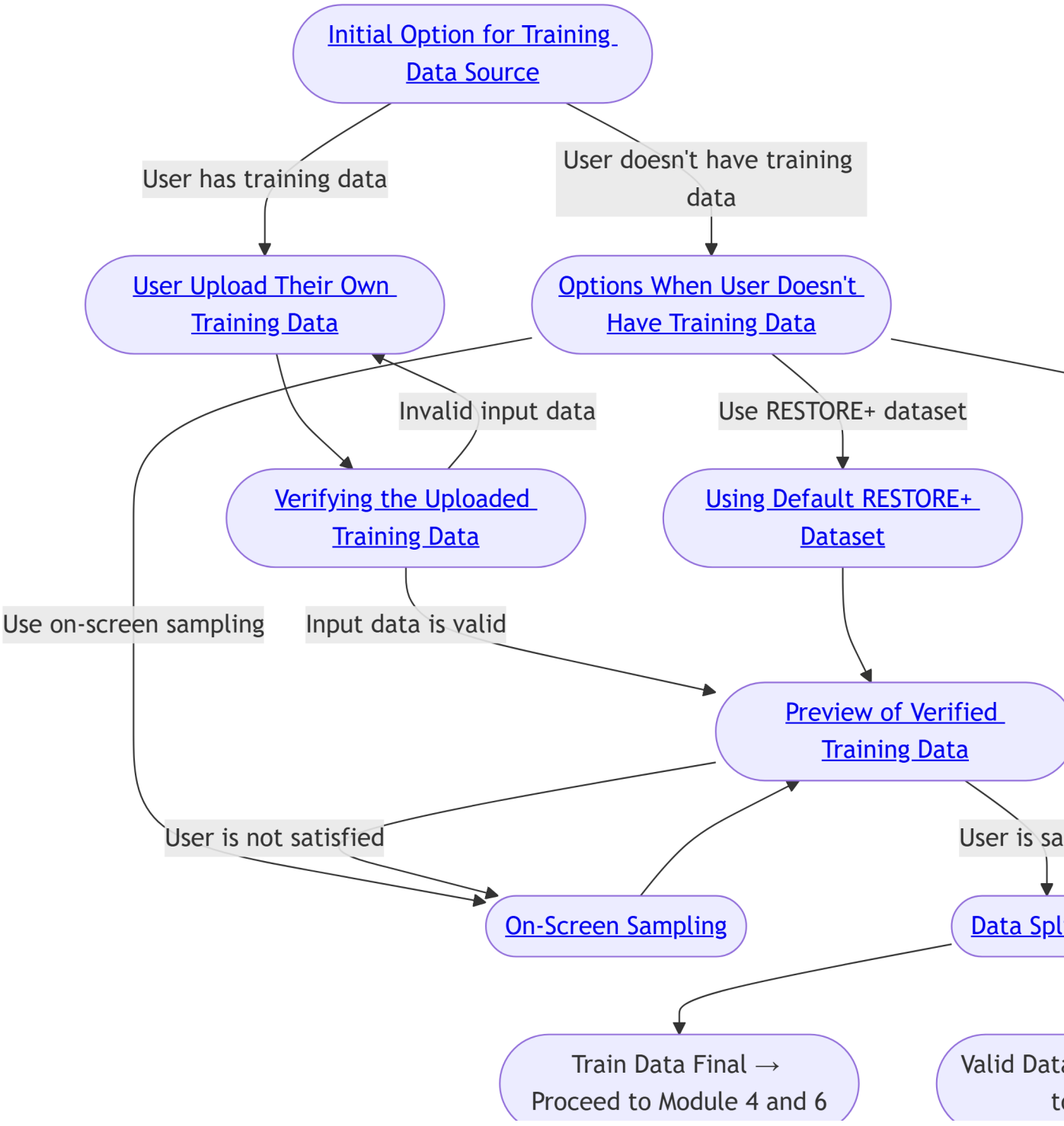
Input from Other Modules

1. User-selected AOI (Section 1.1).
2. Near-cloud-free satellite imagery (Section 1.3).
3. LULC classification table (Section 3.2).

Output

- Georeferenced training datasets
- Georeferenced validation datasets.
- Statistical analyses of the datasets.

Process


Initial Option for Training Data Source
User has training data
User doesn't have training data
User Upload Their Own Training Data
Options When User Doesn't Have Training Data
Invalid input data
Use RESTORE+ dataset
Verifying the Uploaded Training Data
Using Default RESTORE+ Dataset
Use on-screen sampling
Input data is valid
Preview of Verified Training Data
User is not satisfied
User is sa
On-Screen Sampling
Data Spl
Train Data Final → Proceed to Module 4 and 6
Valid Data
t

4.1 Checking Prerequisites from Previous Modules

Front-end

If the user has not completed Module 1 and Module 2, the system displays a warning message prompting them to finish those modules first.

Back-end

The system checks whether the default system's classification is selected in Module 2.

1. If yes, the system proceeds directly to Section 4.4.
2. If no, the system verifies the availability of output products from previous modules. If the required products are not available, the system notifies the user to complete the previous modules before proceeding.

4.2 Training Data Generation

4.2.1 Initial Option for Training Data Source

Front-end

1. The system initiates the workflow by providing the user with two primary options:
   - "I have training data".
   - "I don't have training data".
2. Users that have training data who created their own classification model in Module 2 can either upload their data or/and enter it manually (Phase 1).
3. Users without training data can retrieve it directly from the system public repository.

Back-end

1. If the user chooses to upload their own training data, the system will continue to Section 4.2.2.
2. If the user does not have their own training data, the system will continue to **?@sec-user-does-not-have-training-data**.

4.2.2 User Uploading Their Own Training Data

Front-end

1. The user uploads one of the supported file types.
2. The user chooses Yes or No whether they want to add more training data through on-screen sampling.

Back-end

1. The system provides an upload interface that accepts only `.SHP` and `.ZIP` file (containing point features with LULC class attributes) or `.CSV`/`.XLS`/`.XLSX` format (containing point coordinates and associated LULC class labels established in Module 2). The system will then check if the uploaded data fulfills the requirement. If discrepancies are detected, the system will flag the inconsistent classes. The user can then either:
    - Re-upload corrected training data or
    - Add the missing or inconsistent samples through the system's on-screen sampling interface (Section 4.2.4).
2. If the user chooses to add training data with on-screen sampling, the system will proceed to Section 4.2.4, with additional uploaded samples appearing on the interface.
3. If the user does not want to add training data with on-screen sampling, the system will analyze the training data and proceed to Section 4.4.

4.2.3 User Does Not Have Training Data

Front-end

The user is given the following options to proceed:

1. Use on-screen sampling.
2. Use RESTORE+ default data.
3. Use Epistem-X public repository.

Back-end

1. If the user uses on-screen sampling, the system will proceed to Section 4.2.4.
2. If the user uses RESTORE+ default data, the system will proceed to Section 4.2.5.
3. If the user uses Epistem-X public repository, the system will proceed to Section 4.2.6.

4.2.4 On-Screen Sampling

Front-end

1. The User Interface (UI) provides an interactive environment for visual sampling on a basemap.
2. A two-way binding mechanism ensures that every sampling action on the basemap is immediately reflected in the panel, and vice versa.

Back-end

1. The system will display a panel containing a table with the following columns:
    - Land cover class name.
    - Number of identified samples.
    - Button to add points on the basemap.
2. The user selects the land cover class in the table to which sample data will be added:
    - The user performs on-screen sampling (adding, removing, or relocating points).
    - The system updates the "number of identified samples" column in real time while the user performs on-screen sampling.
3. The system provides a button to save the results of the on-screen sampling, which the user can click once sufficient data has been added and stores it as a variable `OnScreenSampledReferenceData`.

4.2.5 Using Default RESTORE+ Datasets

Front-end

The system provides a "Use Data" button to define the RESTORE+ datasets as the sample data variable.

Back-end

The system recalls RESTORE+ datasets and displays the datasets in a panel containing this information: LULC class and number of identified samples inside the AOI.

4.2.6 Using Epistem-X Repository

Front-end

The user selects a dataset from the repository and closes the panel. The user accesses public training datasets from the Epistem-X repository and can:

1. Browse and select point-based training data.
2. Choose specific coordinates for relevant regions.
3. Select predefined models or spectral signature libraries.
4. Apply these publicly available models/signatures to support the classification process.

A "Use Data" button is provided to define the selected public datasets as the sample data variable.

Back-end

The system displays the Epistem-X public sample data repository in a panel. Each dataset is presented in a table with at least three columns: data location (shown as a map inset), land cover class name, and the number of identified samples.

4.3 Verifying the Uploaded Training Data

Front-end

1. The user receives a notification indicating the status of their uploaded sampling data:
    - Sufficient - the user can proceed to Section 4.4
    - Insufficient - the number of sample points for one or more classes is inadequate. The user can either:
        - Add more training data and proceed to Section 4.2.4, or
        - Skip adding data and proceed directly to Section 4.4 in which case a notification warns that the classification result may have low accuracy.

Back-end

1. The system analyze the sample data to determine:
    - The list and count of unique classes.
    - The number of sample points per class.
2. Verification for Module 1 and 2 includes:
    - Identifying samples located outside the AOI.
    - Comparing the number of unique classes between datasets.
    - Detecting classes missing in either datasets.
    - Flagging classes with insufficient sample points (<20 points).
    - Identifying duplicate samples in the same pixel
3. The system filters out sample data that are either located outside the AOI and do not match the classes defined in Module 2.

4. Informational notifications are provided for: LULC classes located outside the AOI, not matching those defined in Module 2, and with missing or insufficient sample points.
5. Next, the system evaluates the adequacy of sample data quantity for each class.

4.4 Preview of Verified Training Data

Front-end

The user reviews the `TrainDataRecap` displayed in a table.

Back-end

The system presents a recap table with columns for ID, class, number of points, and percentage.

4.5 Data Splitting

Front-end

The user selects one of the following options for splitting the `TrainDataRecap`:

1. Use all sample points for training, or
2. Split sample points into training and validation, specifying the desired percentage.

Back-end

The system provides options for splitting the sample point data for training and validation:

1. Use all sample points for training - the system consolidates all data into a single DataFrame or variable, stored as `TrainDataFinal` for use in Module 4 and 6.
2. Split the sample points between training and validation - the system partitions the data according to the user-specified percentage defined in the `TrainSplitPct` variable. The `TrainDataFinal` (for training) is stored in Module 4 and 6, while `ValidDataFinal` (for validation) is stored in Module 6.

5 Module 4: Spectral Separability Analysis

Module Overview

This module enables users to evaluate the spectral separability of sample classes using Jeffries–Matusita (JM) distance or Transformed Divergence (TD). It provides quantitative metrics for class distinction and visual tools for intuitive interpretation. By identifying well-separated classes and informative covariates, this analysis helps ensure data quality and supports more accurate classification or modeling workflows.

Input

User's Input

1. Separability analysis parameters.
2. User-specified spectral bands for visualization.

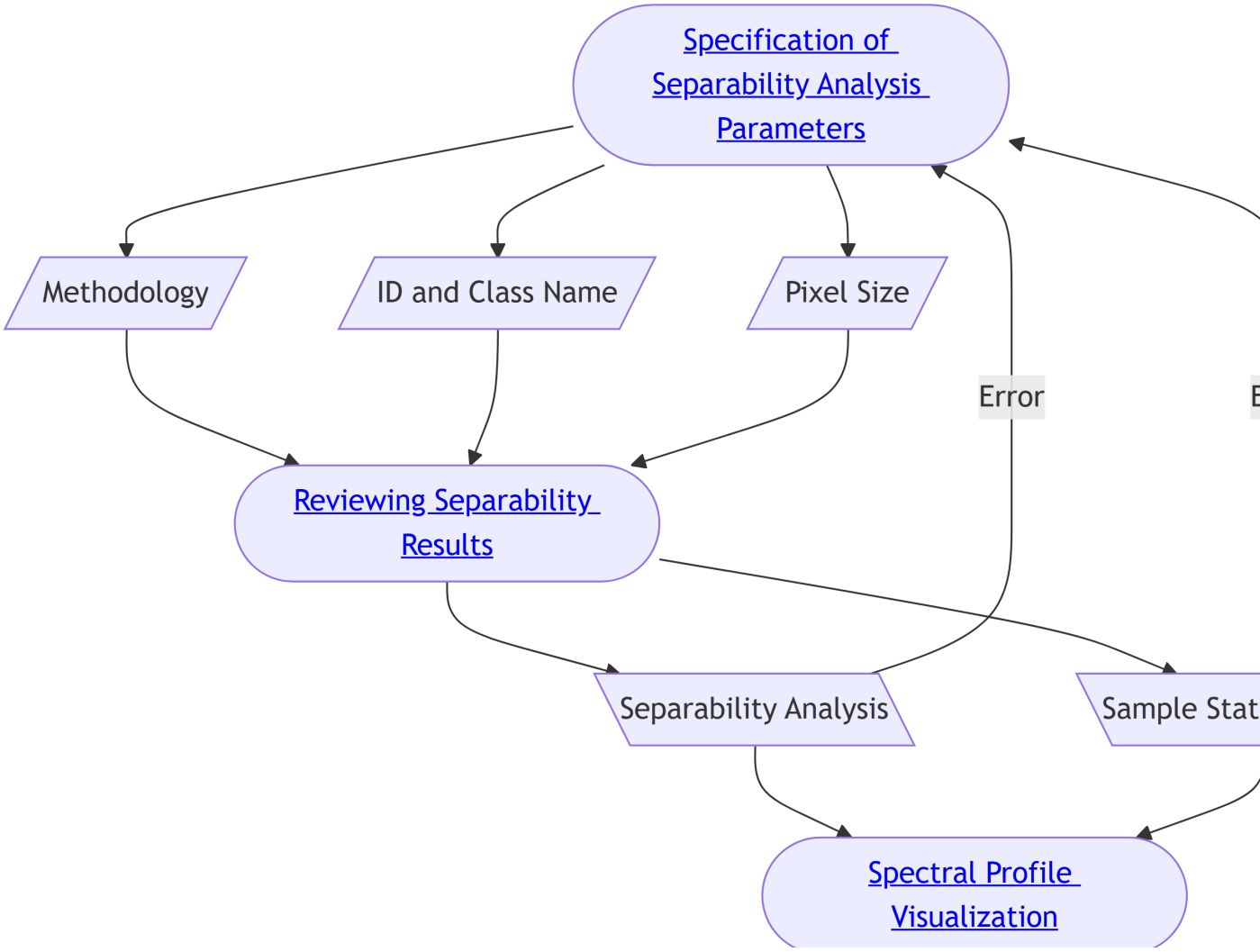
Input from Other Modules

1. Near-cloud-free satellite imagery within the AOI (Section 1.3).
2. `TrainDataFinal` (Section 4.5).

Output

1. Basic statistics (mean, median, variance, and standard deviation).
2. Separability values.

Process


Specification of Separability Analysis Parameters
Methodology
ID and Class Name
Pixel Size
Error
Reviewing Separability Results
Separability Analysis
Sample Stat
Spectral Profile Visualization

5.1 Specification of Separability Analysis Parameters

5.1.1 Specification of Separability Analysis Parameters

Front-end

1. The user defines the parameters and selects the method for separability analysis.
2. Inputs include ID, Class Name, and pixel size.

Back-end

The system provides three parameters:

1. Methodology: Jeffries-Matusita or Transformed Divergence.
2. ID and Class Name as a drop down selection.
3. Pixel Size as numeric input.

5.2 Reviewing Separability Results

5.2.1 Reviewing Separability Result

Front-end

1. If input parameters are incomplete, the user receives an error message.
2. If inputs are complete, the user evaluates whether the separability results are satisfactory.
    - If unsatisfactory, the user may refine samples for classes with low separability by adding or removing points through Module 3 or external editing.
    - If satisfactory, the user may proceed to spectral profile visualization (Section 5.3) .

Back-end

1. The system validates input parameters and prompts the user to review any incomplete entries.
2. If it is incomplete, the system displays an error and prompts the user to review the analysis parameters.
3. The system executes the separability analysis workflow that consist of two processes:
    - Sample Statistics Calculation.
    - Separability Analysis.
4. The system executes the workflow as follows:

- Calculates the sample proportion for each class.
- Extracts pixel values based on sample locations and the selected imagery.
- Computes basic statistics (mean, median, variance, and standard deviation) per class and stores them in a `DataFrame` for visualization.
- Runs the separability analysis using `TrainDataFinal`.
- For each pair of classes, the system calculates separability metrics, identifies the most difficult class pairs to distinguish, and categorizes results into three levels:
  - Good separability (>1.8): Classes are well separated.
  - Weak/Marginal separability (1.0-1.8): Classes overlap partially.
  - Class confusion (<1.0): Classes require improvement.
- Results, along with the basic statistics table, are exported as a `.TXT` report.
- Detailed error messages are provided if any errors occur.

5.3 Spectral Profile Visualization

5.3.1 Spectral Profile Visualization

Front-end

1. The user selects the spectral bands for visualization.
2. If satisfied with the result, the user proceed to the next module.
3. If not satisfied, the user can return to Module 3 to modify the `TrainDataFinal`.

Back-end

1. The system checks the status of the separability analysis:
   - If successful, visualization proceeds.
   - If unsuccessful, visualization is blocked and the user is prompted to complete the parameter specification (Section 5.1).
2. A visualization canvas and required Python libraries are provided to support:
   - Histogram
   - Box plot
   - Scatter plot (2D and 3D)
3. Helper Text guides the user in interpreting the plots effectively.

6 Module 5: Improving Model Quality with Multi-Source Data

Module Overview

> **Caution**
>
> This module is not available in Phase 1

This optional module enables users to refine the model based on feedback from Module 4. It supports the integration of additional covariates to enhance model accuracy and robustness. Users can incorporate additional covariates, including radar, spectral, and non-spectral types (For details please see Section 6.3). The updated configuration facilitates the generation of a higher-quality model compared to the previous iteration.

Input

User's Input

1. User-selected covariate set

Automatic System Input

1. Default covariates.
2. Radar-based covariates.
3. Spectral transformations covariates.
4. Non-spectral covariates.

Input from Other Modules

1. Near-cloud-free satellite imagery.
2. LULC classification table.

Output

1. List of covariates and descriptive table.
2. Variables as a covariate stack.
3. Variables as extracted features.
4. Table/chart of variable importance/influence.

Process

6.1 Selection of Covariate

6.1.1 Selection of Covariate

Front-end:

1. The user selects a list of covariates according to the classification objectives and receives system recommendations.
2. If a Default Covariates is chosen, the user can either:
   - Accept the default covariates set to proceed with retraining the model, or
   - Modify the covariates list, which redirects to the covariate customization options.
3. If a Manual Covariates is chosen, the user selects the three covariates options with system guidance.

Back-end:

1. The system provides two selection modes:
   - System Default Covariates: Retrieves and displays a default set of covariates used in the classification model, including metadata (e.g., description, intended use).
   - Manual Covariate: Displays categorized options:
     - Radar-based covariates.
     - Spectral transformations.
     - Non-spectral covariates.
2. The system also provides recommendations for effective covariate use.

6.2 System Default Covariates

6.2.1 System Default Covariates

Front-end:

1. The user reviews and optionally adjusts the default covariate list.

Back-end:

1. The system displays the default covariates and allows user modifications.
2. Provides information on the trade-offs involved in adjusting covariate set.

6.3 Manual Covariates

6.3.1 Manual Covariates

Front-end:

1. The user reviews a detailed overview of selected covariates and can adjust them through customization options.

Back-end:

1. The system summarizes the chosen covariates before returning to the workflow from Module 4.
2. Detailed covariate categories include:
    - Radar-based covariates: Includes raw radar backscatter (Sentinel-1, PALSAR), temporal and seasonal composites, and derived features (ratios, texture, vegetation indices).
    - Spectral transformations that consists of four parts:
        - Built-up/Soil indices: OSAVI (Optimized Soil-Adjusted Vegetation Index), MSAVI (Modified Soil Adjusted Vegetation Index), NDBI (Normalized Difference Built-up Index), SAVI (Soil Adjusted Vegetation Index).
        - Vegetation indices: GBNDVI (Green-Blue Normalized Difference Vegetation Index), ARVI (Atmospherically Resistant Vegetation Index), CVI (Chlorophyll Vegetation Index), RVMI (Red-Edge Vegetation Moisture Index), GNDVI (Green Normalized Difference Vegetation Index), NDVI (Normalized Difference Vegetation Index), EVI (Enhanced Vegetation Index).
        - Water or water-related indices: AWEI (Automated Water Extraction Index), MNDWI (Modified Normalized Difference Water Index), NDMI (Normalized Difference Moisture Index), NDWI (Normalized Difference Water Index), MNDWI (Modified NDWI).
        - Burn index: NBR (Normalized Burn Ratio).
    - Non-spectral covariates: Includes terrain metrics (elevation, slope, aspect) and distance metrics (to roads, coastlines, and settlements).
3. The system highlights the trade-offs between covariate selection and model training efficiency.

6.4 Model Evaluation and Optimization Feedback

6.4.1 Model Evaluation and Optimization Feedback

Front-end:

1. The user evaluates model performance: "Improved" or "Not Improved".
    - If Improved, the system outputs the LULC classification model with the optimized covariate set.
    - If Not Improved, the user reviews the optimized covariates and decides whether to apply them to generate the LULC classification model.

Back-end:

1. After adjustment, the system prompts the user to evaluate: *"Has model quality now improved?"* If No:
    - The system provides recommendations for covariate ranking analysis and reselection.
    - Covariate optimization is performed to identify the most influential variables.
2. Once finalized, the system proceeds with model retraining using the optimized covariate set.

7 Module 6: LULC Map Generation

Module Overview

This module provides functionality for users to generate the final LULC map as the primary output product, derived from the validated and optimized classification model developed in Modules 1-5. Before the processing begins, it ensures that all required inputs - such as satellite imagery, classification schemes, and training samples - are properly prepared and validated. Users can specify data splits, select classification modes, perform hyperparameter tuning optimization, and apply the trained model to the entire AOI to produce a high-quality, georeferenced LULC map. The module supports Random Forest (RF) based classification for adaptability, aligning with hierarchical or non-hierarchical schemes from Module 2, and incorporates feedback from optional covariate enhancements in Module 5. Upon completion, users receive a displayable, analyzable, and downloadable map with metadata, enabling applications such as deforestation monitoring or conservation planning.

Input

User's Input

1. Agreement to proceed after input review (yes/no).
2. RF hyperparameters: number of trees (`n_tree`), variables per split (`var_split`), minimum leaf population (`min_leaf_pop`).

Automatic System Input

1. None.

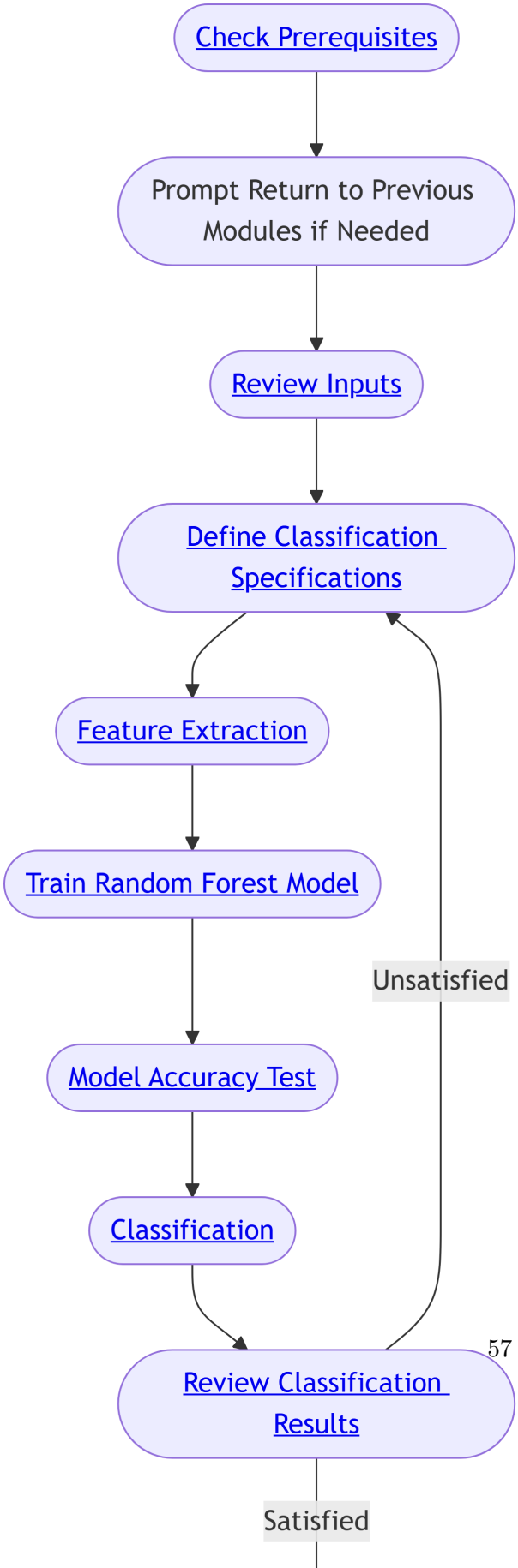
Input from Other Modules

1. Near-cloud-free satellite imagery within the AOI (Module 1).
2. LULC classification scheme table (Module 2).
3. Georeferenced training dataset (Module 3).
4. Validation dataset (Module 3).
5. Validated sample data and separability results (Module 4).

Output

- Trained RF classification model.
- Classified LULC raster map.
- Model accuracy report if validation data available (including overall accuracy, precision, recall, F1-score, Kappa, model error matrix).

Process


Check Prerequisites
Prompt Return to Previous Modules if Needed
Review Inputs
Define Classification Specifications
Feature Extraction
Train Random Forest Model
Model Accuracy Test
Classification
Review Classification Results
Unsatisfied
Satisfied

7.1 Checking Prerequisites from Previous Modules

7.1.1 Front-end

The user is notified if inputs are insufficient or test data is missing, and prompted to return to previous modules (e.g., Module 3 for test data) as needed.

7.1.2 Back-end

1. The system performs a parameter and input suitability test to verify availability and compatibility of inputs from Modules 1-5 (such as imagery, scheme, training data, optional validation data).
2. Test data availability from Module 3.
3. If any inputs are missing or incompatible, the system directs the user to return to the incomplete module.
4. If no test data, suggest the user return to Module 3 and correct the sample.
5. The system directs the user to return to the module that has not been completed.

7.2 Model Training and Classification

7.2.1 Review Inputs

7.2.1.1 Front-end

1. The user reviews the summary text report to decide if they agree to proceed.
    - If yes, continue to define specifications.
    - If no, return to relevant previous modules.
2. Users return to previous modules as needed.

7.2.1.2 Back-end

1. The system generates and displays a summary report of variables for the classification process, including imagery details, classification scheme, and training data summary.
2. Displays a summary text and dataframe.
3. Lists variables used in classification.

7.2.2 Define Classification Specifications

7.2.2.1 Front-end

1. User enters RF hyperparameter values:
    - Number of trees (n_tree)
    - Variables per split (var_split)
    - Minimum leaf population (min_leaf_pop)

7.2.2.2 Back-end

1. Stores user-entered hyperparameters.
2. Applies them as main parameters for Random Forest training.

7.2.3 Feature Extraction

7.2.3.1 Front-end

Runs automatically after specifications are defined.

7.2.3.2 Back-end

Performs feature extraction by sampling pixel values from imagery based on training (and optional validation) samples.

7.2.4 Train Random Forest Model

7.2.4.1 Front-end

Shows user progress during training.

7.2.4.2 Back-end

1. Trains RF model using prepared data:
    - Bootstrap sampling for subsets per tree.
    - Builds multiple decision trees.
    - Uses majority voting for predictions.
2. Produces trained model asset.

7.2.5 Model Accuracy Test

7.2.5.1 Front-end

Displays accuracy report if validation data is available.

7.2.5.2 Back-end

1. Applies model to validation data if available.
2. Generates accuracy metrics: overall accuracy, precision, recall, F1-score, Kappa, error matrix.
3. Skips if no validation data.

7.2.6 Classification (Apply Model to Entire Image)

7.2.6.1 Front-end

Runs after training with progress shown.

7.2.6.2 Back-end

1. Applies trained model to entire imagery.
2. Generates classification summary and result data.

7.2.7 Review Classification Results

7.2.7.1 Front-end

1. User reviews model results and map.
    - If satisfied, proceed to Module 7.
    - If not, adjust parameters or return to prior modules.

7.2.7.2 Back-end

1. Displays model learning and accuracy results.
2. Suggests corrections or parameter checks if unsatisfied.

8 Module 7: Thematic Accuracy Assessment

Module Overview

This module enables users to perform a thematic accuracy assessment on LULC map generated in Module 6, comparing it against independent validation data to quantitatively evaluate its quality and reliability. It supports verification of prerequisites from Modules 1 to Module 6, allows confirmation or adjustment of data partitions and model configurations, and computes key metrics such as confusion matrices (overall accuracy (OA), Kappa coefficient ( ), producer's accuracy (PA), and user's accuracy (UA). The module provides phased outputs including numerical/visual results, improvement recommendations, and a spatial confidence level map, facilitating iterative refinements for enhanced map accuracy in applications like environmental monitoring or policy planning.

This module focuses on the evaluation and validation of LULC classification results derived from remote sensing data. The objective is to provide users with tools and guidance to assess the reliability of their classified maps and determine their suitability for further analysis. Upon completion, the user will have a validated LULC map and a clear understanding of its accuracy.

Input

User's Input

1. Upload of test data if not available from Module 3 (shapefile or CSV as per Module 3).
2. Button press to initiate accuracy test.

Automatic System Input

1. None.

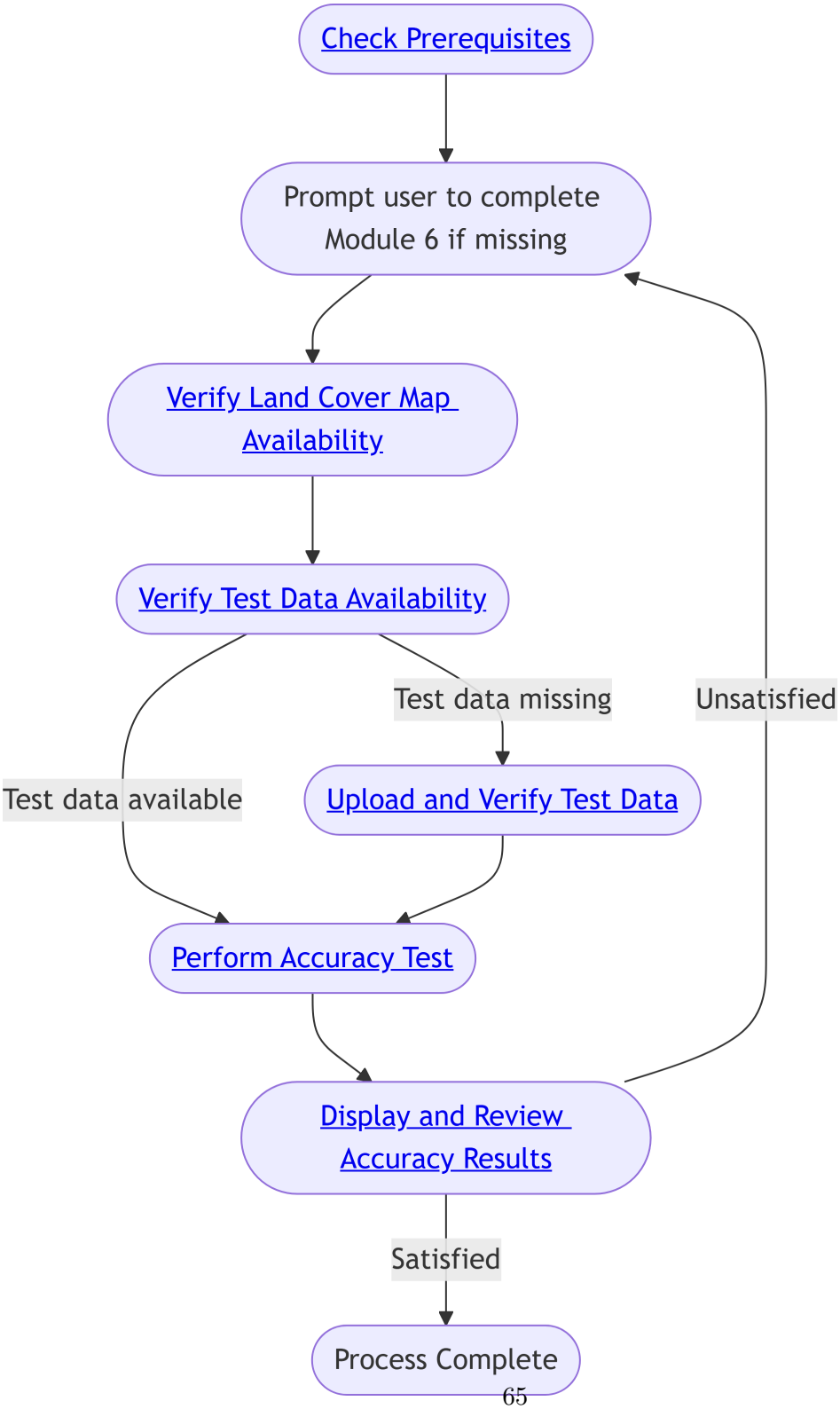
Input from Other Modules

1. Final land cover map (Module 6).
2. Partitioned test samples if available (Module 3).
3. Processed satellite imagery (Module 1, if needed for context).

Output

- Confusion matrix (table).
- Accuracy metrics: Overall Accuracy (OA), Kappa Coefficient, Producer's Accuracy (PA), User's Accuracy (UA).

8.1 Process


Check Prerequisites
Prompt user to complete Module 6 if missing
Verify Land Cover Map Availability
Verify Test Data Availability
Test data missing
Test data available
Upload and Verify Test Data
Perform Accuracy Test
Display and Review Accuracy Results
Unsatisfied
Satisfied
Process Complete

8.2 Checking Prerequisites from Previous Modules

Front-end

The user is notified if the map is missing and prompted to complete Module 6.

Back-end

1. Check for the final land cover map from Module 6.
2. If not available, provide notification to run Module 6.
3. Land cover map availability test.

8.3 Thematic Accuracy Assessment

8.3.1 Verify Land Cover Map Availability

Front-end

User is notified if map is missing and prompted to complete Module 6.

Back-end

1. Check for final land cover map from Module 6.
2. If missing, notify user to run Module 6.

8.3.2 Verify Test Data Availability

Front-end

1. If test data available, allow user to proceed with accuracy test.
2. Else, prompt upload.

Back-end

1. Check for test data from Module 3.
2. If available, proceed to accuracy test.
3. If not, prompt user for upload.

8.3.3 Upload and Verify Test Data

Front-end

1. User uploads test data and receives verification feedback.

Back-end

1. Provide upload form.
2. Verify uploaded data per criteria (format, content).
3. If verification fails, display errors and request re-upload.

8.3.4 Perform Accuracy Test

Front-end

User initiates accuracy test by pressing a button.

Back-end

1. Compare classification map with reference data to generate confusion matrix.
2. Compute accuracy metrics:
    - Overall Accuracy (OA)
    - Kappa Coefficient
    - Producer's Accuracy (PA)
    - User's Accuracy (UA)

8.3.5 Display and Review Accuracy Results

Front-end

User reviews test results. If satisfied, process completes; if not, prompted to revisit Modules 3 and/or 6.

Back-end

1. Display confusion matrix and accuracy metrics.
2. If unsatisfactory, notify user to check training data or classification parameters.

9 Module 8: Post Classification Analysis and Housekeeping

9.1 Module Overview

This module allows users to review all artifacts generated from previous modules, perform post-classification analysis of the LULC map results from Module 6 (limited to Phase 1 information from a single time-series land cover and land use map, such as bar charts or statistical summary tables), and export the LULC map results with metadata and accuracy assessment (if available). It activates only after completing the LULC map generation and accuracy test in Modules 6 and 7. On the back-end, the system generates an overview report rendered to HTML, displays summaries and visuals, and handles exports to Google Drive with progress monitoring and notifications.

Input

User's Input

1. Click to access Module 8.
2. Press download button to initiate export (yes/no).

Automatic System Input

1. None.

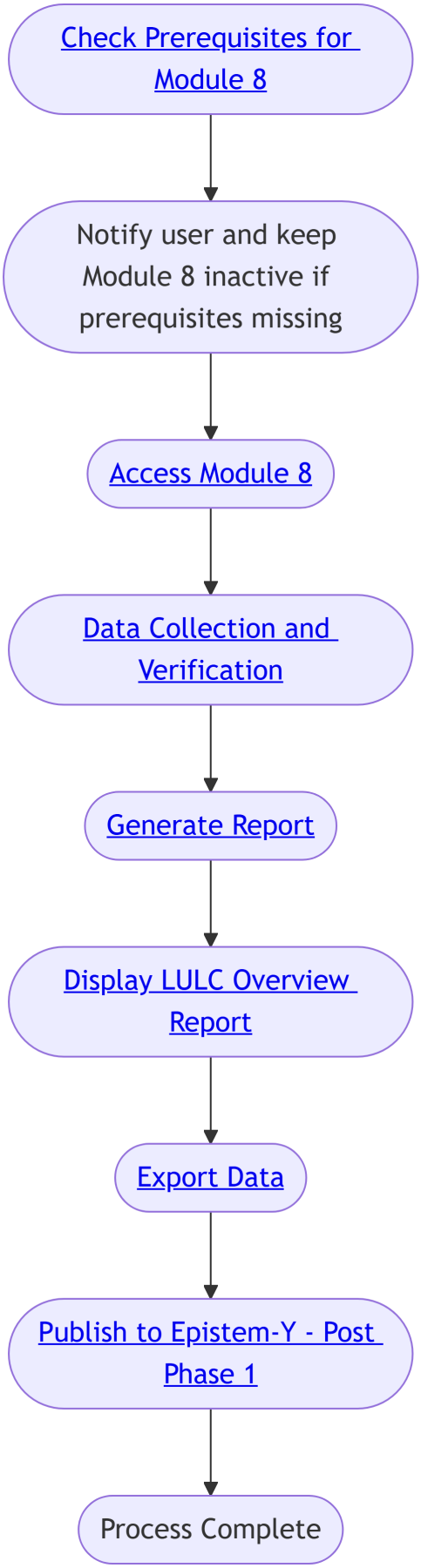
Input from Other Modules

1. Landsat image that has been cropped and has minimal cloud cover (Module 1).
2. Image search report (report_metadata Module 1).
3. LULC classification scheme table (Module 2).
4. Georeferenced training dataset (Module 3).
5. Model specification report (report_metadata Module 6).
6. Classified LULC map (Peta LULC Module 6).
7. Confusion matrix and accuracy test table (Module 7).

Output

1. LULC Classification Overview Report (.html via Quarto).
2. LULC Area Summary Table (digest table of area per land cover class).
3. Area Bar Chart.
4. Exported assets to Google Drive: LULC map + metadata/log-summary + accuracy test results (with warning if Module 7 not run).

9.2 Process


Check Prerequisites for Module 8
Notify user and keep Module 8 inactive if prerequisites missing
Access Module 8
Data Collection and Verification
Generate Report
Display LULC Overview Report
Export Data
Publish to Epistem-Y - Post Phase 1
Process Complete

9.3 Checking Prerequisites from Previous Modules

Front-end

1. The user clicks Module 8 to attempt access.
2. If prerequisites are missing, Module 8 remains inactive, and the user is notified to complete Modules 6 and 7.

Back-end

1. Activate Module 8 only if the user has finished generating the LULC map and accuracy test; check the existence of results from Modules 6 and 7, along with prerequisites from Module 1.
2. The system checks whether output from Module 7 is available.
3. If not, keep Module 8 inactive.

9.4 Post-Classification Analysis and Export

9.4.1 Access Module and Check Prerequisites

Front-end

1. User attempts to access Module 8.
2. If prerequisites missing, notify user and keep module inactive.

Back-end

Activate Module 8 only if outputs from Modules 6 and 7 exist, and Module 1 prerequisites met.

9.4.2 Data Collection and Verification

Front-end

Runs automatically upon access.

Back-end

1. Collect and verify data and variables: AOI, year, image type, cloud percentage, land cover classes, training data digest, model specifications from Module 6, minimum-cloud mosaic and metadata from Module 1, classified LULC map and metadata, accuracy test table from Module 7.

2. Calculate area summary table and generate bar graph visualization.

9.4.3 Generate Report

Front-end

Prepares report for display.

Back-end

Uses Module 8 Report and LULC Map Templates (.qmd) to render summary report to HTML with visuals and stats.

9.4.4 Display LULC Classification Overview Report

Front-end

User views summary including:

- AOI and metadata from Modules 1-6 and 7, such as thumbnails, bar charts, tables, accuracy results.

Back-end

Displays generated report with maps and charts.

9.4.5 Export Data

Front-end

1. Users can download exported assets to Google Drive if quota allows.
2. Download buttons show file size and estimated time.
3. Progress indicators and completion notifications appear.

Back-end

1. Identify classified map and metadata from Module 6.
2. Include accuracy results from Module 7 if available; log warning if not.
3. Prepare export filenames and size estimates.
4. Trigger Google Drive export action.
5. Monitor export job and notify user on completion.

9.4.6 Publish to Epistem-Y

Front-end

(Not in Phase 1) Menu option for publishing assets with terms acceptance and comment fields.

Back-end

(Post-Phase 1) Handle publishing LULC maps with user notes to platform.

10 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

References