Scalable Geospatial Machine Learning for Soil Health and Biomass Mapping

Thesis submitted by

Ratinder Pal Singh
2024MCS2464

under the guidance of

Prof. Aaditeshwar Seth

in partial fulfilment of the requirements

for the award of the degree of

Master of Technology in Computer Science and Engineering

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

July 2026

Certificate

This is to certify that the thesis titled **Scalable Geospatial Machine Learning for Soil Health and Biomass Mapping**, submitted by **Ratinder Pal Singh** (Entry No. **2024MCS2464**) to the Department of Computer Science and Engineering, Indian Institute of Technology Delhi, for the award of the degree of **Master of Technology in Computer Science and Engineering** in Computer Science and Engineering, is a record of the bonafide work carried out by him under my supervision and guidance.

The contents of this thesis have not been submitted, either in part or in full, to any other university or institute for the award of any degree or diploma.

Prof. Aaditeshwar Seth
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
July 2026

Acknowledgements

I would like to express my sincere gratitude to Prof. Aaditeshwar Seth for his guidance, feedback, and continuous support throughout this MTP work. His inputs shaped both the soil-health mapping pipeline and the subsequent biomass extension.

I also thank the CoRE Stack team for reviewing the production-readiness of the soil-health layers and for providing detailed feedback on CRS alignment, masking, missing pixels, and deployment requirements. Their comments helped convert the work from a research prototype into a more robust export and visualization pipeline.

I am grateful to the Department of Computer Science and Engineering, Indian Institute of Technology Delhi, for providing the academic environment and computational support needed for this work. I also thank everyone who helped with discussions, debugging, and project reviews during the weekly progress meetings.

Abstract

National soil health monitoring is critical for sustainable agriculture and food security. While the Government of India's Soil Health Card (SHC) scheme provides millions of localized ground-truth samples, translating these discrete points into continuous, high-resolution spatial maps remains a computational challenge. Furthermore, relying on existing global soil datasets (such as OpenLandMap) often yields inaccurate representations of localized tropical agricultural conditions.

Building upon foundational digital soil mapping techniques, this project presents a fully automated, end-to-end machine learning pipeline to predict Soil Organic Carbon (OC), Nitrogen (N), Phosphorus (P), and Potassium (K) at a 30-meter resolution across 18 Agro-Ecological Zones (AEZs) in India for the 2023-24 agricultural period. The methodology introduces strict Land Use/Land Cover (LULC) agricultural filtering, novel feature engineering (NIRv and pH), and a highly scalable, configuration-driven training architecture using Random Forest regressors.

A rigorous comparative analysis reveals that the custom AEZ-stratified Random Forest models significantly outperform the global OpenLandMap (OLM) baseline. The OLM dataset exhibits severe spatial and data-type quantization artifacts, alongside systematic overprediction caused by temperate-soil training bias. Finally, the generated models and assets are deployed into a dynamic, interactive Google Earth Engine (GEE) web application, providing an accessible interface for nationwide soil health auditing.

This thesis further extends the same geospatial machine learning workflow to above-ground biomass density (AGBD) estimation. GEDI L4A footprint observations were used as biomass ground observations, Google satellite embedding features were used as predictors, and CTrees global 100 m AGB products were used as an external baseline. The biomass extension includes a controlled AEZ 8 experiment, CTrees scaling and comparison, distribution-split analysis, canopy cover density and canopy height diagnostics, and baseline versus binned Random Forest experiments. Overall, the work demonstrates a reusable framework for converting sparse environmental observations into high-resolution spatial products, with a completed production-ready soil-health system and an exploratory biomass/carbon extension.

Contents

List of Tables

List of Figures

Abbreviations

AEZ	Agro-Ecological Zone
AGB	Above-Ground Biomass
AGBD	Above-Ground Biomass Density
AGBref	Above-Ground Biomass Reference Dataset
CCD	Canopy Cover Density
CH	Canopy Height
CHIRPS	Climate Hazards Group InfraRed Precipitation with Station data
CTrees	Carbon Trees global AGB product
GEDI	Global Ecosystem Dynamics Investigation
GEE	Google Earth Engine
LULC	Land Use/Land Cover
MAE	Mean Absolute Error
MODIS	Moderate Resolution Imaging Spectroradiometer
NIRv	Near-Infrared Reflectance of Vegetation
OC	Organic Carbon
OLM	OpenLandMap
RF	Random Forest
RMSE	Root Mean Squared Error
SHC	Soil Health Card
sMAPE	symmetric Mean Absolute Percentage Error
SRTM	Shuttle Radar Topography Mission

Chapter 1

Introduction

1.1 Project Overview

The assessment of soil nutrient profiles is vital for optimizing fertilizer application, minimizing ecological runoff, and maximizing crop yield. Traditional laboratory-based soil testing is resource-intensive and lacks the spatial continuity required for precision agriculture. The integration of satellite remote sensing, environmental covariates, and machine learning offers a pathway to generate spatially exhaustive soil health maps.

This project extends prior methodologies by introducing a robust, automated pipeline capable of processing data at a Pan-India scale. By stratifying the Indian subcontinent into distinct Agro-Ecological Zones (AEZs), the models inherently account for macro-climatic and pedological heterogeneity.

The second part of the thesis extends the same geospatial machine learning perspective to above-ground biomass density estimation. Biomass estimation is relevant to forest carbon accounting, ecological monitoring, and the evaluation of global carbon products. GEDI provides footprint-level above-ground biomass density observations, while global products such as CTrees provide spatially continuous biomass maps. This creates a natural validation and modeling problem: can a locally trained model using satellite embeddings better match GEDI observations than a global product at the same locations?

1.2 Motivation

Agricultural and ecological decisions require spatially continuous information. Soil Health Card samples and GEDI footprints are accurate and valuable, but they are point observations. Farmers, planners, researchers, and production systems usually require complete raster maps. Remote sensing provides wall-to-wall observations, but remote-sensing measurements are indirect proxies. A supervised machine learning pipeline is therefore needed to connect ground measurements to spatially complete covariates.

The first motivation of this work is practical: soil nutrient maps at high spatial resolution can support soil health auditing, fertilizer planning, and targeted interventions. The second motivation is methodological: the same pipeline architecture can be reused for other environmental variables that combine point measurements with satellite features. Biomass was therefore used as a natural extension after the soil-health pipeline was completed.

1.3 Key Contributions

The primary contributions of this work include:

- **End-to-End Automation:** Transitioning from manual script execution to a `config.yaml`-driven pipeline, enabling scalable hyperparameter tuning and model serialization.
- **Precision Feature Engineering:** Integration of Land Use/Land Cover (LULC) masking to strictly isolate agricultural croplands, alongside the addition of specific spectral indices like NIRv and topsoil pH.
- **Global Baseline Auditing:** A rigorous mathematical and spatial comparison between the localized Random Forest predictions and the global OpenLandMap (OLM) baseline [2], identifying critical failure modes in global pedotransfer functions when applied to Indian soils.
- **Interactive GEE Deployment:** Development of a public-facing Google Earth Engine application with dynamic mosaicking and real-time point inspection.
- **Production Handoff and Corrected Exports:** Regeneration of raw unmasked soil-health assets with explicit CRS and grid handling, followed by validation of all 72 RF assets.
- **Biomass Extension:** Construction of a GEDI-based biomass modeling workflow, CTrees comparison, canopy-density/height diagnostics, and baseline versus binned Random Forest experiments.

1.4 Data and Code Availability

To ensure full reproducibility and facilitate future research, the entire codebase, trained models, and dataset structures have been open-sourced and made publicly accessible.

- **GitHub Repository:** The complete end-to-end Python and Google Earth Engine (GEE) pipeline (scripts `1_data_download.py` through `16_csv_to_latex.ipynb`) is available at: `https://github.com/Singhratinder/Soil-Health-Mapping-PanIndia`.
- **Datasets:** Due to file size constraints, the raw and processed datasets are hosted via Google Drive (access links provided in the repository README). The project structure isolates the raw SHC data (`shc_data/YEAR_WISE_DATA`) from the fully normalized, LULC-filtered, and AEZ-stratified data used for training (`shc_data/AEZS/AGRI_2023-24/`).
- **Trained Models:** The 72 finalized AEZ-stratified Random Forest models are serialized in two formats within the repository: `rfr_joblib/` for localized offline Python inference and `rfr_csv/` for flattened tree structures optimized for GEE cloud inference.
- **Interactive Application:** The Pan-India mapped outputs and OLM baseline comparisons can be interacted with directly through the GEE Web Application: `https://ee-mtpictd-ratinder-mcs.projects.earthengine.app/view/pan-india-soil-health-maps`.

1.5 Thesis Organization

Chapter 2 reviews digital soil mapping, remote sensing, Random Forest regression, global soil baselines, GEDI, CTrees, and AGBref. Chapter 3 describes the datasets and feature sources. Chapter 4 presents the soil-health pipeline in detail, including the production fixes. Chapter 5 gives soil-health experiments and results. Chapter 6 presents the biomass extension. Chapter 7 discusses conclusions, limitations, and future work. Appendices provide additional code-level documentation and figures.

Chapter 2

Background and Related Work

2.1 Digital Soil Mapping

Digital soil mapping converts point-level soil observations into continuous spatial predictions by linking measured soil properties to environmental covariates. The covariates can include climate, terrain, land cover, vegetation indices, soil texture layers, and spectral measurements. In this work, the target soil properties are Nitrogen, Phosphorus, Potassium, and Organic Carbon. The modeling approach is supervised regression: each SHC sample provides a target value and the corresponding satellite/environmental features provide the predictor vector.

The key difficulty is that soil properties are not directly observed by satellites. Surface reflectance, vegetation behavior, rainfall, temperature, and terrain can only act as proxies. Therefore, high-resolution soil-health mapping requires careful feature engineering, filtering of irrelevant land-cover classes, and spatial stratification to reduce heterogeneity.

2.2 Soil Health Card Data and Indian Agricultural Context

The Soil Health Card (SHC) scheme provides large numbers of field and laboratory measurements across India. These measurements are localized and valuable, but they are not naturally organized as complete raster surfaces. The core computational challenge is therefore to convert millions of irregularly distributed point samples into continuous maps at a useful spatial resolution. The project focuses on the 2023-24 agricultural period and uses SHC data as ground truth for four nutrients.

2.3 Remote Sensing Features for Soil Nutrient Prediction

The final feature space combines climate, spectral, vegetation, and baseline soil variables. Climate variables include precipitation and land surface temperature. Spectral soil indices include saturation, redness, and greenness-related indices. Seasonal vegetation features include NDVI and NIRv for Kharif, Rabi, and Zaid seasons. Soil texture and pH features are taken from global soil layers.

The motivation is that different feature groups encode different mechanisms. Rainfall influences nutrient leaching and soil moisture regimes. Temperature influences microbial activity and mineralization. Soil texture influences nutrient retention capacity. Vegetation indices indirectly reflect crop vigor and soil fertility patterns.

2.4 Agro-Ecological Zone Stratification

India has strong climatic and pedological heterogeneity. Training one national model can force the model to learn incompatible relationships across very different agro-ecological regions. Therefore, the soil-health pipeline uses AEZ stratification: the country is divided into 18 Agro-Ecological Zones, and separate models are trained within each zone. This design produces 18 AEZs multiplied by 4 nutrients, i.e., 72 Random Forest models.

AEZ-wise training is not only a modeling choice; it also improves interpretability. Feature importance can be examined regionally, and failure modes can be diagnosed in terms of local data availability and regional environmental drivers.

2.5 Random Forest Regression

Random Forest regression is used because it can model nonlinear relationships, handle mixed feature types, and provide internal feature-importance estimates. It is also robust to moderate feature scaling differences and works well when many predictors have nonlinear interactions. In the soil-health pipeline, dynamic binning and inverse bin-frequency sample weights are used to handle skewed nutrient distributions. In the biomass extension, Random Forest is again used as a controlled baseline to learn AGBD from satellite embeddings.

2.6 OpenLandMap Baseline

OpenLandMap provides global soil property layers, including organic carbon. It is useful as a global baseline, but it may not represent localized Indian agricultural soils accurately. The soil-health report explicitly compares custom RF organic carbon predictions against OpenLandMap. The comparison reveals three main baseline failure modes: data-type quantization artifacts, spatial resolution mismatch, and systematic overprediction/depth bias.

2.7 Above-Ground Biomass and GEDI

Above-ground biomass density (AGBD) measures vegetation biomass per unit area, typically in Mg/ha. It is important for carbon accounting, forest monitoring, and ecological assessment. GEDI is a full-waveform lidar instrument installed on the International Space Station. NASA Earthdata describes GEDI as producing detailed observations of the 3D structure of the Earth's surface, measuring forest canopy height, canopy vertical structure, and surface elevation, with Level 4A providing footprint-level above-ground biomass density [4]. The biomass extension in this thesis uses GEDI L4A observations as point-level biomass ground observations.

2.8 CTrees Global AGB Product

The CTrees AGB Carbon data brief describes a global above-ground biomass product from 2000 to present, with a standard spatial resolution of 100 m, annual temporal resolution, and units in Mg/ha [5]. The same data brief specifies the storage format as Integer 16 with Scale: 0.1. This scale factor was important in the project because extracted CTrees values had to be multiplied by 0.1 before being compared with GEDI AGBD. Without this correction, CTrees values would be off by a factor of 10.

2.9 AGBref and Reference Data Challenges

The AGBref paper introduces a global AGB reference dataset derived from National Forest Inventories, permanent research plots, and local AGB maps from airborne LiDAR [6]. The paper emphasizes multiple epochs, multiple spatial supports, uncertainty estimates, and quality flags. This was relevant to the project because an AGBref-based validation route was investigated. However, the practical data-access and matching

constraints made CTrees comparison with GEDI points the more feasible route for this MTP timeline.

2.10 Canopy Cover Density and Canopy Height Layers

The biomass extension used canopy cover density and canopy height layers from the CoRE Stack Pan-India tree-characteristics catalog. These layers were not used as the main RF features in the final AEZ 8 RF comparison, but they were used diagnostically to interpret the AGBD distribution split. The central question was whether low-biomass and main-biomass groups correspond to different canopy regimes.

Chapter 3

Datasets and Feature Sources

3.1 Soil Health Card Ground Truth

Ground-truth soil nutrient data was programmatically extracted from the ICAR Soil Health Card (SHC) portal for the 2023-24 period. To ensure data integrity, district-wise Z-score filtering was applied to remove laboratory and digitization outliers, constraining values to physically realistic ranges.

The target variables in the soil-health part are Nitrogen, Phosphorus, Potassium, and Organic Carbon. The SHC samples are point observations. Each sample is associated with a geographic location and laboratory-measured nutrient values. The point measurements are then linked to satellite/environmental features extracted at the same coordinates.

3.2 Final Soil-Health Feature Space

To capture the complex environmental and pedological drivers of soil nutrient distribution, a finalized stack of 16 specific covariates was extracted for each ground-truth coordinate. The final selected features incorporated into the models are:

- **Climate Variables (2):** Temperature (`temp`) and Precipitation (`precipitation`).
- **Spectral Soil Indices (3):** Saturation Index (SI), Redness Index (RI), and Terrestrial Green Saturation Index (TGSI).
- **Seasonal Vegetation Indices (6):** Normalized Difference Vegetation Index (NDVI Kharif, NDVI Rabi, NDVI Zaid) and Near-Infrared Reflectance of Vegetation (NIRv Kharif, NIRv Rabi, NIRv Zaid).
- **Baseline Pedology and Texture (5):** Sand fraction at 0-5 cm (`sand05`) and 5-15 cm (`sand515`), Silt fraction at 0-5 cm (`silt05`), and Soil pH at 0-5 cm (`pH_0-5`) and 5-15 cm (`pH_5-15`).

Table 3.1: Soil-health feature groups used in the final model.

Feature group	Features	Purpose
Climate	Temperature, precipitation	Macro-climatic regime
Spectral soil indices	SI, RI, TGSI	Soil/surface reflectance patterns
Seasonal vegetation	NDVI/NIRv for Kharif, Rabi, Zaid	Seasonal crop response
Soil texture/pH	Sand, silt, pH	Baseline soil properties

3.3 Land Use/Land Cover Filtering

A critical enhancement in this pipeline is the strict application of an LULC crop mask. By filtering the feature space to only include pixels classified as active croplands, the models are shielded from confounding spectral signatures originating from urban environments, water bodies, or dense forests. The filtered dataset was subsequently spatially joined with the NBSS&LUP AEZ vector boundaries, resulting in 18 discrete, zone-specific datasets.

In the production version, the trained models were exported as raw, unmasked prediction layers. The earlier LULC masking step was retained for training-data construction and GEE display logic, but removed from the exported prediction assets so that production systems can apply any LULC version or modal-corrected mask downstream.

3.4 SoilGrids Features and Missing-Data Implications

SoilGrids texture and pH layers are important predictors. However, their NoData regions can propagate into the final RF prediction. Since the RF classifier requires all predictor bands to be valid, a missing value in any required SoilGrids band causes the final prediction to become masked at that pixel. This became important during production review, where missing output pixels were diagnosed as being inherited mainly from SoilGrids masked areas rather than from the RF model itself.

3.5 GEDI L4A Biomass Data

For biomass estimation, GEDI L4A footprint-level above-ground biomass density observations were used. GEDI footprints provide point-level AGBD estimates that can be spatially joined with satellite embedding features. Quality filtering is necessary because GEDI measurements may include invalid or low-confidence footprints.

3.6 CTrees AGB Product

The CTrees AGB product was used as an external global baseline. The product is global at a standard 100 m resolution, annual, and spans 2000-2025 in the data brief. Its unit is Mg/ha, but the stored data are integer encoded with scale factor 0.1. Therefore, CTrees extracted values were scaled before computing comparison metrics.

3.7 Canopy Cover Density and Canopy Height

Canopy cover density and canopy height layers were used to explain the observed AGBD distribution split. These were diagnostic layers, not final RF predictors in the main AEZ 8 CTrees comparison. The analysis grouped GEDI observations into low, transition, and main biomass groups and compared their CCD and CH class distributions.

3.8 Directory Structure

The final project organization contained separate output directories for soil-health statistics, plots, AEZ-merging analysis, missing-pixel diagnostics, biomass CTrees comparison, jump analysis, and automated biomass training. The soil-health pipeline also includes notebooks for each processing stage, while the biomass workflow includes scripts and notebooks for GEDI fetching, CTrees extraction, RF training, jump diagnostics, and model comparison.

Chapter 4

Soil Health Mapping Pipeline

4.1 Pipeline Overview

This chapter describes the completed soil-health pipeline. The pipeline converts SHC point measurements into 30 m Pan-India nutrient maps through preprocessing, feature extraction, AEZ stratification, Random Forest training, asset export, and GEE deployment.

SHC samples → Feature extraction → LULC filtering → AEZ split → RF training → GEE export → Interactive app

Figure 4.1: High-level soil-health mapping workflow.

4.2 Data Acquisition and Preprocessing

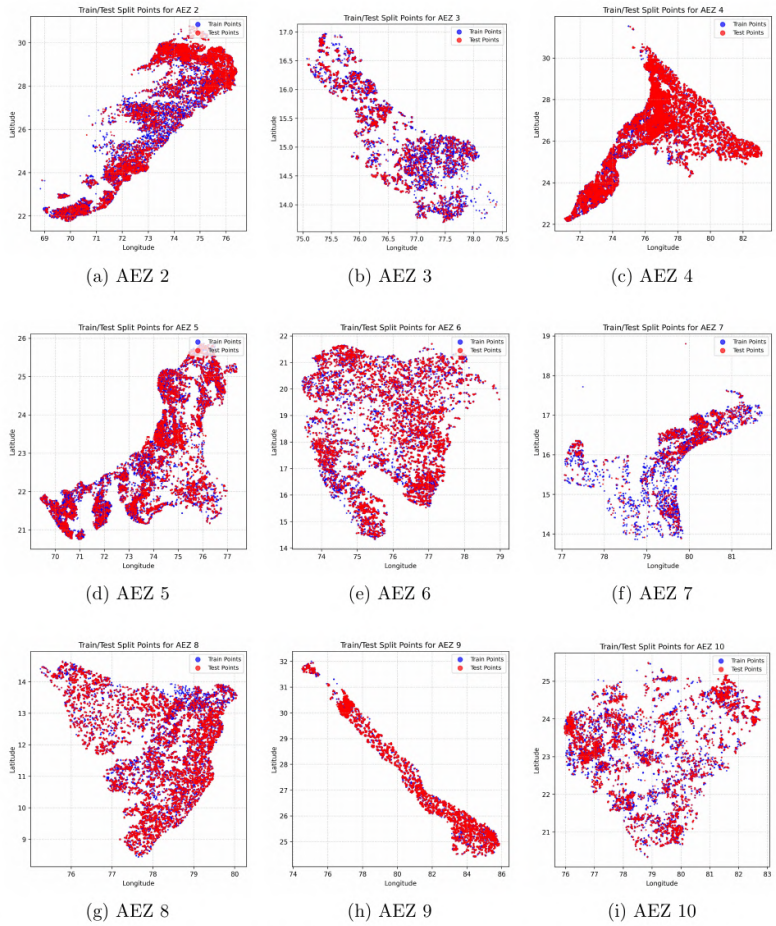
Ground-truth soil nutrient data was programmatically extracted from the ICAR Soil Health Card (SHC) portal for the 2023-24 period. To ensure data integrity, district-wise Z-score filtering was applied to remove laboratory and digitization outliers, constraining values to physically realistic ranges.

The spatial covariates were aggregated using the Google Earth Engine (GEE) Python API [1]. The feature stack includes harmonized Landsat 8 and 9 surface reflectance and derived indices such as NDVI and NIRv, MODIS Land Surface Temperature, CHIRPS precipitation, SRTM topography, and SoilGrids texture data (sand, silt, clay, pH).

4.3 LULC Filtering and AEZ Stratification

A critical enhancement in this pipeline is the strict application of an LULC crop mask. By filtering the feature space to only include pixels classified as active croplands, the models are shielded from confounding spectral signatures originating from urban environments, water bodies, or dense forests. The filtered dataset was subsequently

spatially joined with the NBSS&LUP AEZ vector boundaries, resulting in 18 discrete, zone-specific datasets.



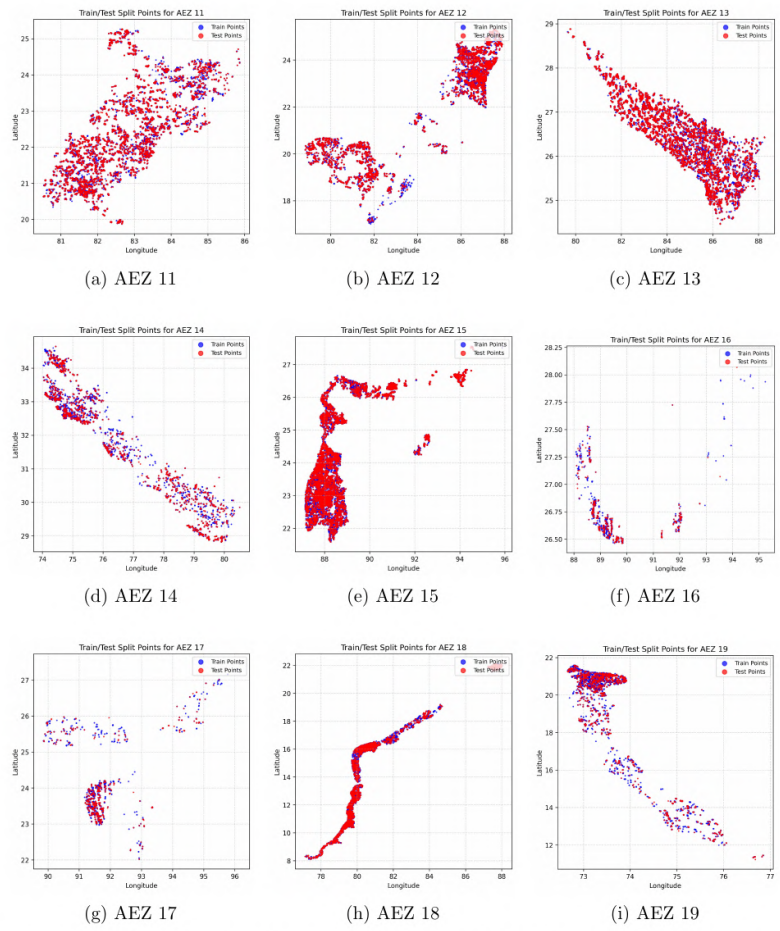
(a) AEZ 2 (b) AEZ 3 (c) AEZ 4

(d) AEZ 5 (e) AEZ 6 (f) AEZ 7

(g) AEZ 8 (h) AEZ 9 (i) AEZ 10

Figure A.1: Train-Test Split Plots for AEZs 2 to 10. Blue points represent the training set (80%), and red points represent the test set (20%).

Figure 4.2: Train-test split visualizations for AEZs 2 to 10. Blue points represent training points and red points represent test points.



(a) AEZ 11 (b) AEZ 12 (c) AEZ 13

(d) AEZ 14 (e) AEZ 15 (f) AEZ 16

(g) AEZ 17 (h) AEZ 18 (i) AEZ 19

Figure A.2: Train-Test Split Plots for AEZs 11 to 19. Blue points represent the training set (80%), and red points represent the test set (20%).

Figure 4.3: Train-test split visualizations for AEZs 11 to 19. These plots help verify that evaluation points are spatially distributed across each AEZ.

4.4 Model Training and Serialization

The training phase utilized Random Forest Regressors, chosen for their robustness to nonlinear environmental relationships and inherent feature importance metrics. To

handle the natural skewness of nutrient distributions, dynamic binning was applied, and inverse bin frequency sample weights were utilized during fitting. Optuna was employed for hyperparameter optimization, optimizing R2 across a cross-validation scheme. The best-performing models were serialized as both `.joblib` files for local inference and flat `.csv` tree structures for GEE cloud inference.

The training notebook was later automated so that all 72 models could be trained in a single run. This avoided the earlier manual process where the AEZ number and nutrient had to be changed repeatedly.

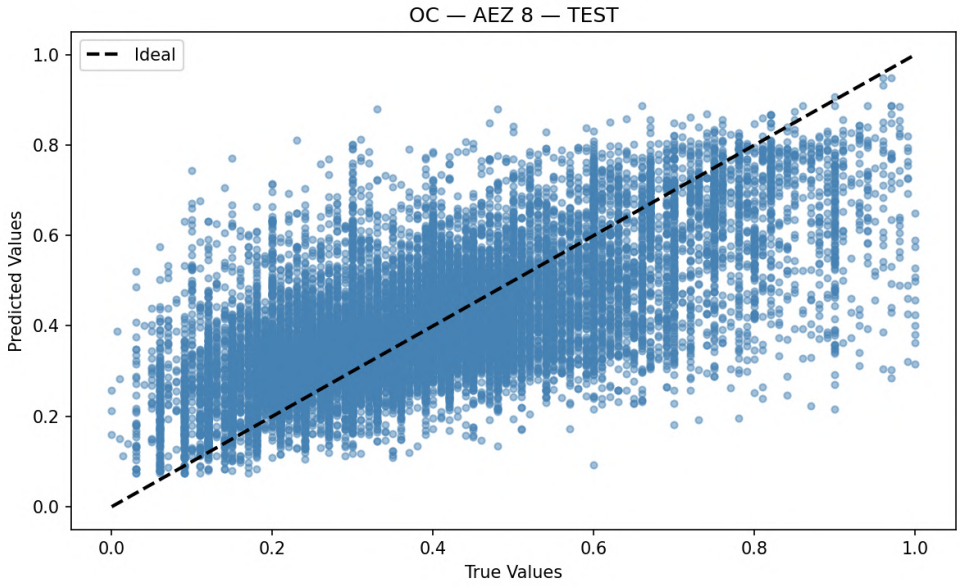


Figure 4.4: Example AEZ 8 OC test true-versus-predicted plot from the soil-health Random Forest training output.

4.5 Model Export and GEE Deployment

The trained RF models were converted into flattened CSV tree structures for Earth Engine inference. This was required because direct model payloads can exceed GEE limits. The inference/export notebook loads the model trees, reconstructs the predictor stack, applies the trained classifier, and exports one AEZ-wise raster asset per nutrient.

The GEE app dynamically mosaics the AEZ-wise assets into seamless Pan-India layers. The user can toggle between N, P, K, RF OC, and OpenLandMap OC layers and inspect values at clicked locations.

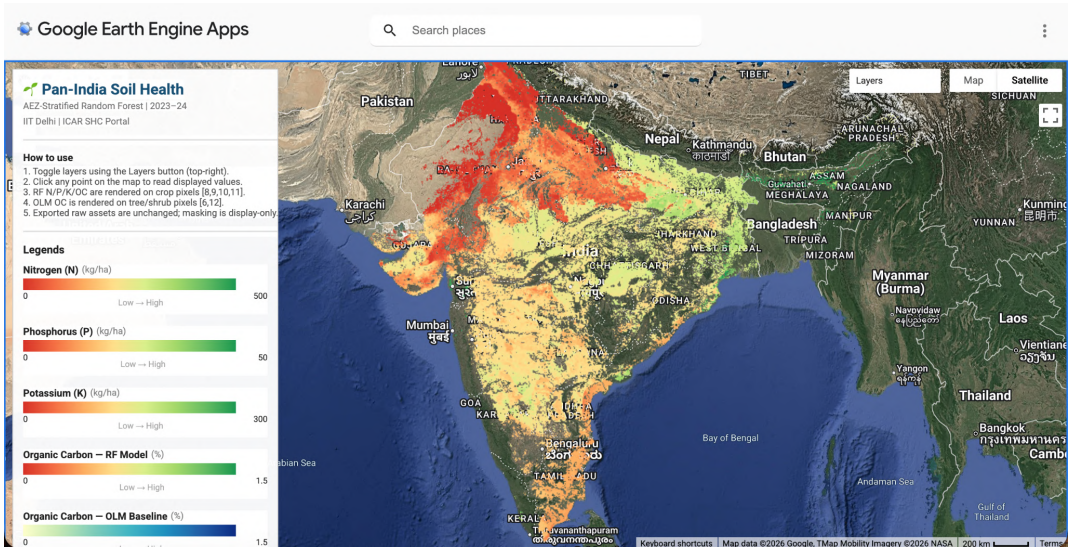


Figure 4.5: GEE app visualization of the Pan-India Nitrogen prediction map.

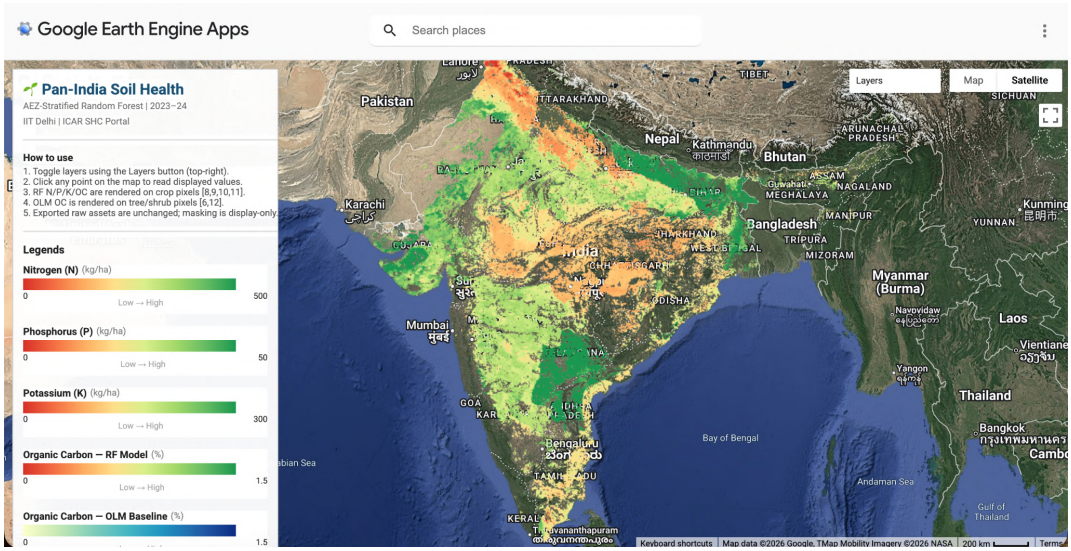


Figure 4.6: GEE app visualization of the Pan-India Phosphorus prediction map.

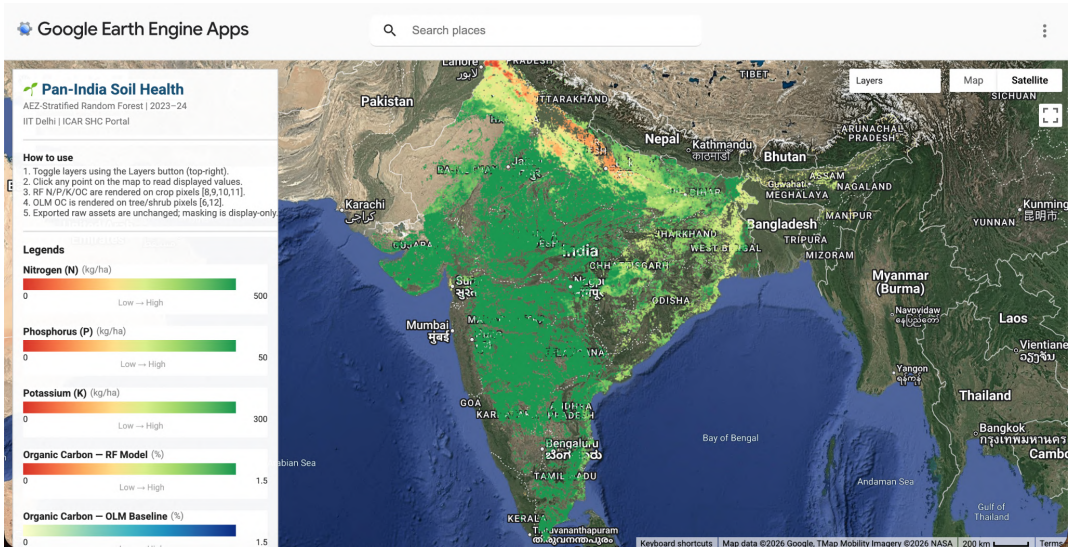


Figure 4.7: GEE app visualization of the Pan-India Potassium prediction map.

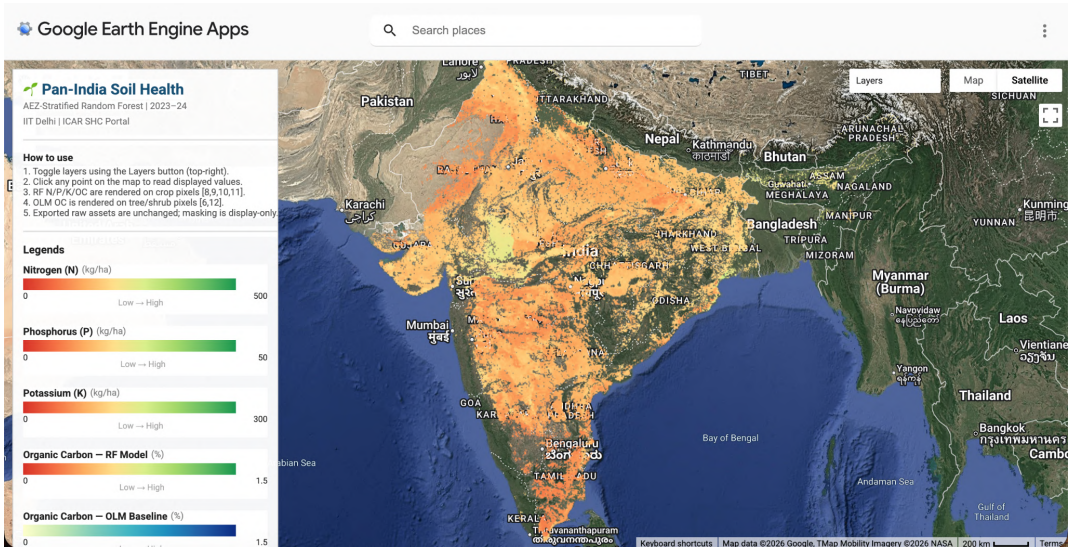


Figure 4.8: GEE app visualization of the Pan-India Organic Carbon RF prediction map.

4.6 OpenLandMap OC Export

The OpenLandMap OC baseline was exported as a raw unmasked layer for comparison with the RF organic carbon map. The final export handled the unit conversion during export itself, so the GEE app did not require the earlier extra correction factor. The OLM baseline was not a replacement for the RF model; it was used to quantify whether a global OC product could provide reliable local estimates over Indian agricultural land.

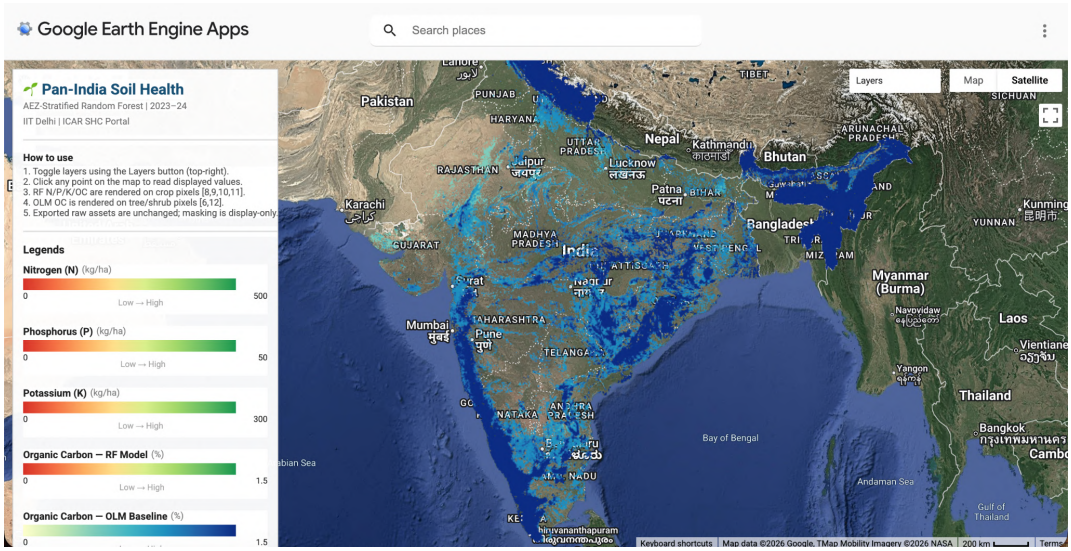


Figure 4.9: GEE app visualization of the OpenLandMap organic carbon baseline layer.

4.7 Production Fix: CRS and Grid Alignment

After the initial handoff, the production team observed a tilting issue in some exported assets. The corrected export pipeline addressed this by explicitly defining a common CRS and transform for every exported asset. All 72 RF assets were re-exported with explicit CRS/grid handling, and the OLM OC baseline was also regenerated as a raw unmasked layer.

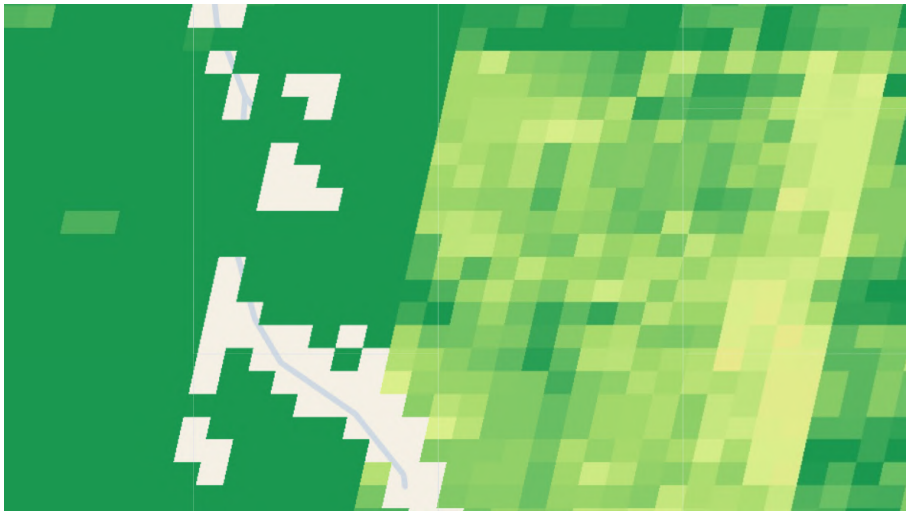


Figure 4.10: Production-team screenshot illustrating the original tilted-grid appearance.

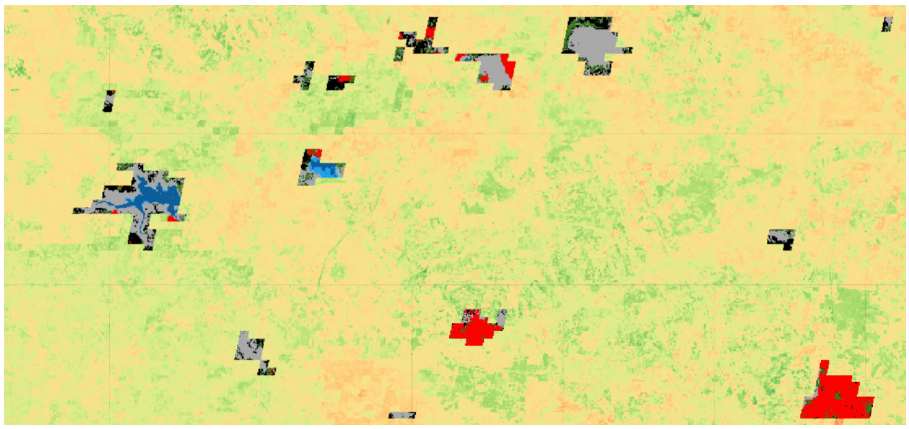


Figure 4.11: Detailed view of the tilted/misaligned appearance observed before the corrected export workflow.

Table 4.1: Validation summary for corrected raw RF assets.

Check	Count
Total assets checked	72
Existence OK	72
Band OK	72
CRS OK	72
Grid aligned OK	72
Sample OK	72

4.8 Missing-Pixel Diagnosis

The production team also asked why missing pixels existed inside some output regions, including near waterbodies and sometimes over crop, barren, shrub, or tree pixels. A diagnostic raster was generated for AEZ 8 / OC and missing prediction pixels were sampled. Each input feature band used by the RF model was checked for validity at those pixels.

The diagnosis showed that missing pixels were mainly caused by missing SoilGrids predictor bands, especially `sand05`, `silt05`, `sand515`, `pH_0-5`, and `pH_5-15`. Landsat annual indices, MODIS temperature, and CHIRPS precipitation were almost always valid in the sampled missing pixels. Therefore, the missing output pixels were primarily inherited from SoilGrids NoData/masked regions rather than from the RF prediction code.

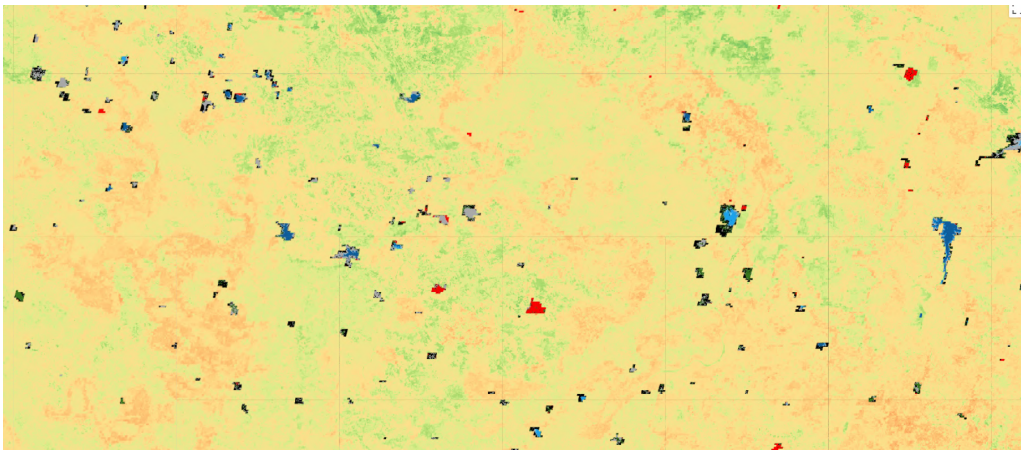
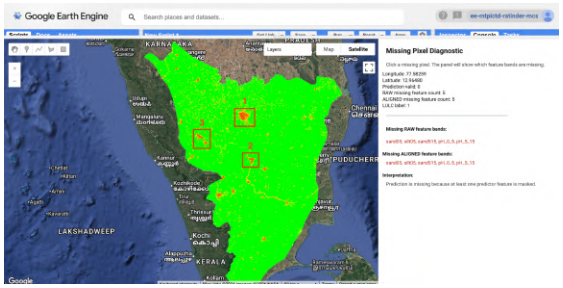
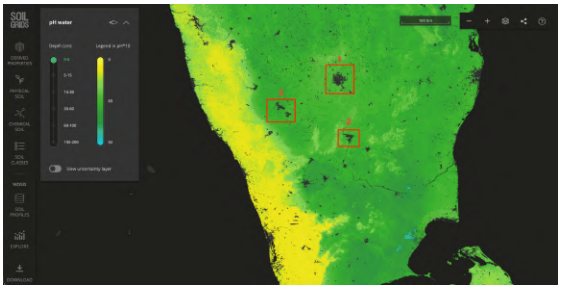


Figure 4.12: Missing-pixel diagnostic interface used to inspect invalid prediction pixels and identify missing input feature bands.



(a) AEZ 8 OC predictions with missing patches.



(b) SoilGrids pH 0-5 cm layer showing corresponding NoData regions.

Figure 4.13: Visual comparison of missing prediction patches with missing SoilGrids pH areas.

The safer scientific choice was to retain NoData where required predictor values are genuinely unavailable instead of filling them artificially. The export pipeline was made more robust by explicitly resampling/reprojecting source bands to the final grid, but source NoData areas were not hallucinated or filled.

Chapter 5

Soil Health Experiments and Results

5.1 Performance Across Agro-Ecological Zones

The AEZ-stratified Random Forest models demonstrated robust predictive capabilities, generalizing well across the diverse Indian landscape.

Table 5.1: Nutrient-wise summary of RF performance across AEZs.

Nutrient	No. of AEZs	Mean Test R2	Median Test R2
Nitrogen (N)	18	0.667	0.721
Phosphorus (P)	18	0.457	0.481
Potassium (K)	18	0.451	0.463
Organic Carbon (OC)	18	0.354	0.361

Table 5.2 gives the complete AEZ-by-nutrient model performance table for all 72 trained Random Forest models. The compact summary in Table 5.1 is retained only for quick interpretation, while the full table reports the exact test metrics used for evaluation.

Table 5.2: Random Forest model performance metrics across all AEZs and nutrients.

AEZ	Nutrient	Test R^2	Test RMSE	Test MAE	Test sMAPE (%)
AEZ 2	Nitrogen	0.8328	39.0316	17.5221	37.9742
AEZ 2	Phosphorus	0.4576	14.5579	9.1861	34.3565
AEZ 2	Potassium	0.5028	103.0515	67.1897	28.0655
AEZ 2	Organic Carbon	0.5203	0.1312	0.0943	25.3516
AEZ 3	Nitrogen	0.5168	72.2250	54.5391	28.3972
AEZ 3	Phosphorus	0.2481	16.1636	12.1265	35.3765
AEZ 3	Potassium	0.3761	115.8662	88.0344	29.6393
AEZ 3	Organic Carbon	0.3523	0.1638	0.1227	33.4826
AEZ 4	Nitrogen	0.7104	46.2545	26.5093	61.9061
AEZ 4	Phosphorus	0.5009	12.3096	6.9354	30.5271
AEZ 4	Potassium	0.4550	73.6872	51.2106	22.6509
AEZ 4	Organic Carbon	0.3468	0.1193	0.0878	24.2518
AEZ 5	Nitrogen	0.7853	53.4730	35.2709	44.6184
AEZ 5	Phosphorus	0.5004	13.7799	9.0111	30.1148
AEZ 5	Potassium	0.1911	114.8190	80.0023	25.8201
AEZ 5	Organic Carbon	0.2391	0.1764	0.1390	27.1023
AEZ 6	Nitrogen	0.6272	52.1266	38.3332	19.6935
AEZ 6	Phosphorus	0.5902	10.7180	6.0232	28.9597
AEZ 6	Potassium	0.3728	174.9519	112.8698	25.9565
AEZ 6	Organic Carbon	0.4442	0.1365	0.1032	23.6930

AEZ 7	Nitrogen	0.2775	65.1005	46.7340	24.1276
AEZ 7	Phosphorus	0.2434	21.5530	16.2973	53.8011
AEZ 7	Potassium	0.3925	241.9329	176.9436	44.8527
AEZ 7	Organic Carbon	0.0692	0.1911	0.1420	33.7129
AEZ 8	Nitrogen	0.5783	61.9592	40.2080	20.5919
AEZ 8	Phosphorus	0.4248	18.0792	10.8593	40.2754
AEZ 8	Potassium	0.5899	120.1281	83.5548	30.9687
AEZ 8	Organic Carbon	0.4617	0.1349	0.1007	28.3388
AEZ 9	Nitrogen	0.8498	51.8407	30.3530	38.7632
AEZ 9	Phosphorus	0.6103	12.3732	7.8381	41.3682
AEZ 9	Potassium	0.5855	57.6576	40.0607	24.1371
AEZ 9	Organic Carbon	0.4152	0.1224	0.0914	24.4510
AEZ 10	Nitrogen	0.3583	60.1226	44.8726	21.1573
AEZ 10	Phosphorus	0.4267	7.5233	5.2035	28.5339
AEZ 10	Potassium	0.4912	77.6777	53.2796	20.0830
AEZ 10	Organic Carbon	0.2005	0.1687	0.1328	24.3232
AEZ 11	Nitrogen	0.5312	56.6854	40.3504	18.4000
AEZ 11	Phosphorus	0.4654	6.1443	4.2174	31.1206
AEZ 11	Potassium	0.3780	84.2013	56.2350	21.0713
AEZ 11	Organic Carbon	0.2536	0.1612	0.1217	26.7035
AEZ 12	Nitrogen	0.4072	72.2133	53.9184	21.9678
AEZ 12	Phosphorus	0.5226	11.1172	6.0918	28.7984
AEZ 12	Potassium	0.4507	62.8393	45.0202	23.3843
AEZ 12	Organic Carbon	0.1965	0.1587	0.1209	24.7560
AEZ 13	Nitrogen	0.7882	39.8338	28.8768	19.7102
AEZ 13	Phosphorus	0.5950	9.8171	7.0258	30.6540
AEZ 13	Potassium	0.4150	50.6192	37.1098	19.1408
AEZ 13	Organic Carbon	0.3668	0.1418	0.1079	23.8154
AEZ 14	Nitrogen	0.8730	71.3039	40.1491	41.6315
AEZ 14	Phosphorus	0.4477	11.7008	7.8058	32.5498
AEZ 14	Potassium	0.5218	70.1822	47.6594	25.6919
AEZ 14	Organic Carbon	0.4997	0.1544	0.1194	22.2222
AEZ 15	Nitrogen	0.7265	68.3942	47.4703	13.1881
AEZ 15	Phosphorus	0.4941	16.3378	10.4949	27.6964
AEZ 15	Potassium	0.4711	70.4292	53.8293	29.1740
AEZ 15	Organic Carbon	0.2906	0.1625	0.1328	20.9695
AEZ 16	Nitrogen	0.7394	68.6052	49.6370	18.8964
AEZ 16	Phosphorus	0.4118	21.0087	13.1755	41.8195
AEZ 16	Potassium	0.2516	73.6526	55.2387	36.7090
AEZ 16	Organic Carbon	0.2370	0.1701	0.1368	21.1509
AEZ 17	Nitrogen	0.5641	79.8158	58.8569	25.1399
AEZ 17	Phosphorus	0.5682	8.6469	5.1923	23.7147
AEZ 17	Potassium	0.3132	40.3668	28.3516	18.4279
AEZ 17	Organic Carbon	0.5615	0.1581	0.1158	24.0037
AEZ 18	Nitrogen	0.7420	56.4275	35.8445	18.7481
AEZ 18	Phosphorus	0.4114	13.2394	8.2128	40.5331
AEZ 18	Potassium	0.6736	128.1927	81.6033	29.5952
AEZ 18	Organic Carbon	0.5827	0.1495	0.1083	31.4263
AEZ 19	Nitrogen	0.6754	103.4729	58.3651	22.4036
AEZ 19	Phosphorus	0.0650	26.4005	16.5530	43.9792
AEZ 19	Potassium	0.4399	120.4237	85.4205	30.5613
AEZ 19	Organic Carbon	0.1887	0.1967	0.1555	29.4108

Nitrogen (N) consistently achieved the highest accuracy, with Test R2 values frequently exceeding 0.70 and occasionally reaching 0.90 in specific homogeneous zones. Potassium (K) and Phosphorus (P) showed moderate to strong predictive skill. Organic Carbon

(OC), inherently the most complex target to model via surface reflectance due to its subsurface distribution, achieved functional positive R2 values, validating the methodology's capacity to capture spatial carbon gradients.

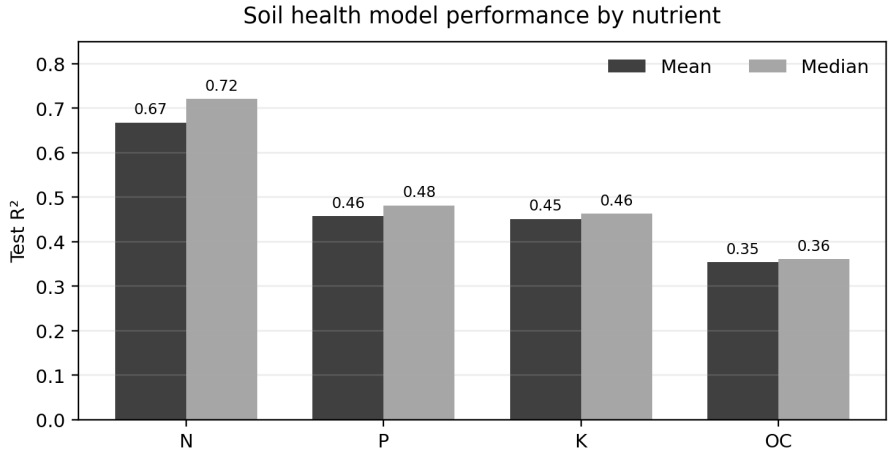


Figure 5.1: Nutrient-wise summary of soil-health model performance across AEZs.

5.2 Feature Importance and Environmental Drivers

To ensure the transparency of the AEZ-stratified approach, the internal Gini impurity-based feature importances of the Random Forest regressors were extracted and analyzed. Because absolute feature importance scores vary in magnitude between different zones, the 16 input covariates were converted into ordinal ranks, where 1 represents the most important feature and 16 represents the least important, for every AEZ.

The Pan-India average ranks for Nitrogen, Phosphorus, Potassium, and Organic Carbon were subsequently calculated to identify the universal macro-drivers of soil chemistry across the subcontinent.

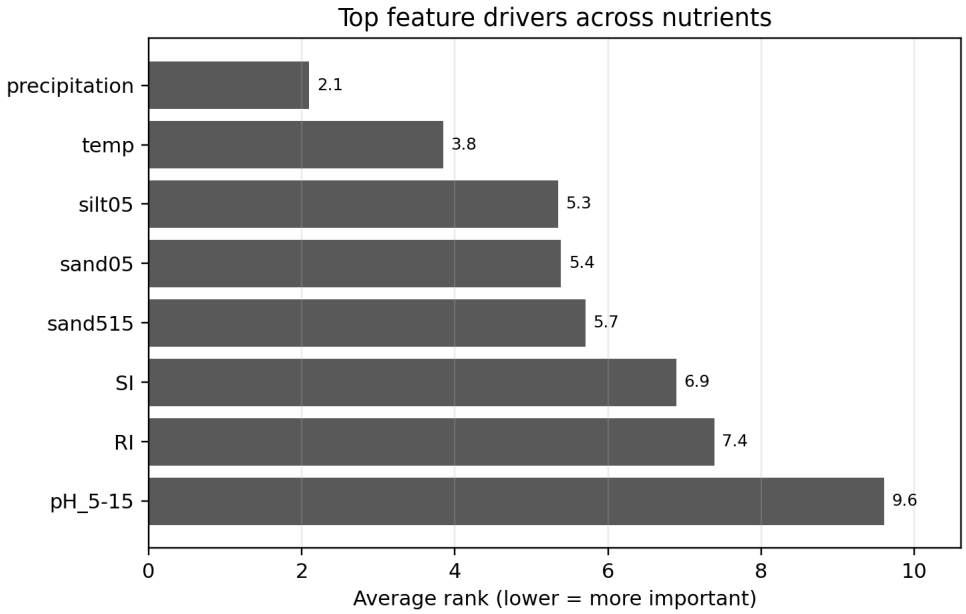


Figure 5.2: Top-ranked feature drivers across the soil-health models. Lower rank means higher importance.

The ranking analysis reveals distinct pedological patterns. Macro-climatic variables, specifically Precipitation and Land Surface Temperature (temp), consistently rank highly across all four nutrients. This aligns with fundamental agronomic principles, as precipitation governs the leaching of water-soluble nutrients, while temperature regulates microbial mineralization rates.

For baseline pedology, surface sand and silt fractions demonstrated strong predictive power, particularly for Phosphorus and Potassium. This confirms the models successfully learned that soil texture mechanically dictates nutrient retention capacity.

5.3 Spatial Baseline Comparison: RF vs. OpenLandMap

To validate the utility of the custom AEZ-stratified models, predictions for Organic Carbon (OC) were strictly compared against the global OpenLandMap (OLM) baseline dataset [2].

To ensure an exact, apples-to-apples spatial comparison, the 250 m native OLM raster was exported at a 30 m scale using nearest-neighbor resampling, preserving the raw global predictions without introducing synthetic interpolation artifacts. The identical LULC agricultural mask was applied. The metrics were computed locally using the exact identical 20% test set split used during the initial model training.

Table 5.3: Mean OC comparison between custom RF and OpenLandMap across AEZs.

Metric	Custom RF OC	OpenLandMap OC
R2	0.347	-15.389
RMSE	0.155	0.764
MAE	0.118	0.633
sMAPE (%)	26.038	76.754

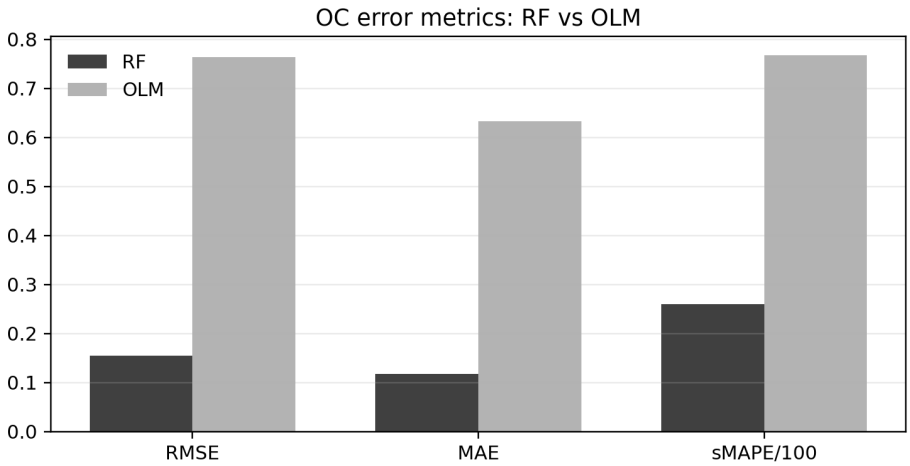


Figure 5.5: Organic carbon error comparison between the custom RF model and OpenLandMap baseline.

The comparative metrics reveal a stark contrast. While the custom RF model achieved positive R2 scores across all zones, the OLM baseline exhibited massively negative R2 values. A negative R2 indicates that the global model's predictions are mathematically worse than simply predicting the mean of the true data.

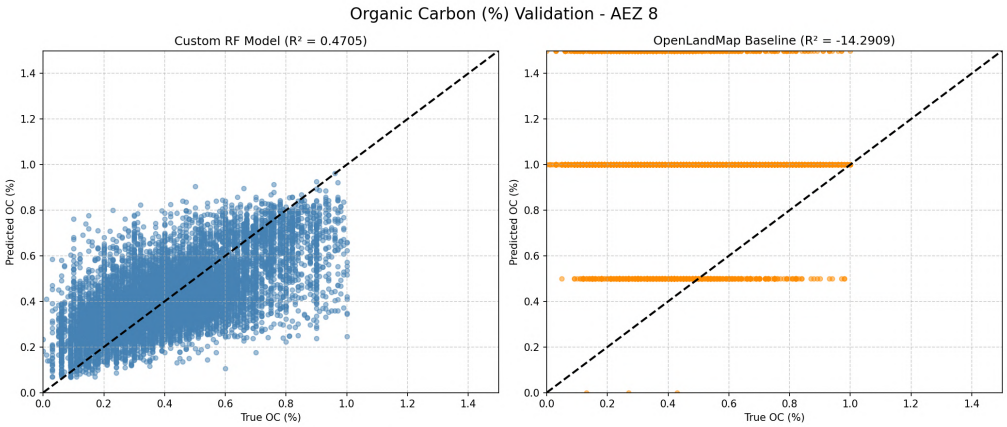


Figure 5.6: Example AEZ 8 organic carbon validation comparing RF predictions with OLM baseline predictions.

5.3.1 Data-Type Quantization Artifacts

To optimize cloud storage, the global OLM dataset stores OC values as 8-bit integers, utilizing a scaling factor of 5 g/kg. When mathematically converted to the standard percentage format used in Indian agronomy, the model is physically constrained to output predictions in rigid 0.5% steps. This extreme data-type quantization destroys the ability to model continuous soil gradients, resulting in distinct horizontal bands visible in the comparison plots. The custom RF model, operating on continuous 32-bit floats, captures local micro-variations.

5.3.2 Spatial Resolution Mismatch

The native 250 m resolution of OLM means that a single pixel covers roughly 6.25 hectares. Within the highly fragmented, small-holder farming landscape of India, dozens of highly variable 30 m ground-truth test points fall within a single OLM pixel and are assigned the exact same predicted value, reducing predictive accuracy.

5.3.3 Systematic Overprediction and Depth Bias

The OLM scatter plots reveal severe clustering at the 0.5% and 1.0% thresholds, despite the true ground-truth OC of Indian croplands heavily concentrating between 0.1% and 0.4%. This systematic overprediction stems from OLM's global training bias toward carbon-rich temperate soils and its reliance on absolute surface modeling. In contrast, the custom RF models were trained directly on the 0-15 cm plow layer from the SHC portal, capturing the carbon depletion characteristic of high-temperature, intensively farmed tropical soils.

5.4 Sample Count versus Accuracy Analysis

Professor Aaditeshwar suggested checking whether AEZs with fewer samples had lower accuracy. The analysis created one master table for AEZ-nutrient pairs, computed correlations between sample count and accuracy metrics, grouped AEZs into low/medium/high sample buckets, and generated scatter plots.

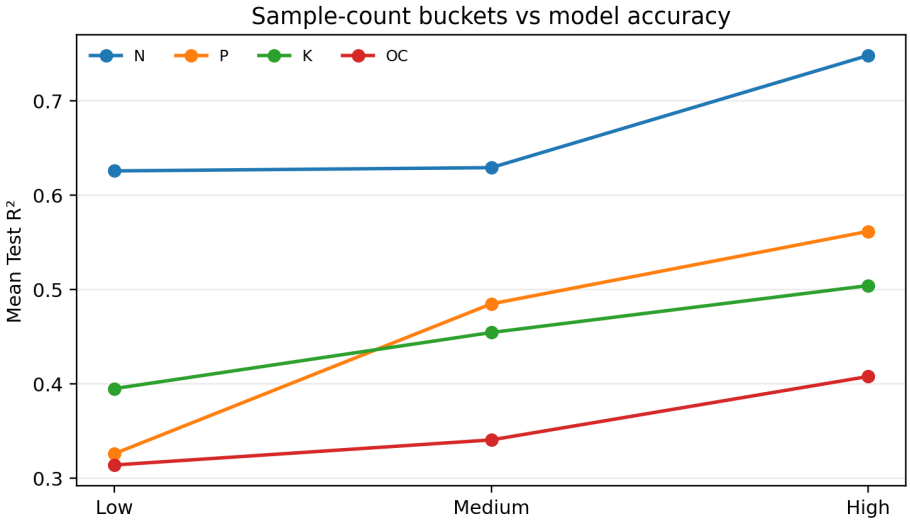


Figure 5.7: Mean R2 by sample-count bucket for each nutrient.

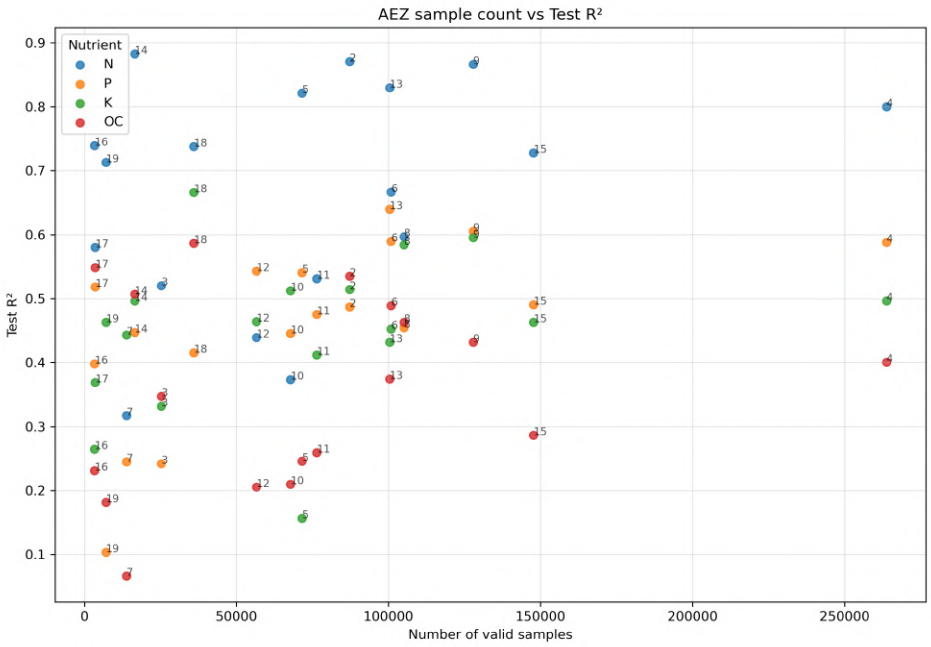


Figure 5.8: Sample count versus R2 by nutrient.

The analysis showed that sample count was related to performance for some nutrients, but the relationship was not sufficient to justify automatic merging of AEZs. Sample count was only one factor; environmental heterogeneity, nutrient distribution, and feature-target relationships also affected accuracy.

5.5 AEZ Merging Pilot

A pilot AEZ-merging experiment was conducted to test whether adjacent low-sample/low-accuracy AEZs could be combined into larger training regions. Candidate adjacent pairs were selected using sample count, accuracy, and environmental similarity. The pilot trained merged models for selected groups and compared them against original AEZ-wise models on same-held-out test sets.

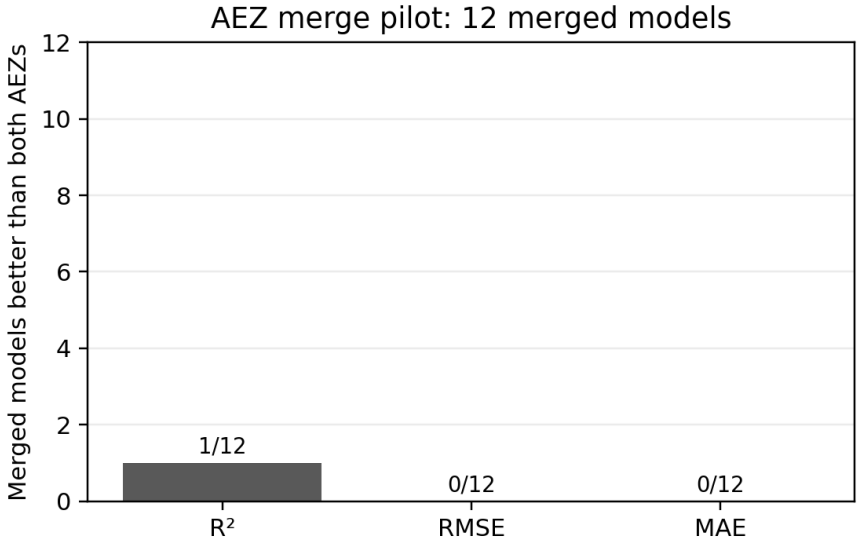


Figure 5.9: Summary of merged-model pilot outcomes. Most merged models did not outperform both original AEZ-wise models.

The main conclusion was that merging adjacent AEZs is not consistently beneficial. In some cases, it increases the sample count but mixes different soil-environment relationships. The recommendation is therefore to retain AEZ-wise models as the default and consider merging only selectively after environmental-similarity analysis and validation.

5.6 Production Validation

The corrected RF assets were validated for existence, band count, CRS, grid alignment, and sample extraction. All 72 RF assets passed the validation checks. This confirmed that the soil-health pipeline had moved from a research model to production-ready raw layers that can be used by downstream systems with their own masking logic.

Chapter 6

Biomass Mapping Extension

6.1 Motivation

After the soil-health pipeline was completed, the project was extended to above-ground biomass density (AGBD) estimation. The motivation was to test whether the same geospatial ML pattern - point observations, remote-sensing features, regional stratification, model training, and global-product comparison - could be applied to biomass/carbon monitoring.

6.2 AGBD Target and GEDI

The target variable is above-ground biomass density in Mg/ha. GEDI L4A provides footprint-level AGBD estimates, which can be interpreted as point-level ground observations for supervised modeling. The first controlled experiment focused on AEZ 8 and year 2022. Later scripts automated GEDI fetching and training across multiple years.

6.3 Attempted AGBref-Based Validation Route

The AGBref paper introduced a global reference dataset of above-ground forest biomass derived from National Forest Inventories, permanent research plots, and local AGB maps from airborne LiDAR [6]. It provides biomass estimates for multiple epochs and spatial supports, along with uncertainty estimates and quality flags. This made AGBref attractive as a possible validation dataset.

However, during the project timeline, directly acquiring and integrating the required AGBref ground-truth data for the AEZ experiments was not feasible. The practical route therefore shifted to using GEDI L4A observations as the reference points and CTrees as the global-product baseline.

6.4 CTrees Comparison Setup

The CTrees AGB product was accessed for the same region and year as the GEDI AEZ 8 experiment. The CTrees data brief specifies a scale factor of 0.1, so extracted CTrees values were multiplied by 0.1 before comparison [5]. The comparison was performed directly at the GEDI point locations, without reprojection of the GEDI points to the 100 m CTrees grid. This was aligned with the project guidance that the comparison could be made by sampling CTrees at the GEDI coordinates.

Table 6.1: RF and CTrees comparison on the same AEZ 8 GEDI held-out points for 2022.

Model	N	R2	RMSE	MAE	sMAPE_percent
RF Model	3878	0.480	32.826	24.900	33.800
CTrees	3878	-0.267	51.235	38.769	57.349

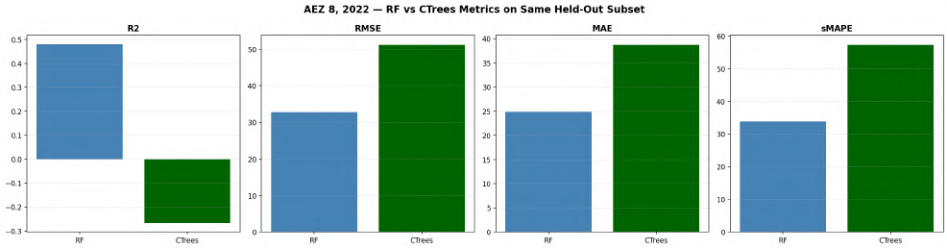


Figure 6.1: RF versus CTrees metric comparison on the same AEZ 8 GEDI held-out subset.

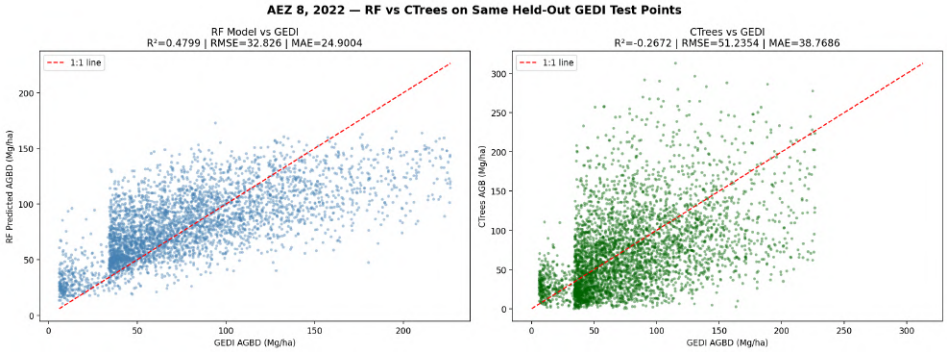


Figure 6.2: RF and CTrees predictions compared against GEDI AGBD on the same test points.

The local RF model substantially outperformed CTrees on the controlled AEZ 8, 2022 test subset. The RF model achieved positive R2, lower RMSE, and lower MAE, while CTrees produced negative R2 on the same points. RF also had lower absolute error on a majority of comparable points.

6.5 CTrees Diagnostics

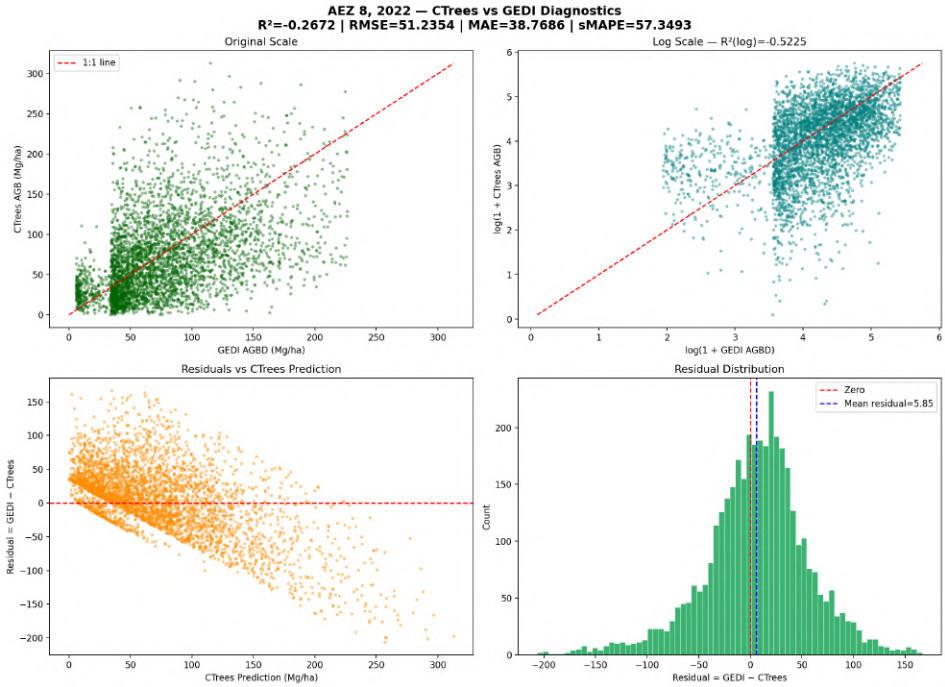


Figure 6.3: CTrees diagnostic plots for the same AEZ 8, 2022 GEDI comparison subset.

The CTrees result remains useful as a global benchmark. However, the AEZ-local RF model is tuned directly to GEDI observations and to the local embedding feature space, whereas CTrees is a global product trained for broad consistency across regions.

6.6 AGBD Distribution Split

During GEDI preprocessing and outlier removal, the AGBD distribution showed a low-value group and a main group. Professor Aaditeshwar suggested investigating whether the split corresponded to different land regimes, such as sparse vegetation, plantations, or forested areas. The points were therefore mapped and compared with canopy cover density and canopy height layers.

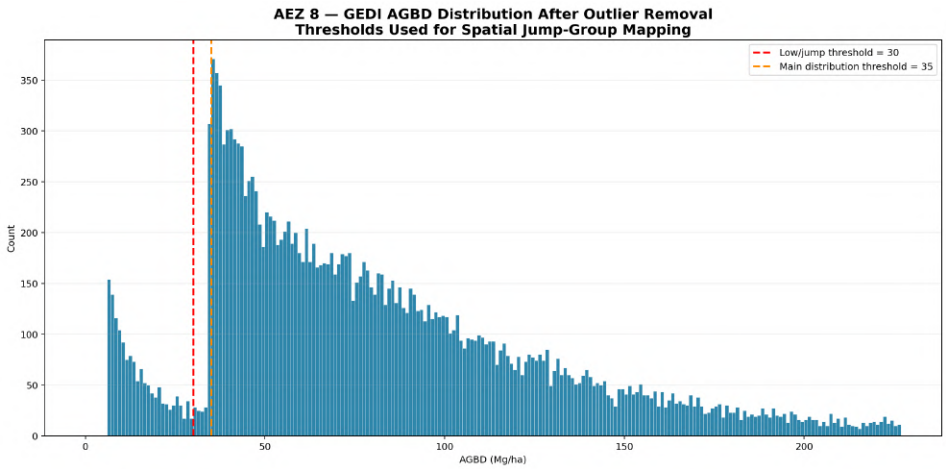


Figure 6.4: AGBD histogram with an observed distribution split in AEZ 8, 2022.

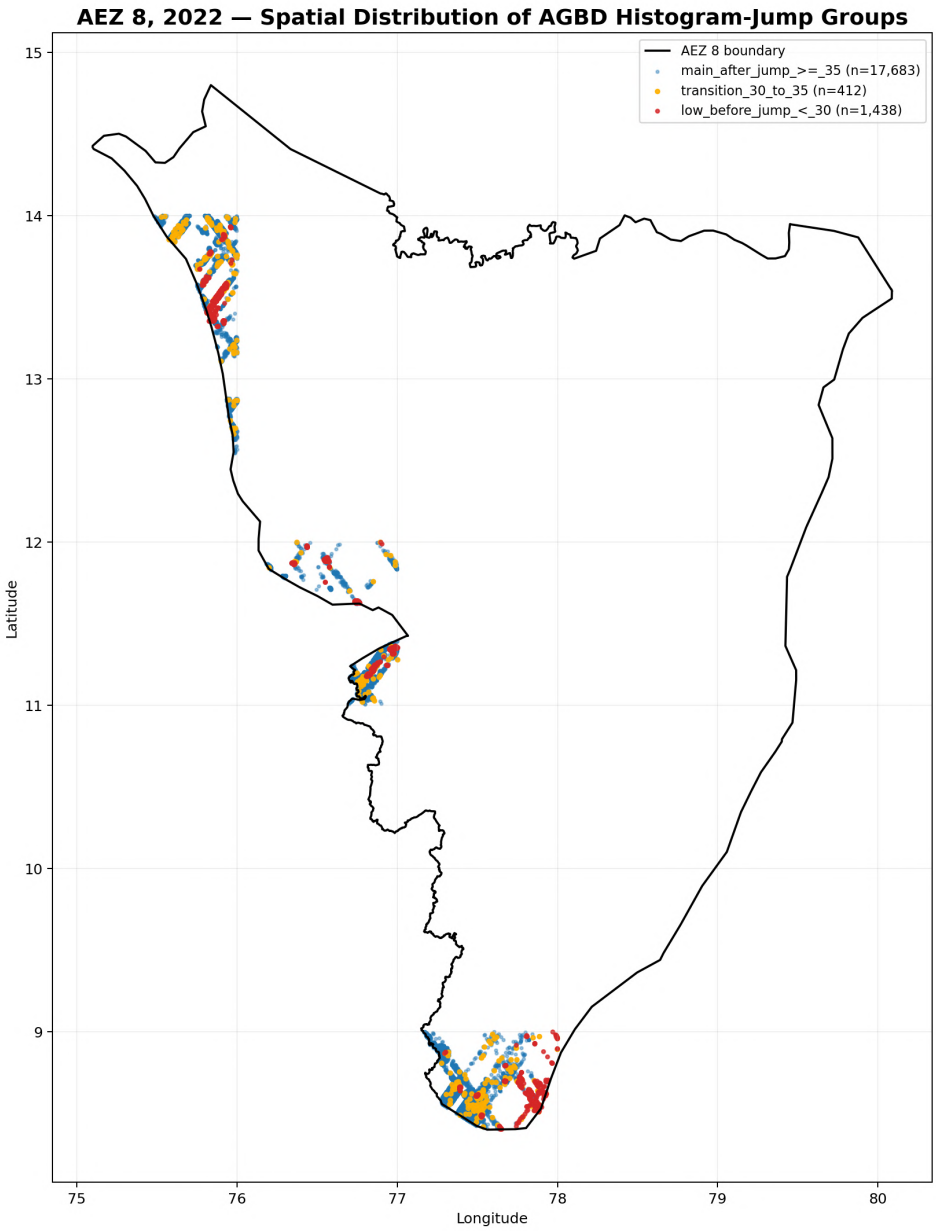


Figure 6.5: Spatial mapping of low and main AGBD groups for AEZ 8, 2022.

The spatial plot did not show a clean large-scale geographical separation. The groups overlap in the sampled region, and the points cover only a limited portion of AEZ 8. Therefore, the split could not be interpreted purely as spatial separation.

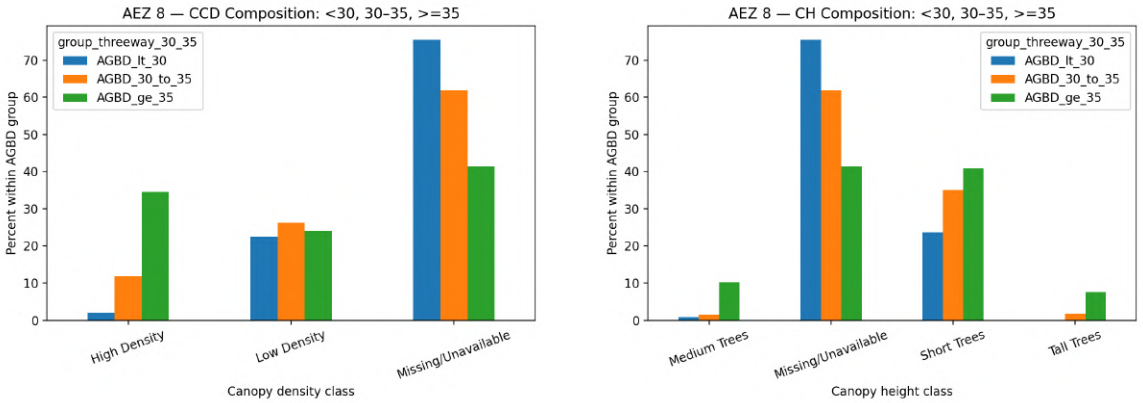
6.7 Canopy Cover Density and Canopy Height Diagnostics

To further inspect the split, fixed AGBD thresholds were used: less than 30 Mg/ha, 30-35 Mg/ha, and greater than or equal to 35 Mg/ha. These groups were compared

with canopy cover density and canopy height classes.

Table 6.2: Three-way AGBD group summary with canopy density and canopy height diagnostics.

AGBD group	N	Mean AGBD	CCD valid N	Mean CCD class	CH valid N	Mean CH class
AGBD_lt_30	1438	14.097	353	0.085	353	0.255
AGBD_30_to_35	412	33.961	157	0.312	157	0.529
AGBD_ge_35	17683	86.103	10352	0.589	10361	1.570



(a) CCD composition (b) CH composition

Figure 6.6: Canopy cover density and canopy height composition for AGBD groups.

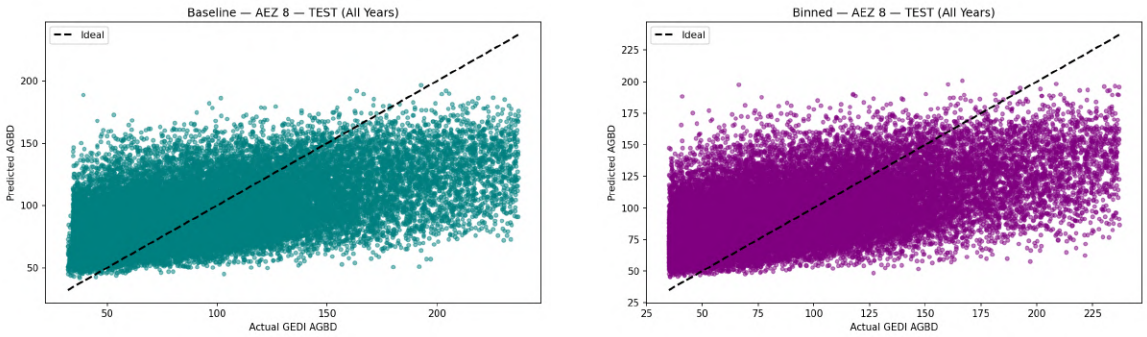
The low-AGBD group had much lower canopy density and height among pixels where CCD/CH values were valid. The main group had higher mean CCD and CH classes. However, CCD/CH coverage was incomplete, so the result was treated as diagnostic rather than as a basis for automatically dropping training samples.

6.8 Baseline RF versus Binned RF

A second modeling experiment tested whether binning/weighting the AGBD distribution could improve biomass prediction. Fixed-bin RF did not consistently improve the baseline. The baseline RF remained the simpler and more reliable model for the current AEZ 8 setup.

Table 6.3: AEZ 8 all-years biomass baseline RF versus fixed-bin RF comparison.

Model	Test R2	Test RMSE	Test MAE
Baseline RF	0.338	37.540	29.180
Fixed-bin RF	0.326	37.360	29.410



(a) Baseline RF (b) Fixed-bin RF

Figure 6.8: All-years AEZ 8 test true-versus-predicted plots for baseline and binned RF models.

6.9 Multi-Year Automation

The biomass workflow was later automated for multiple years, especially AEZ 8 for 2019-2024. The automation included fetching GEDI exports, merging yearly data, training baseline and binned RF models, and comparing the results. This made the biomass workflow structurally closer to the soil-health pipeline, although the biomass part remained exploratory.

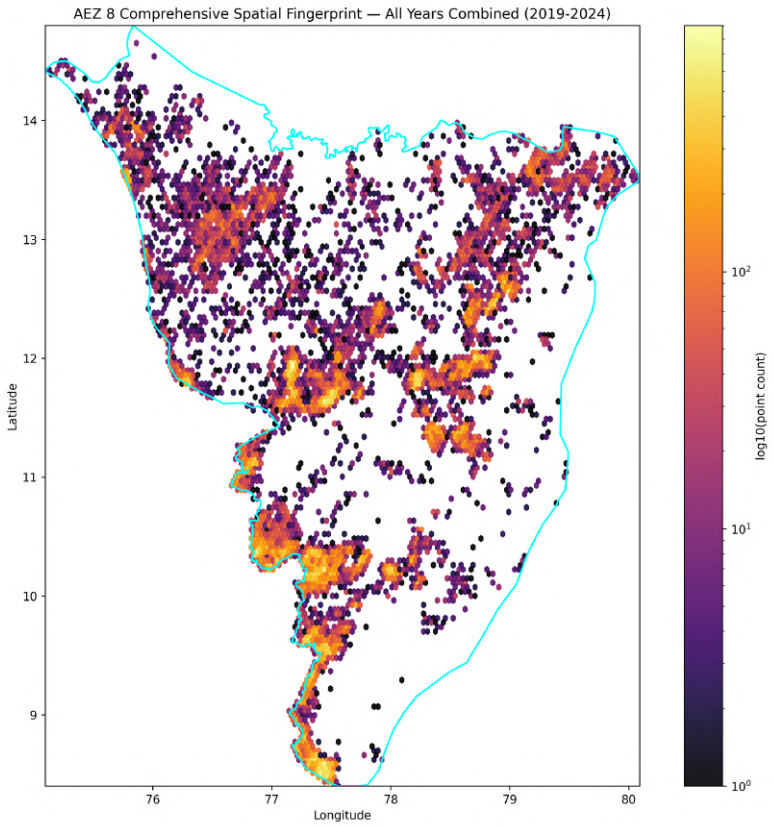
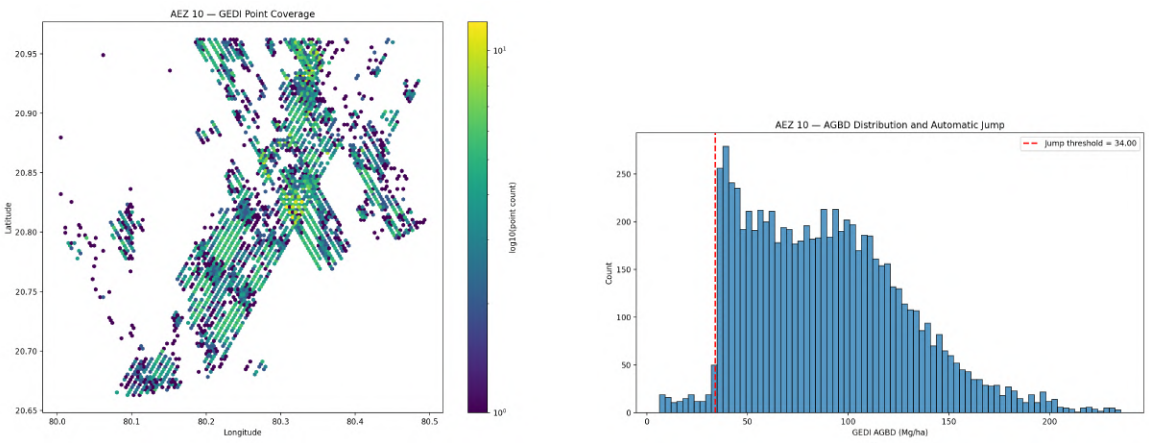


Figure 6.9: AEZ 8 multi-year GEDI coverage visualization from the automated biomass pipeline.

6.10 AEZ 10 Exploration

After AEZ 8, the workflow was also used to inspect AEZ 10. Coverage plots and jump histograms were generated to determine whether better spatial coverage or different distribution regimes existed. These experiments helped identify where the biomass workflow may be more reliable in future work.



(a) AEZ 10 GEDI coverage

(b) AEZ 10 AGBD jump histogram

Figure 6.10: Exploratory AEZ 10 biomass analysis.

6.11 Summary of Biomass Findings

The biomass extension showed that a local RF model can outperform a global product on a controlled GEDI comparison subset. However, the biomass workflow also revealed challenges: GEDI point coverage can be spatially uneven, AGBD distributions can have regime-like jumps, and canopy-density/height layers can have incomplete coverage. These results suggest that future biomass mapping should include stronger ecological stratification, more AEZs, and additional physically meaningful predictors such as terrain, soil, and canopy structure.

Chapter 7

Discussion, Conclusion and Future Work

7.1 Summary of Completed Work

This project successfully demonstrates the architecture and implementation of a fully automated, scalable machine learning pipeline for high-resolution soil nutrient mapping in India.

The rigorous spatial auditing conducted in this study conclusively proves that generalized global soil datasets are fundamentally insufficient for precise localized agricultural planning in the Indian subcontinent due to severe data quantization and climatic biases. By leveraging AEZ-stratified Random Forest models trained on localized 0-15 cm plow-layer ground truth data, this framework successfully overcomes these limitations, providing continuous, high-fidelity mapping of critical soil health indicators. The resulting interactive GEE deployment serves as a foundational digital infrastructure for data-driven precision agriculture and policy intervention.

The biomass extension demonstrated how the same methodological structure can be reused for a different environmental target. GEDI L4A provided AGBD observations, satellite embeddings provided predictors, and CTrees provided a global baseline. The local RF model outperformed CTrees on the same AEZ 8, 2022 GEDI held-out points, while the jump-analysis and CCD/CH diagnostics helped interpret the underlying data distribution.

7.2 What Worked Well

The AEZ-stratified approach was effective for soil health because it reduced spatial heterogeneity and allowed localized models to learn region-specific relationships. The feature stack also produced interpretable results: precipitation, temperature, and texture features appeared as dominant drivers, consistent with agronomic expectations.

The productionization effort was also successful. The corrected raw unmasked assets, explicit CRS/grid handling, and validation reports made the outputs usable for downstream systems. The GEE app provided a practical interface for visual inspection

and value querying.

In biomass, the RF versus CTrees comparison showed that local models can outperform global products when evaluated on local GEDI points. This result motivates further development of local biomass models.

7.3 What Did Not Work Consistently

AEZ merging did not consistently improve soil-health performance. Although sample count affects accuracy in some cases, merging adjacent zones can mix different soil-environment relationships. Therefore, merging should not be automatic.

In biomass, fixed-bin RF did not consistently improve over the baseline RF. The distribution split was real, but simply binning the target was not sufficient to solve high-biomass underprediction. Additional physically meaningful predictors and ecological stratification are likely needed.

7.4 Limitations

The soil-health pipeline depends on the quality and spatial representativeness of SHC samples. Although the dataset is large, the observations are still point-based and may contain laboratory or digitization noise. SoilGrids missing values also propagate into RF predictions because the model requires all predictor bands to be valid.

The biomass extension is exploratory. It focuses mainly on AEZ 8 and selected years, and GEDI coverage is not uniform. CTrees comparison depends on sampling a 100 m global product at GEDI footprint locations. CCD/CH diagnostics are useful but incomplete in coverage.

7.5 Future Work

Future soil-health work can include uncertainty maps, temporal updates, more systematic comparison with independent soil survey data, and production-level integration of downstream LULC masks. Missing-pixel handling could be improved by testing safe imputation strategies, but only where scientifically justified.

Future biomass work should scale the pipeline across more AEZs, use better-covered regions, and add physically meaningful features such as terrain, soil, canopy height, canopy density, forest type, and temporal vegetation metrics. The workflow should

also evaluate other model classes and uncertainty estimation.

7.6 Final Conclusion

Overall, this MTP delivered a production-ready Pan-India soil-health mapping system and established a reusable geospatial machine learning framework for environmental mapping. The soil-health pipeline is complete and deployable, while the biomass extension provides a strong foundation for future carbon and vegetation monitoring work.

Chapter A

Soil-Health Code Organization and Excerpts

This appendix documents the main soil-health code files. The objective is not to explain every line, but to show the role of each file in the reproducible pipeline.

A.1 Data Download

`1_data_download.py` queries the Soil Health Card API, downloads state/district-level feature files, and stores raw JSON/KML-derived information in the local project structure.

```
import os
import json
import requests
from tqdm.auto import tqdm
from itertools import chain
from utils.get_layer_info import get_layer_info
from utils.fetch_features import fetch_kml, parse_kml

DATA_DIR = "./shc_data"
SUB_DIR = "KML_FILES"
ROOT_DATA_PATH = os.path.join(DATA_DIR, SUB_DIR)
os.makedirs(ROOT_DATA_PATH, exist_ok=True)

def get_states_json():
    """
    Fetches the list of states from the Soil Health Card API and saves the response to a JSON file.
    """
    URL = "https://soilhealth4.dac.gov.in/"
    payload = {
        "operationName": "GetState",
        "variables": {},
        "query": "query GetState($getStateId: String, $code: String) {\n  getState(id: $getStateId, code: $code)\n}",
    }
    response = requests.post(URL, json=payload)
    # Ensure the response is successful
    if response.status_code == 200:
```

```
        # Parse the JSON response
        data = response.json()

        # Save the JSON to a file
        with open(os.path.join(ROOT_DATA_PATH, "getStates.json"), "w") as file:
            json.dump(data, file, indent=4)

        print("Response saved to getStates.json")
    else:
        print("Request failed with status code:", response.status_code)


def get_districts_by_state_json(state_hash, state_name):
    """
    Fetches the list of districts for a given state from the Soil Health Card API and saves the response to a JSON file.

    Args:
        state_hash (str): The unique identifier for the state.
        state_name (str): The name of the state.
    """
    URL = "https://soilhealth4.dac.gov.in/"
    payload = {
        "operationName": "GetdistrictAndSubdistrictBystate",
        "variables": {"state": f"{state_hash}"},
        "query": "query GetdistrictAndSubdistrictBystate($getdistrictAndSubdistrictBystateId: String, $name: String, $state: ID, $subdistrict: Boolean, $code: String, $aspirationaldistrict: Boolean) {\n  getdistrictAndSubdistrictBystate(\n    id: $getdistrictAndSubdistrictBystateId\n    name: $name\n    state: $state\n    subdistrict: $subdistrict\n    code: $code\n    aspirationaldistrict: $aspirationaldistrict\n  )\n}",
    }

    response = requests.post(URL, json=payload)
    # Ensure the response is successful
# ... excerpt truncated for thesis draft ...
```

Listing A.1: Excerpt from soil-health data-download script.

A.2 Data Preprocessing

`3_data_preprocessing.ipynb` loads SHC tables, inspects nutrient distributions, removes invalid values, and prepares normalized yearly data for later feature extraction.

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import median_abs_deviation

import yaml

# Load Configuration
with open('config.yaml', 'r') as file:
    config = yaml.safe_load(file)

# Function to plot the properties
def plot_props(df):
    row = 0
    col = 0

    fig, axes = plt.subplots(3, 4, figsize=(20, 10))
    count = 0
    colors = ['red', 'orange', 'yellow', 'green', 'blue', 'violet', '
       pink', 'purple', 'grey', 'brown', 'lime', 'skyblue']

    while row < 3:
        for prop in ['B', 'Mn', 'K', 'N', 'P', 'Cu', 'S', 'OC', 'Zn',
            'pH', 'EC', 'Fe']:
            print(f"{prop} min and max values :  Min = {df[prop].min
               ()}, Max = {df[prop].max()}")

            sns.histplot(df[prop], bins=20, ax=axes[row][col], kde=
               True, kde_kws=dict(cut=3), color=colors[count],
               edgecolor="black")
            axes[row][col].set_title(f"Distribution of {prop}")
            axes[row][col].set_xlabel(f'{prop} values')
            axes[row][col].set_ylabel('Count')
            count += 1
            col = (col+1) % 4

            if count % 4 == 0 and count > 0:
                row += 1

```

```
    plt.tight_layout()
    plt.show()

# Function to calculate Modified Z-Score
def modified_z_score(series):
    median = np.median(series)
    mad = median_abs_deviation(series)
    epsilon = 1e-9 if mad == 0 else 0  # Adjust small value if
       necessary
    mad += epsilon
    return 0.6745 * (series - median) / mad

def apply_z_filter(df):
    df = df[df['OC'] >= 0.0]
    df = df[df['OC'] <= 1.0]
    df = df[df['pH'] >= 1.0]
    df = df[df['pH'] <= 14.0]

    threshold = 3.5
    props = ['B', 'Mn', 'Cu', 'S', 'Zn', 'EC', 'Fe', 'N', 'P', 'K']
# ... excerpt truncated for thesis draft ...
```

Listing A.2: Excerpt from soil-health preprocessing notebook.

A.3 LULC Filtering

`6_lulc_filter.ipynb` loads IndiaSAT/LULC labels and retains agricultural pixels for model training.

```
import os
import glob
import math
import pandas as pd
import numpy as np
import ee

import yaml

# Load Configuration
with open('config.yaml', 'r') as file:
    config = yaml.safe_load(file)

# 1. Initialize GEE
ee.Authenticate()
ee.Initialize(project=config['gee']['project_id'], opt_url='https://
```

```
    earthengine-highvolume.googleapis.com')
print("GEE Initialized successfully.")

# # 2. Define Paths
# INPUT_DIR = "./shc_data/AEZS/AGRI_2023-24/"
# GDRIVE_FOLDER = "GEE_LULC_All_AEZs_ratinder"

# # 3. Load the LULC Asset
# lulc_path = "projects/corestack-datasets/assets/datasets/
    LULC_v3_river_basin/pan_india_lulc_v3_2023_2024"
# lulc_image = ee.Image(lulc_path).select('predicted_label')

# 2. Define Paths and Parameters from Config
INPUT_DIR = config['paths']['aez_data_dir']
DOWNLOAD_DIR = config['paths']['lulc_download_dir']
GDRIVE_FOLDER = config['gdrive_exports']['lulc_folder']

CHUNK_SIZE = 5000
CROP_LABELS = config['parameters']['crop_labels']

# 3. Load the LULC Asset
lulc_path = config['gee']['lulc_asset_path']
lulc_image = ee.Image(lulc_path).select('predicted_label')

print(f"Reading AEZ data from: {INPUT_DIR}")
print(f"Exporting to GDrive folder: {GDRIVE_FOLDER}")

# Get all AEZ files (assuming they follow the pattern AEZ_1_tagged.
    csv, AEZ_2_tagged.csv, etc.)
aez_files = glob.glob(os.path.join(INPUT_DIR, "AEZ_*_tagged.csv"))
# Filter out any files that might already have "LULC" or "Filtered"
    in the name just in case
aez_files = [f for f in aez_files if "LULC" not in f and "Filtered"
    not in f and "Clean" not in f]

print(f"Found {len(aez_files)} AEZ files to process.")

for file_path in aez_files:
    aez_name = os.path.basename(file_path).replace('_tagged.csv', '')
    print(f"\n--- Processing {aez_name} ---")

    # Load and prep index
    df = pd.read_csv(file_path)
    df = df.reset_index().rename(columns={'index': 'orig_idx'})
# ... excerpt truncated for thesis draft ...
```

Listing A.3: Excerpt from LULC filtering notebook.

A.4 Automated Random Forest Training

`8_balanced_RF_model_auto.ipynb` trains all AEZ-nutrient models using the configuration file, dynamic binning, sample weighting, and model serialization.

```
import os
import glob
import optuna
import joblib
import datetime
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import yaml

from sklearn.model_selection import train_test_split, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import VarianceThreshold
from sklearn.metrics import r2_score, mean_squared_error,
    mean_absolute_error

# ============================================================
# 1. Load Configuration
# ============================================================
with open('config.yaml', 'r') as file:
    config = yaml.safe_load(file)

AEZ_CSV_PATH  = config['paths']['aez_data_dir']
OUTPUT_FOLDER = config['paths']['models_joblib_dir']
PLOTS_DIR     = config['paths']['plots_dir']
STATS_FOLDER  = config['paths']['stats_dir']

N_ESTIMATORS     = config['parameters']['rf_n_estimators']
MAX_DEPTH        = config['parameters']['rf_max_depth']
TARGET_AEZS      = config['parameters']['target_aezs']
TARGET_NUTRIENTS = config['parameters']['target_nutrients']

os.makedirs(OUTPUT_FOLDER, exist_ok=True)
os.makedirs(STATS_FOLDER,  exist_ok=True)

# ============================================================
# 2. Constants
# ============================================================

# Bins defined by Soil Health Portal (Govt. of India)
```

```
bin_dict = {
    "N"  : [0, 280, 560, np.inf],
    "P"  : [0, 10, 25, np.inf],
    "K"  : [0, 120, 280, np.inf],
    "OC" : [0.0, 0.50, 0.75, 1.0]
}

selected_feature_names = [
    'temp',
    'SI',
    'RI',
    'precipitation',
    'sand05',
    'silt05',
    'sand515',
# ... excerpt truncated for thesis draft ...
```

Listing A.4: Excerpt from automated RF training notebook.

A.5 RF to CSV Conversion

`9_rf_to_csv.ipynb` converts trained joblib RF models into CSV tree representations that can be loaded into GEE.

```
# ============================================================
# FILE 7 - Batch RF Model  CSV Converter
# Automatically processes ALL .joblib models in the input folder
# ============================================================

import os
import re
import time
import glob
import joblib
import datetime
import ee
from geemap import ml



import yaml

# ------------------------------------------------------------
# 1. Load Configuration
# ------------------------------------------------------------
```

```
with open('config.yaml', 'r') as file:
    config = yaml.safe_load(file)




PROJECT_ID = config['gee']['project_id']
INPUT_FOLDER = config['paths']['models_joblib_dir']
OUTPUT_FOLDER = config['paths']['models_csv_dir']
NUM_PROCESSES = config['parameters']['num_processes']


# ------------------------------------------------------------
# 1. Initialize Earth Engine
# ------------------------------------------------------------
ee.Authenticate()

try:
    ee.Initialize(
        project=PROJECT_ID,
        opt_url='https://earthengine-highvolume.googleapis.com'
    )
    print('[OK] Google Earth Engine initialized successfully!\n')
except ee.EEException as e:
    print('[ERROR] Google Earth Engine failed to initialize!', e)
    raise


os.makedirs(OUTPUT_FOLDER, exist_ok=True)

# Expected filename format: rfr_model_YYYY-MM-DD_AEZ_<N>_<VAR>.joblib
FILE_PATTERN = re.compile(r"rfr_model_(.+)_AEZ_(\d+)_(.+)\.joblib$")

# ------------------------------------------------------------
# ... excerpt truncated for thesis draft ...
```

Listing A.5: Excerpt from RF-to-CSV conversion notebook.

A.6 Raw Inference and Export

`10_predict_export_generate.ipynb` performs raw AEZ-wise inference and exports prediction layers with explicit CRS and grid handling.

```
# # This is the final with crs and crstransform corrected, but before
    all source bands reprojected to final 30m
```

```
# # ============================================================
# # FILE 10 - FINAL RAW INFERENCE & EARTH ENGINE ASSET EXPORT
# # ------------------------------------------------------------
# # PAN-INDIA SOIL HEALTH MAPPING
# #
# # PURPOSE
# #
# # Generate AEZ-wise RAW soil-health prediction rasters for:
# #   N, P, K, OC
# #
# # using the already-trained and locked Random Forest models.
# #
# # FINAL FIXES INCLUDED
# #
# # [OK] No LULC masking during inference
# #    -> Produces raw prediction layers for all inferable pixels.
# #
# # [OK] Explicit output CRS
# #    -> EPSG:4326.
# #
# # [OK] Explicit common export grid using crsTransform
# #    -> All 72 AEZ-level assets align on the same 30 m grid.
# #
# # [OK] No nodata fill values inserted
# #    -> Masked pixels remain masked.
# #
# # [OK] Correct pH band-name handling
# #    -> pH_0-5 / pH_5-15 are safely normalized for GEE classifier
    use.
# #
# # [OK] Existing asset skip logic
# #    -> Safe to rerun without resubmitting completed exports.
# #
# # [OK] Robust error handling and task summary
# #
# # IMPORTANT
# #
# #  Models are NOT retrained or modified.
# #  This exports RAW layers only.
# #  Any LULC masking should be applied later by production users.
# #  The exported "_new" assets have already been validated:
# #     - all 72 exist
# #     - bands correct
# #     - CRS correct
# #     - grid alignment correct
# #     - valid pixels present
# # ============================================================
```

```
48
49 # import os
50 # import re
51 # import glob
52 # import time
53 # import tempfile
54 # import traceback
55
56 # ... excerpt truncated for thesis draft ...
```

Listing A.6: Excerpt from corrected raw inference/export notebook.

A.7 OpenLandMap Export

`11_export_olm_oc.ipynb` exports the raw OpenLandMap OC baseline with corrected unit handling and the same CRS/grid convention.

```
1  # ============================================================
2  # FILE - FINAL RAW OPENLANDMAP OC EXPORT
3  # ------------------------------------------------------------
4  # PAN-INDIA SOIL HEALTH MAPPING
5  #
6  # PURPOSE
7  #
8  # Export a Pan-India OpenLandMap Organic Carbon baseline layer
9  # for comparison with the RF-based OC prediction map.
10 #
11 # FINAL FIXES INCLUDED
12 #
13 # [OK] NO LULC masking
14 #    -> Produces a raw OpenLandMap OC baseline layer.
15 #
16 # [OK] Correct OC unit conversion handled here directly
17 #    -> Previous pipeline exported raw / 10 and app multiplied
18 #       by 5.0, which is equivalent to raw / 2.
19 #    -> This script exports raw / 2 directly.
20 #
21 # [OK] Output band renamed to "OC_OLM"
22 #
23 # [OK] Explicit CRS + 30 m global transform
24 #    -> Consistent with the validated RF export grid.
25 #
26 # [OK] Skip export if asset already exists
27 #
28 # [OK] Robust Earth Engine initialization and clean summary
```

```
#
# IMPORTANT
#
#  This is a comparison baseline, not an RF prediction.
#  No LULC mask is applied.
#  Masked/no-data pixels remain masked.
#  The final GEE app should load this asset directly with
#   NO extra multiply(5.0).
# ============================================================

import ee
import yaml


# ============================================================
# 1. LOAD CONFIGURATION
# ============================================================

print("Loading configuration...")

with open("config.yaml", "r") as file:
    config = yaml.safe_load(file)

PROJECT_ID = config["gee"]["project_id"]


# ============================================================
# ... excerpt truncated for thesis draft ...
```

Listing A.7: Excerpt from OpenLandMap OC export notebook.

Chapter B

Biomass Code Organization and Excerpts

This appendix documents the main biomass workflow files used for CTrees comparison, jump analysis, canopy diagnostics, and baseline versus binned RF experiments.

B.1 CTrees Extraction and Comparison

The CTrees comparison scripts extract CTrees AGB values at GEDI point locations, apply the scale factor of 0.1, and compute metrics against GEDI AGBD.

```
# ============================================================
# TASK 1 - CTREES vs GEDI COMPARISON
# ------------------------------------------------------------
# AEZ 8 | Year 2022
#
# PURPOSE
#
# Compare:
#   1. CTrees 100 m AGB predictions
#   2. GEDI AGBD point values
#
# Also compare CTrees performance against the already-trained
# AEZ 8 RF biomass model on the SAME held-out test subset.
#
# SCIENTIFIC RULE FROM PROFESSOR
#
# - Do NOT reproject GEDI 25 m / point-scale values to 100 m.
# - CTrees values have already been extracted at GEDI lat/lon.
# - Directly compare:
#       GEDI AGBD  vs  CTrees AGB
#
# THIS SCRIPT
#
# [OK] Uses raw GEDI file used during RF training
# [OK] Reproduces exact RF preprocessing:
#      - replace inf with NaN
#      - drop rows with any NaN
#      - keep agbd >= 0
#      - modified Z-score filter, threshold 3.5
```

```
#
# [OK] Reconstructs exact 80/20 train-test split:
#      train_test_split(..., test_size=0.2, random_state=42)
#
# [OK] Loads final CTrees extracted CSV
#
# [OK] Merges CTrees values robustly:
#      - first tries shot_number
#      - falls back to rounded lon/lat
#
# [OK] If RF model exists, compares:
#      - RF on full held-out test set
#      - RF on CTrees-comparable held-out subset
#      - CTrees on the exact same held-out subset
#
# [OK] Also evaluates CTrees on all filtered comparable GEDI points
#
# [OK] Saves:
#      - metrics CSV
#      - counts CSV
#      - point-level comparison CSVs
#      - plots
#      - text report
# ============================================================


# ... excerpt truncated for thesis draft ...
```

Listing B.1: Excerpt from CTrees-vs-GEDI comparison script.

B.2 AGBD Jump Mapping

The jump-analysis script groups AGBD values into low and main regions, creates histograms, and maps the spatial distribution of groups.

```
import os
from pathlib import Path
import yaml
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import median_abs_deviation

ROOT_DIR = Path('/Users/ratinderpalsingh/Documents/cod892_biomass/
    ctreescopy/automated_biomass')
```

```
CONFIG_PATH = ROOT_DIR / 'config' / 'biomass_all_aez_config.yaml'
with open(CONFIG_PATH, 'r') as f:
    config = yaml.safe_load(f)

PATHS = config['paths']
P = config['parameters']
YEARS = P['years']
TARGET_AEZS = P['target_aezs']
POINTS_DIR = PATHS['merged_points_dir']
OUT_DIR = PATHS['jump_analysis_dir']
Z_THRESHOLD = float(P['z_threshold'])
os.makedirs(OUT_DIR, exist_ok=True)

BIN_WIDTH = 1.0
PRE_WINDOW = 5
POST_WINDOW = 3
MIN_PRE_MEAN = 5.0
RATIO_THRESH = 2.0
ABS_DIFF_THRESH = 25.0

def modified_z_score(series):
    median = np.median(series)
    mad = median_abs_deviation(series)
    if mad == 0:
        mad = 1e-9
    return 0.6745 * (series - median) / mad

def compute_sustained_jump(vals, edges, counts, centers):
    smoothed = (
        pd.Series(counts)
        .rolling(window=3, center=True, min_periods=1)
        .mean()
        .to_numpy()
    )
    candidate = None
    for i in range(PRE_WINDOW, len(smoothed) - POST_WINDOW):
        prev_mean = smoothed[i - PRE_WINDOW:i].mean()
        next_mean = smoothed[i:i + POST_WINDOW].mean()
        if prev_mean < MIN_PRE_MEAN:
            continue
        if (next_mean >= RATIO_THRESH * prev_mean and
                (next_mean - prev_mean) >= ABS_DIFF_THRESH):
            candidate = {'idx': i, 'prev_mean': prev_mean, 'next_mean
                ': next_mean}
            break

    if candidate is None:
```

```
56          diffs = np.diff(smoothed)
57          positive_idx = np.where(diffs > 0)[0]
58          best_idx = int(positive_idx[np.argmax(diffs[positive_idx])])
                if len(positive_idx) else int(np.argmax(np.abs(diffs)))
59          return float(edges[best_idx + 1]), '
                fallback_largest_positive_rise', best_idx
60
61      idx = candidate['idx']
62      return float(edges[idx]), 'first_sustained_rise', idx
63
64  def execute_jump_analysis(input_csv_path, timeline_label, aez_key):
65      df0 = pd.read_csv(input_csv_path).replace([np.inf, -np.inf], np.
            nan)
66      df = df0.dropna(subset=['agbd']).copy()
67      df = df[df['agbd'] >= 0].copy()
68      df['agbd_z'] = modified_z_score(df['agbd'])
69      df = df[np.abs(df['agbd_z']) <= Z_THRESHOLD].copy()
70
71      vals = df['agbd'].values
72      q_low, q_high = np.quantile(vals, [0.02, 0.98])
73      work = vals[(vals >= q_low) & (vals <= q_high)]
74      bin_edges = np.arange(np.floor(work.min()) - 1.0, np.ceil(work.
            max()) + 1.0 + BIN_WIDTH, BIN_WIDTH)
75      counts, edges = np.histogram(work, bins=bin_edges)
76      centers = (edges[:-1] + edges[1:]) / 2.0
77
78      threshold, method, highlight_idx = compute_sustained_jump(work,
            edges, counts, centers)
79      df['jump_group'] = np.where(df['agbd'] < threshold, 'before_jump
            ', 'after_jump')
80      before = df[df['jump_group'] == 'before_jump']
81      after = df[df['jump_group'] == 'after_jump']
82
83      group_summary = pd.DataFrame([
84          {'group': 'before_jump', 'n': len(before), 'mean_agbd':
                before['agbd'].mean(), 'median_agbd': before['agbd'].
                median()},
85          {'group': 'after_jump', 'n': len(after), 'mean_agbd': after['
                agbd'].mean(), 'median_agbd': after['agbd'].median()},
86      ])
87      group_summary.to_csv(os.path.join(OUT_DIR, f'aez_{aez_key}_{
            timeline_label}_jump_group_summary.csv'), index=False)
88      df.to_csv(os.path.join(OUT_DIR, f'aez_{aez_key}_{timeline_label}
            _points_with_jump_group.csv'), index=False)
89      after.to_csv(os.path.join(POINTS_DIR, f'gedi_{aez_key}_{
            timeline_label}_agbd_ge35_training.csv'), index=False)
90
```

```
    plt.figure(figsize=(10, 6))
    plt.hist(df['agbd'], bins=80, edgecolor='black', alpha=0.75)
    plt.axvline(threshold, linestyle='--', linewidth=2, label=f'
        Threshold={threshold:.2f}; {method}')
    plt.xlabel('GEDI AGBD (Mg/ha)')
    plt.ylabel('Count')
    plt.legend()
    plt.tight_layout()
    plt.savefig(os.path.join(OUT_DIR, f'aez_{aez_key}_{timeline_label
        }_agbd_jump_histogram_auto.png'), dpi=200)
    plt.close()

for aez in TARGET_AEZS:
    for year in YEARS:
        annual_csv = os.path.join(POINTS_DIR, f'gedi_{aez}_{year}
            _merged_final.csv')
        if os.path.exists(annual_csv):
            execute_jump_analysis(annual_csv, str(year), aez)
    master_csv = os.path.join(POINTS_DIR, f'gedi_{aez}
        _all_years_merged.csv')
    if os.path.exists(master_csv):
        execute_jump_analysis(master_csv, 'all_years', aez)
```

Listing B.2: Excerpt from biomass AGBD jump-mapping script.

B.3 Canopy Cover Density and Height Analysis

The CCD/CH diagnostic script samples canopy structure layers at GEDI points and summarizes canopy class composition across AGBD groups.

```
from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import median_abs_deviation, mannwhitneyu, kruskal

SCRIPT_DIR = Path(__file__).resolve().parent
POINTS_DIR = SCRIPT_DIR.parent / 'data' / 'points'
INPUT_CSV = POINTS_DIR / 'gedi_8_2022_with_indiasat_ccd_ch.csv'
OUT_DIR = SCRIPT_DIR / 'outputs_biomass' / '
    aez8_fixed_threshold_ccd_ch_analysis'
OUT_DIR.mkdir(parents=True, exist_ok=True)

AGBD_COL = 'agbd'
CCD_COL = 'indiasat_canopy_cover_density'
```

```
CH_COL = 'indiasat_canopy_height_class'
Z_THRESHOLD = 3.5

def modified_z_score(series):
    median = np.median(series)
    mad = median_abs_deviation(series)
    if mad == 0:
        mad = 1e-9
    return 0.6745 * (series - median) / mad

def percent_table(df, group_col, label_col):
    tab = df.groupby([group_col, label_col]).size().reset_index(name
        ='count')
    tab['percent_within_group'] = tab.groupby(group_col)['count'].
        transform(lambda x: 100.0 * x / x.sum())
    return tab

def group_summary(df, group_col):
    return df.groupby(group_col).agg(
        n=(AGBD_COL, 'count'),
        agbd_mean=(AGBD_COL, 'mean'),
        agbd_median=(AGBD_COL, 'median'),
        ccd_valid_n=(CCD_COL, lambda x: x.notna().sum()),
        ccd_mean=(CCD_COL, 'mean'),
        ch_valid_n=(CH_COL, lambda x: x.notna().sum()),
        ch_mean=(CH_COL, 'mean'),
    ).reset_index()

def save_bar_plot(class_summary, group_col, label_col, title, xlabel,
    save_path):
    pivot = class_summary.pivot(index=label_col, columns=group_col,
        values='percent_within_group').fillna(0)
    plt.figure(figsize=(8, 5))
    pivot.plot(kind='bar')
    plt.ylabel('Percent within AGBD group')
    plt.xlabel(xlabel)
    plt.title(title)
    plt.xticks(rotation=20)
    plt.tight_layout()
    plt.savefig(save_path, dpi=200)
    plt.close()

df0 = pd.read_csv(INPUT_CSV)
df = df0.replace([np.inf, -np.inf], np.nan).copy()
df['ccd_raw_class'] = df[CCD_COL]
df['ch_raw_class'] = df[CH_COL]

```

```
# IndiaSAT class codes: CCD 2 and CH 8 denote missing/unavailable values.
df[CCD_COL] = df[CCD_COL].replace(2, np.nan)
df[CH_COL] = df[CH_COL].replace(8, np.nan)
df['ccd_label'] = df[CCD_COL].map({0: 'Low Density', 1: 'High Density'})
df['ch_label'] = df[CH_COL].map({0: 'Short Trees', 1: 'Short Trees', 2: 'Medium Trees', 3: 'Medium Trees', 4: 'Medium Trees', 5: 'Medium Trees', 6: 'Tall Trees', 7: 'Tall Trees'})
df['ccd_label_for_counts'] = df['ccd_label'].fillna('Missing/Unavailable')
df['ch_label_for_counts'] = df['ch_label'].fillna('Missing/Unavailable')

df = df.dropna(subset=[AGBD_COL])
df = df[df[AGBD_COL] >= 0].copy()
df['agbd_modified_z'] = modified_z_score(df[AGBD_COL])
df_filtered = df[np.abs(df['agbd_modified_z']) <= Z_THRESHOLD].copy()

df_filtered['group_binary_35'] = np.where(df_filtered[AGBD_COL] < 35, 'AGBD_lt_35', 'AGBD_ge_35')
df_filtered['group_threeway_30_35'] = pd.cut(
    df_filtered[AGBD_COL],
    bins=[-np.inf, 30, 35, np.inf],
    labels=['AGBD_lt_30', 'AGBD_30_to_35', 'AGBD_ge_35'],
    right=False,
)

binary_summary = group_summary(df_filtered, 'group_binary_35')
threeway_summary = group_summary(df_filtered, 'group_threeway_30_35')
threeway_ccd = percent_table(df_filtered, 'group_threeway_30_35', 'ccd_label_for_counts')
threeway_ch = percent_table(df_filtered, 'group_threeway_30_35', 'ch_label_for_counts')

binary_summary.to_csv(OUT_DIR / 'binary35_group_summary.csv', index=False)
threeway_summary.to_csv(OUT_DIR / 'threeway_30_35_group_summary.csv', index=False)
threeway_ccd.to_csv(OUT_DIR / 'threeway_30_35_ccd_class_summary.csv', index=False)
threeway_ch.to_csv(OUT_DIR / 'threeway_30_35_ch_class_summary.csv', index=False)

df_filtered[df_filtered[AGBD_COL] >= 35].to_csv(OUT_DIR / 'gedi_8_2022_agbd_ge_35_candidate.csv', index=False)
df_filtered[df_filtered[AGBD_COL] >= 30].to_csv(OUT_DIR / '
```

```
    gedi_8_2022_agbd_ge_30_candidate.csv', index=False)
```

Listing B.3: Excerpt from canopy density/height diagnostic script.

B.4 GEDI Fetching Automation

The automated biomass workflow includes scripts/notebooks for fetching GEDI exports across years and merging them into model-ready datasets.

```
import os
import re
import time
import random
from pathlib import Path
import yaml
import ee

ROOT_DIR = Path('/Users/ratinderpalsingh/Documents/cod892_biomass/
    ctrees/automated_biomass')
CONFIG_PATH = ROOT_DIR / 'config' / 'biomass_all_aez_config.yaml'
with open(CONFIG_PATH, 'r') as f:
    config = yaml.safe_load(f)

GEE = config['gee']
PATHS = config['paths']
P = config['parameters']
PROJECT_ID = GEE['project_id']
AEZ_ASSET = GEE['aez_asset']
AEZ_FIELD = GEE['aez_field']
EXPORT_FOLDER = GEE['export_folder_drive']
YEARS = P['years']
TARGET_AEZS = P['target_aezs']
TREE_PROB_THRESHOLD = float(P['tree_prob_threshold'])
MIN_TREE_PATCH_PIXELS = int(P['min_tree_patch_pixels'])
EMBED_SCALE = int(P['embed_scale'])
GRID_NX = int(P['grid_nx'])
GRID_NY = int(P['grid_ny'])
SEED = int(P['seed'])
SLOPE_THRESHOLD = float(P['slope_threshold'])
RELATIVE_ERROR_THRESHOLD = float(P['relative_error_threshold'])

ee.Initialize(project=PROJECT_ID, opt_url='https://earthengine-
    highvolume.googleapis.com')

def safe_name(text):
```

```
    return re.sub(r'[^A-Za-z0-9]+', '_', str(text)).strip('_').lower()

def normalize_table_id(tid):
    tid = str(tid)
    return tid if tid.startswith('LARSE/') else f'LARSE/GEDI/GEDI04_A_002/{tid}'

def add_lon_lat(f):
    coords = ee.List(f.geometry().coordinates())
    return f.set({'lon': coords.get(0), 'lat': coords.get(1)})

def add_relative_error(f):
    agbd = ee.Number(f.get('agbd'))
    se = ee.Number(f.get('agbd_se'))
    return f.set('relative_error', se.divide(agbd.max(1e-6)))

def build_tiles(aez_geom, aez_safe):
    bbox = aez_geom.bounds().coordinates().getInfo()[0]
    xs = [p[0] for p in bbox[:-1]]
    ys = [p[1] for p in bbox[:-1]]
    minx, maxx = min(xs), max(xs)
    miny, maxy = min(ys), max(ys)
    lon_edges = [minx + i * (maxx - minx) / GRID_NX for i in range(GRID_NX + 1)]
    lat_edges = [miny + i * (maxy - miny) / GRID_NY for i in range(GRID_NY + 1)]
    tiles = []
    for i in range(GRID_NX):
        for j in range(GRID_NY):
            rect = ee.Geometry.Rectangle([lon_edges[i], lat_edges[j],
                lon_edges[i+1], lat_edges[j+1]], None, False)
            tiles.append({'name': f'{aez_safe}_x{i}_y{j}', 'geom':
                rect.intersection(aez_geom, maxError=500)})
    return tiles

native_bands = [f'A{str(i).zfill(2)}' for i in range(64)]
emb_band_names = [f'emb_{i}' for i in range(64)]
aez_fc = ee.FeatureCollection(AEZ_ASSET)

for YEAR in YEARS:
    startDate = ee.Date.fromYMD(YEAR, 1, 1)
    endDate = startDate.advance(1, 'year')
    for aez in TARGET_AEZS:
        aez_sel = aez_fc.filter(ee.Filter.eq(AEZ_FIELD, aez))
        aez_geom = aez_sel.geometry().simplify(1000)
        embeddings = ee.ImageCollection('GOOGLE/SATELLITE_EMBEDDING/
```

```
            V1/ANNUAL').filterDate(startDate, endDate).filterBounds(
            aez_geom)
76          emb_proj = ee.Image(embeddings.first()).select(0).projection
            ()
77          emb_img = embeddings.mosaic().setDefaultProjection(emb_proj).
            select(native_bands, emb_band_names)
78          dem = ee.ImageCollection('COPERNICUS/DEM/GLO30').filterBounds
            (aez_geom).select('DEM').mosaic().rename('dem')
79          slope = ee.Terrain.slope(dem).rename('slope')
80          grid = ee.Projection('EPSG:3857').atScale(EMBED_SCALE)
81
82          for tile in build_tiles(aez_geom, safe_name(aez)):
83              geom = tile['geom']
84              index = ee.FeatureCollection('LARSE/GEDI/
                GEDI04_A_002_INDEX').filterBounds(geom).filter(
85                  f'time_start >= "{YEAR}-01-01T00:00:00Z" && time_end
                        < "{YEAR+1}-01-01T00:00:00Z"')
86              valid_tables = []
87              for tid in index.aggregate_array('table_id').distinct().
                getInfo():
88                  try:
89                      valid_tables.append(ee.FeatureCollection(
                            normalize_table_id(tid)).filterBounds(geom))
90                  except Exception:
91                      pass
92              if not valid_tables:
93                  continue
94              dw = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1').
                filterDate(startDate, endDate).filterBounds(geom.
                buffer(200))
95              tree_prob = dw.select('trees').mean()
96              dw_label = dw.select('label').mode()
97              is_tree = tree_prob.gte(TREE_PROB_THRESHOLD).And(dw_label
                .eq(1))
98              safe_zone = is_tree.And(is_tree.selfMask().
                connectedPixelCount(MIN_TREE_PATCH_PIXELS, True).gte(
                MIN_TREE_PATCH_PIXELS)).selfMask()
99              predictors = emb_img.clip(geom).reduceResolution(ee.
                Reducer.mean(), bestEffort=True, maxPixels=1024).
                reproject(crs=grid).addBands(
100                 slope.clip(geom).reduceResolution(ee.Reducer.mean(),
                        bestEffort=True, maxPixels=1024).reproject(crs=
                        grid)
101             ).updateMask(safe_zone)
102             gedi = ee.FeatureCollection(valid_tables).flatten().
                filterBounds(geom).filter(ee.Filter.notNull(['agbd',
                'agbd_se'])).filter(
```

```
                ee.Filter.eq('l4_quality_flag', 1)).filter(ee.Filter.
                    eq('degrade_flag', 0)).filter(ee.Filter.eq('
                    algorithm_run_flag', 1)).map(add_lon_lat)
            training = predictors.sampleRegions(collection=gedi,
                properties=['agbd', 'agbd_se', 'shot_number', 'lat',
                'lon'], scale=EMBED_SCALE, tileScale=16, geometries=
                False)
            training = training.map(add_relative_error).filter(ee.
                Filter.notNull(emb_band_names + ['agbd', 'agbd_se', '
                slope', 'relative_error'])).filter(
                ee.Filter.gt('agbd', 0)).filter(ee.Filter.lte('slope
                    ', SLOPE_THRESHOLD)).filter(ee.Filter.lte('
                    relative_error', RELATIVE_ERROR_THRESHOLD))
            export_name = f'gedi_{aez}_{YEAR}_{tile["name"]}'
            ee.batch.Export.table.toDrive(collection=training,
                description=export_name, folder=EXPORT_FOLDER,
                fileNamePrefix=export_name, fileFormat='CSV').start()
```

Listing B.4: Excerpt from GEDI multi-year fetching workflow.

B.5 Baseline and Binned RF Training

The baseline and binned notebooks train Random Forest models for AGBD and compare whether target binning improves performance.

```
import os
from pathlib import Path
import yaml
import optuna
import joblib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import median_abs_deviation
from sklearn.model_selection import train_test_split, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
    mean_absolute_error

ROOT_DIR = Path('/Users/ratinderpalsingh/Documents/cod892_biomass/
    ctreescopy/automated_biomass')
CONFIG_PATH = ROOT_DIR / 'config' / 'biomass_all_aez_config.yaml'
with open(CONFIG_PATH, 'r') as f:
    config = yaml.safe_load(f)
PATHS = config['paths']
```

```
P = config['parameters']
TARGET_AEZS = P['target_aezs']
SEED = int(P['seed'])
POINTS_DIR = PATHS['merged_points_dir']
MODELS_DIR = PATHS['baseline_models_dir']
PLOTS_DIR = PATHS['baseline_plots_dir']
STATS_DIR = PATHS['baseline_stats_dir']
N_ESTIMATORS = int(P['rf_n_estimators'])
MAX_DEPTH = P['rf_max_depth']
OPTUNA_TRIALS = int(P['optuna_trials'])
CV_FOLDS = int(P['cv_folds'])
TEST_SIZE = float(P['test_size'])
Z_THRESHOLD = float(P['z_threshold'])
USE_SLOPE = bool(P['use_slope_as_predictor'])
features = [f'emb_{i}' for i in range(64)] + (['slope'] if USE_SLOPE
    else [])

def smape(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred) / ((np.abs(y_true) + np.
        abs(y_pred)) / 2 + 1e-6)) * 100

def compute_metrics(y_true, y_pred):
    return {
        'R2': round(r2_score(y_true, y_pred), 4),
        'RMSE': round(np.sqrt(mean_squared_error(y_true, y_pred)), 4)
            ,
        'MAE': round(mean_absolute_error(y_true, y_pred), 4),
        'sMAPE': round(smape(y_true, y_pred), 4),
    }

def modified_z_score(series):
    median = np.median(series)
    mad = median_abs_deviation(series)
    if mad == 0:
        mad = 1e-9
    return 0.6745 * (series - median) / mad

def create_objective(X_train_data, y_train_data):
    def objective(trial):
        params = {
            'n_estimators': N_ESTIMATORS,
            'max_depth': MAX_DEPTH,
            'min_samples_split': trial.suggest_int('min_samples_split
                ', 2, 10),
            'min_samples_leaf': trial.suggest_int('min_samples_leaf',
                1, 5),
            'max_features': trial.suggest_float('max_features', 0.3,
```

```
                1.0),
            'random_state': SEED,
            'n_jobs': -1,
        }
        X_tr, X_val, y_tr, y_val = train_test_split(X_train_data,
            y_train_data, test_size=TEST_SIZE, random_state=SEED)
        model = RandomForestRegressor(**params)
        model.fit(X_tr, y_tr)
        return r2_score(y_val, model.predict(X_val))
    return objective

master_rows = []
for aez in TARGET_AEZS:
    path = os.path.join(POINTS_DIR, f'gedi_{aez}
        _all_years_agbd_ge35_training.csv')
    df = pd.read_csv(path).replace([np.inf, -np.inf], np.nan).dropna(
        subset=features + ['agbd'])
    df = df[df['agbd'] >= 0].copy()
    df['z'] = modified_z_score(df['agbd'])
    df = df[np.abs(df['z']) <= Z_THRESHOLD].copy()
    X, y = df[features], df['agbd']
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=TEST_SIZE, random_state=SEED)

    study = optuna.create_study(direction='maximize')
    study.optimize(create_objective(X_train, y_train), n_trials=
        OPTUNA_TRIALS)
    best_params = study.best_params

    kf = KFold(n_splits=CV_FOLDS, shuffle=True, random_state=SEED)
    models, fold_metrics = [], []
    for tr_idx, val_idx in kf.split(X_train):
        model = RandomForestRegressor(n_estimators=N_ESTIMATORS,
            max_depth=MAX_DEPTH, **best_params, random_state=SEED,
            n_jobs=-1)
        model.fit(X_train.iloc[tr_idx], y_train.iloc[tr_idx])
        fold_metrics.append(compute_metrics(y_train.iloc[val_idx],
            model.predict(X_train.iloc[val_idx])))
        models.append(model)
    best_model = models[int(np.argmax([m['R2'] for m in fold_metrics
        ]))]
    test_metrics = compute_metrics(y_test, best_model.predict(X_test)
        )
    joblib.dump(best_model, os.path.join(MODELS_DIR, f'
        rfr_biomass_baseline_AEZ_{aez}.joblib'))
    master_rows.append({'AEZ': aez, **test_metrics})

```

```
pd.DataFrame(master_rows).to_csv(os.path.join(STATS_DIR, '
    pan_india_biomass_baseline_metrics.csv'), index=False)
```

Listing B.5: Excerpt from biomass baseline RF training workflow.

```
import os
from pathlib import Path
import yaml
import optuna
import joblib
import numpy as np
import pandas as pd
from scipy.stats import median_abs_deviation
from sklearn.model_selection import train_test_split, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
    mean_absolute_error

ROOT_DIR = Path('/Users/ratinderpalsingh/Documents/cod892_biomass/
    ctreescopy/automated_biomass')
CONFIG_PATH = ROOT_DIR / 'config' / 'biomass_all_aez_config.yaml'
with open(CONFIG_PATH, 'r') as f:
    config = yaml.safe_load(f)
PATHS = config['paths']
P = config['parameters']
TARGET_AEZS = P['target_aezs']
SEED = int(P['seed'])
POINTS_DIR = PATHS['merged_points_dir']
MODELS_DIR = PATHS['binned_models_dir']
STATS_DIR = PATHS['binned_stats_dir']
N_ESTIMATORS = int(P['rf_n_estimators'])
MAX_DEPTH = P['rf_max_depth']
OPTUNA_TRIALS = int(P['optuna_trials'])
CV_FOLDS = int(P['cv_folds'])
TEST_SIZE = float(P['test_size'])
Z_THRESHOLD = float(P['z_threshold'])
BINNING_METHOD = P.get('binning_method', 'quantile')
FIXED_AGBD_BINS = P.get('fixed_agbd_bins', [35, 50, 75, 100, 125,
    150, 200, 999999])
MIN_BIN_WEIGHT = float(P.get('min_bin_weight', 0.5))
MAX_BIN_WEIGHT = float(P.get('max_bin_weight', 5.0))
features = [f'emb_{i}' for i in range(64)]

def compute_metrics(y_true, y_pred):
    smape = np.mean(np.abs(y_true - y_pred) / ((np.abs(y_true) + np.
        abs(y_pred)) / 2 + 1e-6)) * 100
    return {
```

```
        'R2': round(r2_score(y_true, y_pred), 4),
        'RMSE': round(np.sqrt(mean_squared_error(y_true, y_pred)), 4),
        'MAE': round(mean_absolute_error(y_true, y_pred), 4),
        'sMAPE': round(smape, 4),
    }

def modified_z_score(series):
    med = np.median(series)
    mad = median_abs_deviation(series)
    return 0.6745 * (series - med) / (mad if mad != 0 else 1e-9)

def add_biomass_bins(df):
    df = df.copy()
    if BINNING_METHOD == 'fixed':
        df['bin'] = pd.cut(df['agbd'], bins=FIXED_AGBD_BINS,
            include_lowest=True, right=False)
    else:
        q = min(int(P.get('biomass_quantile_bins', 5)), df['agbd'].
            nunique())
        df['bin'] = pd.qcut(df['agbd'], q=q, duplicates='drop')
    return df.dropna(subset=['bin']).copy()

def make_weights(bin_series):
    counts = bin_series.value_counts()
    median_count = counts.median()
    weights = bin_series.map(lambda b: median_count / counts[b]).
        astype(float)
    return weights.clip(lower=MIN_BIN_WEIGHT, upper=MAX_BIN_WEIGHT)

def create_objective(X_train_data, y_train_data):
    def objective(trial):
        params = {
            'n_estimators': N_ESTIMATORS,
            'max_depth': MAX_DEPTH,
            'min_samples_split': trial.suggest_int('min_samples_split
                ', 2, 10),
            'min_samples_leaf': trial.suggest_int('min_samples_leaf',
                 1, 5),
            'max_features': trial.suggest_float('max_features', 0.3,
                1.0),
            'random_state': SEED,
            'n_jobs': -1,
        }
        X_tr, X_val, y_tr, y_val = train_test_split(X_train_data,
            y_train_data, test_size=TEST_SIZE, random_state=SEED)
        weights = make_weights(X_tr['bin'])
```

```
        model = RandomForestRegressor(**params)
        model.fit(X_tr[features], y_tr, sample_weight=weights)
        return r2_score(y_val, model.predict(X_val[features]))
    return objective

master_rows = []
for aez in TARGET_AEZS:
    path = os.path.join(POINTS_DIR, f'gedi_{aez}_all_years_agbd_ge35_training.csv')
    df = pd.read_csv(path).replace([np.inf, -np.inf], np.nan).dropna(subset=features + ['agbd'])
    df = df[df['agbd'] >= 0].copy()
    df['z'] = modified_z_score(df['agbd'])
    df = df[np.abs(df['z']) <= Z_THRESHOLD].copy()
    df = add_biomass_bins(df)
    X = df[features + ['bin']]
    y = df['agbd']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=TEST_SIZE, random_state=SEED)
    study = optuna.create_study(direction='maximize')
    study.optimize(create_objective(X_train, y_train), n_trials=OPTUNA_TRIALS)
    best_params = study.best_params

    kf = KFold(n_splits=CV_FOLDS, shuffle=True, random_state=SEED)
    models, fold_metrics = [], []
    for tr_idx, val_idx in kf.split(X_train):
        weights = make_weights(X_train.iloc[tr_idx]['bin'])
        model = RandomForestRegressor(n_estimators=N_ESTIMATORS, max_depth=MAX_DEPTH, **best_params, random_state=SEED, n_jobs=-1)
        model.fit(X_train.iloc[tr_idx][features], y_train.iloc[tr_idx], sample_weight=weights)
        fold_metrics.append(compute_metrics(y_train.iloc[val_idx], model.predict(X_train.iloc[val_idx][features])))
        models.append(model)
    best_model = models[int(np.argmax([m['R2'] for m in fold_metrics]))]
    test_metrics = compute_metrics(y_test, best_model.predict(X_test[features]))
    joblib.dump(best_model, os.path.join(MODELS_DIR, f'rfr_biomass_binned_AEZ_{aez}.joblib'))
    master_rows.append({'AEZ': aez, **test_metrics})

pd.DataFrame(master_rows).to_csv(os.path.join(STATS_DIR, 'pan_india_biomass_binned_metrics.csv'), index=False)
```

Listing B.6: Excerpt from biomass binned RF training workflow.

Chapter C

Additional Figures and Tables

C.1 Additional Soil-Health Outputs

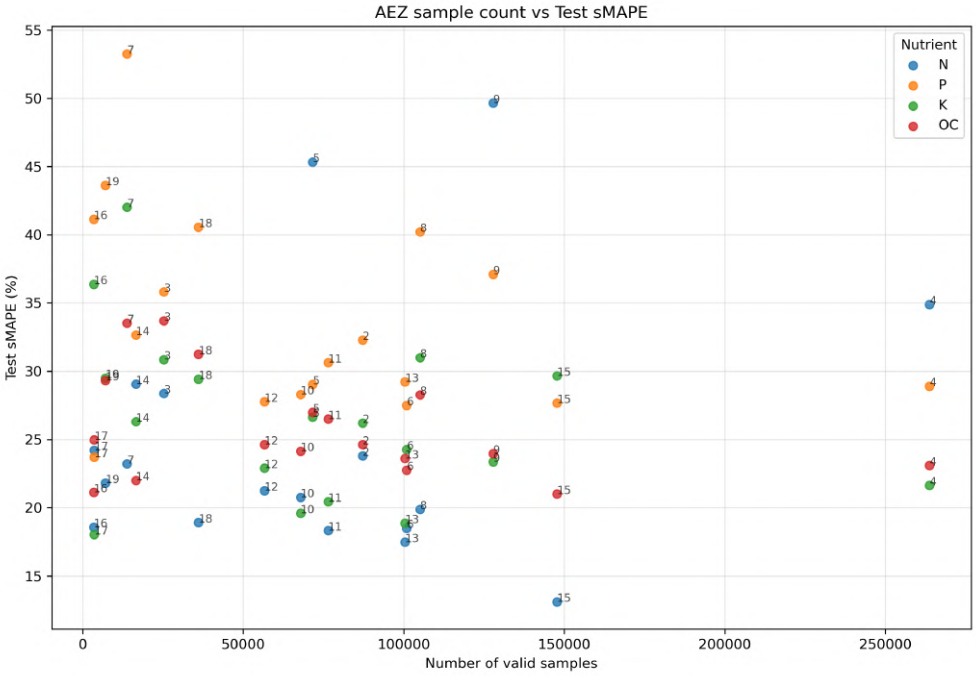


Figure C.1: Sample count versus sMAPE by nutrient.

C.2 Additional Biomass Outputs

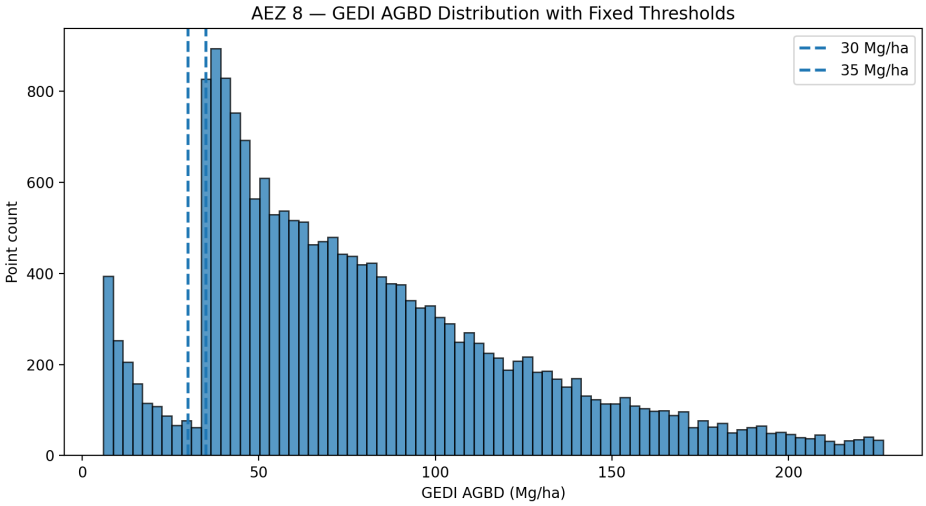


Figure C.2: Fixed-threshold AGBD grouping used for canopy-density and canopy-height diagnostics.

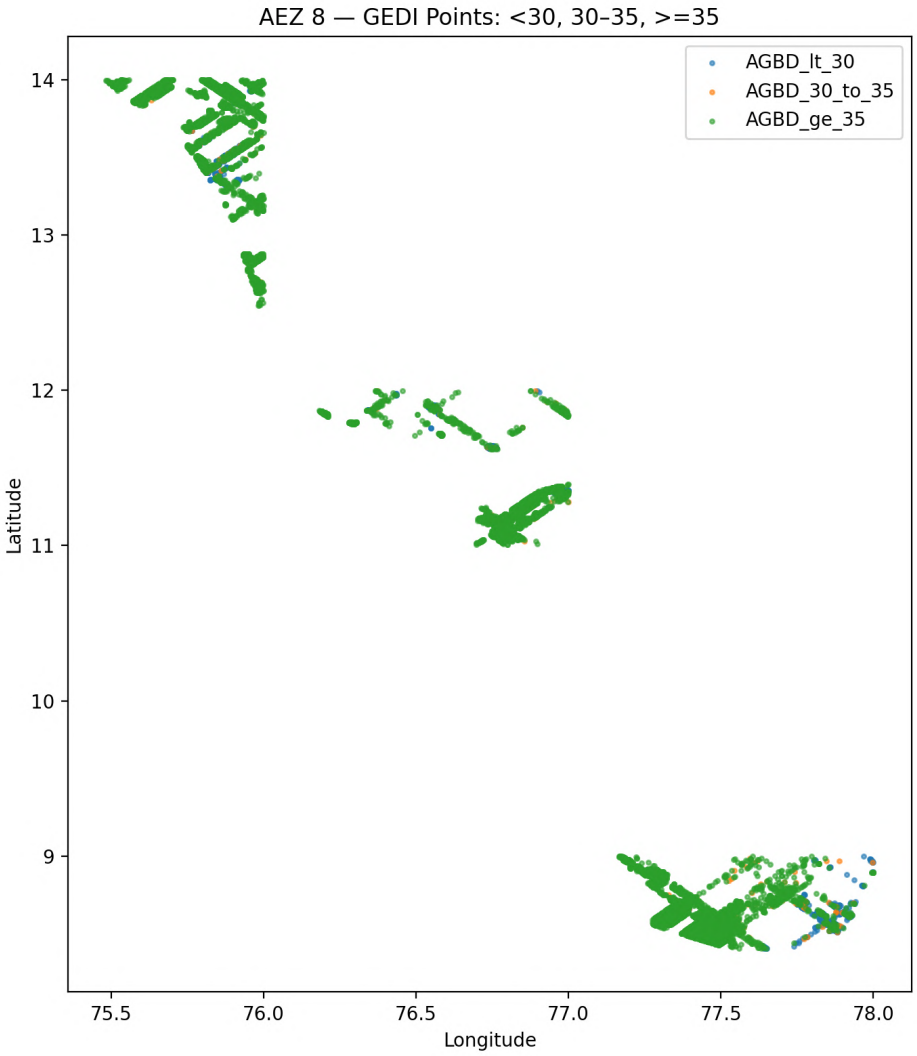


Figure C.3: Three-way AGBD group map used in canopy diagnostics.

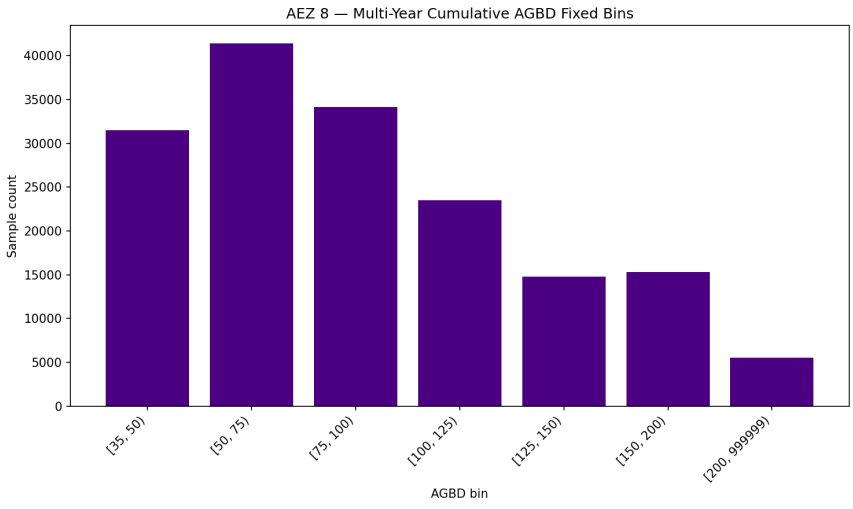


Figure C.4: Binning plot from the AEZ 8 all-years biomass RF experiment.

C.3 Complete Soil-Health Metric Table

The complete 72-row AEZ-by-nutrient model performance table is included in Chapter 5 as Table 5.2. The original source table used to generate it is the project output file `stats/table_master_metrics.tex`.

Bibliography

[1] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., and Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202, 18–27.

[2] Hengl, T. (2018). Soil organic carbon content in x 5 g/kg at 6 standard depths at 250 m resolution. Zenodo. https://doi.org/10.5281/zenodo.2525590

[3] Department of Agriculture and Farmers Welfare. (2024). Soil Health Card Portal, Government of India. https://soilhealth.dac.gov.in/

[4] NASA Earthdata. (2026). Global Ecosystem Dynamics Investigation Lidar. https://www.earthdata.nasa.gov/data/instruments/gedi-lidar

[5] CTrees. (2025). Above-Ground Biomass (AGB) Carbon Data Brief, Version 1.0.

[6] Araza, A., Herold, M., Avitabile, V., and others. (2026). AGBref: A global reference dataset of above-ground forest biomass. Research Square preprint. https://doi.org/10.21203/rs.3.rs-8211898/v1

[7] ISRIC - World Soil Information. (2020). SoilGrids 2.0: Global gridded soil information. https://soilgrids.org/

[8] Singh, R. P. (2026). Soil-Health-Mapping-PanIndia. GitHub repository. https://github.com/Singhratinder/Soil-Health-Mapping-PanIndia