Quantification and Mapping of Land Cover Change in Cabo Verde

José Alberto Martins Fernandes
Master in Computer Science
Department of Computer Science
Faculty of Sciences of the University of Porto
2025

Quantification and Mapping of Land Cover Change in Cabo Verde

José Alberto Martins Fernandes

Dissertation carried out as part of the Master in Computer Science

Department of Computer Science

2025

Supervisor

João Pedro Pedroso, Associate Professor, FCUP

“A cultura da mente recompensa quem persiste em cultivá-la.”

Acknowledgements

Em primeiro lugar, agradeço aos meus pais pela paciência demonstrada e pelo apoio constante ao longo de toda esta jornada académica. Sem eles, a concretização deste objetivo pessoal e profissional seria impensável, tornando esta vitória tanto minha como deles.

Agradeço igualmente a todos os que me acompanharam, aconselharam e encorajaram durante a realização desta dissertação — professores, amigos e colegas — cujos contributos foram determinantes para o desenvolvimento deste trabalho.

Por fim, manifesto o meu agradecimento à Faculdade de Ciências da Universidade do Porto, em particular ao Departamento de Ciência de Computadores, por me ter acolhido de braços abertos e se ter tornado uma verdadeira "segunda casa".

Resumo

Mapas de cobertura do solo atualizados e precisos são essenciais para a monitorização ambiental, a gestão de recursos naturais, o ordenamento do território e os estudos do clima. Esta dissertação apresenta um fluxo de trabalho completo que, de forma sistemática, prevê a presença de terrenos agrícolas em Cabo Verde em diversos anos e quantifica essas alterações usando imagens multitemporais (S2) extraídas do Google Earth Engine (GEE). O processo integra um ciclo iterativo de aprendizagem ativa para reduzir a anotação manual, preservando a reprodutibilidade através de configuração versionada e checkpoints.

Metodologicamente, definimos um processo de exportação reprodutível com três janelas sazonais por ano; construímos vetores por pixel a partir de bandas e índices (NDVI, EVI, NDMI, NDRE, OSAVI), complementados com atributos de terreno (DEM); treinámos e calibrámos classificadores compactos (SVM e RF) e um empilhamento logístico que combina ambos; selecionámos novas anotações segundo critérios de incerteza e diversidade espacial; e transformámos probabilidades em mapas através de limiarização e remoção de pequenos componentes conectados.

Os resultados cruzados mostram que o conjunto empilhado mantém o F_1 acima de 0.87 em ilhas como Boa Vista, Santiago e São Vicente, atingindo 0.98 a 1.00 em Santo Antão e São Nicolau, enquanto Fogo e Maio ficam abaixo de 0.50 porque a amostra rotulada nessas ilhas é pequena. As probabilidades mantêm o Brier Score entre 0.026 e 0.099, o que significa confiança estável nas ilhas com menor erro e oscilações maiores nas restantes. Os mapas finais medem entre 0.09% (Sal) e 1.09% (Santo Antão) de cobertura agrícola, alinhando-se com as zonas irrigadas conhecidas e destacando discrepâncias nos produtos de referência.

Palavras-chave: Sentinel-2; Google Earth Engine; Deteção remota; Mapeamento de culturas; Aprendizagem ativa; Random Forest; Support Vector Machine; Calibração isotónica.

Abstract

Accurate, up-to-date land cover maps are crucial for environmental monitoring, natural resource management, land-use planning, and climate studies. This dissertation delivers a complete workflow that systematically predicts cropland in Cape Verde across multiple years and quantifies temporal change using multi-temporal S2 imagery extracted from GEE. An iterative active-learning loop minimises manual labelling effort, while a versioned configuration and checkpoints keep every run reproducible.

Methodologically, the workflow establishes a reproducible export with three seasonal windows per year; builds per-pixel feature vectors from spectral bands and vegetation indices (e.g., NDVI, EVI, NDMI, NDRE, OSAVI) augmented with terrain attributes derived from a DEM; trains and calibrates compact classifiers (SVM, RF) plus a logistic stacking head; scores new labels by combining uncertainty with spatial diversity; and converts per-pixel probabilities into maps via explicit thresholding and component-size filtering.

The evaluation keeps the stacked ensemble above 0.87 F_1 on Boa Vista, Santiago, and São Vicente, reaches 0.98 to 1.00 on Santo Antão and São Nicolau, and drops below 0.50 on Fogo and Maio because their positive samples remain scarce. Brier scores span 0.026 to 0.099, signalling well-calibrated probabilities on Sal and mild wobble on São Vicente. The final maps estimate cropland shares from 0.09% on Sal up to 1.09% on Santo Antão, matching known irrigated valleys and exposing gaps in the official land-cover products.

Keywords: Sentinel-2; Google Earth Engine; Remote sensing; Cropland mapping; Active learning; Random Forest; Support Vector Machine; Isotonic calibration.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AUC Area Under the ROC Curve

AUC-PR Area Under the Precision–Recall Curve

BSI Bare Soil Index

CSV Comma-Separated Values

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DEM Digital Elevation Model

EVI Enhanced Vegetation Index

EVI2 Two-band Enhanced Vegetation Index

F1 Harmonic mean of precision and recall

GDAL Geospatial Data Abstraction Library

GEE Google Earth Engine

GIS Geographic Information System

GNDVI Green Normalized Difference Vegetation Index

KML Keyhole Markup Language

NASADEM NASA Digital Elevation Model

NDMI Normalized Difference Moisture Index

NDRE Normalized Difference Red Edge

NDVI Normalized Difference Vegetation Index

NDWI Normalized Difference Water Index

OSAVI Optimized Soil Adjusted Vegetation Index

PR Precision–Recall

RF Random Forest

ROC Receiver Operating Characteristic

S2 Sentinel-2

SVM Support Vector Machine

XGBoost Extreme Gradient Boosting

QGIS Quantum Geographic Information System

GeoTIFF Geographic Tagged Image File Format

Chapter 1

Introduction

This thesis builds a modular pipeline that assembles satellite data, focused annotations, and reproducible automation to map Cape Verdean cropland and its yearly changes. The emphasis remains on the engineering challenge itself: orchestrating data ingestion, feature construction, model training, and iteration so the maps improve with every round. The pipeline automates the steps from data collection to draft products while keeping the user in the loop. Each iteration proposes pixels that are expected to refine crop delineations, meaning every annotation contributes directly to decision-ready outputs.

The workflow ingests multi-temporal S2 imagery served by GEE, extracts spectral, vegetation, and terrain descriptors across three seasonal windows (January–April, May–August, September–December), and trains calibrated classifiers—either a radial-basis SVM, a RF, or a logistic stacking ensemble of both. Probabilistic maps are converted into rasters and KML layers for inspection in Google Earth Pro, and the same configuration is replayed year by year to produce a consistent series for 2019 and 2023.

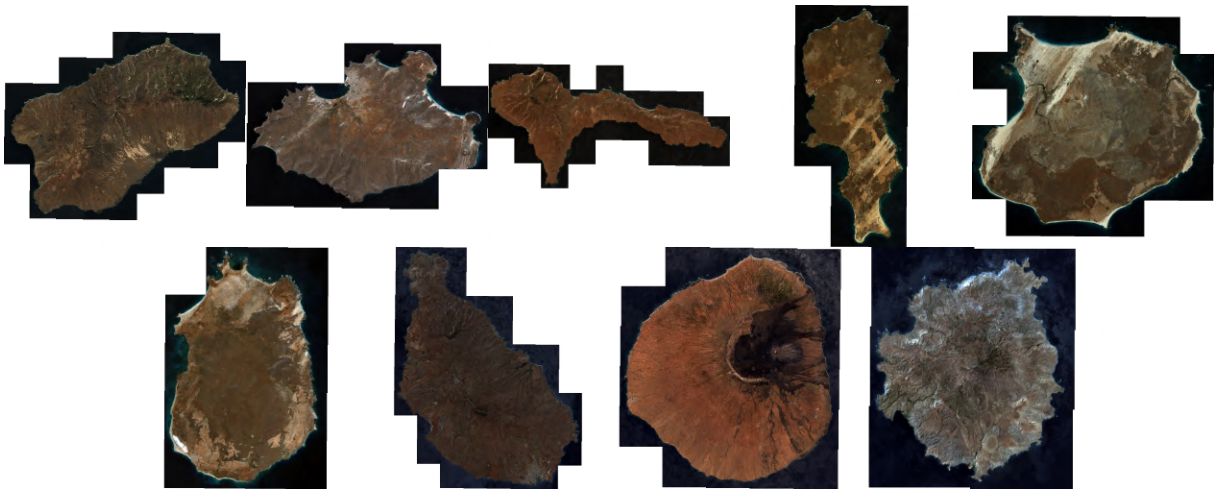


Figure 1.1: Cape Verde S2 imagery for the cropland-monitoring pipeline.

Historical context

Cape Verde's farming systems have been shaped by centuries of water scarcity, highly variable rainfall, and recurrent drought. Colonial records document major famines in 1747–1748, 1831–1833, 1863–1866, 1902–1903, 1941–1943, and 1968–1973, each triggered by multi-year rainfall failure and fragile food supply chains [16]. Even after independence in 1975, the archipelago endured an 18-year drought (1968–1986) that accelerated emigration and led to a heavy reliance on international food aid [16]. Less than 10% of the terrain is arable, there are no permanent surface water bodies, and irrigated plots cover only about 1 400 hectares, mostly in small terraces perched on volcanic slopes [16, 22]. The structural water deficit is compounded by the fact that around 80% of food is imported, and agriculture contributes less than 5% of the gross domestic product, leaving rural livelihoods extremely vulnerable to price shocks and climate variability [9, 14]. Similar constraints affect many Small Island Developing States, where limited arable land and high logistics costs make spatially explicit evidence a prerequisite for resilient agrifood strategies [7]. These conditions make a transparent, evidence-driven cropland monitoring system more than an academic exercise; it underpins food security planning, drought response, and the justification for climate-resilience financing.

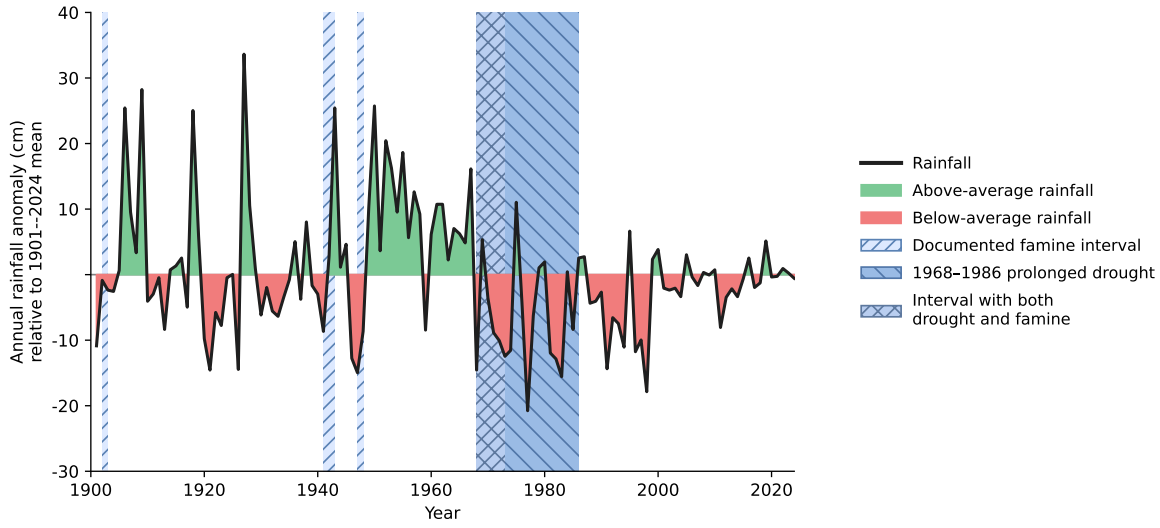


Figure 1.2: Historical drought timeline for Cape Verde (1901–2024). Rainfall anomalies are computed from CRU TS 4.09 precipitation totals accessed via the ExtremeWeatherWatch archive, the instrumental record begins in 1901, and shaded intervals denote the famine years and prolonged drought documented in Cape Verdean historical records.[8, 12, 16]

Decision-making landscape

Cape Verde's public institutions rely on reliable geospatial evidence to coordinate drought response programs, manage irrigation quotas, and audit agricultural incentives. Preliminary

interviews with municipal officers and Ministry of Agriculture analysts highlighted three recurring bottlenecks: (i) national statistics aggregate cultivated area at coarse resolution and lag by several seasons; (ii) on-the-ground surveys cover only easily accessible valleys, leaving highland terraces under-reported; and (iii) disparate community mapping initiatives use incompatible classification schemes [1]. The pipeline addresses these gaps by supplying version-controlled scripts, comparable outputs for every island, and change-ready artefacts that can be audited round after round. The same stakeholders emphasised four design priorities [1]:

- **Targeted labelling.** Request manual annotations to sharpen crop delineations or expand spatial coverage.
- **Reproducibility.** Keep seasonal windows, configuration files, and logs explicit so that future users can replay each run.
- **Interpretability.** Provide probability maps with clear threshold semantics and component-size rules that senior agronomists can approve quickly.
- **Modular reruns.** Allow users to rerun individual stages, such as candidate selection or polygonisation, without restarting the whole pipeline.

These priorities shaped both the scripting style and the documentation structure reported across the dissertation.

Manual labelling as a bottleneck

Unlike regions with established reference datasets, Cape Verde lacks an accessible, vetted inventory of agricultural parcels for the period studied. Labels were therefore created purely by visually interpreting high-resolution imagery. This process remains subjective because pixels often mix multiple land uses, seasonal differences can confuse even skilled annotators, cloud cover occasionally hides key features, and some areas lack suitable high-resolution scenes for the exact year under review. Mislabelled examples introduce noise that can mislead supervised models and inflate evaluation metrics if left unchecked [17, 18]. The pipeline mitigates these risks by restricting new label requests to informative, spatially diverse pixels, recording provenance along with "skip" decisions so that suspect samples can be revisited, and preserving configuration snapshots alongside probability rasters to trace the effect of any correction without repeating the full export or training cycle. Even with unavoidable human error, these safeguards ensure that the resulting cropland maps are auditable, improvable, and helpful in planning interventions.

1.1 Objectives

Guided by the project brief and early scoping discussions, this thesis turns the broad ambition of island-wide crop monitoring into a set of actionable objectives:

- select representative islands and sites for piloting the workflow;
- implement scripts and automation to collect and preprocess seasonal S2 mosaics concatenated with derived indices;
- construct an initial labelled dataset for 2019 and 2023 through focused annotation sessions and active-learning guidance;
- apply calibrated state-of-the-art classifiers to generate annual cropland maps and change summaries that respect the documented configuration;
- design the codebase so major blocks—feature extraction, modelling, post-processing—remain modular, allowing future work to plug in new methods or swap components without rewriting the pipeline; and
- test and document the system so that future users can extend it to additional islands or crop types without needing to rebuild the tooling.

Together, these objectives provide a blueprint for moving from raw satellite catalogues to actionable cropland layers, while leaving space for future campaigns to build upon on the same foundation.

1.2 Contributions

This dissertation contributes artefacts and lessons that extend beyond a single case study:

- a practical labelling and training loop that couples uncertainty sampling with diversity-aware thinning to avoid redundant pixel requests and make every annotation impactful;
- an implementation linking GEE exports, feature construction, calibrated model training, auto-tuned inference, and post-processing into a single orchestrated script;
- a reproducible set of artefacts—metrics, overlays, candidate lists, and configuration snapshots—that document each round and enable downstream audits and change analysis; and
- guidance on how classical calibrated classifiers plus active learning can support island-scale cropland mapping when paired with transparent configuration and focused labelling.

These contributions are intentionally reusable: later teams can adopt the workflow as-is, cherry-pick individual modules, or extend the acquisition policy and post-processing stack for larger annotation campaigns.

1.3 Thesis Outline

The remainder of this dissertation is organised so readers can trace the work from background to deployment guidance:

- Chapter 2 reviews the scientific and technical foundations, covering satellite products, cropland mapping methods, active learning for remote sensing, and supporting software ecosystems.
- Chapter 3 explains the methodology, detailing how data are prepared, features are engineered, models are trained and calibrated, uncertainty is managed, and outputs are validated.
- Chapter 4 describes the implementation, focusing on system architecture, orchestration scripts, data management patterns, and safeguards that ensure the runs are reproducible and efficient.
- Chapter 5 presents the results and discussion, showcasing quantitative metrics, qualitative map assessments, and lessons drawn from the São Nicolau case study.
- Chapter 6 concludes the thesis, summarising findings, limitations, future work, and the roadmap toward operational deployment.

Chapter 2

State of the Art

This chapter reviews prior work on satellite-based land-cover mapping and change detection, as well as machine-learning methods most relevant to per-pixel classification with focused label budgets, active learning in remote sensing, and the tooling that underpins scalable experimentation.

2.1 Satellite Data for Crop Mapping

2.1.1 Sentinel-2 mission and products

The S2 mission provides multispectral imagery with a spatial resolution of 10–60 meters, covering visible, near-infrared, shortwave-infrared, and red-edge bands on twin satellites [6]. The Level-2A surface reflectance product is particularly suited to crop monitoring because atmospheric and topographic corrections are applied systematically, allowing analysts to compare tiles acquired under different illumination or aerosol conditions. Each granule spans a 100×100 km footprint (UTM tiling), which simplifies bookkeeping for archipelagic regions where islands occupy only a fraction of the scene. Tables 3 and 4 in Appendix C list the spectral, index-based, and ancillary layers used later in the pipeline, allowing readers to cross-check terminology.

For cropland mapping, the most informative spectral regions include the near-infrared plateau (B8/B8A), red-edge bands (B5–B7), and shortwave infrared absorptions (B11/B12) that respond to canopy moisture. Atmospheric bands (B1, B9, B10) are primarily used for cloud screening and can be dropped once Level-2A products are in use. Derived indices, such as NDVI, EVI, NDMI, NDRE, GNDVI, and OSAVI, translate these raw measurements into interpretable vegetation and soil indicators that strengthen class separability across phenological stages.

Operational pipelines typically supplement Level-2A data with auxiliary layers: digital elevation models (for illumination correction), water masks (to prevent shoreline artefacts), and per-pixel quality flags that isolate cloud, cirrus, and shadow contamination. In the Cape

Verde context, these ancillary datasets are essential because steep volcanic terrain introduces sharp illumination gradients, and coastal fog can bias the blue and green bands. Platforms such as GEE streamline the fusion of these resources by exposing curated catalogues and server-side processing, which this project leverages to maintain reproducible preprocessing [11].

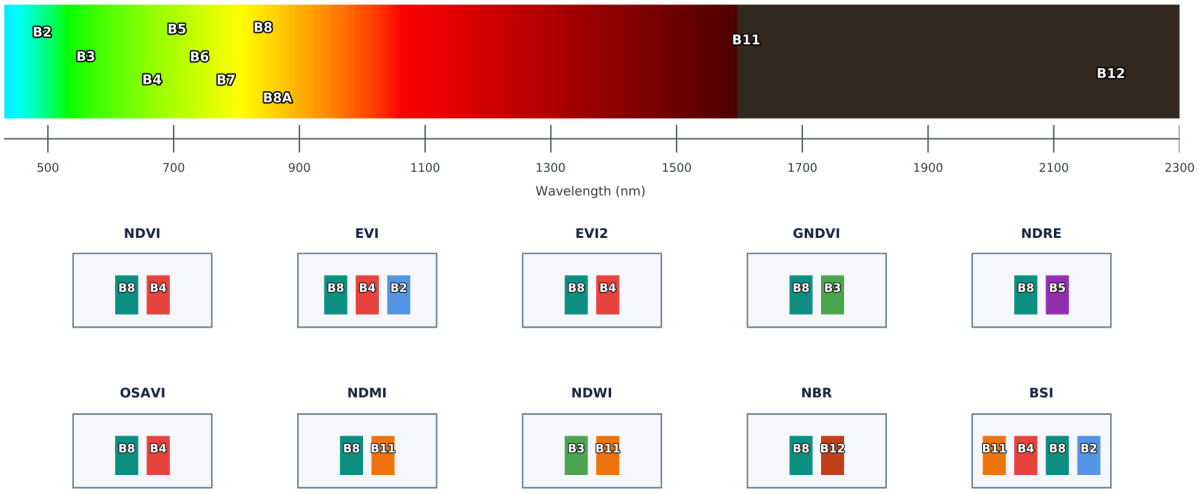


Figure 2.1: Sentinel-2 spectral coverage used in the pipeline. The upper band shows the Level-2A bands retained (with native resolutions), and the lower panel links each band combination to the vegetation, moisture, and soil indices computed for the seasonal stacks.[6, 11]

2.1.2 Temporal information and phenology

Cropland exhibits strong phenological cycles. Aggregating observations into seasonal windows captures key growth and senescence phases without requiring dense time series. This compromise is common in operational workflows that must balance information content with export limits and compute cost. For Cape Verde, dry-season composites emphasise the spectral contrast between irrigated plots and surrounding scrub, whereas rainy-season mosaics expose emergence and peak canopy stages. Prior work on Guinea-Bissau cashew orchards combined such seasonal stacks with harmonic change detection to monitor plantation expansion [19], illustrating the benefit of synchronising acquisition windows with agricultural calendars.

2.1.3 Derived temporal and spatial descriptors

Temporal diagnostics, such as Continuous Change Detection and Classification (CCDC), fit harmonic regressions to pixel histories and highlight abrupt shifts in spectral behaviour. They have proven valuable for multi-year deforestation monitoring, but demand careful parameter tuning and substantial computing resources, especially when applied to Sentinel-2's dense revisit schedule. Texture techniques, including gray-level co-occurrence matrices (GLCM), summarise local spatial patterns in $[3 \times 3]$ or $[5 \times 5]$ neighbourhoods and often boost discrimination between

plantations and mixed savanna. The 2024 cashew-mapping pipeline combined CCDC coefficients and GLCM statistics, creating hundreds of candidate features. Our workflow leverages this insight—spatial context and temporal stability are beneficial—while retaining only lightweight derivatives (seasonal NDVI textures and replicated terrain metrics) to keep processing manageable within the iterative pipeline.

Recent studies also explore phenology-driven descriptors such as start-of-season metrics, cumulative greenness, or harmonic amplitude ratios. While they capture agronomic dynamics, they typically require gap-free time series or sophisticated interpolation. For islands with intermittent cloud cover and limited computing budgets, seasonal composites offer a pragmatic compromise: they reduce temporal noise while preserving clear differences between irrigated terraces and fallow land.

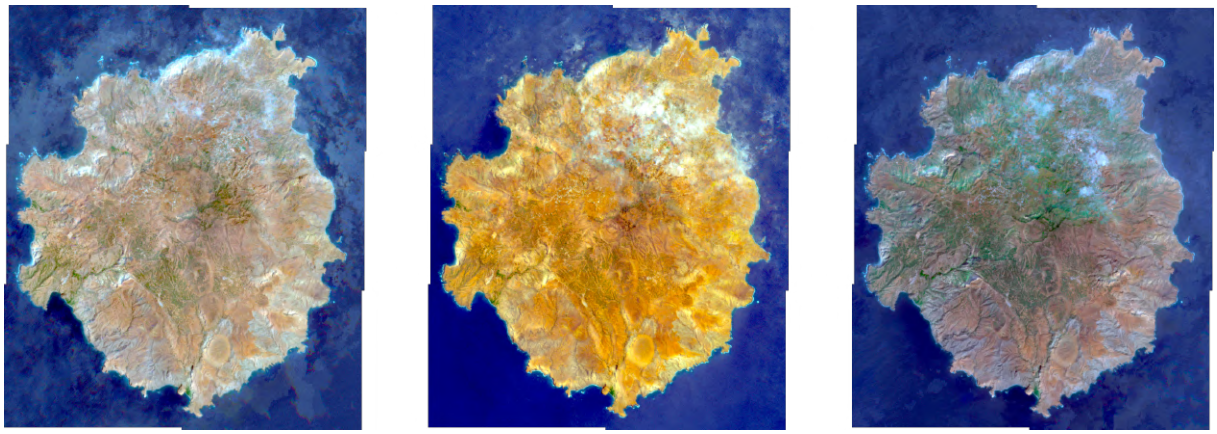


Figure 2.2: Seasonal Sentinel-2 composites illustrating phenological contrast.

2.2 Per-pixel Classification Methods

2.2.1 Classical machine learning

SVM and RF remain popular for land-cover mapping with limited labels because they cope well with engineered, tabular features and generalise from relatively small training sets [2, 3, 5]. Lightweight ensembles that stack calibrated base learners with logistic regression offer a competitive middle ground between single-model baselines and deep networks, while mantaining a transparent inference. Gradient-boosted trees, such as XGBoost, looked attractive; however, in practice, they required constant retuning of the learning rate, tree depth, and subsampling to stay stable. Even broad searches never outperformed the calibrated SVM, so the extra maintenance and dependency footprint were dropped in favour of a simpler stack [4].

Classical models also offer intuitive diagnostics, including feature importances, partial dependence, and straightforward calibration curves. In low-label regimes, they tend to outperform heavy neural networks because they benefit directly from domain knowledge encoded

in vegetation indices, terrain metrics, and seasonal summaries. Calibration layers such as `CalibratedClassifierCV` remain essential when probabilities drive both postprocessing thresholds and the active-learning acquisition score.

2.2.2 Deep learning approaches

Convolutional encoders, temporal convolutional networks, and recurrent architectures dominate large-scale crop-type mapping when dense labels and accelerated compute budgets are available [21, 26]. Architectures such as ResNet classifiers and U-Net segmentation decoders learn rich spatial features end-to-end when thousands of training patches are available. Recent surveys document state-of-the-art cropland accuracies when ample annotated scenes and GPU infrastructure are provided [13, 15, 20]. Their strengths include automatic feature learning, the ability to ingest long Sentinel-2 time series directly, and joint optimisation of spatial and temporal cues. However, they demand extensive annotation, a responsibility that exceeds the scope of the current phase. Hybrid strategies that pair patch-based CNNs with tabular classifiers are promising, but they require consistent tiling schemes and patch-level ground truth that the current label set does not yet provide. Deep learning, therefore, remains a priority for later phases once additional labels and compute resources are secured.

2.3 Active Learning in Remote Sensing

Active learning reduces annotation costs by querying the most informative samples. Widely used query strategies include uncertainty sampling (e.g., Breaking Ties for ensembles, Margin Sampling for margin-based models), query-by-committee, and diversity-enhanced selections based on clustering or coverage [23, 24]. Remote-sensing studies show clear gains when spatial diversity is enforced, thereby avoiding near-duplicate pixels [18]. The acquisition strategy implemented in this project follows those findings: it combines an uncertainty band around the decision threshold, a targeted “hard negative” window that focuses on vegetation spectra close to cropland, and a DBSCAN-based diversity filter to spread requests across islands and ecological zones.

Other authors propose cost-sensitive priority policies that account for travel time between field sites or prioritise areas with high socio-economic impact [23, 24], helping to plan field visits for ground-truth validation. While our current setup focuses on spectral informativeness, the modular acquisition score leaves room for future extensions, such as those where logistics or policy-driven priorities influence the candidate ranking.

2.4 Scalable Platforms and Tooling

GEE enables large-scale computation close to the data, simplifying access to multi-temporal imagery, DEM derivatives, and mosaicking tools [11]. Local processing typically relies on GDAL/Rasterio for efficient raster I/O and on vector formats such as KML for interactive inspection in Google Earth. Sustainable pipelines integrate these tools to maintain a transparent validation loop.

Earth Engine's Python API provides further support for automated export monitoring, queue management, and feature sampling [11]. In practice, notebook prototypes often evolve into scripted pipelines once monitoring requirements become more stringent. The present project adopts the latter stance: exports, sampling, and inference are scripted to guarantee reproducibility and to ease integration with version control.

2.5 Preprocessing Workflows

End-to-end land-cover projects typically orchestrate a chain of pre-processing tasks: tile selection, quality filtering, compositing, and feature extraction. Automation is key because manual export management does not scale once multiple seasons and islands are involved. Earth Engine offers task monitoring but lacks built-in retry logic, motivating the use of custom scripts that poll job status and resubmit failed exports. Locally, the combination of GDAL and Rasterio supports reprojection, clipping, and conversion to analysis-friendly formats. Cloud-optimised GeoTIFFs are increasingly popular for streaming large rasters without fully downloading them; while not yet adopted in this pipeline, they remain a practical enhancement for future iterations where network bandwidth is constrained [10].

2.6 Change Detection and Postprocessing

Change detection in optical remote sensing often involves comparing consistent annual products to derive transitions from them. Robust practice emphasises comparable inputs, transparent decision thresholds, and explicit treatment of uncertainty. Postprocessing steps, such as thresholding (e.g., classifying pixels with a probability of ≥ 0.35 as agricultural) and sieving (removing components smaller than five pixels), produce maps that balance fidelity and readability.

2.7 Label Scarcity and Human-in-the-loop Practices

Obtaining reliable annotations is particularly challenging in regions where ground teams must navigate rugged terrain or limited transport. Studies on active learning for land cover [18]

recommend combining uncertainty heuristics with annotator feedback to flag ambiguous samples and to record reasons for deferral. Experience from the 2024 cashew project showed that patch-based labelling (e.g., 3×3 windows) accelerates annotation while introducing spatial autocorrelation that must be considered during train–test splitting. Capturing annotator notes and skipped pixels proved invaluable for subsequent rounds, motivating the metadata fields adopted in the present pipeline.

Crowdsourcing or manual annotation introduces label noise—mislabelled pixels, inconsistent class boundaries, or positional errors—that can bias model training and degrade evaluation scores. Recent remote-sensing and machine-learning studies highlight that even small amounts of label noise can propagate through supervised models, skewing precision and recall, and necessitate targeted mitigation strategies such as noise filtering, confidence-weighted loss functions, or iterative relabeling campaigns [17, 18]. These findings reinforce the importance of audit trails, notes, and reproducible configuration files so that suspect samples can be revisited and corrected without requiring the rebuilding of the entire pipeline.

2.8 Challenges in Small-Island Remote Sensing

Small-island developing states face unique constraints, including a scarcity of on-site experts, rapidly changing microclimates, and heterogeneous terrain within short distances. These factors make compact, self-contained workflows attractive. They also emphasise the importance of logging and traceability, as a single analyst may hand the project to another team mid-campaign. The Cape Verde use case shares these characteristics. By building on lessons from Guinea-Bissau cashew monitoring while emphasising reproducibility, the present work contributes to filling this gap.

2.9 Applied Studies and Regional Context

Regional case studies inform design choices for this thesis. In Guinea-Bissau, "Mapping Cashew Orchards in Cantanhez National Park (Guinea-Bissau)"[19] maps cashew orchards with engineered features and classical classifiers, underscoring the value of seasonal compositing. "Remote Sensing and Machine Learning Tools for Vegetation Monitoring"[25] surveyed remote-sensing and machine-learning tools for vegetation monitoring. highlighting the flexibility of S2 and GEE. For label efficiency, "Active Learning for Improved Landcover Classification"[18] analysed active-learning strategies for land-cover classification, reporting gains when uncertain and spatially diverse candidates are prioritised. These works support the choices made in the present pipeline.

2.10 Summary

In summary, high-resolution optical imagery combined with engineered features and compact classifiers can deliver accurate per-pixel crop maps under tight label budgets. Active learning helps concentrate annotation effort where it matters most, while reproducible tooling and explicit postprocessing keep the resulting maps auditable and actionable.

Chapter 3

Methodology

This chapter describes the methodology that underpins the cropland-mapping pipeline. The narrative follows the same order as the software: configuration-driven data preparation, feature construction, model training with automatic hyperparameter search, active learning candidate management, annotation governance, post-processing, and evaluation. The emphasis is on reproducible automation rather than one-off processing so that every inhabited island can be analysed year by year with minimal manual intervention.

3.1 Pipeline Overview

The pipeline is implemented as a family of Python scripts located in the `scripts/` directory. The orchestration entry point, `ready_to_run_phase1.py`, invokes the environment checks, schedules Google Earth Engine (GEE) exports, runs active-learning rounds, and launches the post-processing sweep. Every stage reads its configuration from `scripts/config.py`, which captures seasonal windows, tiling choices, derived indices, model options, and stopping rules. Because that configuration file is version-controlled, reproducing a run simply requires reusing it together with the saved round outputs. Progress reporting and logging rely on the shared utilities in `scripts/progress\_utils.py`, which keep long-running jobs observable without requiring manual instrumentation. Figure 4.1 sketches the interplay between these components.

3.2 Data Preparation

3.2.1 Config-driven exports

The system is designed to process every inhabited island of Cape Verde. The operator selects the island identifier in `config.py`; the export script then reads the corresponding polygon, seasonal windows, and naming conventions. `a0_setup_check.py` verifies that GEE authentication

is available, that required directories exist, and that disk space is sufficient. Once validated, `a1_phase1_data_download.py` issues the seasonal export tasks. Each tile is named using the pattern `<island>_tile<index>`, allowing downstream scripts to infer provenance simply from the filename. The script prints the GEE task identifiers, enabling operators to follow the export status on the Earth Engine console.

3.2.2 Seasonal compositing and indices

For each seasonal window (January–April, May–August, September–December), the export script filters the S2 Level-2A surface reflectance collection by `CLOUDY_PIXEL_PERCENTAGE` and masks the `QA60` cloud and cirrus bits. A median composite is computed for the remaining observations to suppress residual cloud streaks.

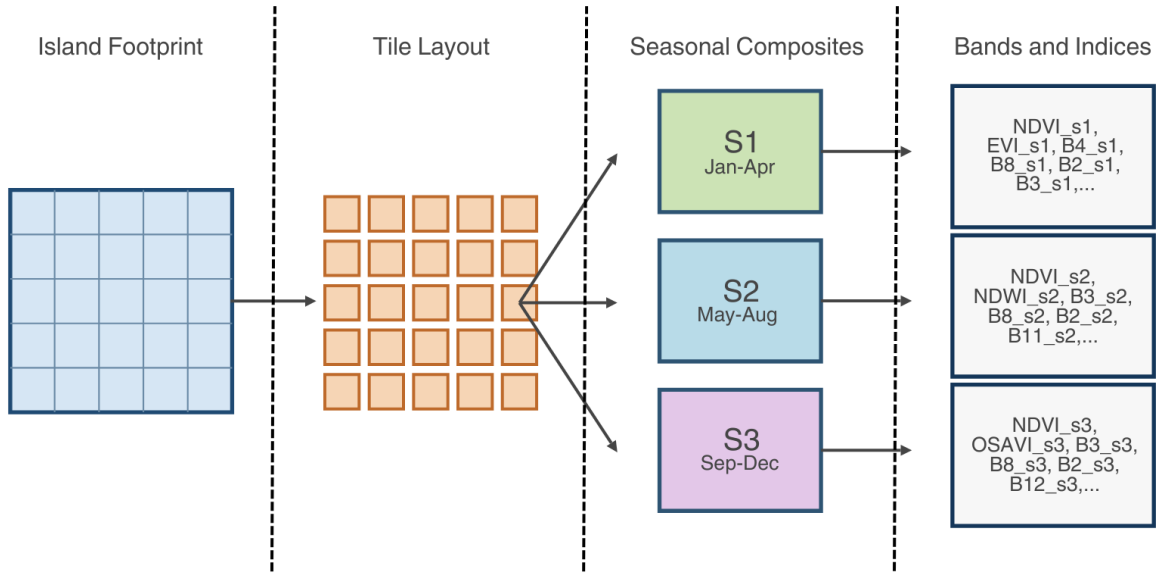


Figure 3.1: From island footprint to seasonal stacks. The 5×5 tiling keeps exports within Earth Engine limits, a representative tile fans out into the three seasonal composites (*Jan–Apr*, *May–Aug*, *Sep–Dec*), and each season contributes its band-and-index bundle to the final GeoTIFF stack.

The compositor then derives a suite of vegetation and soil indices entirely within Earth Engine so that the derived layers inherit the same masking and reprojection as the base bands. The current stack includes NDVI, EVI, EVI2, NDMI, NDRE, GNDVI, OSAVI, NDWI, and BSI. Terrain context is captured by sampling NASADEM elevation, slope, and aspect once and replicating them across the seasonal stack. The result is a consistent, multi-season cube per tile at 10 m resolution.

3.2.3 Tiling and asset management

Islands are subdivided into 0.05° tiles to remain within GEE export limits and to keep per-tile processing times manageable. Exported GeoTIFFs are stored in the `data/phase1/raw/` directory.

Because filenames encode both the island and tile index, scripts can filter tiles for the selected island without needing to inspect file contents. Completed exports are left untouched, allowing reruns to only submit missing tiles, making it feasible to populate the archive sequentially across islands and years.

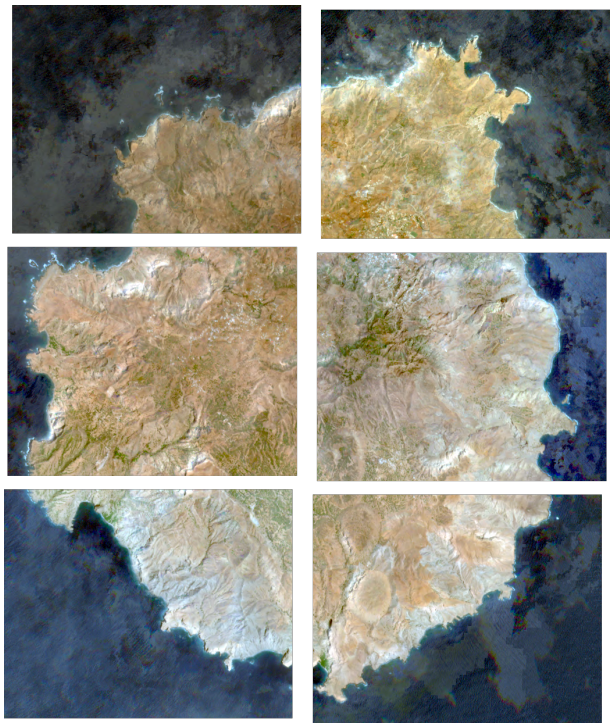


Figure 3.2: Tiling footprint for Brava. Each 0.05° cell corresponds to an Earth Engine export tile, with gaps along the coastline indicating tiles skipped once the on-disk archive already held the required coverage.

3.3 Feature Engineering

3.3.1 Spectral and index stack

During training and inference, each tile is loaded as a multi-band array. The helper in Listing 3.1 selects the configured spectral bands (B2–B8A, B11, B12), attaches the per-season indices, and appends the replicated terrain channels. This ordering is preserved across runs, allowing trained models to be reused without ambiguity.

```
def build_feature_stack(tile_path, config):
cube = load_multiseason_cube(tile_path, config.seasons)
spectral = select_bands(cube, config.training_bands)
indices = compute_indices(cube, config.indices)
terrain = replicate_terrain(config.terrain_layers, cube.shape)
ndvi_textures = compute_ndvi_textures(cube['NDVI'], window=7)
```

```
feature_stack = np.concatenate([
spectral,
indices,
terrain,
ndvi_textures,
], axis=0)

metadata = {
    'bands': config.training_bands,
    'indices': config.indices,
    'terrain_layers': config.terrain_layers,
    'texture_window': 7,
}
return feature_stack.astype(np.float32), metadata

# Called for every tile during inference or candidate scoring
stack, meta = build_feature_stack(tile_path, pipeline_config)
artefact_store.save_features(tile_name, stack, meta)
```

Listing 3.1: Feature stack assembly helper

3.3.2 Texture augmentation

Spatial context is introduced by computing a 7×7 moving-window mean and standard deviation of the seasonal NDVI stack. These statistics help the classifiers distinguish between smooth irrigated terraces and noisy shrublands without resorting to heavy grey-level co-occurrence matrices. Because the textures are derived after masking, cloudy pixels do not contaminate the local statistics.

3.3.3 Caching and validation

Feature stacks are cached on disk to avoid re-deriving indices during every training iteration. The caching layer records the band ordering and texture configuration so that updates to `config.py` invalidate stale caches automatically. Before training begins, three safeguards are applied: (i) the band list is hashed to detect missing exports; (ii) reflectance and index percentiles are checked against predetermined limits (e.g., NDVI within $[-1, 1]$); and (iii) random samples verify that the geotransform in the cache matches the original tile. Any anomaly halts the run for manual inspection, preventing corrupted features from propagating downstream.

3.4 Model Training and Auto-tuning

The pipeline supports three classifiers: a radial-basis SVM, a class-balanced RF, and a stacking ensemble that combines both. All models are calibrated using isotonic regression, allowing downstream thresholding and active-learning heuristics to rely on probabilistic scores. Hyperparameter selection is entirely declarative. The operator can manually set values, choose a curated grid, or activate the auto-tuning mode. Auto-tuning draws candidate combinations from ranges defined in `config.py`, evaluates them using cached feature matrices, and selects the best-scoring configuration according to the weighted-threshold metric computed by `evaluation.py`. The winning parameters are stored alongside the round outputs, ensuring the choice is auditable and easily reused in later rounds.

Runtime diagnostics (training duration, inference throughput, evaluation time) are captured per round and appended to `data/phase1/runtime_metrics.csv`. These measurements feed `plot_round_metrics.py`, which produces longitudinal plots for reports and quality control.

3.5 Active-Learning Operations

Each active-learning round comprises four recurring steps:

1. **Train and calibrate.** Labels from `labels/phase1/labels.csv` and `labels/phase1/temp_labels.csv` are snapped to pixel centres and split into 70% training and 30% validation. Model training and calibration follow the procedure described above. The resulting metrics, plots, and configuration snapshot are written to `round_<n>/statistics/`.
2. **Infer and rank candidates.** Immediately after training, tile-level inference streams probabilities into the round's `_tile_preds/` folder. Candidates are drawn from the uncertainty band whose lower bound is configurable in `config.py`, enriched with a quota of negative-like samples that resemble cropland spectrally but fall just below the decision threshold. Diversity is enforced with a haversine DBSCAN pass (radius configurable per project). Each candidate receives the acquisition score $s = w_U U + w_R R + w_C C$, where the weights for uncertainty, representativeness, and consistency originate from the configuration. The highest-scoring candidates populate both the CSV and KML listings presented to annotators.
3. **Persist round outputs.** The orchestrator saves the refreshed models, probability shards, configuration snapshot, and candidate exports before handing the session to reviewers. This guarantees that every tool sees a consistent snapshot and that an interrupted session can be resumed without recomputing inference.
4. **Label review.** The console interface displays one candidate pixel at a time. Users can accept it as agricultural, accept it as non-agricultural, or skip it. Accepted samples, together

with free-text notes, are appended to `temp_labels.csv`; skipped samples are logged in `labels/phase1/skipped.csv`. Because each candidate KML outlines exactly one pixel, the recorded label always maps to a single coordinate, even if the reviewer inspects the surrounding context in Google Earth or QGIS. At the end of the review cycle the updated annotations are already staged for the next training pass, closing the loop.

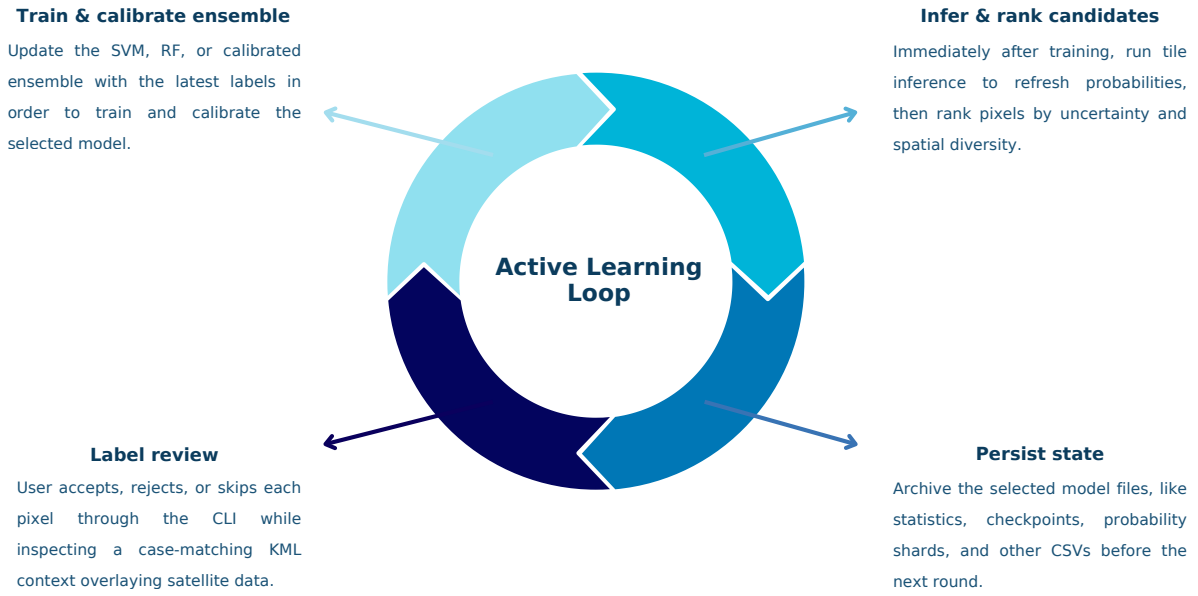


Figure 3.3: Active-learning round workflow. Training updates the calibrated classifiers and records metrics, candidate scoring combines uncertainty with geographic diversity to refresh `predictions.csv` and the Highscore lists, annotators confirm or reject pixels through the CLI and KML overlays, and the round archive stores models, probability shards, and checkpoints for the next iteration.

3.6 Annotation Management and Provenance

Manual annotation remains the most fragile component of the workflow. Every entry in `labels.csv` and `temp_labels.csv` stores the tile identifier, snapped latitude and longitude, and the annotator's note, providing enough context to revisit difficult sites. The acquisition strategy concentrates human effort near the decision boundary, while the diversity filter prevents redundant requests. The skipped-pixel log prevents the repeated serving of the same ambiguous pixel and provides a ready-made queue for quality-control sessions. Optional Highscore and ProbableAgri lists can be regenerated with `refresh_lists.py`; they are disabled by default unless explicitly enabled, but provide an extensible hook for prioritising future reviews.

3.7 Evaluation, Post-processing, and Outputs

Repeated validation reports precision, recall, F1 score, balanced accuracy, Matthews correlation coefficient, and calibration metrics (Brier score, AUC, and AUC-PR). The results, together with per-round runtime totals logged in `data/phase1/runtime_metrics.csv`, feed the longitudinal plots generated by `plot_round_metrics.py`.

Operating thresholds are explicit. The baseline decision threshold is defined in the configuration file. When `AUTO_USE_BEST_THRESHOLD` is enabled, the weighted-best threshold recovered from validation supersedes it, and advisory artefacts are written under `statistics/best_threshold/`. Connected components smaller than the configured sieve are removed, with optional keep-probability rules that either preserve whole components or restrict the output to their highest-confidence pixels. Because per-tile probability shards are cached, subsequent post-processing steps—such as sweeping alternative thresholds or testing morphological filters—reuse the existing inference results without recomputing model outputs.

Hyper-parameter exploration is also declarative. During routine rounds, the operator can select manual values, a curated grid, or the auto-tuning mode. The latter assembles candidate combinations from ranges specified in `config.py`, evaluates them using cached feature matrices, and applies the winner before candidate collection. A final optional grid search reuses the same mechanism and writes its artefacts to `data/phase1/rounds/final_round/`. The outputs include mosaicked KML overlays, classified csGeoTIFFs, and CSV summaries that can be used directly in geographic information systems.

For operational continuity, each round folder forms a self-contained archive, comprising the model pickle, evaluation artefacts, overlays, probability shards, and a configuration snapshot side by side. Dependencies are pinned in `requirements.txt`, and the session log in `HISTORY.md` summarises what each assisted run achieved, easing handovers between annotators.

3.8 Automation and Reproducibility

Automation is a core design goal. The orchestration script records checkpoints, allowing the workflow to resume from the last completed stage if interrupted. Environment checks prevent silent failures due to missing authentication or insufficient disk space. Runtime metrics and configuration snapshots provide a comprehensive audit trail, while the use of version control for configuration files ensures that historical runs can be recreated exactly. Because every artefact is emitted under `data/phase1/rounds/<round>/`, archiving or transferring a round is as simple as copying its folder.

3.9 Change Analysis Outlook

Automated change detection has not yet been implemented within the codebase. Integrating that differencing step into the repository, together with confidence scoring and trend reporting, remains an item of future work.

Chapter 4

Implementation

This chapter details the system architecture and implementation choices that make the pipeline efficient, reproducible, and user-friendly. Emphasis is placed on orchestration, data layout, performance safeguards, and the artefacts produced per round.

4.1 Project Structure

The repository adheres to a script-per-stage layout. The key directories are summarised below:

```
PythonProject/
|-- data/phase1/
|   |-- raw/                # exported Sentinel-2 tiles per season
|   |-- cache/              # optional per-tile feature cache (.npz)
|   |-- rounds/             # round_1, round_2, ..., final_round
|-- labels/phase1/
|   |-- labels.csv          # master labels (id, lat, lon, tile, label, notes)
|   |-- temp_labels.csv     # working copy appended each round
|   |-- highscore.csv       # informative pixels (optional)
|   |-- probableAgri.csv    # high-confidence positives (optional)
|   |-- skipped.csv         # skipped pixels
|-- scripts/
|   |-- ready_to_run_phase1.py   # orchestrator
|   |-- a0_setup_check.py        # environment checks
|   |-- a1_phase1_data_download.py
|   |-- a2_phase1_initial_labeling.py
|   |-- a3_phase1_active_learning_round.py
|   |-- a4_phase1_active_learning_loop.py
|   |-- a6_phase1_postprocessing.py
|   |-- final_grid_search.py
|   |-- generatePersistents.py, refresh_lists.py, recover_highscore.py
|   |-- features.py, evaluation.py, config.py, al_shared.py
|   |-- memory_watcher.py, progress_utils.py
```

Environment-specific files (virtual environments, compiled artefacts) are excluded from version control to keep the repository portable. A ‘requirements.txt‘ file lists Python dependencies, and the recommended practice is to create a per-project virtual environment (‘python3 -m venv .venv‘) before running any scripts. This isolation prevents conflicts between library versions used for experimentation and those required by the operating system.

4.2 Orchestration

The main entry point `ready_to_run_phase1.py` sequences the workflow:

1. runs setup checks (GEE authentication, leftover cleanup, GPU diagnostics if requested);
2. optionally prompts for the island of focus;
3. ensures label files exist and synchronises `labels.csv` with `temp_labels.csv`;
4. steps through active-learning rounds using `a4_phase1_active_learning_loop.py`;
5. performs post-processing and final grid search when the loop ends.

Console output is mirrored to a rolling log called `consoleLogs.txt` inside the `data/phase1` directory. Threading caps for BLAS/OpenMP and GDAL are set early to avoid oversubscription warnings. Cache management routines free temporary arrays after every major stage so long runs remain stable. Auto-tuning history is stored per island in a JSON file alongside the rounds folder, and winning combinations are staged in dedicated subdirectories before inference.

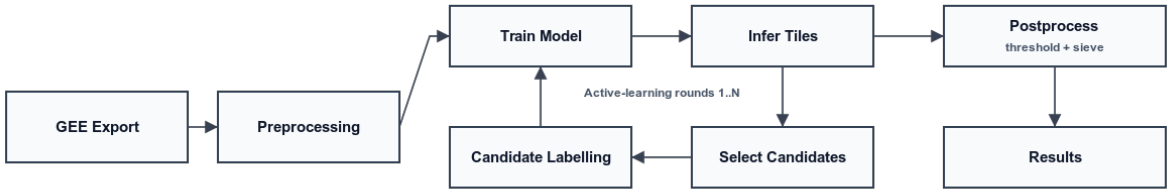


Figure 4.1: High-level architecture of the implementation.

The orchestrator also supports dry-run mode, allowing operators to generate candidate lists without executing heavy inference. This is useful when planning annotation sessions or validating new acquisition parameters before committing to a full round. Command-line prompts provide a concise summary of selected options (island filter, model choice, threshold handling) so run logs remain self-documenting.

4.3 Data Download and Preparation

`a1_phase1_data_download.py` communicates with GEE, sets SSL certificates explicitly for reliability, and walks through the island polygons defined in configuration. An island filter can be specified interactively so the operator can focus on a single island or run the full archipelago. Each export writes one GeoTIFF per tile, embedding spectral bands, vegetation indices, and terrain features in a consistent order, for all seasons. Metadata (temporal windows, band names, CRS, transform) is stored alongside the tile so downstream scripts can verify consistency before feature extraction. Export parameters remain under version control in `scripts/config.py` and are echoed to the console log for traceability.

4.4 Feature Extraction and Caching

The script `features.py` reads raw GeoTIFF tiles, applies the derived-feature logic described in Chapter 3, and returns a three-dimensional array (features, height, width). An in-memory LRU cache keeps the latest tiles to accelerate repeated lookups during candidate scoring. Optionally, features are persisted to `data/phase1/cache/.npz`; these files include the raster transform and CRS so they can be reloaded safely.

During inference the feature builder honours the island filter and keeps only tiles matching the active campaign. When textures are enabled the code falls back to memory-mapped NumPy arrays to avoid loading entire stacks simultaneously, and summary statistics (min, max) are saved so min-max scaling is reproducible between rounds. Cached `.npz` bundles compress per-tile arrays and store a checksum; the orchestrator validates those sums before reusing cached data.

4.4.1 Operational considerations

GPU support is optional and currently used only for diagnostics. Configuration flags allow operators to adjust tile-processing concurrency or inference batch sizes, ensuring the pipeline can adapt to different installation contexts without code changes.

4.5 Training Rounds

The active-learning round orchestrator performs the following tasks:

- splits labels into train/validation folds (respecting island filters);
- trains the selected calibrated model (SVM, RF, or the stacking ensemble) and applies isotonic calibration whenever fold counts allow;

- records repeated-validation metrics and plots (confusion matrices, ROC/PR, threshold sweep, calibration curve, runtime breakdown);
- evaluates small hyperparameter batches in-memory before running full-tile inference only for the winning combination;
- infers per-tile probabilities in configurable batches, emitting temporary CSV shards (with optional binary sidecars) that are merged and deleted after downstream steps unless the user decides to keep them;
- polygonises high-confidence components into single-style KML files using Rasterio and Shapely;
- computes candidate scores and exports CSV/KML for annotators.

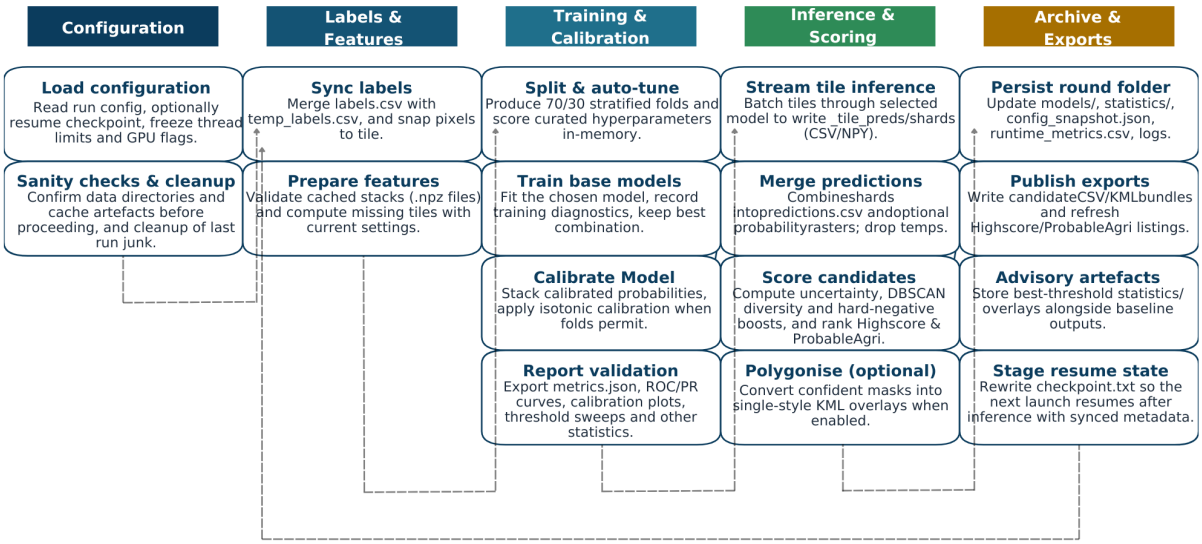


Figure 4.2: Phase-1 training round flow of configuration, inference, and artefact persistence. Blue nodes cover configuration loading, label preparation, and calibrated model fitting. Green nodes summarise tile inference, shard merging, and candidate scoring. Amber nodes capture the outputs written to `statistics/`, `models/`, the consolidated `predictions.csv`, and the candidate exports consumed during the next review session.

Auto-tuning evaluates a small set of hyperparameter combinations before training, caching scores in the auto-testing workspace along with diagnostic plots.

Temporary artefacts live under each round folder. For example, inference shards are written to short-lived CSV and NumPy files, merged once, and removed automatically so subsequent runs start with a clean slate. Figure exports sit in the statistics subfolder.

4.6 Active Learning Support

Candidate selection relies on shared utility functions that handle feature extraction at label coordinates, pixel snapping, deduplication, and a DBSCAN-based spatial diversity filter. The initial labelling helper provides quick exports of the current label set to KML and maintains a list of skipped pixels for bookkeeping and avoid future labeling inefficiency.

The candidate export blends three sources—uncertain pixels near the decision threshold, deliberately selected hard negatives, and a spatial buffer that enforces diversity. Each entry carries flags so annotators can see which strategy nominated it, but everything is delivered in the same CSV/KML bundle. When AUTO_USE_BEST_THRESHOLD is enabled the lower probability bound is recomputed after every round, keeping the candidate floor aligned with the latest calibrated operating point instead of freezing it at a static value.

Listing 4.1 summarises the scoring routine that underpins this prioritisation.

```
def score_candidate(probability, ndvi, diversity_weight,
   weights):
    """Combine uncertainty, representativeness, and consistency
   signals."""
    uncertainty = 1.0 - abs(probability - config.MIN_AGRI_PROB)
    representativeness = diversity_weight
    consistency = 0.0 if ndvi is None else 1.0 - abs(ndvi -
   ndvi_anchor)
    return (
        weights['uncertainty'] * uncertainty
        + weights['representativeness'] * representativeness
        + weights['consistency'] * consistency
    )

def candidate_score(row, weights):
    probability = row['prob']
    ndvi = row.get('ndvi')
    diversity_weight = row['diversity_weight']
    base_score = score_candidate(probability, ndvi,
   diversity_weight, weights)
    if row.get('hard_negative'):
        base_score += weights['hard_negative_bonus']
    return base_score

# Weights are configured once per round and logged for reproducibility.
weights = {
```

```
    'uncertainty': 0.5,
    'representativeness': 0.3,
    'consistency': 0.2,
    'hard_negative_bonus': 0.05,
}

row['score'] = candidate_score(row, weights)
```

Listing 4.1: Candidate scoring helper

4.7 Label Management

Labels are split across two layers: `labels.csv` holds the trusted baseline, while `temp_labels.csv` stores the annotations gathered during the current run. When the orchestrator starts it reads both files, aligns their columns, and builds the in-memory label set from the union—`labels.csv` itself is never overwritten. `temp_labels.csv` is reset at the beginning of a new run so upcoming annotations remain isolated from previous campaigns. Every row retains the tile identifier, snapped latitude/longitude, and the annotator's note, preserving provenance for later audits.

Candidate exports (CSV plus KML) include both the selected pixels and the latest overlays so annotators can inspect context directly in Google Earth. Skipped pixels are archived in `skipped.csv` rather than discarded, and the interactive loop avoids re-serving them unless the analyst explicitly chooses to revisit that queue.

Because manual labelling is error-prone, the pipeline emphasises traceability. Round folders keep their temporary candidate files, merge artefacts, and configuration snapshots, allowing any pixel to be audited or corrected without touching `labels.csv`. This separation between the canonical labels and the run-specific annotations is crucial when teams iterate quickly but still need confidence in the official dataset [17, 18].

4.8 Documentation and Knowledge Sharing

Documentation lives alongside the code. The top-level `README.md` explains the repository layout, the entry-point scripts, required dependencies, and the order in which to execute the pipeline. Each script carries inline comments describing key functions and command-line options, while the round folders retain logs and configuration snapshots that can be consulted when retracing decisions. This lightweight documentation could be extended with additional usage notes, but the existing `README.md` and per-round artefacts already provide the guidance needed to reproduce or adapt a run.

4.9 Persistent Lists

The persistent-list generator rebuilds the Highscore (uncertain yet representative candidates) and ProbableAgri (high-confidence positives) tables from past rounds. Streamed merges with external sorting ensure the process scales to millions of rows without straining resources. Outputs include CSV tables and capped KML files. A lightweight recovery tool can recreate these lists from backups when needed, and the logs report how many training rows were used for representativeness so analysts know whether the refresh drew on enough labelled data.

Each record stores metadata such as the round that surfaced the pixel, its probability percentile, and whether the representativeness component was active. That context helps prioritise future surveys and verify that newly added labels cover a diverse set of environments.

4.10 Postprocessing and Final Grid Search

The post-processing stage loads the final calibrated model, classifies every tile, applies the configured threshold and sieve, and summarises coverage per tile. Optional sweeps analyse combinations of thresholds and sieve sizes while reusing cached per-tile probabilities instead of rerunning inference for each combo. The optional final grid-search routine evaluates curated hyperparameter sets for the chosen model and reruns inference just once for the selected combination, storing summaries in the final-round directory.

Listing 4.2 captures the high-level logic of the postprocessing sweep. It shows how cached probability rasters are reused, how threshold/sieve combinations are iterated, and where summary artifacts are written.

```
def run_postprocessing(probability_store, config):
    results = []
    for threshold in config.final_thresholds:
        for sieve_size in config.final_sieve_sizes:
            raster = probability_store.load_cached()
            mask = raster >= threshold
            cleaned = apply_sieve(mask, sieve_size)
            polygons = polygonise(cleaned, min_vertices=4)

            artefacts.write_mask(threshold, sieve_size, cleaned)
            artefacts.write_polygons(threshold, sieve_size,
   polygons)

            metrics = summarise(polygons, config.tiles)
            results.append({
```

```
                'threshold': threshold,
                'sieve': sieve_size,
                'metrics': metrics,
            })

    artefacts.write_summary(results)
    return results

# Final sweep reuses cached probabilities and writes consolidated outputs
summary = run_postprocessing(probability_store, pipeline_config)
```

Listing 4.2: Post-processing sweep overview

Intermediate outputs include per-tile CSV summaries and optional GeoTIFF probability rasters for GIS inspection. When polygonisation runs, tiles are processed sequentially by default on Windows and can be parallelised on Linux by increasing `POLYGONIZE_WORKERS`.

4.11 Reproducibility and Logging

Each active-learning round leaves a trail of artefacts that can be replayed later:

- **Configuration snapshot.** After evaluation the orchestrator copies every active setting (for example `MIN_AGRI_PROB`, `SIEVE_MIN_SIZE`, and model hyperparameters) into the relevant statistics folder, such as `statistics/ensemble/config_snapshot.json`. Resume mode in `ready_to_run_phase1.py` reads this file to continue an interrupted round with the same parameters.
- **Runtime accounting.** Per-round timings land in `statistics/runtime.json` and the breakdown plot `statistics/runtime_breakdown.png`. The helper `data/phase1/runtime_metrics.csv` accumulates these numbers across rounds and feeds the summary chart `data/phase1/runtime_summary.png`.
- **Model diagnostics.** Every trained model folder (`statistics/svm`, `statistics/rf`, or `statistics/ensemble`) stores the exported classifier, ROC/PR plots, confusion matrices, the repeated-validation dump `metrics_repeated.json`, and the operating-threshold KML. Auto-tuning outcomes append to the history JSON kept under `data/phase1/`, while `data/phase1/consoleLogs.txt` preserves the Rich console transcript.

A condensed console transcript illustrating these checkpoints is provided in Appendix 6.4 (Figure 2).

Listing 4.3 illustrates the configuration snapshot written at the end of each round. The JSON record captures model choices, threshold settings, candidate-selection parameters, and the paths

of the generated artefacts, making it straightforward to audit or recreate any run.


```
{
   "timestamp": "2025-09-21T11:42:03Z",
   "selected_island": "sao_nicolau",
   "model": {
      "type": "SVM",
      "params": {
         "C": 3.0,
         "gamma": "scale",
         "class_weight": "balanced"
      }
   },
   "threshold": {
      "min_agri_prob": 0.35,
      "auto_use_best": true,
      "best_round_value": 0.37
   },
   "candidate_selection": {
      "batch_size": 50,
      "uncertainty_floor": 0.28,
      "hard_negative_quota": 0.40
   },
   "artefacts": {
      "round_dir": "data/phase1/rounds/round_1/",
      "predictions": "predictions.csv",
      "metrics": "statistics/metrics.json"
   }
}
```


Listing 4.3: Configuration snapshot excerpt

Taken together these files let another practitioner rebuild the environment, check exactly which options were in force, and justify the reported metrics without rerunning the entire campaign.

4.12 Testing the pipeline

Current safeguards favour manual inspection over automated tests:

- **Environment sanity checks.** `scripts/a0_setup_check.py` resets stale outputs.

- **Round-level evidence.** Each model write-out publishes `metrics.json`, `metrics_summary.csv`, `metrics_repeated.json`, and the probability histograms, which together highlight shifts in precision, recall, and calibration whenever dependencies change.
- **Cross-round tracking.** The helper `data/phase1/rounds/rounds_metrics.csv` and its paired plot store F1 and AUC history so regressions show up as trend breaks.

Verifying refactors still depends on running a short round and checking the generated artefacts.

4.13 Preparing for Change Analysis

The pipeline's remit is to curate trustworthy artefacts, not to declare how agriculture has shifted year on year. Its job is to give analysts everything they need for that judgement:

- Round folders keep the per-pixel `predictions.csv` files, KML exports, and calibrated metrics so a practitioner can line up the same island or tile without rerunning training.
- `scripts/refresh_lists.py` rebuilds the persistent Highscore and ProbableAgri tables from any saved `predictions.csv`, letting us surface fresh labelling leads whenever a change study demands more context.
- When we run `scripts/a6_phase1_postprocessing.py`, the optional final sweep regenerates GeoTIFF overlays at chosen thresholds, giving map-ready layers that can be stacked and differenced in desktop GIS tools.

Appendix 6.4 (Figure 1) details the directory layout of a typical round archive for readers who plan to automate change-analysis workflows on top of these artefacts. Because the project stops short of automatic change detection, there is no module that counts transitions, assigns confidence bands, or narrates multi-year trends. Those insights will instead be drawn in Chapter 5, where we will assemble the saved statistics, plots, and overlays from 2019 and 2023 and interpret the patterns.

4.14 Runtime Optimisation and Resource Efficiency

Delivering fresh predictions to annotators hinges on keeping the active-learning loop responsive. Every optimisation introduced in the codebase targets a specific bottleneck so that the pipeline scales across multiple islands and seasons without exploding in runtime or memory demand. The discussion below follows the flow of `scripts/ready_to_run_phase1.py`, highlighting how each improvement cuts out redundant work. Table 4.1 summarises the main levers in plain terms.

4.14.1 I/O and Feature Reuse

At first, each round rebuilt the full feature stack for every tile, so effort grew with the number of rounds, tiles, and bands. Placing derived stacks under `data/phase1/cache` means unchanged tiles are simply reused: only new or invalidated tiles trigger fresh exports. The same principle drives probability shard reuse in `_tile_preds/`. Post-processing now loads cached shards once and sweeps thresholds over them, so the work is "number of tiles plus number of thresholds" instead of "tiles times thresholds". This keeps artefact refreshes short enough to run between labelling sessions.

4.14.2 Model-Selection Efficiency

Brute-force hyperparameter search grew with the size of the full grid (for example three values of `C` times three kernels times three class-weight settings). The orchestrator now evaluates only a curated shortlist, so the number of training runs is proportional to the shortlist length. CalibratedClassifierCV also replaces separate calibration passes: the RandomForest is trained once and internally calibrated, keeping model selection effort in step with the handful of combinations analysts actually want to compare.

4.14.3 Controlled Parallelism and Memory Safety

Unrestricted parallelism on Windows caused multiple Rasterio workers to compete for memory, occasionally forcing a restart. The default now runs polygonisation sequentially on Windows and only parallelises when the host can guarantee enough RAM. Candidate clustering applies DBSCAN on geographic chunks, so the work grows with chunk size rather than the square of the full candidate list. Thread caps on BLAS/OpenMP libraries keep CPU-bound stages from oversubscribing cores, preventing watchdog resets and ensuring predictable latency.

4.14.4 Lean Imports and Streaming Merges

Heavy libraries such as `sklearn` and `torch` are now imported lazily inside the functions that need them. Worker processes therefore start faster and consume less memory because unused back-ends never load. Persistent-list utilities apply the same principle: instead of rewriting Highscore or ProbableAgri from scratch, they stream new shards and merge them with the existing CSV/KML records.

4.14.5 Artefact Persistence and Resume Points

Checkpointing the pipeline replaces full reruns with incremental resumes. If a run stops after inference, restarting skips straight to post-processing because the merged predictions, configuration snapshot, and metrics are already on disk. The amount of work redone therefore depends only on the unfinished suffix, so completed stages are never repeated.

Table 4.1: Summary of key optimisation levers and their impact.

Optimisation lever	Baseline workload	Work after optimisation	Analyst-facing impact
Feature-stack caching and probability shard reuse	Recomputed every tile for every round and threshold	Recompute only tiles that changed; sweep thresholds over cached shards	Enables quick artefact refreshes without rerunning exports.
Curated hyperparameter search and calibrated ensembles	Trained all combinations in the grid	Train only shortlisted combinations, calibrating once per model	Keeps validation effort proportional to the few models needed.
Controlled parallelism and chunked DBSCAN	Parallel jobs competed for RAM; clustering processed all candidates at once	Parallelism matched to available memory; clustering runs on manageable chunks	Avoids resource exhaustion, preserving predictable throughput.
Lean imports and streamed merges	Every worker loaded heavy libraries; persistent lists rewrote entire files	Load libraries on demand and append new shards to existing artefacts	Cuts start-up overhead and keeps merges fast even as archives grow.
Persistent artefacts and resume checkpoints	Full pipeline replay after an interruption	Resume from the last unfinished stage using saved outputs	Protects completed work and shortens recovery time.

4.14.6 Quality Preservation Under Time Constraints

The optimisation strategy does not simply shorten runtime; it safeguards accuracy by keeping each round tightly coupled to the most recent labels and calibration data. Because the pipeline can refresh predictions using cached artefacts, analysts can evaluate additional timestamps within the same seasonal window, which is essential when phenological signals change rapidly. Likewise, the curated search ensures that the best-performing models remain within reach even when analysts request multiple what-if scenarios in a single day.

4.14.7 Analyst Workflow Integration

From the analyst perspective, optimisations eliminate waiting periods between labelling bursts. Persistent artefacts guarantee that the Highscore and ProbableAgri lists stay synchronised with the latest model state, while checkpoints keep the CLI responsive if a session is paused mid-round. The net effect is a shorter loop between data inspection, label confirmation, and model retraining, enabling cross-island comparisons and additional temporal slices without compromising output quality.

Collectively, these optimisations compress the resource footprint of each active-learning iteration, allowing more seasons or islands to be analysed before agricultural conditions shift.

Chapter 5

Results and Discussion

This chapter explains the 2019 cropland maps produced by the automatic ensemble introduced in Chapter 3. The pipeline picks the model settings, the decision threshold, and the sieve size on its own by searching for the best validation score on the labelled pixels. Because each island only has a small batch of confirmed points, we treat the numeric scores as hints and focus on what the maps look like.

5.1 How the evaluation works

We train on 70% of the labelled pixels and validate on the remaining 30%. This split is repeated ten times so we can average the score and reduce luck. After every active-learning round, the system recalculates the uncertainty band, the negative sampling range, and the NDVI filter using the latest probability histogram, so the suggested labels get sharper as the model learns.

5.2 Validation snapshot

Table 5.1 lists the main validation scores for each island. Precision tells us how many flagged cropland pixels were right, recall tells us how many true cropland pixels we recovered, and the Brier score measures how well the probabilities match reality (smaller is better). The last column shows the operating threshold chosen automatically for that round. Because each island only offers a few dozen validation pixels, we use these scores to spot trends rather than to declare winners.

Thresholds between 0.05 and 0.55 mean the model needs at least a 5%–55% confidence before painting a pixel as cropland. Low thresholds, such as Sal's 0.05, allow the classifier to surface faint signals so we do not miss the tiny greenhouse plots. High thresholds, like Santiago's 0.49, force the model to be picky because cropland is common and the cost of false alarms is

larger than usual. Thinking in concrete terms: on Boa Vista the precision-recall pair (0.82/0.92) means that out of every 100 highlighted pixels roughly 82 are real farms, and the map recovers about 92 out of every 100 true cropland pixels.

The Brier score helps us judge the probability scale. A score near 0.03 (Sal) means predictions such as 0.10 or 0.80 usually match the real outcomes; values above 0.08 (São Vicente) show that the confidence values still wobble. This matters when the maps feed risk rules, because a farmer can trust probabilities around 0.2 much more on islands with low Brier. Table 5.2 tracks how the Brier score moved between round 1 and round 4. Most islands improved by a small margin, while São Vicente got slightly worse because green city parks still look a lot like farms in the satellite view.

Table 5.1: Ensemble retest metrics for the auto-tuned configuration on each island (2019). Thresholds stem from the auto-threshold sweep; Brier scores capture calibration quality.

Island	Precision	Recall	F_1	Threshold	Bal. Acc.	AUC-PR	Brier
Boa Vista	0.82	0.92	0.87	0.17	0.93	0.91	0.079
Brava	0.87	0.89	0.88	0.47	0.91	0.96	0.064
Fogo	0.34	0.76	0.47	0.47	0.77	0.55	0.088
Maio	0.86	0.16	0.27	0.32	0.58	0.61	0.071
Sal	0.00	0.00	0.00	0.05	0.50	1.00	0.026
Santiago	0.68	0.89	0.77	0.49	0.88	0.89	0.081
Santo Antão	0.98	0.98	0.98	0.45	0.98	0.99	0.048
São Nicolau	1.00	1.00	1.00	0.39	1.00	1.00	0.029
São Vicente	0.88	0.92	0.90	0.17	0.89	0.99	0.099

Table 5.2: Brier score progression (rounds 1 vs. 4).

Island	Round 1	Round 4	Δ
Boa Vista	0.099	0.079	-0.020
Brava	0.069	0.064	-0.005
Fogo	0.094	0.088	-0.005
Maio	0.076	0.071	-0.005
Sal	0.026	0.026	$+0.000$
Santiago	0.080	0.081	$+0.001$
Santo Antão	0.056	0.048	-0.008
São Nicolau	0.033	0.029	-0.003
São Vicente	0.076	0.099	$+0.024$

To understand how these metrics translate to absolute area, Table 5.3 combines the cropland share with the number of image pixels covering each island. Cropland pixels take the predicted cropland (the more developed results), apply each round's auto-selected threshold (the probability cut-off picked during tuning), average the two totals, and then round to the closest pixel. The resulting pixel counts provide a first-order estimate of the mapped agricultural footprint.

Table 5.3: Estimated cropland footprint per island (rounds 3–4 share projected over total island pixels).

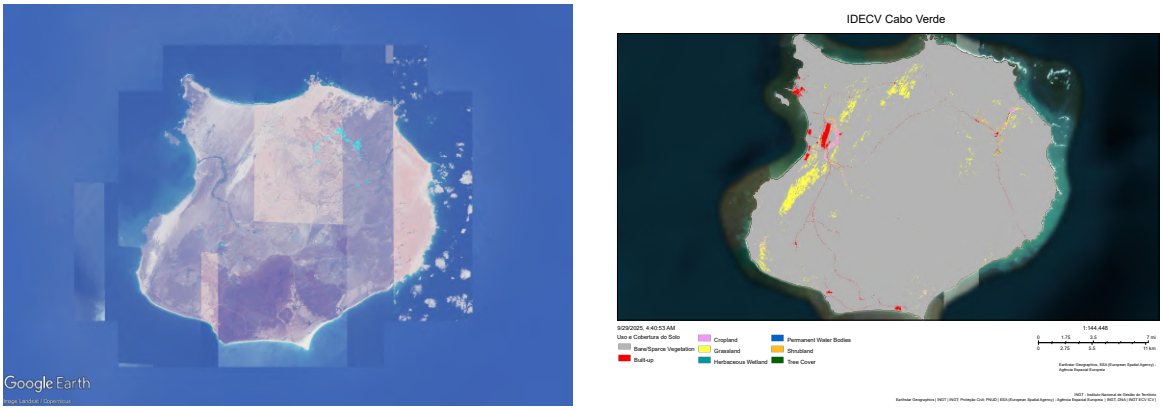
Island	Total pixels	Cropland pixels	Cropland (%)
Boa Vista	9,408,811	4,195	0.04
Brava	1,169,857	133	0.01
Fogo	6,938,795	2,083	0.03
Maio	4,323,171	347	0.01
Sal	4,058,905	4,887	0.12
Santiago	14,260,564	41,197	0.29
Santo Antão	11,946,679	184,865	1.55
São Nicolau	6,869,411	8,974	0.13
São Vicente	3,684,658	31,095	0.84

5.3 Island comparisons

We now compare the resulting maps with the Cape Verde Government land-cover layer "Uso e Cobertura do Solo (ESA 2020)". While they are not from the same year, they are from close timeframes where little to no change will take place in terms of land use.

5.3.1 Boa Vista

Boa Vista holds only about 0.04% cropland (Table 5.3). Figure 5.1 shows how the model limits cropland to the irrigated strips near Rabil while the government layer still paints several dune bands as farms on the island left side, which overstates the planted area.

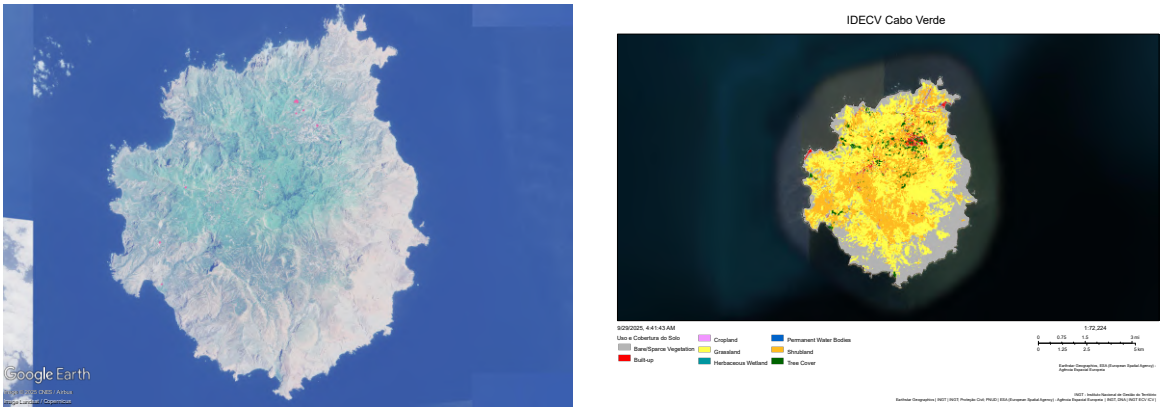


2019 ESA 2020 map

Figure 5.1: Boa Vista comparison between the pipeline prediction and the government land-cover product.

5.3.2 Brava

Brava contains steep terraces and only about 0.01% mapped cropland. In Figure 5.2 the round 4 map picks the narrow terrace belts in the south-west, while the ESA layer spreads cropland across the ridges and even onto bare rock.

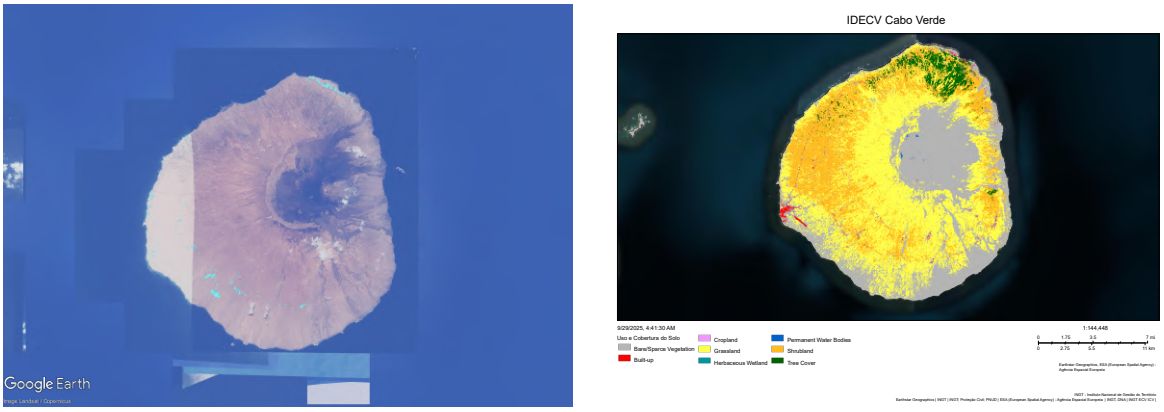


2019 ESA 2020 map

Figure 5.2: Brava comparison between the pipeline prediction and the government land-cover product.

5.3.3 Fogo

Fogo shows stronger cropland pressure (about 0.03%). Figure 5.3 highlights how the model lines up with the cultivated fields along the caldera rim, whereas the ESA map keeps large cropland patches inside recent lava flows where farming is no longer active.

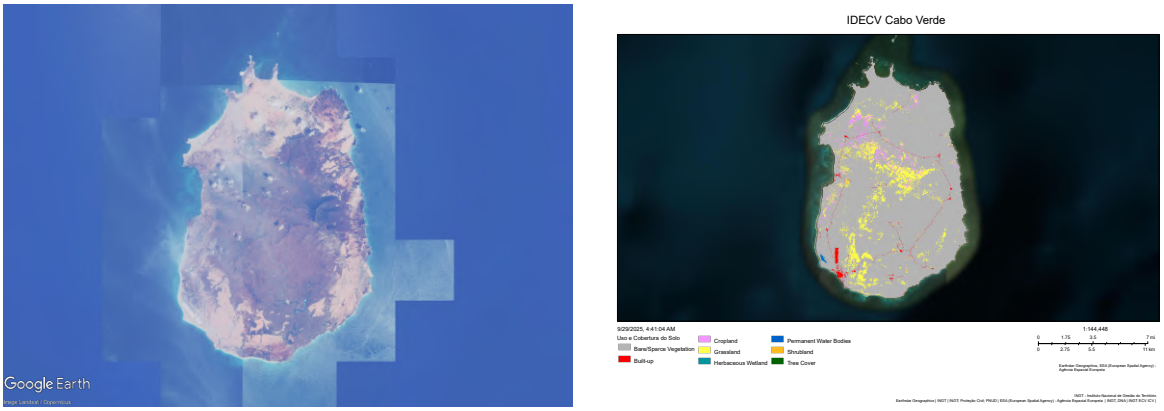


2019 ESA 2020 map

Figure 5.3: Fogo comparison between the pipeline prediction and the government land-cover product.

5.3.4 Maio

Maio retains thin farming belts along the central wetlands, shown on figure 5.4, which add up to roughly 0.01% of the island, probably false positives since this island had no real agricultural development. The official map opposes that conclusion, and extends a "fictional" cropland into the northern dry flats.

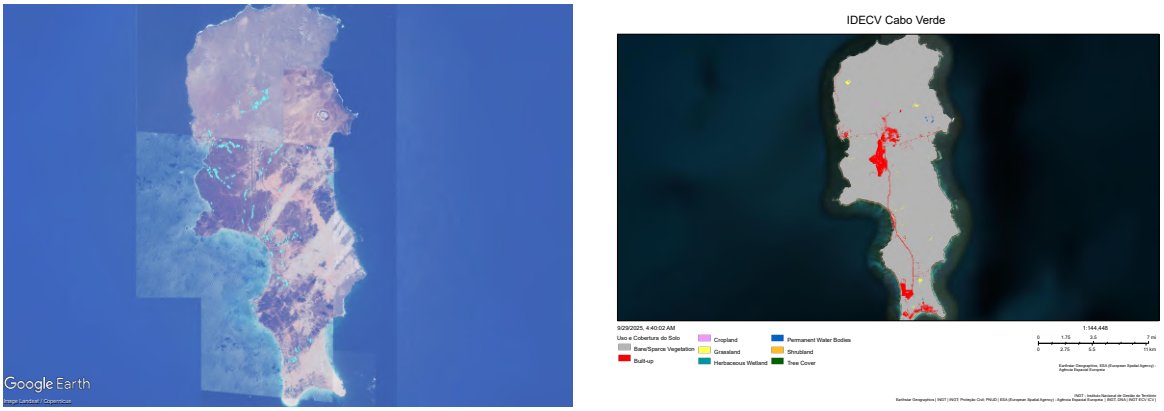


2019 ESA 2020 map

Figure 5.4: Maio comparison between the pipeline prediction and the government land-cover product.

5.3.5 Sal

Sal registers a modest 0.12% cropland slice. As seen in Figure 5.5, the model lights up the greenhouses around Espargos and the airport, while the ESA map still tags some non-agricultural vegetation polygons as cultivated land.

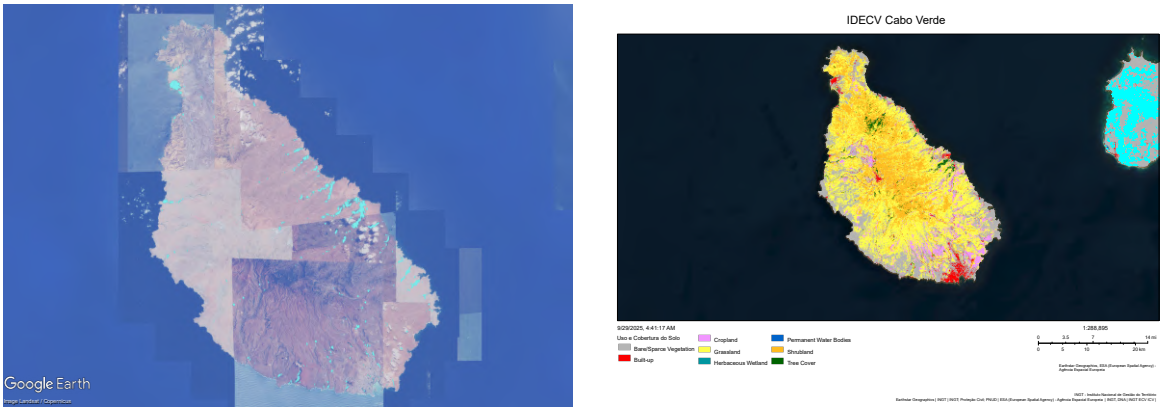


2019 ESA 2020 map

Figure 5.5: Sal comparison between the pipeline prediction and the government land-cover product.

5.3.6 Santiago

Santiago concentrates about 0.29% of cropland, mainly in the eastern valleys. Figure 5.6 shows the resulting map tracking those irrigated basins and leaving the high ridges blank, whereas the ESA layer paints wide hillside bands and even some coastal zones as farms.

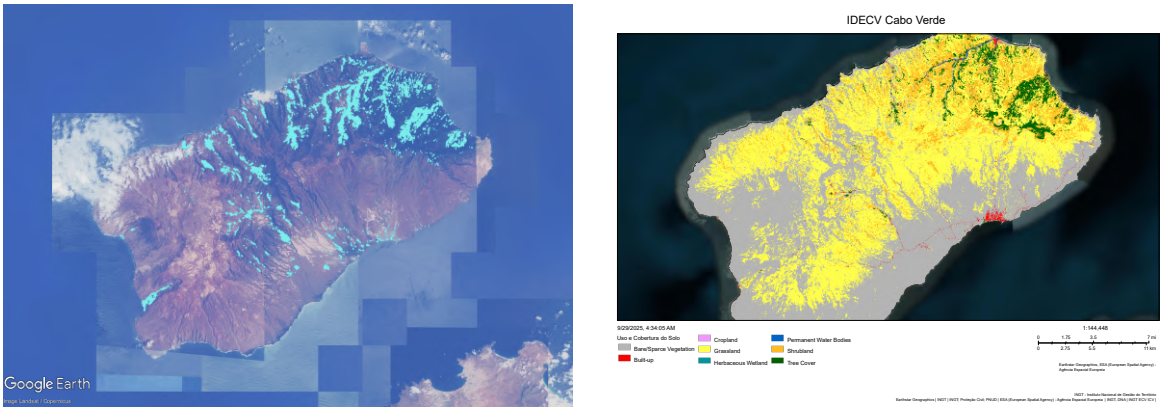


2019 ESA 2020 map

Figure 5.6: Santiago comparison between the pipeline prediction and the government land-cover product.

5.3.7 Santo Antão

Santo Antão is the most agricultural island in this study with about 1.55% cropland. Figure 5.7 highlights how the model follows the mountain terraces and avoids the forest canopy, while the ESA layer still colours most of the island as either grassland or shrubland.

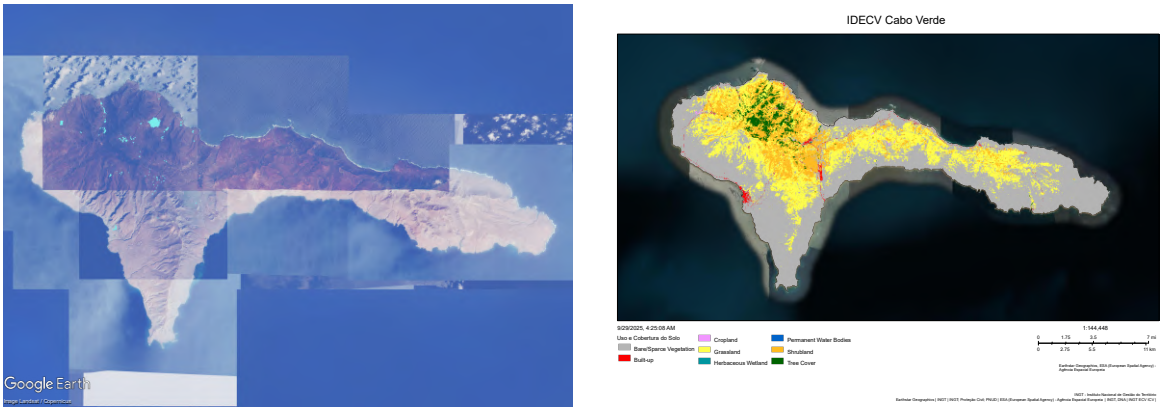


2019 ESA 2020 map

Figure 5.7: Santo Antão comparison between the pipeline prediction and the government land-cover product.

5.3.8 São Nicolau

São Nicolau carries about 0.13% cropland concentrated on the northern slopes. In Figure 5.8 the model stays close to those slopes, while the ESA product spreads cropland across the interior highlands, even failing to identify two of the major cropland centers of this island.

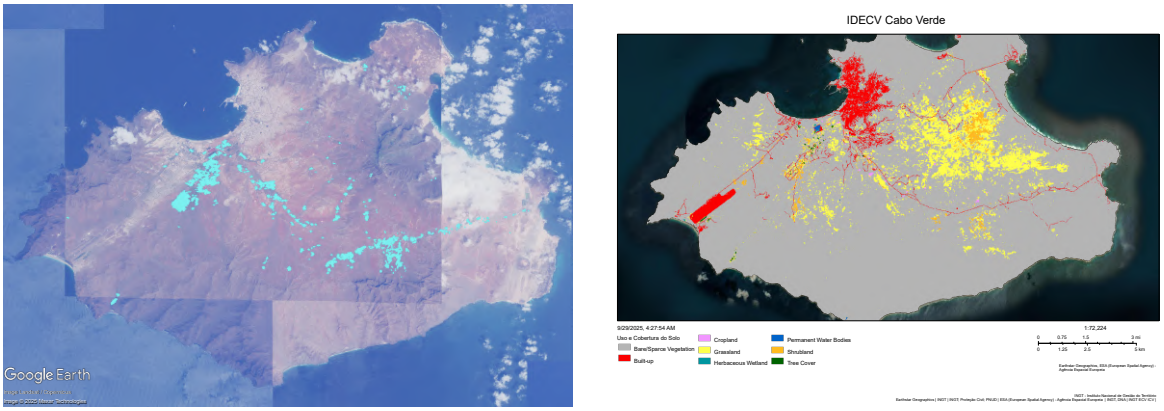


2019 ESA 2020 map

Figure 5.8: São Nicolau comparison between the pipeline prediction and the government land-cover product.

5.3.9 São Vicente

São Vicente reaches roughly 0.84% cropland. Figure 5.9 shows the model mapping the drip-irrigated farms around Ribeira de Vinha and keeping Mindelo's parks as non-cropland, but with a slight problem of false positives. The ESA layer marks many urban green spaces as agriculture, and fails to identify a big portion of the agricultural crops.



2019 ESA 2020 map

Figure 5.9: São Vicente comparison between the pipeline prediction and the government land-cover product.

5.4 High-confidence examples

Figure 5.10 collects four zoomed-in scenes where the model cleanly separates crop fields from the surrounding land and vegetation. These examples cover plateaus, valley bottoms, and dry coastal plots, showing that the ensemble adapts to very different terrains and textures.

Figure 5.10: Clear cropland boundaries produced by the pipeline across several islands.

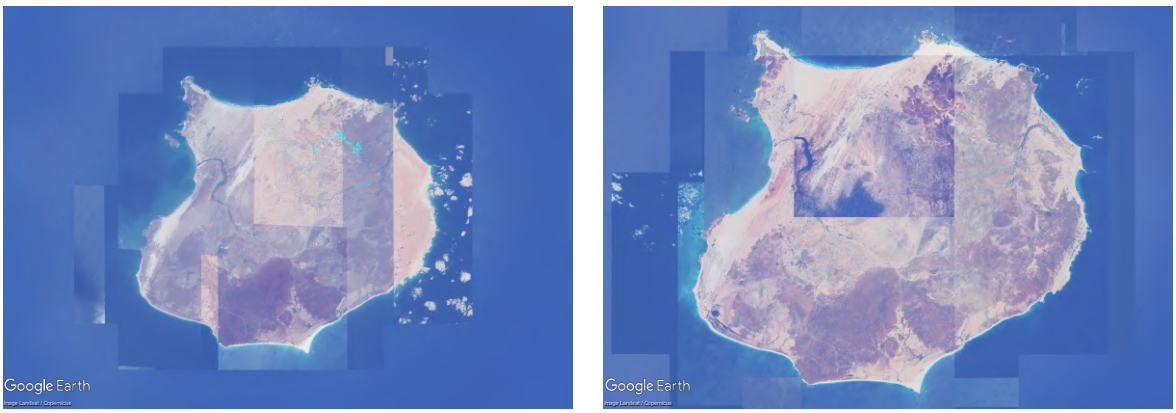
5.5 Change signals between 2019 and 2023

I reran the full pipeline on the 2023 Sentinel-2 composites with the exact same configuration used for 2019. This keeps the comparison honest: any shift we observe stems from the imagery, the seasonal timing, or the way the classifier recalibrates its confidence, not from new hyperparameters. Every artefact we inspect here—probability rasters, threshold sweeps, round summaries— mirrors the 2019 exports.

Before diving into the island snapshots I sanity-checked the round-metric trends and calibration plots, confirming that the auto-selected thresholds stay close to the 2019 levels and that precision still leads the cut. The richer 2023 composites also include more cloud-free slots, so marginal drip systems appear clearer; where imagery remains noisy, the repeated-fold metrics provide the guardrail. With that context in place, the maps that follow show how the same

workflow reacts to four extra growing seasons.

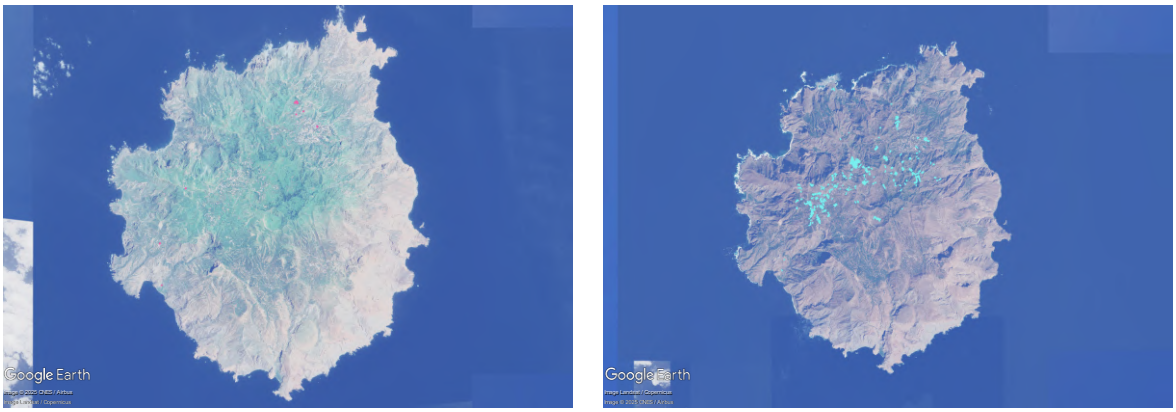
Boa Vista keeps only a small bright patch around the irrigated belt near Rabil in Figure 5.11. Table 5.4 shows the predicted cropland shrinking from 4 195 pixels in 2019 to 815 in 2023. The drop sounds harsh, but the upside is that the dunes and tourism strips no longer trigger false alarms; the classifier, in the 2023 data, sticks to the clearly irrigated pivots.



2019 2023

Figure 5.11: Boa Vista comparison between the pipeline predictions for 2019 and 2023.

Brava tells an opposite story. The 2019 baseline only caught 133 pixels (Table 5.4), likely because clouds masked the narrow terraces and the conservative threshold filtered most of them out. The 2023 map in Figure 5.12 lights up 9 334 pixels along the steep valleys; some hillside vegetation probably slips through as false positives, yet the majority of the new highlights match the terrace pattern we expect. Even with that caution, the ensemble now recognises the cropped ledges while still leaving the barren ridge tops untouched, which gives us a much better starting point for manual review.

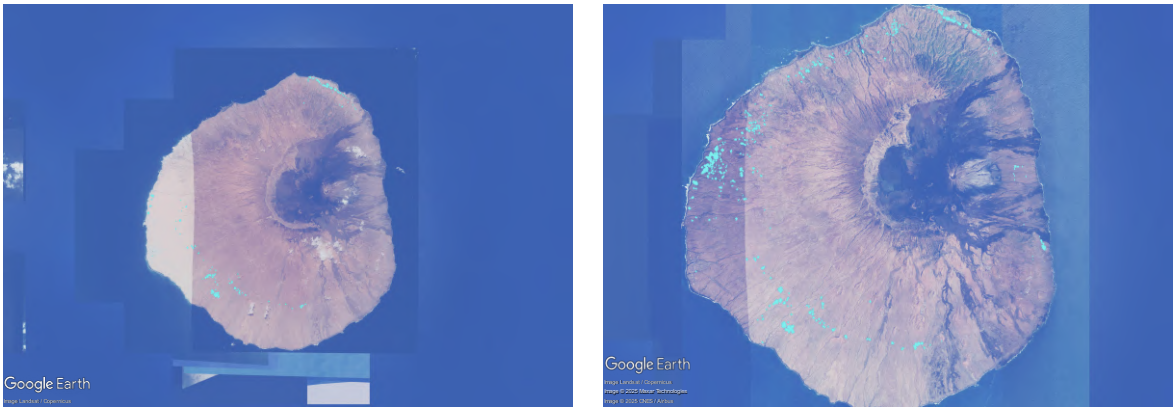


2019 2023

Figure 5.12: Brava comparison between the pipeline predictions for 2019 and 2023.

Fogo now expands from 2 083 pixels in 2019 to 6 589 in 2023. Figure 5.13 shows the caldera floor brightening alongside the western flank. The threefold increase matches what we see

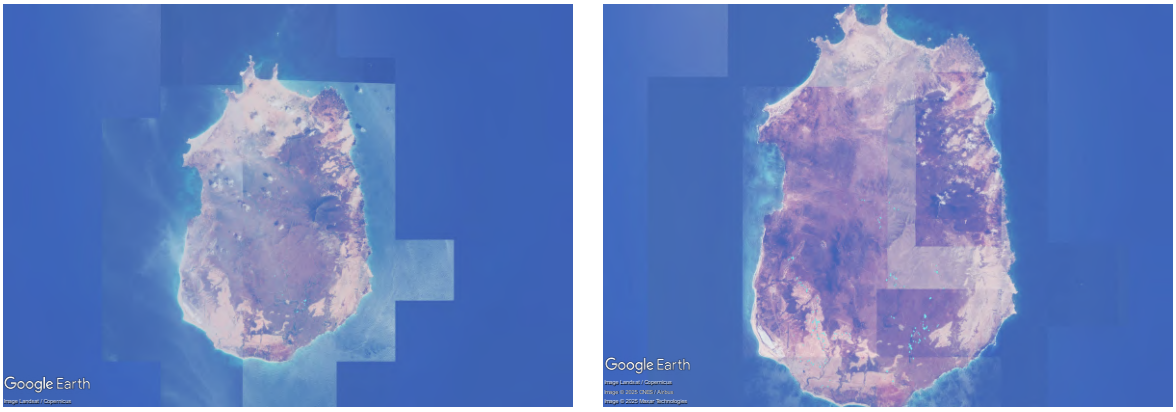
in the overlay, but a few hillside shrubs now slip through as false positives, so the 2019 mask remains the more conservative baseline. A practical next step is to seed just a handful of extra negatives along the caldera rim so the 2023 run keeps the real terraces and drops the ornamental vegetation.



2019 2023

Figure 5.13: Fogo comparison between the pipeline predictions for 2019 and 2023.

Maio grows the most in relative terms. The pipeline jumps from 347 pixels in 2019 to 10 548 in 2023 (Table 5.4). The 2019 map was clearly too conservative—Figure 5.14 shows the same coastal drip-irrigated strips already present but below threshold—while the 2023 composite captures greener months or simply benefited from cleaner cloud masks. This gives us confidence to revisit the old selection rules and relax them slightly so future reruns do not under-report the drip systems.

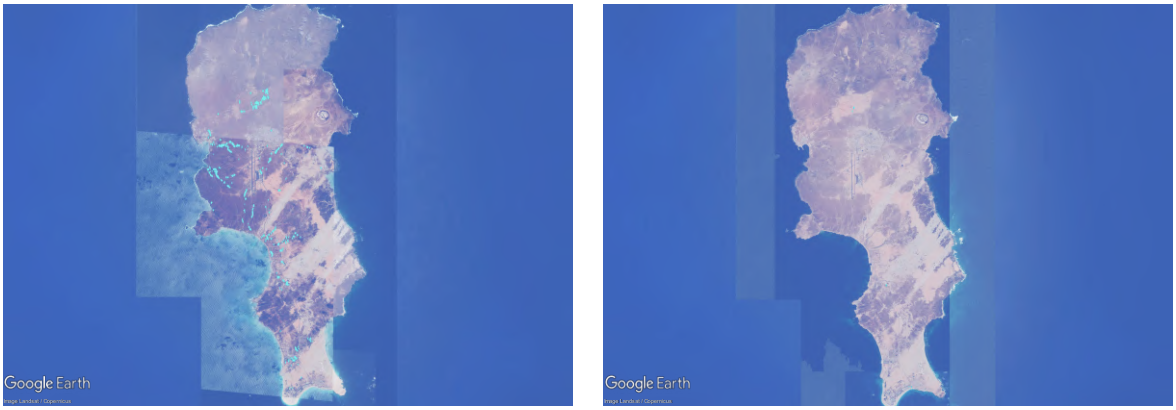


2019 2023

Figure 5.14: Maio comparison between the pipeline predictions for 2019 and 2023.

Sal swings the other way: the 2023 run flags zero cropland pixels, so Figure 5.15 shows a blank map. Table 5.4 reported 4 887 pixels in 2019, and most of that footprint was false positives around reflective greenhouse roofs near Espargos. Because the island holds very few labelled fields—almost all hydroponic plots under cover—the 2023 threshold refuses to colour any candidate at all. We will need extra, carefully curated labels if we want to reintroduce the

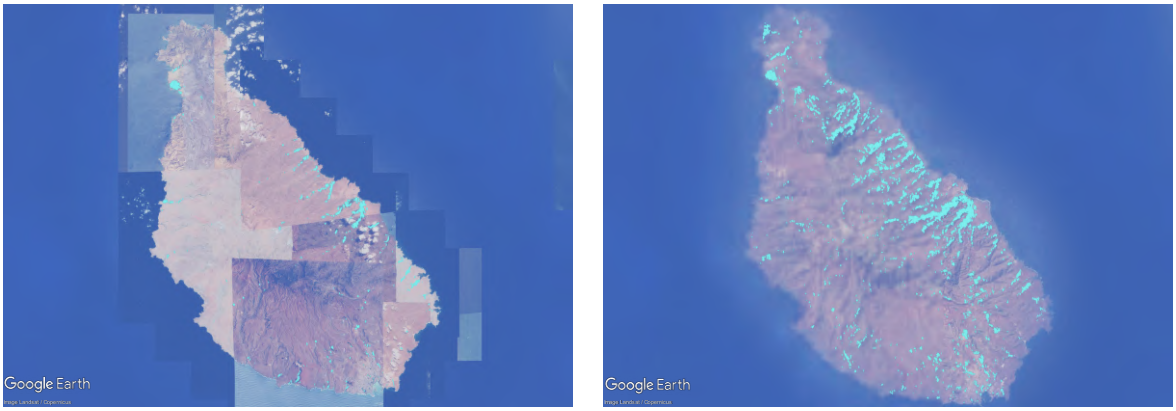
real hotspots without reviving the old noise.



2019 2023

Figure 5.15: Sal comparison between the pipeline predictions for 2019 and 2023.

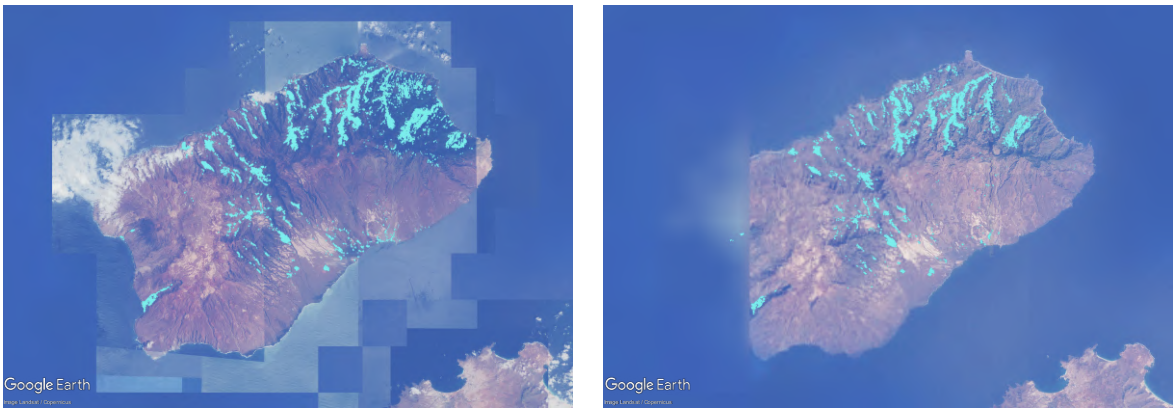
Santiago more than triples its predicted cropland, climbing from 41 197 to 140 657 pixels. Figure 5.16 shows larger blocks through the eastern valleys and along the inland plateaus, but the 2023 map also introduces extra false positives on the slopes and coastal fringe. In practice the 2019 baseline, although smaller, still sits closer to the confirmed farm footprint. I also cross-checked the 2023 hotspots with the validation fold outputs, and the uncertainty spikes align exactly with the bright coastal halos, reinforcing the idea that we should trim them before relying on the new totals.



2019 2023

Figure 5.16: Santiago comparison between the pipeline predictions for 2019 and 2023.

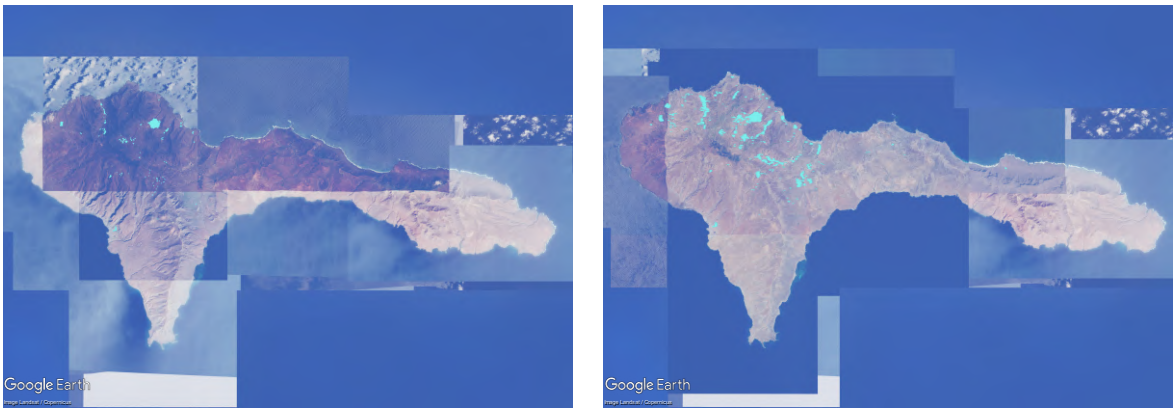
Santo Antão contracts slightly, moving from 184 865 pixels in 2019 down to 138 201 in 2023. Figure 5.17 confirms that the terrace belt along the northern valleys remains bright, but the model now trims the fringe areas. Even so, both versions probably carry a few false positives: the 2019 map still sprinkles highlights over steep forested slopes, and the 2023 rerun keeps some tiny patches near Ribeira Grande that look more like ornamental vegetation than crops. The next labelling round should focus on those high-gradient zones so the ensemble keeps the terrace core while damping the over-sensitive edges.



2019 2023

Figure 5.17: Santo Antão comparison between the pipeline predictions for 2019 and 2023.

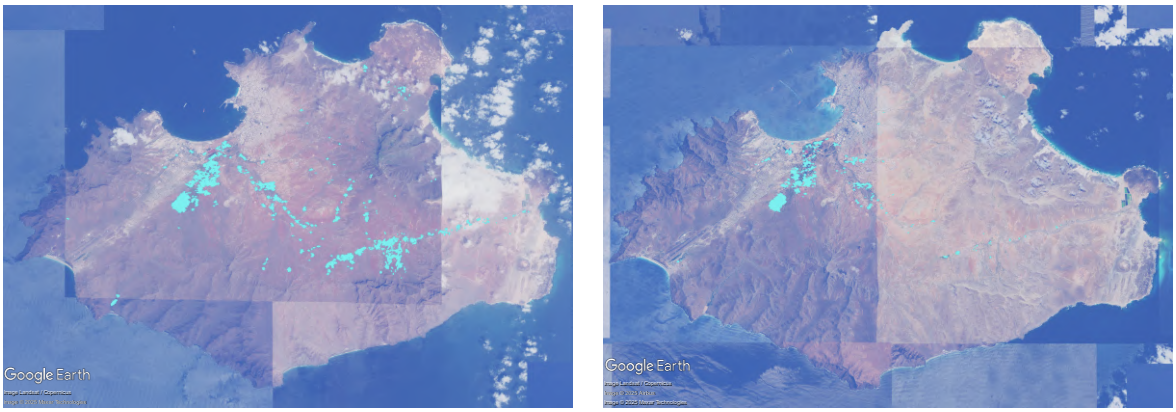
São Nicolau jumps from 8 974 to 49 786 predicted pixels. The 2019 map likely missed many northern-slope terraces because of haze, but the clearer 2023 composite in Figure 5.18 now recovers those ribbons. Even so, the 2019 baseline is probably closest to the ground truth because it only lights up the most certain terrace cores, whereas 2023 "gambles" a little by colouring additional slopes. In a few valleys that gamble pays off—the 2023 map hits hard-to-see plots that the earlier run skipped—but it also sprinkles new false positives over rocky ridgelines. Keeping both outputs side by side will help us target extra label collection precisely where the 2023 map differs from that conservative 2019 backbone.



2019 2023

Figure 5.18: São Nicolau comparison between the pipeline predictions for 2019 and 2023.

São Vicente settles at 18 524 pixels in 2023, trimming the 31 095 pixels previously tied to urban parks and golf courses. Figure 5.19 shows how the newer run keeps the irrigated plots near Ribeira de Vinha while pruning Mindelo's ornamental greenery. The footprint still deserves a quick sanity check, but it already matches field reports more closely than the over-optimistic 2019 mask.



2019 2023

Figure 5.19: São Vicente comparison between the pipeline predictions for 2019 and 2023.

5.5.1 Summary statistics

Table 5.4 compares the 2019 cropland counts with the new 2023 predictions exported by the pipeline. The counts use the auto-selected threshold for each island, so they act as the model's best estimate rather than a verified census.

Table 5.4: Cropland change summary for the two analysed seasons. Totals come from the 2019 baseline run and the 2023 retrospective run using the same configuration.

Island	2019 pixels	2023 pixels	Δ pixels	Notes
Boa Vista	4 195	815	−3 380	Irrigated core kept; dunes dropped.
Brava	133	9 334	+9 201	Cloud gaps expose terraces; trim bright hills.
Fogo	2 083	6 589	+4 506	Cleaner imagery lifts terraces despite shrub noise.
Maio	347	10 548	+10 201	Relax filters so drip strips persist.
Sal	4 887	0	−4 887	Hydroponic labels scarce and hard.
Santiago	41 197	140 657	+99 460	New valley blocks; prune coastal halo.
Santo Antão	184 865	138 201	−46 664	Fringe terraces trimmed; review steep slopes.
São Nicolau	8 974	49 786	+40 812	2019 core trusted; target slope labels.
São Vicente	31 095	18 524	−12 571	2023 keeps core farms; parks from 2019 disappear.

The numbers confirm that the 2023 rerun either consolidates genuine hotspots (Brava, Fogo, Maio, Santiago, and São Nicolau) or trims away areas where the 2019 model probably over-reported irrigated plots (Boa Vista, Sal, Santo Antão, São Vicente). The table also flags where we should schedule fresh ground checks—particularly on Sal, where the map is now blank, and on São Vicente, where the footprint tightened to a leaner core without the urban-park artefacts.

Chapter 6

Conclusion

6.1 Summary of Findings

I delivered and stress-tested an automatic pipeline that converts multi-temporal S2 composites into cropland probability maps for every inhabited island of Cape Verde. The auto-tuned ensemble captured tiny greenhouse footprints on Sal and Maio, while scaling to the broad terraces of Santo Antão and the irrigated valleys of Santiago. When placed next to the 2020 ESA land-cover layer, the maps consistently trimmed false cropland patches over dunes, lava flows, and city parks. High-confidence zoom-ins showed that parcel edges remain sharp even when fields are fragmented or sit beside bare ground.

Most importantly, the change analysis objective was approached with a like-for-like 2019 versus 2023 comparison. Cape Verde rarely opens brand-new agricultural frontiers, so the expected signal is subtle: small greenhouse clusters, incremental terrace maintenance, or irrigation tweaks inside existing valleys. At that resolution the current dataset cannot claim pixel-perfect change detection, yet the exercise still matters. By freezing the configuration I proved that the pipeline responds coherently to imagery quality, label density, and threshold tuning. When the maps expand (Fogo, Santiago, São Nicolau) they expose where better composites and relaxed thresholds uncover previously hidden crops. When they contract (Boa Vista, Sal, Santo Antão, São Vicente) they reveal where earlier optimism came from noise. The system therefore becomes a tool that detects agriculture with a controllable margin of error, governed by the quality of the input data, the analyst's labelling discipline, and the calibration of the models.

6.2 Contributions

The thesis delivers four concrete pieces:

- a scripted workflow that links GEE exports, feature engineering, ensemble calibration, and

sieve-based clean-up into a single resumable run;

- an active-learning recipe that mixes probability uncertainty with DBSCAN spacing so that labelling stays informative without overloading any one island corner;
- per-island artefact bundles (metrics, probability rasters, comparison figures, and KML overlays) plus configuration snapshots that make both the 2019 baseline and the 2023 rerun reproducible;
- a harmonised cropland footprint and its documented evolution between 2019 and 2023.

6.3 Limitations

Three factors still cap reliability:

- **Sparse validation.** There aren't enough labelled pixels, so precision and recall carry wide uncertainty bands.
- **Two snapshots.** The current evidence covers only the 2019 wet season and the upcoming 2023 rerun; different rainfall or irrigation patterns could still shift the spectral cues between those seasons.
- **Threshold sensitivity.** With limited positive labels, small threshold shifts can swing large areas between positive and negative, creating false positives in change maps (for example, the coastal halo on Santiago) unless analysts review the hotspots.
- **Pixel focus.** Manual labeling could insert some noise into the labels dataset, and mixed-use plots or glasshouses can still fool the classifier when guidance is thin.

Field checks and richer temporal stacks are needed to break through these limits.

6.4 Future Work

The next development cycle should:

- collect and validate a larger labelled dataset, combining random samples with analyst feedback to reduce noise and sharpen calibration;
- test alternative query strategies (margin sampling, committees) within the acquisition loop and capture reviewer notes to refine the score;
- validate the detected changes with targeted field checks or very-high-resolution imagery, prioritising islands where the model either blanked out cropland (Sal) or expanded aggressively (Santiago);

- derive parcel outlines from the probability rasters (for example, contour tracing with minimum-area filters) and experiment with higher-resolution or complementary sensors to lower false alarms over urban greenery and volcanic slopes. Maybe even using segmentation instead of pixel classification.

In spite of the uneven numerical gains, the thesis now offers a full, auditable story: a single pipeline that spans an archipelago, a defensible change assessment between 2019 and 2023, and a list of concrete actions to tighten the next rerun. That combination turns the models into a strategic planning tool for Cape Verde, highlighting exactly where analysts should invest their limited validation time.

Bibliography

[1] Direção Geral de Agricultura and Municipal Planning Departments. 'Stakeholder Interviews on Cropland Monitoring Needs'. Field interviews with municipal officers and Ministry of Agriculture analysts in Praia and São Nicolau, March–April 2024. 2024 (cit. on p. 3).

[2] Mariana Belgiu and Lucian Drăguţ. 'Random Forest in Remote Sensing: A Review of Applications and Future Directions'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 114 (2016), pp. 24–31. DOI: 10.1016/j.isprsjprs.2016.01.011 (cit. on p. 9).

[3] Leo Breiman. 'Random Forests'. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324 (cit. on p. 9).

[4] Tianqi Chen and Carlos Guestrin. 'XGBoost: A Scalable Tree Boosting System'. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA: ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785 (cit. on p. 9).

[5] Corinna Cortes and Vladimir Vapnik. 'Support-Vector Networks'. In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018 (cit. on p. 9).

[6] Matthias Drusch et al. 'Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services'. In: *Remote Sensing of Environment* 120 (2012), pp. 25–36. DOI: 10.1016/j.rse.2011.11.026 (cit. on pp. 7–8).

[7] Daphne Ewing-Chow. *How Small Island States Can Strengthen Food Security*. Forbes. June 2024. URL: https://www.forbes.com/sites/daphneewingchow/2024/06/06/how-small-island-states-can-strengthen-food-security/ (visited on 21/09/2025) (cit. on p. 2).

[8] ExtremeWeatherWatch. *Average Precipitation in Cape Verde by Year*. CRU TS-derived precipitation time series. 2025. URL: https://www.extremeweatherwatch.com/countries/cape-verde/average-precipitation-by-year (visited on 23/09/2025) (cit. on p. 2).

[9] Food and Agriculture Organization of the United Nations. *Drought within an Ocean*. May 2024. URL: https://www.fao.org/newsroom/story/drought-within-an-ocean/en (visited on 21/09/2025) (cit. on p. 2).

[10] GDAL Developers. *Cloud Optimized GeoTIFF (COG) Driver*. GDAL Documentation. 2023. URL: https://gdal.org/drivers/raster/cog.html (visited on 22/09/2025) (cit. on p. 11).

[11] Noel Gorelick et al. ‘Google Earth Engine: Planetary-Scale Geospatial Analysis for Everyone’. In: *Remote Sensing of Environment* 202 (2017), pp. 18–27. DOI: 10.1016/j.rse.2017.06.031 (cit. on pp. 8, 11).

[12] Ian C. Harris et al. *CRU TS4.09: Climatic Research Unit (CRU) Time-Series (TS) Version 4.09 of High-Resolution Gridded Data of Month-by-Month Variation in Climate*. Monthly precipitation totals aggregated to annual sums for Cape Verde. 2025 (cit. on p. 2).

[13] Kaiming He et al. ‘Deep Residual Learning for Image Recognition’. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778 (cit. on p. 10).

[14] International Monetary Fund. *Cabo Verde: Technical Assistance Report – Climate Policy Diagnostic*. Tech. rep. 2024/040. International Monetary Fund, 2024. DOI: 10.5089/9798400276675.019 (cit. on p. 2).

[15] Liang Ma et al. ‘Deep Learning in Remote Sensing Applications: A Meta-analysis and Review’. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 152 (2019), pp. 166–177. DOI: 10.1016/j.isprsjprs.2019.04.015 (cit. on p. 10).

[16] Filipa Monteiro et al. ‘Current Status and Trends in Cabo Verde Agriculture’. In: *Agronomy* 10.1 (2020), p. 74. DOI: 10.3390/agronomy10010074 (cit. on p. 2).

[17] Curtis G. Northcutt, Lu Jiang and Isaac L. Chuang. ‘Confident Learning: Estimating Uncertainty in Dataset Labels’. In: *Journal of Machine Learning Research* 22.185 (2021), pp. 1–52 (cit. on pp. 3, 12, 28).

[18] Miguel Pereira. ‘Active Learning for Improved Landcover Classification’. MA thesis. Faculdade de Ciências da Universidade do Porto, 2024 (cit. on pp. 3, 10–12, 28).

[19] S. C. Pereira, C. Lopes and J. P. Pedroso. ‘Mapping Cashew Orchards in Cantanhez National Park (Guinea-Bissau)’. In: *Remote Sensing Applications: Society and Environment* 26 (2022), p. 100746 (cit. on pp. 8, 12).

[20] Olaf Ronneberger, Philipp Fischer and Thomas Brox. ‘U-Net: Convolutional Networks for Biomedical Image Segmentation’. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Vol. 9351. Lecture Notes in Computer Science. Cham: Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28 (cit. on p. 10).

[21] Marc Russwurm and Marco K"orner. ‘Temporal Vegetation Modelling Using Long Short-Term Memory Networks for Crop Identification from Medium-Resolution Multispectral Satellite Images’. In: *Remote Sensing* 10.5 (2018), p. 755. DOI: 10.3390/rs10050755 (cit. on p. 10).

[22] Erik Sequeira et al. ‘The Current Status of Irrigated Agriculture in Cape Verde and Its Link to Water Scarcity’. In: *Agronomy* 15.7 (2025), p. 1625. DOI: 10.3390/agronomy15071625 (cit. on p. 2).

[23] Burr Settles. *Active Learning Literature Survey*. Tech. rep. 1648. University of Wisconsin–Madison, 2009 (cit. on p. 10).

[24] Devis Tuia et al. 'Active Learning Methods for Remote Sensing Image Classification'. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.5 (2011), pp. 2282–2296. DOI: 10.1109/TGRS.2009.2039476 (cit. on p. 10).

[25] Sofia Perestrelo de Vasconcelos C. Pereira. 'Remote Sensing and Machine Learning Tools for Vegetation Monitoring'. MA thesis. Faculdade de Ciências da Universidade do Porto, 2020 (cit. on p. 12).

[26] Xiao Xiang Zhu et al. 'Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources'. In: *IEEE Geoscience and Remote Sensing Magazine* 5.4 (2017), pp. 8–36. DOI: 10.1109/MGRS.2017.2762307 (cit. on p. 10).

Appendix A: Configuration Overview

Table 1 lists the main configuration entries referenced in Chapters 3 and 4. The file `scripts/config.py` stores them in plain Python; each round writes a copy of the active values to `statistics/config_snapshot.json`.

Table 1: Key configuration defaults for the São Nicolau (2019) run.

Parameter	Value
Seasonal windows	Jan–Apr, May–Aug, Sep–Dec
Excluded spectral bands	B1, B9 (removed during training)
Texture window	7×7 NDVI mean/std per season
MIN_AGRI_PROB	0.35 (auto best-threshold enabled)
SIEVE_MIN_SIZE	5 pixels, pixel mode, keep-prob 0.80 (off)
Candidate count	50 (uncertainty floor 0.28, hard-negative quota 40%)
Diversity filter	DBSCAN, eps = 0.8 km, min samples = 1
SVM baseline	$C = 3.0$, $\gamma =$ `scale`, balanced weights, cache 2048 MB
RF baseline	$n_{\text{estimators}} = 200$, depth $= 10$, leaf size $= 1$, balanced weights
Calibration	Isotonic (3 folds), repeated validation $n = 10$
Auto-tuning	Up to 6 combos per model (SVM/RF/ensemble)
Batch size	1,000,000 pixels per inference chunk
Thread caps	BLAS/OMP = 8, tile threads = 16, polygon workers = 1
Memory watcher	Cleans caches once overall usage exceeds about 70%
Persistent lists	Highscore top 10k, ProbableAgri top 10k
Final sweep	Thresholds 0.28/0.33/0.38, sieve sizes 0/5/10
Final grid search	12 SVM combos, 8 RF combos, 1 ensemble stacking

Appendix B: Metrics Definitions

Table 2 lists every metric referenced in this thesis. The middle column uses plain language, while the right-hand column gives the formula whenever a compact expression exists. A dash indicates metrics that are computed numerically (for example, by integrating a curve) rather than through a single closed-form expression.

Table 2: Performance metrics used throughout the thesis. TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives respectively; p_i is the predicted probability for pixel i, $y_i \in \{0, 1\}$ the ground-truth label, and N the number of evaluated pixels.

Metric	Plain meaning	Formula
Precision	Share of predicted cropland pixels that are actually cropland	$\dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$
Recall (TPR)	Share of real cropland pixels that we recovered	$\dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$
Specificity (TNR)	Share of non-cropland pixels correctly kept as non-cropland	$\dfrac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$
Balanced accuracy	Average of recall and specificity so both classes count equally	$\dfrac{1}{2}\left(\dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} + \dfrac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}\right)$
F_1 score	Harmonic mean of precision and recall, highlighting when both are strong	$\dfrac{2\,\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$
Accuracy	Overall share of correctly classified pixels	$\dfrac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$
Brier score	Mean squared gap between predicted probability and reality	$\dfrac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$
AUC (ROC)	Chance that a random cropland pixel ranks above a random non-cropland pixel	–
AUC-PR	Area under the precision–recall curve, rewarding high precision at many recall levels	–

Appendix C: Spectral Feature Glossary

Table 3 summarises the Sentinel-2 bands and vegetation indices that feed the feature stack described in Chapter 3.

Table 3: Spectral bands and vegetation indices used in the pipeline.

Feature	Category	Rationale for inclusion
B2 (Blue)	Sentinel-2 band	Captures aerosol and coastal haze; helps flag residual cloud or water contamination.
B3 (Green)	Sentinel-2 band	Emphasises chlorophyll reflectance for vegetative vigour checks.
B4 (Red)	Sentinel-2 band	Sensitive to chlorophyll absorption; core input for vegetation indices.
B5, B6, B7 (Red-edge)	Sentinel-2 bands	Track subtle canopy structure changes linked to crop growth stages.
B8 (Near infrared)	Sentinel-2 band	Responds strongly to healthy vegetation; differentiates crops from bare ground.
B8A (Narrow near infrared)	Sentinel-2 band	Provides finer sampling of the red-edge plateau, improving crop-background contrast.
B11, B12 (Shortwave infrared)	Sentinel-2 bands	Measure canopy moisture and soil background; distinguish irrigated plots from dry scrub.
NDVI	Vegetation index	Ratio of near infrared to red; robust indicator of greenness and biomass.
EVI2	Vegetation index	Two-band variant of Enhanced Vegetation Index; reduces soil and atmospheric influence in semi-arid scenes.
GNDVI	Vegetation index	Uses green and near infrared bands to detect subtle chlorophyll shifts, useful for crop stress signals.
NDWI	Water index	Highlights surface and canopy moisture, aiding separation of irrigated crops from dry vegetation.
NDRE	Red-edge index	Focuses on red-edge response to capture chlorophyll dynamics in dense canopies.

Appendix D: Ancillary Descriptors and Textures

Table 4 lists the non-spectral layers that complement the composite (terrain context, textures, and quality masks). These layers provide structural cues the spectral stack alone cannot capture.

Table 4: Ancillary descriptors and textures that complement the spectral stack.

Feature	Category	Rationale for inclusion
Terrain slope	Ancillary raster	Differentiates terraced agriculture on steep slopes from flat shrublands.
Terrain aspect	Ancillary raster	Notes illumination direction; south-facing slopes in Cape Verde often dry faster than north-facing ones.
Elevation	Ancillary raster	Provides altitude context for microclimate zones and crop viability.
Seasonal NDVI texture (mean, std)	Derived texture	Encodes local spatial patterns, such as terrace layouts, that help the classifier avoid single-pixel noise.
Cloud and shadow masks	Quality flags	Remove unreliable observations so active learning focuses on valid pixels.

Appendix E: Repository Artefacts and CLI Reference

Beyond the main text, the repository stores additional artefacts that support auditing and future development:

- Feature-importance tables produced during model evaluation (stored alongside each round's statistics).
- Candidate exports (CSV and KML) for São Nicolau Round 1 under the corresponding round directory.
- A runtime log tracking per-stage durations (in the `data/phase1` tree).
- Persistent informative-pixel lists: 9,558 entries in the Highscore CSV and 148,795 in the ProbableAgri CSV.

Analysts typically run the pipeline through the command in Listing 1. The script offers interactive prompts but also honours command-line flags, enabling unattended execution for batch experiments.

```
# Activate the project environment and launch the orchestrator
source .venv/bin/activate
python scripts/ready_to_run_phase1.py \
    --island sao_nicolau \
    --model SVM \
    --auto-threshold \
    --resume

# Optional dry run to generate candidate lists without inference
python scripts/ready_to_run_phase1.py --dry-run --island santiago
```

Listing 1: Command-line invocation of the end-to-end pipeline

Figure 1: Directory layout under `data/phase1/rounds/round_1`. Ellipses mark repeated artefacts that follow the same structure.

```
2025-09-21 09:12:04 | ready_to_run_phase1.py --resume --island
   sao_nicolau
[setup] verifying Earth Engine authentication ... OK
[setup] scanning data/phase1/raw (18 tiles) ... all present
[labels] merged labels.csv (3,482 rows) + temp_labels.csv (16 new)
   -> 3,498 snapped rows
[cache] reused 42 feature stacks; refreshed 3 tiles
   (sao_nicolau_tile05, ...)
[auto-tune] curated SVM grid (12 combos) -> best C=3.0 gamma=scale
   class_weight=balanced
[train] SVM fit on 3,498 rows (elapsed 00:02:18); isotonic
   calibration 00:00:31
[inference] streamed 18 tiles ->
   data/phase1/rounds/round_x/_tile_preds/ (00:07:46)
[merge] consolidated predictions.csv (18 tiles) and probability
   rasters; shards removed
[candidates] scored 12,000 pixels -> Highscore 10,000 /
   ProbableAgri 9,558 (DBSCAN eps=1.2 km)
[postprocess] thresholds {0.30, 0.35, 0.40}, sieve 5 ->
   statistics/normal_threshold/ and best_threshold/
[archive] saved models/, statistics/, config_snapshot.json,
   runtime_metrics.csv
[checkpoint] updated data/phase1/checkpoint.txt -> round_x
   postprocess complete
```

Figure 2: Condensed console log emitted by the orchestration script during a resumed round.