HELLENIC REPUBLIC
MINISTRY OF DEVELOPMENT
GENERAL SECRETARIAT FOR RESEARCH AND INNOVATION
HELLENIC FOUNDATION FOR RESEARCH AND INNOVATION

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN
"BASIC RESEARCH FINANCING" (Horizontal support for all Sciences)
ID 16618 – Subproject 1 (MIS: 5163923)

D3.2 Report and tools for EO data acquisition, processing, and transformation

Spatially Explicit Digital Twin of the Greek Agro-Hydro-System

ID 14815

Plan Details

Report and tools for EO data acquisition, processing, and transformation
Deliverable number: D3.2
Creation Date: 02/04/2025
Last modification date: 20/09/2025
Dissemination Level: Public

1. Introduction

This deliverable (D3.2) documents the implementation of the Earth Observation (EO) data workflows developed within Work Package 3 (WP3) to support the DT-Agro. Building on the evaluation and selection of EO datasets presented in D3.1, this report focuses on the streamlining of data acquisition, processing, and transformation procedures, as defined under Task T3.3, so that the selected datasets are directly usable by the DT-Agro modeling and spatial database infrastructure.

This report focuses on the operational tools, scripts, and procedures used for data acquisition, preprocessing, transformation, and integration into the DT-Agro spatial database and modeling framework. The objective of this deliverable is to demonstrate that a standardized, automated, and reproducible EO data pipeline has been established, enabling the consistent preparation of heterogeneous datasets for national-scale agro-hydrological modeling. This work directly supports Milestone M3.1, i.e. the establishment of EO data flow to DT-Agro. This data flow transforms heterogeneous EO products (NetCDF, GeoTIFF, gridded reanalysis) into analysis-ready datasets aligned to the DT-Agro requirements (projection, resolution, naming, and metadata).

Although the DT-Agro is described in detail in WP2 deliverables (D2.1 and D2.2), D3.2 includes the necessary context on how EO inputs are prepared and structured to enable ingestion by the modelling framework. It integrates the outcomes of the evaluation activities carried out under Task T3.2, where selected EO products were assessed and compared against alternative sources and available reference data to quantify accuracy and identify limitations, as well as the implementation activities of Task T3.3, which aimed to streamline data acquisition and processing into reproducible, operational workflows.

2. Overview of Data Sources

The EO datasets processed under WP3 represent the key variables for DT-Agro initialization, parameterization, evaluation and updating. These include:

- **Meteorological data**: Meteorological forcing is based on AgERA5 reanalysis data, providing daily precipitation, mean air temperature and reference evapotranspiration. AgERA5 offers spatially continuous coverage and long temporal records, which are essential for national-scale applications and historical analyses.

- **Soil datasets**: Soil hydraulic and physical properties are derived from the Greek Soil Map and complemented by global datasets from ISRIC and ESDAC. These datasets provide the parameters required for soil water balance calculations, runoff estimation and erosion modelling.
- **Agricultural Parcel datasets**: Incorporated through the Integrated Administration and Control System (IACS) database. The IACS dataset provides parcel-level information on declared crop types, agricultural land boundaries, and management-related attributes, offering a high level of thematic detail for cultivated areas. Although IACS is not an EO product, it constitutes a critical ancillary dataset that complements EO-derived information
- **Soil Moisture data**: EO-based surface soil moisture products (currently supporting evaluation and future assimilation) using Sentinel-1 & 2 Data, EU-DEM with topography and soil moisture field measurements. These products support the evaluation of antecedent moisture conditions and are used to characterize soil wetness conditions and refine Curve Number (CN) estimates under varying moisture.
- **Digital Elevation Model**: Topographic information is derived from the Copernicus EU-DEM. The DEM provides the basis for terrain analysis, hydrological routing, flow accumulation and erosion-related calculations.
- **Land Cover / Land Use**: Land cover information is obtained from the Copernicus Land Monitoring Service (CLMS), i.e. CORINE Land Cover (all available years) and CLC Backbone (2018, 2021, 2023) to describe land cover patterns and temporal changes. These data are complemented by IACS information in agricultural areas, which provides field-level crop and management details.
- **Imperviousness Density**: CLMS imperviousness layers used to represent the fraction of sealed surfaces. These fractions are directly used in the SCS-CN parameterization and in the new runoff-generation method that explicitly integrates impervious areas at cell level.
- **NDVI**: Vegetation dynamics are represented using NDVI products from the CLMS, derived from Sentinel-3 observations and provided as 10-daily composites.

3. EO Data Acquisition Workflows - Tools and Scripts

The EO data pipeline was implemented primarily through automated Python workflows, supported where necessary by ArcGIS Pro geoprocessing tools (ArcPy) for national-scale raster management. The workflow was designed to be rerunnable, enabling both historical reprocessing and future near-real-time updates.

Across datasets, the workflow includes automated acquisition via API or bulk download, decoding and extraction (including unzipping and NetCDF handling), quality control and unit conversions, harmonisation to DT-Agro spatial specifications, and export into standardized formats for storage in the DT-Agro spatial database.

The scripts rely on widely used open-source libraries (e.g. xarray, netCDF4, pandas, rasterio, geopandas) and produce outputs in GeoTIFF for rasters and CSV for station series and aggregated time series. Intermediate products are stored in structured directories and naming conventions (dataset, variable, year, version) to ensure traceability.

To ensure spatial consistency across all EO inputs, datasets were harmonised to the same projection i.e. EGSA87 / Greek Grid (EPSG:2100) and analysis resolution of 100 m (static and agro-hydrological layers), with support for coarser grids where required by forcing data.

3.1 Meteorological data acquisition and preprocessing

Meteorological EO forcing data were acquired from the Copernicus Climate Data Store (CDS) using the AgERA5 reanalysis dataset. After assessing multiple criteria, the project team selected the AgERA5 dataset as the primary Earth Observation-based meteorological data source for DT-Agro.

AgERA5 is a climate reanalysis product developed under the Copernicus Climate Change Service (C3S) and derived from ERA5, providing daily global agrometeorological information specifically tailored for agricultural and agro-ecological applications. The dataset covers the period from 1979 to the present and includes key variables such as precipitation, air temperature, solar radiation, wind speed, and reference evapotranspiration, at a spatial resolution of 0.1°. Its long temporal coverage, internal consistency, and direct availability through the Copernicus Climate Data Store (CDS) make it suitable for national-scale agro-hydrological modelling and Digital Twin applications. Within DT-Agro, AgERA5 provides the core meteorological forcing required for hydrological balance calculations, evapotranspiration estimation, and crop water demand assessment.

3.1.1 Data Extraction

Automated retrieval was implemented using the CDS API and Python scripting. Daily time series are extracted at the exact point locations of 140 meteorological stations operated by the Hellenic National Meteorological Service (HNMS). The geographic coordinates of all stations were expressed in the WGS84 reference system (latitude and longitude), and each station was associated with a unique identifier that was preserved throughout the entire data acquisition and processing workflow to ensure full traceability. Variables downloaded include 2 m mean air temperature, precipitation flux (converted to daily precipitation depth where required), and reference evapotranspiration (used for ET and crop-water modules).

To comply with CDS request limitations and avoid oversized downloads, data retrieval was structured on a year-by-year basis for all stations. Each request returned a compressed .zip archive containing NetCDF files with daily time series for the requested variable, year, and station. Files were named systematically to include the variable type, year, and station identifier (indicatively 2m_temperature_1979_16600.0.zip, precipitation_flux_1979_16600.0.zip, reference_evapotranspiration_1979_16600.0.zip). Separate Python scripts were developed for the extraction of air temperature, precipitation flux, and reference evapotranspiration

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN Funded by the European Union NextGenerationEU

3.1.2 Preprocessing and Format Transformation

Once downloaded, all data were processed using Python workflows. The .zip archives were automatically extracted, and the contained NetCDF files were handled using the xarray and pandas libraries. Although NetCDF files can also be opened using GIS software such as QGIS, this option was not used in the present project.

Additional preprocessing steps were applied. Air temperature values were converted from Kelvin to degrees Celsius, while precipitation and evapotranspiration units were verified to ensure consistency with hydrological modelling requirements. All processed data were reorganized into station-specific folders and exported as CSV files, maintaining the same systematic naming convention to preserve the link between variable, year, and station.

The meteorological workflow was designed to support both gridded forcing and point-based extractions for later bias correction and evaluation activities. Detailed evaluation methodologies (e.g. performance metrics) are reported in WP5.

3.1.3 Evaluation of AgERA5 against Meteorological Observations

As part of the EO data assessment activities, an evaluation procedure was implemented to examine the consistency and reliability of the AgERA5 reanalysis dataset against ground-based meteorological observations over Greece. The objective of this analysis was to assess potential biases and uncertainties in AgERA5 prior to its operational use within the DT-Agro, and to inform subsequent bias-correction and interpolation strategies.

The year 2023 was selected for the evaluation, as it represents the most recent period with relatively complete and spatially representative station records across Greece. From the full AgERA5 archive downloaded via the Copernicus CDS, only data corresponding to the year 2023 were used for validation against available station observations.

The evaluation focused on two key meteorological variables that are critical for agro-hydrological modelling, i.e. daily 24-hour mean air temperature, and daily precipitation. For consistency with hydrological and climatic analyses, daily AgERA5 values were aggregated into monthly mean temperature and monthly cumulative precipitation. Ground-based observations from 777 meteorological stations distributed across Greece for the year 2023 were processed, ensuring that both datasets were compared on identical temporal scales.

Prior to comparison, unit standardization was applied. Air temperature values provided by AgERA5 in Kelvin were converted to degrees Celsius, while precipitation values were verified to be expressed in millimeters. For the spatial matching of datasets, the AgERA5 grid cell corresponding to each station location was identified using nearest-neighbor extraction, whereby each station was assigned the value of the closest AgERA5 grid point for each month.

The comparison between AgERA5 and station observations was designed to quantify systematic and random errors, as well as the ability of the reanalysis dataset to reproduce observed temporal variability. To this end, a set of standard statistical indicators was defined, including bias, mean absolute error (MAE), root mean square error (RMSE), and the Pearson

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN Funded by the European Union NextGenerationEU

correlation coefficient. These metrics were computed separately for temperature and precipitation and summarized across stations using descriptive statistics and graphical diagnostics.

The evaluation procedure provides a methodological foundation for assessing the suitability of AgERA5 for agricultural and hydrological applications in Greece. While the full quantitative results and their spatial interpretation are presented in Deliverable D5.2, the methodological framework established here supports Task T3.2 and ensures that EO meteorological inputs integrated into DT-Agro are critically evaluated prior to operational use.

3.2 Soil Data acquisition, processing and integration

Within WP3, soil datasets were selected, harmonised, and transformed to ensure spatial consistency, computational efficiency, and compatibility with EO-driven workflows and the DT-Agro modelling core.

Soil data were compiled from multiple sources to ensure full spatial coverage of Greece and to support the derivation of Curve Number (CN) values and the soil hydraulic properties required by the DT-Agro. The primary source of soil information was the Greek Soil Map at a scale of 1:30,000 (OPEKEPE, n.d.). To complement this national dataset and address spatial gaps, additional EO soil information was obtained from the European Soil Database v2.0 at a scale of 1:1,000,000 (Panagos et al., 2012) and from top-soil physical property datasets derived from the LUCAS Land Use/Cover Area frame Survey (Panagos et al., 2012; Orgiazzi et al., 2018).

Raster datasets from ISRIC SoilGrids, ESDAC, and the Greek national soil database were first collected and harmonised. All datasets were subset to the topsoil layer (0-30 cm), clipped to the Greek territory, and reprojected to the national reference system (EGSA87, EPSG:2100) to ensure spatial consistency. The preprocessing workflow includes clipping to Greece, reprojection to EPSG:2100, resampling to 100 m, and generating consistent soil property rasters across sources for use in model parameter estimation converts soil texture and related soil attributes into raster layers aligned to the DT-Agro grid. Key soil property values, namely sand, silt, and clay percentages, were extracted at sampling point locations to enable direct comparison between observed values from the Greek Soil Map and predicted values from the international datasets.

Soil texture classes were assigned to each sampling point using the USDA soil texture classification system. Texture classes derived from the international datasets were then compared against those obtained from the Greek Soil Map. The agreement between datasets was evaluated using categorical accuracy metrics, including producer's accuracy, user's accuracy, and overall accuracy. This analysis provided a clear assessment of the suitability of global soil datasets for applications in Greece and informed the selection and adjustment of soil parameters used in DT-Agro. The evaluation of global datasets against the Greek Soil Map is reported in WP5 deliverables.

3.3 Soil moisture

Soil moisture is included in the DT-Agro as a spatially explicit EO-derived dataset providing estimates of near-surface soil water content over agricultural areas of Greece. The dataset is generated through an automated processing workflow that integrates Sentinel-1 SAR observations with optical vegetation information from Sentinel-2 NDVI products, ancillary geospatial data, and in-situ soil moisture measurements for calibration and validation.

Sentinel-1 SAR data are preprocessed using standard radiometric calibration, noise removal, speckle filtering, and terrain correction procedures based on the Copernicus EU-DEM. Vegetation effects on radar backscatter are accounted for using NDVI, enabling the isolation of the soil signal. A machine-learning model, trained on in-situ observations and EO-derived predictors, is then applied to produce gridded soil moisture estimates.

All soil moisture products are reprojected to EGSA87 (EPSG:2100), harmonised to the DT-Agro spatial grid, and stored as georeferenced raster layers. These datasets are directly integrated into the DT-Agro model and support hydrological modeling, crop water balance assessment, and irrigation analysis. The workflow is fully reproducible and designed to support future extensions, including enhanced validation and climate-related applications. These datasets are suitable for use in soil water balance assessment, irrigation monitoring and model evaluation.

3.4 NDVI acquisition, decoding, harmonisation, and derivation of Kc and ETc

As part of the EO data acquisition and processing, vegetation index data were incorporated to support vegetation monitoring, evapotranspiration modeling and CN estimation within the framework of DT-Agro. The Normalized Difference Vegetation Index (NDVI) datasets were obtained from the Copernicus Land Monitoring Service (CLMS), specifically the Global NDVI 300 m Version 2 product (Copernicus Land Monitoring Service, 2021). The dataset is derived from Sentinel-3/OLCI optical observations and provides global 10-daily composites of BRDF-normalized, top-of-canopy reflectances.

Data were downloaded automatically using Python scripts, which interact with the CLMS API and the Copernicus Global Land Service (CGLS) data portal. The downloaded files are provided in NetCDF4 format following Climate and Forecast (CF) v1.6 metadata conventions and include NDVI values, associated uncertainty (NDVI_unc), number of observations (NOBS), and quality flags (QFLAG). Each NDVI file corresponds to a 10-day period (days 1–10, 11–20, and 21–end of month), with filenames indicating the start date of the period. The original product is provided on a global regular latitude/longitude grid (EPSG:4326) with a spatial resolution of 1/336° (~300 m), referenced to the WGS84 ellipsoid. Physical NDVI values were recovered programmatically using the scaling relation provided by the product documentation:

$$\text{NDVI\_real} = (\text{DN} \times 0.004) - 0.08.$$

Invalid and non-terrestrial pixels were filtered based on the product flags and reserved digital values (e.g. missing data and sea masks). This scaling means that DN = 0 corresponds to NDVI = -0.08 and DN = 250 corresponds to NDVI = 0.92. The conversion was performed automatically within the Python preprocessing workflow using xarray and rasterio libraries. Pixels representing sea, missing data, or invalid reflectance values were filtered according to the metadata flags (e.g., NDVI = 255 for missing data, NDVI = 254 for sea).

After decoding, NDVI rasters were clipped to Greece, reprojected to EPSG:2100, and resampled to 100 m to match the DT-Agro modelling grid. Additionally, harmonised NDVI products were used to derive indicative crop coefficient (Kc) maps. Since Kc depends on land cover and crop type, the workflow supports crop-specific formulations. Indicatively, the relationship proposed by Montgomery et al. (2015) was adopted for cotton under conditions similar to Greece. Combined with reference evapotranspiration (ET_o) from AgERA5, the NDVI-derived Kc supports the generation of ETc layers for use in evapotranspiration and crop-water balance modules. Kc values are assigned from a knowledge base for each land-cover / crop type and are dynamically adjusted using NDVI, capturing intra-seasonal variations in crop development and canopy cover.

3.5 Land cover and Imperviousness density preprocessing

CORINE Land Cover (CLC) data were used in DT-Agro to provide consistent spatial information on land use and land cover for parameterization of hydrological, crop, and soil erosion processes. CLC Backbone datasets for the reference years 2018, 2021, and 2023 were obtained from the CLMS and processed through a standardized preprocessing workflow to ensure compatibility with other EO-derived and ancillary datasets. All land cover products were harmonised to provide consistent national coverage and support hydrological parameterization.

All CLC datasets were spatially subset to the Greek territory, including a small buffer zone to avoid edge effects during resampling and routing operations. The datasets were then reprojected from their native geographic coordinate system (EPSG:4326) to the national reference system EGSA87 (EPSG:2100), which is used consistently across the DT-Agro spatial database. This reprojection ensured spatial consistency with soil maps, DEM derivatives, meteorological grids, and other EO layers.

Regarding the CLC Backbone 2018 and 2021 datasets, rasters were reprojected to EGSA87 (EPSG:2100), resampled from 10 m to 100 m (categorical resampling), and clipped to a national mask so that all layers share identical spatial extent and grid alignment. A key processing issue was the difference between years. The 2023 CLC Backbone was downloaded as tiles (AOI: Greece) from the WEkEO platform and required automated merging using ArcPy, followed by reprojection to EGSA87, resampling to 100 m, and extraction by the national boundary. The tiling issue was resolved through an ArcPy-based mosaic workflow, which merged tiles into a single seamless raster for Greece. The 2018 and 2021 products were harmonized through equivalent reprojection, resampling, and clipping procedures.

Imperviousness density information was incorporated into the DT-Agro to explicitly represent the spatial distribution of sealed surfaces and to support hydrological parameterization, particularly the estimation of surface runoff through the SCS-CN methodology. Imperviousness data were obtained from the CLMS High Resolution Layer (HRL) Imperviousness Density products, which provide percentage estimates of sealed surfaces at high spatial resolution. As a first preprocessing step, the datasets were clipped to the territorial extent of Greece and reprojected to the national coordinate system EGSA87 (EPSG:2100) to ensure consistency with all other EO and ancillary datasets used in the project.

The resulting imperviousness density maps provide, for each 100 m grid cell, the fraction of impervious area expressed as a percentage. These layers are directly used in the DT-Agro hydrological module to support the explicit separation of each grid cell into pervious and impervious sub-areas. In this framework, the impervious fraction is assigned a CN equal to 100, while the remaining pervious fraction is characterized by land-cover- and soil-dependent CN values.

All preprocessed imperviousness density layers were stored in the common DT-Agro spatial database together with land cover, soil, and topographic datasets. The standardized projection, resolution, and file structure allow imperviousness information to be seamlessly combined with other EO-derived inputs and to be updated in future runs if newer Copernicus HRL products become available.

3.6 DEM preprocessing and terrain derivatives

Topography was derived from the Copernicus EU-DEM. The DEM was clipped to Greece, reprojected to EGSA87, and resampled to 100 m to match the modelling grid. From the harmonised DEM, terrain derivatives were generated to support hydrological routing and erosion modelling, including fill DEM, slope, flow direction, flow accumulation, flow length, and additional routing-related indices required by the modelling framework. For these key terrain ArcGIS Pro was used.

4. Validation and Accuracy checks

The WP3 workflows incorporate consistency checks at multiple stages to ensure that processed EO layers are usable and physically plausible. These checks include:

- verification of units and scaling transformations (e.g. NDVI scaling, temperature unit conversion),
- spatial alignment checks (extent, projection, resolution),
- inspection of missing-data patterns and quality masks, and
- verification that categorical resampling preserves land cover class distributions at 100 m.

More advanced validation of EO products against observations (e.g. reanalysis evaluation, soil dataset accuracy assessments) is documented in later deliverables, as it pertains to system functioning and interpretation rather than the operational data pipeline.

5. Integration into the DT-Agro

All EO data and in-situ, processed within WP3 are integrated into the DT-Agro through a unified spatial data infrastructure that ensures consistency, traceability, and operational compatibility with the model core. The integration framework is designed to allow heterogeneous datasets, differing in spatial resolution, temporal frequency, and source, to be harmonised and ingested seamlessly into the DT-Agro modelling environment. Workflows developed are designed to support dynamic datasets. Time-varying datasets such as meteorological variables (AgERA5 temperature, precipitation, reference evapotranspiration), vegetation indices (NDVI), soil moisture, and land surface characteristics are handled through automated scripts that are structured by date, period, and version, allowing new data to be appended seamlessly to existing archives.

The integration involves a harmonised spatial database, where all datasets are stored in standardized formats, projections, and resolutions. This harmonisation ensures that each grid cell represents a consistent spatial unit across all DT-Agro components.

Meteorological datasets, including bias-corrected precipitation, temperature, and reference evapotranspiration, are integrated as gridded daily time series on the meteorological grid. These datasets provide the primary atmospheric forcing for the hydrological and crop water balance modules. The spatial database links each agro-hydrological grid cell to the corresponding meteorological cell, enabling efficient data access during model execution without repeated spatial interpolation.

Soil datasets, compiled from the Greek Soil Map, ISRIC SoilGrids, and ESDAC, are integrated as static parameter layers describing soil texture, hydraulic properties, water holding capacity, and erodibility factors. These layers feed directly into multiple DT-Agro processes, including CN estimation, soil moisture dynamics, deep percolation calculations, and soil erosion modelling. The soil layers ensure that all model components rely on a consistent representation of subsurface properties.

Land cover and land use information from CORINE Land Cover and CLC Backbone products, complemented by IACS parcel-level data where available, are integrated as categorical raster layers and vector datasets. These datasets control land-surface parameterization, crop identification, and management practices within DT-Agro. They are used to assign crop coefficients (Kc), vegetation parameters, CN for pervious areas, and cover-management factors for erosion modelling. Temporal versions of land cover datasets allow the system to reflect land-use changes over time.

Imperviousness density layers from the Copernicus HRL are integrated as continuous raster datasets representing the fraction of sealed surfaces within each grid cell. These layers are directly coupled with the runoff-generation algorithms, enabling the explicit separation of pervious and impervious sub-areas within each cell. This integration significantly improves runoff estimation in urban, peri-urban, and infrastructure-affected areas.

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN Funded by the European Union NextGenerationEU

NDVI from the CLMS, are integrated as time-varying rasters. These datasets dynamically update vegetation-related parameters, such as crop coefficients (Kc) and cover factors (C), enabling DT-Agro to represent seasonal crop development and spatial variability in vegetation condition. NDVI-derived parameters are accessed by crop growth, evapotranspiration, and erosion modules during each simulation step.

Surface soil moisture datasets derived from Sentinel-1 SAR and supporting EO products are integrated as auxiliary dynamic layers. At the current stage, these datasets are used primarily for model evaluation, calibration support, and consistency checks of simulated soil moisture patterns. The integration framework is designed to allow future use of EO soil moisture for state updating and data assimilation once methodological development is finalized.

This organization enables reproducibility, supports reprocessing when updated EO products become available, and allows the DT-Agro to evolve from historical simulations toward near-real-time and scenario-based applications. Through this structured integration approach, the spatial database connects EO data acquisition workflows with the DT-Agro modelling core, ensuring that all processed datasets are operationally exploitable within the Digital Twin framework.

6. Climate change impact and mitigation and adaptation strategies

Although the climate change impact, mitigation and adaptation strategies are planned for the end of the project, the methodological foundation has already been established. The datasets, techniques and modeling workflows form the basis for future simulations under different climatic conditions.

Due to technical challenges faced during data acquisition, preprocessing and model implementation, as well as limited time available during the reporting period, a complete climate scenario could not be implemented at this stage.

Nevertheless, the system has been designed to support this functionality. The established workflow allows the replacement of historical or near-real-time meteorological inputs with projected datasets (e.g., AgERA5-derived climate scenarios or downscaled regional climate model outputs). This will enable the estimation of future crop water demand, evapotranspiration patterns, and potential impacts on irrigation requirements once the data for 2024 and beyond become available.

7. Field data supporting EO evaluation and model validation

In parallel with EO workflows, multiple field campaigns were conducted to support calibration and validation of EO products and model outputs. During the reporting period, targeted field visits were conducted in representative agricultural regions of Greece. In particular, during summer 2024, field campaigns were organized in the areas of Katerini and Kavala, where data on irrigation water consumption and agricultural water management practices were collected in cooperation with local stakeholders and irrigation organizations.

These data were used to support the evaluation of model inputs related to irrigation demand and to provide reference information for EO-based indicators.

Additionally, an operational network of soil moisture monitoring stations was used to provide in-situ measurements for the calibration and validation of EO-derived soil moisture products. These station observations were essential for assessing the consistency and reliability of satellite-based soil moisture estimates and for supporting the development of automated soil moisture processing workflows described in earlier sections of this deliverable. The station data were spatially matched with EO products and model outputs to enable quantitative comparison and performance assessment.

Additional field campaigns focusing on more extensive soil moisture measurements, UAV-based water stress indicators, crop condition assessment, and irrigation system performance were under preparation during the reporting period and are planned to be implemented after April 2025. These activities form part of the continuation of the project within the associated doctoral research programme and will further strengthen model validation and EO evaluation in subsequent phases.

Overall, the field data collected up to April 2025 provide a solid empirical basis for evaluating EO datasets and supporting the initial calibration and validation of DT-Agro. The established field-data acquisition framework ensures that future measurements can be seamlessly integrated into the existing EO and modelling workflows, enhancing the robustness and operational readiness of the Digital Twin. These efforts complement the operation of approximately 43–45 soil moisture stations, and all collected datasets were organized for integration into the project database.

8. Tools, Scripts, and Documentation

The Python scripts developed for EO data acquisition, preprocessing, and transformation are provided in an Annex to this deliverable. These scripts document:

- data access methods and APIs,
- preprocessing steps for each dataset,
- spatial harmonisation procedures, and
- export formats used for integration.

The inclusion of scripts ensures transparency, reproducibility, and transferability of the EO workflows and supports future extensions of DT-Agro. Scripts are available in the Annex.

9. Conclusions

Deliverable D3.2 documents the successful implementation of a standardized and operational framework for the acquisition, processing, transformation, and integration of EO and ancillary datasets into the DT-Agro. Building upon the data selection and evaluation presented in Deliverable D3.1, this deliverable focuses on the practical realization of reproducible workflows that transform heterogeneous EO products into harmonised inputs ready for use within the DT-Agro spatial database and modeling environment.

Through the development of automated, script-based workflows, EO datasets covering meteorological conditions (AgERA5), vegetation dynamics (NDVI), land cover and imperviousness (CLMS), soil properties (Greek Soil Map, ISRIC, ESDAC), topography (EU-DEM), and auxiliary datasets (including IACS and in-situ station metadata) were systematically acquired, preprocessed, and standardized. All datasets were reprojected to a common spatial reference system (EPSG:2100), resampled to the target spatial resolutions (100 m for agro-hydrological variables and 1 km for meteorological forcings), and organized in consistent formats (GeoTIFF, CSV) suitable for direct ingestion by DT-Agro.

Emphasis was placed on streamlining workflows for large and temporally extensive datasets. The use of Python-based tools enabled automated downloading via APIs, structured handling of compressed archives, transformation of NetCDF products into analysis-ready formats, and repeatable preprocessing chains. This approach ensures transparency, traceability, and reproducibility of all EO data handling steps, while minimizing manual intervention and reducing the risk of processing errors.

In parallel, evaluation activities were embedded within the data pipeline, including the comparison of AgERA5 meteorological variables against ground-based observations for the year 2023 and the cross-comparison of soil datasets from national and international sources. These assessments informed dataset selection and preprocessing choices and ensured that the EO inputs meet the accuracy and consistency requirements of downstream agro-hydrological modeling.

The processed datasets are integrated into DT-Agro through a centralized spatial database that supports both static and time-varying inputs. The adopted data structures and workflows explicitly support temporal scalability: the same scripts and procedures can be re-executed as new EO observations, updated reanalysis products, or additional in-situ data become available. This design enables DT-Agro to operate with dynamically evolving datasets, supporting historical reprocessing, near-real-time updates, and future scenario-based applications without modification of the core workflows.

Overall, Deliverable D3.2 confirms the establishment of a functional and standardized EO data flow to DT-Agro, fulfilling Milestone M3.1. The implemented pipelines provide a robust technical foundation for the Digital Twin, ensuring that EO data acquisition, processing, and transformation are operational, scalable, and fully aligned with the requirements of subsequent work packages. As the project continues within the framework of the associated doctoral programme, these workflows will be further extended to incorporate additional datasets, longer time series, and climate scenario inputs, strengthening the role of DT-Agro as a dynamic, EO-driven Digital Twin for the Greek agro-hydro-system.

References

Copernicus Land Monitoring Service (2021). Normalised Difference Vegetation Index 2020–present (raster 300 m), global, 10-daily – version 2. European Commission's Joint Research Centre. DOI: 10.2909/ae760a70-708e-459a-8eec-6852462a5faf

Montgomery, J., Hornbuckle, J., Hume, I., & Vleeshouwer, J. (2015). IrriSAT – weather based scheduling and benchmarking technology.

OPEKEPE Greek Payment and Control Agency for Guidance and Guarantee Community Aid. Available online: https://www.opekepe.gr/opekepe-organisation-gr/opekepe-e-services-gr/pliroforiaka-systimata/gewpliroforiako-systimaedafologikwn-dedomenwn (accessed on 08.01.2025).

Orgiazzi, A.; Ballabio, C.; Panagos, P.; Jones, A.; Fernández-Ugalde, O. LUCAS Soil, the largest expandable soil dataset for Europe: A review. Eur. J. Soil Sci. 2018, 69, 140–153.

Panagos, P.; van Liedekerke, M.; Jones, A.; Montanarella, L. European Soil Data Centre: Response to European policy support and public data requirements. Land Use Policy 2012, 29, 329–338.

Annex

[Python]

```
import pandas as pd
import cdsapi
import os
stations = pd.read_csv("stations.csv")
print(f"Loaded {len(stations)} stations.")
client = cdsapi.Client()
output_dir = "sis_daily_data_temperature"
os.makedirs(output_dir, exist_ok=True)
print(f"Output folder: {output_dir}")
variable = "2m_temperature"
years = list(range(1979, 2025))  # 1979-2024
months = [f"{m:02d}" for m in range(1, 13)]
days = [f"{d:02d}" for d in range(1, 32)]

for i, row in stations.iterrows():
    station_id = row["id"]
    lat, lon = row["lat"], row["lon"]
    print(f"\n[{i+1}/{len(stations)}] Processing station {station_id} at ({lat}, {lon})")
    for year in years:
        filename = os.path.join(output_dir, f"{variable}_{year}_{station_id}.nc")
        print(f"    Downloading {filename} ...")
        request = {
            "variable": [variable],
            "statistic": ["24_hour_mean"],
            "year": [str(year)],
            "month": months,
            "day": days,
            "format": "netcdf",
            "area": [lat+0.05, lon-0.05, lat-0.05, lon+0.05],
            "version": "2_0"
        }
        try:

            client.retrieve("sis-agrometeorological-indicators", request, filename)
            print(f"    Finished {variable} for {year}")
        except Exception as e:
            print(f"    ERROR: {variable} for {year}: {e}")
print("\nAll stations processed.")
```

Downloading the parameter of 24h Mean Temperature for 140 stations

[Python]

```
import pandas as pd
import cdsapi
import os
stations = pd.read_csv(r"C:\Users\GISlab\Documents\ΕΛΙΔΕΚ\DT-Agro model\Data\stations.csv")

print(f"Loaded {len(stations)} stations.")
client = cdsapi.Client()
output_dir = "sis_daily_data_precipitation"
os.makedirs(output_dir, exist_ok=True)
print(f"Output folder: {output_dir}")
variable = "precipitation_flux"
years = list(range(1979, 2025))  # 1979-2024
months = [f"{m:02d}" for m in range(1, 13)]
days = [f"{d:02d}" for d in range(1, 32)]
for i, row in stations.iterrows():
    station_id = row["id"]
    lat, lon = row["lat"], row["lon"]
    print(f"\n[{i+1}/{len(stations)}] Processing station {station_id} at ({lat}, {lon})")
    for year in years:
        filename = os.path.join(output_dir, f"{variable}_{year}_{station_id}.nc")
        print(f"    Downloading {filename} ...")
        request = {
            "variable": [precipitation_flux],
            "year": [str(year)],
            "month": months,
            "day": days,
            "format": "zip",
            "area": [lat+0.05, lon-0.05, lat-0.05, lon+0.05],
            "version": "2_0"
        }
        try:
            client.retrieve("sis-agrometeorological-indicators", request, filename)
            print(f"    Finished {variable} for {year}")
        except Exception as e:
            print(f"    ERROR: {variable} for {year}: {e}")
print("\nAll stations processed.")
```

Downloading the parameter of Precipitation Flux for 140 stations

[Python]

```
import pandas as pd
import cdsapi
import os
stations = pd.read_csv(r"C:\Users\GISlab\Documents\ΕΛΙΔΕΚ\DT-Agro model\Data\stations.csv")
  # Columns: id, lat, lon
print(f"Loaded {len(stations)} stations.")
client = cdsapi.Client()
output_dir = "sis_daily_data_ETo"
os.makedirs(output_dir, exist_ok=True)
print(f"Output folder: {output_dir}")
variable = "reference_evapotranspiration"
years = list(range(1979, 2025))  # 1979-2024
months = [f"{m:02d}" for m in range(1, 13)]
days = [f"{d:02d}" for d in range(1, 32)]
for i, row in stations.iterrows():
    station_id = row["id"]
    lat, lon = row["lat"], row["lon"]
    print(f"\n[{i+1}/{len(stations)}] Processing station {station_id} at ({lat}, {lon})")
    for year in years:
        filename = os.path.join(output_dir, f"{variable}_{year}_{station_id}.nc")
        print(f"    Downloading {filename} ...")
        request = {
            "variable": [variable],
            "year": [str(year)],
            "month": months,
            "day": days,
            "format": "zip",
            "area": [lat+0.05, lon-0.05, lat-0.05, lon+0.05],
            "version": "2_0"
        }
        try:
            client.retrieve("sis-agrometeorological-indicators", request, filename)
            print(f"    Finished {variable} for {year}")
        except Exception as e:
            print(f"    ERROR: {variable} for {year}: {e}")
print("\nAll stations processed.")
```

Downloading the parameter of Reference Evapotranspiration ET_o for 140 stations

```
import pandas as pd

import glob
import os
input_folder =
r"C:\Users\GISlab\cds_downloads\2023\2mMeanTemp23\csv_output\converted_xlsx\converted_celsius"
output_file = "monthly_mean_temperature2023.csv"
all_files = glob.glob(os.path.join(input_folder, "*.xlsx"))
all_daily = []
for f in all_files:
    df = pd.read_excel(f)
    df["time"] = pd.to_datetime(df["time"])
    df["year_month"] = df["time"].dt.to_period("M").dt.to_timestamp()
    all_daily.append(df)
daily_all = pd.concat(all_daily, ignore_index=True)
monthly_all = daily_all.groupby(["year_month", "lat",
"lon"])["TempMean24h_Celsius"].mean().reset_index()
monthly_all.to_csv(output_file, index=False)
print("Monthly mean temperature saved to monthly_mean_temperature2023.csv")
```

Aggregating daily temperature data to monthly mean temperature for the year 2023

[Python]

```
import os
import zipfile
zip_folder = r"C:\Users\GISlab\Documents\ΕΛΙΔΕΚ\DT-
Agro model\Data\sis_daily_data_temperature_from16707\zip_files"
output_folder = r"C:\Users\GISlab\Documents\ΕΛΙΔΕΚ\cds"
os.makedirs(output_folder, exist_ok=True)
for file in os.listdir(zip_folder):
    if file.endswith(".zip"):
        zip_path = os.path.join(zip_folder, file)
        folder_name = os.path.splitext(file)[0]
        extract_to = os.path.join(output_folder, folder_name)
        os.makedirs(extract_to, exist_ok=True)
        print(f"Extracting {file} → {extract_to}")
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(extract_to)
print("All ZIP files extracted into individual folders.")
```

Unzip files

[Python]

```
import os
import zipfile
import xarray as xr
import datetime

print('Start:',datetime.datetime.now())

def isleap(year):
    return bool((not year%4 and year%100) or not year%400)

def make_dates(start_year,end_year):
    months=[[3,0,3,2,3,2,3,3,2,3,2,3],[3,1,3,2,3,2,3,3,2,3,2,3]]
    return [datetime.date(y,m,d) for y in range(start_year,end_year+1) for m in range(1,13)
for d in range(1,29+months[isleap(y)][m-1])]

input1=r'C:\Users\GISlab\agera5_rain\unzip'
input2=r''
output_folder=r'C:\Users\GISlab\agera5_rain\csv'
data_name='Precipitation_Flux'
os.makedirs(output_folder,exist_ok=True)

def read_value(ncfile):
    with xr.open_dataset(ncfile) as ds:
        return float(ds[data_name].values[0,0,0])

stations={}

def add_stations(input_folder):
    if not input_folder:
        return
    for i in os.listdir(input_folder):
        folder=os.path.join(input_folder,i)
        code=i[24:29]
        year=int(i[19:23])
        if code not in stations:
            stations[code]={}
        for j in os.listdir(folder):
            month=int(j[45:47])
            day=int(j[47:49])
            date=datetime.date(year,month,day)
            stations[code][date]=os.path.join(folder,j)

add_stations(input1)
add_stations(input2)

dates=make_dates(1979,2024)
dateset=set(dates)

issues={}

for i in stations:
    if set(stations[i].keys())!=dateset:
        issues[i]=[]
        for j in dates:
            if j not in stations[i]:
                issues[i].append(j)

if issues:
    print(f'Found issues with {len(issues)} stations',end='')
    if len(issues)<50:
        print(':\n')
        for i in issues:
            print(i)
    else:
        print('. Check logfile.')
        with open('logfile.txt','w') as file:
            for i in issues:
                file.write(f'{i}\n')

for i in stations:
        with open(os.path.join(output_folder,f'station_rain_{i}.csv'),'w') as csv:
        for j in dates:
            if j.year==1979 and j.month==1 and j.day==1:
                print(f'Entering station {i} on:',datetime.datetime.now())
            if j in stations[i]:
                csv.write(f'{j},{read_value(stations[i][j])}\n')
            else:
                csv.write(f'{j},\n')

print('End:',datetime.datetime.now())
```

NetCDF to .csv

[ArcPy]

```
import arcpy
import os
```

```
input_root = r"C:\Users\GISlab\DT-Agro_Data\_CLCplus_RAS2023\Reproj"
output_root = r"C:\Users\GISlab\DT-Agro_Data\_CLCplus_RAS2023\Resam"
merged_output = r"C:\Users\GISlab\DT-Agro_Data\_CLCplus_RAS2023\Merged_RAS_2023EGSA87.tif"

new_cellsize = "100"

resample_method = "NEAREST"

target_sr = arcpy.SpatialReference(2100)

arcpy.env.overwriteOutput = True
if not os.path.exists(output_root):
    os.makedirs(output_root)

def resample_all(input_folder, output_folder):
    resampled_list = []

    for dirpath, dirnames, filenames in os.walk(input_folder):
        for filename in filenames:
            if filename.lower().endswith(".tif"):
                in_raster = os.path.join(dirpath, filename)


                rel_path = os.path.relpath(dirpath, input_folder)
                out_dir = os.path.join(output_folder, rel_path)
                if not os.path.exists(out_dir):
                    os.makedirs(out_dir)

                out_raster = os.path.join(
                    out_dir, os.path.splitext(filename)[0] + "_100m.tif"
                )

                print(f"Resampling: {in_raster}")
                try:
                    arcpy.management.Resample(
                        in_raster,
                        out_raster,
                        new_cellsize,
                        resample_method
                    )
                    resampled_list.append(out_raster)
                    print(f" Saved: {out_raster}")
                except Exception as e:
                    print(f" Failed: {in_raster}\n   {e}")

    return resampled_list


def merge_rasters(raster_list, output_raster):
    if not raster_list:
        print("No rasters found to merge!")
        return

    print("Merging all resampled rasters...")
    try:
        arcpy.management.MosaicToNewRaster(
            inputs=raster_list,
            output_location=os.path.dirname(output_raster),
            raster_dataset_name_with_extension=os.path.basename(output_raster),
            coordinate_system_for_the_raster=target_sr,
            pixel_type="8_BIT_UNSIGNED",
            cellsize=new_cellsize,
            number_of_bands=1,
            mosaic_method="FIRST"
        )
        print(f"Merged raster saved at: {output_raster}")
    except Exception as e:
        print(f"Merge failed: {e}")


print("Starting resampling and merging...")
resampled_rasters = resample_all(input_root, output_root)
merge_rasters(resampled_rasters, merged_output)
print("Done. All rasters resampled and merged successfully.")
```

Reproject, Resample & Merge rasters

```
import os
import glob
import datetime as dt
import fiona
import rasterio
import rasterio.mask
import zipfile
from shapely.geometry import box
from terracatalogueclient import Catalogue
from terracatalogueclient.config import CatalogueConfig, CatalogueEnvironment
from terracatalogueclient.exceptions import SearchException

def crop(file_path, out_path=None):
    with fiona.open(r"C:/Users/GISlab/Documents/COPERNICUS/_SHAPE/POLYGON_GREECE_geo.shp",
"r") as shapefile:
        shapes = [feature["geometry"] for feature in shapefile]
    with rasterio.open(file_path) as src:
        out_image, out_transform = rasterio.mask.mask(src, shapes, crop=True)
        out_meta = src.meta.copy()
    out_meta.update({
        "driver": "GTiff",
        "height": out_image.shape[1],
        "width": out_image.shape[2],
        "transform": out_transform
    })
    if out_path is None:
        base, ext = os.path.splitext(file_path)
        out_path = f"{base}_cropped{ext}"
    with rasterio.open(out_path, "w", **out_meta) as dest:
        dest.write(out_image)
    return out_path

def extract_zip(zip_path, extract_to):
    try:
        with zipfile.ZipFile(zip_path, "r") as z:
            z.extractall(extract_to)
        return True
    except Exception:
        return False


collections = ["clms_global_ndvi_300m_v2_10daily_geotiff"]

config = CatalogueConfig.from_environment(CatalogueEnvironment.CGLS)
catalogue = Catalogue(config)

start_date = dt.date(2020, 7, 1)
end_date = dt.date(2024, 12, 31)
aoi = box(19.09755, 34.55076, 30.15528, 41.9202)
aoi_wkt = aoi.wkt

OUTPUT_BASE = r"C:/Users/GISlab/Documents/COPERNICUS/downloads"
os.makedirs(OUTPUT_BASE, exist_ok=True)

for collection_id in collections:
    available_ids = {c.id for c in catalogue.get_collections()}
    if collection_id not in available_ids:
        print(f"Collection {collection_id} not found.")
        continue

    try:
        products = list(catalogue.get_products(
            collection_id,
            start=start_date,
            end=end_date,
            geometry=aoi_wkt
        ))
    except SearchException as e:
        print(f"Search failed for {collection_id}: {e}")
        continue

    print(f"{collection_id}: Found {len(products)} products intersecting AOI (test range)")
    if not products:
        continue

    out_dir = os.path.join(OUTPUT_BASE, collection_id)
    os.makedirs(out_dir, exist_ok=True)

    for idx, p in enumerate(products, 1):
        print(f"\n[{idx}/{len(products)}] Requesting product: {p.id}")


        before = set(glob.glob(os.path.join(out_dir, "**"), recursive=True))

        try:
                        catalogue.download_products([p], out_dir, file_types=None, force=True)
        except Exception as e:
            print(f"  Download failed for {p.id}: {e}")
            continue

        after = set(glob.glob(os.path.join(out_dir, "**"), recursive=True))
        new_paths = [p for p in after - before if os.path.isfile(p)]
        if not new_paths:

            product_folder = os.path.join(out_dir, p.id)
            if os.path.isdir(product_folder):
                new_paths = [f for f in glob.glob(os.path.join(product_folder, "**", "*.*"),
recursive=True) if os.path.isfile(f)]

        if not new_paths:
            print("  No new files were detected after download. Check catalogue client output
or network.")
            continue

                extracted_any = False
        all_files = []
        for fp in new_paths:
            all_files.append(fp)
            if fp.lower().endswith(".zip"):
                print(f"  Extracting zip: {os.path.basename(fp)}")
                extract_dir = os.path.splitext(fp)[0]
                ok = extract_zip(fp, extract_dir)
                if ok:
                    extracted_any = True
                    extracted_files = [f for f in glob.glob(os.path.join(extract_dir, "**",
"*.*"), recursive=True) if os.path.isfile(f)]
                    all_files.extend(extracted_files)

        if extracted_any:
            product_folder = os.path.join(out_dir, p.id)
            if os.path.isdir(product_folder):
                all_files += [f for f in glob.glob(os.path.join(product_folder, "**", "*.*"),
recursive=True) if os.path.isfile(f)]
```

```
        tiff_files = [f for f in all_files if f.lower().endswith((".tif", ".tiff"))]
        if not tiff_files:
            print("  No TIFF files found in downloaded package. Found files:")
            for f in sorted(all_files)[:20]:
                print("   ", os.path.basename(f))
            print("  If dataset provides NetCDFs, you may need a different workflow (extract
variable and rasterize).")
            continue

        for tif in tiff_files:
            try:
                print(f"  Cropping TIFF: {os.path.basename(tif)}")
                cropped = crop(tif)
                print(f"    Saved cropped: {os.path.basename(cropped)}")
            except Exception as e:
                print(f"    Crop failed for {tif}: {e}")

    print(f"\nFinished processing collection: {collection_id}")

print("\nDone.")
```

NDVI and other Copernicus Land Monitoring Systems’ data downloading

```
import os
import logging
from pathlib import Path

try:
    from hda import Client, Configuration
except ImportError:
    raise SystemExit("Missing 'hda' package. Install with: pip install hda")

logging.basicConfig(level=logging.INFO, format="%(asctime)s %(levelname)s: %(message)s")

WEKEO_USER = os.getenv("WEKEO_USER")
WEKEO_PASS = os.getenv("WEKEO_PASS")
if not WEKEO_USER or not WEKEO_PASS:
    raise SystemExit("Set WEKEO_USER and WEKEO_PASS environment variables before running the
script.")

OUTPUT_DIR = Path(r"C:\Users\GISlab\DT-Agro_Data\IMPDEN21")
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

conf = Configuration(user=WEKEO_USER, password=WEKEO_PASS)
hda_client = Client(config=conf)
logging.info("HDA client initialized successfully")

search_payload = {
    "dataset_id": "EO:EEA:DAT:HRL:IMP",
    "bbox": [
        18.743528076164104,
        34.67516461419754,
        30.539190418918118,
        42.019934722677085
    ],
    "productType": "Imperviousness Density",
    "resolution": "100m",
    "year": "2021",
    "itemsPerPage": 200,
    "startIndex": 0
}

products = hda_client.search(search_payload)
logging.info("Found %d products.", len(products))
```

```
if not products:
    logging.error("No products found. Check your bounding box / dataset_id / year.")
else:
    for idx, product in enumerate(products, start=1):
        prod_id = getattr(product, "id", None) or getattr(product, "identifier", None) or
str(product)
        logging.info("(%d/%d) Processing product: %s", idx, len(products), prod_id)

        try:
            if hasattr(product, "download") and callable(product.download):
                product.download(str(OUTPUT_DIR))
                logging.info("Downloaded product %s to %s", prod_id, OUTPUT_DIR)
                continue


            if hasattr(hda_client, "download"):
                try:
                    hda_client.download(product, directory=str(OUTPUT_DIR))
                    logging.info("Downloaded product %s via client.download to %s", prod_id,
OUTPUT_DIR)
                    continue
                except Exception:
                    pass

            logging.error("No automatic download method available for product %s.", prod_id)
            logging.info("Product object sample attributes: %s", [a for a in dir(product) if
not a.startswith("_")][:50])
            props = getattr(product, "properties", None)
            if props:
                logging.info("Product properties keys: %s", list(props.keys())[:50])
            else:
                logging.info("No properties attribute on product object; inspect product
representation:")
                logging.info(repr(product))

        except Exception as e:
            logging.error("Download failed for %s: %s", prod_id, e)

logging.info("Script finished.")
```

Imperviousness density downloading

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN
Funded by the European Union NextGenerationEU