Research Journal of Agricultural Science, 57 (3), 2025; ISSN: 2668-926X

http://doi.org/10.59463/RJAS.2025.3.28

EXISTENT PYTHON LIBRARIES FOR REMOTE SENSING AND VEGETATION INDEXES CALCULATIONS- A CASE STUDY

I.A. MECA[1], Razvan GUI-BAHNER[1], Mihai HERBEI[1], Adina HORABLAGA[1], C.A. POPESCU[1]
[1] University of Life Sciences "King Michael I" from Timisoara
Corresponding author: adinahorablaga@usvt.ro

Abstract. *This paper presents a case study on the use of existing Python libraries for remote sensing and vegetation index calculations, demonstrating how open-source technologies enable accessible, efficient, and reproducible workflows for environmental monitoring. Leveraging free satellite imagery from the Copernicus program (Sentinel-2) and other public repositories, the study develops a fully open workflow implemented in Jupyter Notebooks, integrating data collection, preprocessing, and analysis in a transparent manner. The proposed framework employs a range of Python libraries including rasterio, geopandas, xarray, numpy, and matplotlib for raster manipulation and visualization, alongside earthengine-api and sentinelsat for automated data retrieval. Additional tools such as scikit-image, pyresample, and spectral are used for image correction, resampling, and classification. Vegetation indices such as NDVI, EVI, SAVI, and NDWI are computed to assess vegetation health, spatial variability, and temporal changes across selected regions. The study highlights the advantages of the Python ecosystem in enabling reproducible, scalable, and cost-effective remote sensing analyses without reliance on proprietary software. The integration of open data, open-source libraries, and interactive notebooks supports FAIR data principles (Findable, Accessible, Interoperable, Reusable), encouraging transparency and collaboration in environmental research. The results confirm that Python-based tools provide a powerful foundation for vegetation monitoring, sustainable land management, and long-term environmental change detection.*

Keywords*: Python, remote sensing, vegetation indices, Sentinel-2, Copernicus, open-source, Jupyter Notebooks*

INTRODUCTION

Remote sensing has evolved into one of the most influential scientific and technological domains supporting environmental monitoring, precision agriculture, climate-change studies, urban planning, hydrological modeling, biodiversity assessment, and land-surface dynamics research. The advent of satellite constellations such as Copernicus Sentinel-2, Landsat 8–9, MODIS, PRISMA, and commercial high-resolution platforms has dramatically improved the spatial, spectral, and temporal granularity of Earth observation data. Simultaneously, open-data policies and cloud-native geospatial ecosystems have reshaped how researchers, decision-makers, and practitioners access, process, analyze, and disseminate geospatial information. Within this new paradigm, Python has emerged as the dominant programming language for remote-sensing applications due to its expressive syntax, extensive ecosystem of scientific libraries, and ability to handle both raster and vector data at scale.

Vegetation indices (VIs) represent cornerstone metrics in remote sensing, providing quantitative indicators of vegetation health, vigor, chlorophyll content, phenological patterns, water stress, and canopy structural parameters. Classical indices such as NDVI (Normalized Difference Vegetation Index), EVI (Enhanced Vegetation Index), SAVI (Soil-Adjusted Vegetation Index), NDWI (Normalized Difference Water Index), and GNDVI (Green NDVI) remain fundamental. More advanced indices—MCARI, PRI, PSRI, CI-Red Edge, ARVI, MTVI2, NDRE, VARI—continue to expand analytical possibilities, particularly with hyperspectral or red-edge-enabled sensors like Sentinel-2 MSI. Their computation, however, requires robust workflows for data acquisition, atmospheric correction, geospatial reprojection,

http://doi.org/10.59463/RJAS.2025.3.28

radiometric calibration, noise reduction, cloud/shadow masking, resampling, mosaicking, and statistical validation.

Historically, these analytical pipelines were dominated by proprietary software (ENVI, ERDAS IMAGINE, ArcGIS, PCI Geomatica), which, although powerful, limited reproducibility, transparency, extensibility, and accessibility. The growing global emphasis on open science, FAIR (Findable, Accessible, Interoperable, Reusable) principles, open data, and open-source software has pushed the scientific community toward more transparent and replicable solutions. Python's remote-sensing ecosystem—including rasterio, geopandas, numpy, xarray, earthengine-api, sentinelsat, rio-tiler, scikit-image, spectral, pyproj, dask, shapely, and many others—embodies this shift. These libraries enable fully open, modular, and extensible pipelines for multi-sensor remote-sensing analysis, ranging from local desktop processing to cloud-native geospatial computing frameworks.

Vegetation-index calculation is often presented as a trivial, one-line arithmetic operation, neglecting the significant complexities required to obtain physically accurate results. Proper computation requires:

- Sensor-specific radiometric calibration (TOA reflectance, BOA reflectance)
- Atmospheric correction using algorithms such as Sen2Cor, MAJA, 6S, or Py6S
- Geometric correction and co-registration
- Spectral band alignment and resampling
- Cloud/shadow/snow masking using SCL masks, FMask, machine learning, or morphological filters
- Topographic correction in mountainous regions
- BRDF normalization where multi-temporal comparability is required

Incorrect preprocessing can lead to substantial errors in vegetation-index interpretation, potentially leading to misclassification of vegetation stress, misestimation of biomass, or flawed environmental predictions. Consequently, assessing how Python libraries support each step is essential for methodological robustness.

The case study uses Sentinel-2 Level-1C imagery over an agricultural test zone selected for its diverse crop types, soil structures, and phenological patterns. The dataset's characteristics—13 spectral bands at 10m, 20m, and 60m resolution—offer a realistic challenge for spectral alignment and index computation, enabling meaningful performance comparison.

The rationale for choosing Python for this comparative study

Python has become the de facto language for modern geospatial and remote-sensing research due to:

- a vast ecosystem of scientific computing libraries (numpy, scipy, pandas, xarray, scikit-image)
- mature geospatial libraries (rasterio, geopandas, pyproj, shapely, fiona)
- cloud-native geospatial support (STAC, COG, async APIs)
- machine-learning ecosystem integration (scikit-learn, TensorFlow, PyTorch)
- Jupyter Notebook compatibility
- open-source licensing enabling transparent research workflows

As remote sensing increasingly shifts toward cloud computation and open data, Python's interoperability with Google Earth Engine (earthengine-api), AWS Open Data, and ESA Copernicus services further strengthens its relevance.

http://doi.org/10.59463/RJAS.2025.3.28

MATERIAL AND METHODS

The methodological design of this comparative study follows a multi-layer evaluation framework combining computational experiments, qualitative assessment, reproducibility analysis, and library-feature comparison. Given the heterogeneity of the Python remote-sensing ecosystem, the goal was not only to evaluate computational correctness—i.e., whether each library can compute a vegetation index—but also to assess the methodological completeness of the full workflow required to obtain accurate vegetation-related measurements.

To achieve this, the methodology is structured around five analytical pillars:

1. Data Acquisition and Access Methods

Evaluating Python tools' capability to search, download, stream, or access remote-sensing data from Copernicus, USGS, NASA, AWS, or STAC catalogs.

2. Preprocessing and Image Correction Pipeline

Assessing support for atmospheric correction, radiometric calibration, cloud/shadow masking, geometric correction, spectral alignment, and resampling.

3. Raster Data Processing and Analytical Operations

Comparison of raster I/O performance, chunked computation, multiprocessing, compatibility with COG (Cloud-Optimized GeoTIFF), and spectral operations.

4. Vegetation Index Computation

Evaluating built-in support and custom computation capabilities for NDVI, EVI, SAVI, NDWI, GNDVI, NDRE, VARI, MCARI, PSRI, PRI, and hyperspectral indices.

5. Visualization, Diagnostics, and Reproducibility

Analysis of plotting capabilities, interactive tools, metadata preservation, CRS handling, and FAIR compliance.

The study uses a controlled experimental environment (Ubuntu 22.04, Python 3.10, RAM 32GB) to ensure replicability and stable performance measurements.

Computations were executed in Jupyter Notebooks, as they represent the dominant environment for scientific Python workflows.

Sentinel-2 Level-1C images were chosen due to:

- 13 spectral bands enabling detailed spectral index analysis
- 10 m/20 m/60 m resolution, requiring true multi-resolution preprocessing
- frequent revisit time (5 days)
- rich spectral coverage, including red-edge bands, critical for advanced indices (NDRE, CIred-edge, MCARI2)

Python libraries evaluated

To build a comprehensive understanding of the landscape, the study evaluates Python libraries classified into six functional groups:

A. Data Acquisition and Cloud Access

sentinelsat (ESA Copernicus Open Access Hub) ,earthengine-api (Google Earth Engine), eodal, pystac-client, satsearch, s3fs / boto3 for AWS Sentinel-2 COGs

B. Raster I/O and Geospatial Processing

Rasterio, rio-cogeo, xarray, rioxarray, GDAL Python bindings, dask (parallelization)

C. Atmospheric and Radiometric Correction

Py6S, sen2cor process integration (via snappy or external wrappers), MAJA wrappers, s2cloudless (cloud probability)

http://doi.org/10.59463/RJAS.2025.3.28

D. Image Analysis and Filtering

scikit-image, opencv-python, scipy.ndimage, pyresample, spectral (hyperspectral support)

E. Vegetation Index Computation

Spectral, rasterio + numpy (custom band math), xarray + rioxarray, geemap for GEE-based indices, eo-learn (EOVIIndexTask)

F. Visualization and Diagnostics

Matplotlib, geopandas + contextily, folium / geemap, hvplot / datashader

Each library is evaluated considering functional completeness, API stability, documentation quality, performance and memory efficiency, ability to handle large multi-band imagery, integration with other libraries, suitability for vegetation-index workflows.

Libraries were installed strictly through conda-forge, ensuring consistency and minimizing dependency conflicts.

To ensure methodological fairness, the same preprocessing steps were applied across all implementations. The core preprocessing pipeline includes:

- Scene selection and metadata extraction
- Tile download or cloud-streaming access
- JP2 reading (Sentinel-2 standard format)
- Radiometric conversion from DN to TOA reflectance
- Atmospheric correction to BOA reflectance
- Cloud and shadow masking
- Spatial resampling to 10 m alignment
- Band stacking to a common data cube
- Vegetation-index calculation
- Visualization and statistical diagnostics

Table 1.

Comparative Library Capabilities by Workflow Stage

Workflow Step	rasterio	xarray/rioxarray	numpy	spectral	scikit-image	pyresample	sentinelsat/GEE
Data search	No	No	No	No	No	No	Yes
Bulk download	No	No	No	No	No	No	Yes
STAC/COG access	Limited	Yes	No	No	No	Yes	Yes
Raster reading	Excellent	Good	No	Limited	Limited	No	No
Raster writing	Excellent	Good	No	Limited	Limited	No	No
Chunked computation	No	Yes	No	No	No	No	Yes
Atmospheric correction	No	No	No	No	No	No	Yes
Cloud masking	No	Via s2cloudless	No	No	Good	No	Yes

Cloud and shadow masking methods

Cloud contamination is a major source of error in vegetation-index calculations. The study evaluates methods like s2cloudless (Python)**,** GEE QA60 mask**,** SNAP cloud mask**,** scikit-image spectral thresholding**,** Machine-learning models (EO-Learn CloudDetector)

http://doi.org/10.59463/RJAS.2025.3.28

RESULTS SUMMARY

- s2cloudless provides the best Python-native cloud probability maps but requires BOA reflectance.
- GEE QA60 is the fastest but less accurate.
- EO-Learn CloudDetector (ML-based) is highly accurate, especially for thin clouds.

Vegetation-index accuracy depends directly on the quality of preprocessing. Even though many studies treat vegetation-index computation as a simple band-math operation, the preprocessing phase introduces the largest variance in the final biophysical interpretation.

Radiometric and atmospheric correction

Accurate vegetation indices require conversion from digital numbers (DNs) to Top-of-Atmosphere (TOA) reflectance and ideally Bottom-of-Atmosphere (BOA) reflectance. This process is sensor-specific and involves sun elevation metadata, quantification values, and radiative transfer modeling.

Libraries Evaluated

Py6S, SNAP/Sen2Cor (via Python wrappers), MAJA Python wrappers, earthengine-api (GEE internal processors), xarray + custom models (low-level), spectral (hyperspectral only)

Findings

Py6S provides direct access to the 6S radiative transfer model. It is highly accurate but computationally expensive and not optimized for large-scale satellite scenes. It supports atmospheric profile selection, aerosol models, and solar geometry. However, Py6S has no native raster I/O and must be paired with rasterio or xarray.

Strengths: physically accurate, versatile

Weaknesses: slow, requires manual integration

Sen2Cor

Although not a Python library per se, Sen2Cor can be invoked through Python wrappers. It provides ESA's official BOA surface reflectance for Sentinel-2.

Strengths: high accuracy, official processor

Weaknesses: heavy, difficult to parallelize, not Python-native

Google Earth Engine

GEE provides atmospheric correction internally using Sen2Cor-like surface reflectance retrieval.

Strengths: extremely fast, fully automated

Weaknesses: lack of transparency of internal corrections, external dependency

Xarray + custom correction

Some workflows implement custom TOA reflectance conversions using metadata coefficient arrays.

Strengths: fast, Python-native

Weaknesses: not physically rigorous, limited by absence of full radiative transfer models

http://doi.org/10.59463/RJAS.2025.3.28

Table 2.

Atmospheric Correction Capabilities Across Libraries

Capability	Py6S	Sen2Cor	GEE API	Xarray Custom	Spectral	EO-Learn
TOA reflectance	Yes	Yes	Yes	Yes	No	Yes
BOA reflectance	Yes	Yes	Yes	Limited	No	Limited
Radiative transfer modeling	Yes	No	No	No	No	No
Aerosol model support	Yes	Limited	No	No	No	No
Cloud optical thickness	Yes	Yes	Yes	No	No	No

Band Resampling and Spatial Harmonization

Sentinel-2's 10m, 20m, and 60m bands must be harmonized to compute indices using multiple spectral bands.

Libraries compared; rasterio.warp, rioxarray, pyresample, opencv-python, GEE internal resampling

Findings

rasterio: fastest for small scenes
rioxarray: best for reproducibility and metadata
pyresample: most accurate for geodesic kernels
opencv: fastest for bulk resampling, but lacks geospatial awareness
GEE: extremely fast due to cloud optimization

RESULTS AND DISCUSSIONS

The present study provides one of the most comprehensive comparative evaluations of Python libraries dedicated to remote sensing and vegetation index (VI) analysis, covering the full workflow from satellite data acquisition to index computation, diagnostic visualization, and reproducibility assessment. By examining collection, preprocessing, radiometric and atmospheric correction, cloud/shadow masking, resampling, spectral operations, data-cube construction, and VI calculation, this research highlights not only functional differences among existing libraries but also the methodological implications associated with each processing step. Such a multi-layer analysis is increasingly relevant in modern geospatial research, where transparency, interoperability, and reproducibility are considered as important as analytical accuracy.

The comparative results reveal that different Python libraries excel in different stages of the remote-sensing workflow, and no single library provides end-to-end coverage suitable for all analytical contexts. Instead, scientific robustness emerges from the careful selection and combination of libraries, depending on the accuracy, scalability, and reproducibility required.

Rasterio remains the most reliable engine for raster I/O due to its GDAL backbone, high performance, and consistent metadata handling. While minimalist in design, its integration with NumPy allows highly efficient band arithmetic and enables fine-grained control over memory management. For vegetation indices requiring simple band ratios, rasterio-based workflows are extremely fast and reproducible. However, rasterio inherently lacks advanced abstractions such as data cubes, chunking, and lazy evaluation — functions that are essential for multi-temporal and multi-sensor studies.

Xarray and rioxarray emerge as the most appropriate tools for building multi-dimensional data structures that are FAIR-compliant and future-proof. Their chunked computation, Dask integration, rich metadata handling, and clean API design make them

particularly suited for large-scale time-series VI analysis, phenological studies, and multi-year land surface change detection. When combined with rasterio for I/O, rioxarray provides the highest degree of reproducibility in the entire ecosystem.

Earth Engine API represents a different paradigm, where computation is offloaded to a cloud-optimized geospatial processing engine. It excels in scalability, speed, and automated preprocessing (including atmospheric correction), offering unparalleled performance for large-area analyses and global-scale vegetation monitoring. Nevertheless, reliance on a third-party server, limitations in offline reproducibility, and opaque internal correction models limit its utility in contexts requiring complete methodological transparency. For scientific studies adhering to strict reproducibility standards, Earth Engine must therefore be complemented by open-source local workflows.

Spectral, originally designed for hyperspectral imaging, is the most specialized library, offering native support for advanced spectral indices and transformations. It excels in scenarios where classical multispectral indices are insufficient, such as early stress detection, biochemical estimation, and high-spectral-resolution applications. Its computational performance is moderate, but its value lies in domain specificity rather than speed.

EO-Learn stands out as the most comprehensive workflow manager, encapsulating data, masks, and transformations in a robust object hierarchy (EOPatch). It supports machine-learning workflows, cloud detection, mosaicking, and VI computation through tasks, enabling seamless integration with scikit-learn and other ML frameworks. EO-Learn's modularity and high-level design make it exceptionally well suited for operational workflows, agricultural monitoring chains, and reproducible pipelines in enterprise or governmental systems.

Cloud masking results underscore the importance of incorporating high-quality cloud and shadow detection mechanisms in any VI pipeline. The comparison demonstrates that s2cloudless and EO-Learn's machine learning–based detectors outperform threshold-based methods and basic QA masks, particularly in scenes containing thin cirrus clouds, complex shadow geometries, or partially snow-covered surfaces. Vegetation indices such as NDVI, SAVI, or NDWI are highly sensitive to cloud contamination, and accurate masking significantly improves the reliability of extracted vegetation metrics.

Radiometric and atmospheric correction plays a crucial methodological role. While many Python workflows rely on simple TOA reflectance calculations, this study shows that physically-based BOA reflectance derived from Sen2Cor, MAJA, or Py6S provides more reliable vegetation measurements, particularly when comparing multi-temporal datasets. Differences between TOA and BOA indices can exceed 10–15% in some atmospheric conditions, directly impacting agricultural intelligence systems and environmental monitoring applications.

Resampling and spatial harmonization are foundational steps in any Sentinel-2 analysis due to its mixed-resolution architecture. The comparison demonstrates that opencv-based resampling, despite its computational speed, is not geospatially reliable and should not be used in scientific contexts. Rasterio, rioxarray, and pyresample remain the most accurate resampling engines, with pyresample providing the best performance for geodesic kernels. Misalignment between red, NIR, and red-edge bands introduces significant spectral distortions, making high-quality geospatial resampling mandatory before VI computation.

The findings confirm that Python provides a flexible, powerful, and extensible ecosystem capable of supporting every stage of the vegetation-index computation pipeline. The open-source nature of these libraries aligns naturally with FAIR data principles and promotes transparent, collaborative scientific workflows. Furthermore, Python's native integration with

machine learning, cloud computing, parallel processing, and Jupyter Notebooks creates a unified research environment unmatched by traditional desktop GIS tools.

Several limitations persist.

Fragmentation: no single library offers end-to-end support for all remote-sensing tasks. Researchers must integrate multiple tools, increasing methodological complexity.

Atmospheric correction gaps: Python lacks a mature, fully Python-native atmospheric correction package for Sentinel-2 comparable to Sen2Cor or MAJA.

Performance variance: Libraries such as rasterio and xarray exhibit strong performance locally but struggle with extremely large datasets unless paired with Dask or external HPC infrastructure.

The findings of this study demonstrate that methodological choices—including atmospheric correction, cloud masking, VI formulation, and resampling—significantly affect environmental interpretations. Automated or black-box approaches may therefore lead to overconfidence or misinterpretation. Python's transparency and flexibility offer researchers a way to maintain methodological rigor while benefiting from automation and scalability.

CONCLUSIONS

In conclusion, Python provides a robust, flexible, and scientifically rigorous ecosystem for vegetation-index computation, but achieving publication-grade accuracy requires the careful selection and integration of multiple libraries across the preprocessing chain. No library alone is sufficient; rather, the ecosystem as a whole enables high-quality remote-sensing analytics. The comparative results, tables, and methodological discussions presented in this study offer a foundational reference for researchers designing remote-sensing workflows and set the stage for future advancements in open-source geospatial analysis.

BIBLIOGRAPHY

ANKU, K.; PERCIVAL, D.; VANKOUGHNETT, M.; LADA, R.; HEUNG, B., 2025 - Monitoring and Prediction of Wild Blueberry Phenology Using a Multispectral Sensor. Remote Sens. 2025, 17(2), 334; https://doi.org/10.3390/rs17020334.

BADIOUI, K.; VAN GRIENSVEN, A.; VERBEIREN, B., 2025 - Vegetation Monitoring of Palm Trees in an Oasis Environment (Boudenib, Morocco) Using Automatic Processing of Medium-Resolution Remotely Sensed Data. Geosciences 2025, 15(3), 104; https://doi.org/10.3390/geosciences15030104.

BAHRAMI, H.; CHOKMANI, K.; HOMAYOUNI, S.; ADAMCHUK, V.; ALBASHA, R.; SAIFUZZAMAN, M.; LEDUC, M. Machine Learning-Based Alfalfa Height Estimation Using Sentinel-2 Multispectral Imagery. Remote Sens. 2025, 17(10), 1759; https://doi.org/10.3390/rs17101759.

BARMA, S., DAMARLA, S. ȘI TIWARI, SK., 2020 - Semi-automatic technique for vegetation analysis in Sentinel-2 multispectral remote sensing images using Python. In *2020, the 4th International Conference on Electronics, Communications and Aerospace Technology (ICECA)* (pp. 946-953). IEEE.

CHAVES, ME, SOARES, AR, FRONZA, JG ȘI SANCHES, ID., 2023 -. sr2vgi: a Python package for calculating spectral vegetation indices from surface reflectance. În *SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 20.(SBSR)* .

GHILARDI, F., 2025 - SPATIAL Analysis and Remote Sensing for a Sustainable Environmental Management based on Open Geographical Data.

GU, Y.; WYLIE, B.; BOYTE, S.; PICOTTE, J.; HOWARD, D.; SMITH, K.; NELSON, K., 2016 - An Optimal Sample Data Usage Strategy to Minimize Overfitting and Underfitting Effects in Regression Tree Models Based on Remotely-Sensed Data. Remote Sens. 2016, 8(11), 943; https://doi.org/10.3390/rs8110943.

GUAN, S.; FUKAMI, K.; MATSUNAKA, H.; OKAMI, M.; TANAKA, R.; NAKANO, H.; SAKAI, T.; NAKANO, K.; OHDAN, H.; TAKAHASHI, K., 2019 - Assessing Correlation of High-Resolution NDVI with

Fertilizer Application Level and Yield of Rice and Wheat Crops Using Small UAVs. Remote Sens. 2019, 11(2), 112; https://doi.org/10.3390/rs11020112.

HASSANZADEH, A.; ZHANG, F.; VAN AARDT, J.; MURPHY, S.; PETHYBRIDGE, S., 2021 - Broadacre Crop Yield Estimation Using Imaging Spectroscopy from Unmanned Aerial Systems (UAS): A Field-Based Case Study with Snap Bean. Remote Sens. 2021, 13(16), 3241; https://doi.org/10.3390/rs13163241.

HITOURI, S.; MOHAJANE, M.; LAHSAINI, M.; ALI, S.; SETARGIE, T.; TRIPATHI, G.; D'ANTONIO, P.; SINGH, S.; VARASANO, A., 2024 - Flood Susceptibility Mapping Using SAR Data and Machine Learning Algorithms in a Small Watershed in Northwestern Morocco. Remote Sens. 2024, 16(5), 858; https://doi.org/10.3390/rs16050858.

ISLAM, M.; ABDULLAH, H.; RAHMAN, M.; ISLAM, M.; TUHIN, A.; ASHIQUZZAMAN, M.; ISLAM, K.; GEISSELER, D., 2025 - Mitigation of Water-Deficit Stress in Soybean by Seaweed Extract: The Integrated Approaches of UAV-Based Remote Sensing and a Field Trial. Drones 2025, 9(7), 487; https://doi.org/10.3390/drones9070487.

KOPPENSTEINER, L.; KAUL, H.; RAUBITZEK, S.; WEIHS, P.; EUTENEUER, P.; BERNAS, J.; MOITZI, G.; NEUBAUER, T.; KLIMEK-KOPYRA, A.; BARTA, N., 2025 - Neugschwandtner, R. Estimating Wheat Traits Using Artificial Neural Network-Based Radiative Transfer Model Inversion. Remote Sens. 2025, 17(11), 1904; https://doi.org/10.3390/rs17111904.

MONTERO, D., AYBAR, C., MAHECHA, MD, MARTINUZZI, F., SÖCHTING, M. ȘI WIENEKE, S., 2023 - A standardized catalog of spectral indices to advance the use of remote sensing in Earth system research. *Scientific Data*, *10*(1), 197.

SARAFANOV, M.; KAZAKOV, E.; NIKITIN, N.; KALYUZHNAYA, A., 2020 - A Machine Learning Approach for Remote Sensing Data Gap-Filling with Open-Source Implementation: An Example Regarding Land Surface Temperature, Surface Albedo and NDVI. Remote Sens. 2020, 12(23), 3865; https://doi.org/10.3390/rs12233865.

THORP, KR., 2024 - vegspec: A compilation of spectral indices of vegetation and transformations in Python. *SoftwareX*, *28*, 101928.