

Politecnico di Milano

An Accessible Framework for Burned Area Classification

Geoinformatics Project

Matteo Gobbi Frattini
[Data]

Sommario

1. Introduction	3
Project Background	3
Motivation	3
2. Related Work	4
Traditional Approaches: NBR and dNBR	4
Alternative Approaches: Machine Learning for Burned Area Detection	5
3. Project Development	6
Overview of the Tool	6
Implementation	7
Image Loading (image_loader.py)	7
Preprocessing (preprocessing.py)	7
Model Training and Classification (processing.py)	8
Graphical User Interface (app.py)	9
Used Tools	9
Programming Languages and Frameworks	9
Geospatial & Image Processing Libraries	9
Executable Packaging	10
4. User Workflow	10
Main Menu (BurnAreaClassifierApp)	10
Training a New Model (TrainModelUI)	11
Overview	11
Satellite Selection	12
Image Selection and Preprocessing Panel	12
Model Settings Panel	14
Execution and Navigation Controls	15
Validation Results	15
Classifying Using an Existing Model (UseExistingModelUI)	16
Overview	16
Satellite Selection	17
Model Selection	17
After Fire Image Selection and Preprocessing Options	18

Execution and Navigation Controls	19
Testing a Model's Performance (TestModelPerformanceUI)	19
Overview.....	19
Satellite Selection	20
Image Selection and Preprocessing Panel	21
Model Selection & Settings.....	21
Execution and Classification Settings	21
Testing Results	22
5. Example Case Studies	22
Introduction	22
Training a New Model on the Alexandroupoli Fire Dataset.....	23
Step 1: Data Procurement	23
Step 2: Satellite Selection	23
Step 3: Loading the Images.....	23
Step 4: Previewing the Images	24
Step 5: Setting Preprocessing Options	26
Step 6: Defining Model Training Parameters	26
Step 6: Output and Display Options	27
Step 7: Running the Training and Classification	28
Classifying the Rhodes Fire Using the Trained Model.....	31
Step 1: Data Procurement	31
Step 2: Satellite Selection	32
Step 3: Loading the Model	32
Step 4: Loading the Image	33
Step 5: Previewing the Image	34
Step 6: Setting Preprocessing Options	35
Step 7: Output Options	35
Step 8: Running the Classification	36
Testing the Model's Performance on the Rhodes Fire	37
Step 1: Data Procurement	37
Step 2: Satellite Selection	38
Step 3: Loading the Images.....	38

Step 4: Preprocessing Options.....	38
Step 5: Model Loading and Validation Settings	39
Step 6: Classification Settings	39
Step 7: Execution.....	40
6. Conclusions.....	41
Future Considerations.....	42

1. Introduction

Project Background

This project is about creating a practical and user-friendly tool that helps users train and apply machine learning models for burned area detection. Burned areas are typically identified using indices like the Differenced Normalized Burn Ratio (dNBR), which requires both pre-fire and post-fire images. However, obtaining a pre-fire image can be challenging due to cloud cover or data availability, and processing large satellite datasets can be time-consuming.

To address these issues, I developed a graphical user interface (GUI) that simplifies the process of preparing satellite images, generating labeled datasets, training models, and applying them for classification. Support Vector Machines (SVM) were chosen as the classification method to provide a balance between accuracy and computational efficiency. The tool is designed for users who need an intuitive way to work with remote sensing data and machine learning without writing code.

Motivation

This tool was developed to provide a structured and accessible way for users to apply Support Vector Machines (SVM) to burned area classification. While traditional methods rely on dNBR-based thresholding, machine learning offers a more flexible approach by allowing users to train and apply models based on their specific datasets.

The tool allows users to:

- Train and apply machine learning models for burned area classification in an intuitive way.
- Use different workflows, including training new models with pre- and post-fire images or applying existing models using only post-fire images.
- Easily preprocess images, with built-in tools for cloud and water masking.
- Automate training dataset creation, using dNBR-based labeling to generate samples.

- Evaluate model performance, providing precision, recall, and confusion matrices to help users assess classification quality.

The goal is to make machine learning more practical for wildfire analysis by reducing the need for complex programming and manual data processing. With this in mind, I aimed to make the tool as general and adaptable as possible. It supports both Sentinel-2 and Landsat 8 imagery, allowing users to work with different datasets depending on availability and preference. Additionally, the tool accommodates various levels of preprocessing. Users can start from raw satellite images, already masked images, or even a folder containing individual band files, which the software can stack and preprocess automatically. Rather than replacing existing methods, this tool serves as an accessible way for users to integrate machine learning into their workflow with minimal technical barriers.

2. Related Work

Traditional Approaches: NBR and dNBR

Burned area detection is commonly performed using spectral indices, particularly the Normalized Burn Ratio (NBR) and its derivative, the Differenced Normalized Burn Ratio (dNBR). These indices take advantage of the spectral properties of vegetation and burned areas in the near-infrared (NIR) and shortwave infrared (SWIR) bands of satellite imagery.

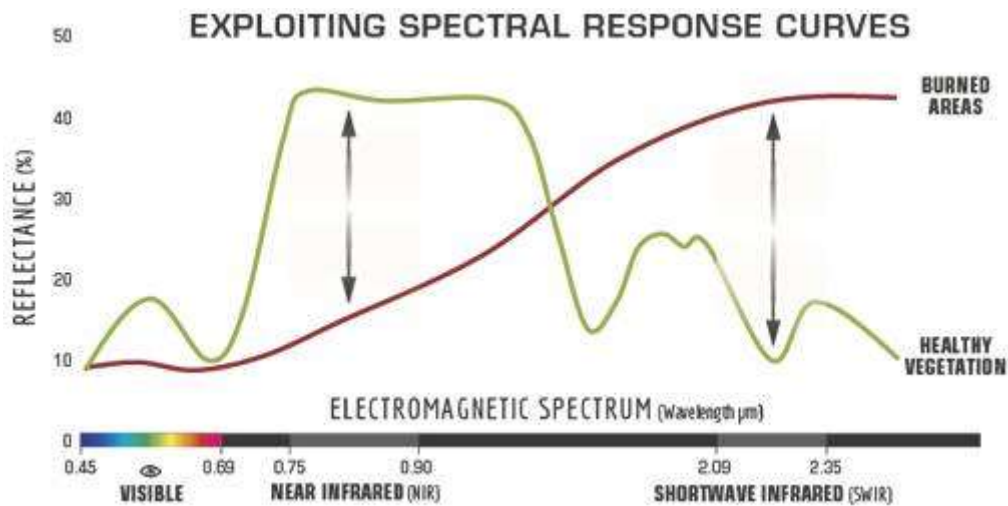
The NBR is calculated as:

$$NBR = \frac{NIR - SWIR}{NIR + SWIR}$$

Where:

- NIR (Near-Infrared Band) reflects healthy vegetation strongly.
- SWIR (Shortwave Infrared Band) absorbs more energy in burned areas due to charred surfaces and reduced vegetation.

Higher NBR values indicate healthy vegetation, while lower values correspond to burned areas, water, or bare soil.



The dNBR is used to measure fire severity by comparing pre- and post-fire NBR values:

$$dNBR = NBR_{Pre} - NBR_{Post}$$

A larger dNBR value indicates a greater change in surface conditions, typically due to burning. dNBR values can vary from case to case, and so, if possible, interpretation in specific instances should also be carried out through field assessment. However, the United States Geological Survey (USGS) proposed a classification table to interpret the burn severity, which can be seen below:

Severity Level	dNBR Range (scaled by 10^3)	dNBR Range (not scaled)
Enhanced Regrowth, high (post-fire)	-500 to -251	-0.500 to -0.251
Enhanced Regrowth, low (post-fire)	-250 to -101	-0.250 to -0.101
Unburned	-100 to +99	-0.100 to +0.99
Low Severity	+100 to +269	+0.100 to +0.269
Moderate-low Severity	+270 to +439	+0.270 to +0.439
Moderate-high Severity	+440 to +659	+0.440 to +0.659
High Severity	+660 to +1300	+0.660 to +1.300

The dNBR method is widely used because it is straightforward and interpretable. However, it depends on the availability of high-quality pre-fire imagery and can be affected by seasonal variations in vegetation.

Alternative Approaches: Machine Learning for Burned Area Detection

Machine learning provides an alternative approach by using labeled datasets to classify burned areas rather than relying solely on spectral index thresholds. Instead of defining fixed dNBR cutoff values, models learn patterns in the data, making them adaptable to different fire conditions and vegetation types.

While various machine learning algorithms have been explored for burned area classification, this project focuses on integrating SVM into a user-friendly framework rather than evaluating the accuracy of different models. The tool enables users to train models on their own datasets

and apply them efficiently, offering flexibility while keeping computational requirements manageable.

3. Project Development

Overview of the Tool

The software is a graphical user interface (GUI) tool designed to help users classify burned areas using Support Vector Machines (SVM). It brings together image preprocessing, model training, classification, and evaluation in an accessible and structured way.

It is designed to be user-friendly and accessible, packaged as a standalone executable using PyInstaller. This will allow users to run the application without needing to install Python or any dependencies manually.

Key features of the tool include:

- Support for Sentinel-2 and Landsat 8 imagery, giving users flexibility in choosing datasets based on availability and resolution.
- Multiple preprocessing options, including cloud and water masking, band stacking, and NBR computation.
- Automated training sample generation, using dNBR thresholds or extreme values to label burned and unburned pixels.
- Model training and classification using SVM, with options to either train a new model or apply an existing one.
- Performance evaluation tools, including precision, recall, and confusion matrices.
- A user-friendly GUI, built with Tkinter, making the workflow intuitive and easy to navigate.

For preprocessing, the tool integrates GeoPre, a geospatial processing library that I developed during the Geospatial Processing course. GeoPre includes functions for key remote sensing tasks such as stacking bands and masking clouds and water, making data preparation more efficient and standardized.

Additionally, the software was designed with generalization and flexibility in mind. It supports different no-data values, allowing users to work with datasets from various sources without worrying about inconsistencies in missing data representation. The tool also provides adjustable classification parameters, making it adaptable to different fire conditions and sensor specifications.

Implementation

The software is structured into four main components: image loading, preprocessing, model training & classification, and the graphical user interface (GUI). Each module is designed to handle a specific part of the workflow.

Image Loading (image_loader.py)

This module is responsible for handling raw imagery that is stored as separate raster files (one per spectral band) in a folder. It acts as a bridge between the software and the `stack_bands` function of GeoPre, ensuring that the correct bands are selected and merged into a single stacked image that is ready for further processing.

Since the tool is designed to support multiple satellite sources, including Sentinel-2 and Landsat 8/9, this module ensures consistency across different datasets by selecting the appropriate bands and preparing a standardized stacked image.

Key Functions:

- `load_image(folder_path, satellite)`: It determines which bands are required based on the provided satellite type and calls `stack_bands` from GeoPre, which merges the selected bands into a single .tif file with the appropriate parameters. The resulting stacked image is then returned as an output. The function is designed to be adaptable, handling different no-data values and file naming conventions across various datasets.

Preprocessing (preprocessing.py)

This module is responsible for preparing the stacked satellite imagery for analysis by applying essential preprocessing steps. It ensures that the input data is clean, properly formatted, and ready for feature extraction and model training. The preprocessing module handles common remote sensing challenges, such as cloud contamination and water masking, and computes spectral indices that are crucial for burned area classification.

Preprocessing is a critical step to improve the accuracy and reliability of the classification results. The module integrates functionalities from GeoPre, which provides efficient implementations for cloud masking and water masking.

Key Functions:

- `mask_clouds_any(source, output_path, satellite, method, mask_shadows, nodata=np.nan)`: This function calls the appropriate GeoPre function for cloud masking based on the selected satellite and method. It allows the user to choose a cloud masking method and whether to also mask cloud shadows
- `mask_water(image_path, output_path = None, ndwi_threshold = 0.01, nodata = np.nan)`: This function detects and masks water areas using the Normalized Difference Water Index (NDWI). It has two versions: one for Landsat and

one for Sentinel-2. The NDWI threshold is set by the user to determine which pixels are classified as water.

- `apply_masks(image_path, satellite, final_output_path = None, method='auto', mask_clouds=True, mask_shadows=True, mask_water=True, ndwi_threshold=0.01, nodata=None)`: This function applies both water and cloud masking in sequence. If `mask_water=True`, it first removes water pixels. If `mask_clouds=True`, it then applies cloud masking. The function ensures that a consistent nodata value is applied throughout the image.
- `compute_NBR(source, satellite, nodata=None)`: This function computes the Normalized Burn Ratio (NBR) using Near-Infrared (NIR) and Shortwave Infrared (SWIR) bands, ensuring that the correct bands are used based on the satellite type.

Additionally, the preprocessing module contains helper functions for detecting nodata values (`detect_nodata_from_path`, `detect_nodata`) and filtering spectral bands to ensure correct band selection across different satellite datasets.

Model Training and Classification (`processing.py`)

This module is responsible for training the Support Vector Machine (SVM) model, generating training data, applying the model for classification, and evaluating its performance. It plays a central role in enabling the tool to move from raw satellite imagery to classified burned area maps.

The training process relies on dNBR-based labeling, where burned and unburned areas are determined using pre-defined thresholds. The trained SVM model can then be applied to classify post-fire images, even without a pre-fire reference.

Key functions:

- `compute_dNBR(pre_NBR, post_NBR)`: Computes the Differenced Normalized Burn Ratio (dNBR) by subtracting the post-fire NBR from the pre-fire NBR. This metric is used to label training samples for model training.
- `create_training_set(post_image_path, dNBR, method='threshold', threshold=300, ext_burned_threshold=500, ext_unburned_threshold=50, samples=5000, post_image_nodata=None)`: Generates labeled training data by extracting pixel values from the post-fire image and assigning labels based on dNBR thresholds.
- `train_and_evaluate_svm(X_sampled, y_sampled, test_size=0.25, kernel="linear", random_state=42)`: Trains an SVM classifier using the extracted training dataset and evaluates its performance. The function supports user-defined kernel choices and test/train split ratios.
- `classify(model, post_image_path, output_path=None, nodata_value=None)`: applies a trained SVM model to classify a post-fire image, saving a burned area classification map.

- `test_model(model, X_test, y_test)`: Tests an existing model on a validation dataset and computes accuracy metrics, including precision and recall.
- `save_model(model, model_path)` and `load_model(model_path)`: Saves trained models for later use and reloads them when needed, enabling users to apply previously trained models without retraining from scratch.

Graphical User Interface (app.py)

The graphical user interface (GUI) provides users with an intuitive way to interact with the tool, eliminating the need for manual scripting. Built with Tkinter, it enables users to seamlessly manage data loading, preprocessing, model training, classification, and evaluation through a structured workflow.

The GUI is structured into multiple Tkinter-based classes, including:

- `BurnAreaClassifierApp` – Main menu, allowing users to choose between training a new model, using an existing model, or testing a model's performance.
- `TrainModelUI` – Handles loading pre- and post-fire images, preprocessing options, and model training settings.
- `UseExistingModelUI` – Enables users to apply a pre-trained model to a post-fire image for classification.
- `TestModelPerformanceUI` – Facilitates evaluation of an existing model by comparing predictions to dNBR-based ground truth.

The GUI takes care of all user interactions, from selecting images through a file browser to displaying the classified map preview. It provides a structured workflow that simplifies the process of loading satellite images, preprocessing them, training models, applying trained models, and visualizing classification results.

The interface ensures that users can seamlessly transition between steps without needing to interact with the underlying code. Through intuitive menus and dialogs, users can configure preprocessing settings, select training parameters, and evaluate model performance with ease.

Used Tools

The development of this software relies on a set of geospatial processing, machine learning, and graphical interface libraries. It is packaged as a standalone executable using PyInstaller, so users can run the application without installing Python or additional dependencies.

Programming Languages and Frameworks

- **Python** – The core language used for development
- **Tkinter** – Used to build the graphical interface

Geospatial & Image Processing Libraries

- **Rasterio** – For reading, writing, and manipulating raster images.

- **GDAL** – Used for handling geospatial data and performing transformations.
- **NumPy** – Provides efficient numerical operations for satellite image processing.
- **scikit-learn** – Used for machine learning functionalities, specifically Support Vector Machines (SVM).
- **GeoPre** – A geospatial processing library I developed in a Geospatial Processing course, containing functions for stacking bands, masking clouds, and removing water.

Executable Packaging

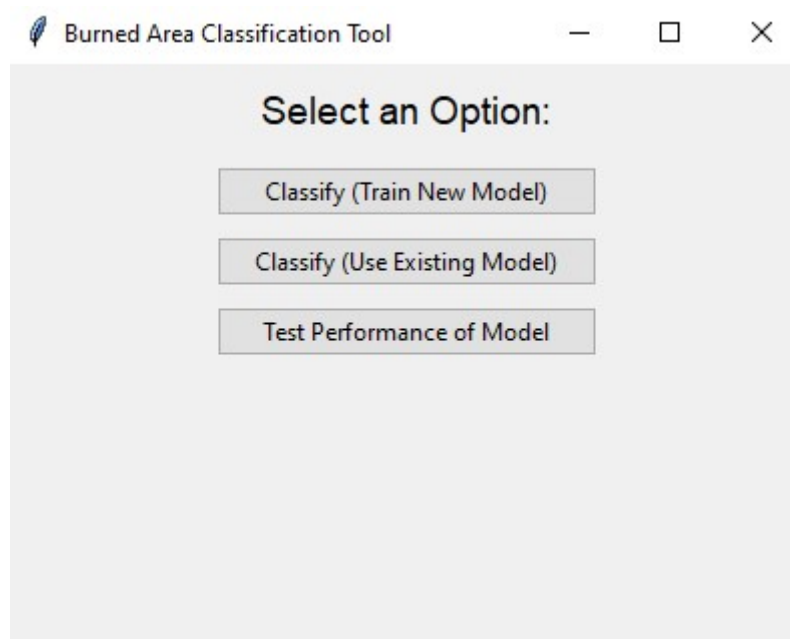
- **PyInstaller** - Converts the Python application into a standalone executable, allowing users to run the software without needing Python installed.

4. User Workflow

This chapter explains how users interact with the tool, outlining the step-by-step workflow for different use cases. The software allows users to train a new model, apply an existing model, or test an existing model, depending on their needs. The following sections describe the complete workflow, including user interactions, interface elements, and expected outputs for each stage.

Main Menu (BurnAreaClassifierApp)

When the user starts the application, they are presented with the main menu, which serves as the entry point for all functionalities. The interface allows users to choose between three options:



- **Classify (Train New Model)** – Initiates the process of training a new SVM model using pre- and post-fire images.

- **Classify (Use Existing Model)** – Loads a previously trained model and applies it to a new post-fire image for classification.
- **Test Performance of Model** – Evaluates an existing model's performance by comparing its predictions against dNBR-based validation data.

Training a New Model (TrainModelUI)

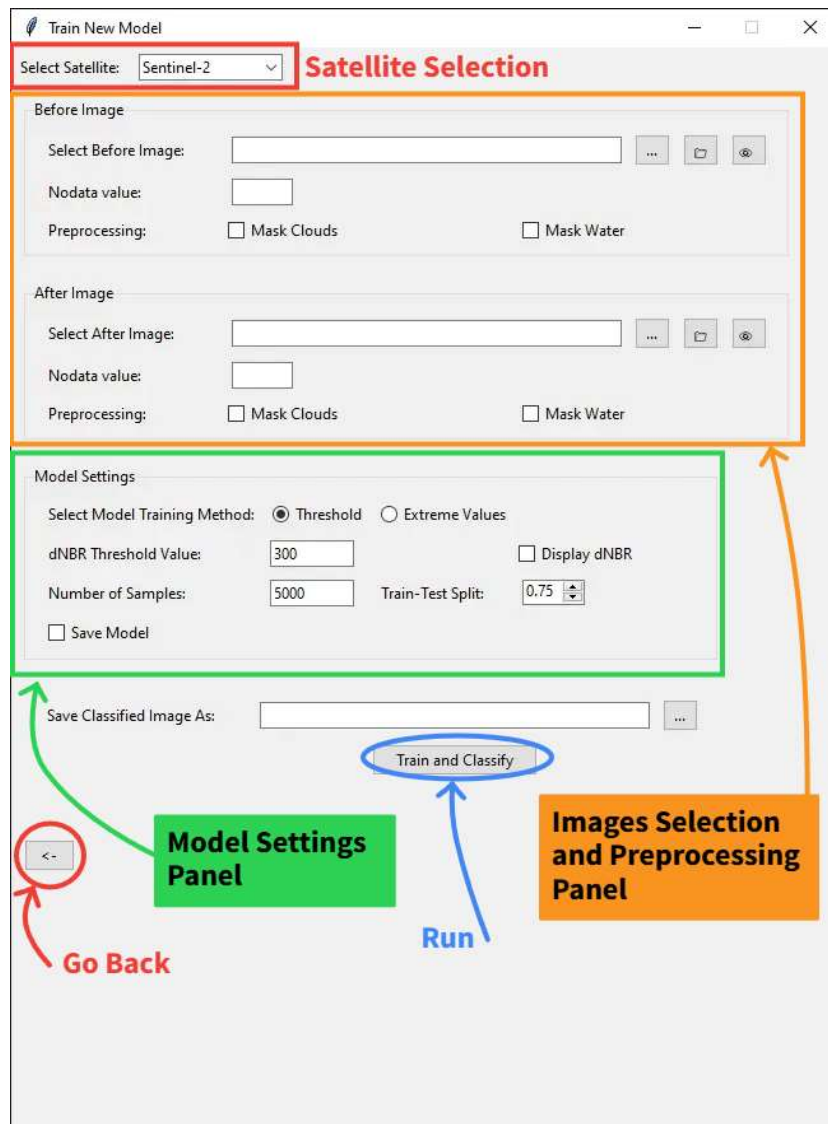
When users choose to train a new model, they follow a structured workflow that ensures proper preprocessing, training data creation, and model training. This section describes each step of the process.

Overview

The "Train New Model" window allows users to:

- Select pre-fire and post-fire images.
- Apply preprocessing steps like cloud and water masking.
- Define training parameters such as dNBR threshold and number of samples.
- Train an SVM model and save it for future use.
- Classify the full after-fire image with the trained model

Below is a labeled screenshot highlighting the different sections of the interface:



Satellite Selection

Users can choose between Sentinel-2 and Landsat 8/9 datasets; the software adjusts the options and band selection accordingly.

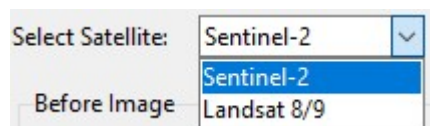


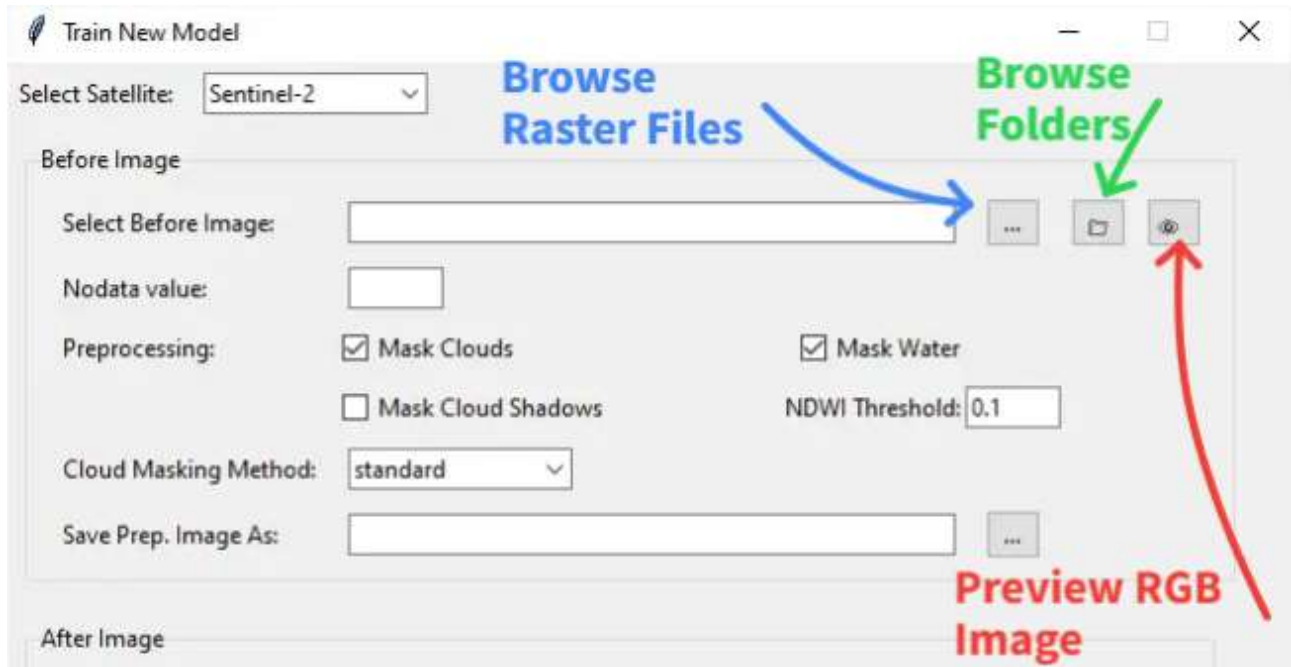
Image Selection and Preprocessing Panel

The user has two options for selecting input images:

- Choose a raster file that already contains all bands.
- Select a folder containing single-band files, which the software will automatically stack together into a single raster.

Once an image is loaded, the software attempts to detect the nodata value automatically. If detected, it is autofilled in the input box, but users can manually specify a different value if needed.

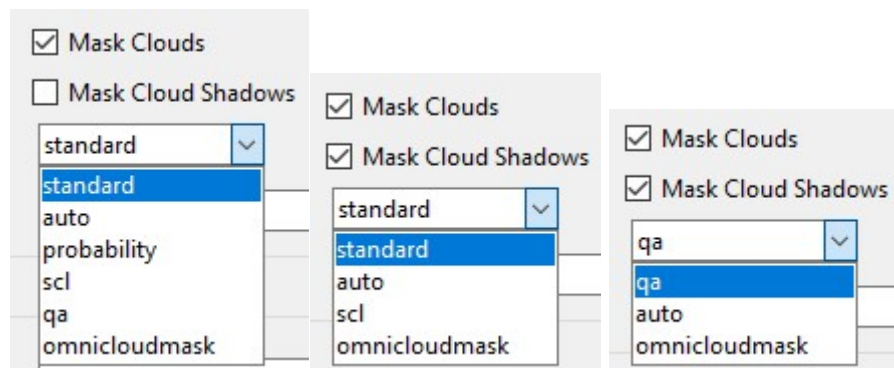
Users can also preview the loaded image using the Preview RGB Image button, which provides a quick visual check of the selected dataset before proceeding.



For preprocessing, users can enable:

- **Cloud masking**, with methods that adjust dynamically based on whether shadow masking is enabled and the type of satellite data selected, as shown below
- **Water masking**, using NDWI with a default threshold of 0.01, which can be modified by the user.

Below we can see different cloud masking methods based on satellite type and whether the user wants to mask cloud shadows or not



Additionally, users can specify a path to save the preprocessed file. If no path is provided, the software automatically saves the preprocessed image using the original filename with `_masked.tif` appended


Model Settings Panel

The Model Settings Panel allows users to configure how the training dataset is created, adjust sampling parameters, and specify output preferences.

Users can choose between two training methods:

1. Threshold Method:

- Uses a single threshold to separate burned and unburned areas.
- The default dNBR threshold is 300 (editable by the user).
- Pixels with dNBR below 300 are labeled as unburned, and those with dNBR above 300 are labeled as burned.

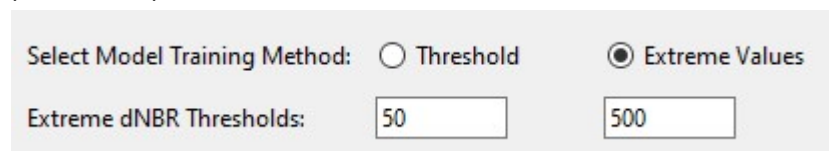


Select Model Training Method: ☒ Threshold ☐ Extreme Values

dNBR Threshold Value:

2. Extreme Values Method:

- Uses two thresholds to define burned and unburned samples more strictly.
- The default thresholds are 50 (unburned) and 500 (burned) (both editable by the user).
- The training set consists only of samples where dNBR is above 500 (burned) and below 50 (unburned).



Select Model Training Method: ☐ Threshold ☒ Extreme Values

Extreme dNBR Thresholds:

Other configurable settings include:

- **Number of Samples:** Users can manually define how many points to extract for training (default: 5000 samples).
- **Train-Test Split:** Users can adjust the percentage of data allocated for training vs. testing (default: 75% (0.75) training, 25% testing).
- **Save Model Option:** If enabled, the user must specify a path where the trained model will be saved.
- **Display dNBR Option:** If checked, the software will output the computed dNBR image during execution.

These settings allow users to fine-tune their training data and adjust classification performance as needed.

Model Settings

Select Model Training Method: ☒ Threshold ☐ Extreme Values

dNBR Threshold Value: ☐ Display dNBR

Number of Samples: Train-Test Split:

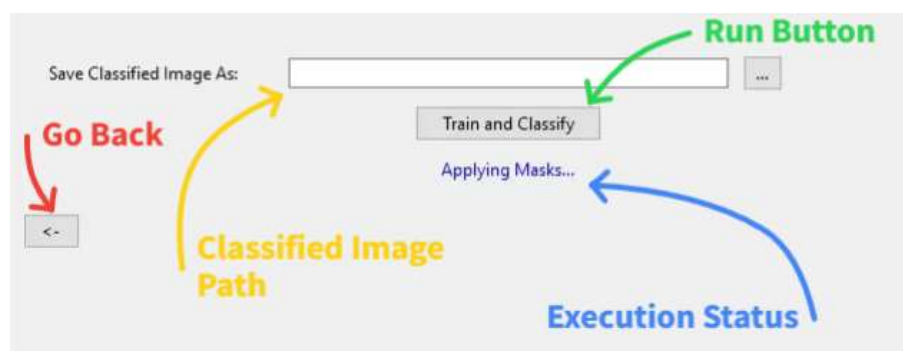
☒ Save Model

Execution and Navigation Controls

At the bottom of the Train New Model window, users can configure output settings and control execution. The key components of this section include:

- **Classified Image Path:** Users can specify where the classified image will be saved. If left empty, the software automatically names the output file using the after-fire image name with `_classified.tif` appended.
- **Train and Classify Button:** This button starts the training process using the selected settings. Once clicked, the tool generates training data, trains the SVM model, and classifies the post-fire image. The execution status is displayed below this button, allowing users to monitor progress.
- **Execution Status:** A message appears below the button indicating the current step of the process (e.g., 'Applying Masks...', 'Training Model...'). This helps users track the workflow in real-time.
- **Go Back Button:** This button allows users to return to the main menu, canceling the current operation if necessary.

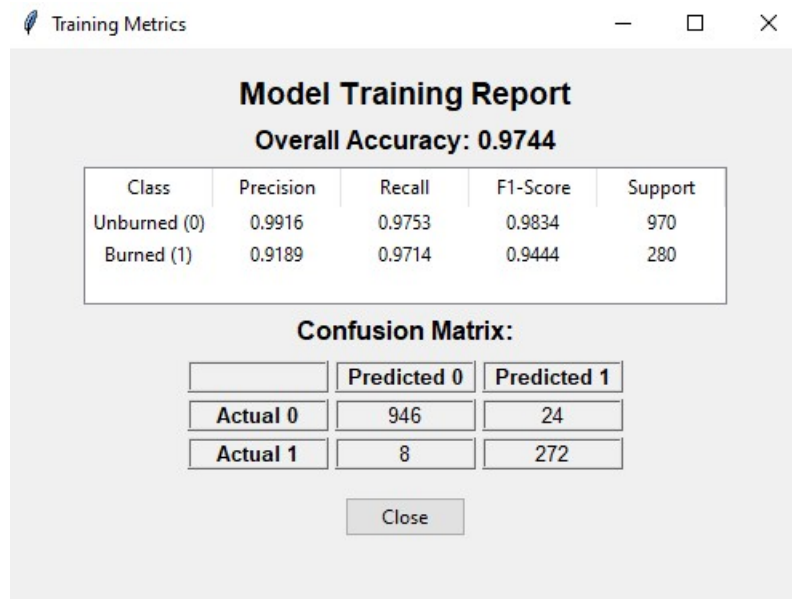
These buttons provide a straightforward way for users to execute model training or return to the previous menu if adjustments are needed.



Validation Results

After the training is complete, the model is validated on the testing set according to the train-test split defined by the user. A Training Metrics Report window appears, displaying validation accuracy, precision, recall, and F1-score for burned and unburned classes, along with the

complete confusion matrix, allowing users to assess the model's performance.



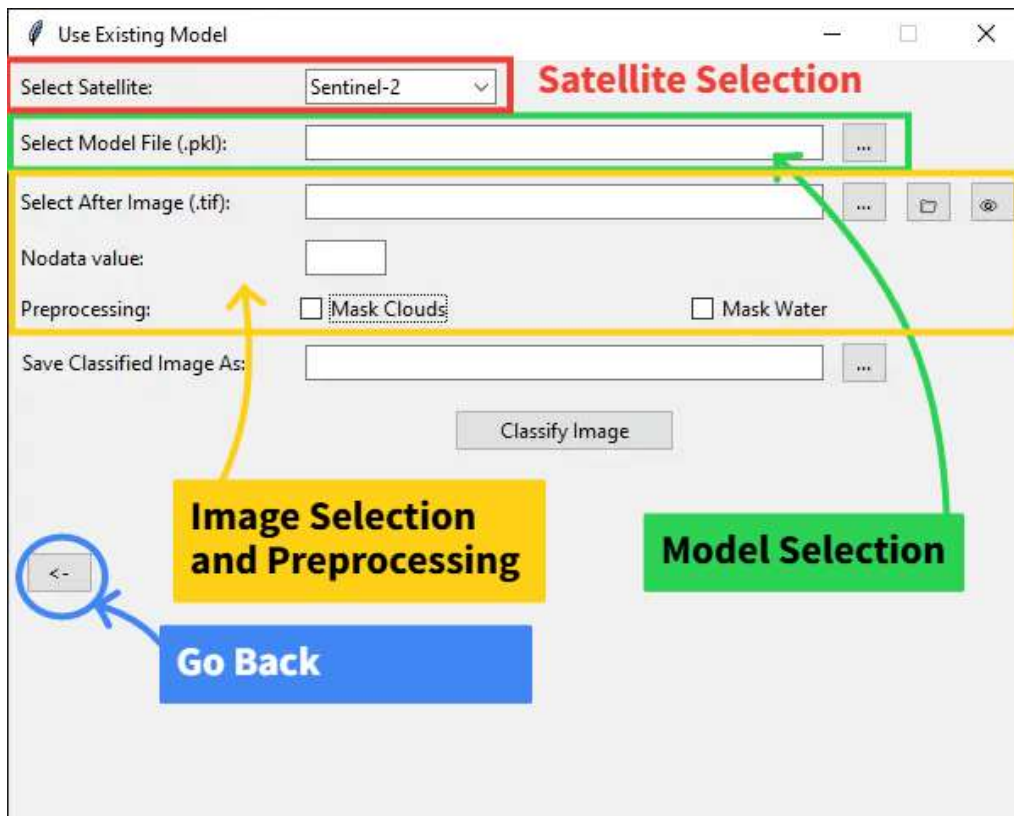
Classifying Using an Existing Model (UseExistingModelUI)

This mode allows users to apply a previously trained SVM model to classify a new post-fire image without requiring a pre-fire image.

Overview

The "Use Existing Model" window allows users to:

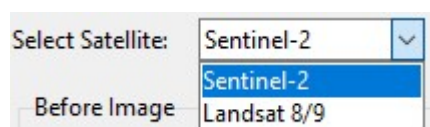
- Select a pre-trained SVM model.
- Load a post-fire image for classification.
- Apply cloud and water masking if needed.
- Generate a classified burned area map.



Satellite Selection

Users can choose between Sentinel-2 and Landsat 8/9 datasets; the software adjusts the options and band selection accordingly. However, it is crucial that the selected satellite type matches the one used during model training. For instance, a model trained on Sentinel-2 data will only work correctly on Sentinel-2 images, and the same applies to Landsat 8/9.

To ensure accurate classification, users must make sure that the selected satellite type, the loaded model, and the post-fire image are consistent. Mixing Sentinel-2 models with Landsat images (or vice versa) will lead to incorrect results.



Model Selection

Users must select a pre-trained SVM model stored in .pkl format. This model will be used to classify the post-fire image based on patterns learned during training. If no model is provided, classification cannot proceed.

It is also essential to ensure that the band configuration of the classification image matches the one used for training. If a model was trained on Sentinel-2 data with only bands B2, B3,

B4, and B8, it will not work correctly on an image with a different band combination (e.g., B4, B5, B6, B8A) or an image containing all 12 bands. The spectral information must remain consistent between training and classification for the model to function correctly.

Users should verify that both the satellite type and the band selection align with the model's training data to avoid misclassification.

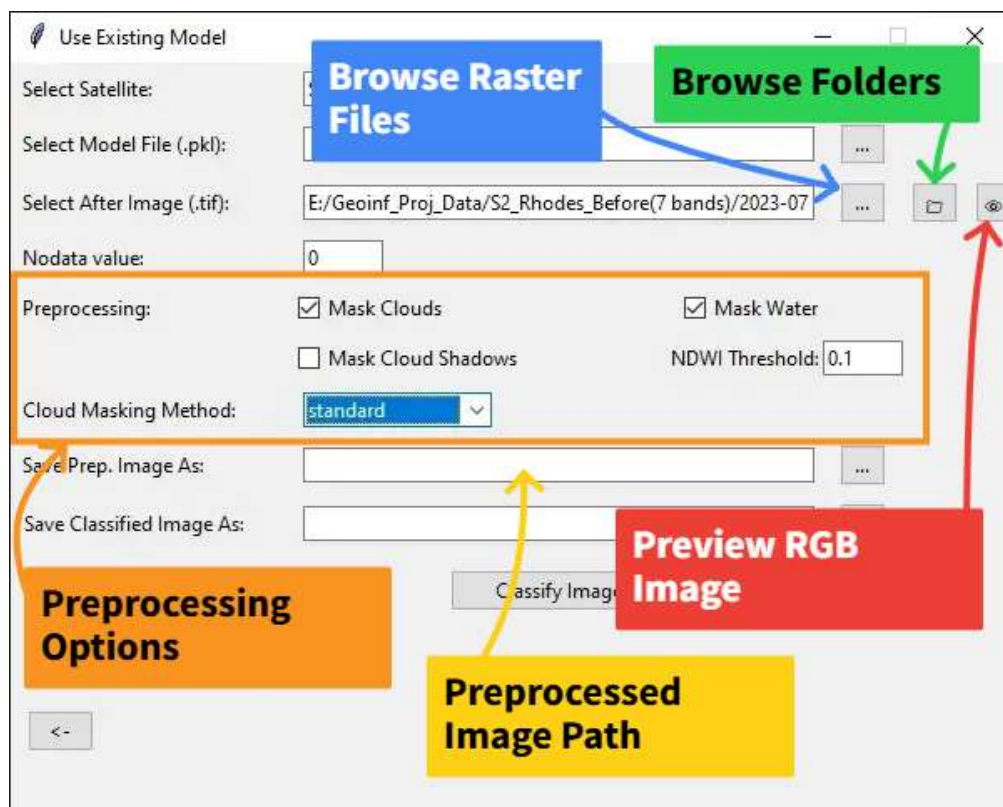
After Fire Image Selection and Preprocessing Options

Similar to the Training a New Model workflow, the user has two options for selecting the after-fire image:

- Choose a raster file that already contains all bands.
- Select a folder containing single-band files, which the software will automatically stack together into a single raster.

Once the image is loaded, the software attempts to detect the nodata value automatically. If detected, it is autofilled in the input box, but users can manually specify a different value if needed.

Users can also preview the loaded image using the Preview RGB Image button, which provides a quick visual check of the selected dataset before proceeding



For preprocessing, users can enable:

- **Cloud masking**, with methods that adjust dynamically based on whether shadow masking is enabled and the type of satellite data selected, as shown below

- **Water masking**, using NDWI with a default threshold of 0.01, which can be modified by the user.

Additionally, users can specify a path to save the preprocessed file. If no path is provided, the software automatically saves the preprocessed image using the original filename with `_masked.tif` appended.

Execution and Navigation Controls

At the bottom of the Use Existing Model window, users can configure output settings and control execution. The key components of this section include:

- **Classified Image Path:** Users can specify where the classified image will be saved. If left empty, the software automatically names the output file using the after-fire image name with `_classified.tif` appended.
- **Classify Image Button:** This button starts the classification process using the selected settings. The execution status is displayed below this button, allowing users to monitor progress.
- **Execution Status:** A message appears below the button indicating the current step of the process (e.g., 'Applying Masks...', 'Classifying Image...'). This helps users track the workflow in real-time.
- **Go Back Button:** This button allows users to return to the main menu, canceling the current operation if necessary.

These buttons provide a straightforward way for users to execute model training or return to the previous menu if adjustments are needed.



Testing a Model's Performance (TestModelPerformanceUI)

This mode allows users to evaluate the accuracy of a trained model by comparing its classification results against dNBR-derived ground truth labels.

Overview

The "Test Model Performance" window allows users to:

- Select a pre-trained SVM model.
- Load both pre-fire and post-fire images.

- Apply preprocessing steps (cloud and water masking) to ensure data consistency.
- Define the number of sample points and dNBR threshold for validation.
- Optionally classify the post-fire image and visualize dNBR.

The screenshot shows the 'Test Model Performance' window with the following components:

- Satellite Selection:** A red box highlights the 'Select Satellite:' dropdown menu, which is currently set to 'Sentinel-2'.
- Images Selection and Preprocessing Panel:** An orange box outlines the 'Before Image' and 'After Image' sections. Each section includes a 'Select [Before/After] Image:' field with file selection icons, a 'Nodata value:' input field, and 'Preprocessing:' checkboxes for 'Mask Clouds' and 'Mask Water'.
- Model Selection and Settings:** A green box highlights the 'Model Settings' section, which includes a 'Select Model File (.pkl):' field, a 'Number of Samples:' input field (set to 1000), and a 'Threshold dNBR:' input field (set to 300).
- Execution and Classification Settings:** A blue box highlights the 'Classification Settings' section, which includes checkboxes for 'Classify after image' and 'Display dNBR'.

Arrows indicate the flow of the process: a blue arrow points from the 'Execution and Classification Settings' box to the 'Classification Settings' section; a green arrow points from the 'Model Selection and Settings' box to the 'Model Settings' section; and an orange arrow points from the 'Images Selection and Preprocessing Panel' box to the 'Before Image' and 'After Image' sections. A 'Test Model' button is located below the 'Classification Settings' section.

Satellite Selection

Users can choose between Sentinel-2 and Landsat 8/9 datasets; the software adjusts the options and band selection accordingly. However, it is crucial that the selected satellite type

matches the one used during model training. For instance, a model trained on Sentinel-2 data will only work correctly on Sentinel-2 images, and the same applies to Landsat 8/9.

To ensure an accurate analysis of the results, users must make sure that the selected satellite type, the loaded model, and the post-fire image are consistent. Mixing Sentinel-2 models with Landsat images (or vice versa) will lead to incorrect results.



Image Selection and Preprocessing Panel

The image selection and preprocessing steps follow the same process as described in Section 4.2. Users can:

- Select both pre-fire and post-fire images.
- Enable cloud and water masking if necessary.
- Adjust the nodata value, which is autofilled when detected.
- Preview the selected image using the Preview RGB Image button.

For a detailed explanation of preprocessing settings, refer to Section 4.2.

Model Selection & Settings

Users must select a trained SVM model stored in .pkl format. The model is tested against dNBR values to assess classification accuracy. However, for reliable evaluation, it is essential to ensure that the dataset used for testing matches the model's training configuration. The satellite type must be consistent with the one used during training, as models trained on Sentinel-2 imagery will not perform correctly on Landsat 8/9 data and vice versa. Additionally, the band configuration must remain identical; a model trained using specific bands (e.g., B2, B3, B4, B8) will not function correctly on an image containing a different set of bands (e.g., B4, B5, B6, B8A) or all available bands. The spectral properties must align to maintain classification accuracy.

Users can also define:

- **Number of sample points:** Determines how many pixels are used for testing (default: 1000).
- **Threshold dNBR:** Specifies the cutoff for burned/unburned classification (default: 300).

Execution and Classification Settings

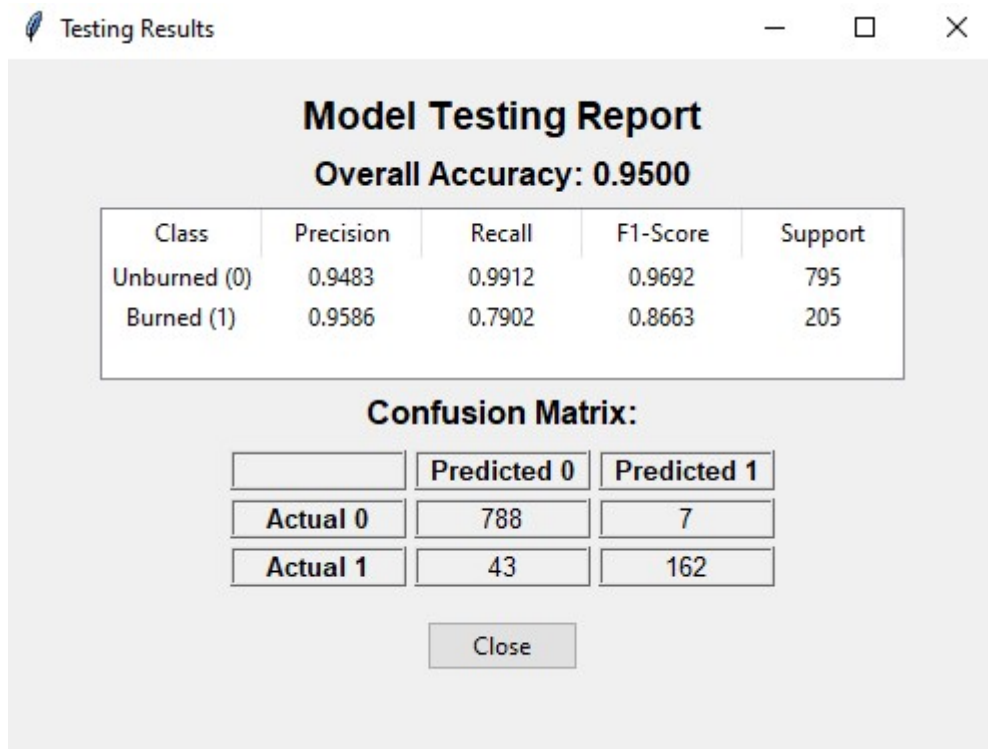
Users can enable two optional settings:

- **Classify after image:** If checked, the model will classify the entire post-fire image.
- **Display dNBR:** Shows the computed dNBR map for visual validation.

Once all settings are configured, users click Test Model to begin evaluation. The software will compare the classification output with dNBR-derived labels and compute accuracy metrics.

Testing Results

After the evaluation is complete, a Model Testing Report window appears, displaying the overall accuracy, precision, recall, and F1-score for burned and unburned classes. The confusion matrix provides a breakdown of actual vs. predicted classifications, helping users assess the model's performance.



5. Example Case Studies

Introduction

This chapter presents real-world examples demonstrating the workflow and effectiveness of the burned area classification tool. The goal is to illustrate how users can train a model, apply it to classify new fire events, and evaluate its performance using validation metrics. By following these case studies, users can better understand the practical applications and limitations of the tool.

The selected case studies focus on the Alexandroupoli 2023 Fire as the training dataset and the Rhodes 2023 Fire as the classification and testing dataset. These examples showcase different functionalities of the tool:

- Training a new model using pre- and post-fire images of the Alexandroupoli fire.

- Applying the trained model to classify burned areas in the Rhodes fire using only post-fire imagery.
- Quantitatively testing the model's accuracy by comparing its results with dNBR-derived labels for the Rhodes fire.

Training a New Model on the Alexandroupoli Fire Dataset

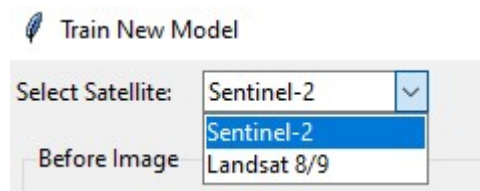
Step 1: Data Procurement

First, we acquire the necessary satellite images. For this case, we downloaded the pre-fire and post-fire Sentinel-2 images from Google Earth Engine, selecting:

- 7 spectral bands: B2 (Blue), B3 (Green), B4 (Red), B8 (NIR), B8A, B11, B12.
- Cloud masking bands: SCL for post-fire image, MSK_CLDPRB for pre-fire image.

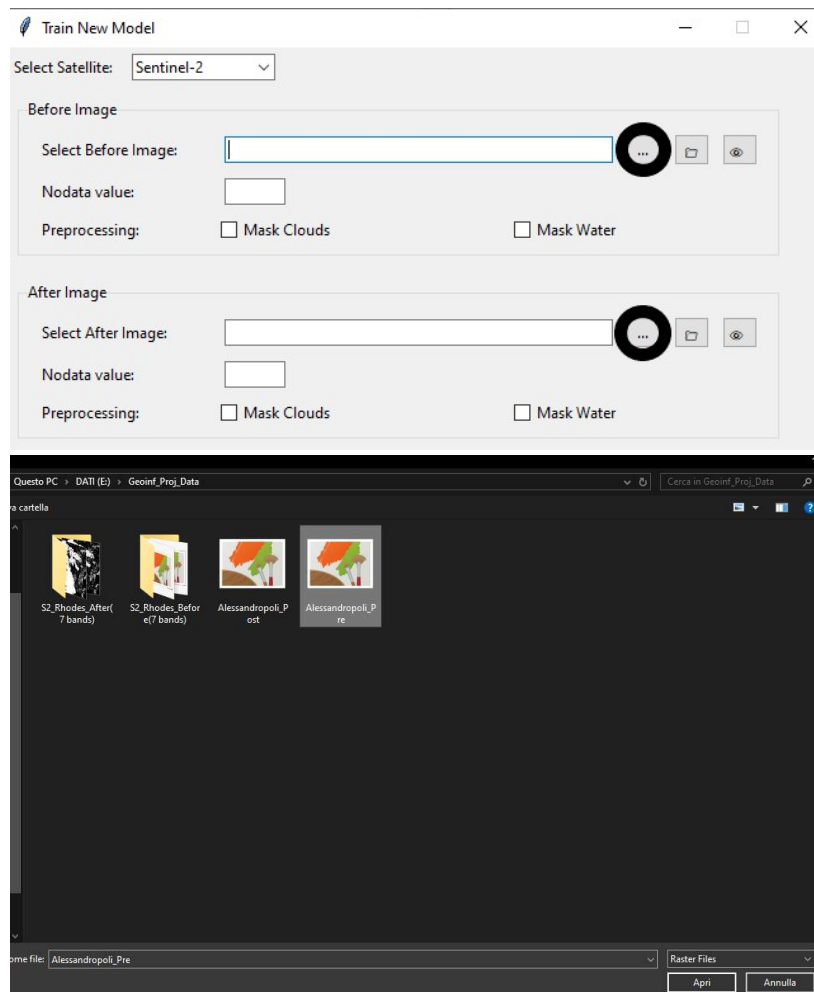
Step 2: Satellite Selection

Since we are working with Sentinel-2 images, we select it on the drop-down menu



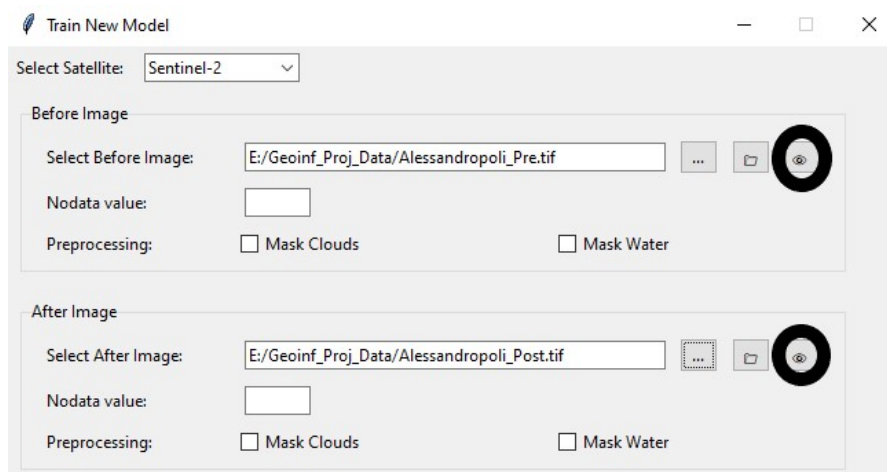
Step 3: Loading the Images

The user loads the images into the software by clicking the “browse files” button and selecting the file, as shown in the picture:

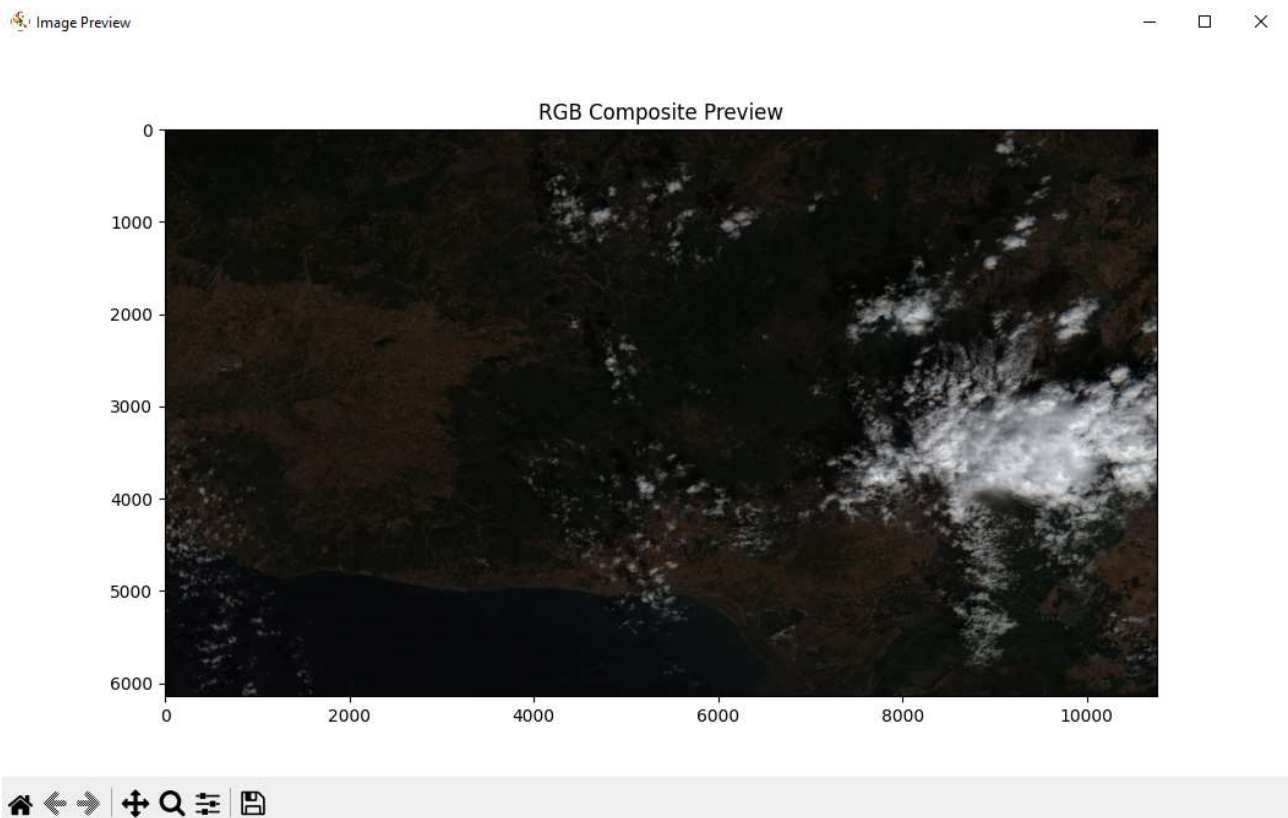


Step 4: Previewing the Images

To verify the correctness of the images, we use the Preview RGB Image button:



We can see the before image:



And the after image:



Step 5: Setting Preprocessing Options

To improve data quality we preprocess both images:

- **Water masking** is enabled for both images because the region contains water bodies. A threshold of $NDWI > 0.01$ is used to detect and remove water pixels from the dataset.
- **Cloud masking** is applied according to the available cloud bands:
 - o **Pre-fire image:** Since this image includes the MSK_CLDPRB band, the probability method is used to detect and mask clouds.
 - o **Post-fire image:** This image includes the Scene Classification Layer (SCL) band, which allows for both cloud and shadow detection. The SCL method is applied, with shadow masking enabled to further refine the quality of the burned area detection.

In this case, we leave the save path for the preprocessed images empty, allowing the software to automatically generate output filenames by appending `_masked.tif` to the original file path.

The screenshot shows the 'Train New Model' dialog box with the following settings:

- Select Satellite:** Sentinel-2
- Before Image:**
 - Select Before Image: E:/Geoinf_Proj_Data/Alessandropoli_Pre.tif
 - Nodata value: (empty)
 - Preprocessing: ☒ Mask Clouds, ☒ Mask Water, ☐ Mask Cloud Shadows
 - NDWI Threshold: 0.1
 - Cloud Masking Method: probability
 - Save Prep. Image As: (empty)
- After Image:**
 - Select After Image: E:/Geoinf_Proj_Data/Alessandropoli_Post.tif
 - Nodata value: (empty)
 - Preprocessing: ☒ Mask Clouds, ☒ Mask Water, ☒ Mask Cloud Shadows
 - NDWI Threshold: 0.1
 - Cloud Masking Method: scl
 - Save Prep. Image As: (empty)

Step 6: Defining Model Training Parameters

For this example, we use:

- Threshold training method, setting the dNBR threshold to 300.
- Number of training samples: 10,000.
- Train-test split ratio: 0.8.

Model Settings

Select Model Training Method: ☒ Threshold ☐ Extreme Values

dNBR Threshold Value: ☐ Display dNBR

Number of Samples: Train-Test Split:

☐ Save Model

We also save the trained model for future classification. To do so we first check “Save Model”, then we click the browse button to easily select the folder and the file name:

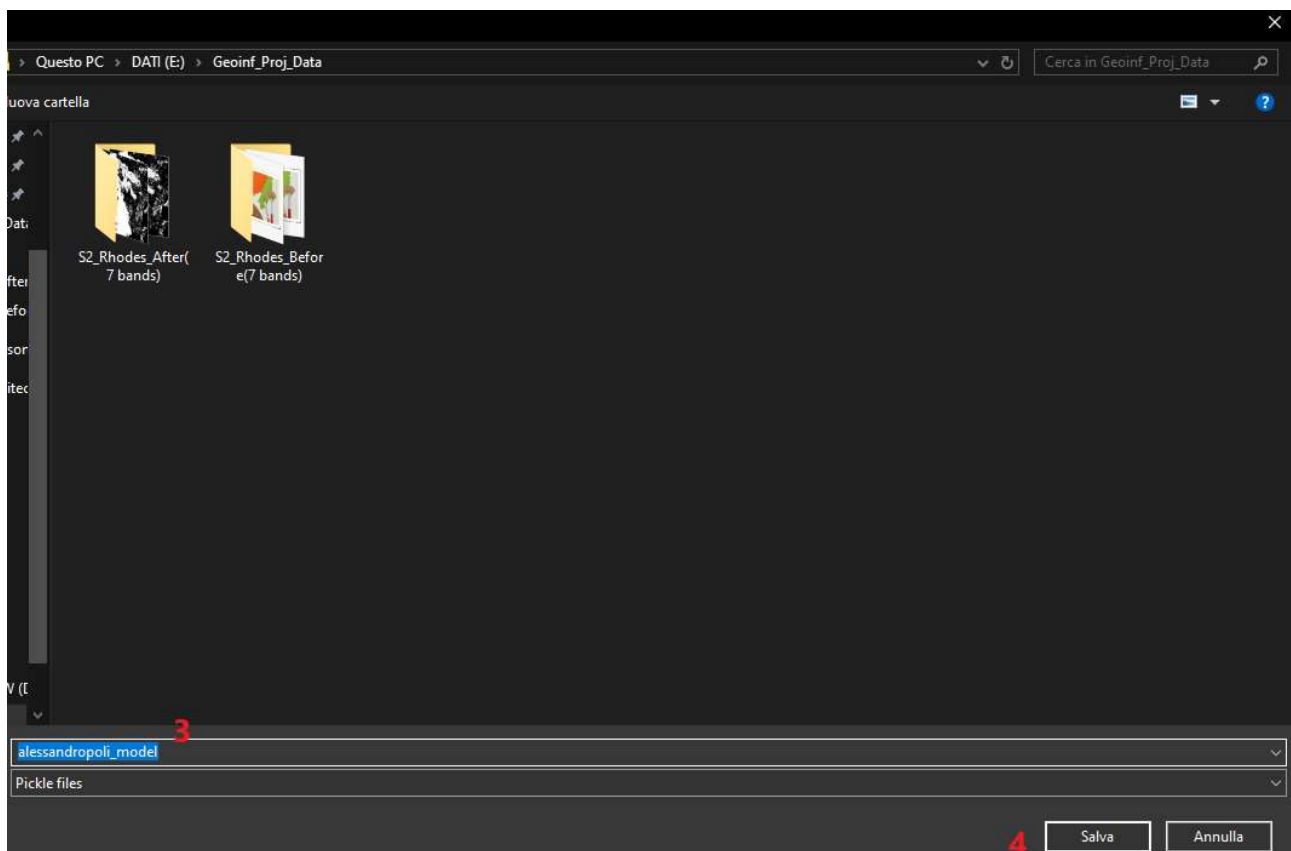
Model Settings

Select Model Training Method: ☒ Threshold ☐ Extreme Values

dNBR Threshold Value: ☐ Display dNBR

Number of Samples: Train-Test Split:

☒ Save Model ...



Step 6: Output and Display Options

For this example, we also want to visualize the dNBR so we check it in the model settings.

The last thing we need to do is to set the output path for the classified image (again, using the browse button as before).

At the end we should get a setup like this:

The screenshot shows the 'Train New Model' dialog box with the following settings:

- Select Satellite:** Sentinel-2
- Before Image:**
 - Select Before Image: E:/Geoinf_Proj_Data/Alessandropoli_Pre.tif
 - Nodata value: (empty)
 - Preprocessing: ☒ Mask Clouds, ☒ Mask Water, ☐ Mask Cloud Shadows
 - NDWI Threshold: 0.1
 - Cloud Masking Method: probability
 - Save Prep. Image As: (empty)
- After Image:**
 - Select After Image: E:/Geoinf_Proj_Data/Alessandropoli_Post.tif
 - Nodata value: (empty)
 - Preprocessing: ☒ Mask Clouds, ☒ Mask Water, ☒ Mask Cloud Shadows
 - NDWI Threshold: 0.1
 - Cloud Masking Method: scl
 - Save Prep. Image As: (empty)
- Model Settings:**
 - Select Model Training Method: ☒ Threshold, ☐ Extreme Values
 - dNBR Threshold Value: 300
 - Number of Samples: 10000
 - Train-Test Split: 0.80
 - ☒ Save Model
 - Save Model Path: E:/Geoinf_Proj_Data/alessandropoli_model.pk
 - ☒ Display dNBR
- Save Classified Image As:** E:/Geoinf_Proj_Data/Alessandropoli_classified.tif
- Train and Classify** button

We are now ready to run the training and the classification.

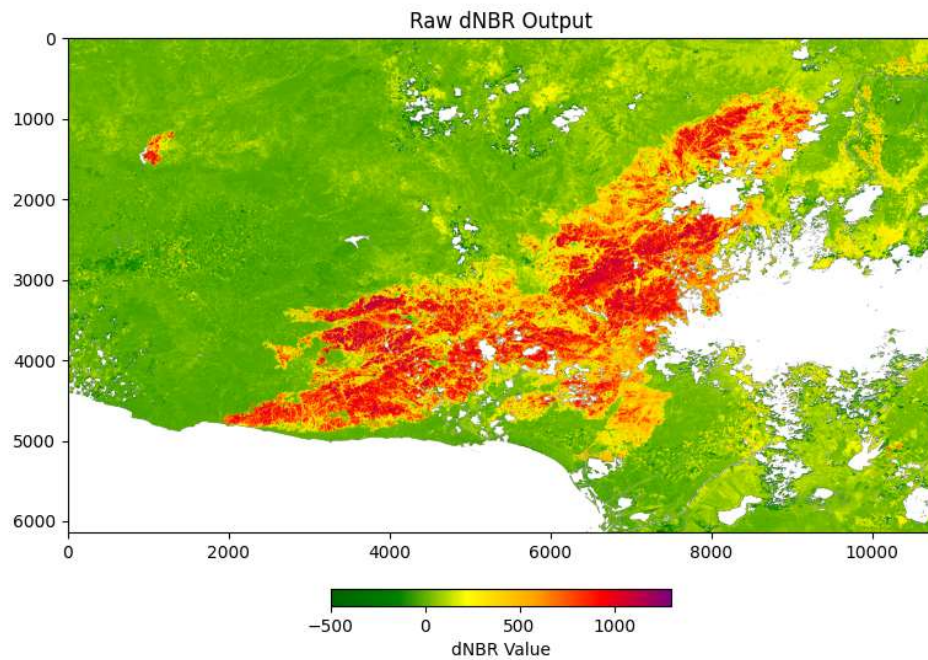
Step 7: Running the Training and Classification

Once all settings are configured, we initiate the model training by clicking Train and Classify. The software processes the images, applies the selected preprocessing steps, and trains the SVM model based on the defined parameters.

1. **Preprocessing Execution:** The software applies water and cloud masking according to the selected settings.
2. **Computing NBR and dNBR:** The software computes the Normalized Burn Ratio (NBR) for both images and then calculates dNBR. : Since the "Display dNBR" option was enabled, the computed dNBR map is displayed, allowing users to verify burned area differences.

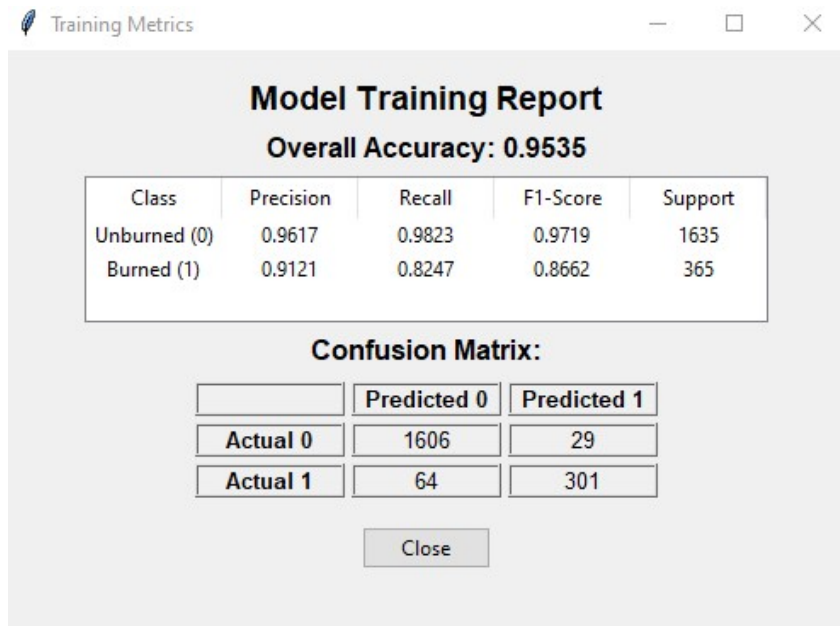
dNBR

— □ ×

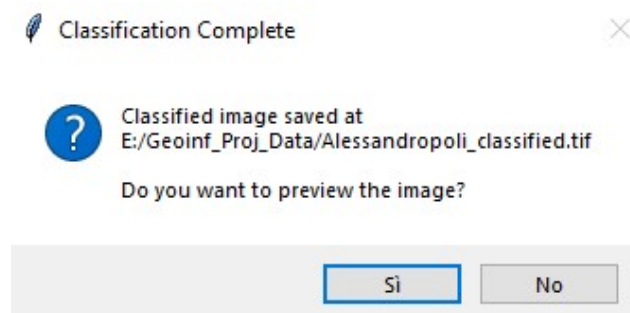


Home Left Right Pan Zoom In Zoom Out Layers Save

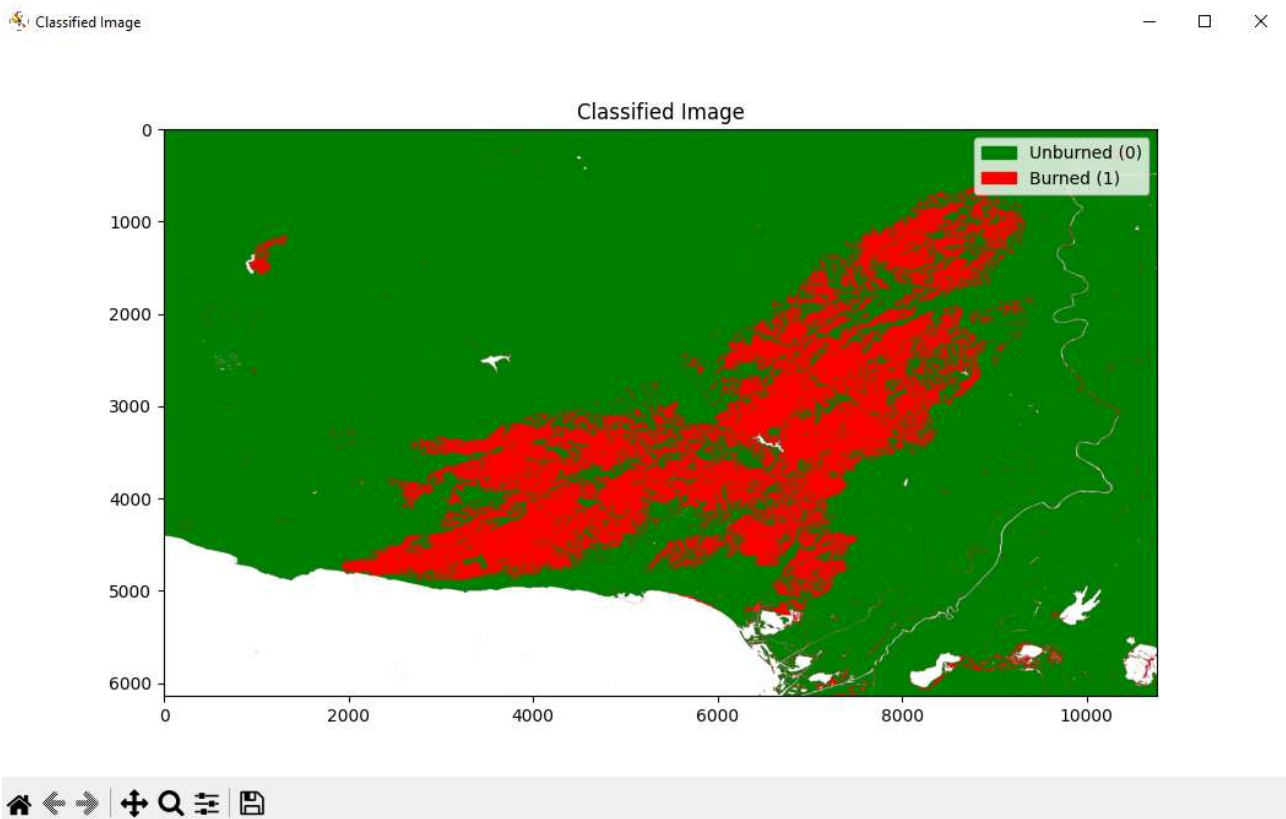
3. **Training the Model and Validating Performance:** The model is trained using the selected parameters and validated against the test set. Upon completion, a Training Metrics Report window appears, showing the overall accuracy of the model and other metrics



4. **Classification Execution and Completion Dialog:** The trained model is applied to classify the post-fire image. Once classification is complete, a pop-up window informs the user and prompts whether they want to preview the classified image.



5. **Displaying the Classified Image:** If the user clicks "Yes," a preview of the classified image is displayed.



This preview provides a quick visual assessment of the classification results. However, for further analysis and post-processing, the classified raster files can be opened in specialized GIS software such as QGIS or ArcGIS to extract additional insights and refine the classification.

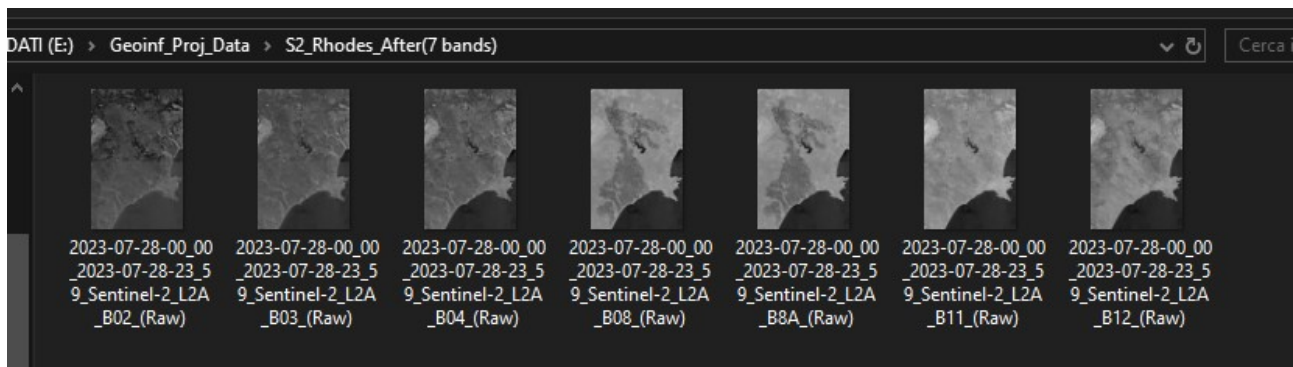
Classifying the Rhodes Fire Using the Trained Model

After training the model on the Alexandroupoli fire, we apply it to classify burned areas in the Rhodes 2023 Fire. This step demonstrates how a pre-trained model can be used for burned area detection when only post-fire imagery is available.

Step 1: Data Procurement

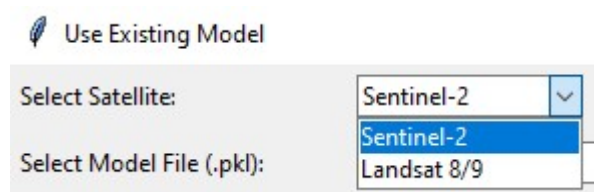
For this case, we downloaded the post-fire Sentinel-2 image from EO Browser, resulting in a folder containing single-band raster files. The selected bands match the ones used for training: B2 (Blue), B3 (Green), B4 (Red), B8 (NIR), B8A, B11, B12.

The image is the post-fire capture of the Rhodes 2023 Fire.



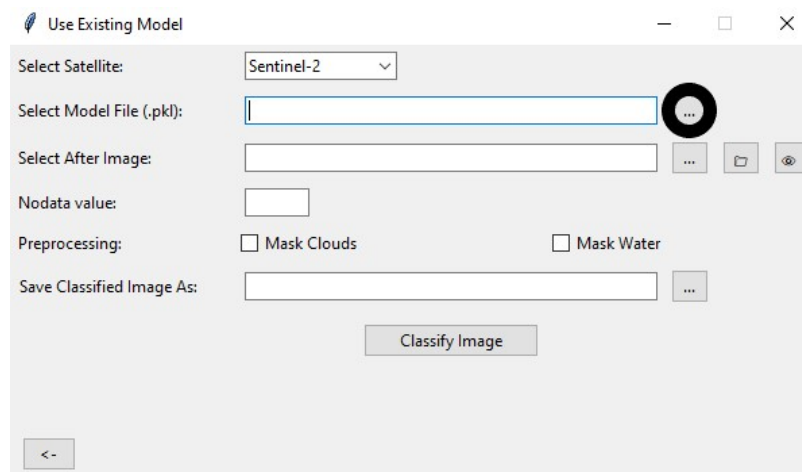
Step 2: Satellite Selection

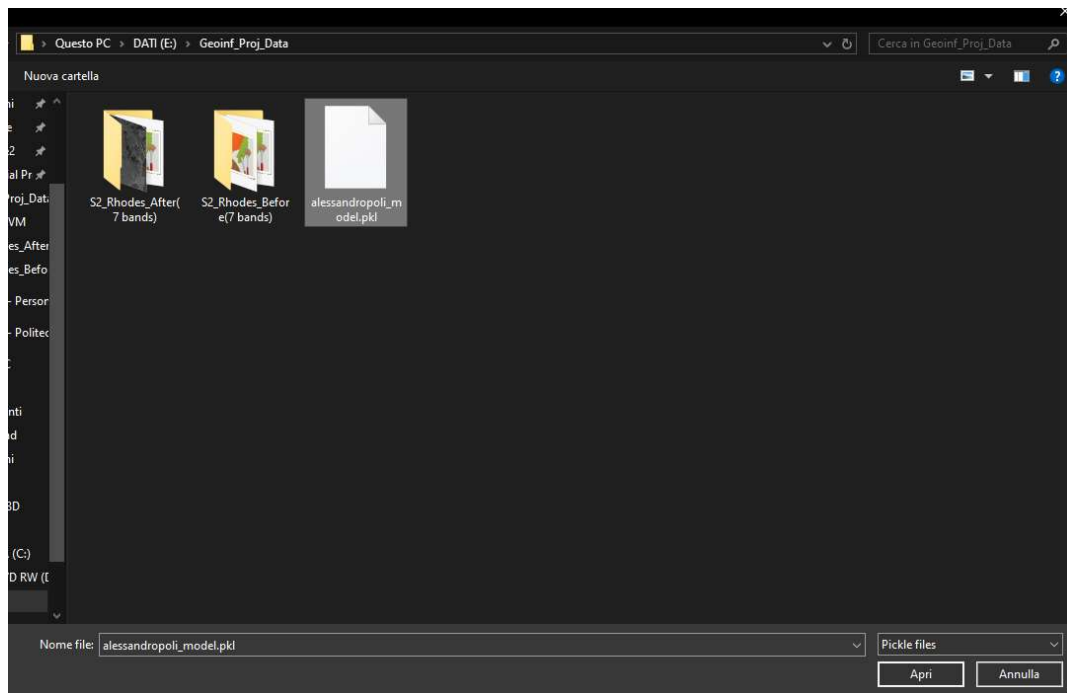
Since we are working again with Sentinel-2 images, we select it on the drop-down menu



Step 3: Loading the Model

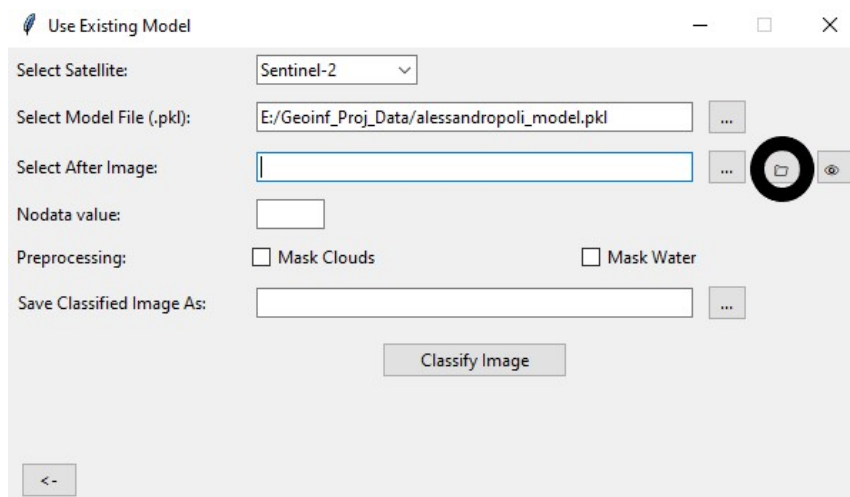
To classify the Rhodes fire, we load the previously trained Alexandroupoli model by clicking the Browse File button and selecting the corresponding .pkl model file.

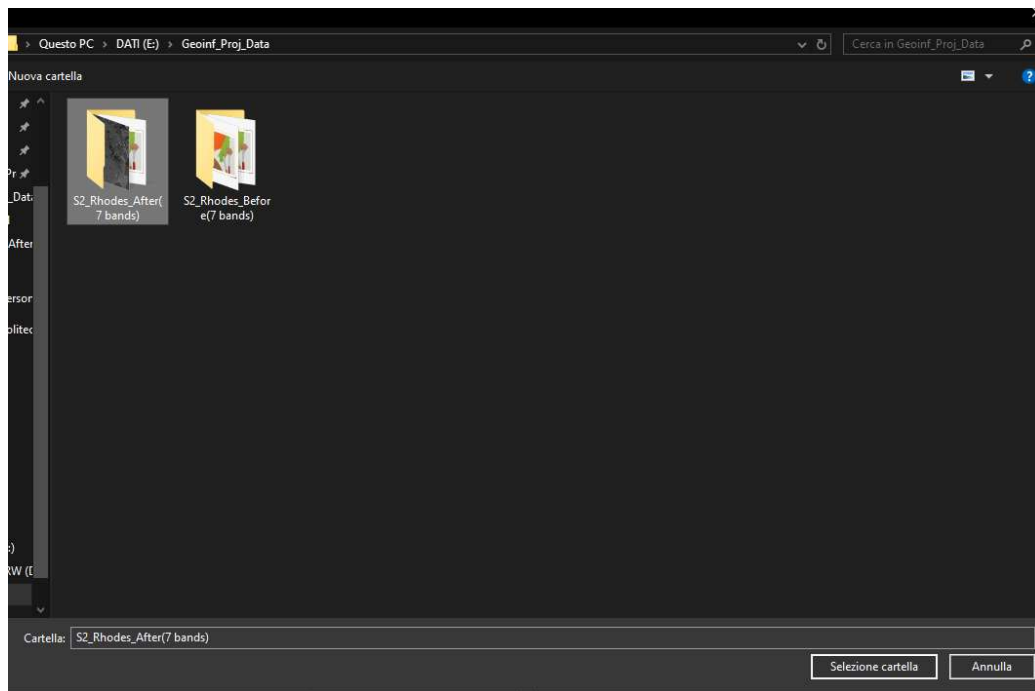




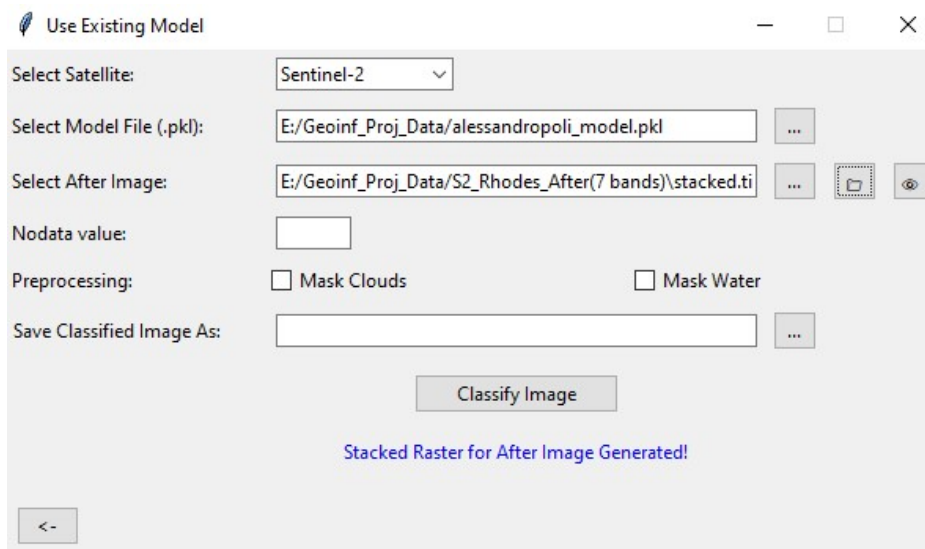
Step 4: Loading the Image

Since the post-fire image consists of separate band files, we use the Browse Folder button to navigate to the folder containing the bands. Instead of selecting a single raster file, the software will automatically detect and stack the selected bands into a single raster.



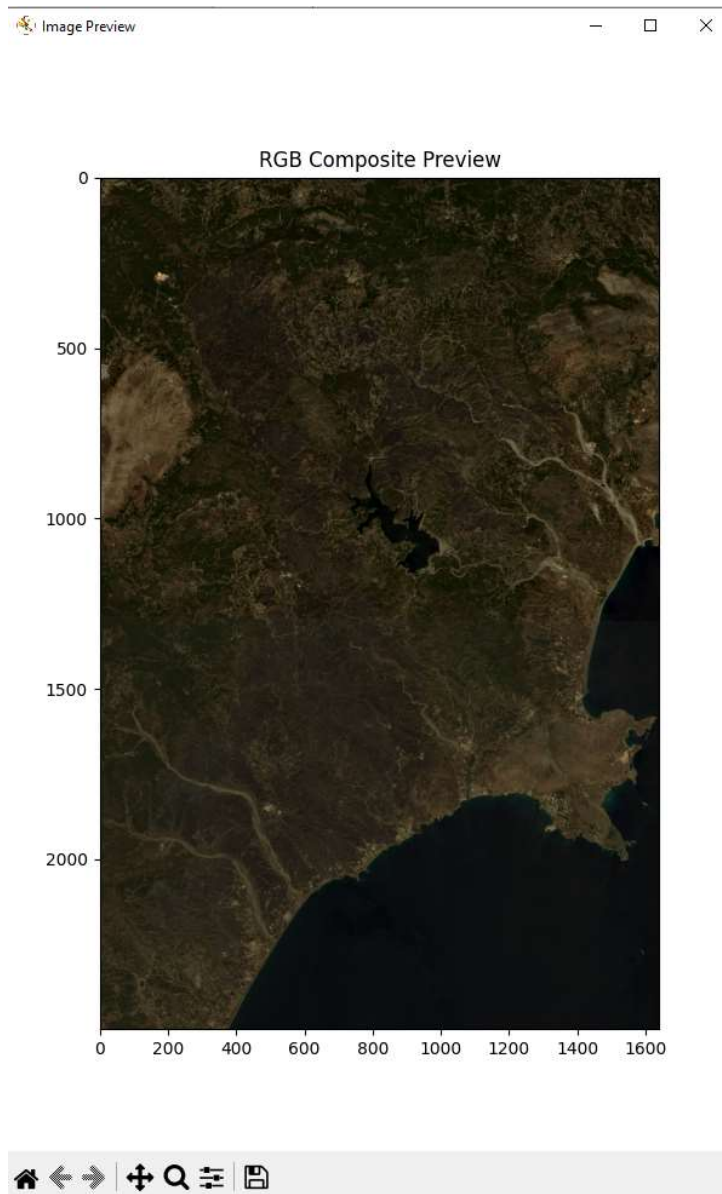


Once stacking is completed, the software notifies the user with a confirmation message indicating that the stacked image has been successfully created and is ready for processing.



Step 5: Previewing the Image

Again, we can see an RGB preview on the image clicking on the preview image button



Step 6: Setting Preprocessing Options

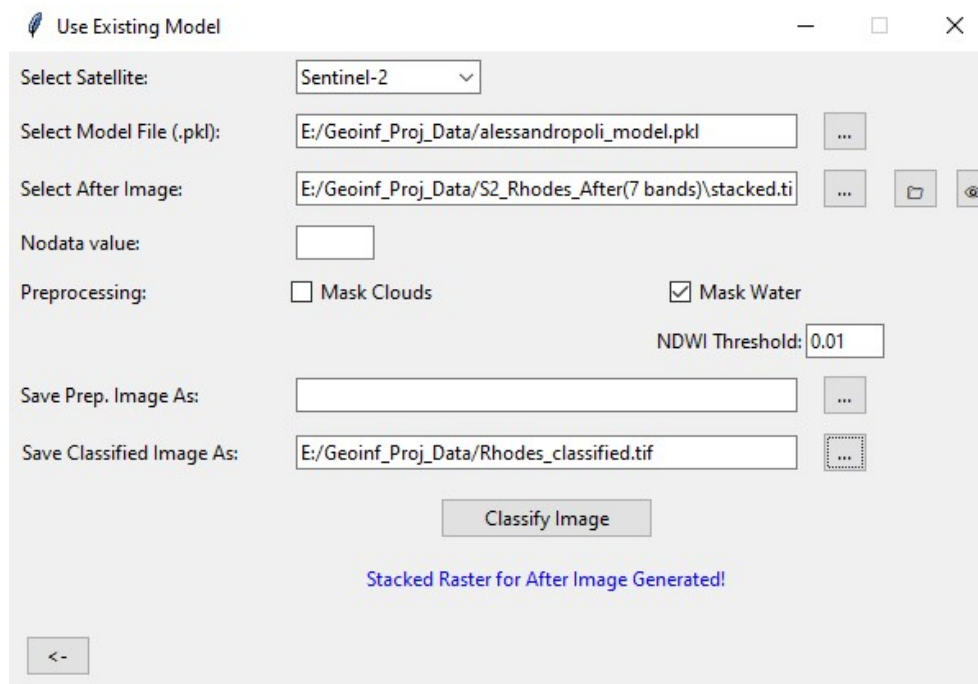
For this image, we only need to apply water masking, as cloud contamination is not a concern. We use the standard NDWI threshold of 0.01 to detect and remove water pixels from the dataset.

Again, we leave the save path for the preprocessed images empty, allowing the software to automatically generate output filenames by appending `_masked.tif` to the original file path.

Step 7: Output Options

The last thing we need to do is to set the output path for the classified image (again, using the browse button as before).

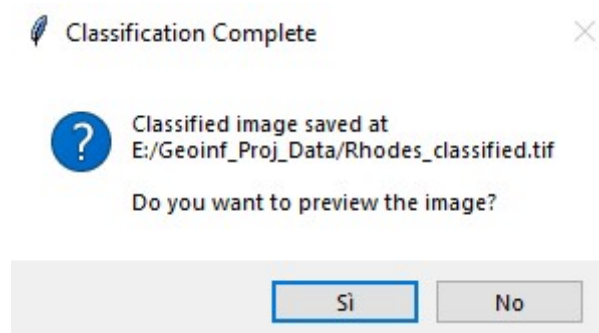
At the end we should get a setup like this:



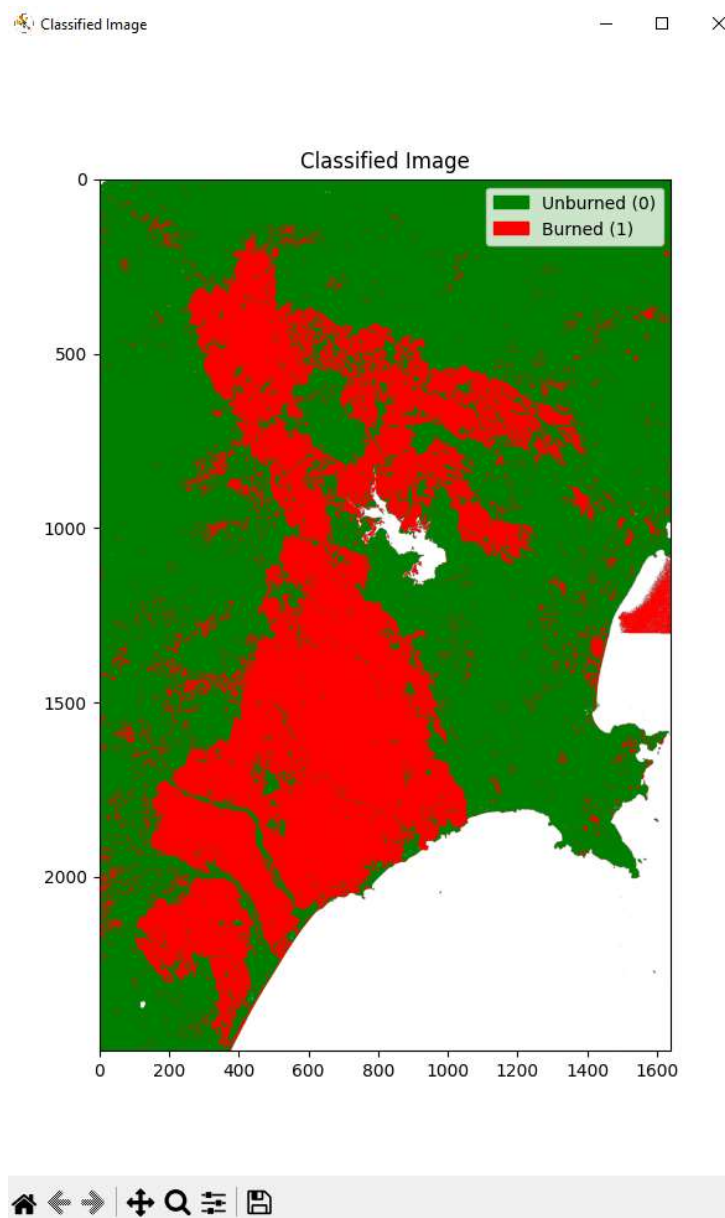
Step 8: Running the Classification

Once all settings are configured, we initiate the model training by clicking Classify Image. The software preprocesses the image and classified it according to the loaded model

1. **Preprocessing Execution:** The software applies water and cloud masking according to the selected settings.
2. **Classification Execution and Completion Dialog:** The loaded model is applied to classify the image. Once classification is complete, a pop-up window informs the user and prompts whether they want to preview the classified image.



3. **Displaying the Classified Image:** If the user clicks "Yes," a preview of the classified image is displayed.



As said before, this preview provides a quick visual assessment of the classification results. However, for further analysis and post-processing, the classified raster files can be opened in specialized GIS software such as QGIS or ArcGIS to extract additional insights and refine the classification.

Testing the Model's Performance on the Rhodes Fire

After applying the trained model to classify burned areas in the Rhodes 2023 Fire, we now aim to quantitatively assess the model's accuracy. To achieve this, we will compute the dNBR from pre-fire and post-fire images and use a threshold-based ground truth for validation.

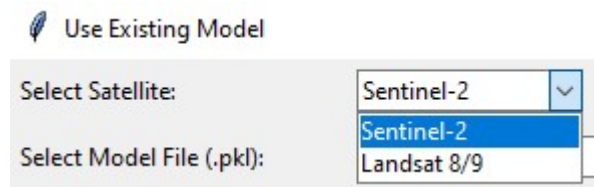
Step 1: Data Procurement

The post-fire image has already been downloaded in the previous part. Now, we need the pre-fire image to compute dNBR for validation. The pre-fire image is obtained using the same approach as before, downloaded from EO Browser, resulting in a folder containing single-

band raster files. The selected bands match the ones used for training: B2 (Blue), B3 (Green), B4 (Red), B8 (NIR), B8A, B11, B12.

Step 2: Satellite Selection

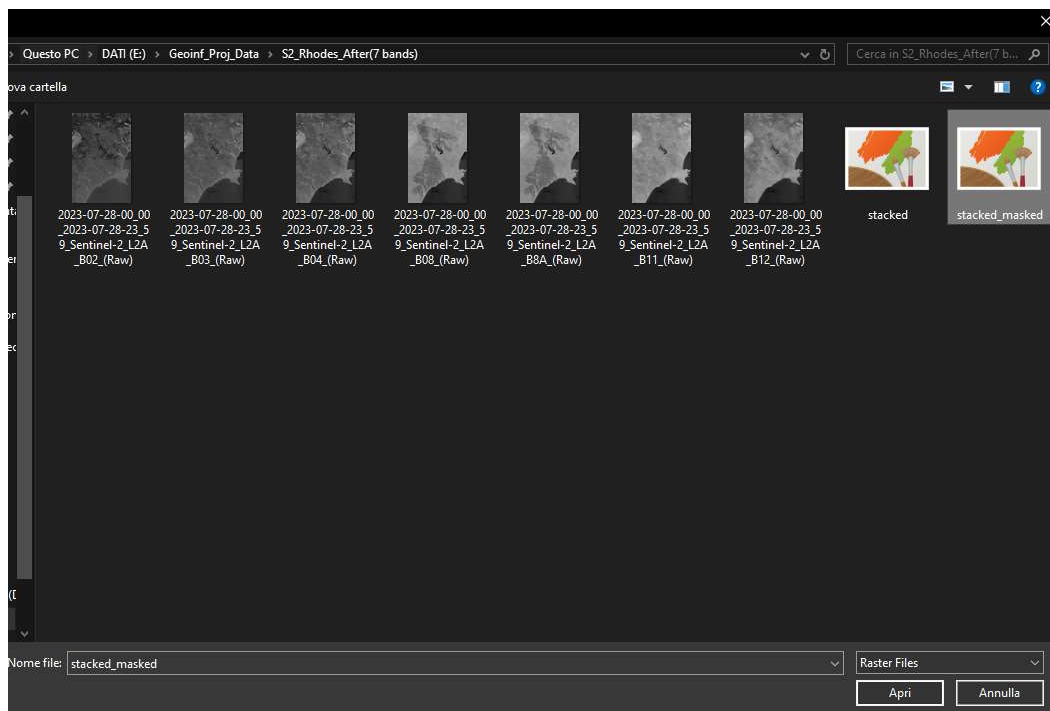
Since we are working with Sentinel-2 imagery, we ensure that the correct satellite type is selected in the dropdown menu for consistency with the training dataset.



Step 3: Loading the Images

The pre-fire image is loaded by selecting the folder containing the individual bands exactly as we did earlier. The software automatically stacks them into a single raster.

The post-fire image has already been preprocessed in the previous section, so we directly load the `_masked.tif` version instead of reprocessing it.



Step 4: Preprocessing Options

Water masking is applied to the pre-fire image using the default NDWI threshold (0.1) to remove water pixels.

The post-fire image remains unchanged, as we loaded the already preprocessed version.

Before Image

Select Before Image:
E:/Geoinf_Proj_Data/S2_Rhodes_Before(7 bands)\stacked.
...
Folder icon
Eye icon

Nodata value:

Preprocessing:
☐ Mask Clouds
☒ Mask Water
NDWI Threshold:

Save Prep. Image As:
...

After Image

Select After Image:
E:/Geoinf_Proj_Data/S2_Rhodes_After(7 bands)/stacked_r
...
Folder icon
Eye icon

Nodata value:

Preprocessing:
☐ Mask Clouds
☐ Mask Water

We can see that the software automatically detects the no-data value for the masked image

Step 5: Model Loading and Validation Settings

The pre-trained model (trained on the Alexandroupoli Fire dataset) is loaded again using the Browse File button. We then need to define the number of sample points for validation, setting it to 2000. We could also change the dNBR threshold that distinguish burned areas from unburned ones, but we maintain a threshold of 300 to ensure consistency with the training process, where this value was used to separate burned and unburned areas.

Model Settings

Select Model File (.pkl):
E:/Geoinf_Proj_Data/alessandropoli_model.pkl
...

Number of Samples:
Threshold dNBR:

Step 6: Classification Settings

Since we have already classified the post-fire image previously, we uncheck 'Classify After Image'. However, we check 'Display dNBR' to compare the ground-truth thresholded dNBR against the model's classification, allowing us to visually assess agreement.

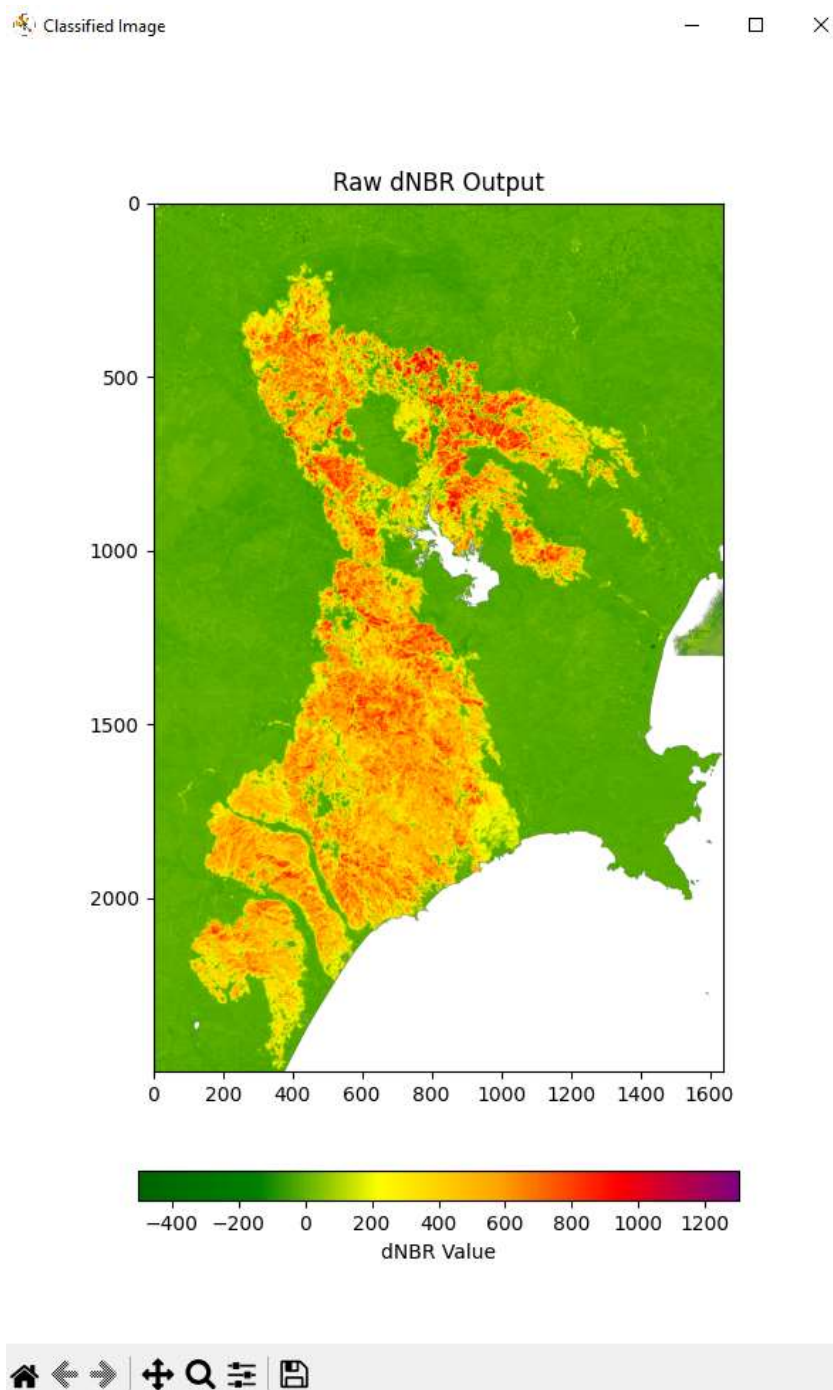
Classification Settings

☐ Classify after image
☒ Display dNBR

Step 7: Execution

Once all settings are configured, we start the validation process by clicking on the “Test Model” button:

1. **Preprocessing Execution:** The pre-fire image undergoes water masking, ensuring that the dataset is clean for dNBR computation.
2. **dNBR Computation and Visualization:** The software computes the Normalized Burn Ratio (NBR) for both pre-fire and post-fire images. The dNBR is then calculated and, since "Display dNBR" is enabled, displayed, allowing us to visually inspect the burn severity.



3. **Model Testing and Accuracy Assessment:** The software samples 2000 validation points from the dNBR map and compares them with the model's classification. The Training Metrics Report appears, displaying:

Testing Results

Model Testing Report

Overall Accuracy: 0.9675

Class	Precision	Recall	F1-Score	Support
Unburned (0)	0.9830	0.9747	0.9788	1541
Burned (1)	0.9174	0.9434	0.9302	459

Confusion Matrix:

	Predicted 0	Predicted 1
Actual 0	1502	39
Actual 1	26	433

Close

This validation step provides insight into the transferability and robustness of the trained model, offering users a quantitative measure of its performance on a different fire event.

6. Conclusions

This project aimed to create a practical tool for burned area detection using Support Vector Machines (SVM). By integrating image preprocessing, model training, classification, and validation into a user-friendly workflow, the tool simplifies the application of machine learning to wildfire analysis. The graphical interface makes it easier for users to process satellite imagery, generate training datasets, and classify burned areas without needing extensive programming experience.

The case studies provided insight into how the tool performs in different scenarios, particularly by applying a model trained on one fire event (Alexandroupoli 2023) to another (Rhodes 2023). The validation step using dNBR as ground truth helped assess the model's accuracy and highlighted some challenges, particularly in generalizing results across different fire conditions.

A key takeaway is the tool's flexibility—it supports Sentinel-2 and Landsat 8/9 imagery, allows for different preprocessing approaches, and provides multiple training options. However, the results also suggest that models trained on specific fire conditions may not always transfer

well to other regions or sensor configurations. This reinforces the importance of careful model validation and dataset selection.

Future Considerations

While the tool already offers a streamlined way to apply machine learning to burned area detection, there are several ways it could be further improved:

- Exploring additional classification methods, such as Random Forest or deep learning models, to compare performance.
- Expanding dataset compatibility to include other satellite sources, such as MODIS or VIIRS, for broader applicability.
- Improving hyperparameter tuning to optimize model performance with less manual input.
- Enhancing user experience, such as interactive burn severity mapping and more detailed reporting options.

These enhancements could make the tool even more versatile for wildfire analysis and post-fire impact assessment. While there is room for improvement, the current implementation provides a useful starting point for applying machine learning techniques to remote sensing in an accessible way.