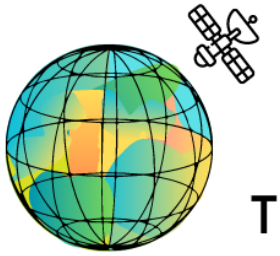
POLITECNICO DI MILANO

Geoinformatics Project

M.Sc. Geoinformatics Engineering

a.y. 2022/23



THEMATIC CLASSIFICATION

Image Classification

QGIS Plugin

Authors:
Mattia Koren - m. 993021
Gaia Vallarino - m. 996164

Supervisor:
Prof. Marco Gianinetto

version 1.0

Image Classification - QGIS Plugin

Mattia Koren, Gaia Vallarino

June 19, 2023

Contents

Abstract

In the QGIS environment, the Supervised Classification plugin is a powerful tool for classification tasks on geospatial data.
This plugin makes use of GDAL (Geospatial Data Abstraction Library) and scikit-learn, two popular libraries for machine learning and geospatial analysis. The quick loading, processing, and manipulation of geographic data is made possible by GDAL, while a wide range of classification techniques and assessment measures are offered by scikit-learn.
A user-friendly interface is provided by the plugin for data input, model training, preprocessing, and result visualization. It supports a number of classification algorithms, such as Support Vector Machines, K-Means, and Gaussian Naive Bayes.
Users may quickly categorize geospatial data, assess model performance, and produce classified outputs for additional analysis and decision-making with the help of the Supervised Classification plugin.

1 Description of Project

1.1 Purpose

This is the user documentation for the QGIS plugin ClassEO, which has a main goal the classification (both supervised and unsupervised) of land cover classes on satellite imagery.

The purpose of this document is to help the user familiarize, learn how to use the plugin, and identify the main errors that might arise from its use.

In the appendices of this document it is also possible to find a case study scenario (Appendix A: Case Study), and a guide for anyone who wants to improve the existing code (Appendix B: Programmer's Guide)

1.2 Goals

The *nameplugin* QGIS plugin aims to provide a user-friendly and efficient tool for performing supervised and unsupervised classification of land cover classes starting from satellite imagery on the QGIS environment.

The plugin uses scikit-learn, a famous python machine learning library, to provide a selection of classification algorithms best suited for Earth Observation tasks and the analysis of geospatial data. The plugin also employs the gdal library (Geospatial Data Abstraction Library) to enhance the performance and efficiency of the classification (here an hypertext to Chapter V, tools used in the project).

A more in-depth overview of the libraries and tools used in the project can be found in chapter V.

The primary goals of the project are the following:

1. **Easy-to-Use Interface**: the GUI is built following a specific workflow that strives to be the most user-friendly, without requiring extensive knowledge of machine learning topics, The goal would be for the user to

directly start working with the application without needing to have an in-depth study of the documentation, ensuring a familiar user experience.

2. **Diverse Classification Algorithms**: the application wants to support a wide range of classification algorithms. While at the moment mostly supervised classification are implemented (e.g., Support Vector Machines, Gaussian Naïve-Bayes, Neural Networks, and many others), the plugin also supports unsupervised classification methods – like K-Means Clustering, and it is available for further implementations, thank to our handy programmer's guide.

3. **Flexible Data Handling**: the plugin can handle different types of data commonly used in supervised classification for earth observation, like satellite imagery and remote sensing data. It provides ample functionalities to load and preprocess the input data, including feature extraction and data splitting.

4. **Result Visualization and Assessment**: the application is able to provide interactive maps and statistical metrics to better evaluate and understand the accuracy of the results, so that the user can take more informed decisions.

5. **Documentation and Support**: the project wants to provide comprehensive documentation to everybody approaching the plugin, be them a perspective user or someone who wants to enhance it, address issues, or expand its functionalities.

1.3 Scope

The goal of this project is to create a classification tool utilizing machine learning methods for remote sensing data. Users of the tool are able to create rule images, classify image data using the tool, and train models. It supports classification techniques like Gaussian Naive Bayes and K-Means.

Large datasets are processed individually by the tool, which also offers updates on its progress during the training and classification phases. It also has features such as log reporting, accuracy evaluation, and model storing.

The project's goal is to offer an efficient and practical approach for classifying land cover in remote sensing applications.

1.4 Tools and Resources Used

1.4.1 GDAL

The GDAL library is used by the Supervised Classification plugin to increase classification effectiveness. An open-source library called GDAL provides capabilities for manipulating geographical data.

GDAL is crucial to the plugin in the following areas:

1. **Data loading and access**: GDAL offers methods for importing and using geographic datasets, such as satellite images. The plugin easily reads and extracts data using GDAL. Data transformation and manipulation capabilities are provided by GDAL for preparing and modifying input data. Operations like reprojection, resampling, cropping, and data subset extraction fall under this category
2. **Support for Data Formats**: GDAL enables users to deal with a variety of sources by supporting a number of geographic data formats. The plugin makes use of GDAL's format support for simple workflow integration with classification.
3. **Performance Optimization**: GDAL's optimized algorithms and data structures enhance classification speed, even for large datasets. This optimization improves efficiency in analyzing and classifying geospatial data.

By incorporating GDAL, the Supervised Classification plugin provides advanced capabilities, performance optimizations, and a smoother workflow, enhancing user experience and productivity.

1.4.2 scikit-learn

Scikit-learn, a powerful machine learning package, is employed by the Supervised Classification plugin to improve the classification procedure. For data analysis and modeling, scikit-learn offers a comprehensive set of tools and algorithms for data analysis and modeling.

Scikit-learn is essential to the plugin's operation in the following areas:

1. **Algorithm Selection and Training**: scikit-learn provides a variety of classification techniques, including support vector machines, random forests, and decision trees. These algorithms are used by the plugin to

train classification models with user-specified parameters.

2. **Model Evaluation and Validation**: scikit-learn offers tools for assessing and testing the effectiveness of categorization model performance. These features are used by the plugin to evaluate the precision and dependability of the trained models.

3. **Predictive Analysis**: Once trained, scikit-learn's classification models can be used to predict the classes of incoming input data. This ability is used by the plugin to categorize geospatial data using the taught models.

4. **Integration and Compatibility**: scikit-learn is a favoured option for machine learning jobs because of its smooth integration with other scientific libraries in the framework of Python. To provide seamless integration and interoperability with other elements of the classification workflow, the plugin makes use of scikit-learn's compatibility.

The Thematic Classification plugin uses scikit-learn to give users access to a variety of classification methods, model evaluation tools, and prediction capabilities, enabling accurate and efficient geospatial data classification.

1.4.3 Plugin Builder

Plugin Builder had the following role in the creation of the Thematic Classification plugin:

1. **Project Structure**: PB assists in building a well-structured project with specified directories and files, laying the groundwork for plugin development.

2. **Plugin Template**: PB provides a plugin template that contains the required template code, enabling developers to easily begin constructing the plugin without having to start from scratch.

3. **Plugin Configuration**: PB offers a user-friendly interface for setting plugin metadata, defining dependencies, and specifying plugin settings, which helps with plugin configuration.

4. **Code Generation**: To save time and effort during development, Plugin Builder automates the creation of code snippets for plugin functionality such menu items, toolbar buttons, dialogs, and event handling.

5. **Documentation Generation**: PB assists in the generation of documentation templates, making it simpler to record the features, applications, and API of the plugin for future use.

By utilizing Plugin Builder's features, the Thematic Classification plugin's development process is made more simplified, effective, and maintainable. This frees up developers to concentrate on creating the plugin's essential features.

1.4.4 Qt Builder

The Thematic Classification plugin uses Qt Designer, which comes together with the QGIS software, in the following ways:

1. **Visual interface Design:** Qt Designer is a potent tool for designing graphical user interfaces (GUIs). It offers a drag-and-drop interface for adding and organizing UI elements, allowing us to design and alter the plugin's aesthetic appearance.

2. **Widget Customization**: Qt offers a large selection of pre-built widgets that may be quickly modified to match th desired design. It offers possibilities to change widget settings, specify event handling, allowing us to design a user interface that is both aesthetically pleasing, effiicient and user-friendly.

3. **Signal-Slot system**: The signal-slot system enables smooth communication between UI elements and underlying functions. Qt makes it easier to implement this mechanism by easing the way in which user actions, like button clicks or menu selections, are linked to the associated plugin actions and functions.

4. **Integration with Qt Framework:** The Qt framework, on which the creation of the plugin is based, is smoothly integrated with Qt Designer. The UI design developed in Qt Designer and the code used in Qt-based apps are compatible and consistent thanks to this connection.

5. **Simple Synchronization Design Process**: We can quickly prototype, test, and improve the plugin's interface thanks to Qt Designer's support for an iterative design approach. It is simple to sync changes made in Qt Designer with the code base, enabling productive cooperation between designers and engineers.

We may design a visually beautiful and simple user interface for the Supervised Classification plugin using Qt Designer.

2 Dataset Description

The geospatial datasets that the Supervised Classification plugin works with often contain raster data, such as satellite imagery or data from remote sensing. These databases gather data about the surface of the planet and offer important insights for a variety of applications, such as the classification of land cover, vegetation study, urban planning, and environmental monitoring.

The geospatial datasets that the Supervised Classification plugin works with often contain raster data, such as satellite imagery or data from remote sensing. These databases gather data about the surface of the planet and offer important insights for a variety of applications, such as the classification of land cover, vegetation study, urban planning, and environmental monitoring.

Geospatial datasets often include the following essential components:

Geospatial datasets are fundamentally spatially referenced, which means they include details on the location and size of the collected data. For the data to be correctly analyzed and understood in its geographic context, this information is essential.

Spectral Bands: Sensors that record data in a variety of spectral bands are frequently used to acquire satellite pictures and remote sensing data. Each band denotes a particular range of wavelengths and offers particular knowledge about the Earth's surface. These spectral bands may include the visible, NIR, SW, and TIR bands.

Pixel Values: The data in the dataset is represented as pixel values, which are the values for each spectral band that have been measured or estimated at a particular place. Depending on the data pre-processing techniques used, these pixel values can range from digital numbers to radiance or reflectance values.

Metadata: Geospatial datasets frequently have metadata, which gives extra details about the dataset, such as the date of capture, sensor characteristics, the coordinate reference system, and any quality assessment metrics that may be provided. Understanding the data's origin, quality, and constraints requires this metadata.

It is crucial to adequately preprocess and prepare the data before using it for classification. To maintain consistency and compatibility across various datasets, this may involve operations like data normalization, atmospheric correction, geometric correction, and spatial resampling.

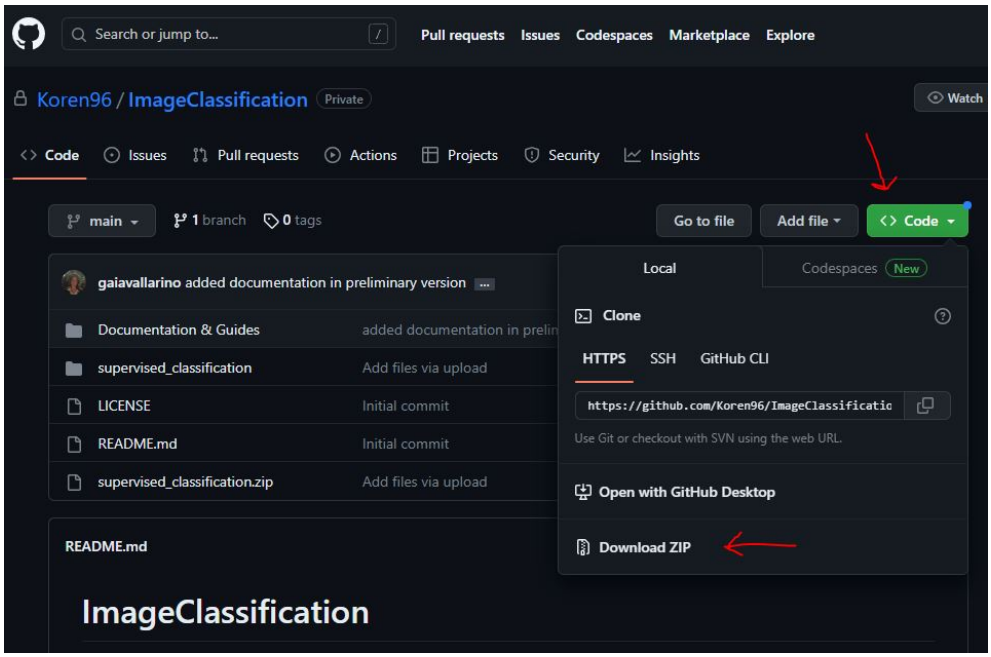
WRITE SOMETHING ABOUT SENTINEL IMAGES / PORTAL WHERE WE GOT THEM / COUPLE OF SCREENSHOTS OF THE PORTAL / FORMAT OF IMAGES

3 Plugin

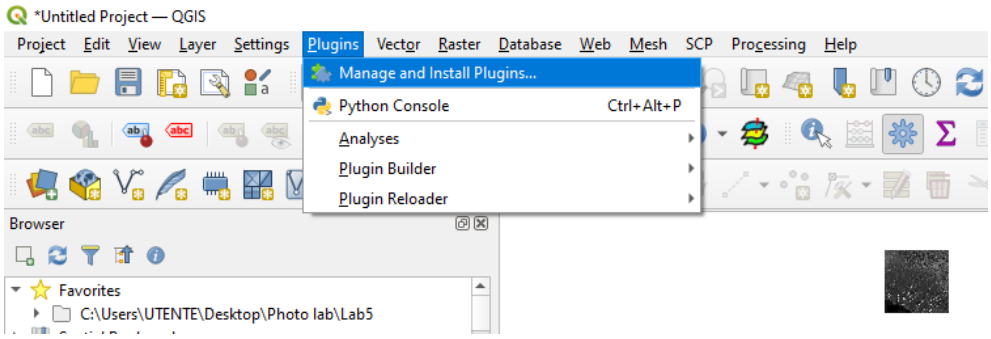
3.1 Installation Instructions

Please use the following methods to install the supervised classification plugin:

1. How to Download the Plugin:
    - Visit the GitHub repository at: `https://github.com/Koren96/ImageClassification`
    - To download the complete repository as a.zip file, click on "Download ZIP" from the "Code" menu.
    - Save the .zip file onto your computer.

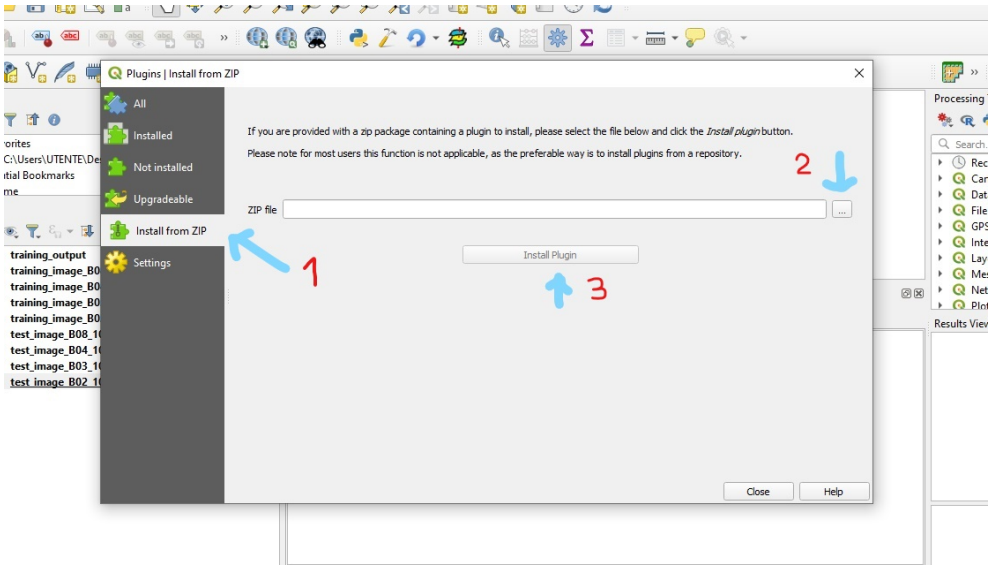


2. QGIS Plugin Manager launch:
    - Start QGIS
    - Access the Plugin Manager, from Plugins menu -¿ Manage and Install Plugins



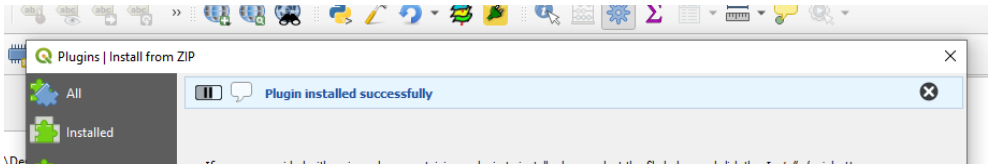
3. Installing from ZIP file:

   - Find the "Install from ZIP" section in the Plugin Manager. (1)
   - Next to the input field, click the "..." button. Browse to and choose the.zip file from step 1's download. (2)
   - To start the installation procedure, click on "Install". (3) If the installation procedure was successful, you will see a blue bar on top of the dialog window, as shown in the figure below.



4. Taking Care of Installation Errors:

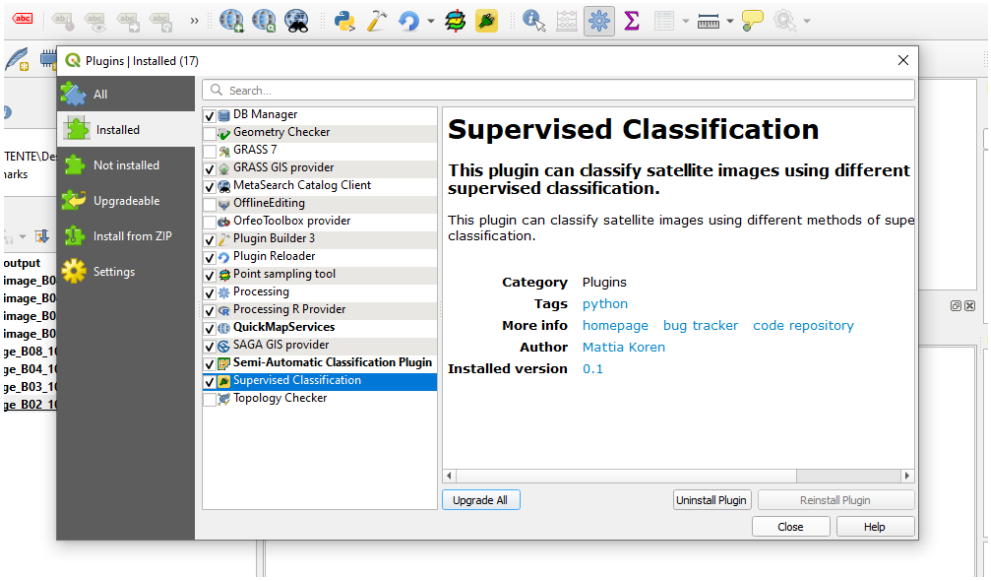
   - If an installation issue occurs, it can be due to a dependency or missing library. Make sure your system has all the necessary libraries installed.



- You can install the dependencies from the *OSGeo Shell* (you can find it in the same folder where QGIS was installed) or from the *command line*. Further information about this can be found in the paragraph **Requirements**

5. Open the Plugin:

   - Close the Plugin Manager once the installation is complete. If you want to double-check the successful installation, you can go on the "Installed" tab of the plugin manager. It should be in the list of plugins.



- Go to the "Plugins" area of the QGIS menu and choose "Thematic Classification" from the list of available plugins.
- The Thematic Classification plugin may now be used to carry out classification tasks inside of QGIS.

3.1.1 Requirements

In order to avoid installation issues, here is a list of required libraries and modules that need to be present before installing the plugin:

Name	Version	Description
scikit-learn	1.0	Description of library 1
Library 2	2.3.1	Description of library 2
Library 3	0.7.2	Description of library 3

3.2 Plugin Overview and GUI Guide

In the above figures you can see the main view of the Thematic Classification plugin.

In the following paragraph the different widgets and buttons and their functioning will be explained in detail. The description will proceed in the order in which the widgets appear.

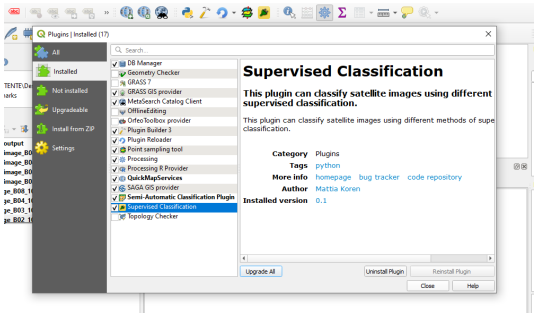
1. **Import Classification Model**: this is an optional widget that allows the user to use a pre-trained classification model, in case they already have one they want to use on the classification they are about to perform. Clicking on this button will enable the "Select a Model" function underneath, which would be otherwise non-selectable.

   Attention

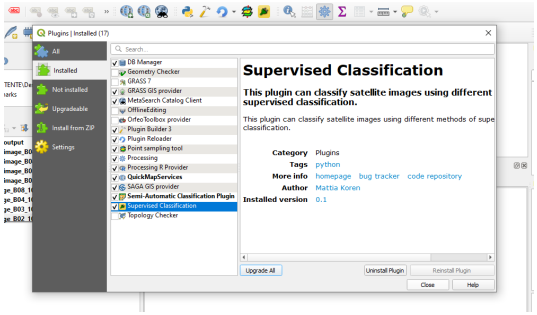
   If this option is selected, the other options discussed below, up to "Output Classification Image" will be non-selectable

2. **Input Image Bands**: this is a mandatory selection widget, which allows the user to choose among the imported layers on QGIS which ones the classification is going to be performed on. The "Refresh" button next to the widget allows for new layers to be added to the pull-down list, in case any were added while the plugin window was open.

   - Null Image Value: is an optional function that allows to select which value in the input images will be recognized as "null" in the image

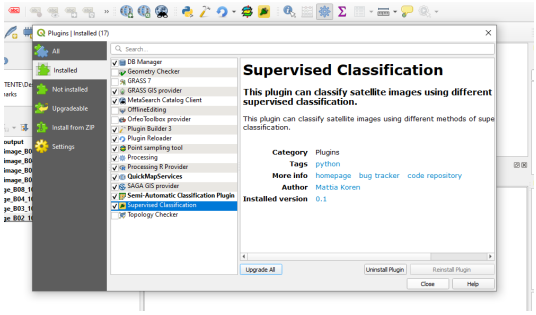


- Class for Null Image Value: is an optional function that allows the user to select which class the null value are mapped to.

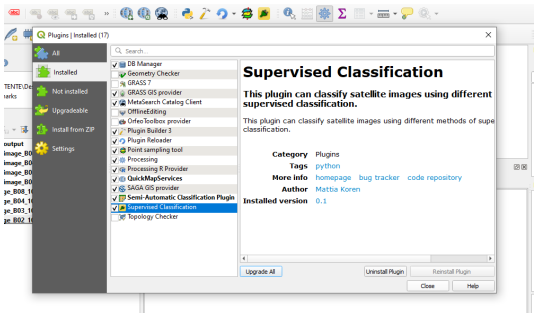


3. **Input Training Data / ROI**: In this widget it is possible to select the layer with already classified samples that will be used to perform the supervised classification in case a pre-existing model was not selected. This layer will have to be open in QGIS.

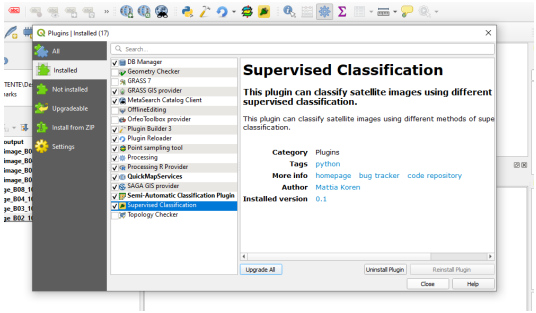
   - Null ROI Value: if in the classification there were any samples of null classes, this is where the user can input the relevant value in said layer.



4. **Output Classification Image**: in this part of the GUI, the user can select where the output classified image will be saved. The output will be saved in .tiff format.



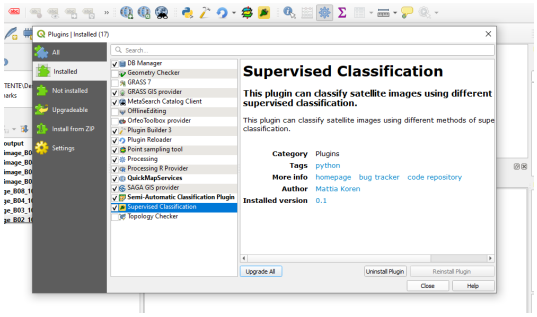
5. **Load Input Image Using Tiles**: this is an optional command. If the user selects it, they can choose how many rows and columns to divide the input image in when it gets loaded into memory. This is done to ensure a smooth and efficient pre-processing of the image, in particular in case of big files, which would otherwise be quite computationally heavy.



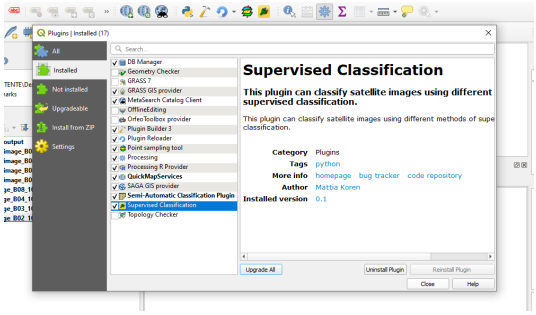
6. **Classify Input Image Using Tiles**: this is an optional command. If the user selects it, they can choose the number of tiles to divide the file in when going through the classification phase. This is done to ensure a smooth and efficient classification of the image, in particular in case of big files, which would otherwise be quite computationally heavy.

7. **Select Classification Algorithm**: in this section, the user is prompted to select a classification method.

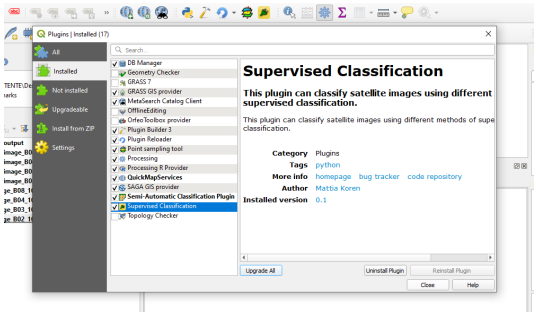
   - The user may choose between unsupervised and supervised methods.
   - The user is able to review the documentation of the method on the scikit-learn page, via the "Help" button that becomes clickable once an algorithm is selected.



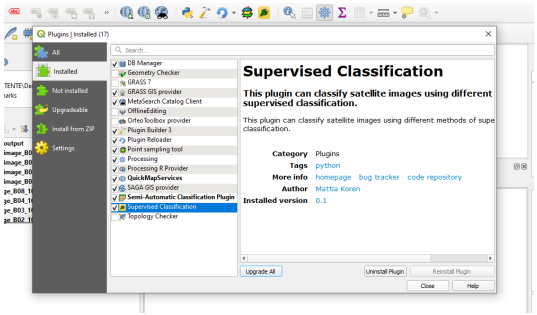
- The user may choose to use custom parameters in the selected algorithm, by clicking the "Use Custom Classification Parameters" button, and then switching the parameters to be employed.



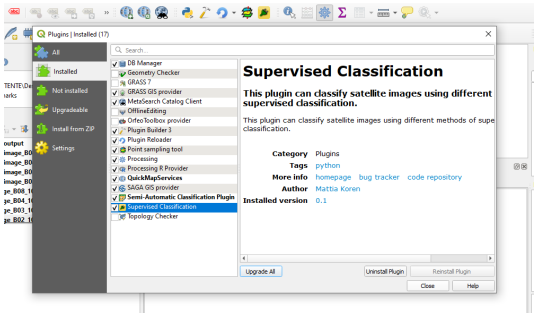
8. **Calculate Classification Model Accuracy**: by selecting this option, the user will receive a percentage value of accuracy of its classification model. This is possible only if the user has not selected to use a pre-existing model.



9. **Save Classification Model**: by selecting this button, the user will save the classification model that was just trained. It will be saved as a .joblib file.



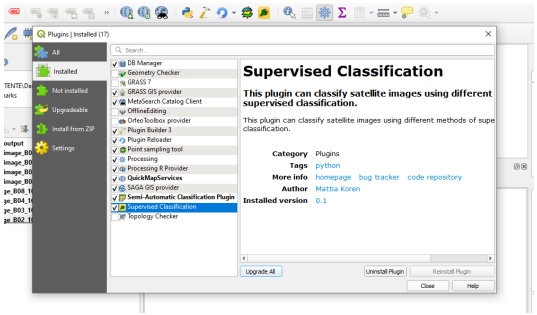
10. **Save Rule Images**: this button allows the user to generate and save the rule images for the classification that is taking place. A rule image contains values of probability of each pixel of belonging to a certain class. It is a useful tool in case the user wants to have a more in-depth look of why one pixel was classified under a certain class.



11. **Log**: the Log panel shows certain clou aspects of the process, such as the timesteps for the different processes, an overview of the parameters used, and much more useful information. Via the "Save Log (.txt)?" button, the user can decide to save the log file, which might come useful later on to remember what was set during a particular run of the application.

12. **Progress Bar**: the progress bar is implemented for the user to see the progress of the process once the OK button is pressed and the code is running.

13. **Message Bar**: the message bar only comes out when a critical error or a status update happens, after pressing OK. The critical errors may be reviewed at the section "Common Errors" of this document.



3.3 Workflow Examples

3.3.1 Common Workflows (Use Cases)

3.3.2 Common Errors

4 Support and Contacts

In case you have any doubts or questions about the project, you can find us at the following email addresses:

Name	**Email Address**
Mattia Koren	mattia.koren@gmail.com
Gaia Vallarino	gaiavallarino1@gmail.com

The repository for this project can be found at this page:

name of repo

A Definitions, Acronyms, Abbreviations, Commonly-used Terms

B Case Study

C Programmer's Guide

What you can find in the following pages is the guide we have built that explain the code and its functionalities. We have decided to include this tool in order to increase its expandability, in case anybody will ever decide to enhance its functionalities.

The separate document, as well as the source jupyter notebook, can be found in the same folder as this guide.