POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Master of science in Geoinformatics Engineering Dipartimento di Ingegneria Civile e Ambientale

Design and Development of an Application for Remote Sensing the Water Quality of the Insubric Lake

Supervisor: **Prof. Maria Antonia Brovelli**
Co-supervisor: **Dr. Lorenzo Amici**

Thesis by:

Ali Badr Eldin Ali Mohamed	10760530
Mohammed Abd Alslam Mohammed Elkhalifa	10776702

Academic year 2022 - 2023

Contents

CHAPTER 1: INTRODUCTION 10

1.1 General Overview 10
1.1.1 Water use and quality standards 11
1.1.2 Water's role in climate change 12
1.1.3 Water pollutants 13

1.2 SIMILE Project 16
1.2.1 SIMILE Data 18
1.2.2 The SIMILE Business intelligence platform 19

1.3 Thesis Objective 21

1.4 Thesis Outline 21

CHAPTER 2: AREA OF INTEREST AND DATA 22

2.1 Area of Interest 22
2.1.1 Lake Como 22
2.1.2 Lake Lugano 24
2.1.3 Lake Maggiore 25
2.1.4 Lake Varese 26

2.2 Data 28
2.2.1 Satellite Observation 28
2.2.2 Additional Dataset 36

CHAPTER 3: PROCEDURE DESIGN 41

3.1 Pre-processing 44

3.2 Processing Algorithms 44
3.2.1 Algorithmic Approach for Sentinel-3 OLCI (Chlorophyll-a and Total Suspended Matter Mapping) 44
3.2.2 Algorithmic Approach for Landsat-8 TIRS Land Surface Water Temperature Mapping 56

3.3 Post-processing 59
3.3.1 Co-registration 59
3.3.2 Outlier Rejection 60

CHAPTER 4: IMPLEMENTATION 61

4.1 Tools and Technologies 61
4.1.1 Software Tools 61
4.1.1.1 Docker Desktop 62
4.1.1.2 Xming 63
4.1.2 Python Libraries 65
4.1.3 Copernicus Hub 70
4.1.4 USGS 71

4.2 Graphical user interface 71
4.2.1 Run the GUI (Graphical User Interface) 71
4.2.2 Additional Dataset Insertion 71
4.2.3 Download Sentinel-3 Imagery 73
4.2.4 Create Maps 75

CHAPTER 5: RESULTS 77

5.1 Output Maps 77
5.1.1 Chlorophyll-a (CHL) 78
5.1.2 Total Suspended Matter (TSM) 82
5.1.3 Co-registered Images 85
5.1.4 Land Water Surface Temperature (LWST) 90
5.1.5 Example of good quality Maps 92
5.1.5.1 Chlorophyll-a (CHL) 92
5.1.5.2 Total Suspended Matter (TSM) 96

5.2 Statistics 99

CHAPTER 6: CONCLUSION AND FUTURE WORK 106

6.1 Conclusion 106

6.2 Future work 107

BIBLIOGRAPHY 108

APPENDIX 115

Code 115
The Graphical User Interface (GUI) 115

Download Sentinel 3. 122
Additional Parameters 125
Production of CHL-a and TSM Maps........ 129
Production of LWST 136

ACKNOWLEDGEMENTS 139

List of Figures

Figure 1. Location of the study area, CRS: WGS 84/Pseudo-Mercator........ 17
Figure 2. Schema of the SIMILE project [78] 19
Figure 3. Location of Lake Como. 23
Figure 4. Location of Lake Lugano. 24
Figure 5. Location of Lake Maggiore........ 26
Figure 6. Location of Lake Varese........ 27
Figure 7. Temperature Profile 38
Figure 8. Location of Tremezzo station........ 39
Figure 9. Atmospheric Correction Parameters Calculator [72] 40
Figure 10. Flowchart for WQPs Mapping........ 42
Figure 11. C2RCC Bio-optical model [62]........ 46
Figure 12. C2RCC processor architecture. Neural nets are shown as green boxes, and quality tests as blue boxes [62]........ 48
Figure 13. C2RCC SNAP Plugin Window [67] 50
Figure 14. Radiative transfer equation [72]. 57
Figure 15. Build the image. 63
Figure 16. Docker Compose File. 64
Figure 17. Start Xming. 65
Figure 18. Additional Dataset Insertion Page 73
Figure 19. Download the Sentinel-3 page........ 74
Figure 20. Create maps 76
Figure 21. Chlorophyll-a (CHL) before outlier rejection (04/01/2023)........ 78
Figure 22. Chlorophyll-a (CHL) after outlier rejection (04/01/2023)........ 79
Figure 23. Chlorophyll-a (CHL) before outlier rejection (22/01/2023)........ 80
Figure 24. Chlorophyll-a (CHL) after outlier rejection (22/01/2023)........ 81

Figure 25. Total Suspended Matter (TSM) before outlier rejection (04/01/2023)............................82
Figure 26. Total Suspended Matter (TSM) after outlier rejection (04/01/2023)...............................83
Figure 27. Total Suspended Matter (TSM) before outlier rejection (31/01/2023)............................84
Figure 28. Total Suspended Matter after Outlier Rejection (31/01/2023) 85
Figure 29. Co-registered Chlorophyll-a (CHL) Concentration Map (04/01/2023) 86
Figure 30. Co-registered Chlorophyll-a (CHL) Concentration Map (10/01/2023) 87
Figure 31. Co-registered Total Suspended Matter (TSM) Concentration Map (10/01/2023). 88
Figure 32. Co-registered Total Suspended Matter (TSM) Concentration Map (07/02/2023). 89
Figure 33. Land Water Surface Temperature (LWST) Map before outlier rejection (18/02/2023).90
Figure 34. Land Water Surface Temperature (LWST) Map after Outlier Rejection (18/02/2023) .91
Figure 35. Chlorophyll-a Map (02/02/2023)..92
Figure 36. Chlorophyll-a Map (07/02/2023)..93
Figure 37. Chlorophyll-a Map (10/01/2023)..94
Figure 38. Chlorophyll-a Map (14/02/2023)..95
Figure 39. Total Suspended Mater Map (02/02/2023) 96
Figure 40. Total Suspended Mater Map (10/01/2023) 97
Figure 41. Total Suspended Mater Map (14/02/2023) 98
Figure 42. Chl-a outlier presence 100
Figure 43. TSM outlier presence...101
Figure 44. Map's boxplot...102
Figure 45. Samples of Chl-a and TSM Parameters 105

List of Tables

Table 1. Morphometric and hydrological characteristics of the four Lakes in Northern Italy. 28
Table 2. Overview of the source data for WQP maps [74] 29
Table 3. OLCI Band characteristics [57] 34
Table 4. Landsat 8 Band characteristics [56]..36
Table 5. Training ranges of the neural nets [62]...47
Table 6. The output L2 products are generated by C2RCC [63]...56
Table 7. Python Libraries 70
Table 8. Images used to run the Application. 77
Table 9. Chl-a statistics 103
Table 10. TSM statistics 104

Listings

Listing 1. Design and Creation of Graphical User Interface (GUI) 121

Listing 2. Function that used to Download Sentinel-3 imagery. 124

Listing 3. Function that used to Download, Insert the Temperature and Atmospheric correction parameters.. 128

Listing 4. Function that used to generate CHL-a and TSM Maps 135

Listing 5. Function that used to generate LWST Maps 138

Abstract

The availability of clean and safe water is essential for human consumption, irrigation, recreational activities, and ecosystem health. In this study, we address the crucial issue of monitoring water quality in lakes to determine its suitability for various purposes in the framework of the SIMILE Interreg project, thus taking a step forward toward the 6th Sustainable Development Goal (SDG) of the United Nations ("Ensure availability and sustainable management of water and sanitation for all"). The design and development of an application that leverages remote sensing technology implemented for the production of Water Quality Parameter (WQP), which are obtained by processing Sentinel-3 OLCI and Landsat-8 TIRS images. The processing chain includes the use of the C2RCC processor to obtain Chl-a (Chlorophyll-a) and TSM (Total Suspended Matter) and the Barsi method to produce maps of water surface temperature. The maps underwent a filtering process to remove anomalous values resulting from various factors, including clouds, water reflection (such as glint), and mixed pixels. These filtered maps were then compared to in-situ data for further analysis and validation. The filtering procedure involved outlier rejection based on the 3 σ rule. Values that were identified as local anomalies underwent additional scrutiny to assess potential local patterns, such as the presence of small gulfs and inflow/outflow streams. The idea of making the application as automated and guided as possible is to foster the production of WQP maps. Constant and timely monitoring allows for the early detection of pollution sources, the identification of emerging contaminants, and the assessment of the effectiveness of pollution control measures. Also, it helps guide policy decisions, resource allocation, and the development of strategies to protect and conserve water resources.
The processing workflow, which involved the analysis of Water Quality Parameter (WQP) maps for January and February 2023, adopted the same methodology as in SNAP. However, a notable distinction was the automation of the process, achieved through the utilization of the Python programming language. In general, the application was primarily developed using the Docker Python container, providing a robust and scalable environment. To enhance usability and accessibility, a user-friendly monitoring application was designed, featuring a graphical user interface (GUI) and incorporating advanced remote sensing capabilities. This integration of RS technologies enables non-expert operators to effectively monitor and manage water quality, simplifying the overall monitoring process.
Keywords: Water Quality Parameters (WQP), Co-registration, Graphical User Interface, Chlorophyll-a, Total Suspended Matter, Land surface water temperature.

Sommario

La disponibilità di acqua pulita e sicura è essenziale per il consumo umano, l'irrigazione, le attività ricreative e la salute dell'ecosistema. In questo studio affrontiamo la questione cruciale del monitoraggio della qualità dell'acqua nei laghi per determinarne l'idoneità a vari scopi nell'ambito del progetto SIMILE Interreg, facendo così un passo avanti verso il 6° Obiettivo di Sviluppo Sostenibile (SDG) delle Nazioni Unite ("Garantire la disponibilità e la gestione sostenibile dell'acqua e dei servizi igienico-sanitari per tutti").
La progettazione e lo sviluppo di un'applicazione che sfrutta la tecnologia di telerilevamento implementata per la produzione di Water Quality Parameter (WQP), che si ottengono elaborando le immagini Sentinel-3 OLCI e Landsat-8 TIRS. La catena di elaborazione include l'uso del processore C2RCC per ottenere Chl-a (Chlorophyll-a) e TSM (Solidi Sospesi Totali) e il metodo Barsi per produrre mappe della temperatura superficiale dell'acqua. Le mappe sono state sottoposte a un processo di filtraggio per rimuovere valori anomali risultanti da vari fattori, tra cui nuvole, riflessi d'acqua (come glint) e pixel misti. Queste mappe filtrate sono state quindi confrontate con i dati in situ per un'ulteriore analisi e convalida. La procedura di filtraggio prevedeva il rifiuto dei valori anomali basato sulla regola 3 σ. I valori che sono stati identificati come anomalie locali sono stati sottoposti a un ulteriore controllo per valutare potenziali modelli locali, come la presenza di piccoli golfi e flussi di afflusso/deflusso. L'idea di rendere l'applicazione il più possibile automatizzata e guidata è quella di favorire la produzione di mappe WQP. Il monitoraggio costante e tempestivo consente la diagnosi precoce delle fonti di inquinamento, l'identificazione dei contaminanti emergenti e la valutazione dell'efficacia delle misure di controllo dell'inquinamento. Inoltre, aiuta a guidare le decisioni politiche, l'allocazione delle risorse e lo sviluppo di strategie per proteggere e conservare le risorse idriche.
Il flusso di lavoro di elaborazione, che ha comportato l'analisi delle mappe dei parametri di qualità dell'acqua (WQP) per gennaio e febbraio 2023, ha adottato la stessa metodologia di SNAP. Tuttavia, una notevole distinzione è stata l'automazione del processo, ottenuta attraverso l'utilizzo del linguaggio di programmazione Python. In generale, l'applicazione è stata sviluppata principalmente utilizzando il contenitore Docker Python, fornendo un ambiente robusto e scalabile. Per migliorare l'usabilità e

l'accessibilità, è stata progettata un'applicazione di monitoraggio di facile utilizzo, dotata di un'interfaccia utente grafica (GUI) e che incorpora funzionalità avanzate di rilevamento remoto. Questa integrazione delle tecnologie RS consente agli operatori non esperti di monitorare e gestire efficacemente la qualità dell'acqua, semplificando l'intero processo di monitoraggio.

Parole chiave: Parametri di qualità dell'acqua (WQP), Co-registrazione, Interfaccia utente grafica, Clorofilla-a, Solidi Sospesi Totali, temperatura della superficie dell'acqua terrestre.

Chapter 1: Introduction

In this chapter, a general contextual overview, the SIMILE project, the thesis objectives, and the thesis structure will be presented.

1.1 General Overview

Water is a vital resource that sustains life on Earth. It covers approximately 71% of the planet's surface [1] and is essential for the existence of all living organisms, from microorganisms to human beings [2]. The chemical composition of water consists of two hydrogen and one oxygen atoms, water forming (H2O) molecules [3]. The unique properties of water make it an incredibly versatile substance, capable of existing in all three states of matter: solid (i.e., ice caps), liquid (i.e., lakes), and gas (water vapor) [1]. Additionally, water acts as a universal solvent due to its ability to dissolve a wide range of substances. Our planet is known as the 'water planet' due to the abundance of water found on its surface, with oceans, seas, rivers, lakes, and glaciers all constituting this resource [4]. From supporting ecosystems and agriculture to powering industries and enabling human survival, water plays a crucial role in various aspects of our lives. Understanding the importance of water, its usage, and the need for its protection from pollutants is essential for ensuring a sustainable and healthy environment.

It is estimated that approximately 97.5% of the total water on Earth is saline, while the remaining 2.5% is freshwater [1]. However, within that freshwater portion, approximately 68.7% exists in frozen form as ice and glaciers, about 30.1% is groundwater, and only around 1.2% is available on the surface of the Earth in the form of lakes and rivers [1]. Therefore, it is crucial to ensure that water used for human consumption is "safe," i.e., characterized by being liquid (i.e., odorless, colorless, and pleasant tasting), while also being free from hazardous chemical agents and pathogens [8].

This study specifically focuses on lakes, which comprise a small fraction of the Earth's freshwater resources.

1.1.1 Water use and quality standards

Water is not only vital but also a precious and irreplaceable resource that is used in diverse ways across different sectors, playing a crucial role in sustaining life and supporting various human activities [4]. In households, water is essential for drinking, cooking, sanitation, and personal hygiene. Industries rely on water for manufacturing processes, cooling systems, and energy production. Agriculture heavily depends on water for irrigation to grow crops and sustain livestock [1]. Additionally, water is used for recreational purposes, transportation, and maintaining the ecological balance in rivers, lakes, and oceans.

The Sustainable Development Goals (SDGs), established by the United Nations [5], outline a comprehensive framework for global development. Ensuring the quality of water is crucial for achieving these goals and promoting human health, protecting ecosystems, and supporting sustainable water resource management. Water quality standards and guidelines (i.e., World Health Organization Standards) play a vital role in this endeavour, as they are established to protect human health, preserve ecosystems, and maintain the overall integrity of the environment [6].

By preserving and enhancing water quality in accordance with WHO guidelines, we contribute to SDG 6 (Clean Water and Sanitation), SDG 14 (Life Below Water), and SDG 15 (Life on Land), which explicitly recognize the significance of clean water, conservation of aquatic ecosystems, and sustainable land and water management.

Water quality standards ensure that water is safe and suitable for various purposes, such as drinking, agriculture, industry, and recreational activities. Safe drinking water is essential for human health, preventing waterborne diseases, and ensuring the well-being of communities. By adhering to water quality standards for drinking water, we meet SDG 6, which aims to ensure access to clean water and sanitation for all.

Moreover, maintaining water quality is crucial for preserving aquatic ecosystems, as highlighted by SDG 14. Clean and healthy water bodies are essential for the survival of aquatic life, including fish, plants, and other organisms. By adhering to water quality standards, protecting biodiversity, maintaining the balance of aquatic ecosystems, and supporting sustainable fisheries and marine life conservation.

Furthermore, water quality standards contribute to SDG 15, which focuses on sustainable land and water management. By ensuring that water resources are free from pollutants and contaminants, protecting soil quality, preserving natural habitats, and promoting sustainable agricultural practices. Healthy water bodies also play a role in mitigating the impacts of climate change, such as flooding and droughts, by serving as natural buffers and reservoirs.

Preservation of water quality standards is key to achieving the SDGs related to clean water, conservation of aquatic ecosystems, and sustainable land and water management. Adhering to these standards safeguards human health, protects ecosystems, and ensures the availability of clean water resources for present and future generations.

1.1.2 Water's role in climate change

Water is intricately linked to climate change as both a driver and an impact of global environmental shifts. Water is the only substance that exists on Earth in each of its three states and easily changes from one state to another [1]. Water, never sitting still, always changes its location (i.e., continental scale) by changing state from liquid to vapor to ice and back again in a continuous pattern called the water cycle or the hydrologic cycle, leading to altered precipitation patterns, increased evaporation, and changes in the availability and quality of water resources [7].

Water quality is predicted to deteriorate as temperatures rise. Such circumstances promote the spread of poisonous algae and bacteria, affecting the health of aquatic ecosystems because aquatic organisms have a specific range of temperatures they can tolerate, exacerbating the problem of water shortages, which is primarily the result of human activities. Temperature rises contribute to increased water evaporation and the melting of glaciers and ice caps, resulting in sea-level rise and probable flooding of coastal communities. Changes in rainfall patterns can cause droughts or heavy rains, affecting agriculture, water supply, and ecosystems [8].

1.1.3 Water pollutants

The protection of water sources from pollutants is crucial to safeguarding human health and preserving ecosystems. Industrial and agricultural activities, as well as urbanization, have led to the contamination of water bodies with pollutants such as chemicals, heavy metals, fertilizers, pesticides, and sewage. These pollutants can have detrimental effects on aquatic life and can enter the food chain, posing risks to human health. Therefore, implementing effective pollution control measures and promoting sustainable practices are necessary to prevent water pollution and its adverse consequences.

As indicated by Madhav [8], there are diverse categorizations of water pollution. The two principal sources of water contamination can be observed as points and non-points.

1) Point indicates the contaminants that belong to a line supply. An example of this would be effluent discharge from industries in water bodies;

2) Nonpoint, in contrast, means contaminants come from many sources. Polluted water after rain that has passed through numerous areas may also be believed to be a non-point source of contamination.

Natural groundwater pollution is chiefly due to geological configuration with shallow groundwater masses, percolation from low-quality surface water bodies, saltwater invasion, or the consequences of geothermal fluids. Anthropogenic water pollution is normally attributed to the intense application of farming pesticides and manures, mining desecration, dumping of industrial effluents, waste dumping locations, and defective good formation [12]. The ionic composition of groundwater is governed by many natural as well as human-induced aspects. Natural factors have power over water composition, including the precipitation model and quantity, a geological attribute of the watershed and aquifer, meteorological factors, and diverse rock-water interaction courses in the aquifer [13, 14]. Human actions that manipulate the chemical composition of water comprise discarded solid devastation, household and industrial desecration, and mining and farming actions [15]. Sources of water pollution can be either ketogenic or anthropogenic. Anthropogenic activities causing pollution can be further classified into industrial, agricultural, and domestic [16]. Despite the fact that industrial consumption of water is very low in comparison to farming purposes, unmanaged dumping of industrial waste matter on land and/or on surface water bodies makes water resources inappropriate for further uses [17, 18].

1.1.3.1 Different pollutants, their sources, and environmental impacts

According to a study conducted by Madhav et al. (2020) [8], not to enumerate:

- **Phosphate (PO_4):** PO4 is a vital nutrient for plants, animals, and humans. It is available in three forms: organic phosphorus (associated with organic molecules), orthophosphate, and polyphosphate. In a few water bodies, the value of phosphorus is low enough to limit the growth of algae and/or aquatic plants. In this matter, researchers state that phosphorus is a restrictive nutrient. Therefore, eutrophication (harmful algae growth) and water excellence are sustained by calculating the access to the PO_4 [19]. Natural disintegration of rocks and minerals, farming discharge, erosion, and sedimentation, and input by animals are nonpoint sources, while point sources are sewage wastewater and industrial effluents [20]. A little concentration of PO_4 may boost the development of algae and aquatic plants, leading to eutrophication of the aqueous ecosystem. PO_4 in water may cause algal bloom development in water bodies [21, 22];
- **Suspended solids and sediments:** Suspended solids (SS) refer to the matter, both inorganic and organic, that is held in the water column of various water bodies by instability. These solids typically consist of particles with a diameter smaller than 62 mm [23]. The consequence of SS on aquatic biota is dependent on four key factors: (1) the quantity of SS; (2) the period of contact with SS; (3) the geochemistry of SS; and (4) the particle-size allocation of SS. The presence of solids can obstruct the sun's infiltration into the water, which is necessary for the photosynthesis of benthic flora. Finer suspended solids may harm the gills of fishes and originate asphyxiation [24];
- **Thermal pollution:** the discharge of hot water from industries, such as power plants, into natural water bodies. This raises the water temperature, reduces dissolved oxygen levels, and disrupts aquatic ecosystems. The release of hot water alters the range of organisms that can survive and promotes bacterial growth. It also leads to thermal layering, with hot water accumulating at the surface [25, 26];

- **Nutrients and agricultural runoff:** The release of nutrients from agricultural runoff, fertilizer industry wastewater, and sewage into bodies of water results in eutrophication. This process stimulates the growth of algae and aquatic weeds, degrading the water quality and harming aquatic life [6].

1.1.3.2 Water monitoring

Monitoring water quality is a critical aspect of ensuring its safety and identifying potential pollution sources [6]. There are two major methods of determining the eminence of water:

- Chemical indicators involve taking water samples and analyzing the presence and concentration of various constituents. If the levels of these constituents exceed safety standards or are considered too high, the water is classified as contaminated. This method, known as chemical indicators of water quality, focuses on evaluating the chemical composition of water to determine its suitability for various purposes [14];
- Biological indicators of water quality involve studying the organisms that inhabit water bodies, such as fish, insects, and other invertebrates. By examining the diversity and abundance of these organisms, we can infer the quality of the water. If a water body supports a wide variety of living organisms, it indicates good water quality. Conversely, if a water body lacks fish or other organisms, it suggests poor water quality. This method relies on the presence and health of aquatic life as an indicator of the overall quality and ecological health of the water body [15, 16].

Constant and timely monitoring and interventions are essential to developing adequate management strategies for the water resource, as encouraged in the 17 Sustainable Development Goals (SDGs) [5]. In Goal 6, “Ensure access to water and sanitation for all,” and Goal 14, “Conserve and sustainably use the oceans, seas, and seas in resources,” due to their integrated and indivisible nature, other goals are related to water, in particular Goal 13, “Adopting urgent measures to combat climate change and its impacts”. Monitoring helps in identifying pollution hotspots, evaluating the effectiveness of pollution control measures, and guiding decision-making for water resource management. [9]

Monitoring water resources is essential for maintaining their quality, ensuring sustainable water management practices, and safeguarding human and environmental health. Regular monitoring allows for the early detection of pollution sources, the identification of emerging contaminants, and the assessment of the effectiveness of pollution control measures. It helps guide policy decisions, resource allocation, and the development of strategies to protect and conserve water resources. By monitoring water quality and quantity, stakeholders can make informed decisions to address emerging challenges, mitigate the impacts of climate change, and promote the sustainable use of this precious resource [10]

The duration of water monitoring activities can vary depending on the specific parameters being assessed (refer to Section 1.1.3.1), such as temperature, turbidity (clarity), and watercolor. These parameters define the physical, chemical, and ecological status of the water body. Short-term monitoring programs may involve frequent sampling over a few days or weeks to capture dynamic changes (parameter fluctuations) in water quality. Long-term monitoring programs, on the other hand, extend over years or even decades to understand seasonal variations, long-term trends, and the cumulative impacts of pollution on water ecosystems. The period of monitoring is governed by factors such as the size of the study, regulatory requirements, research aims, and available resources [11].

1.2 SIMILE Project

SIMILE is an integrated monitoring system for subalpine lakes and their ecosystems. The main goal is the protection of water quality for Lugano, Maggiore, and Como lakes (Figure 1), whose catchments are shared between Italy and Switzerland, so management policies should necessarily take into account the transboundary character of water issues. These lakes are the study area of SIMILE (Informative System for the Integrated Monitoring of Insubric Lakes and Their Ecosystems), which aims at improving the coordinated management and strengthening stakeholder participation in the processes of knowledge and monitoring of the water resources, thus taking a step forward toward the 6th Sustainable Development Goal (SDG) of the United Nations ("Ensure availability and sustainable management of water and sanitation for all").

The insubric lakes have water surfaces, volumes, and maximum depths ranging between 48.7 and 212.5 km^2, 6.5 and 37.5 km^3, and 288 and 410 m, respectively [27]. Monitoring of these water bodies is performed in Italy under the provisions of the WFD (Water Framework Directive), Directive 2000/60/EC. They are monitored with varying frequency according to their present condition, and collected data define the lake's chemical and ecological status. [28]

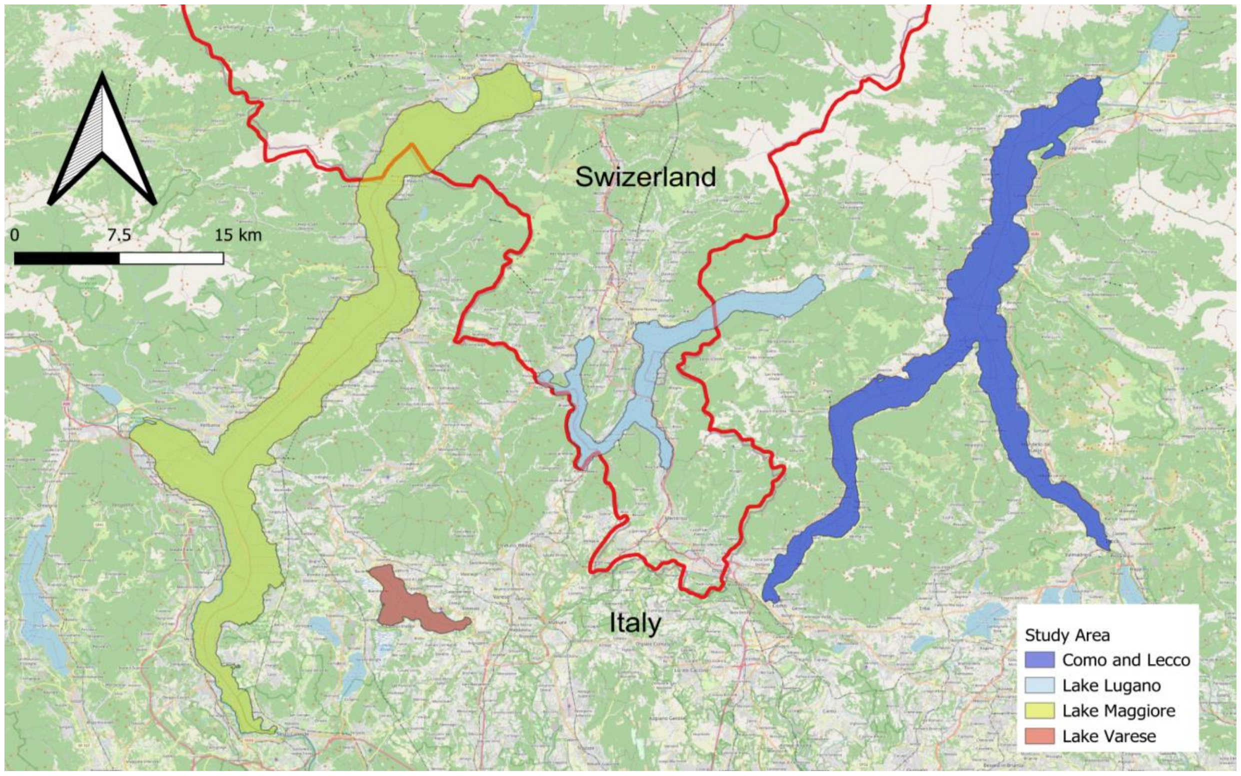


Figure 1. Location of the study area, CRS: WGS 84/Pseudo-Mercator

1.2.1 SIMILE Data

SIMILE relies on in-situ sensor data, satellite observations, and a mobile application for citizen science to collect data about lake water quality. [29]

Sensor Data: High-frequency data collection in SIMILE involves the use of in-situ sensors placed on buoys or platforms, allowing for the integration of traditional discrete monitoring of lakes. These sensors offer benefits such as high temporal resolution, consistent data collection, low cost, and wireless connectivity [30]. They are particularly useful in detecting rapid changes in physical, biological, and chemical parameters, including short-duration phenomena like algal blooms. In addition, they can serve as "early warning" systems for events that can impact water quality or cause floods [30]. However, these sensors require regular maintenance, including calibration and cleaning, and their measurements should be compared with field and laboratory methods. Fouling, extreme weather, and vandalism pose challenges and potential costs for maintenance [31].

Satellite Observations: Satellite images are utilized in lake quality monitoring to complement traditional approaches by providing high-frequency data covering entire lake areas. These images, obtained from spatial agencies' web portals, are processed using open-source algorithms to extract important Water Quality Parameters (WQPs):

- Total Suspended Matter;
- Chlorophyll-a;
- lake surface water temperature.

Sentinel-3 and Landsat 8 satellites are commonly used for data extraction, while anomalies in chlorophyll-a or Total Suspended Matter can be further examined using Sentinel-2 imagery. The WQP products are accessible to users on a collaborative platform [32]. The platform is based on two separate applications: one for WQP producers (providers) and one for WQP users.

The citizen science aspect of SIMILE is important for at least two reasons. Firstly, it helps to raise awareness among citizens about the significance of lake ecosystems, and secondly, the data collected by citizens support the analyses and monitoring of lake water quality. The activities related to citizen science in the SIMILE project were dedicated to the development of the free and open-source mobile application that allows citizens to report various issues observed in a lake by submitting pictures, filling out a simple survey, or inserting values of measurement performed by themselves [33]. Utilizing a novel strategy of combining these different data sources, modeling, and evaluating them using a business intelligence platform, i.e., a web data-driven decision support system (Figure 2), which will promote the use of such information for decision and policymaking for the public administration of the Insubric Lakes [29].

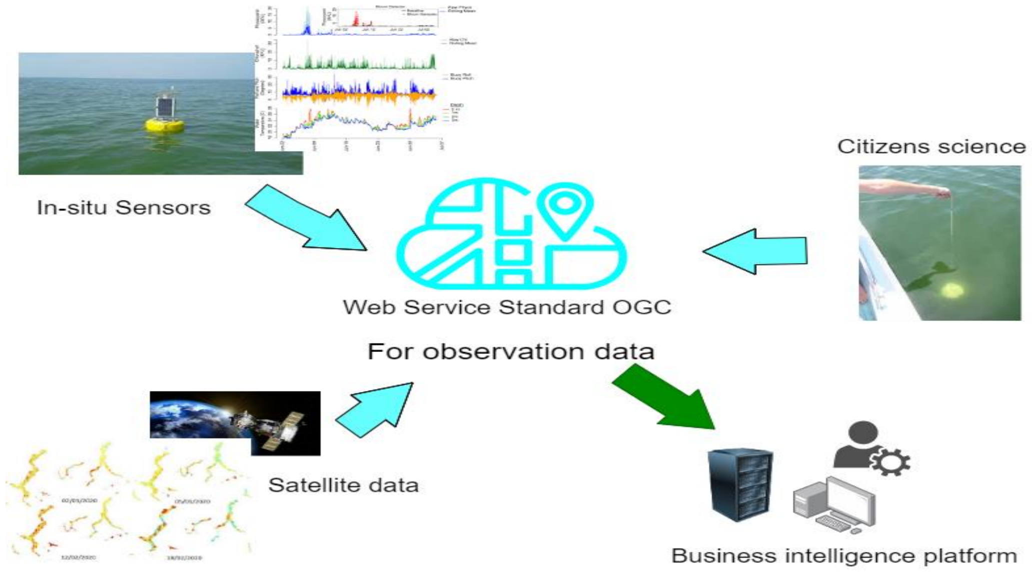


Figure 2. Schema of the SIMILE project [78]

1.2.2 The SIMILE Business intelligence platform

Water resource managers must be able to access accurate and up-to-date data in order to carry out operations successfully and plan interventions based on an adaptive approach informed by the current status and detected risks. Unfortunately, the necessary information is frequently collected using disparate methodologies, represented using highly heterogeneous data formats, and stored and disseminated on a plethora of disparate applications and solutions, many of which are only available on desktop applications. This variability impedes and frequently prevents the appropriate and optimum use of scientific information in decision-making processes. SIMILE seeks to address this issue by merging all the data described in previous sections (sensors, citizen science, and satellite) into a single platform that provides timely, accessible, updated, and targeted information to decision-makers (administrators or stakeholders). To accomplish this goal, the project will establish a Business Intelligence (BI) platform, which is defined as "a data-driven decision support system that combines data gathering, data storage, and knowledge management with analysis to provide input to the decision process" [34]. In the environmental domain, decision support systems are frequently confined to GIS systems and applications that spatially aggregate multiple layers of information. Although GISs may geographically represent phenomena, they often lack sophisticated capabilities for performing assessments of environmental data for trend detection, scenario identification, and phenomenon comprehension using visuals, tables, and indicators. SIMILE's strategy consists of implementing a variety of Open Geospatial Consortium (OGC) services (e.g., SOS, WMS, WCS, etc.) that provide standard data access. Because of the compatibility of the existing services, it will be feasible to add extensible functionality for data extraction, transformation, and loading (ETL). These functionalities have the scope of selecting and extracting from the source systems only the portions of data that are relevant, cleaning and transforming the selected data, applying appropriate pre-processing and elaborations, and finally loading the transformed data in a specific data warehouse designed to be agile and efficient.

A Web API was developed to facilitate Online Analytical Processing (OLAP) and the creation of Web user interfaces. To safeguard data and protect information, an authentication and authorization system will be used to limit access to functionality. The BI Web interface should be built to be user-type-centric, which means that depending on the function of the user, the visual aspect and information content should vary to match their individual demands quickly and efficiently. Intense engagement with the many sorts of envisioned users is thus essential; in fact, the design process necessitates active involvement from a variety of stakeholders, including developers, analysts, testers, and managers.

1.3 Thesis Objective

The primary objective of this endeavor is to create user-friendly automated or semi-automated tools that utilize remote sensing to derive Water Quality Parameters. This involves developing a graphical user interface (GUI) for the tool to enhance usability.

1.4 Thesis Outline

The general structure of this thesis can be described as follows:

Chapter 1 introduces the work.

Chapter 2 focuses on the Area of Interest (AOI) and the data. It provides an overview of the AOI and discusses the data used, its source, and collection methods.

Chapter 3 discusses the processing steps and the algorithm employed. It outlines the sequential stages of data processing, including pre-processing and post-processing. The chapter also highlights the specific algorithm utilized for processing the data, emphasizing its role in achieving the research objectives.

Chapter 4 focuses on the software utilized, Python libraries employed, and their respective configurations. The chapter highlights the software platform chosen for the study and its relevance to the research objectives. It discusses the specific Python libraries leveraged for data analysis, emphasizing their functionalities and how they were configured to meet the study's requirements.

Chapter 5 presents the obtained results and Analysis.

Chapter 6 includes the general conclusions derived from the work.

Chapter 2: Area of Interest and Data

2.1 Area of Interest

Before carrying out the analyses, it was important to define the boundaries of the areas of interest (AOIs) for this study. Italy’s Alpine and Subalpine regions are home to a significant concentration of lakes, with over 500 lakes exceeding 0.2 km^2 in surface area and holding a total water volume exceeding 150x109 m3 [48]. These regions are characterized by diverse landscapes shaped by mountainous terrains, deep valleys, and glaciation, resulting in the formation of basins that accommodate these lakes. Understanding the distribution, characteristics, and dynamics of these lakes is vital for environmental management, sustainable development, and conservation. Among them, the Deep Southern Subalpine Lakes (DSL), including Lakes Garda, Iseo, Como, Lugano, and Maggiore, hold approximately 80% of the total water volume and play a crucial role in the study area's water resources. The DSL lakes' unique hydrological dynamics, influenced by their location along the southern border of the Alpine chain, contribute to their significant water accumulation [48]. In contrast, the southern region of the DSL, particularly the plain of the river Po, heavily relies on these water resources for agriculture and industry, supporting Italy's densely populated and productive region. The study area focuses on four lakes within the Deep Southern Subalpine Lakes (DSL) region. These lakes are Lake Varese, Lake Como, Lake Maggiore, and Lake Lugano Furthermore, these lakes have been the subject of extensive scientific research, serving as valuable sites for studying various aspects of limnology and promoting experiments and research initiatives across multiple sectors [48].

2.1.1 Lake Como

The first selected study area is Lake Como, also known as Lago di Como or Lario, is a stunning lake located in Lombardy, northern Italy. Situated approximately 40 kilometers north of Milan, Lake Como lies in a picturesque depression surrounded by limestone and granite mountains. The elevation of the lake is 199 meters, while the surrounding mountains reach heights of about 600 meters in the south and over 2,400 meters in the northeast.

Lake Como has three branches of similar length, each approximately 26 kilometers long. One branch extends northward past Colico, while the other two stretch in different directions. One goes southwestward towards the city of Como, and the other goes southeastward beyond Lecco, which is also known as Lecco Lake. The Bellagio Promontory marks the point where the lake splits into these two branches. With a length of about 47 kilometers and a width of up to 4 kilometers, Lake Como covers an area of 146 square kilometers. It boasts a maximum depth of 414 meters. The lake is fed by the Adda river, which enters near Colico and exits at Lecco. It also receives water from various other rivers and mountain streams, including the Mera. Lake Como is prone to frequent floods and experiences the influence of two winds: the Tivano from the north in the morning and the Breva from the south in the afternoon [51].

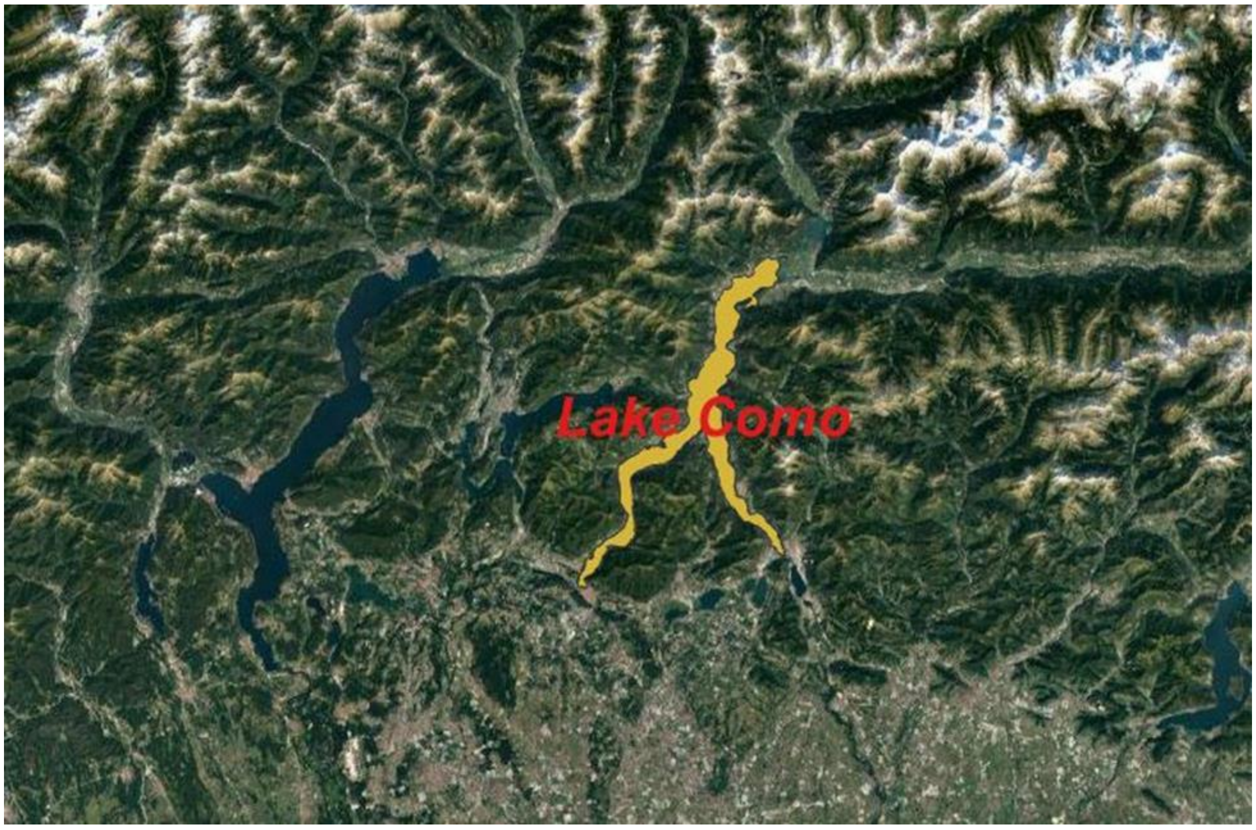


Figure 3. Location of Lake Como.

2.1.2 Lake Lugano

The second selected study area is Lake Lugano, a captivating natural glacial lake located at the southern edge of the Central Alps, spanning the border between Switzerland and Italy. Its geographical coordinates are E 9° 0' 56.35'' and N 46° 0' 23.77'', with an altitude of 271 meters. The lake is distinguished by a causeway, constructed on a natural moraine, which divides it into two primary basins: the north basin and the south basin. The north basin of Lake Lugano is notable for its considerable depth, reaching 288 meters. It possesses a unique characteristic known as Meromixis, where the lake's water remains stratified for extended periods due to a salinity disparity between the deep and surface waters. This phenomenon results in a separation of the water layers, with limited mixing occurring between them. The south basin of Lake Lugano, in contrast, does not exhibit the same level of stratification as the north basin. The waters in this basin experience more frequent mixing and circulation, allowing for a more homogeneous water column [52].

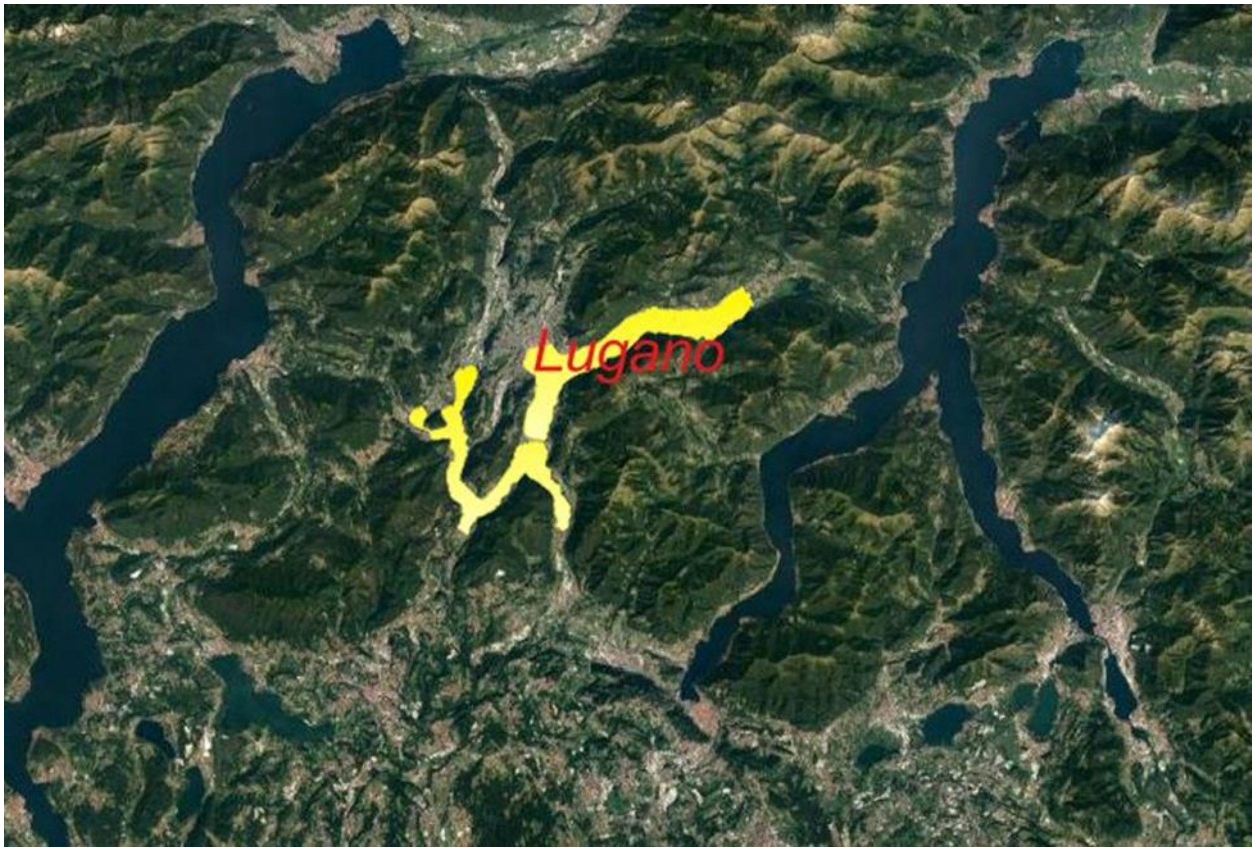


Figure 4. Location of Lake Lugano.

2.1.3 Lake Maggiore

The third selected study area is Lake Maggiore, situated at the foothills of the South-Western Alps, is a prominent lake with a significant portion (approximately 80%) lying in North-Western Italy, between the Piedmont and Lombardy regions. The northern part of the lake extends into Canton Ticino in Southern Switzerland. With a surface area of 213.0 square kilometers, lake Maggiore encompasses a vast expanse of water. It has a volume of 38.1 cubic kilometers and reaches a maximum depth of 370.0 meters at the Ghiffa limnological measurement site. The reference elevation of the lake is η = 193.5 meters above sea level.

Lake Maggiore serves as a crucial drainage point for the upstream portion of the Ticino River watershed, which is the second-largest river in Italy by discharge, following the Po River. Consequently, the lake holds significant importance as the most relevant tributary to the Ticino River. The catchment area of Lake Maggiore spans 6599 square kilometers and encompasses several alpine valleys known for their high rainfall. Additionally, it includes smaller surrounding lakes, with notable examples being Lakes Lugano, Orta, and Varese.

In total, Lake Maggiore receives inflows from 33 different sources. Among these, the inflowing Ticino River holds particular significance as the main tributary to the lake [53].

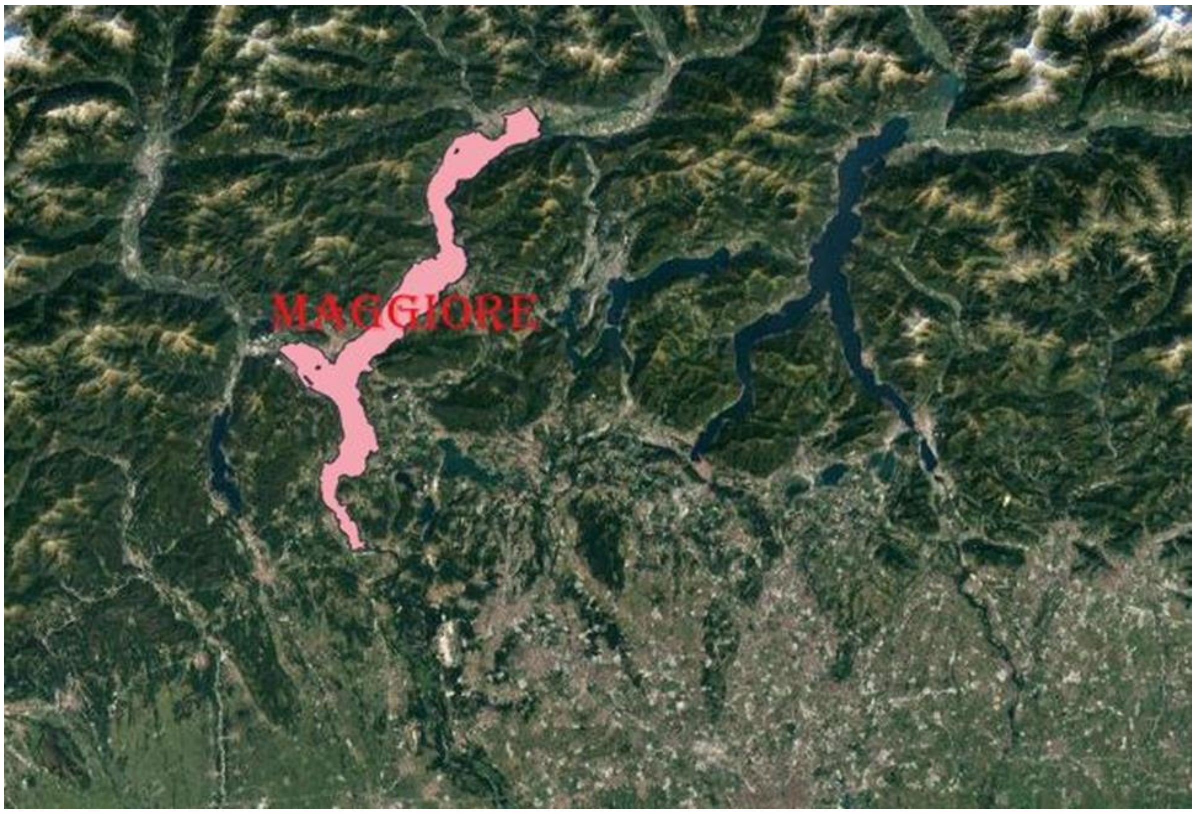


Figure 5. Location of Lake Maggiore.

2.1.4 Lake Varese

The fourth selected study area is Lake Varese, situated in northern Italy at coordinates 45° 49' N and 8° 44' E. It is located at the foothills of the Alps Mountain range, with a mean altitude of 236 meters above sea level. Lake Varese is classified as a monomictic and eutrophic shallow lake, characterized by a mean depth of 11 meters and a maximum depth of 26 meters. The lake covers a surface area of 14.8 square kilometers and has a volume of 153 × 106 cubic meters. Its theoretical renewal time is estimated to be 1.7-1.9 years [49]. The catchment area of Lake Varese spans 115.5 square kilometers and features a relatively high population density, averaging 700 inhabitants per square kilometer. The region surrounding the lake is associated with various industrial and commercial activities. Lake Varese receives water inputs from two tributaries: the Brabbia channel and the Tinella stream, with annual average discharges of 23 × 106 and 10 × 106 cubic meters per year, respectively. Additionally, the lake has one effluent, the Bardello Stream, with an annual average discharge of 80.4 × 106 cubic meters per year [49].

These hydrological characteristics, including the lake's depth, surface area, volume, and renewal time, provide important insights into its dynamics and water balance. The surrounding catchment area, with its population density and industrial activities, contributes to the potential inputs and impacts on the lake's water quality and ecosystem. Understanding the hydrological processes and water exchanges within Lake Varese and its catchment area is essential for assessing its environmental health and developing appropriate management strategies [49].

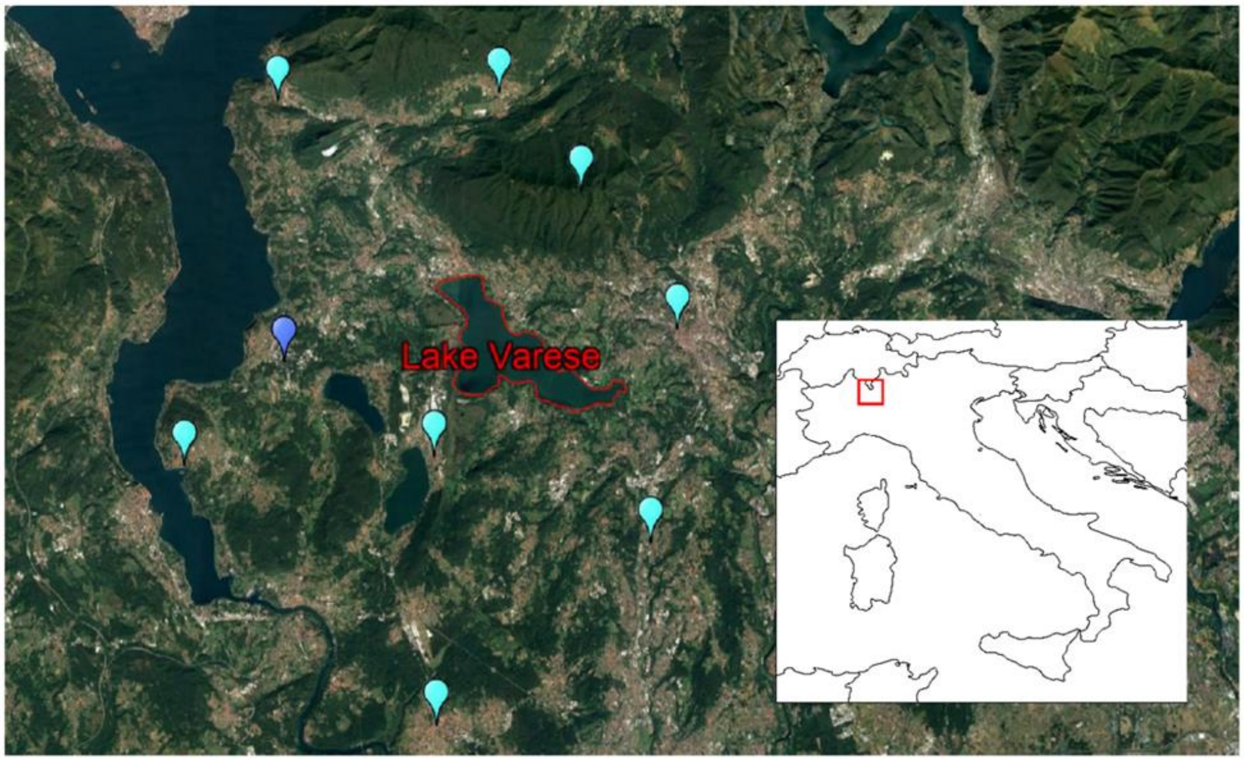


Figure 6. Location of Lake Varese [79].

The positions of the available meteorological stations in the region are indicated by the light blue dots (Regional Environmental Protection Agency Lombardia, ARPA), and the dark blue dot (the Joint Research Center). The red polygon in the lower right corner map indicates the location of the model grid-cell from the ERA-Interim atmospheric reanalysis, where Lake Varese is located [50].

Parameters	Como	Lugano	Maggiore	Varese
Altitude (m)	198	271	193	238
Area (km^2)	146	28	213	14.8
Maximum depth (m)	410	288	370	26
Mean depth (m)	154	171	178	10.7
Volume(km^3)	22.5	4.69	37.5	0.1
Renewal time (years)	4.5	12.4	4.1	1.9

Table 1. Morphometric and hydrological characteristics of the four Lakes in Northern Italy.

2.2 Data

The selection of data for this research was driven by the specific requirements of the analysis process. A diverse range of data types, including sensors, satellite images, and computed datasets, were carefully chosen from multiple sources. The following section provides a comprehensive description of the data used in this research, covering its sources, characteristics, and methodologies. This overview establishes a solid foundation for the subsequent analyses and ensures credibility in the research findings.

2.2.1 Satellite Observation

Satellite images play a crucial role in monitoring the quality of lakes, offering an enhanced approach compared to traditional methods. By providing high-frequency data covering large lake areas, satellite images offer valuable insights. In this research, free satellite images were obtained from spatial agencies' web portals and processed using open-source algorithms. Specifically, the images from Sentinel-3 A/B OLCI were utilized to extract suspended matter (TSM) and chlorophyll-a (CHL-a), while Landsat 8 TIRS images were used to extract lake surface water temperature (LSWT). This utilization of satellite images enables a comprehensive analysis of the Water Quality Parameters, contributing to a more accurate understanding of the lake ecosystem.

Water Quality Parameters	Spatial resolution	Source
Chlorophyll	300m	ESA - Sentinel-3 A/B OLCI
Total Suspended Matter	300m	ESA - Sentinel-3 A/B OLCI
Lake Surface Water Temperature	30m[1]	NASA - Landsat 8 TIRS

Table 2. Overview of the source data for WQP maps [74]

2.2.1.1 Sentinel-3 Images

The Sentinel-3 mission is a collaborative effort between ESA and EUMETSAT, aimed at Earth Observation. It plays a crucial role in various applications by measuring sea surface topography, sea and land surface temperature, and ocean and land surface color with high accuracy. The mission ensures consistent quality, high availability (>95%), and reliable data, carrying on the ocean measurement capabilities of ENVISAT. The mission

[1] The resolution of original Landsat8 TIRS bands are not unique, but the LSWT products are resampled to 30m.

comprises four main instruments: OLCI, SLSTR, SRAL, and MWR [54]. The S3 (Sentinel-3) satellite carries a suite of seven sensors designed to measure various properties of the ocean. These sensors include instruments for measuring Ocean Color, Sea Surface Temperature (SST), radar and microwaves, Doppler positioning, a laser reflector, and ESA's Global Navigation Satellite System (GNSS). The mission's objective is to provide a range of thematic areas and services, including Numerical Ocean Prediction, Maritime Safety and Security, Open Ocean and Ice Monitoring, Atmospheric services, Global Land Monitoring Applications, and more. Through these capabilities, Sentinel-3 aims to contribute to a better understanding and monitoring of our oceans and their associated ecosystems, as well as supporting a wide range of applications and services related to ocean and environmental management [57]. The OLCI (Ocean and Land Color Instrument) products are available on three main levels. Level-1 (L1) products contain radiance measurements for each pixel and calibrated, orthorectified (geolocated), and spatially resampled Top-of-Atmosphere (TOA) radiances. Level-2 Water (L2W) products, such as OL_2_WRR and OL_2_WFR, are derived from the OLCI Level-2 processor and provide geophysical and bio-optical constituents at reduced (~1 km/pixel) and full (300 m/pixel) resolutions. L2W products are particularly relevant for monitoring coastal areas, offering actual concentrations or inherent optical properties (IOP's) of water constituents. Additionally, third-party processors can extract remote sensing reflectance and water quality variables from OLCI Level-1 data, expanding the range of analysis and derived products available for further assessment.

The OLCI instrument on the Sentinel-3 satellite has a set of bands specifically designed for measuring Ocean Color over the open sea and coastal zones. These bands were inherited from the MERIS (Medium Resolution Imaging Spectrometer) instrument and have been complemented to optimize the measurement capabilities. To improve atmospheric correction, an additional band at 1.02 μm was added. This band helps in better characterizing the atmospheric effects on the observed signals, allowing for more accurate retrieval of ocean color information. Furthermore, a channel at 673 nm with a width of 7.5 nm was included to enhance measurements of Chlorophyll fluorescence. Chlorophyll fluorescence is a phenomenon exhibited by marine plants and algae, and its measurement provides valuable information about the presence and productivity of phytoplankton in the water. The OLCI instrument, with its wide swath width of 1270 km, enables the observation of a broad area of the Earth's surface in a single pass. This wide coverage is advantageous for efficient data collection over large regions, facilitating global monitoring and analysis. To mitigate the effects of sun glint, which is the specular reflection of sunlight on water surfaces, the OLCI instrument is tilted across the track by 12.6° in the opposite direction to the sun. This tilt helps minimize the interference caused by sun glint, allowing for more accurate and reliable measurements of ocean color [57]. The OLCI bands (Table 3), inherited from the MERIS instrument, have been complemented to optimize the measurement of ocean color over both open sea and coastal zones. An additional band at 1.02 μm was added to improve atmospheric correction, and a channel at 673 nm (7.5 nm wide) was included for enhanced Chlorophyll fluorescence measurements. Therefore, the OLCI instrument combines wide coverage, sun glint reduction, and optimized spectral bands to provide valuable data for studying ocean color and conducting atmospheric correction. [57].

The sensors and image types chosen for the Sentinel-3 mission were driven by project requirements. The OLCI instrument, with its daily acquisition frequency and suitable wavelength bands and radiometric content, is well-suited for studying aquatic environments. Its images have been successfully utilized in various studies and applications. In the case of SIMILE, the parameters monitored using Sentinel-3 are chlorophyll-a (CHL-a) and total suspended solids (TSM). OLCI's high acquisition frequency, with daily images captured around 10:00 in the specified latitudes, ensures a substantial dataset of information over the course of a year. However, some images may be excluded due to high cloud cover or other disturbances. Despite these exclusions, the availability of data allows for the determination of the temporal trend in parameter concentrations in lakes. Additionally, OLCI's medium-high spatial resolution of 300m is adequate for monitoring objectives and facilitates the identification of anomalies in parameter concentrations within the typical range of algal blooms.

Band	λ centre (nm)	Width (nm)	Function
Oa01	400	15	Aerosol correction, improved water constituent retrieval
Oa02	412.5	10	Yellow substance and detrital pigments (turbidity)
Oa03	442.5	10	Chlorophyll absorption maximum, biogeochemistry, vegetation
Oa04	490	10	High Chlorophyll,
Oa05	510	10	Chlorophyll, sediment, turbidity, red tide
Oa06	560	10	Chlorophyll reference (Chlorophyll minimum)
Oa07	620	10	Sediment loading
Oa08	665	10	Chlorophyll (2nd Chlorophyll absorption maximum), sediment, yellow substance/vegetation
Oa09	673.75	7.5	For improved fluorescence retrieval and to better account for smile together with the bands 665 and 680 nm

Oa10	681.25	7.5	Chlorophyll fluorescence peak, red edge
Oa11	708.75	10	Chlorophyll fluorescence baseline, red edge transition
Oa12	753.75	7.5	O2 absorption/clouds, vegetation
Oa13	761.25	2.5	O2 absorption band/aerosol correction.
Oa14	764.375	3.75	Atmospheric correction
Oa15	767.5	2.5	O2A used for cloud top pressure, fluorescence over land
Oa16	778.75	15	Atmos. corr./aerosol corr.
Oa17	865	20	Atmospheric correction/aerosol correction, clouds, pixel co-registration
Oa18	885	10	Water vapour absorption reference band. Common reference band with SLSTR instrument. Vegetation monitoring

Oa19	900	10	Water vapour absorption/vegetation monitoring (maximum reflectance)
Oa20	940	20	Water vapour absorption, Atmospheric correction/aerosol correction
Oa21	1020	40	Atmospheric correction/aerosol correction

Table 3. OLCI Band characteristics [57]

2.2.1.2 Landsat 8 Images

The LDCM satellite Launched on February 11, 2013, is part of the long-standing Landsat program initiated by the United States. It follows the legacy of previous Landsat satellites in acquiring global space imagery. LDCM is equipped with two main instruments: the Operational Land Imager (OLI) and the Thermal Infrared Sensor (TIRS). The OLI operates at nine different wavelengths, allowing it to capture images with a maximum resolution of 15 meters. This high resolution enables observations of various features such as cirrus clouds and coastal waters. On the other hand, the TIRS focuses on capturing thermal infrared images to study surface temperature characteristics and heat/moisture transfer. It operates at a spatial resolution of 100 meters, providing valuable data for monitoring temperature trends and patterns. The TIRS sensor can also resample its data to a 30-meter resolution, allowing for better visualization of temperature differences, even at a smaller scale [56].

One of the significant advantages of the LDCM satellite is its improved design, which enhances reliability and extends its service life by at least 5 years compared to previous Landsat satellites. Additionally, both the OLI and TIRS instruments operate in a scanning mode, which reduces radiometric distortion in the acquired images. The availability of the TIRS thermal sensor on Landsat 8 has facilitated the monitoring of lake temperatures. By analyzing the most superficial layer of water, the sensor can define temperature trends in lakes. With an average acquisition frequency of approximately every 16 days, the TIRS sensor captures data that enables the observation of temperature changes over time. Moreover, the spatial resolution of the TIRS at 100 meters, which can be resampled to 30 meters, has proven useful in highlighting temperature differences. For example, it can detect temperature variations of about 1°C in the inlet areas of major rivers, providing valuable insights into thermal characteristics and processes associated with water bodies [55]. In the study, Landsat 8 images are utilized to generate surface water temperature (LSWT) maps.

	Bands	Wavelength (micrometers)	Resolution (meters)
Landsat 8 Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) Launched February 11, 2013	Band 1 - Coastal aerosol	0.43 – 0.45	30
	Band 2 - Blue	0.45 – 0.51	30
	Band 3 - Green	0.53 – 0.59	30
	Band 4 - Red	0.64 – 0.67	30
	Band 5 - Near Infrared (NIR)	0.85 – 0.88	30

	Band 6 - SWIR1	1.57 – 1.65	30
	Band 7 - SWIR 2	2.11 – 2.29	30
	Band 8 - panchromatic	0.50 – 0.68	15
	Band 9 - Cirrus	1.36 – 1.38	30
	Band 10 – Thermal Infrared (TIRS) 1	10.60 – 11.19	100
	Band 11 – Thermal Infrared (TIRS) 2	11.50 – 12.51	100

Table 4. Landsat 8 Band characteristics [56]

2.2.2 Additional Dataset

To generate output maps using the C2RCC (Case-2 Regional CoastColour (3.2.1)) processor for chlorophyll (CHL) and total suspended matter (TSM), as well as the Land Surface Water Temperature (LSWT) maps, several parameters are required. These parameters are retrieved from datasets with time series precisely matching the date of acquisition of the Sentinel-3 imagery. For the C2RCC processor, the essential parameters include the CHL factor, CHL exponent, Salinity, and temperature, among others. These parameters are obtained from time series datasets that align with the date of acquisition of the Sentinel-3 imagery. Notably, the temperature parameter is periodically retrieved from ARPA Lombardia, providing the necessary information for accurate processing.

On the other hand, to produce LSWT maps, the procedure retrieves atmospheric correction parameters from Landsat-8 imagery captured on the acquisition date. These atmospheric correction parameters consist of the coefficient of atmospheric transmission (τ) and the ascending/descending solar radiation (Lu/Ld). These parameters are organized as time series data in the working folder of the procedure and obtained from the web page for each Landsat 8 acquisition date (Atmospheric Correction Parameter Calculator) and utilized for atmospheric correction during the mapping process.

By incorporating these parameter datasets into the respective processing procedures, the C2RCC processor can generate CHL and TSM maps. In contrast, the LSWT maps rely on the atmospheric correction parameters obtained from Landsat-8 imagery. These maps provide valuable information about the water ecosystem and land surface temperature, contributing to a better understanding and monitoring of coastal and aquatic environments.

2.2.2.1 Salinity

Salinity is the measure of dissolved salts in water, serving as a key parameter to classify bodies of water. Freshwater systems, such as lakes and rivers, have low salt concentrations, usually below 1,000 ppm (1 psu). In contrast, the oceans are highly saline, averaging around 35,000 ppm (35 psu). With approximately 97 percent of Earth's water being considered saline, including inland seas and underground reservoirs, the oceans hold the largest reservoir of saline water. Salinity levels influence physical properties like density and freezing point, impacting ocean circulation and climate patterns. Additionally, salinity plays a crucial role in the distribution of marine organisms, as many species have specific salinity requirements. Monitoring and understanding salinity provides insights into aquatic ecosystem health, water movement, and interactions between freshwater and saltwater environments.

2.2.2.2 CHL factor and CHL exponent

The Chlorophyll exponent (CHL exponent) is a parameter used to describe the spectral shape of the chlorophyll absorption curve in water. It provides information about the distribution and concentration of chlorophyll pigments in aquatic environments.

2.2.2.3 Temperature

The project utilized the average temperature of the scene by referencing the value of the air temperature at the time of satellite image acquisition. To obtain this temperature data, the project relied on the measurements provided by ARPA Lombardia (Regional Environment Protection Agency, https://www.arpalombardia.it). The temperature information was periodically retrieved from a specific station called Tremezzo, which has the ID sensor 14606. Figure 7 illustrates the temperature profile for the past five years.

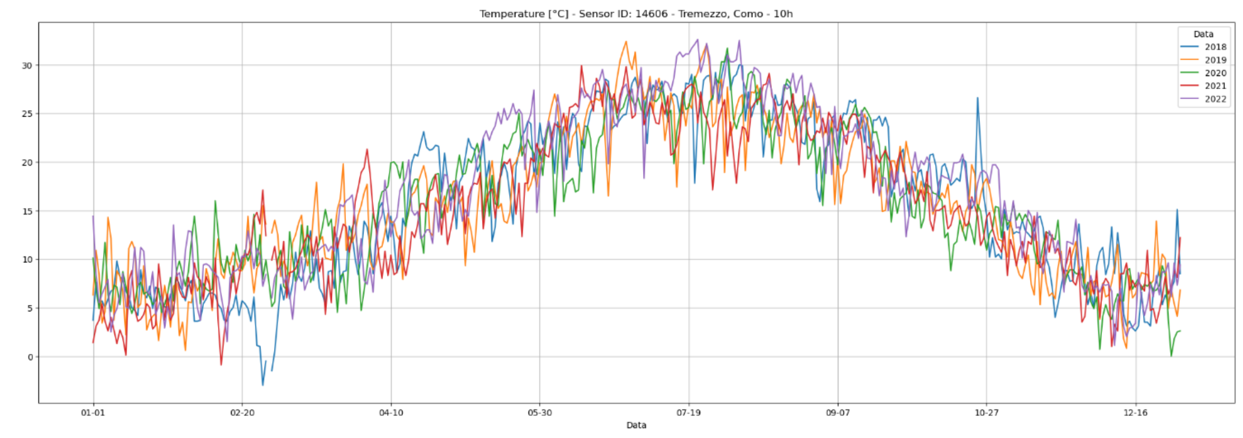


Figure 7. Temperature Profile

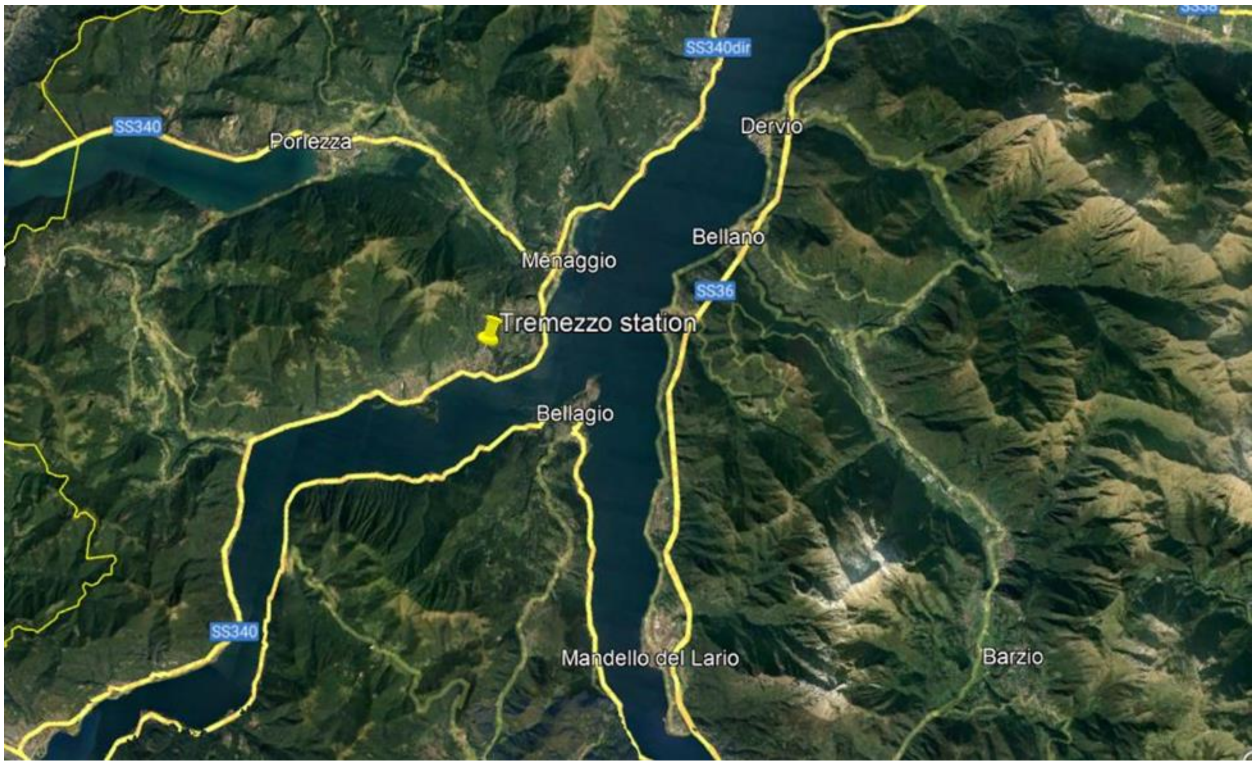
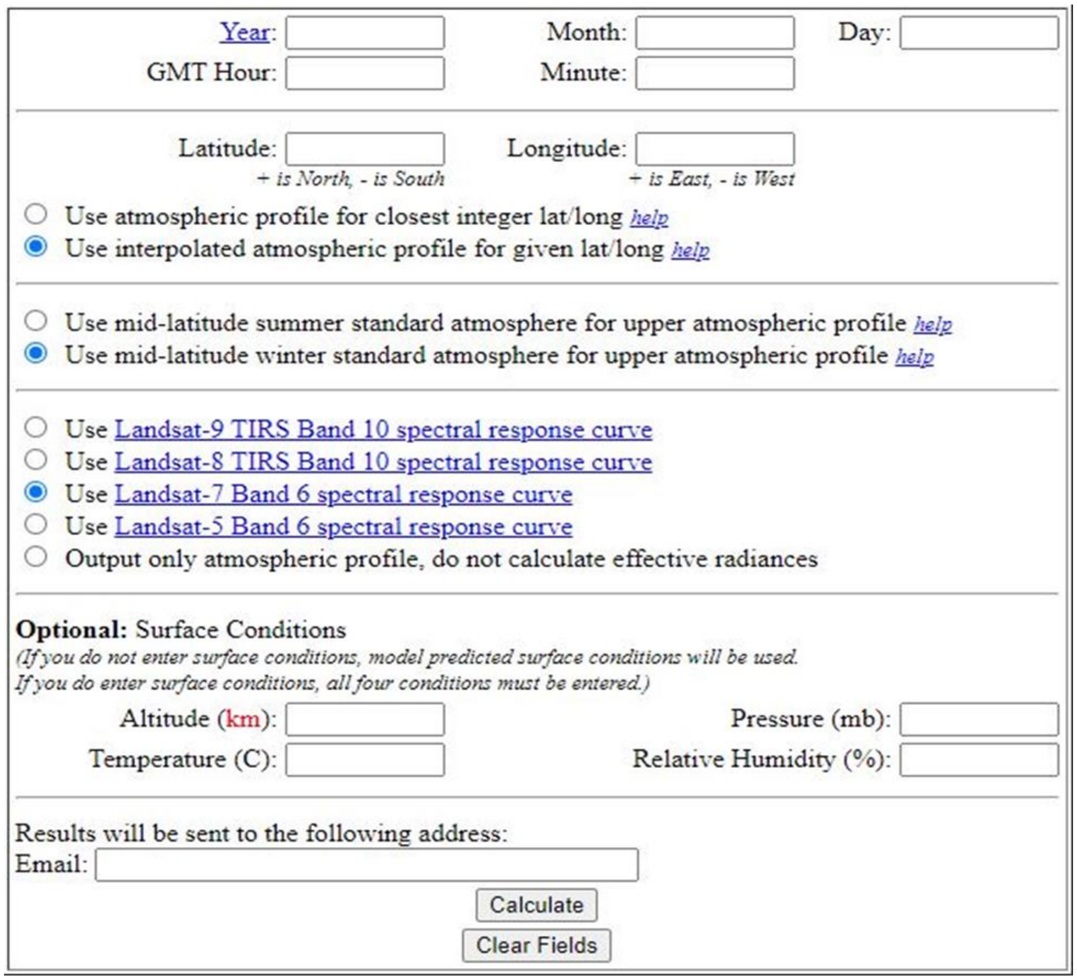


Figure 8. Location of Tremezzo station

2.2.2.4 Atmospheric correction parameters.

Ascending (Lu) and descending (Lu) solar radiances refer to the measurements of solar radiation at different angles of incidence with respect to a satellite's orbit. These measurements are essential for various applications, including atmospheric correction and radiative transfer calculations. Atmospheric transmission (τ), on the other hand, represents the fraction of sunlight that reaches the Earth's surface after travelling through the atmosphere, without being absorbed or scattered. τ can be calculated using tools like the Atmospheric Correction Parameter Calculator provided by NASA (https://atmcorr.gsfc.nasa.gov/).



Year: Month: Day:
GMT Hour: Minute:

Latitude: Longitude:
+ is North, - is South   + is East, - is West

○ Use atmospheric profile for closest integer lat/long help
[✓ Use interpolated atmospheric profile for given lat/long help]

○ Use mid-latitude summer standard atmosphere for upper atmospheric profile help
[✓ Use mid-latitude winter standard atmosphere for upper atmospheric profile help]

○ Use Landsat-9 TIRS Band 10 spectral response curve
○ Use Landsat-8 TIRS Band 10 spectral response curve
[✓ Use Landsat-7 Band 6 spectral response curve]
○ Use Landsat-5 Band 6 spectral response curve
○ Output only atmospheric profile, do not calculate effective radiances

Optional: Surface Conditions
(If you do not enter surface conditions, model predicted surface conditions will be used. If you do enter surface conditions, all four conditions must be entered.)
Altitude (km): Pressure (mb):
Temperature (C): Relative Humidity (%):

Results will be sent to the following address:
Email:

Figure 9. Atmospheric Correction Parameters Calculator [72]

Chapter 3: Procedure Design

Unlike in-situ data, remote sensing technologies can simultaneously monitor large areas with meaningful temporal coverage, capturing the geographical and temporal variability of optically active Water Quality Parameters (WQPs) [36]. Due to the general ongoing improvement of satellite sensor design and resolutions, inland water quality studies and monitoring applications based on optical satellites have increased dramatically in recent years [35] [37]. To address the issues raised by in-situ data (refer to Section 1.2.1), remote sensing techniques are being investigated as a supplement to in-situ observations. [35]. In recent years, the availability of open data, computational resources, and processing platforms has encouraged the development and testing of algorithms to determine the constituents of water bodies and their spatiotemporal modeling for addressing patterns [35]. Now, remote sensing methods represent an opportunity for retrieving synoptic views of water bodies to monitor the spatial and temporal variability of optically active WQPs [36].

Water Quality Parameter maps based on weekly Chl-a and TSM concentration maps were created using the Sentinel-3 A/B Ocean and Land Color Instrument (OLCI), while monthly LSWT images were created using the Landsat-8 satellite Thermal Infrared Sensor (TIRS). Even though the satellite observations came from different sensors (refer to Chapter 2), they were all processed with different procedures using ESA's open-source software SNAP (Sentinel Application Platform) [46], which has been validated for use in inland aquatic environments [47].

Chl-a concentrations, which are indicative of the trophic status in aquatic environments, are commonly used to assess phytoplankton biomass dynamics, including harmful algal blooms, and their impacts on aquatic food webs, biogeochemical cycles, and aquaculture. TSM represents the organic and mineral-suspended solids in water and is highly correlated with water turbidity and transparency (traditionally measured using a Secchi disk). Monitoring TSM spatially and temporally can provide valuable information for sediment transport, water quality evaluation, and lake management research. CDOM found both on land and in water, is a mixture of organic molecules derived from vegetation, algae, and bacteria. CDOM affects light penetration in water and can serve as a proxy for carbon concentration in lakes, thus supporting carbon cycle research and water treatment efforts [45].

The processing and analysis of satellite data will allow a synoptic view of the quality status of the surface water in the Insubric lakes. The maps generated during the project following the processing steps (Figure 10) will allow the evaluation of the temporal and spatial variability of the surface water quality status. The results produced will be important inputs for ecological models and will allow us to identify the presence of any anomalies due to algal blooms and/or river plumes.

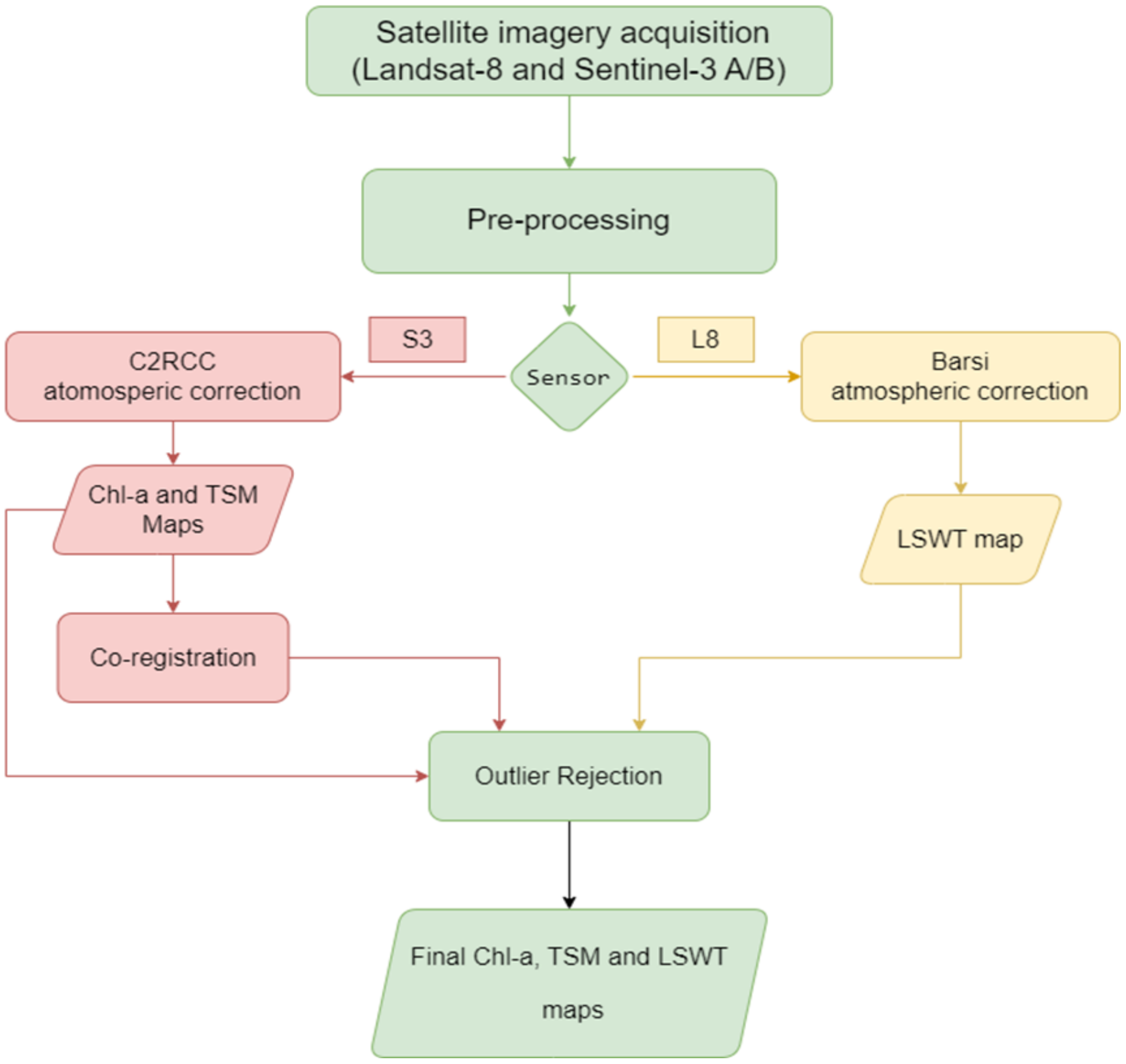

Satellite imagery acquisition (Landsat-8 and Sentinel-3 A/B)
Pre-processing
S3
Sensor
L8
C2RCC atomosperic correction
Barsi atmospheric correction
Chl-a and TSM Maps
LSWT map
Co-registration
Outlier Rejection
Final Chl-a, TSM and LSWT maps

Figure 10. Flowchart for WQPs Mapping

The production of maps comes with limitations and challenges that must be considered. The temporal resolution and number of WQP maps produced are dependent on the weather conditions at the time of image acquisition. Because SIMILE uses optical satellite images, it must deal with cloud covers that might obscure the sensors' view for days. When cloud cover did not obscure the entire lake region, satellite images were processed. As a result, the dataset's temporal resolution is lower than the satellite sensors' revisit time [38]. Sun glint is a significant problem when observing watercolor from space, resulting in anomalously bright pixels that should be treated [39]. In addition, standard satellite data processing techniques, which are mainly designed for oceans, presume an unbounded water surface and, as a result, ignore the presence of neighboring land. As an outcome, radiance reflected by the land and subsequently scattered by the atmosphere in the field of view of a satellite sensor observing the target water is a source of perturbations, resulting in water reflectance-based product errors. This phenomenon is known as the adjacency effect, and it always happens when a scattering medium is placed in the proximity of a surface [40]. The optical characteristics of the lake waters under study, categorized as case 2 (to distinguish them from case 1 waters corresponding to the oceans), are heavily influenced by inorganic and/or organic sediments, necessitating high accuracy in atmospheric correction methods to properly retrieve water parameters. The problem of atmospheric correction in case 2 has yet to be overcome. As a result, much effort has gone into developing atmospheric correction processors that span a wide range of methodologies [41] [42]. However, the performance of the processors varies depending on the scenario (sun and observation geometry, atmospheric, optical, and site-specific conditions), and there is currently no standardized approach; however, atmospheric correction processors continue to evolve as new methods and data become available. This brings the need to continue to test various atmospheric correction procedures as well as water quality retrieval methods using in situ data that accounts for a wide range of water types and environmental circumstances [43] [44].

3.1 Pre-processing

Data preprocessing is a crucial and critical step in any spatial data analysis workflow that can have a significant impact on the accuracy and reliability of the results obtained.

The pre-processing techniques employed for the Sentinel-3 OLCI Level 1 data were composed of:

- Image cropping to the region of interest which cuts down on processing time;
- Re-projection to the reference map projection of Italy (WGS84/ UTM Zone 32).

On the contrary, the pre-processing techniques employed for the Landsat-8 data are composed by:

- Image cropping to the region of interest, which cuts down on processing time;
- Resampling (nearest neighbour method): a technique in which the value of each cell in an output raster is calculated using the value of the nearest cell in an input raster. The nearest neighbour assignment does not change any of the values of cells in the input layer. Resampling is necessary since the Landsat 8 bands' resolutions vary (refer to 2.2.1.2).

3.2 Processing Algorithms

3.2.1 Algorithmic Approach for Sentinel-3 OLCI (Chlorophyll-a and Total Suspended Matter Mapping)

While in the open ocean, the light field emerging from the water surface and measured by a satellite is determined by the concentration of the phytoplankton pigment chlorophyll-a, in coastal waters, estuaries, rivers, and inland waters, other substances contribute to the optical signature of the water body, i.e., the reflected light spectrum. Conventionally, the manifold of constituents is reduced to three components [60]: phytoplankton pigments, inorganic suspended sediments, and yellow substances.

Retrieval of concentrations of these constituents is done by the Case 2 Regional Processor (C2RCC), which is made available in the SNAP Toolbox.

The C2RCC processor was originally developed by Doerffer and Schiller [58][59]. According to Brockmann [62], it consists of a bio-optical model that incorporates the optical characteristics (absorption and scattering coefficients) of water constituents as well as the concentration ranges and co-variations among the water constituents, which resulted in the creation of a database that consists of approximately 5 million cases, encompassing a wide range of scenarios and data points. As a fundamental technique, the created database was used to train neural networks (NN), which perform the inversion of the water leaving reflectance ("water signal") spectrum as well as top-of-atmosphere radiances ("satellite signal"), i.e., the determination of the water leaving radiance from the top-of-atmosphere radiances, as well as the retrieval of the inherent optical properties (IOPs) of the water body. For clarity, the top-of-atmosphere full spectrum (all sensor bands) is input to a neural net, and the water-leaving reflectance in the VIS and NIR bands is the output (the radiances are converted into reflectance's). The training can be understood as nonlinear multiple regression, as described in [58].

The bio-optical model used to parameterize the In-water modeling uses 5 components for scattering and absorption: The total absorption is composed of pigment absorption (apig) and two components for detritus (adet) and gel stuff (agelb). These two components differ by their spectral slopes. The total scattering is composed of two components, namely a white scatterer (bwhit) and a typical sediment scatterer (btsm). The white scatterer represents calcareous sediments. These are larger particles that are reflected in a corresponding phase function. All IOPs are defined at 443nm wavelengths. These 5 components are an abstraction of the large variety of IOPs in natural waters. IOPs are a proxy for the concentration of the three optically active constituents mentioned before, and Figure 11 shows the relationship between these, the conversion from IOPs to concentration is done using scaling factors. These can be regionally variable, and thus the main output of the C2RCC processor is the 5 IOPs [62].

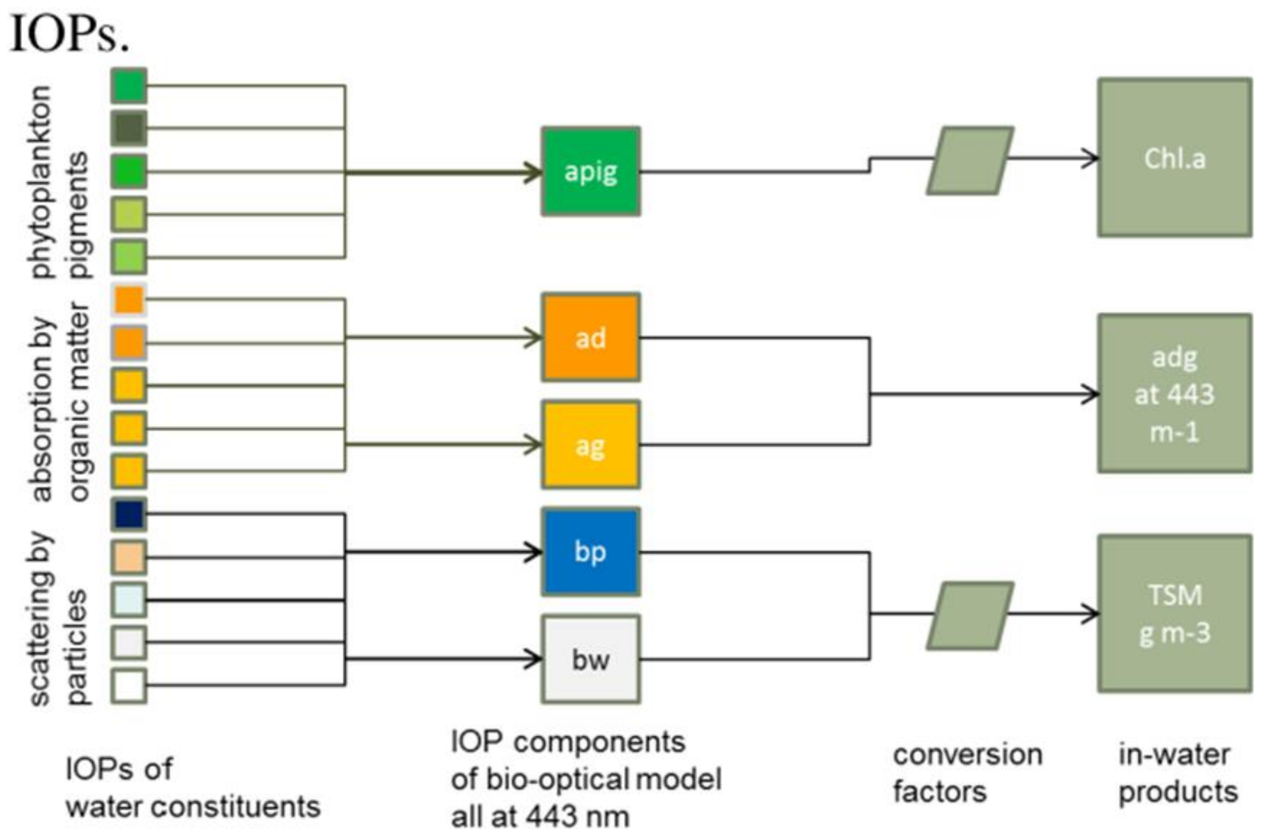


Figure 11. C2RCC Bio-optical model [62]

In the IOPs found in the NOMAD database [61], not only the correlation between the scattering and absorbing components was modeled using a probability approach, thus, to allow for a partial correlation as found in the NOMAD database, but also the value ranges of the concentrations were also determined. The neural nets trained with these ranges are called the "normal nets". A special set of neural nets for extreme ranges of absorption and scattering have also been trained, which is found by default in the SNAP version of the processors. The ranges are summarized in Table 5.

IOP	Normal min	Normal max	Extreme min	Extreme max
a_pig	~ 0	5.3	~ 0	51
a_det	~ 0	5.9	~ 0	60
a_gelb	~ 0	1.0	~ 0	60
a_part	~ 0	60	~ 0	590
a_wit	~ 0	60	~ 0	590

Table 5. Training ranges of the neural nets [62]

The C2RCC processor breaks down into two major parts: the atmospheric correction part, and the in-water part. Figure 12 presents the architecture of the C2RCC processor [62].

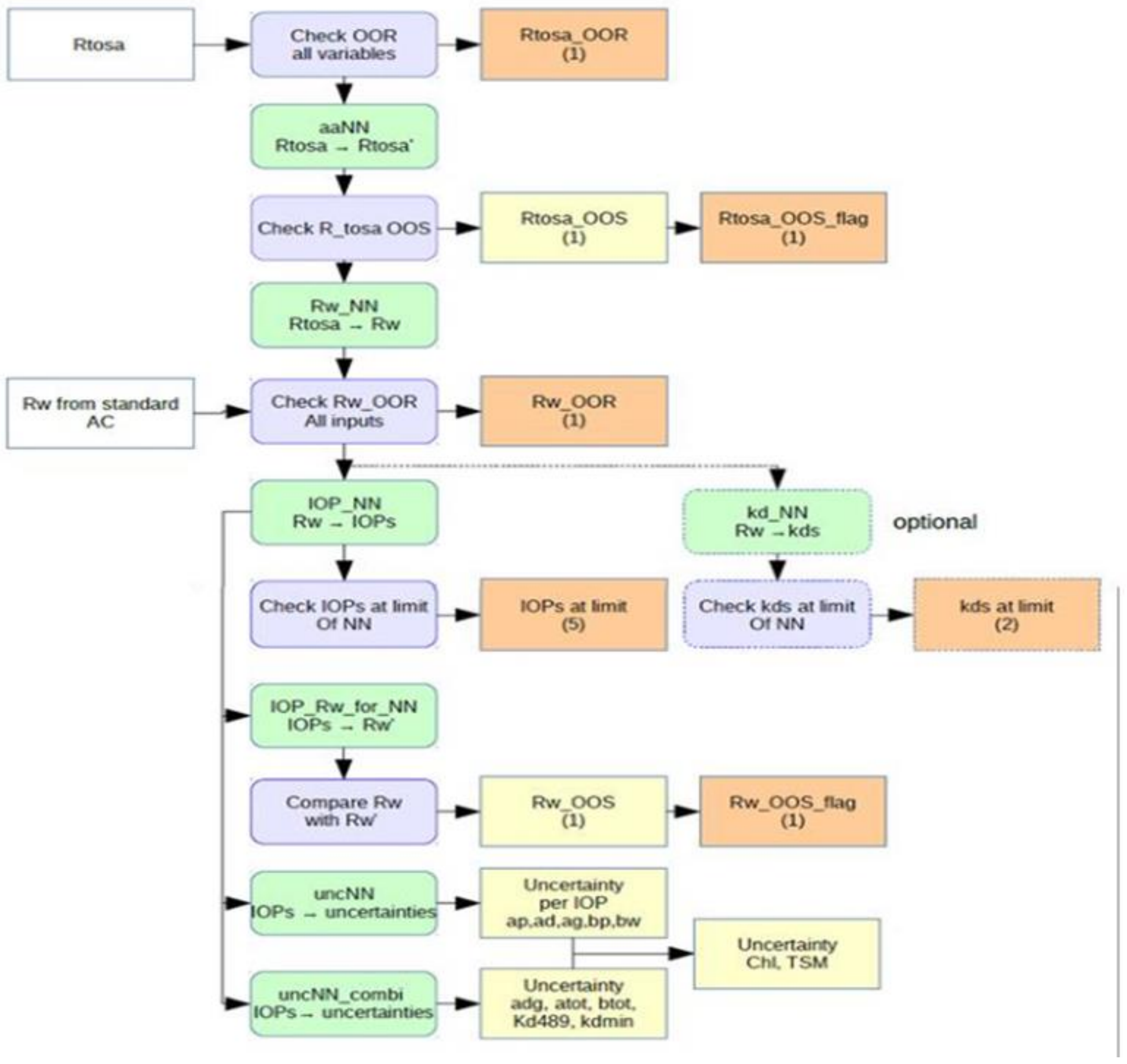


Figure 12. C2RCC processor architecture. Neural nets are shown as green boxes, and quality tests as blue boxes [62]

The main input to the atmospheric part is the top-of-atmosphere radiances/reflectances of the sensor, the input spectra are corrected for gaseous absorption (white box Rtosa in Figure 12). Air pressure, and thus a proper altitude correction, is an inherent part of neural network processing. The main output of the atmosphere part is directional water leaving reflectance's produced by the atmospheric correction neural net (green box Rw_NN). The atmosphere part contains out-of-range tests (blue box check OOR) and out-of-scope tests (green box “aaNN” and blue box “check R_tosa”) of the TOA reflectances, resulting in corresponding quality flags.

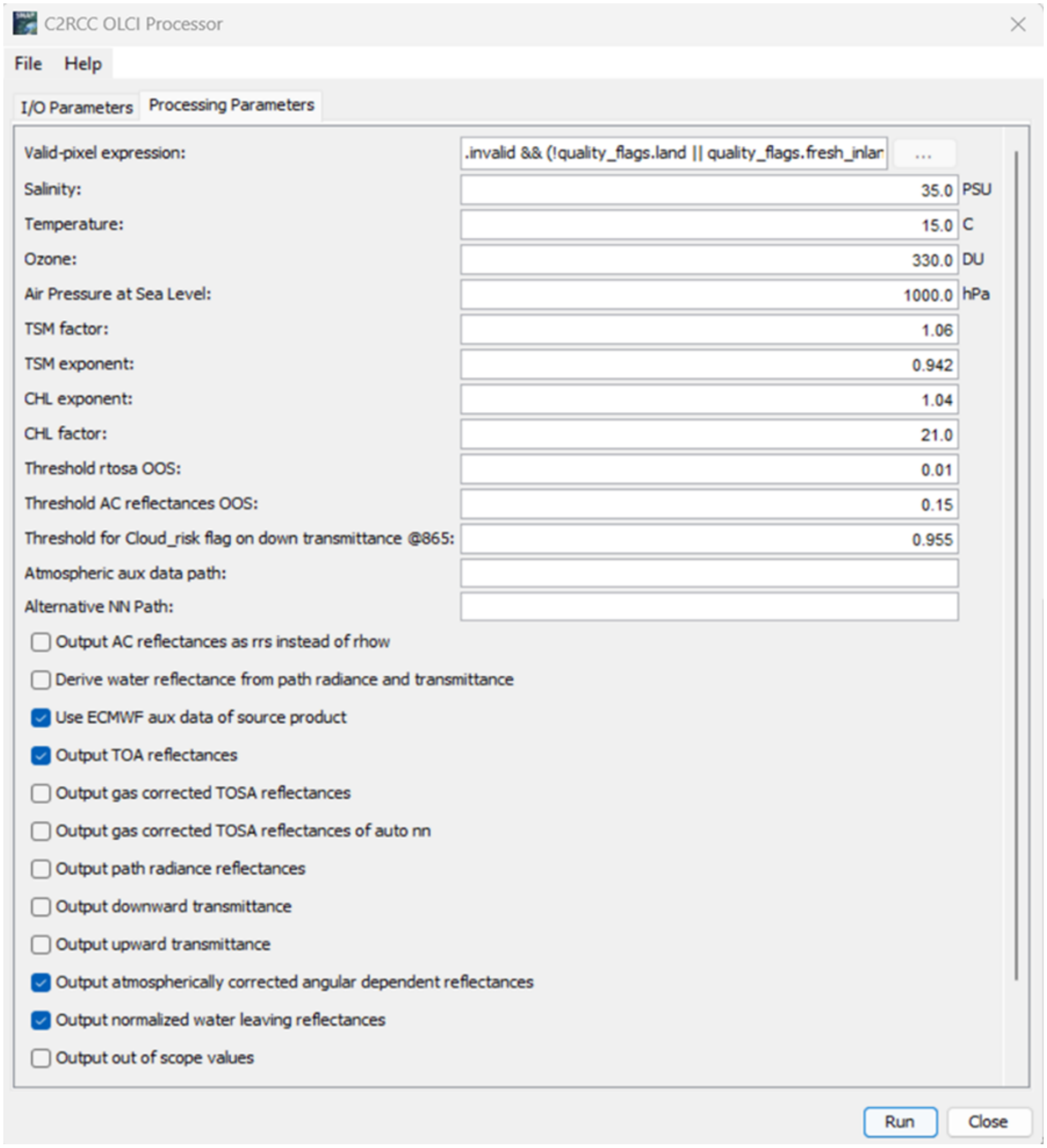
The in-water part gets as input the directional water leaving reflectances from the atmosphere part. Alternatively, it will be also possible to use water leaving reflectance from an external input so that the in-water part can work with alternative atmospheric correction. However, this is currently not implemented in the SNAP processor.

The C2RCC processor has proven to be suitable for inland WQPs mapping while defining the proper modeling parameters [63][64][65], C2RCC processor for atmospheric correction has been updated: the current C2RCC is a modified version of the original Case 2 Regional Processor, which has been adapted to many multispectral satellites (e.g., Sentinel-2, Sentinel-3, Landsat-8). Now the C2RCC is composed of a set of three processors (i.e., C2-Nets: C2RCC, C2X, and C2X-COMPLEX) [55].

In essence, the C2RCC is a processor for the atmospheric correction of satellite images which uses neural networks to transform the radiances measured by the sensor outside the atmosphere into values of ascending reflectance’s and from these derive the inherent optical properties (IOP) of the water: coefficient of phytoplankton absorption at 443 nm (iop_apig) and backscattering coefficient of total suspended solids (iop_btot). The radiances are converted into reflectance’s by the neural network dedicated to atmospheric correction, using the solar irradiance value, some meteorological and atmospheric parameters provided by the ECWMF network, and the mutual position of the Sun and the sensor at the time of acquisition. Then another neural network derives the IOPs from the reflectance’s. Both neural networks work with a dataset of reflectance spectra and corresponding radiances. They are generated with codes that simulate the radiative transfer in water and atmosphere, related to different values of the inherent optical properties of the waters, in a range of directions of radiation exit from the water surface and for different positions of the Sun in the sky, and weather conditions.

Finally, the concentrations of the Water Quality Parameters are obtained from the IOPs: chlorophyll-a [mg/m^3] and total suspended solids [mg/m^3], through coefficients obtained from a large dataset of measurements carried out in different aquatic environments which define the bio-optical model for water, i.e., the link between optical properties and concentrations of Water Quality Parameters.

The default parameterization of the C2RCC bio-optical model, which appears in the plugin window (Figure 13) upon first launch, is changed and adjusted to the SIMILE ecosystem.



Valid-pixel expression:	.invalid && (!quality_flags.land || quality_flags.fresh_inlar	
Salinity:	35.0	PSU
Temperature:	15.0	C
Ozone:	330.0	DU
Air Pressure at Sea Level:	1000.0	hPa
TSM factor:	1.06	
TSM exponent:	0.942	
CHL exponent:	1.04	
CHL factor:	21.0	
Threshold rtosa OOS:	0.01	
Threshold AC reflectances OOS:	0.15	
Threshold for Cloud_risk flag on down transmittance @865:	0.955	
Atmospheric aux data path:		
Alternative NN Path:		

- ☐ Output AC reflectances as rrs instead of rhow
- ☐ Derive water reflectance from path radiance and transmittance
- ✓ Use ECMWF aux data of source product
- ✓ Output TOA reflectances
- ☐ Output gas corrected TOSA reflectances
- ☐ Output gas corrected TOSA reflectances of auto nn
- ☐ Output path radiance reflectances
- ☐ Output downward transmittance
- ☐ Output upward transmittance
- ✓ Output atmospherically corrected angular dependent reflectances
- ✓ Output normalized water leaving reflectances
- ☐ Output out of scope values

Figure 13. C2RCC SNAP Plugin Window [67]

Processing Parameters [67]:

- **Valid-pixel expression**
  The arithmetic expression defines the pixels which are valid for processing. Pixels that are not valid will be marked as no data in the target product.
- **Salinity**
  The value used as water salinity for the scene.
- **Temperature**
  The value used as water temperature for the scene.
- **Ozone**
  The value used as ozone if not provided by auxiliary data.
- **Air Pressure**
  The surface air pressure at sea level if not provided by auxiliary data.
- **TSM Factor**
  TSM conversion factor ($TSM = TSMfac * (iop\_btot)^{TSMexp}$). Note: This parameter was formerly named TSMfakBpart. Also, the equation to derive TSM from the IOPs has been changed in version 7.0.1 of S3TBX.
- **TSM Exponent**
  TSM conversion exponent ($TSM = TSMfac * (iop\_btot)^{TSMexp}$). Note: This parameter was formerly named TSMfakBwit. Also, the equation to derive TSM from the IOPs has been changed in version 7.0.1 of S3TBX.
- **CHL Exponent**
  Chlorophyll exponent ($CHL = CHLfak * (iop\_apig)^{CHLexp}$).
- **CHL Factor**
  Chlorophyll factor ($CHL = CHLfak * (iop\_apig)^{CHLexp}$).
- **Threshold rtosa OOS**
  Threshold for out of scope of nn training dataset flag for gas corrected TOA reflectances.
- **Threshold AC reflectances OOS**
  Threshold for out of scope of nn training dataset flag for atmospherically corrected reflectances.
- **Threshold for cloud flag on down transmittance @865**
  Threshold for cloud test based on downwelling transmittance @865.
- **Atmospheric aux data path**
  Path to the atmospheric auxiliary data directory. Use either this or the specified

products on the I/O Parameters (ozone, air pressure) tab. If the auxiliary data is not available at this path, the data will automatically be downloaded.

- **Alternative NN Path**
  Path to an alternative set of neuronal nets. Use this to replace the standard set of neuronal nets. For MSI there exists a neural net where ranges of parameters were extended to moderate-to-extreme cases (C2X-Nets). However, this is a parameter that should only be used during the development of new neural nets.
- **Output AC reflectance's as RRS instead of RHOW**
  Write remote sensing reflectance's instead of water leaving reflectance's.
- **Derive water reflectance from path radiance and transmittance**
  Alternative way of calculating water reflectance. Still experimental.
- **Use ECMWF aux data of source product**
  Use ECMWF auxiliary data (total_ozone, sea_level_pressure) from the source product.
- **Output TOA reflectance's**
  Add TOA reflectance's to the target product.
- **Output gas-corrected TOSA reflectance's**
  Add TOSA reflectance's to the target product.
- **Output gas-corrected TOSA reflectance's of auto nn**
  Add TOSA reflectance's of the auto-associative neural net to the target product.
- **Output path radiance reflectance's**
  Add path radiance reflectance's to the target product.
- **Output downward transmittance**
  Add downward transmittance to the target product.
- **Output upward transmittance**
  Add upward transmittance to the target product.
- **Output atmospherically corrected angular dependent reflectance's**
  Add atmospherically corrected angular dependent reflectance's to the target product.
- **Output normalized water leaving reflectance's**
  Add normalized water leaving reflectance's to the target product.
- **Output of out-of-scope values**
  Add out of scope to the target product.

- **Output of irradiance attenuation coefficients**
  Add irradiance attenuation coefficients to the target product.
- **Output uncertainties**
  Add uncertainties to the target product.

All parameters were left as default except chlorophyll-a parameter, salinity, and temperature. For the chlorophyll-a parameter, the factor and exponent for phytoplankton absorption at a wavelength of 443 nm (represented as a_pig in Figure 11) were derived from previous measurement campaigns conducted prior to the project (section 2.2.2.2). The chosen parameterization aligned with the most recent in situ measurements available and is reported below:

$$CHL = CHLfak * (a\_pig443)^{CHLexp}$$

Where:

- CHLfak: represent chlorophyll factor, set to 19.8;
- CHLexp: represent chlorophyll exponent, set to 0.65.

A series of internal controls produces masks (flags) in output, which signal the pixels for which the neural networks have found anomalous values, in terms of reflectance's measured in single bands ("Out of Range") or in terms of the shape of the spectrum ("Out of Scope"), for which the outcome of the atmospheric correction procedure will be affected by greater uncertainty. These masks can be used in SNAP to exclude likely noisy pixels from the results.

In addition to the coefficients of the bio-optical model, other modifiable parameters are salinity and temperature. Based on evidence it has been seen that these parameters affect the results less than the coefficients of the bio-optical model. Based on the literature values, it was decided to use the value of 0.5 PSU (Practical Salinity Unit) for the salinity, while for the average temperature of the scene, the value of the air temperature at the time of acquisition was used.

In the end, the output products (L2) generated by applying C2RCC on OLCI L1 data are shown in Table 6 where the products in bold were used in the mapping of WQPs.

Product Name	Description
Rtoa 400–1020 nm	Top-of-atmosphere reflectance
Rrs 400–1020 nm	Atmospherically corrected angular dependent remote sensing reflectance
Rhow 400–1020 nm	Atmospherically corrected angular dependent water-leaving reflectance, Rhow = Rrs × π
kd489	Irradiance attenuation coefficient at 489 nm
kdmin	Mean irradiance attenuation coefficient at the three bands with minimum kd
kd_z90max	Depth of the water column from which 90% of the water-leaving irradiance comes from (1/kdmin)
iop_apig	Inherent optical properties

iop_adet	Absorption coefficient of phytoplankton pigments at 443 nm
iop_agelb	Absorption coefficient of detritus at 443 nm
iop_bpart	Absorption coefficient of gelbstoff at 443 nm
iop_bwit	Scattering coefficient of marine particles at 443 nm
iop_adg	Scattering coefficient of white particles at 443 nm
iop_atot	Detritus + gelbstoff absorption at 443 nm (iop_adet + iop_agelb)
iop_btot	phytoplankton + detritus + gelbstoff absorption at 443 nm (iop_apig + iop_adet + iop_agelb)
conc_tsm	Total suspended matter dry weight concentration (iop_bpart × 0.986 + iop_bwit × 1.72)
conc_chl	Chlorophyll-a concentration = $CHLfak * (a\_pig443)^{CHLexp}$
SD	Secchi depth = 2.39 × (kd489−0.86) [24]

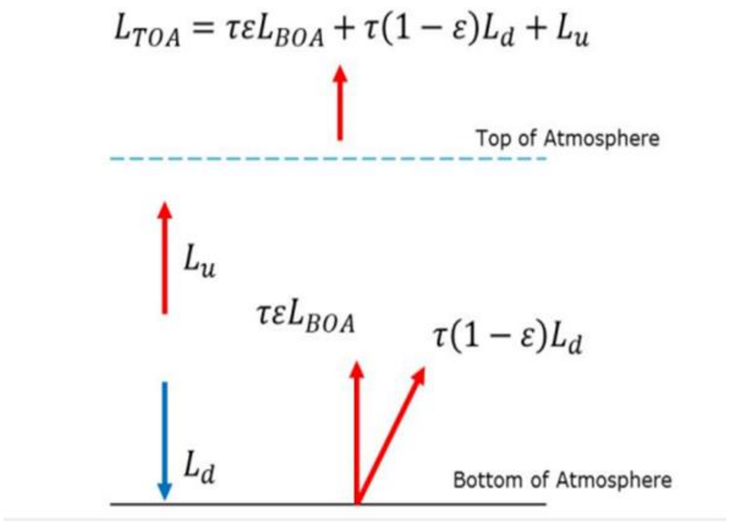
Turb1	Turbidity = 0.99 × iop_bpart + 0.24

Table 6. The output L2 products are generated by C2RCC [63].

3.2.2 Algorithmic Approach for Landsat-8 TIRS Land Surface Water Temperature Mapping

Landsat-8 TIRS L1 data used are not atmospherically corrected, removing the effects of the atmosphere in the thermal region is the essential step necessary to use the thermal band imagery for absolute temperature studies. The emitted signal leaving a target on the ground is both attenuated and enhanced by the atmosphere. Unlike other Earth observation missions, the Landsat production system does not generate derived physical parameter products, such as sea surface temperature, from the calibrated at-satellite radiance data. The NASA/GSFC Land Cover Satellite Project Science Office has created a tool that allows for the generation of surface temperature products while accounting for the influences of the atmosphere on the area of interest [72].

The approach suggested by Barsi [72] allows to derive the surface temperature starting from radiometric temperatures recorded in satellite sensors (L8-TIRS-L1). NASA has developed and made freely accessible online (https://atmcorr.gsfc.nasa.gov/) an atmospheric parameter calculator that estimates the coefficient at the desired date of atmospheric transmission, and the ascending and descending solar radiances, Lu and Ld, using local interpolations of global atmospheric profile models. These values are taken from the website for each of the thermal images of the Landsat-8 TIRS sensor acquisition date time and are used to calculate the corrected BOA (Bottom of Atmosphere) radiances, also known as LBOA, using the radiative transfer equation:



$L_{TOA} = \tau\varepsilon L_{BOA} + \tau(1-\varepsilon)L_d + L_u$
Top of Atmosphere
L_u
$\tau\varepsilon L_{BOA}$
$\tau(1-\varepsilon)L_d$
L_d
Bottom of Atmosphere

Figure 14. Radiative transfer equation [72].

Where:

L_{TOA}: is the Top-Of-Atmosphere radiance.

L_{BOA}: is the Bottom-Of-Atmosphere radiance.

τ: is the atmospheric transmission.

ε: is the emissivity of the surface.

Lu: is the Ascending solar radiance.

Ld: is the Descending solar radiance.

- In the expression, the emissivity coefficient (ε) of the surface is assumed to be constant and equal to 0.98;
- Lu, Ld, and τ are obtained from the atmospheric online calculator (section 2.2.2.4).

Starting from the BOA radiances, the surface temperature of the water is obtained by inverting Planck's equation, approximated with specific constants for Landsat 8, which can be read in the metadata of the images.

$$T = \frac{K_2}{\ln\left(\frac{K_1}{L\lambda} + 1\right)}$$

Where:

T: is the temperature in Kelvin.

K_1, K_2 are calibration constants Landsat thermal constant 8 (1321.08,774.89) respectively.

$L\lambda$ is spectral radiance (LBOA) in W/m2 ·sr·µm.

֍ The conversion from radiance to temperature in degrees (°C) can be done in SNAP via the Band Math module, applying the expression to the thermal band following, which includes the two steps foreseen by the Barsi method.

The creation of water quality parameter (WQP) maps utilizing Sentinel-3 OLCI (Ocean and Land Color Instrument) and TIRS (Thermal Infrared Sensor) data does in fact require image masking. This procedure aids in excluding pixels that could obstruct the precise assessment of water quality metrics because they are affected by various types of noise, such as clouds, ice, or sun glint.

On both sets of images, band-math functionalities are used to accomplish image masking within the implementation of each algorithm. Pixels impacted by undesired elements can be found and masked off by using precise band combinations and thresholds.

3.3 Post-processing

3.3.1 Co-registration

Co-registration is an essential step in the processing of C2RCC output data (L2 product of OLCI), as the original input data for the algorithm was obtained from different satellites S3A and S3B (sentinel-3A, sentinel-3B), as it ensures that the images from different time periods are accurately aligned. All images of the sentinel-3 will be adjusted to match the spatial referencing of the master image (S3A_IT_20221105T093500_L1). The selection of the master image was done with respect to the clarity of the image over the area of interest in terms of cloud. This alignment is necessary to compare and analyze changes in the Earth's surface over time (change detection).

The co-registration was achieved through a process called eFolki [76], which is a fast and robust optical-flow estimation technique derived from Lucas-Kanade [76]. It has a remarkably simple and parallel structure, which makes it ideally suited for massively parallel computing architectures to handle large images. [75]

The eFolki algorithm evaluates the displacement between two intensity images without external data (i.e., control points). The robustness makes the registration method efficient under all conditions encountered: change detection under non-interferometric conditions and relief effects. The accuracy of the estimated and determined offset is on the order of one-tenth of a pixel in the most difficult configurations. [75].

The importance of co-registration in processing is highlighted in several studies. For example, in a study by Z. Li and J. Bethel [77], the authors used Sentinel-3 OLCI data to monitor water quality in the Yangtze River Estuary. They found that co-registration errors can significantly affect the accuracy of water quality measurements and emphasized the need for accurate co-registration.

3.3.2 Outlier Rejection

As indicated by Toro [71], the comparison with the in-situ measures provided by project partners as well as the inspection of the statistics of the WQP maps showed the need for the filtering of data to mark outliers and exclude them from the maps. However, in the case of WQPs, we are referring to values that could pertain to specific local behaviors, which implies that the filtering must avoid possible reasonable anomalies due to physical reasons. Thus, the time series of in-situ data have been considered, and the limnologists who study the waters of the lakes under study have been interviewed, to select plausible ranges of values. To perform the outlier rejection, a 3-sigma filtering has been applied to single out data that showed behavior different from that of the lake population. Then, the out-of-range values detected with the 3-sigma filtering were explored on the map to interpret the reason for the anomaly in terms of geographical location (e.g., lower surface temperature values could be reasonable when detected in an area affected by inflowing waters, higher surface temperature due to shallow waters, etc.) or image characteristics (e.g., lower temperatures due to cloud coverage). The goal of the entire filtering process was to remove pixels that could have resulted in the wrong measurement while preserving as much information as possible.

Chapter 4: Implementation

4.1 Tools and Technologies

The purpose of this implementation is to introduce a Graphical User Interface (GUI) into the existing tool that produces Chlorophyll-a (CHL) and Total Suspended Matter (TSM) maps, as well as Land/Water Surface Temperature (LWST) maps. The existing tool lacks a user-friendly interface, which poses challenges for users in terms of importing parameters, downloading images, and initiating the processing to obtain desired outputs. These limitations will be addressed by developing a GUI that streamlines the user experience and simplifies the workflow. By designing and integrating a GUI into the existing tool, the interaction between the tool and the user will be improved, enabling a smoother and more efficient user experience.

Initially, the idea was to build a plugin in QGIS that could handle the process. However, it was discovered that configuring the Snappy library in QGIS required Python versions 3.5 or 3.6. Unfortunately, for higher Python versions, a manual build of jpy (Python-Java Bridge) was needed, which involves advanced programming skills. This limitation made the plugin less valuable, as it would force users to work with only Python 3.5 or 3.6. Therefore, an alternative approach was chosen, which involves creating a GUI using the Tkinter library instead.

4.1.1 Software Tools

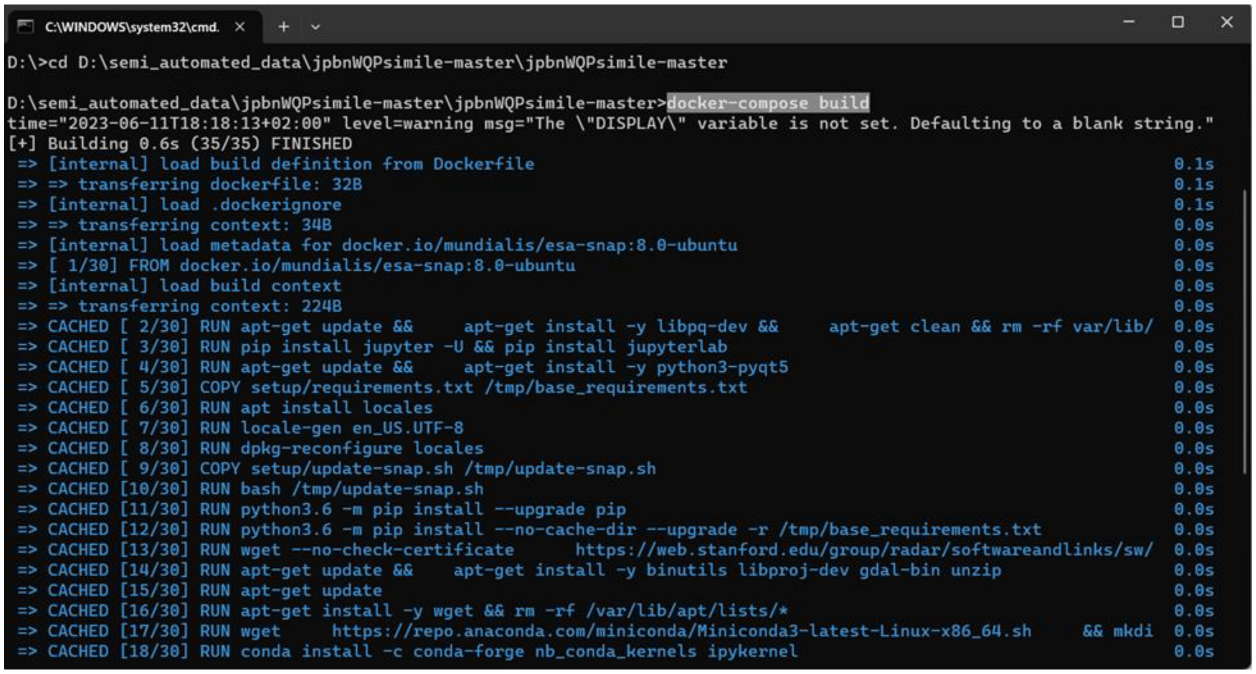
Various instruments and techniques were employed in conducting the analyses carried out in this work. These tools and methods played a crucial role in collecting, processing, and analyzing the data, enabling comprehensive insights and accurate results. The combination of these instruments and techniques facilitated a multidimensional approach to address the research problem effectively. In the following sections, we will provide an overview of the specific instruments and techniques utilized, highlighting their significance in the analysis process.

4.1.1.1 Docker Desktop

Docker Desktop is an application that enables the rapid and secure development and sharing of containerized applications and microservices across macOS, Linux, and Windows platforms. It includes an embedded Kubernetes setup for seamless app development and supports certified images, templates, and a wide range of languages and tools. Integration with Docker Hub allows for extending the development environment to a secure repository, enabling auto-building, continuous integration, and collaboration [68]. Using containers, Docker packages software into standardized units that encompass all necessary components, such as libraries, system tools, code, and runtime. When a container image runs on Docker Engine, it transforms into a container, ensuring consistent operation regardless of the underlying infrastructure. Whether it's a Linux or Windows application, containerized software maintains uniformity and runs reliably across various environments, including development and staging. The isolation provided by containers safeguards the software's functionality and promotes consistent performance.

In response to our application's requirements, the adoption of Docker Compose becomes essential. The specific needs of our application demand a multi-container architecture, emphasizing the need for seamless collaboration among various services. Docker Compose serves as an optimal solution for managing and orchestrating these interconnected services, streamlining their interaction, and ensuring efficient operation.

To run the tool and establish a Virtual Machine (VM) within a Docker container, the following steps are taken. First, Docker Desktop is ensured to be installed on the system as the platform for containerization. Next, the directory where the VM Docker container files are located is accessed using the command prompt. The VM Docker container is built by executing the command "docker-compose build" in the command prompt, with the directory of the folder containing the Docker Compose file specified. This triggers the build process, resulting in the fetching of the necessary dependencies and configurations as specified in the Docker Compose file. Once the build process is successfully completed, the container can be run using the command "docker-compose up -d". This command initiates the launch of the container in detached mode, allowing it to operate in the background. By following these steps, the tool can be set up and operational, facilitating the establishment of a Virtual Machine environment within a Docker container. The utilization of containerization benefits, including consistency, isolation, and portability, enables efficient management and deployment of the tool.



```
D:\>cd D:\semi_automated_data\jpbnWQPsimile-master\jpbnWQPsimile-master

D:\semi_automated_data\jpbnWQPsimile-master\jpbnWQPsimile-master>docker-compose build
time="2023-06-11T18:18:13+02:00" level=warning msg="The \"DISPLAY\" variable is not set. Defaulting to a blank string."
[+] Building 0.6s (35/35) FINISHED
 => [internal] load build definition from Dockerfile                                                        0.1s
 => => transferring dockerfile: 32B                                                                         0.1s
 => [internal] load .dockerignore                                                                           0.1s
 => => transferring context: 34B                                                                            0.0s
 => [internal] load metadata for docker.io/mundialis/esa-snap:8.0-ubuntu                                    0.0s
 => [ 1/30] FROM docker.io/mundialis/esa-snap:8.0-ubuntu                                                    0.0s
 => [internal] load build context                                                                           0.0s
 => => transferring context: 224B                                                                           0.0s
 => CACHED [ 2/30] RUN apt-get update &&     apt-get install -y libpq-dev &&     apt-get clean && rm -rf var/lib/  0.0s
 => CACHED [ 3/30] RUN pip install jupyter -U && pip install jupyterlab                                     0.0s
 => CACHED [ 4/30] RUN apt-get update &&     apt-get install -y python3-pyqt5                               0.0s
 => CACHED [ 5/30] COPY setup/requirements.txt /tmp/base_requirements.txt                                   0.0s
 => CACHED [ 6/30] RUN apt install locales                                                                  0.0s
 => CACHED [ 7/30] RUN locale-gen en_US.UTF-8                                                               0.0s
 => CACHED [ 8/30] RUN dpkg-reconfigure locales                                                             0.0s
 => CACHED [ 9/30] COPY setup/update-snap.sh /tmp/update-snap.sh                                            0.0s
 => CACHED [10/30] RUN bash /tmp/update-snap.sh                                                             0.0s
 => CACHED [11/30] RUN python3.6 -m pip install --upgrade pip                                               0.0s
 => CACHED [12/30] RUN python3.6 -m pip install --no-cache-dir --upgrade -r /tmp/base_requirements.txt      0.0s
 => CACHED [13/30] RUN wget --no-check-certificate      https://web.stanford.edu/group/radar/softwareandlinks/sw/  0.0s
 => CACHED [14/30] RUN apt-get update &&    apt-get install -y binutils libproj-dev gdal-bin unzip          0.0s
 => CACHED [15/30] RUN apt-get update                                                                       0.0s
 => CACHED [16/30] RUN apt-get install -y wget && rm -rf /var/lib/apt/lists/*                               0.0s
 => CACHED [17/30] RUN wget     https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh     && mkdi  0.0s
 => CACHED [18/30] RUN conda install -c conda-forge nb_conda_kernels ipykernel                              0.0s
```

Figure 15. Build the image.

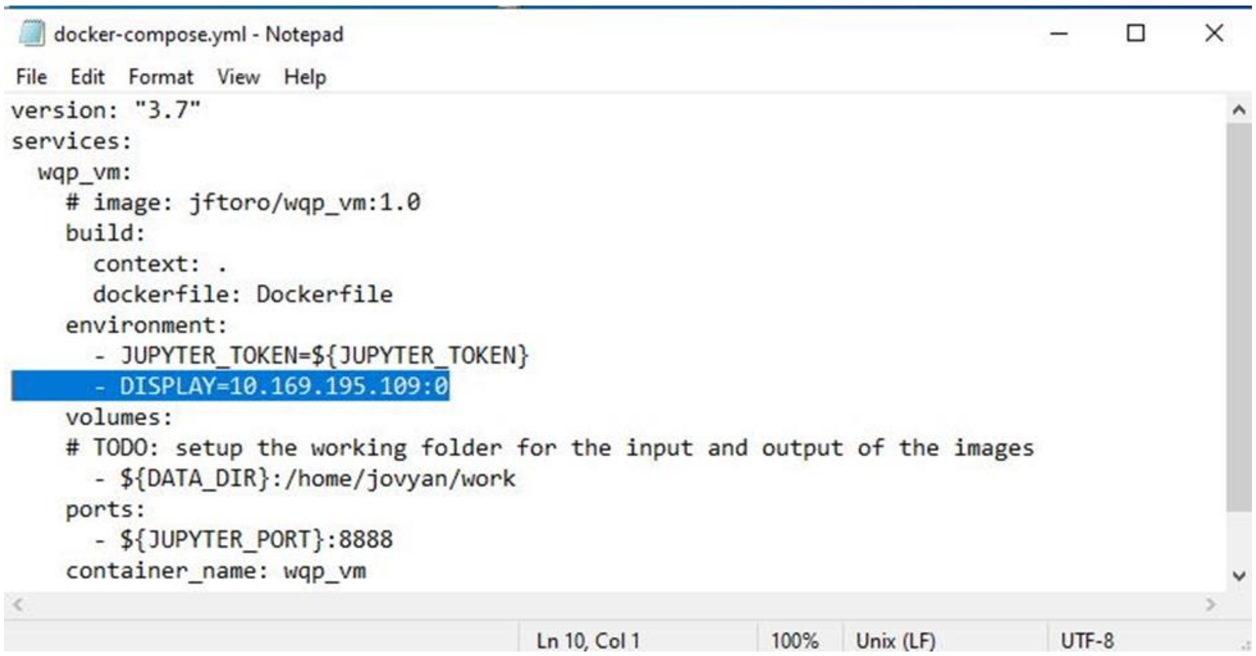
4.1.1.2 Xming

In the realm of Docker containers, which are lightweight and isolated environments, there arises a challenge when it comes to running graphical user interface (GUI) applications. By default, Docker containers lack the essential libraries and components required for GUI functionality, and they are typically run in headless mode without access to a display. This poses a problem for Unix-based systems seeking to execute graphical programs within Docker containers. However, the solution lies in utilizing Xming, an X Window System server designed for Windows. Xming acts as a window to the graphical environment of the host machine, enabling communication and displaying the GUI output from the Docker container. By leveraging Xming, the limitations of Docker containers are overcome, granting them the necessary graphical capabilities to run GUI applications seamlessly.

To integrate Xming with Docker, the X socket located at /tmp/.X11-unix on the host machine needs to be made accessible to the container. This is achieved by mounting the contents of the /tmp/.X11-unix directory into a Docker volume assigned to the container and using the host networking mode. This setup allows the container to connect to the Xming server and utilize its graphical capabilities.

In addition to mounting the X socket, the container must be configured with the appropriate DISPLAY environment variable. The DISPLAY variable instructs the X clients (graphical programs) within the container to connect to the Xming server. In the Docker Compose file, the DISPLAY environment variable is assigned the IP address of the Xming server, which establishes communication between the Docker container and the graphical environment.

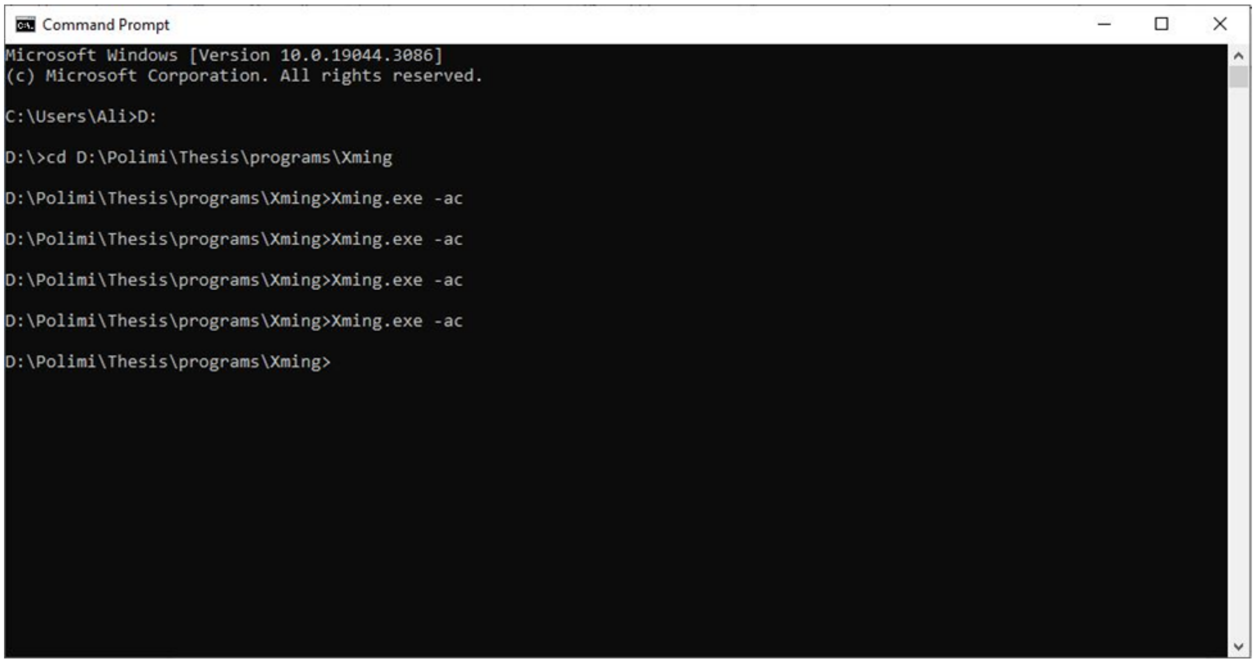
By specifying the IP address of the Xming server in the Docker Compose file, the container is configured to connect to the Xming server using the designated IP address. This assignment of the DISPLAY variable allows the graphical programs within the Docker container to identify and communicate with the Xming server effectively. It ensures proper rendering and interaction with the graphical environment, providing a seamless and immersive graphical user interface experience for the containerized application.



```
version: "3.7"
services:
  wqp_vm:
    # image: jftoro/wqp_vm:1.0
    build:
      context: .
      dockerfile: Dockerfile
    environment:
      - JUPYTER_TOKEN=${JUPYTER_TOKEN}
      - DISPLAY=10.169.195.109:0
    volumes:
    # TODO: setup the working folder for the input and output of the images
      - ${DATA_DIR}:/home/jovyan/work
    ports:
      - ${JUPYTER_PORT}:8888
    container_name: wqp_vm
```

Figure 16. Docker Compose File.

To run Xming and utilize it as the X Window System Server, the directory where the Xming executable is located should be specified. The Xming folder is then navigated to a command prompt. The following command is executed to start Xming "Xming -ac". By using the -ac option, the Xming server is configured to allow any host to connect. This enables communication between the Xming server and the Docker container.



```
Microsoft Windows [Version 10.0.19044.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ali>D:

D:\>cd D:\Polimi\Thesis\programs\Xming

D:\Polimi\Thesis\programs\Xming>Xming.exe -ac

D:\Polimi\Thesis\programs\Xming>Xming.exe -ac

D:\Polimi\Thesis\programs\Xming>Xming.exe -ac

D:\Polimi\Thesis\programs\Xming>Xming.exe -ac

D:\Polimi\Thesis\programs\Xming>
```

Figure 17. Start Xming.

4.1.2 Python Libraries

In this project, Python was used to develop a tool and GUI. Python is a versatile, beginner-friendly programming language known for its simplicity and readability. It was created by Guido van Rossum in 1991. Python is widely used for web development, software development, mathematics, and system scripting. Its high-level nature and built-in data structures make it easy to learn and use. With a thriving open-source community, Python offers a vast array of libraries and functionality, making it a popular choice for developers seeking rapid application development and code reuse.

4.1.2.1. Snappy

Snappy is the primary library of this project for producing WQP maps. It is a Python package developed by ESA. This package serves as a Python interface for ESA's open-source software SNAP (Sentinel Application Platform), which has been validated for use in inland aquatic environments [47]. Therefore, a primary library allows the use of the functionalities, algorithms, and operations offered by SNAP through the Python programming language.

4.1.2.2 Tkinter

For the creation of the GUI in this project, the Tkinter library is utilized. Tkinter is an open-source and portable graphical user interface (GUI) library specifically designed for use in Python scripts. It serves as a Python interface to the Tk GUI toolkit, which is the same toolkit used by Tcl/Tk and Perl. The Tk library, implemented in C, forms the foundation for Tkinter's functionality. As a result, Tkinter can be described as a multi-layered implementation, leveraging the underlying C implementation of the Tk library to provide a user-friendly and robust GUI framework for Python applications.

Libraries	Description
os	The os module in Python allows for cross-platform interaction with the operating system, offering functions for file/directory management, path manipulation, and system command execution.

sys	The sys module in Python provides access to system-specific parameters and functions, allowing for interaction with the Python interpreter and operating system. Also, it’s used to modify the system path in order to import libraries from a specific directory.
Pandas	Pandas is a powerful open-source data analysis and manipulation library for Python. It provides data structures like DataFrames for efficiently handling and analyzing structured data, along with functions for data cleaning, transformation, and statistical operations.
numpy	NumPy is a fundamental library for scientific computing in Python
geopandas	GeoPandas is an open-source Python library that extends the functionality of Pandas to include geospatial data analysis.
rasterio	Rasterio is a Python library that provides a simple and efficient way to read, write, and manipulate raster datasets

Rasterstats	Rasterstats is a Python library that calculates statistics and performs zonal analysis on raster datasets based on vector geometries.
Matplotlib	Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in a variety of formats. It provides a comprehensive set of plotting functions and customization options for creating high-quality graphs, charts, and figures.
Seaborn	Seaborn is a Python data visualization library built on top of Matplotlib.
base64	Base64 is a Python module that provides functions for encoding and decoding binary data using the Base64 encoding scheme. It allows for converting data into a printable ASCII format, often used for tasks like data transmission or storing binary data in text-based formats.
zipfile	The zipfile module in Python provides tools for working with ZIP archives. It allows you to create, extract, and manipulate ZIP files, making it useful for tasks such as compression, archiving, and managing multiple files in a single archive.

shutil	The shutil module in Python provides high-level file operations and utilities for copying, moving, and deleting files and directories. It simplifies common file and directory manipulation tasks, offering convenience functions to handle file operations effectively.
dotenv	The dotenv library in Python allows you to load environment variables from a .env file into your Python script or application. It simplifies the process of managing and accessing configuration settings by providing a convenient way to store and load environment-specific variables.
requests	The requests library in Python simplifies HTTP requests, allowing you to send HTTP/1.1 requests and handle responses easily. It provides a user-friendly interface for making HTTP calls, supporting various methods, headers, authentication, and handling of cookies
sodapy	The sodapy library is a Python client for the Socrata Open Data API (SODA), allowing you to retrieve and interact with data from Socrata-powered open data portals. It provides an easy-to-use interface for querying datasets, filtering results, and accessing metadata from Socrata platforms.

threading	The threading library in Python provides a way to create and manage threads, allowing for concurrent execution of multiple tasks within a single program. It simplifies thread creation, synchronization, and communication, enabling developers to implement multithreaded applications.
queue	The queue library in Python provides a thread-safe implementation of various queue data structures, such as FIFO (First-In, First-Out) queues and LIFO (Last-In, First-Out) stacks. It allows for safe sharing of data between multiple threads, ensuring synchronized access and efficient inter-thread communication.

Table 7. Python Libraries

4.1.3 Copernicus Hub

Copernicus is an open-access data platform that offers free and comprehensive access to products from the Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5P satellites. It provides valuable Earth observation data for various applications. The developed tool enables the seamless download of Sentinel-3 satellite images by leveraging the capabilities of the Copernicus Open Access Hub. By utilizing the sodapy library, the tool establishes a connection with the data hub and executes a query based on specific search criteria, including dataset ID, bounding box, date range, and product type. The retrieved images are then automatically downloaded to a designated directory. This functionality greatly simplifies the acquisition process of Sentinel-3 imagery, providing researchers with a valuable resource for their analysis and investigations. [70]

4.1.4 USGS

Downloading Landsat images from the USGS (United States Geological Survey) can be achieved using the Earth Explorer web portal (https://earthexplorer.usgs.gov/). These resources provide a convenient way to access and obtain Landsat satellite imagery for various geographical regions and time periods, facilitating research and analysis in fields such as remote sensing, environmental monitoring, and land cover assessment. In this tool, we utilized the USGS (United States Geological Survey) platform to download Landsat 8 images.

4.2 Graphical user interface

4.2.1 Run the GUI (Graphical User Interface)

To run the application, the image needs to be built and the container should be run, as demonstrated in (4.1.1.1). Subsequently, the Xming paragraph should be executed to initialize the required environment (4.1.1.2). Once that is done, the GUI file can be launched, and the resulting graphical user interface will be displayed on the Xming window, providing the desired interface for interacting with the application[2].

4.2.2 Additional Dataset Insertion

On the additional dataset insertion page of the GUI, users are provided with a range of functionalities to facilitate their data analysis and processing. This page serves as the initial interface where users can download temperature data, insert temperature readings, and input atmospheric correction parameters.

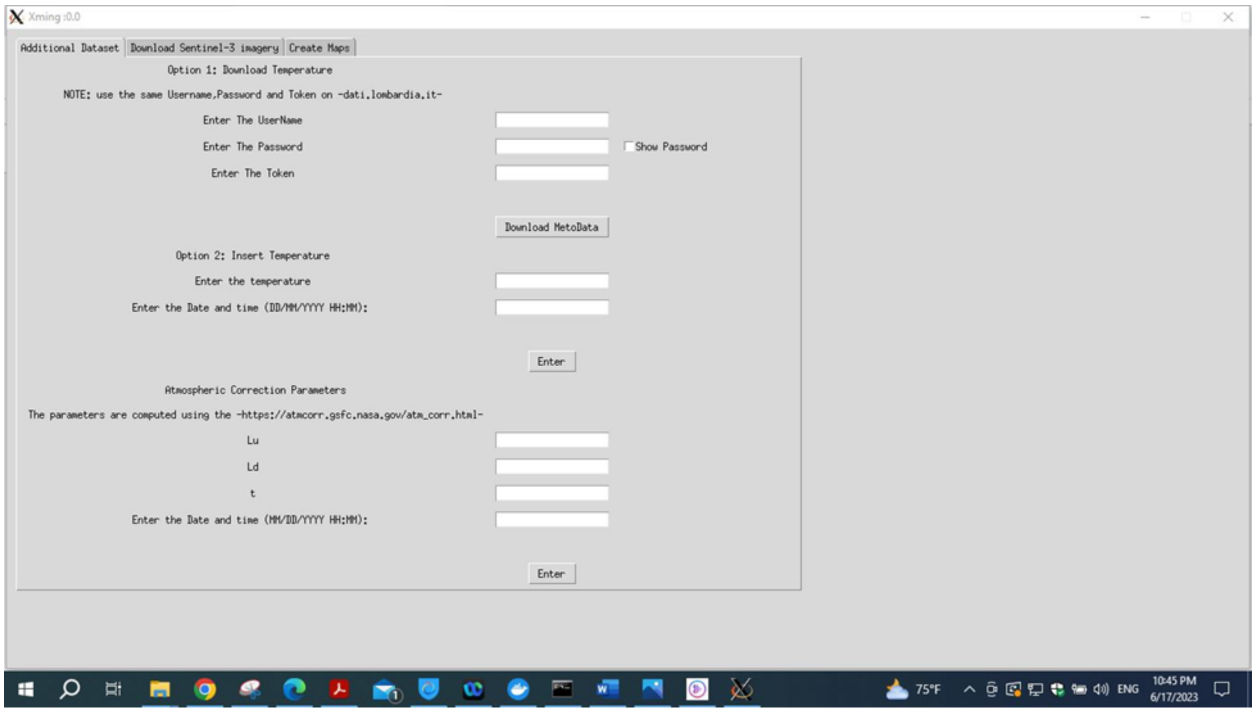
The first option available on this page is the ability to download temperature data. Users are instructed to use the same username, password, and token as their credentials on the "dati.lombardia.it" website. By entering their username, password, and token in the corresponding entry boxes, users can securely access and download the desired temperature data. The "Download Temperature " button triggers the necessary actions to retrieve the data and make it available for further analysis. The second option offered on this page is the capability to insert temperature readings manually. Users are prompted

[2] In Listing 1 you can find the Python code that used to create the GUI window

to enter the temperature value along with the corresponding date and time. By inputting this information into the designated entry boxes, users can accurately record and store the temperature data of interest. Clicking the "Enter" button ensures the successful insertion of the temperature reading into the dataset, enabling users to maintain comprehensive records. The temperature information obtained on this page serves a crucial role in the computation of Chl (chlorophyll-a) and TSM (total suspended matter) maps.

In addition to temperature data, this page also provides users with the capability to input atmospheric correction parameters specifically tailored for Landsat 8 images. Informative labels guide users through the process, indicating that these parameters are computed using the "https://atmcorr.gsfc.nasa.gov/atm_corr.html" website. Users are required to enter values for Lu, Ld, t, and the respective date and time. This information is vital for performing accurate atmospheric corrections on the dataset, enhancing the precision and reliability of subsequent analyses.

To further streamline the user experience, the page includes additional labels and entry boxes for guidance and data entry. A checkbox is also provided, allowing users to toggle the visibility of their password for added security. The tool incorporates feedback to provide users with valuable information and messages regarding the actions performed on the dataset. This feedback plays a crucial role in informing users about the progress of the download process and any potential errors that may occur during the process. For instance, if the user enters incorrect information for the username, password, token, or date, an error message will be displayed in the result label, guiding them to rectify the issue. Similarly, if any unexpected errors arise during the download or insertion process, relevant error messages will be shown to help users troubleshoot and resolve the problem effectively. This feedback mechanism ensures that users are aware of any issues and can take the necessary steps to ensure accurate data management and analysis.


Xming :0.0

Additional Dataset | Download Sentinel-3 imagery | Create Maps

Option 1: Download Temperature

NOTE: use the same Username,Password and Token on -dati.lombardia.it-

Enter The UserName

Enter The Password ☐ Show Password

Enter The Token

Download MetoData

Option 2: Insert Temperature

Enter the temperature

Enter the Date and time (DD/MM/YYYY HH:MM):

Enter

Atmospheric Correction Parameters

The parameters are computed using the -https://atmcorr.gsfc.nasa.gov/atm_corr.html-

Lu

Ld

t

Enter the Date and time (MM/DD/YYYY HH:MM):

Enter

75°F ENG 10:45 PM 6/17/2023

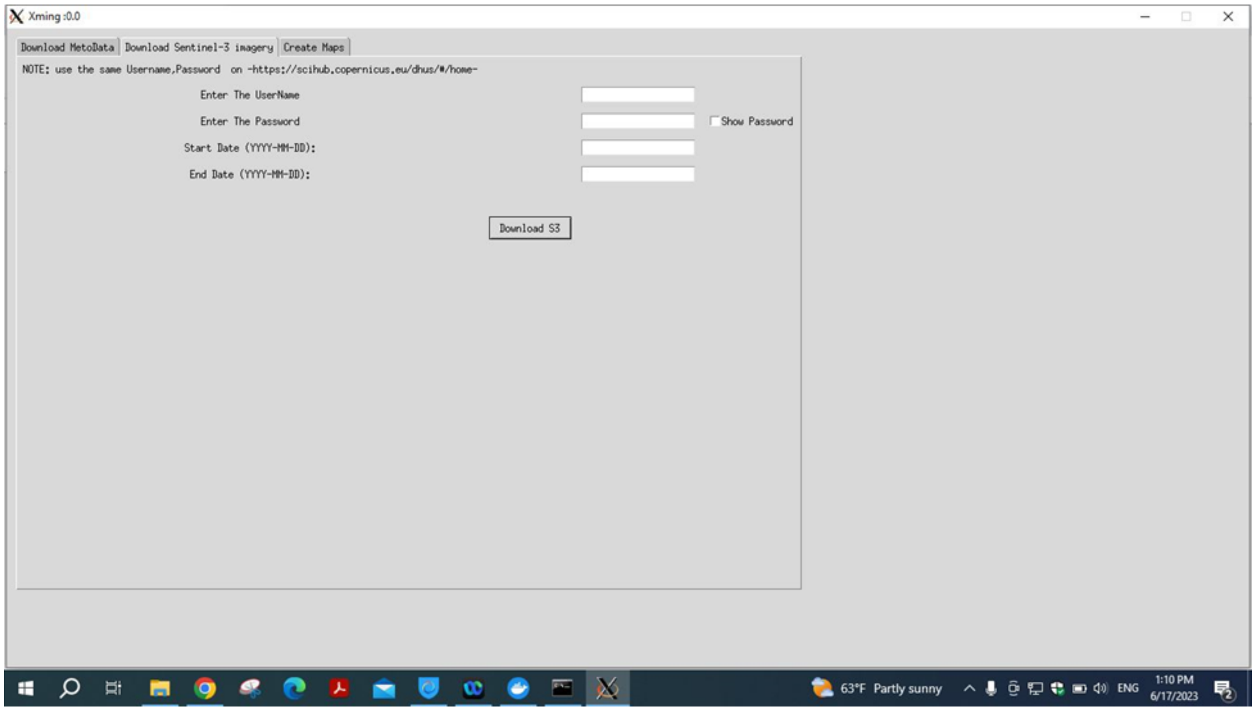
Figure 18. Additional Dataset Insertion Page[3]

4.2.3 Download Sentinel-3 Imagery

The "Download Sentinel-3" page is specifically designed for the seamless retrieval of Sentinel-3 imagery. This page features a user-friendly graphical interface that streamlines the download process from the designated website, "https://scihub.copernicus.eu/dhus/#/home-". Users are guided through the process with intuitive elements, such as input fields for entering their username and password, which need to be consistent with their credentials on the mentioned website. Additionally, the page offers convenient date range selection through designated input fields for specifying the desired start and end dates in the format of YYYY-MM-DD. To enhance security, the password characters are concealed, but users have the option to reveal them by toggling the "Show Password" checkbox. The tool incorporates feedback to provide users with valuable information and messages regarding the actions performed on the dataset. This feedback plays a crucial role in informing users about the progress of the download process and any potential errors that may occur during the process. Finally, the

[3] In Listing 3 you can find the function that used to download and insert both the temperature and Atmospheric correction parameters.

"Download S3" button, labeled as "run button" triggers the underlying function "download button click" to initiate the download process based on the provided inputs. With its user-friendly design and efficient functionality, the "Download Sentinel-3" page empowers users to effortlessly access and acquire Sentinel-3 imagery for their specific needs.



NOTE: use the same Username,Password on -https://scihub.copernicus.eu/dhus/#/home-

Figure 19. Download the Sentinel-3 page.[4]

[4] In listing 2 you can find the function used to download Sentinel-3 data

4.2.4 Create Maps

The “Create Maps” page is specifically designed to facilitate the generation of Chl (chlorophyll-a), TSM (total suspended matter), and LSWT (sea surface temperature) maps. This page offers users a comprehensive interface with user-friendly features and options to customize and create these maps. At the heart of the "Create Maps" page is a prominent "Run" button, which triggers the execution of algorithms and processes necessary for generating the desired maps. This button provides a straightforward and intuitive means for initiating the map creation process.

To enhance flexibility and user control, the page includes a combobox labeled "Select Sensor." This combobox allows users to choose the specific sensor or data source for map generation. It offers a selection of different satellite sensors or systems; these options are Sentinel-3 and Landsat 8. By providing a choice between Sentinel-3 and Landsat 8, users can leverage the distinct characteristics and data offerings of each sensor to tailor their map generation process to their specific requirements.

Furthermore, the tool provides users with valuable feedback and status updates throughout the map creation process. This feedback label dynamically displays important information such as progress updates, error messages, or notifications of successful completion. By keeping users informed, it ensures a transparent and interactive experience during map creation.

Additionally, the "Create Maps" page provides checkboxes that allow users to select the temperature data source for the map generation process. These checkboxes enable users to choose between the inserted temperature or the downloaded temperature. These checkboxes allow users to tailor the map generation process based on their preference for temperature data. This feature provides users with the flexibility to utilize either the temperature data they have inserted into the system or the downloaded temperature data, depending on their specific requirements or data availability.

The page also includes input fields for specifying the start and end dates of the data range used in map creation. These fields, labeled "Start Date (YYYY-MM-DD)" and "End Date (YYYY-MM-DD)," enable users to define the temporal scope of the analysis, ensuring the maps are generated based on the desired time period. The “Create Maps” page empowers users to effortlessly create maps based on their preferences. It provides an efficient and interactive environment for analyzing and visualizing Chl-a, TSM, and LSWT data.

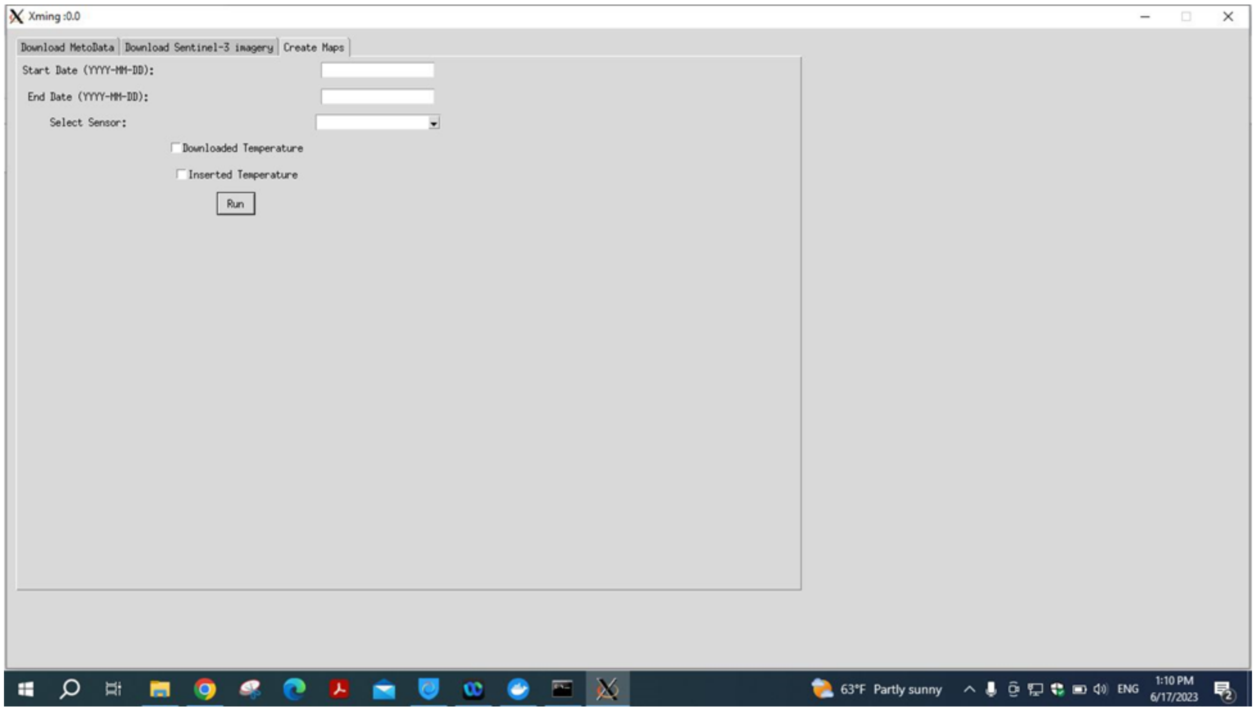


Figure 20. Create maps [5].

[5] In listing 4 and 5 you can find the function used to create TSM,Chl-a and LSWT maps

Chapter 5: Results

5.1 Output Maps

The application was run based on the data for both January and February 2023, with the number of input images shown in Table 8.

Sensor	Number of Images
Sentinel-3 A/B OLCI	86
Landsat-8 TIRS	3

Table 8. Images used to run the Application.

The application outputs provide representations of chlorophyll-a, Total Suspended Matter, and Land surface water temperature maps, collectively referred to as Water Quality Parameters (WQPs). The output of the application is presented in the following manner:

- The results of the original maps were obtained directly from the C2RCC;
- The outputs showcase the results after the images have undergone co-registration (only for OLCI products), guaranteeing precise alignment;
- The outputs demonstrate the effects of outlier rejection, wherein data is filtered to identify and exclude outliers from the maps.

In certain maps, there are areas within the lakes where no pixels are visible. This absence of pixels is a result of an image masking procedure that helps exclude pixels that could interfere with an accurate assessment of water quality metrics. These pixels may be affected by different types of noise, such as clouds or sun glints, and removing them improves the precision of the assessment.

5.1.1 Chlorophyll-a (CHL)

The chlorophyll-a maps offer valuable data regarding the concentration of chlorophyll-a in four lakes. The following figures illustrate the disparity between the maps before and after the removal of outliers, with the excluded pixels marked in red prior to outlier rejection.

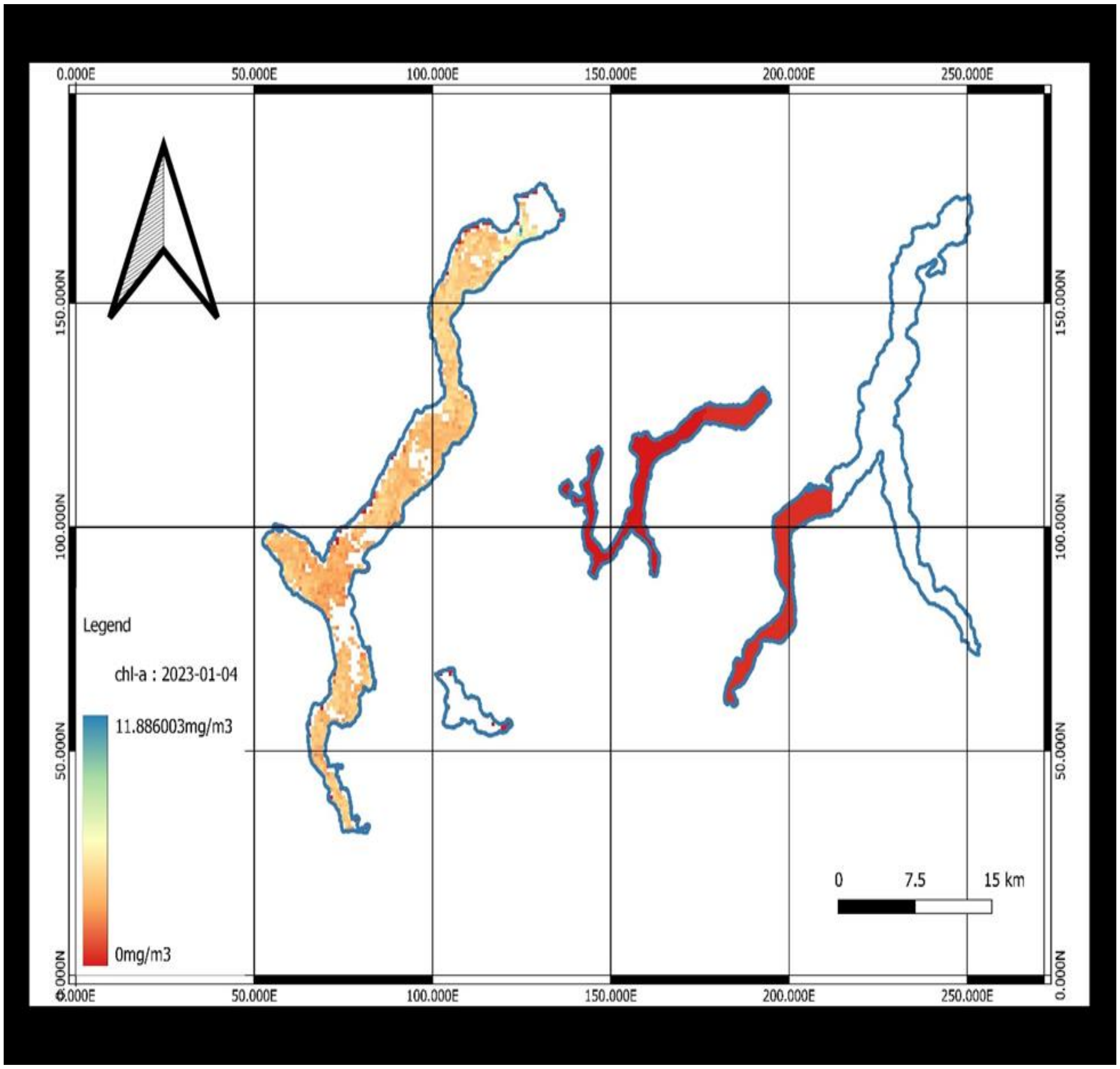


Figure 21. Chlorophyll-a (CHL) before outlier rejection (04/01/2023)

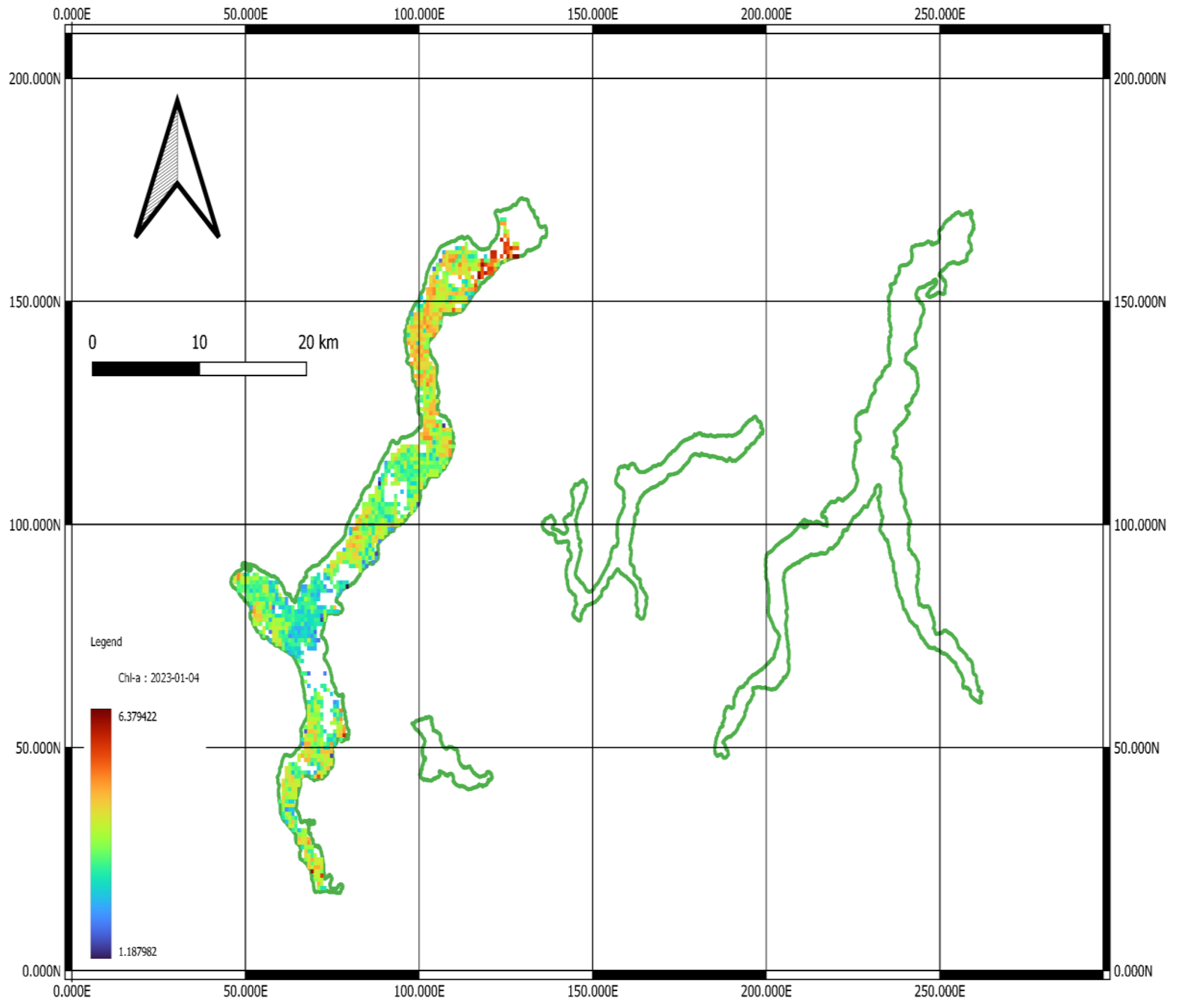


Figure 22. Chlorophyll-a (CHL) after outlier rejection (04/01/2023)

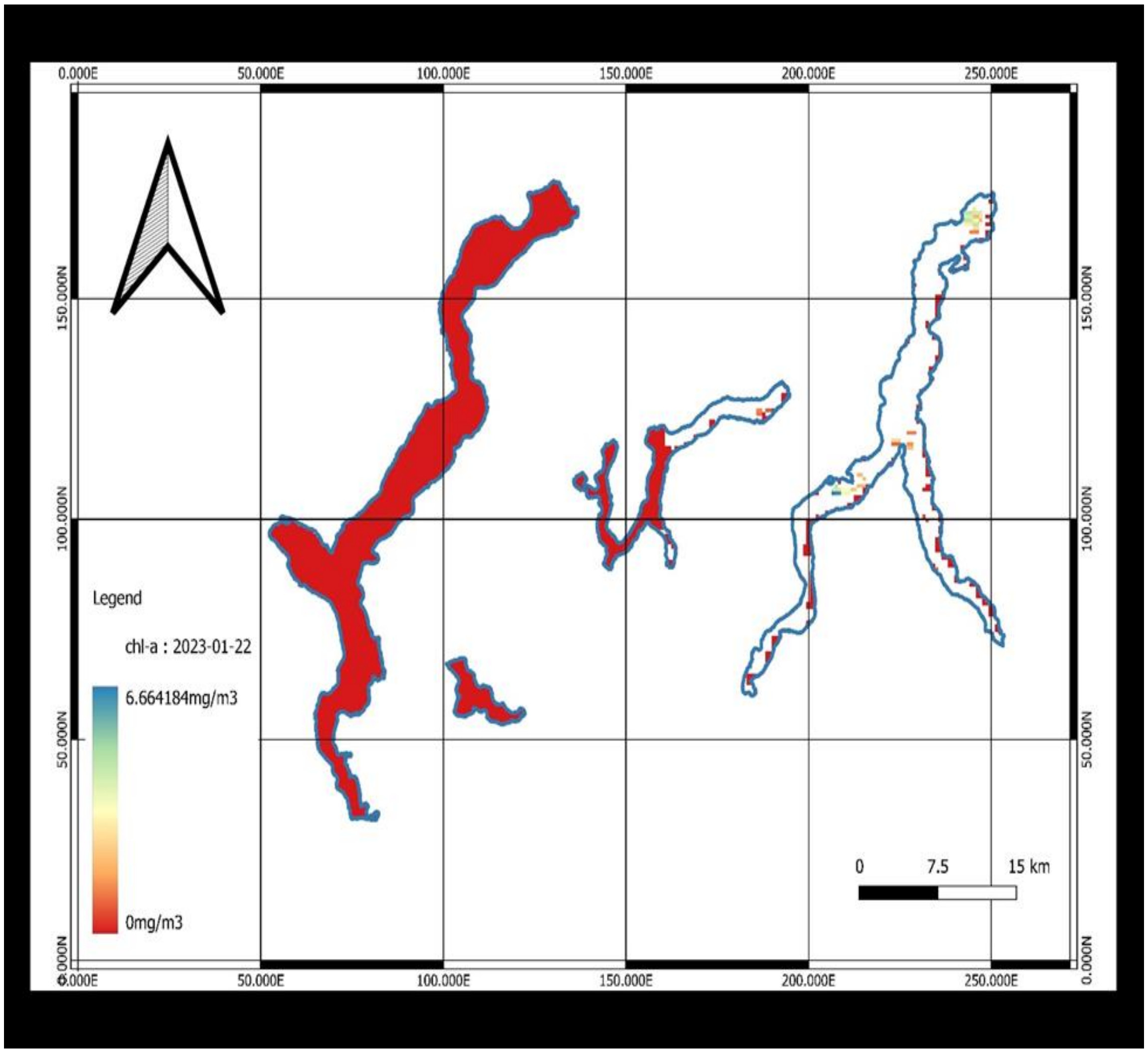


Figure 23. Chlorophyll-a (CHL) before outlier rejection (22/01/2023)

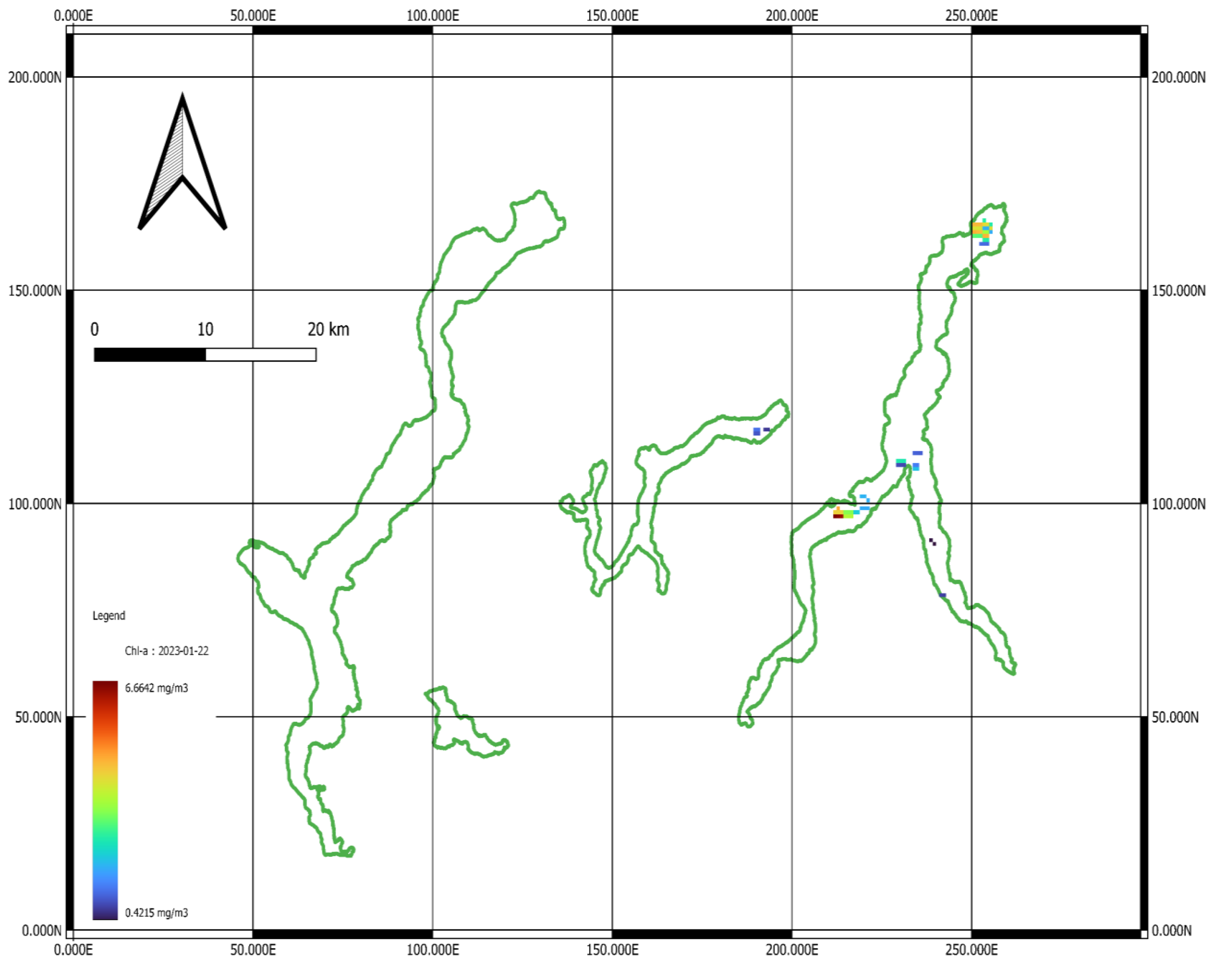


Figure 24. Chlorophyll-a (CHL) after outlier rejection (22/01/2023)

❖ The result displayed in Figure 25 was eliminated because it lacked an adequate number of pixels.

5.1.2 Total Suspended Matter (TSM)

The Total Suspended Matter maps, depicted in the chlorophyll-a maps, illustrate the data before and after undergoing outlier rejection.

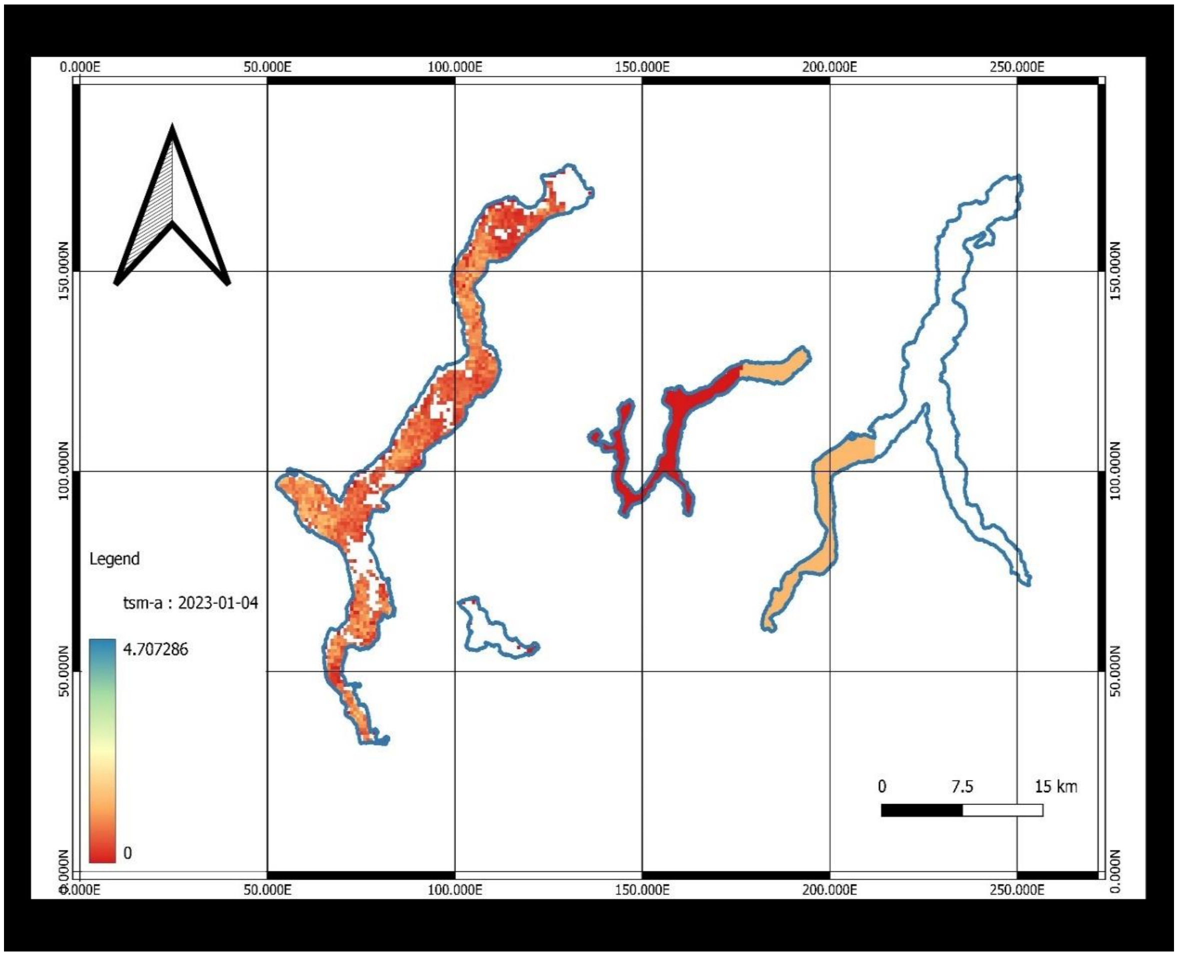


Figure 25. Total Suspended Matter (TSM) before outlier rejection (04/01/2023)

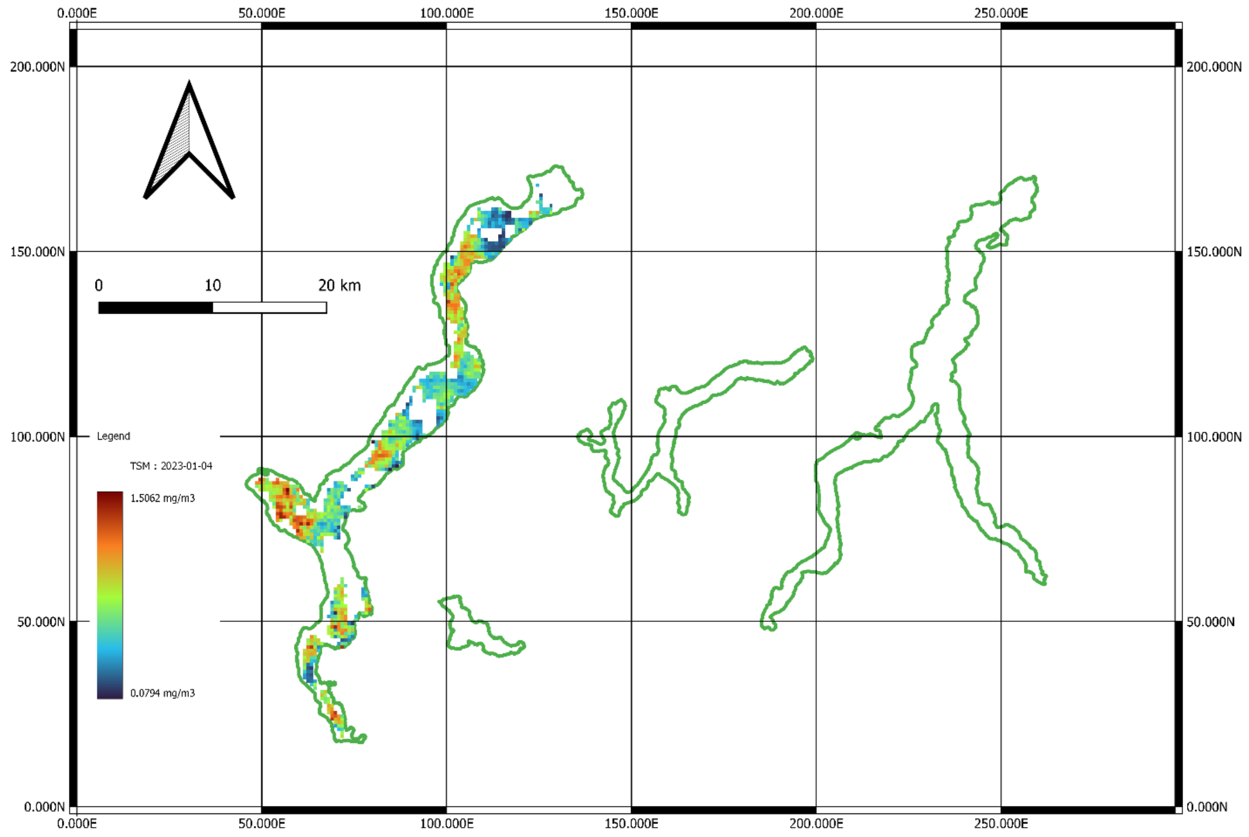


Figure 26. Total Suspended Matter (TSM) after outlier rejection (04/01/2023)

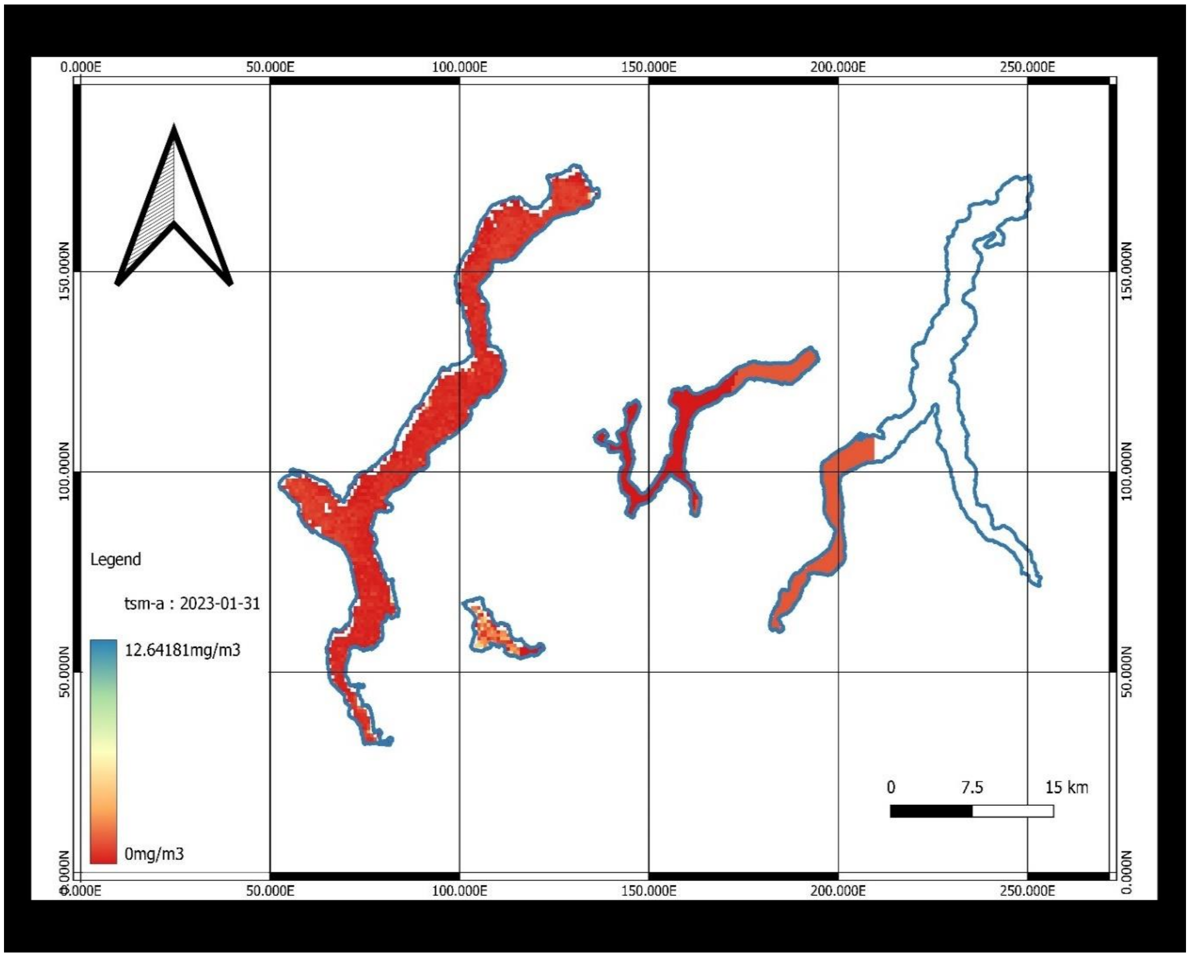


Figure 27. Total Suspended Matter (TSM) before outlier rejection (31/01/2023)

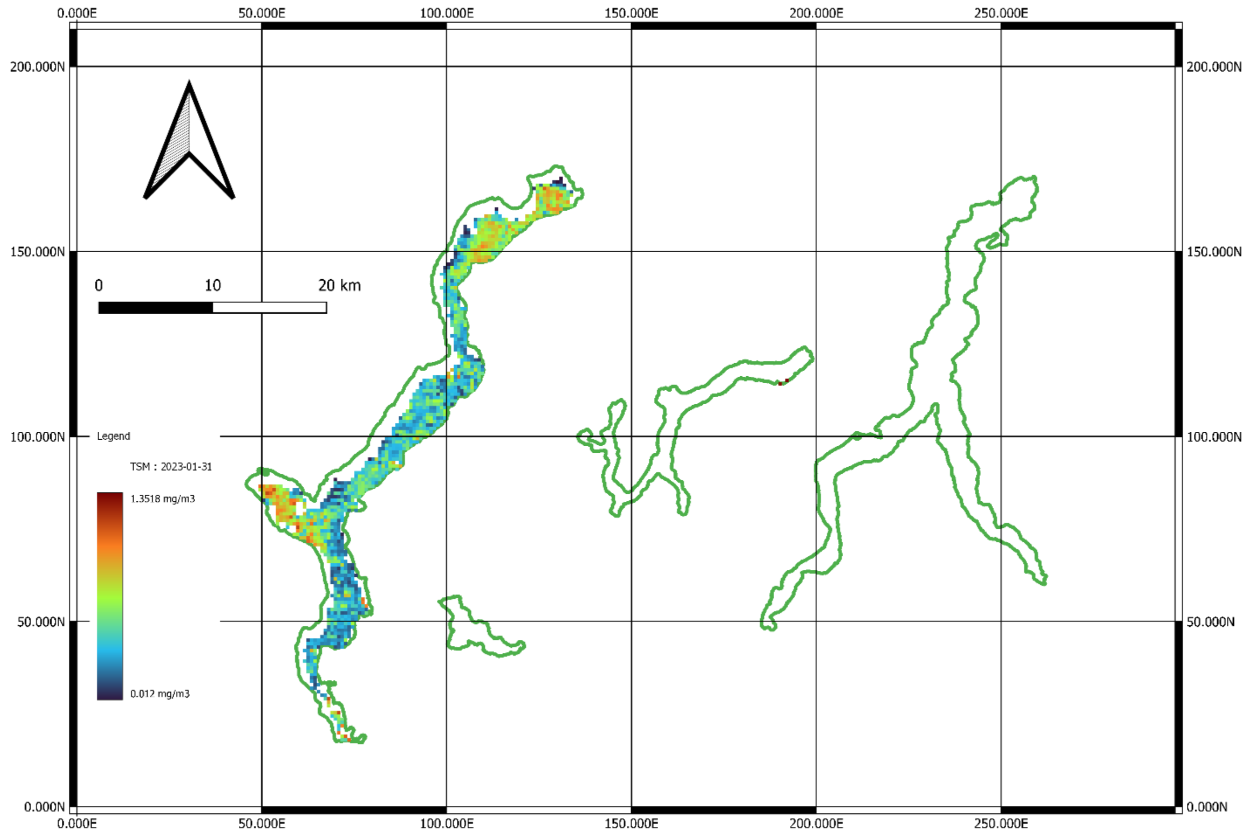


Figure 28. Total Suspended Matter after Outlier Rejection (31/01/2023)

5.1.3 Co-registered Images

These outputs were obtained after co-registration, which ensures that images from different time periods are aligned accurately. Co-registration is a process that ensures proper alignment of these images, enabling reliable comparisons and analysis (3.3.1). The figures below show output maps after co-registration and before outlier rejection.

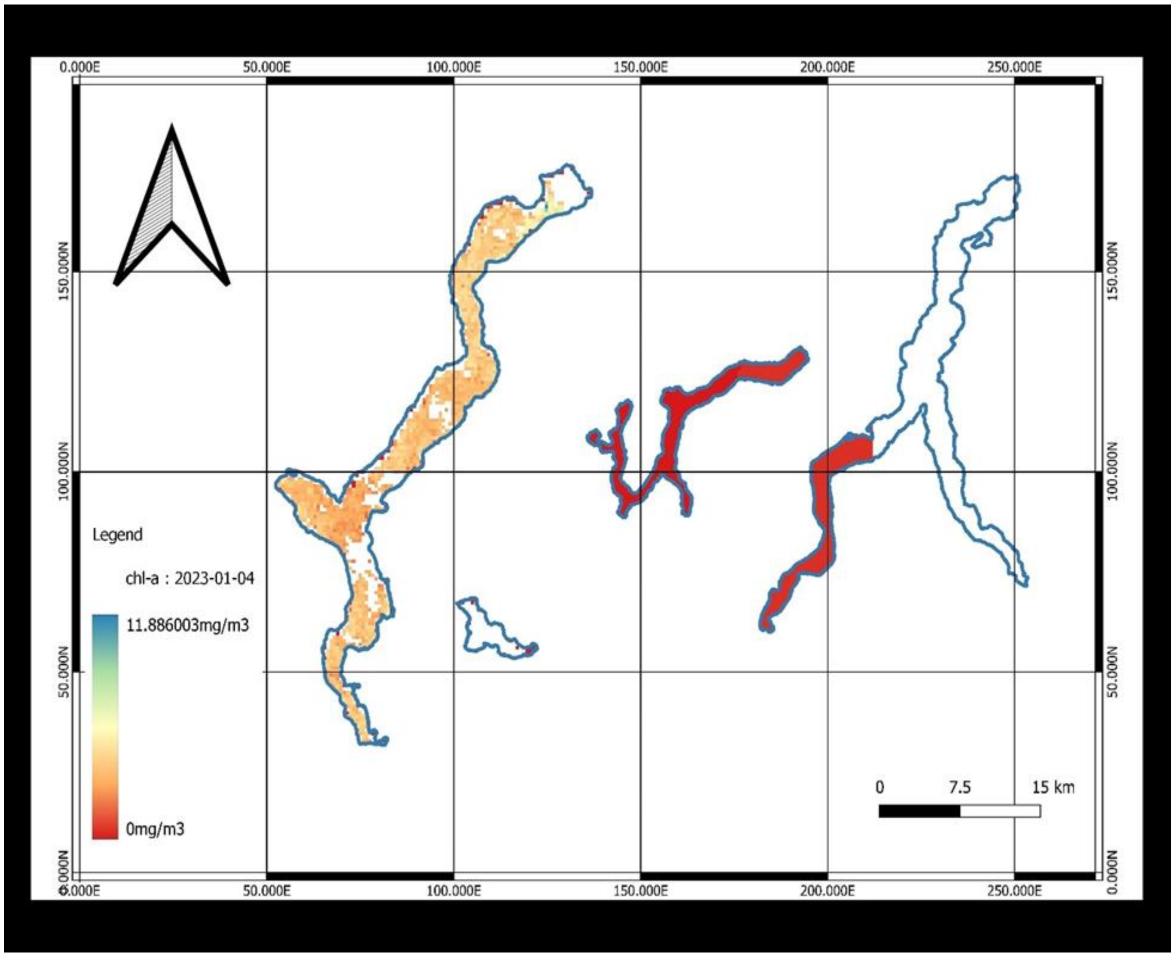


Figure 29. Co-registered Chlorophyll-a (CHL) Concentration Map (04/01/2023)

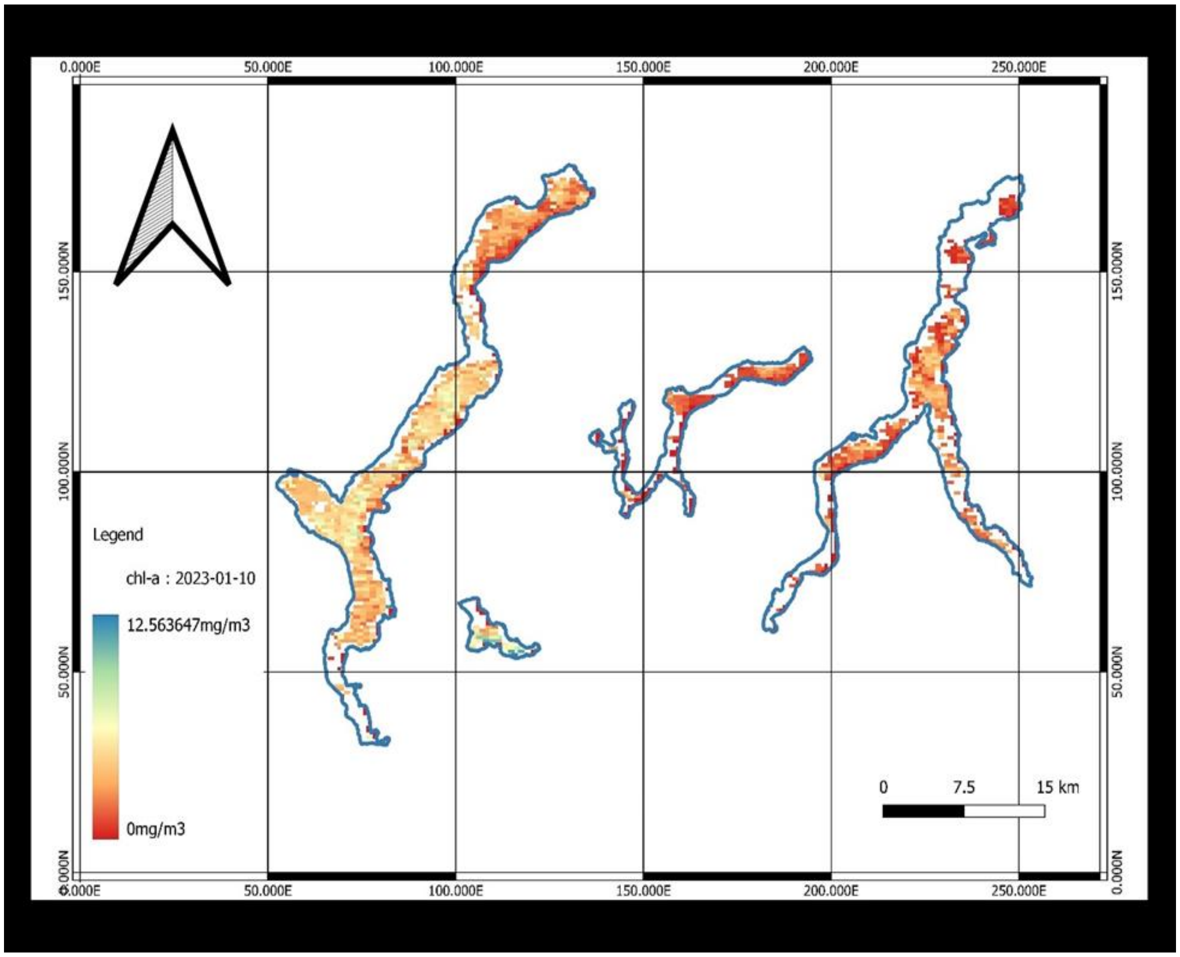


Figure 30. Co-registered Chlorophyll-a (CHL) Concentration Map (10/01/2023)

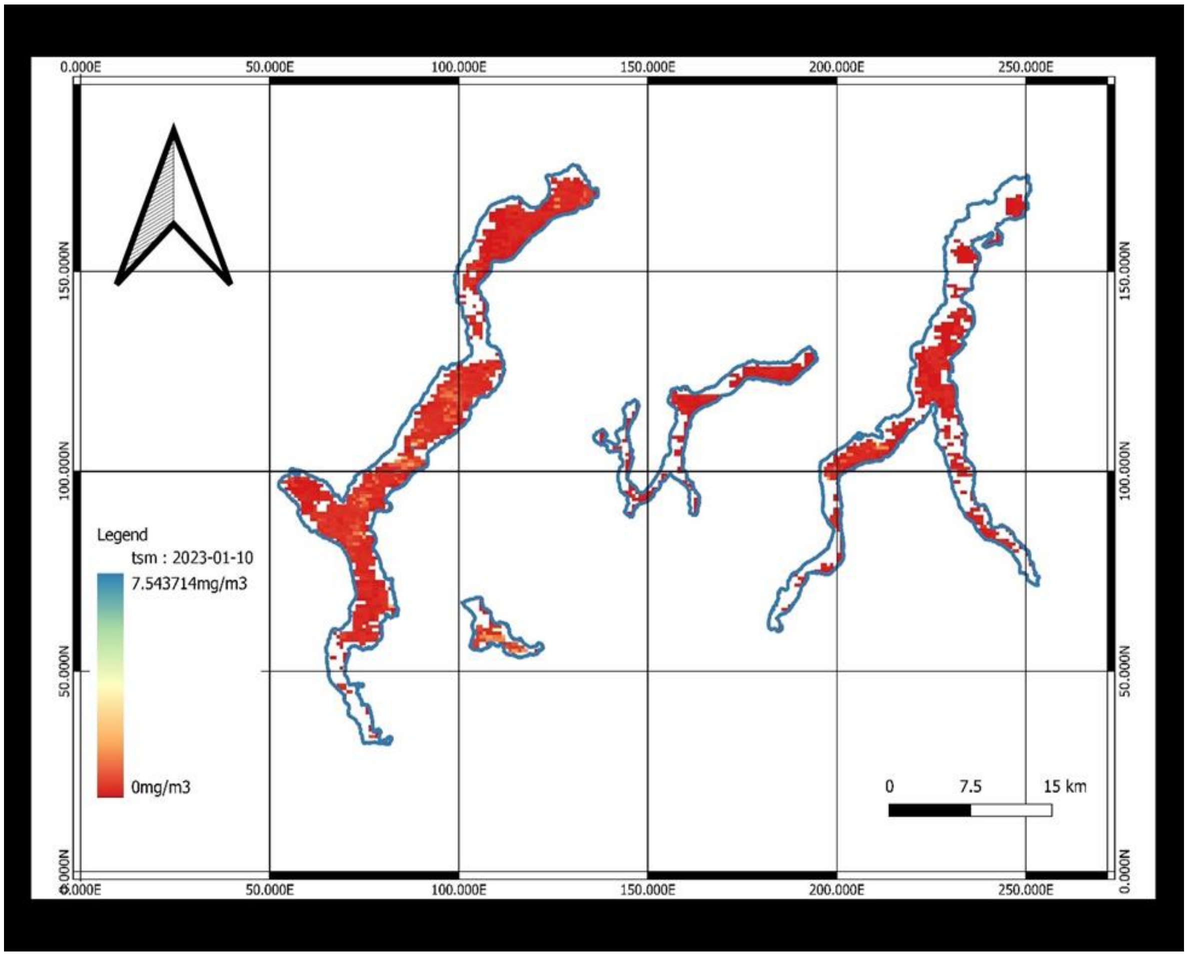


Figure 31. Co-registered Total Suspended Matter (TSM) Concentration Map (10/01/2023).

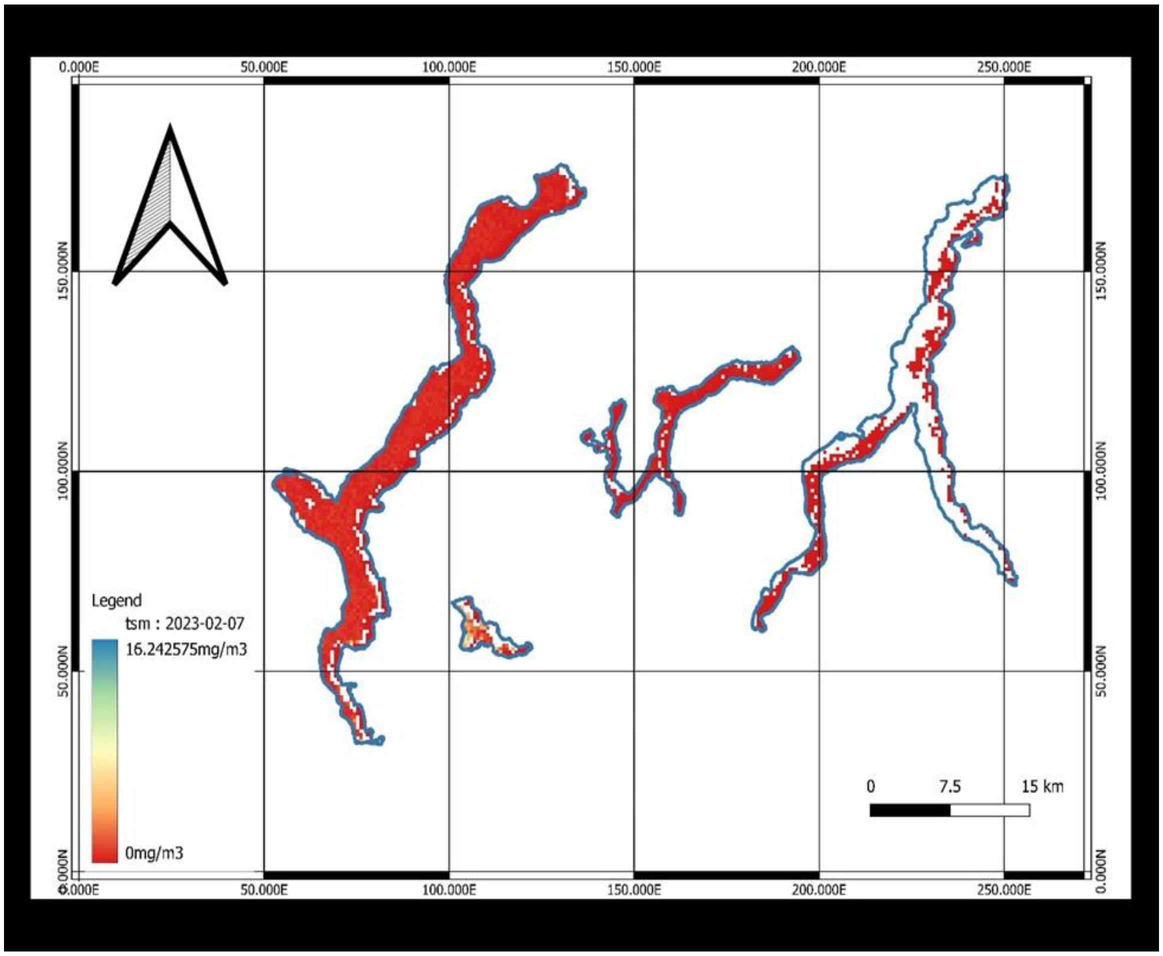


Figure 32. Co-registered Total Suspended Matter (TSM) Concentration Map (07/02/2023).

5.1.4 Land Water Surface Temperature (LWST)

The Lake Surface Water Temperature (LSWT) map offered in this application serves as a depiction of the temperature at the surface of a lake, which is a crucial indicator of both lake hydrology and biogeochemistry. By analyzing temperature trends over extended periods, it becomes possible to assess the impact of climate change on the lake's ecosystem [73].

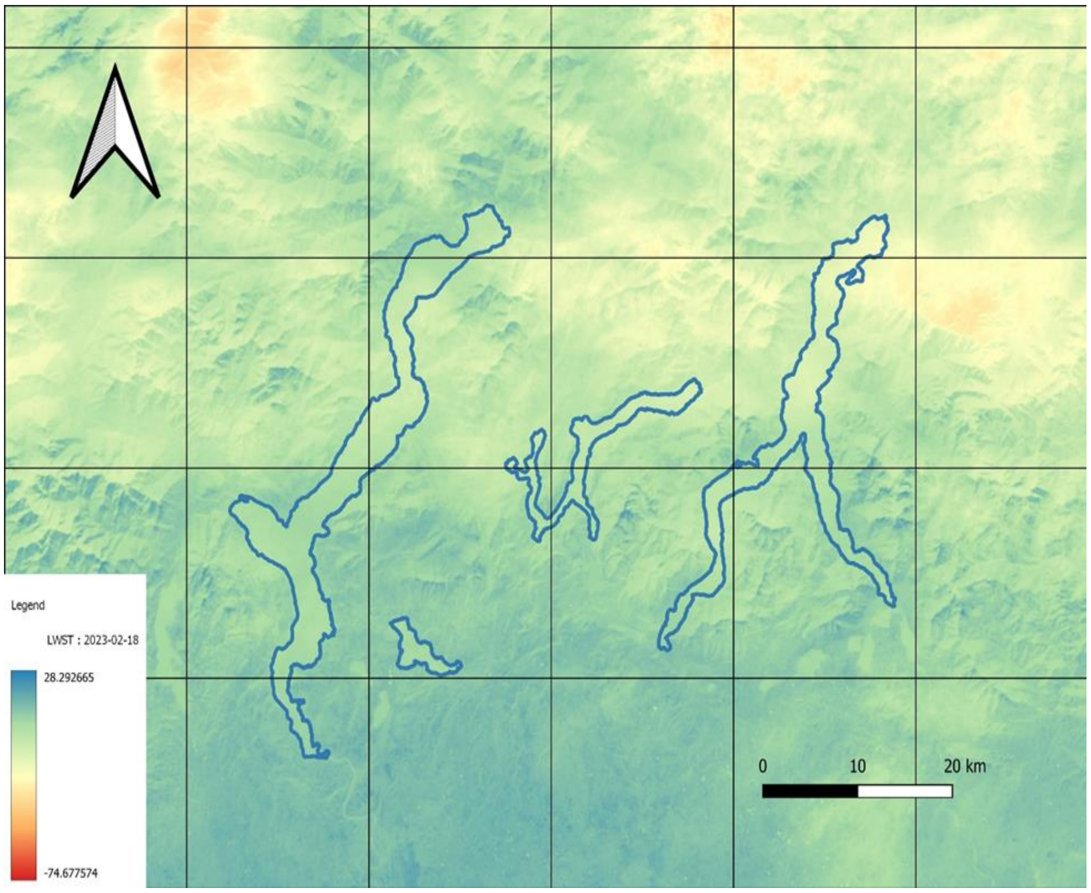


Figure 33. Land Water Surface Temperature (LWST) Map before outlier rejection (18/02/2023).

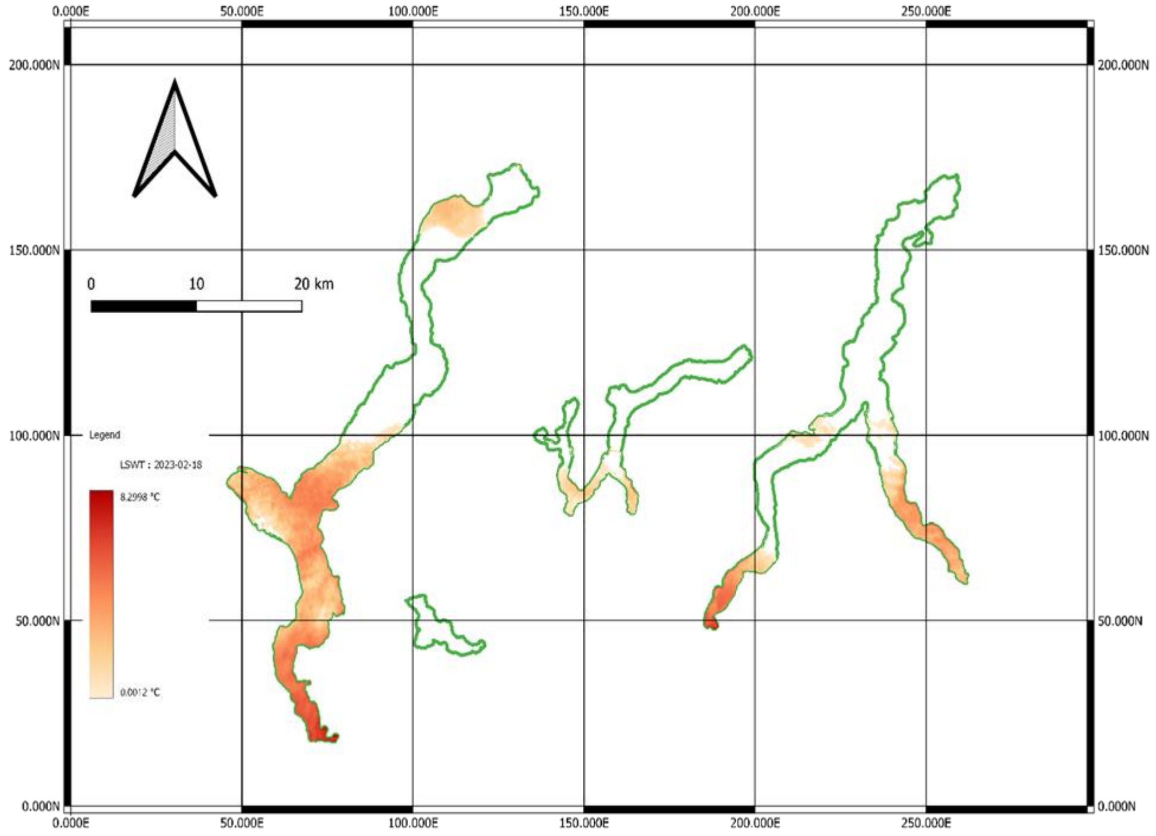


Figure 34. Land Water Surface Temperature (LWST) Map after Outlier Rejection (18/02/2023)

5.1.5 Example of good quality maps

Good quality final maps showcase a rich pixel density, effectively representing chlorophyll-a and total suspended matter values.

5.1.5.1 Chlorophyll-a (CHL)

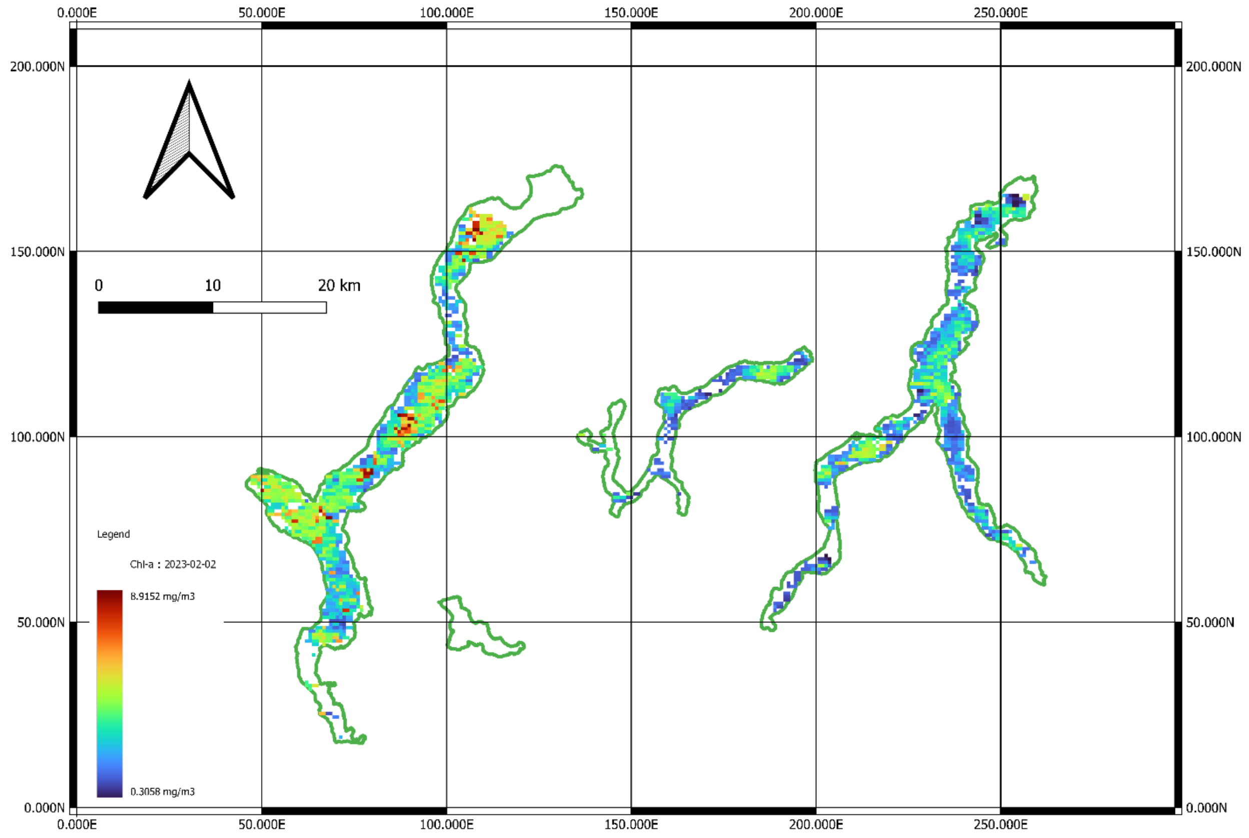


Figure 35. Chlorophyll-a map (02/02/2023)

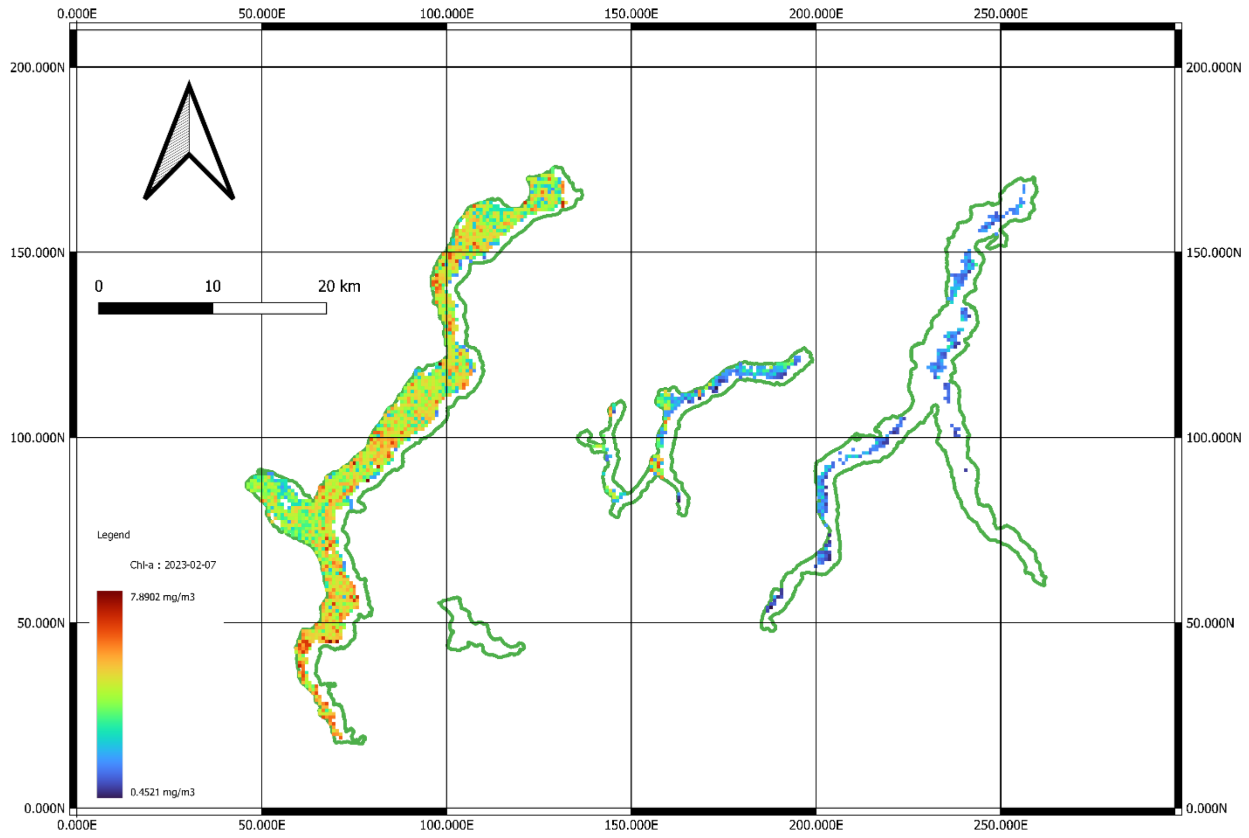


Figure 36. Chlorophyll-a map (07/02/2023)

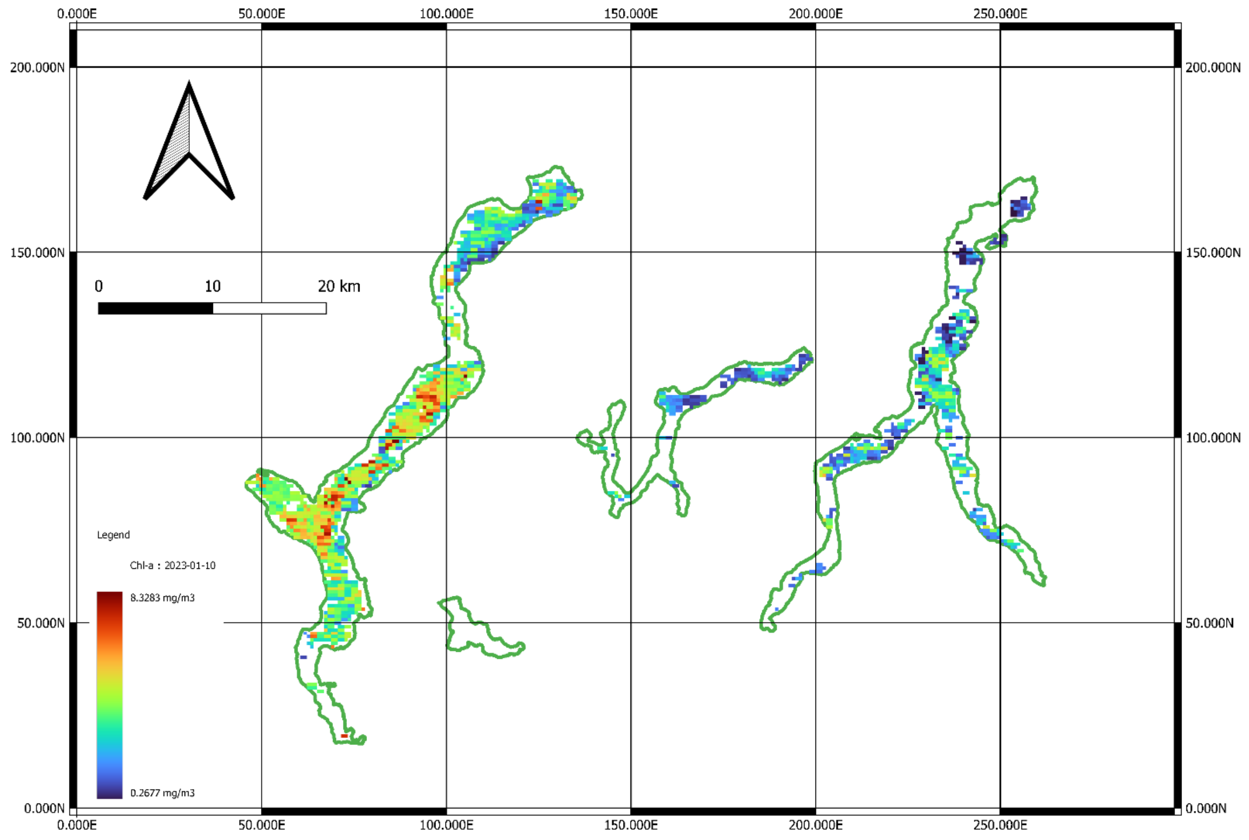


Figure 37. Chlorophyll-a map (10/01/2023)

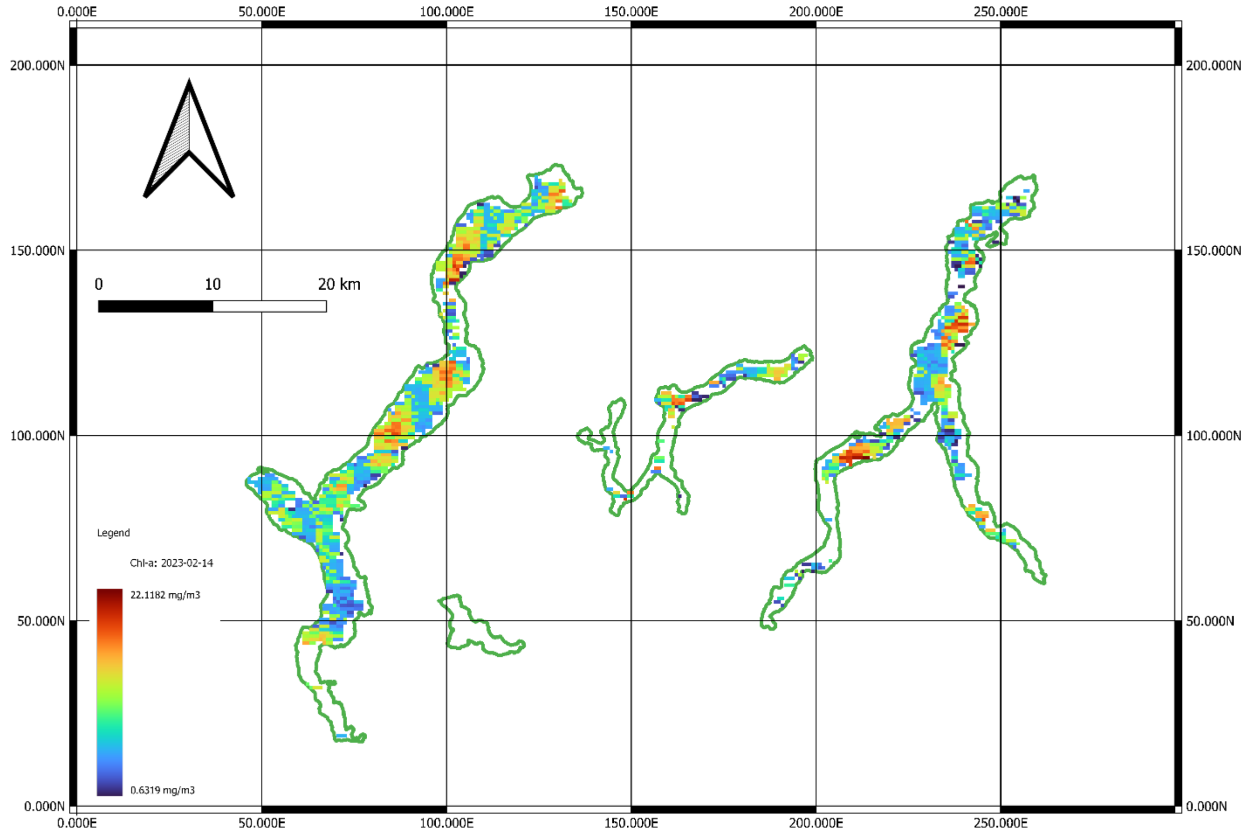


Figure 38. Chlorophyll-a map (14/02/2023)

5.1.5.2 Total Suspended Matter (TSM)

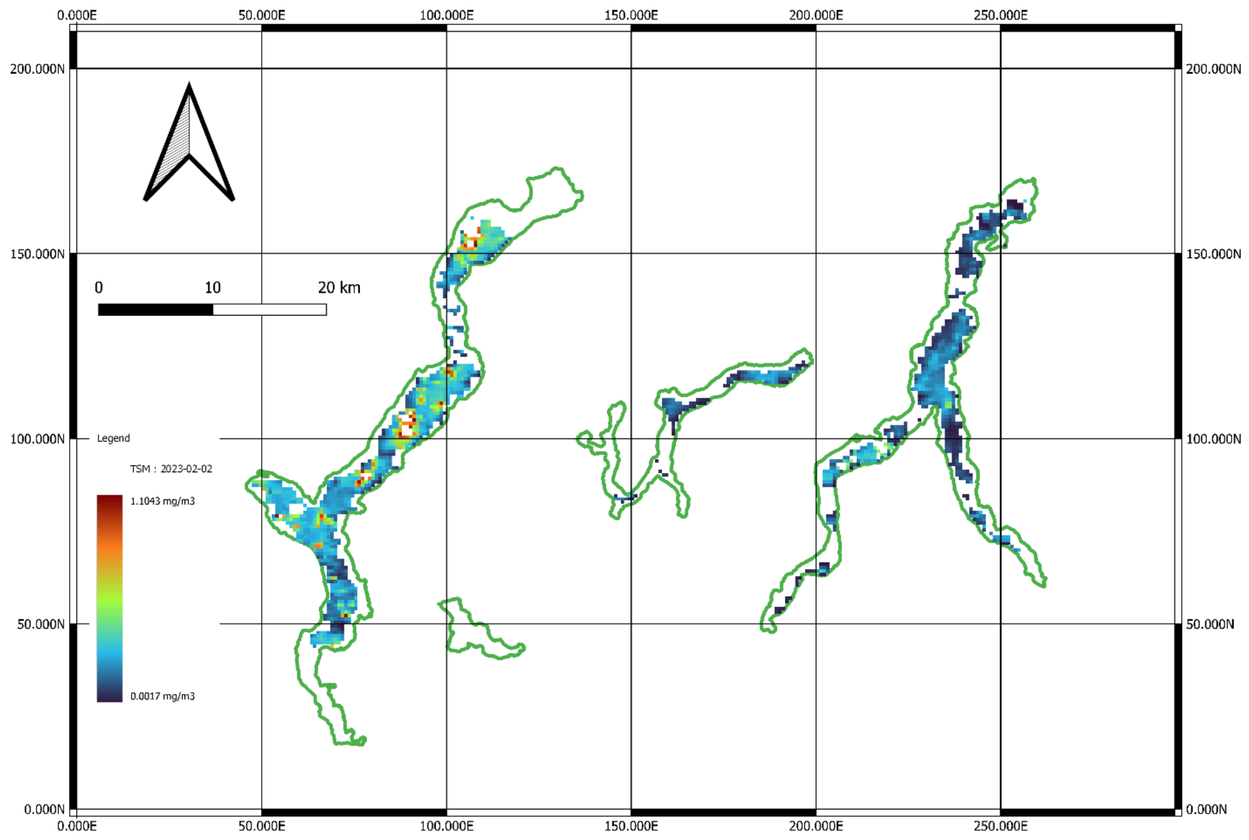


Figure 39. Total Suspended Mater map (02/02/2023)

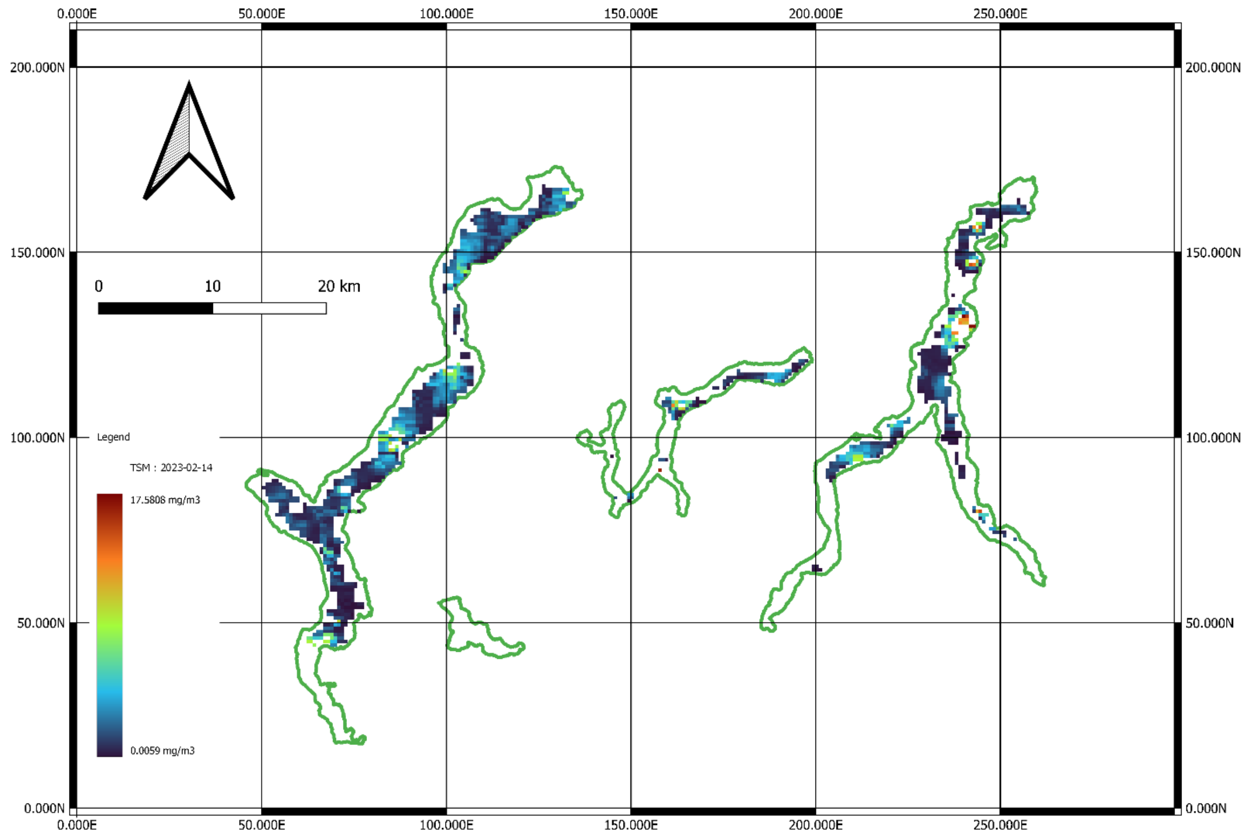


Figure 40. Total Suspended Mater map (10/01/2023)

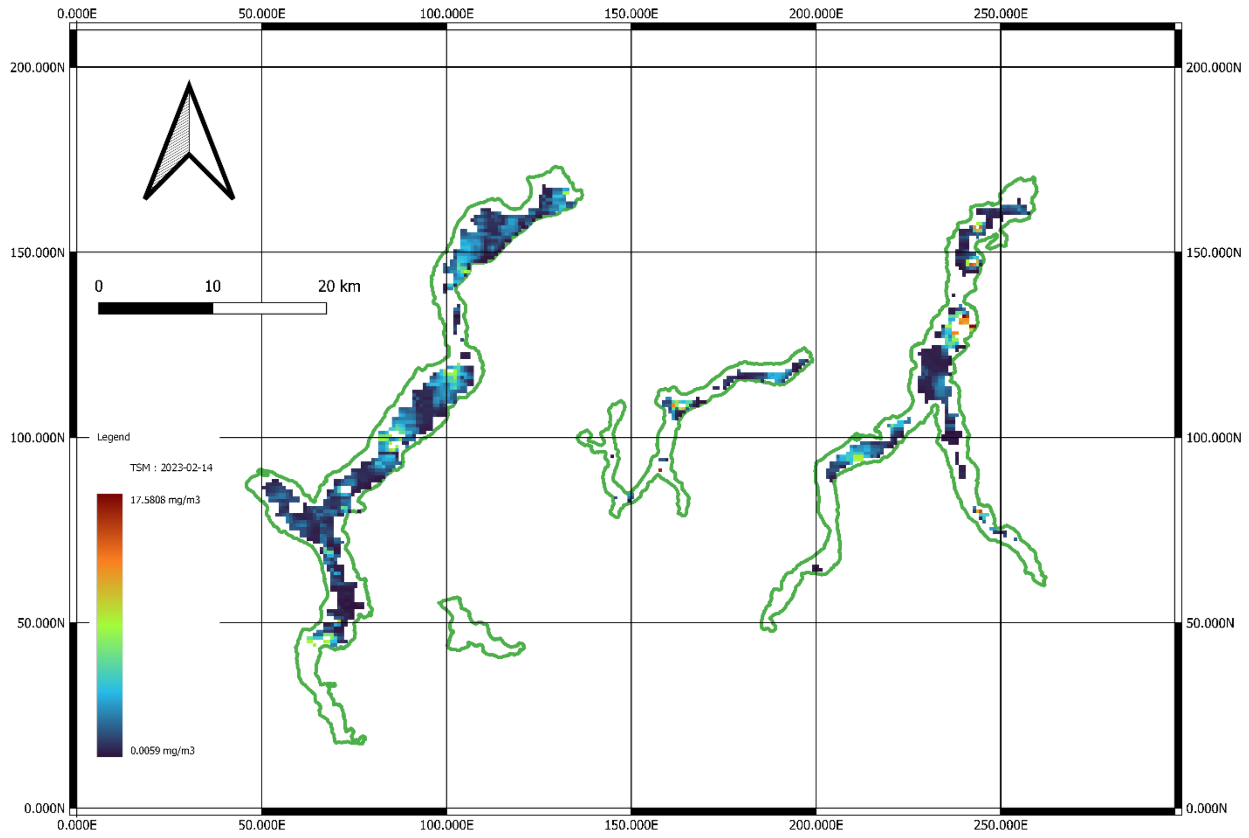


Figure 41. Total Suspended Mater map (14/02/2023)

5.2 Statistics

The analysis of outlier rejection for Chl-a and TSM, as depicted in Figure 42 and Figure 43, respectively, reveals some interesting patterns. Specifically, for Chl-a, the analysis indicates that the highest number of outlier pixels was detected in Lake Maggiore, followed by Lake Como, and finally Lake Lugano. On the other hand, the areas with the least out-of-range values were predominantly found along the lake borders.

Similarly, for TSM, the analysis demonstrates that the greatest number of outlier pixels were identified in Lake Maggiore, followed by Lake Lugano, and finally Lake Como. Moreover, the regions with the fewest out-of-range values were primarily situated along the lake's shorelines.

These findings highlight the significance of Lake Maggiore in terms of containing a larger number of outlier pixels for both Chl-a and TSM measurements. Additionally, they suggest that the lake borders tend to exhibit comparatively fewer out-of-range values, indicating a potential variation in the water quality or composition across different areas of the lakes.

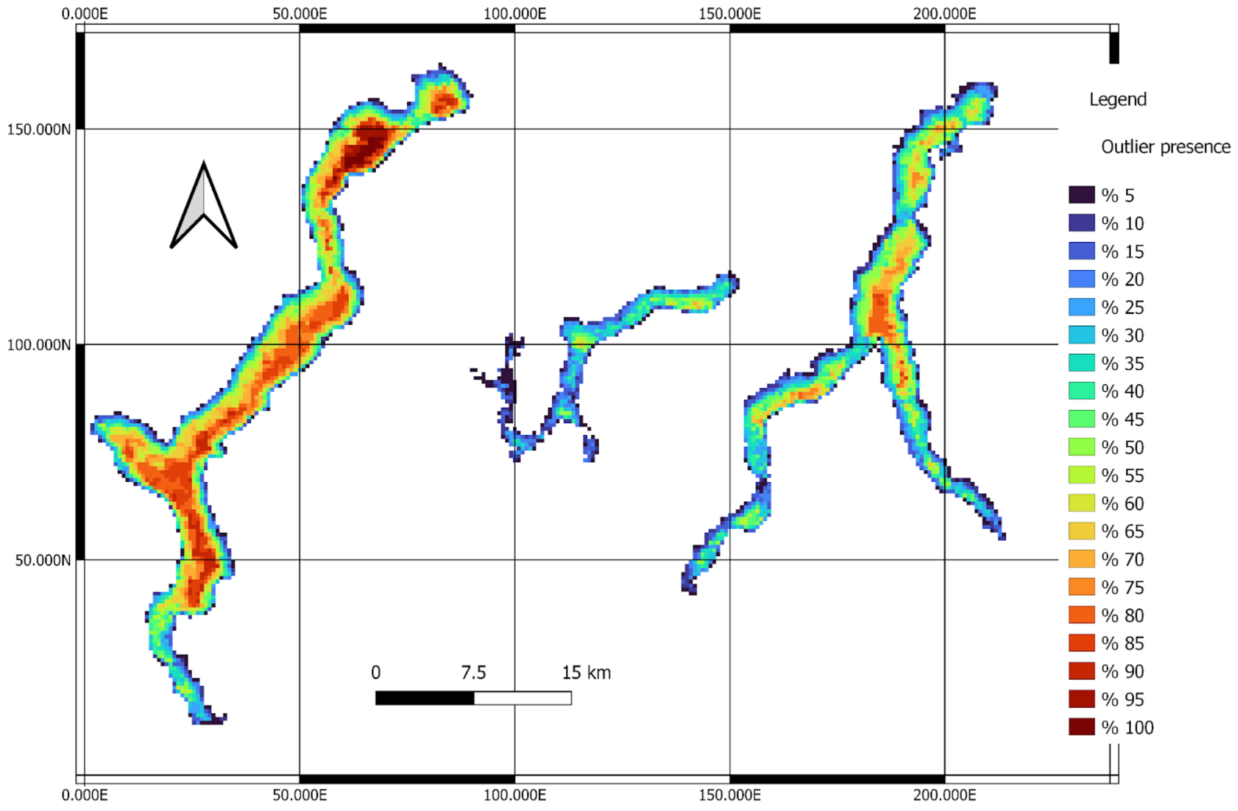


Figure 42. Chl-a outlier presence

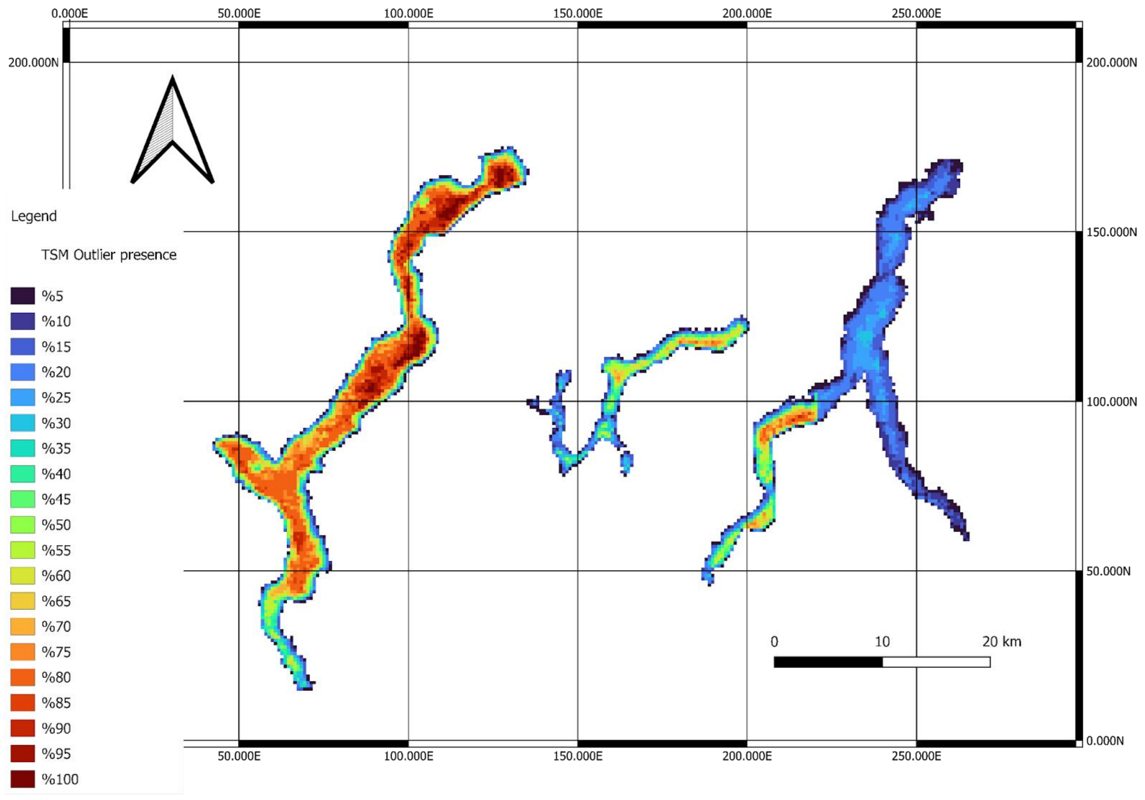
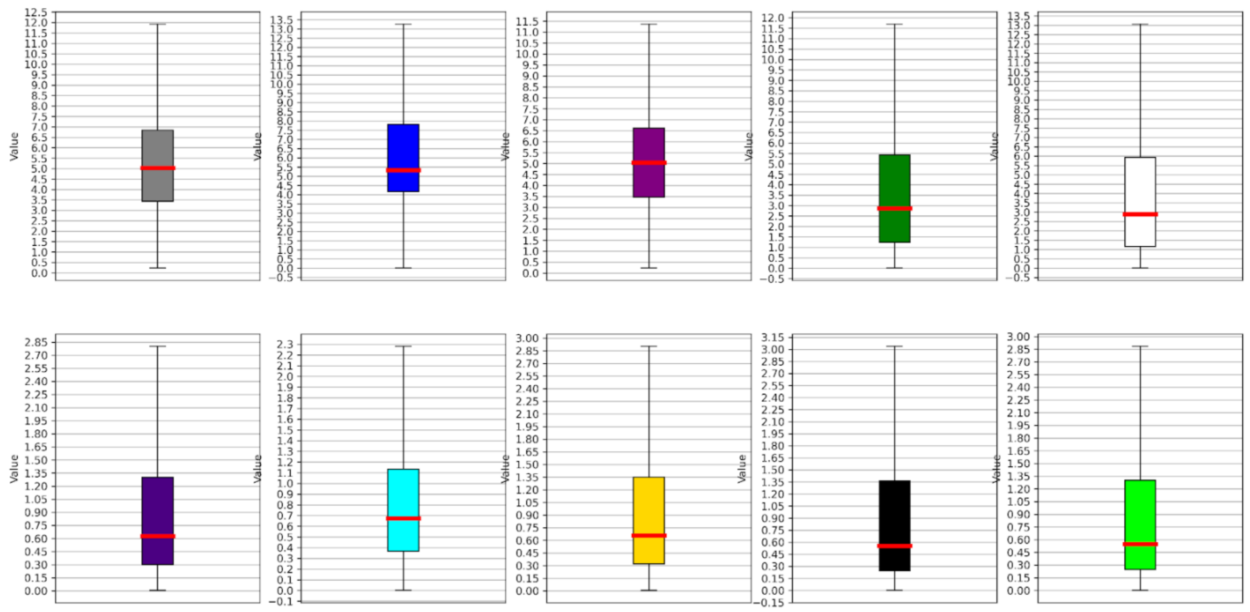


Figure 43. TSM outlier presence

The concentration maps of Chl-a and TSM were generated by applying adjustments to the processing parameters of the C2RCC (3.2.1) algorithm, specifically focusing on temperature values. Three distinct sets of concentration maps were computed to examine the impact of temperature variations. The first set utilized the original temperature data obtained from the Tremezzo sensor. In the second set, the temperature values were reduced by 10 degrees compared to the original readings. Conversely, the third set employed temperature values that were increased by 10 degrees relative to the original measurements. To visually represent the influence of temperature on the concentration, not only a boxplot was created as in Figure 38 but also Table 9 for Chl-a and Table 10 for TSM.



▨ Chl-a concentration calculated with reduced temperature

🟦 Chl-a concentration calculated with ARPA temperature

🟪 Chl-a concentration calculated with increased temperature

🟩 The difference between Chl-a calculated with the increased temperature and the one calculated with the ARPA temperature

⬜ The difference between Chl-a calculated with the increased temperature and the one calculated with the ARPA temperature

🟪 TSM concentration calculated with reduced temperature

🟦 TSM concentration calculated with ARPA temperature

🟨 TSM concentration calculated with increased temperature

⬛ The difference between TSM calculated with the increased temperature and the one calculated with the ARPA temperature

🟩 The difference between TSM calculated with the reduced temperature and the one calculated with the ARPA temperature

Figure 44. Map's boxplot

Chl-a Product	Mean	Standard Deviation
Calculated with the original temperature	6.381	3.774
Calculated with reduced temperature	5.757	3.669
Calculated with increased temperature	5.656	4.022
Difference between the calculated with original temperature and the calculated with reduced temperature	3.854	3.402
Difference between the calculated with original temperature and the calculated with increased temperature	4.011	3.614

Table 9. Chl-a statistics

TSM Product	Mean	Standard Deviation
Calculated with the original temperature	1.062	1.389
Calculated with reduced temperature	1.018	1.276
Calculated with increased temperature	1.088	1.431
Difference between the calculated with original temperature and the calculated with reduced temperature	1.083	1.572
Difference between the calculated with original temperature and the calculated with increased temperature	1.0233	1.416

Table 10. TSM statistics

For CHL-a concentrations, we see that the highest mean concentration of 6.381 is observed when the original temperature is used, while the mean concentrations decrease to 5.757 and 5.656 when the temperature is reduced and increased, respectively. This indicates that temperature alone may not be the sole determining factor in influencing concentration levels. Regarding the standard deviation, we observe slight variations between the different temperature scenarios. The standard deviation values for CHL-a concentrations range from 3.774 (original temperature) to 4.022 (increased temperature). These findings suggest that temperature adjustments can have a marginal impact on the variability of CHL-a concentrations, with a slightly higher dispersion observed at increased temperatures.

Moving on to TSM concentrations, similar trends are observed. The mean TSM concentration is highest for the original temperature (1.062) and decreases to 1.018 and 1.088 for the reduced and increased temperatures, respectively. These results indicate that temperature modifications can influence the average TSM concentration, albeit with relatively small changes.

Examining the standard deviation values for TSM concentrations, we find slight variability between the temperature scenarios. The standard deviation ranges from 1.389 (the original temperature) to 1.431 (the increased temperature). This implies that temperature adjustments can introduce minor fluctuations in the dispersion of TSM concentrations.

Overall, these results highlight the influence of temperature on both CHL-a and TSM concentrations. The mean concentration values show slight variations with temperature changes, indicating that temperature can play a role in altering the average concentrations of these parameters. Furthermore, the standard deviation values suggest that temperature modifications can also impact the variability or spread of the concentration values.

Snooping deeper, concentration fluctuations of both parameters (Chl-a and TSM) are in fact influenced by the temperature, as shown in Figure 45 , which illustrates 100 samples of pixel values for each set of the concentration maps.

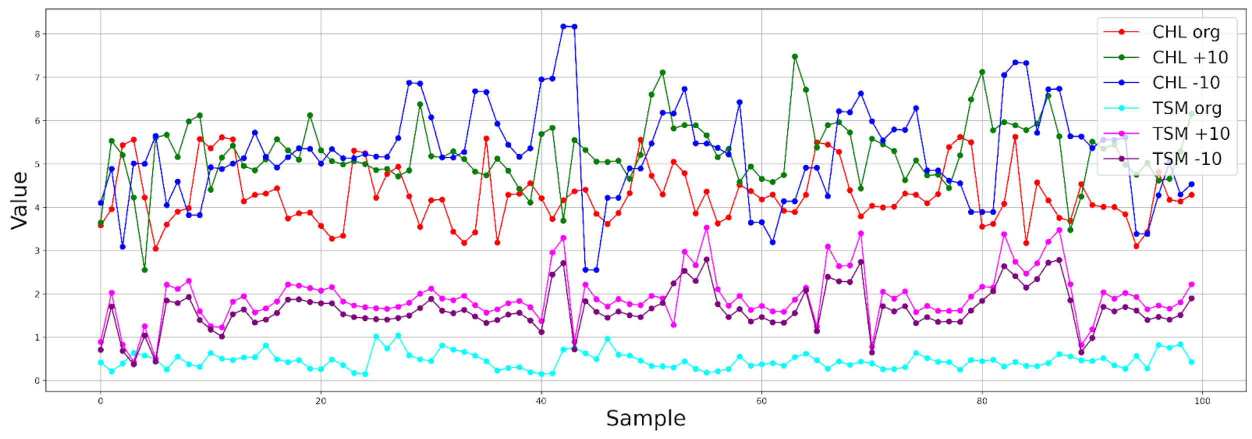


Figure 45. Samples of Chl-a and TSM Parameters

While the provided temperature data by the project companions is typically adopted, incorporating the inseparable temperature (4.2.2) value provided by the application users can offer more flexibility in terms of time. However, this increased flexibility may come at the cost of accuracy.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Monitoring water resources is crucial for understanding their quality and addressing the impacts of climate change. The project's focus was on the monitoring of subalpine lakes and their ecosystems, namely Lugano, Varese, Maggiore, and Como, utilizing remote sensing technology, performed in the framework of the SIMILE Italy-Switzerland Interreg project. The primary objective was to develop a user-friendly monitoring tool with a graphical user interface (GUI) that incorporates remote sensing capabilities. This tool aims to enhance usability and accessibility, making it easier for operators who are not experts in remote sensing to effectively monitor water quality.

In particular, the production of Water Quality Parameter maps incorporates a comprehensive processing chain and outlier rejection method. This intricate process relies on the utilization of imagery data from both the Sentinel-3 OLCI and Landsat 8 TIRS satellites. The map creation leverages the C2RCC processor, which is a specialized tool available within the open-source free software SNAP. In addition to the satellite imagery, local measures provided by SIMILE project partners play a pivotal role as processing parameters. For co-registration purposes, the OLCI data is subjected to precise alignment due to the presence of A/B Sentinel-3 satellite data. This ensures accurate and consistent mapping results. To ensure data quality, outlier rejection is applied to all generated maps. By following the rigorous 3 Sigma rules, any values deemed unacceptable are identified and removed from the final maps. The Water Quality Parameter maps focus on three essential indicators: CHL-a (chlorophyll-a), TSM (total suspended matter), and LWST (Lake water surface temperature). These maps serve as crucial indicators for assessing water quality.

The GUI plays a significant role in streamlining the entire process for users, offering a user-friendly and straightforward approach. To utilize the application, certain key requirements are necessary, including credentials for ARPA and HDA. The GUI greatly enhances user convenience by providing easy access to temperature data downloads. It offers seamless flexibility, allowing users to manually input data or effortlessly access and download Sentinel-3 images. The application's intuitive interface enables users to initiate the desired output generation process with ease. By providing an intuitive interface, the GUI greatly simplifies the overall user experience.

In addition to user interaction with temperature input, a comprehensive statistical analysis was conducted to investigate the relationship between temperature variations and the concentrations of Chl-a and TSM. The findings of the statistical analysis provided significant insights into the temperature variations' influence on Chl-a and TSM concentrations. The results highlighted the significance of temperature as a contributing factor to the variability observed in Chl-a and TSM levels. The analysis demonstrated the presence of a significant relationship between temperature and the concentrations of Chl-a and TSM. In addition, giving the user an alert when the inserted value is used reduces the output accuracy of WQPs.

6.2 Future work

Future work could focus on developing an automated download system for Landsat 8 imagery that not only facilitates the retrieval of the data but also directly places it in the specific folder required by the tool or software being used. By eliminating the need for manual file management, this system would streamline the process and allow for seamless integration of Landsat 8 imagery into the workflow, ensuring efficient utilization of the data without any additional manual steps.

Bibliography

[1] URL USGS, "U.S.Geological Survey, Water science school https://www.usgs.gov/special-topics/water-science-school/science/how-much-water-there-earth."

[2] URL NASA, "NASA, Astrobiology, and Life in the Universe. https://astrobiology.nasa.gov/education/alp/water-so-important-for-life/."

[3] K. A. Sharp, "Water: Structure and Properties," in *eLS*, Wiley, 2001. doi: 10.1038/npg.els.0003116.

[4] URL NASA "National Aeronautics and Space Administration", "NASA https://www.nasa.gov/specials/ocean-worlds/."

[5] UNDP, "United Nations Development Program https://www.undp.org/sustainable-development-goals."

[6] Lorna Fewtrell and Jamie Bartram, *Water quality: guidelines, standards, and health: assessment of risk and risk management for water-related infectious disease*,"

[7] T. C. Winter, J. W. (Judson W.), O. L. Franke, W. M. Alley, and Geological Survey (U.S.), *Groundwater and surface water: a single resource* U.S. Geological Survey, 1998

[8] S. Madhav *et al.*, "Water Pollutants: Sources and Impact on the Environment and Human Health," 2020, pp. 43–62. doi: 10.1007/978-981-15-0671-0_4.

[9] EPA, "United States Environmental Protection Agency. (2021). Water Monitoring and Assessment https://www.epa.gov/awma."

[10] NOAA, "National Oceanic and Atmospheric Administration, 2021 https://www.noaa.gov/."

[11] J. C. Loftis, G. B. Mcbride, and J. C. Ellis, "Considerations of Scale in Water Quality Monitoring and Data Analysis," 1991

[12] Gao, YYang, T., and Jin, J.: Nanoparticle pollution and associated increasing potential risks to the environment and human health: a case study of China. Environ. Sci. Pollut. Res. 22(23), 19297–19306 (2015)

[13] 13. Sweet, L., Strohm, B.: Nanotechnology—life-cycle risk management. Hum. Ecol. Risk Assess. 12(3), 528–551 (2006)

[14] Klaine, S.J., Alvarez, P.J.J., Batley, G.E., Fernandes, T.F., Handy, R.D., Lyon, D.Y., et al.: Nanomaterials in the environment: behavior, fate, bioavailability, and effects. Environ Toxicol. Chem. 27(9), 1825–1851 (2008)

[15]] Das, R., Shahnavaz, Z., Ali, M.E., Islam, M.M., Hamid, S.B.A.: Can we optimize arc discharge and laser ablation for well-controlled carbon nanotube synthesis? Nanoscale Res. Lett. 11(1), 510 (2016)

[16] Zhang, X., Guo, W., Ngo, H.H., Wen, H., Li, N., Wu, W.: Performance evaluation of powdered activated carbon for removing 28 types of antibiotics from water. J. Environ. Manage. 172, 193–200 (2016)

[17] Asfaram, A., Ghaedi, M., Hajati, S., Rezaeinejad, M., Goudarzi, A., Purkait, M.K.: Rapid removal of Auramine-O and Methylene blue by ZnS: Cu nanoparticles loaded on activated carbon: a response surface methodology approach. J. Taiwan Inst. Chem. Eng. 53, 80–91 (2015)

[18] Abdelbassit, M.S.A., Alhooshani, K.R., Saleh, T.A.: Silica nanoparticles loaded on activated carbon for simultaneous removal of dichloromethane, trichloromethane, and carbon tetrachloride. Adv. Powder Technol. 27(4), 1719–1729 (2016)

[19] Das, R., Ali, M.E., Hamid, S.B.A., Annuar, M., Ramakrishna, S.: Common wet chemical agents for purifying multiwalled carbon nanotubes. J. Nanomater. 2014, 237 (2014)

[20] Das, R.: Carbon nanotube purification. In: Nanohybrid Catalyst based on Carbon Nanotube, pp. 55–73. Springer (2017)

[21] Xu, G.-R., Xu, J.-M., Su, H.-C., Liu, X.-Y., Zhao, H.-L., Feng, H.-J., et al.: Two-dimensional (2D) nanoporous membranes with sub-nanopores in reverse osmosis desalination: latest developments and future directions. Desalination (2017)

[22] Bahena, J.L.R., Cabrera, A.R., Valdivieso, A.L., Urbina, R.H.: Fluoride adsorption onto a-Al2O3 and its effect on the zeta potential at the alumina-aqueous electrolyte interface. Sep. Sci. Technol. 37(8), 1973–1987 (2002)

[23] Waters, T. F. (1995). Sediment in streams: sources, biological effects, and control. American Fisheries Society.

[24] Bilotta, G. S., & Brazier, R. E. (2008). Understanding the influence of suspended solids on water quality and aquatic biota Water Research, 42(12), 2849–2861.

[25] Clark, J. R. (1969). Thermal pollution and aquatic life. Scientific American, 220(3), 18–27.

[26] Davidson, B., & Bradshaw, R. W. (1967). Thermal pollution of water systems Environmental Science and Technology, 1(8), 618–630

[27] Fenocchi, A., Rogora, M., Sibilla S., Ciampittiello M., and Dresti C. (2018: Forecasting the evolution in the mixing regime of a deep subalpine lake under climate change scenarios through numerical modeling (Lake Maggiore, Northern Italy/Southern Switzerland). Climate Dynamics 51: 3521–3536.

[28]] Salmaso, N., and Mosello, R. (2010: Limnological research in the deep southern subalpine lakes: synthesis, directions, and perspectives. Advances in Oceanography and Limnology 1: 29–66

[29] Brovelli, M. A., Cannata, M., and Rogora, M., 2019. SIMILE, a geospatial enabler of the monitoring of sustainable development goal 6 (Ensure availability and sustainability of water for all)

[30] Banas, D., Grillas, P., Auby, I., Lescuyer, F., Coulet, E., Moreteau, J.-C. and Millet, B., 2005. Short time scale changes in underwater irradiance in a wind-exposed lagoon (Vaccares La- ` goon, France): efficiency of infrequent field measurements of water turbidity or weather data to predict irradiance in the water column. Hydrobiologia 551(1), pp. 3–16.

[31] Garel, E., Nunes, S., Neto, J. M., Fernandes, R., Neves, R., Marques, J. C. and Ferreira, 2009. The autonomous Simpatico system for real-time continuous water-quality and current velocity monitoring: examples of application in three Portuguese estuaries. Geo-Marine Letters, 29(5), pp. 331–341.

[32] Toro Herrera, J. F., Carrion, D., and Brovelli, M. A., 2021. A collaborative platform for water quality monitoring: SIMILE webGIS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIII-B4-2021, pp. 201–207

[33] Vavassori, A., Biraghi, C. A., Bratic, G., Carrion, D., Zamboni, G., and Brovelli, M. A., 2021. Citizen science tools for lake monitoring in the framework of the United Nations Sustainable Development Goals: The Project SIMILE, GI Forum 1, pp. 179–186.

[34] Negash, S., Gray, P., 2008: Business intelligence. In Handbook on Decision Support systems 2. Springer, Berlin, Heidelberg: pp. 175-193.

[35] Topp, S. N., Pavelsky, T. M., Jensen, D., Simard, M., & Ross, M. R. V. (2020). Research Trends in the Use of Remote Sensing for Inland Water Quality Science: Moving Towards Multidisciplinary Applications. Water, 12(1). https://doi.org/10.3390/w12010169

[36] Giardino, C., Bresciani, M., Stroppiana, D., Oggioni, A., & Morabito, G. (2014). Optical remote sensing of lakes: An overview on Lake Maggiore. Journal of limnology, 73, 210–214. https://doi.org/10.4081/jlimnol.2014.817

[37] Bresciani, M., Pinardi, M., Free, G., Luciani, G., Ghebrehiwot, S., Laanen, M., Peters, S., Della Bella, V., Padula, R., & Giardino, C. (2020). The Use of Multisource Optical Sensors to Study Phytoplankton Spatio-Temporal Variation in a Shallow Turbid Lake. Water, 12(1), 284. https://doi.org/10.3390/w12010284

[38] Hestir, E. L., Brando, V. E., Bresciani, M., Giardino, C., Matta, E., Villa, P., & Dekker, A. G. (2015). Measuring freshwater aquatic ecosystems: The need for a hyperspectral global mapping satellite mission. Remote Sensing of Environment, 167, 181–195. https://doi.org/10.1016/j.rse.2015.05.023

[39] Overstreet, B. T., & Legleiter, C. J. (2017). Removing sun glint from optical remote sensing images of shallow rivers. Earth Surface Processes and Landforms, 42(2), 318–333. https://doi.org/10.1002/esp.4063

[40] De Keukelaere, L., Sterckx, S., Adriaensen, S., Knaeps, E., Reusen, I., Giardino, C., Bresciani, M., Hunter, P., Neil, C., Van der Zande, D., & Vaiciute, D. (2018). Atmospheric correction of Landsat-8/OLI and Sentinel-2/MSI data using iCOR algorithm: Validation for coastal and inland waters. European Journal of Remote Sensing, 51(1), 525–542.

[41] Warren, M. A., Simis, S. G. H., Martinez-Vicente, V., Poser, K., Bresciani, M., Alikas, K., Spyrakos, E., Giardino, C., & Ansper, A. (2019). Assessment of atmospheric correction algorithms for the Sentinel-2A MultiSpectral Imager over coastal and inland waters. Remote Sensing of Environment, 225, 267–289. https://doi.org/10.1016/j.rse.2019.03.018

[42] Warren, M. A., Simis, S. G. H., & Selmes, N. (2021). Complementary water quality observations from high and medium resolution Sentinel sensors by aligning chlorophyll-a and turbidity algorithms. Remote Sensing of Environment, 265, 112651. https://doi.org/10.1016/j.rse.2021.112651

[43] Soriano-González, J., Urrego, E. P., Sòria-Perpinyà, X., Angelats, E., Alcaraz, C., Delegido, J., Ruíz-Verdú, A., Tenjo, C., Vicente, E., & Moreno, J. (2022). Towards the Combination of C2RCC Processors for Improving Water Quality Retrieval in Inland and Coastal Areas. Remote Sensing, 14(5). https://doi.org/10.3390/rs14051124

[44] Spyrakos, E., O'Donnell, R., Hunter, P. D., Miller, C., Scott, M., Simis, S. G. H., Neil, C., Barbosa, C. C. F., Binding, C. E., Bradt, S., Bresciani, M., Dall'Olmo, G., Giardino, C., Gitelson, A. A., Kutser, T., Li, L., Matsushita, B., Martinez-Vicente, V., Matthews, M. W., … Tyler, A. N. (2018). Optical types of inland and coastal waters. Limnology and Oceanography, 63(2), 846– 870. https://doi.org/10.1002/lno.10674

[45] Niroumand-Jadidi, M., Bovolo, F., Bruzzone, L., & Gege, P. (2021). Inter-Comparison of Methods for Chlorophyll-a Retrieval: Sentinel-2 Time-Series Analysis in Italian Lakes. Remote Sensing, 13(12). https://doi.org/10.3390/rs13122381

[46] Luciani, G., Bresciani, M., Biraghi, C.A., Ghirardi, N., Carrion, D., Rogora, M., Brovelli, M.A., 2021. Satellite Monitoring system of Subalpine lakes with open source software: The case of SIMILE project. Baltic J. Modern Computing, 9(1), 135-144

[47] Free, G., Bresciani, M., Pinardi, M., Ghirardi, N., Luciani, G., Caroni, R., & Giardino, C. (2021). Detecting Climate Driven Changes in Chlorophyll-a in Deep Subalpine Lakes Using Long Term Satellite Data. Water, 13(6), 866. doi.org/10.3390/w13060866

[48] Salmaso, N., & Mosello, R. (2010). Limnological research in the deep southern subalpine lakes: Synthesis, directions and perspectives. Advances in Oceanography and Limnology, 1(1), 29–66. https://doi.org/10.4081/aiol.2010.5294

[49] Zaccara, S.; Canziani, A.; Roella, V.; Crosa, G. A northern Italian shallow lake as a case study for eutrophication control. *Limnology* **2007**, *8*, 155–160. [**Google Scholar**] [**CrossRef**]

[50] Chirico, N., António, D. C., Pozzoli, L., Marinov, D., Malagó, A., Sanseverino, I., Beghi, A., Genoni, P., Dobricic, S., & Lettieri, T. (2020). Cyanobacterial Blooms in Lake Varese: Analysis and Characterization over Ten Years of Observations. *Water*, *12*(3), 675. https://doi.org/10.3390/w12030675

[51] URL https://www.britannica.com/place/Como-Italy. (n.d.). *britannica Lake Como*.

[52] Lepori, F., & Roberts, J. J. (2015). Past and future warming of a deep European lake (Lake Lugano): What are the climatic drivers? *Journal of Great Lakes Research*, *41*(4), 973–981. https://doi.org/10.1016/J.JGLR.2015.08.004

[53] H. Saidi, M. Ciampittiello, C. Dresti, G. Ghiglieri ,The climatic characteristics of extreme precipitations for short-term intervals in the watershed of Lake MaggioreTheor. Appl. Climatol., 113 (1) (2013), pp. 1-15, 10.1007/s00704-012-0768-x

[54] URL https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-3/overview.(n.d.). copernicus.

[55] SIMILE (Sistema Informativo per il Monitoraggio Integrato dei Laghi insubrici e dei loro Ecosistemi). (n.d.). https://atmcorr.gsfc.nasa.gov/

[56] URL https://landsat.gsfc.nasa.gov/satellites/landsat-8/

[57] URL https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-3-olci/resolutions/radiometric

[58] Doerffer, R., & Schiller, H. (2007). The MERIS case 2 water algorithm. International Journal of Remote Sensing,28(3), 517−535

[59] Doerffer R. and H. Schiller (2008): MERIS Regional Coastal and Lake Case 2 Water Project - Atmospheric Correction ATBD, GKSS Research Center 21502 Geesthacht Version 1.0 18. May 2008

[60] IOCCG (2000). Remote Sensing of Ocean Colour in Coastal, and Other Optically Complex Waters. Sathyendranath, S. (ed.), Reports of the International Ocean-Colour Coordinating Group, No. 3, IOCCG, Dartmouth, Canada

[61] Werdell, P.J. and S.W. Bailey (2005): An improved bio-optical data set for ocean color algorithm development and satellite data product validation. Remote Sensing of Environment, 98(1), 122-140.

[62] Brockmann, C.; Doerffer, R.; Peters, M.; Stelzer, K.; Embacher, S.; Ruescas, A. Evolution of the C2RCC neural network for Sentinel 2 and 3 for the retrieval of ocean colour products in normal and extreme optically complex waters. In

Proceedings of the Living Planet Symposium 2016, Prague, Czech Republic, 9–13 May 2016; pp. 9 13.

[63] Kyryliuk, D., & Kratzer, S. (2019). Evaluation of Sentinel-3A OLCI Products Derived Using the Case-2 Regional CoastColour Processor over the Baltic Sea. Sensors, 19(16). https://doi.org/10.3390/s19163609

[64] Luciani, G., Bresciani, M., Biraghi, C. A., Ghirardi, N., Carrion, D., Rogora, M., & Brovelli, M. A. (2021). Satellite Monitoring system of Subalpine lakes with open-source software: The case of SIMILE Project. Baltic Journal of Modern Computing, 9(1). https://doi.org/10.22364/bjmc.2021.9.1.08

[65] Toming, K., Kutser, T., Uiboupin, R., Arikas, A., Vahter, K., & Paavel, B. (2017). Mapping Water Quality Parameters with Sentinel-3 Ocean and Land Colour Instrument imagery in the Baltic Sea. Remote Sensing, 9(10). https://doi.org/10.3390/rs9101070

[66] Soriano-González, J., Urrego, E. P., Sòria-Perpinyà, X., Angelats, E., Alcaraz, C., Delegido, J., Ruíz-Verdú, A., Tenjo, C., Vicente, E., & Moreno, J. (2022). Towards the Combination of C2RCC Processors for Improving Water Quality Retrieval in Inland and Coastal Areas. Remote Sensing, 14(5). https://doi.org/10.3390/rs14051124

[67] URL ESA, SNAP https://seadas.gsfc.nasa.gov/help-8.0.0/c2rcc/C2RCC_OLCI_ProcParameters.html. (n.d.). C2RCC

[68] URL https://www.docker.com/products/docker-desktop/

[69] URL https://snap.stanford.edu/snappy/

[70] URL https://scihub.copernicus.eu/

[71] Toro Herrera, J. F., Carrion, D., Bresciani, M., & Bratic, G. (2022). SEMI-AUTOMATED PRODUCTION AND FILTERING OF SATELLITE DERIVED WATER QUALITY PARAMETERS. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *43*(B3-2022), 1019–1026. https://doi.org/10.5194/isprs-archives-XLIII-B3-2022-1019-2022

[72] Barsi, J. A., Schott, J. R., Palluconi, F. D., & Hook, S. J. (2005). Validation of a web-based atmospheric correction tool for single thermal band instruments (J. J. Butler, A c. Di; pag. 58820E). https://doi.org/10.1117/12.619990

[73] https://land.copernicus.eu/global/products/lswt. (n.d.). *Copernicus*.

[74] Bratic, G., Carrion, D., Cannata, M., Rogora, M., Strigaro, D., & Brovelli, M. A. (2022). LAKE WATER QUALITY MONITORING TOOLS. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *43*(B4-2022), 599–606. https://doi.org/10.5194/isprs-archives-XLIII-B4-2022-599-2022

[75] A. Plyer, E. Colin-Koeniguer and F. Weissgerber, "A New Coregistration Algorithm for Recent Applications on Urban SAR Images," in IEEE Geoscience and Remote Sensing Letters, vol. 12, no. 11, pp. 2198-2202, Nov. 2015, doi: 10.1109/LGRS.2015.2455071.

[76] Plyer, A., Colin-Koeniguer, E., & Weissgerber, F. (2015). A New Coregistration Algorithm for Recent Applications on Urban SAR Images. *IEEE Geoscience and Remote Sensing Letters*, *12*(11), 2198–2202. https://doi.org/10.1109/LGRS.2015.2455071

[77] Z. Li and J. Bethel, "Image coregistration in SAR interferometry," in Proc. Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci., 2008, pp. 433–438.

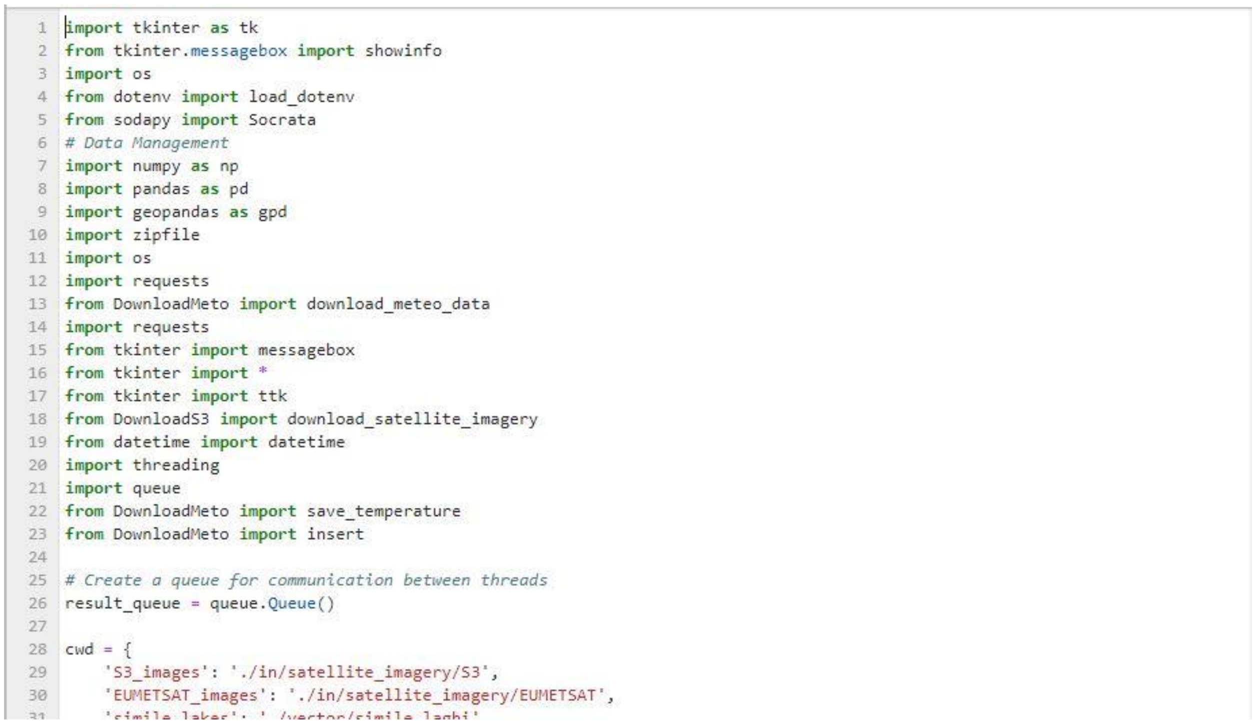
[78] Brovelli, M. A., Cannata, M., & Rogora, M. (2019). Simile, a geospatial enabler of the monitoring of sustainable development goal 6 (ensure availability and sustainability of water for all). *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(4/W20), 3–10. https://doi.org/10.5194/isprs-archives-XLII-4-W20-3-2019

[79] Chirico, N.; António, D.C.; Pozzoli, L.; Marinov, D.; Malagó, A.; Sanseverino, I.; Beghi, A.; Genoni, P.; Dobricic, S.; Lettieri, T. Cyanobacterial Blooms in Lake Varese: Analysis and Characterization over Ten Years of Observations. *Water* 2020, *12*, 675. https://doi.org/10.3390/w12030675

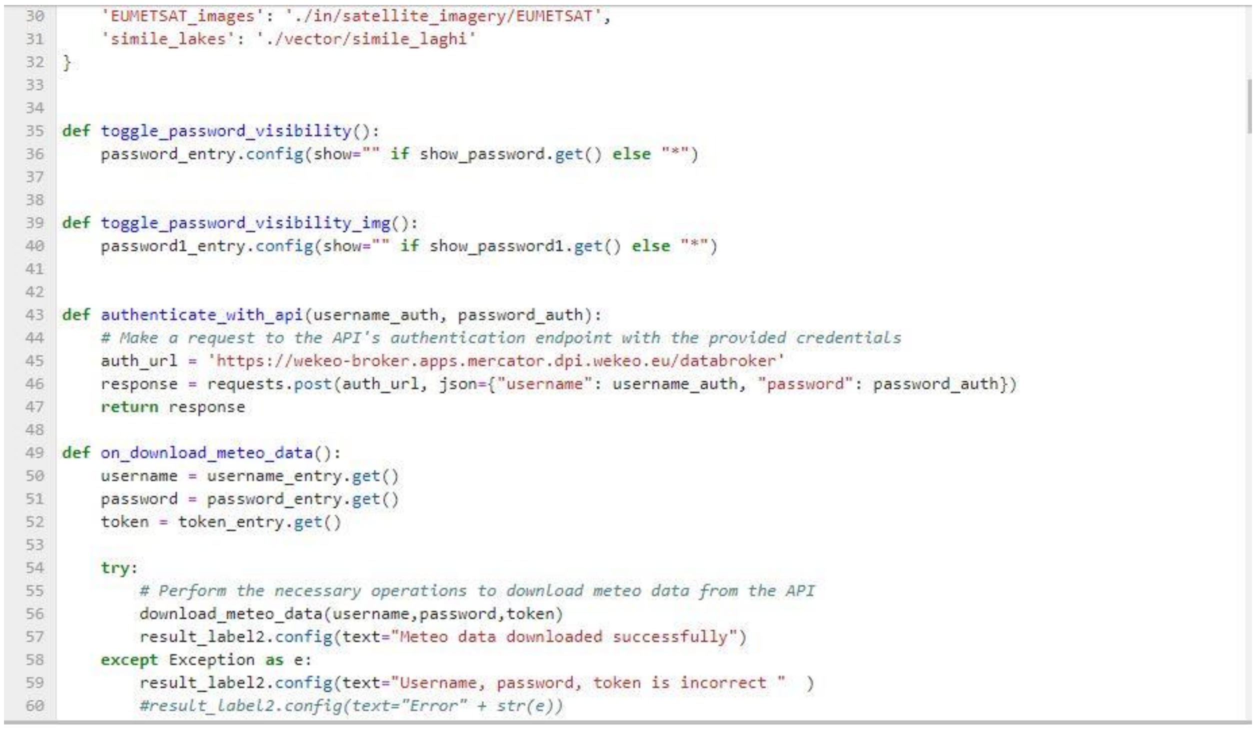
Appendix

Code

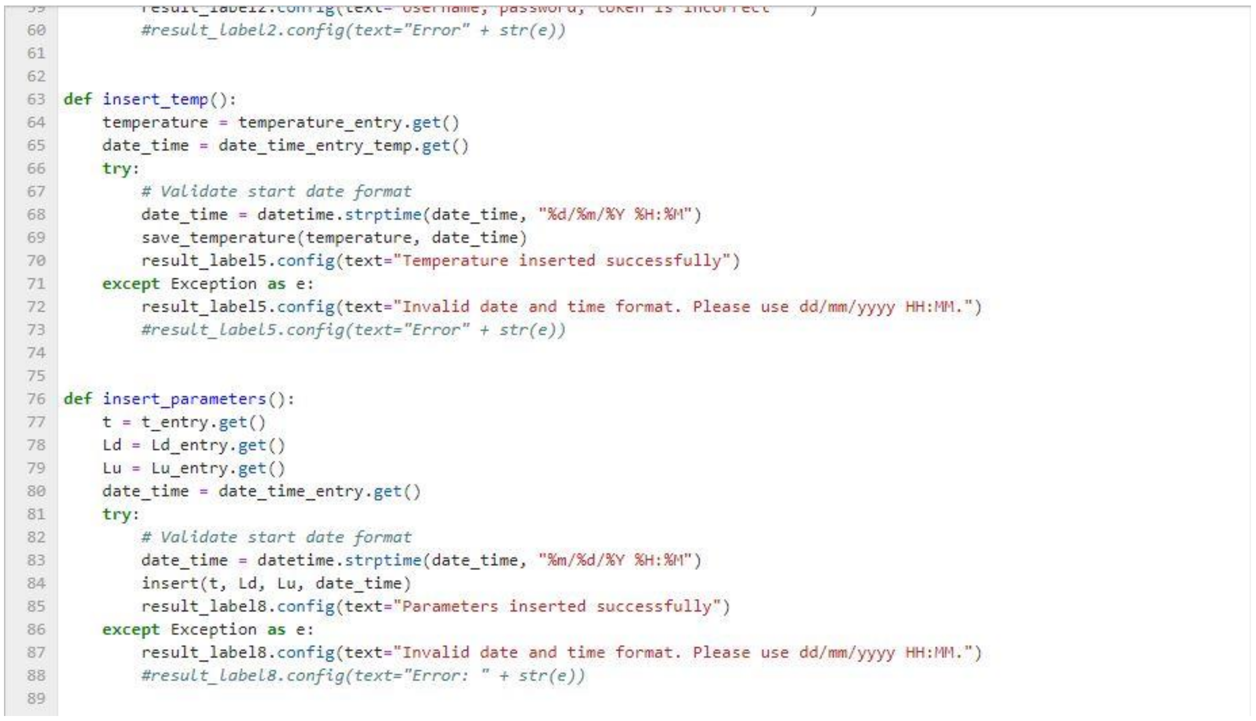
The Graphical User Interface (GUI)



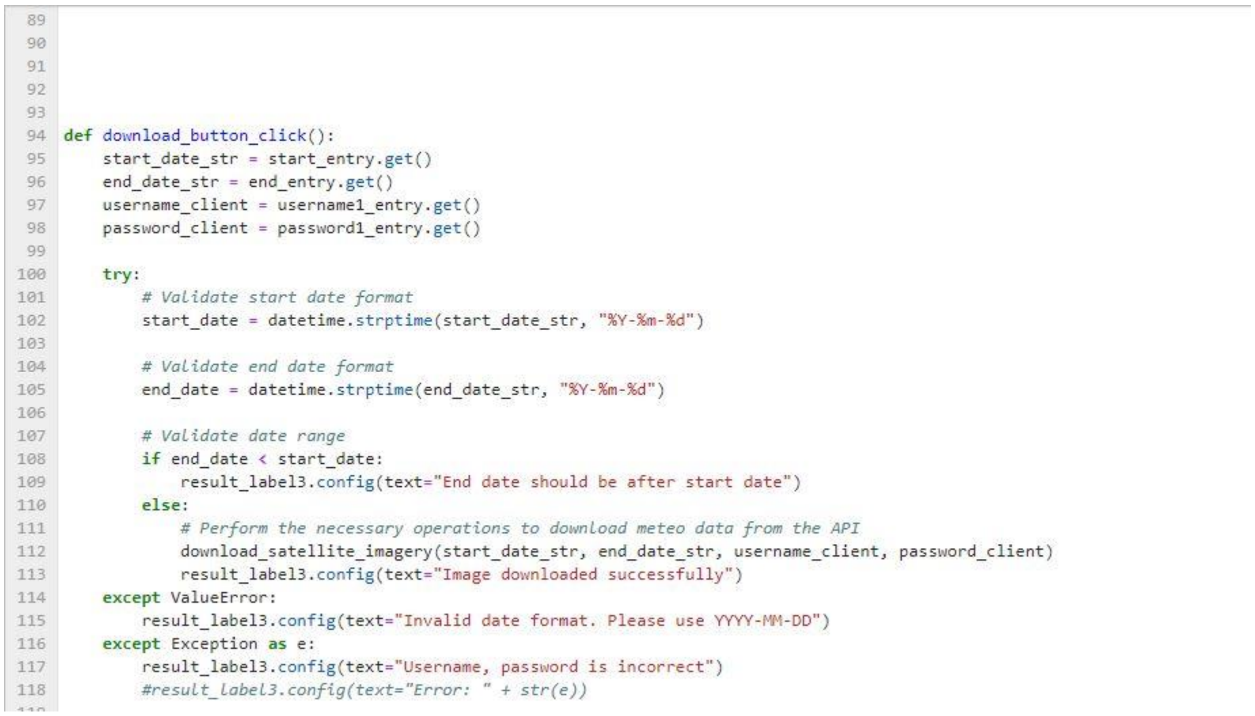
```python
import tkinter as tk
from tkinter.messagebox import showinfo
import os
from dotenv import load_dotenv
from sodapy import Socrata
# Data Management
import numpy as np
import pandas as pd
import geopandas as gpd
import zipfile
import os
import requests
from DownloadMeto import download_meteo_data
import requests
from tkinter import messagebox
from tkinter import *
from tkinter import ttk
from DownloadS3 import download_satellite_imagery
from datetime import datetime
import threading
import queue
from DownloadMeto import save_temperature
from DownloadMeto import insert

# Create a queue for communication between threads
result_queue = queue.Queue()

cwd = {
    'S3_images': './in/satellite_imagery/S3',
    'EUMETSAT_images': './in/satellite_imagery/EUMETSAT',
    'simile_lakes': './vector/simile_laghi'
```



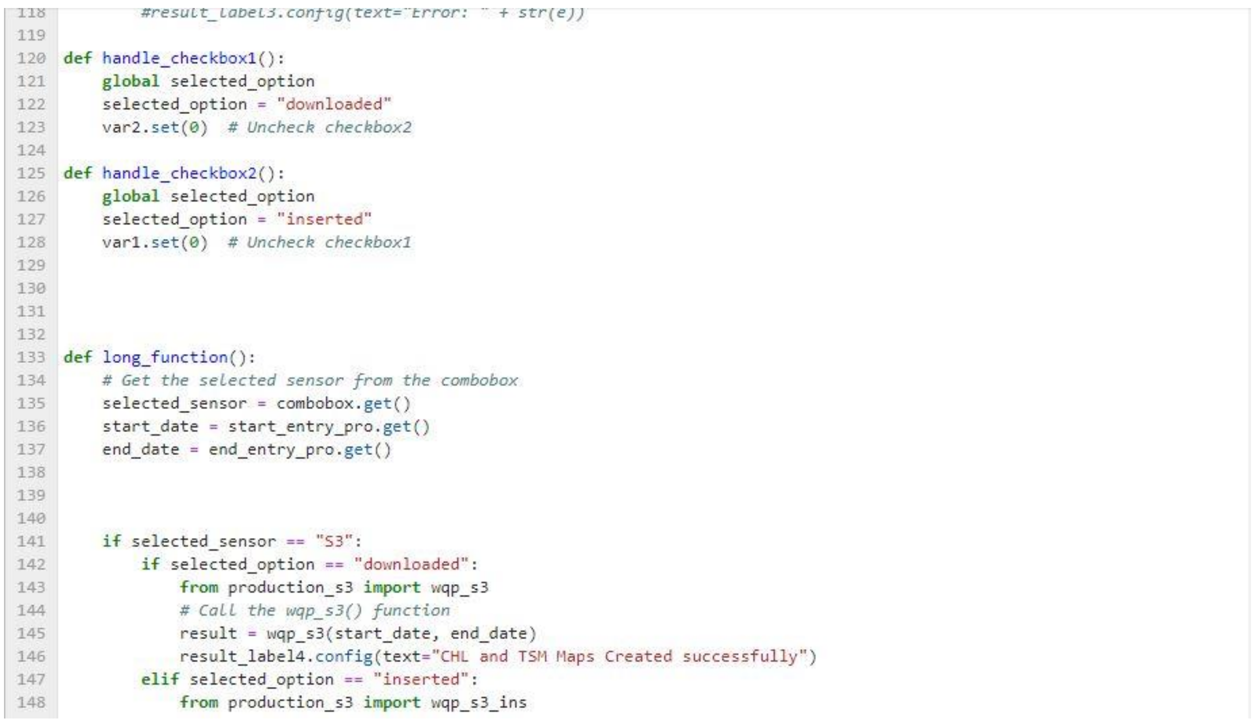
```python
    'EUMETSAT_images': './in/satellite_imagery/EUMETSAT',
    'simile_lakes': './vector/simile_laghi'
}


def toggle_password_visibility():
    password_entry.config(show="" if show_password.get() else "*")


def toggle_password_visibility_img():
    password1_entry.config(show="" if show_password1.get() else "*")


def authenticate_with_api(username_auth, password_auth):
    # Make a request to the API's authentication endpoint with the provided credentials
    auth_url = 'https://wekeo-broker.apps.mercator.dpi.wekeo.eu/databroker'
    response = requests.post(auth_url, json={"username": username_auth, "password": password_auth})
    return response

def on_download_meteo_data():
    username = username_entry.get()
    password = password_entry.get()
    token = token_entry.get()

    try:
        # Perform the necessary operations to download meteo data from the API
        download_meteo_data(username,password,token)
        result_label2.config(text="Meteo data downloaded successfully")
    except Exception as e:
        result_label2.config(text="Username, password, token is incorrect "  )
        #result_label2.config(text="Error" + str(e))
```



```python
        #result_label2.config(text="Error" + str(e))


def insert_temp():
    temperature = temperature_entry.get()
    date_time = date_time_entry_temp.get()
    try:
        # Validate start date format
        date_time = datetime.strptime(date_time, "%d/%m/%Y %H:%M")
        save_temperature(temperature, date_time)
        result_label5.config(text="Temperature inserted successfully")
    except Exception as e:
        result_label5.config(text="Invalid date and time format. Please use dd/mm/yyyy HH:MM.")
        #result_label5.config(text="Error" + str(e))


def insert_parameters():
    t = t_entry.get()
    Ld = Ld_entry.get()
    Lu = Lu_entry.get()
    date_time = date_time_entry.get()
    try:
        # Validate start date format
        date_time = datetime.strptime(date_time, "%m/%d/%Y %H:%M")
        insert(t, Ld, Lu, date_time)
        result_label8.config(text="Parameters inserted successfully")
    except Exception as e:
        result_label8.config(text="Invalid date and time format. Please use dd/mm/yyyy HH:MM.")
        #result_label8.config(text="Error: " + str(e))
```

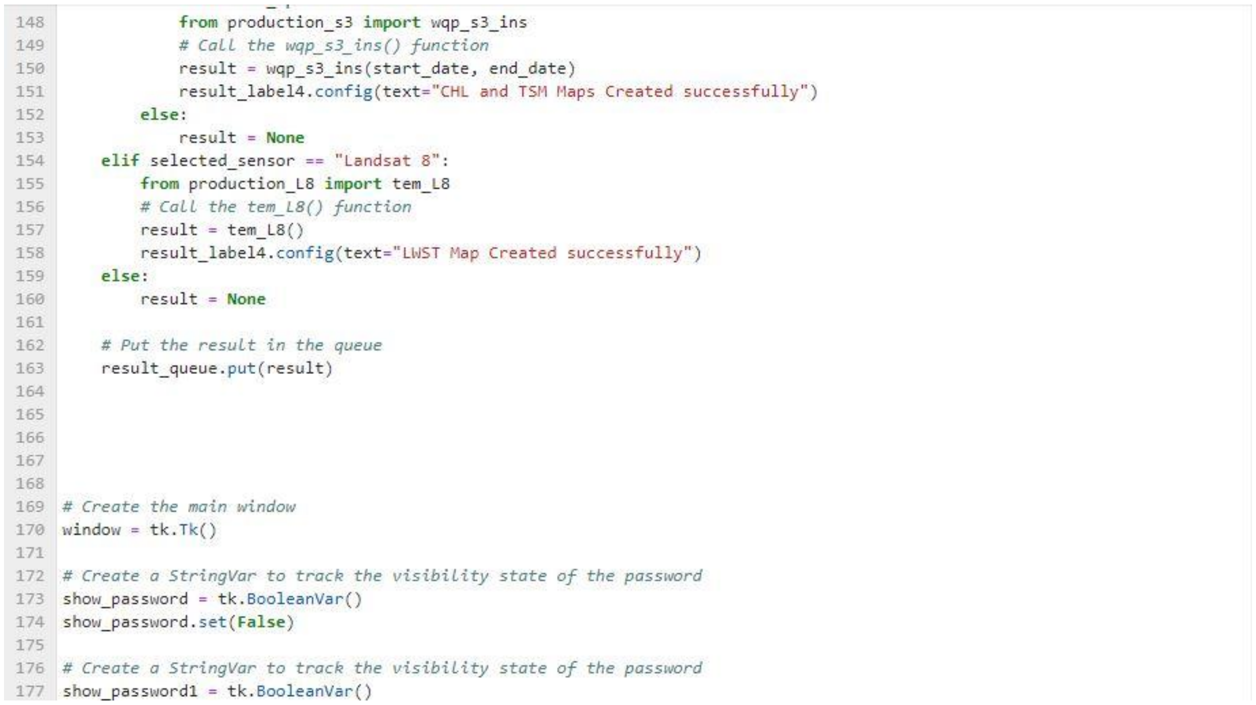


```python





def download_button_click():
    start_date_str = start_entry.get()
    end_date_str = end_entry.get()
    username_client = username1_entry.get()
    password_client = password1_entry.get()

    try:
        # Validate start date format
        start_date = datetime.strptime(start_date_str, "%Y-%m-%d")

        # Validate end date format
        end_date = datetime.strptime(end_date_str, "%Y-%m-%d")

        # Validate date range
        if end_date < start_date:
            result_label3.config(text="End date should be after start date")
        else:
            # Perform the necessary operations to download meteo data from the API
            download_satellite_imagery(start_date_str, end_date_str, username_client, password_client)
            result_label3.config(text="Image downloaded successfully")
    except ValueError:
        result_label3.config(text="Invalid date format. Please use YYYY-MM-DD")
    except Exception as e:
        result_label3.config(text="Username, password is incorrect")
        #result_label3.config(text="Error: " + str(e))
```

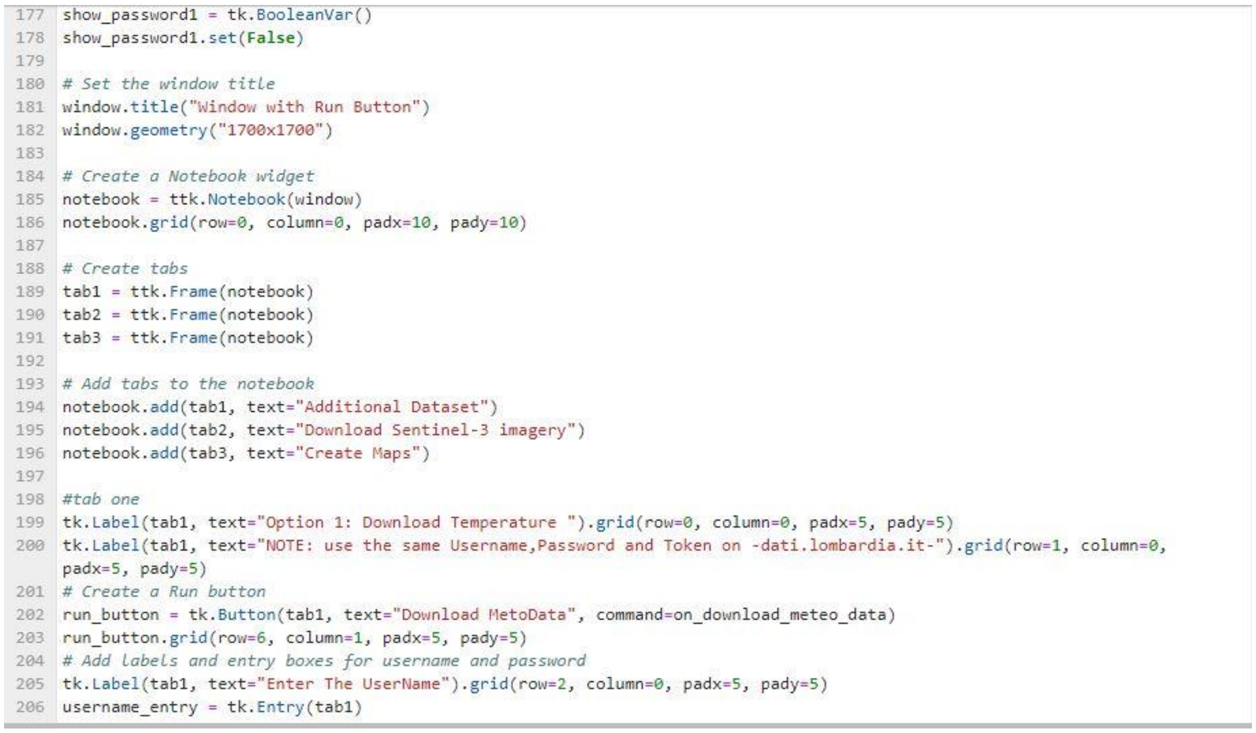


```python
        #result_label3.config(text="Error: " + str(e))
```

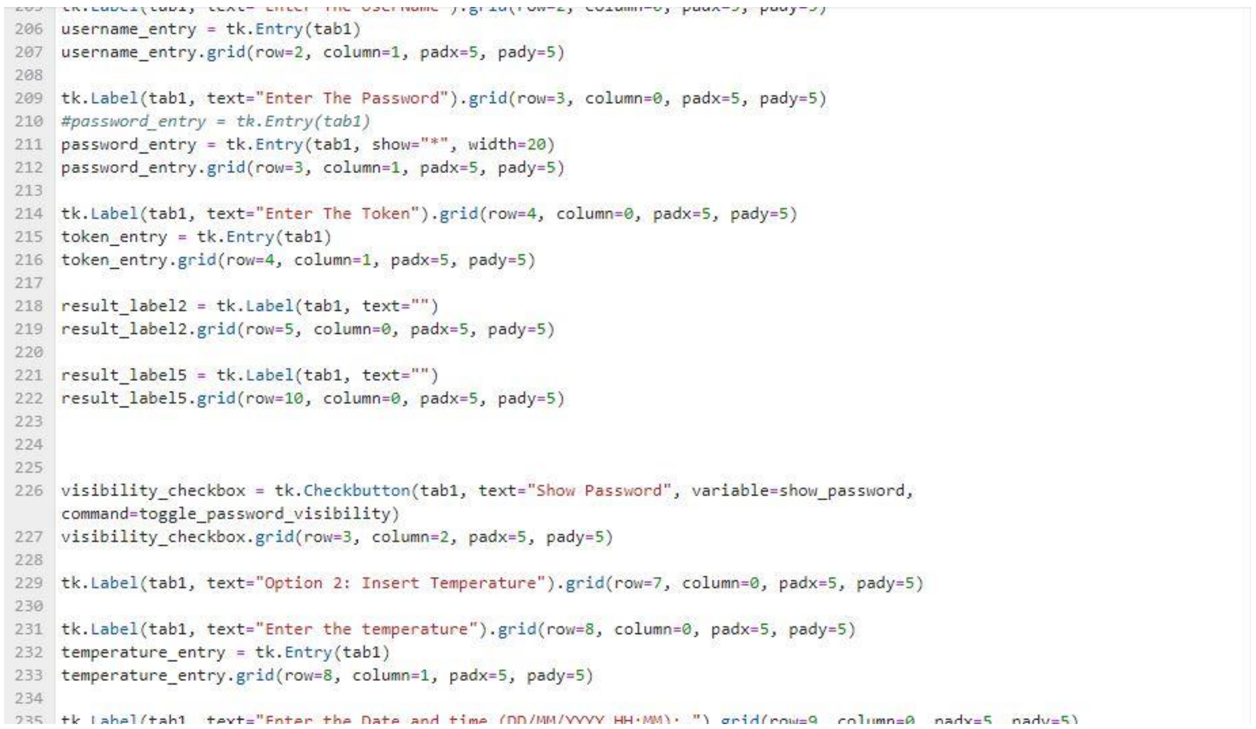
```python

def handle_checkbox1():
    global selected_option
    selected_option = "downloaded"
    var2.set(0)  # Uncheck checkbox2

def handle_checkbox2():
    global selected_option
    selected_option = "inserted"
    var1.set(0)  # Uncheck checkbox1




def long_function():
    # Get the selected sensor from the combobox
    selected_sensor = combobox.get()
    start_date = start_entry_pro.get()
    end_date = end_entry_pro.get()



    if selected_sensor == "S3":
        if selected_option == "downloaded":
            from production_s3 import wqp_s3
            # Call the wqp_s3() function
            result = wqp_s3(start_date, end_date)
            result_label4.config(text="CHL and TSM Maps Created successfully")
        elif selected_option == "inserted":
            from production_s3 import wqp_s3_ins
```



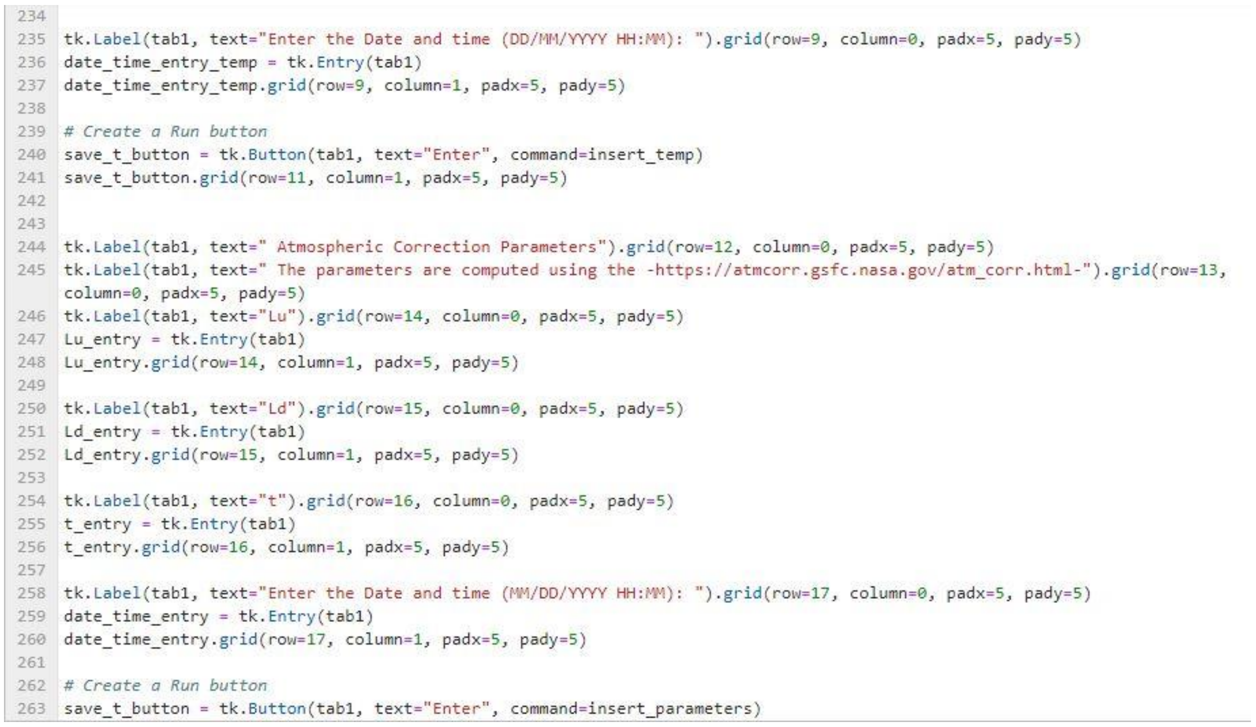
```python
            from production_s3 import wqp_s3_ins
            # Call the wqp_s3_ins() function
            result = wqp_s3_ins(start_date, end_date)
            result_label4.config(text="CHL and TSM Maps Created successfully")
        else:
            result = None
    elif selected_sensor == "Landsat 8":
        from production_L8 import tem_L8
        # Call the tem_L8() function
        result = tem_L8()
        result_label4.config(text="LWST Map Created successfully")
    else:
        result = None

    # Put the result in the queue
    result_queue.put(result)





# Create the main window
window = tk.Tk()

# Create a StringVar to track the visibility state of the password
show_password = tk.BooleanVar()
show_password.set(False)

# Create a StringVar to track the visibility state of the password
show_password1 = tk.BooleanVar()
```



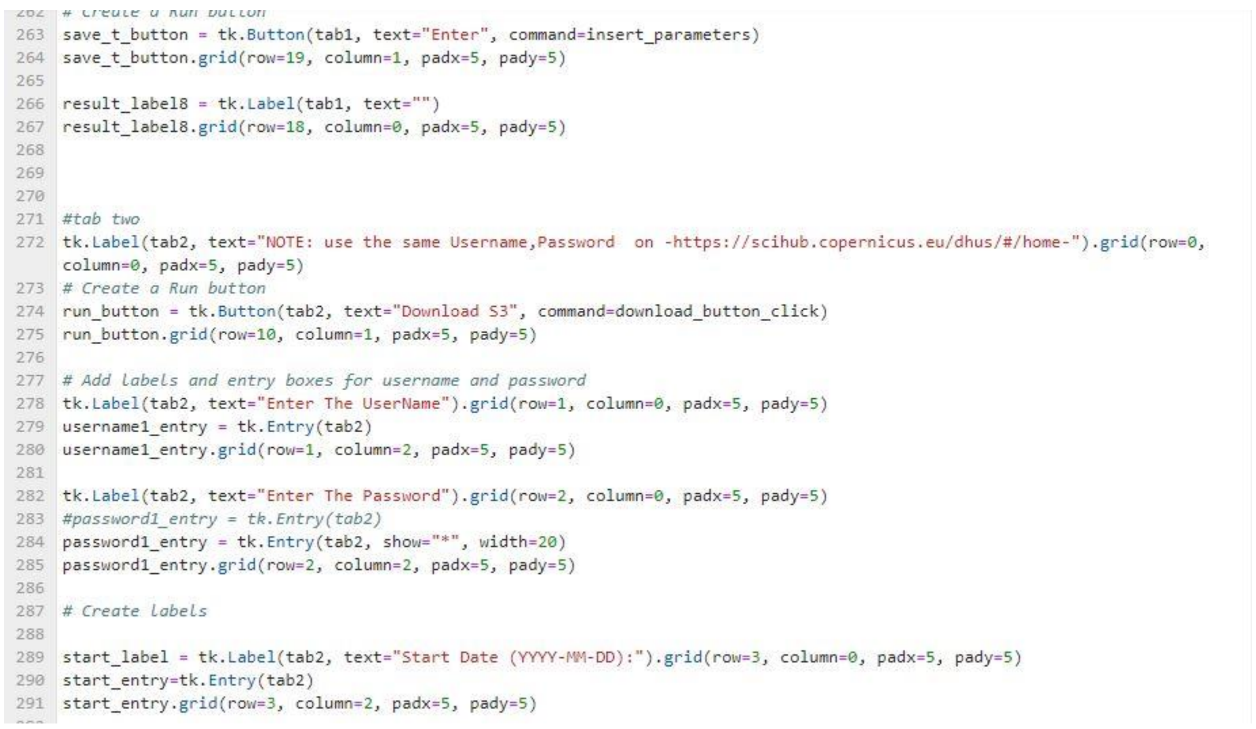
```python
show_password1 = tk.BooleanVar()
show_password1.set(False)

# Set the window title
window.title("Window with Run Button")
window.geometry("1700x1700")

# Create a Notebook widget
notebook = ttk.Notebook(window)
notebook.grid(row=0, column=0, padx=10, pady=10)

# Create tabs
tab1 = ttk.Frame(notebook)
tab2 = ttk.Frame(notebook)
tab3 = ttk.Frame(notebook)

# Add tabs to the notebook
notebook.add(tab1, text="Additional Dataset")
notebook.add(tab2, text="Download Sentinel-3 imagery")
notebook.add(tab3, text="Create Maps")

#tab one
tk.Label(tab1, text="Option 1: Download Temperature ").grid(row=0, column=0, padx=5, pady=5)
tk.Label(tab1, text="NOTE: use the same Username,Password and Token on -dati.lombardia.it-").grid(row=1, column=0, padx=5, pady=5)
# Create a Run button
run_button = tk.Button(tab1, text="Download MetoData", command=on_download_meteo_data)
run_button.grid(row=6, column=1, padx=5, pady=5)
# Add labels and entry boxes for username and password
tk.Label(tab1, text="Enter The UserName").grid(row=2, column=0, padx=5, pady=5)
username_entry = tk.Entry(tab1)
```



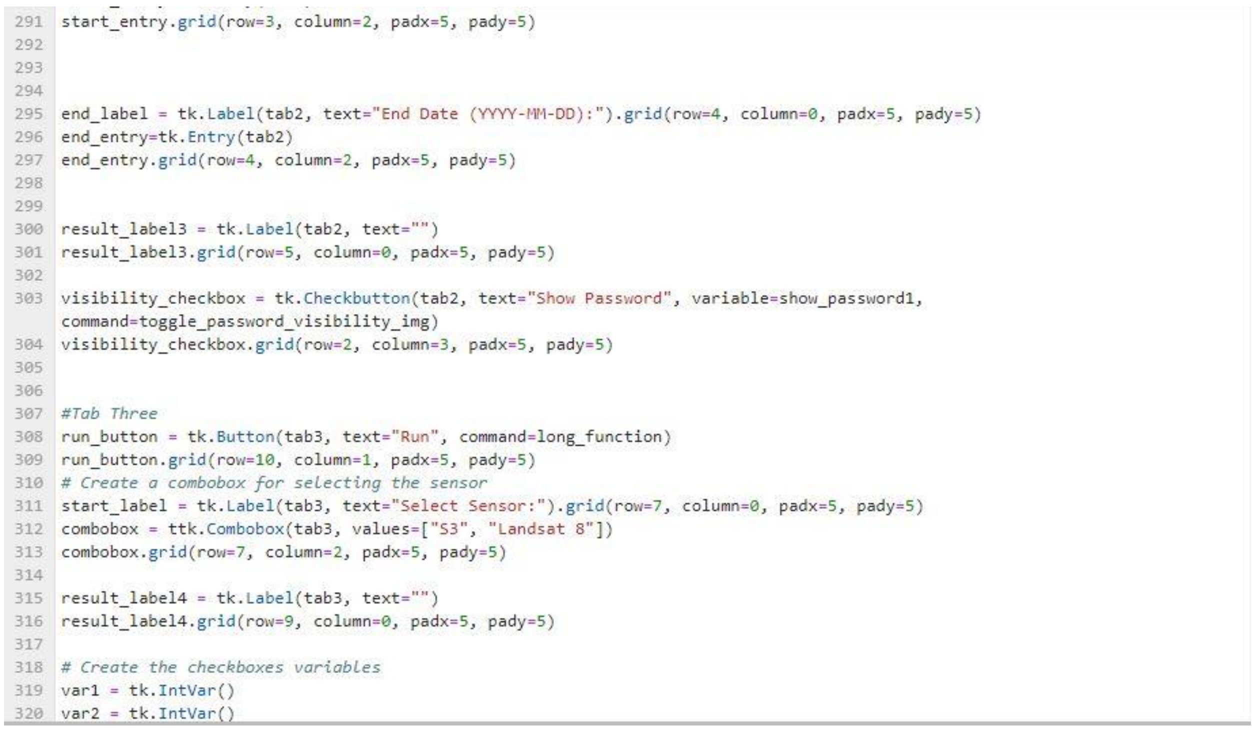
```
tk.Label(tab1, text="Enter The UserName").grid(row=2, column=0, padx=5, pady=5)
username_entry = tk.Entry(tab1)
username_entry.grid(row=2, column=1, padx=5, pady=5)

tk.Label(tab1, text="Enter The Password").grid(row=3, column=0, padx=5, pady=5)
#password_entry = tk.Entry(tab1)
password_entry = tk.Entry(tab1, show="*", width=20)
password_entry.grid(row=3, column=1, padx=5, pady=5)

tk.Label(tab1, text="Enter The Token").grid(row=4, column=0, padx=5, pady=5)
token_entry = tk.Entry(tab1)
token_entry.grid(row=4, column=1, padx=5, pady=5)

result_label2 = tk.Label(tab1, text="")
result_label2.grid(row=5, column=0, padx=5, pady=5)

result_label5 = tk.Label(tab1, text="")
result_label5.grid(row=10, column=0, padx=5, pady=5)



visibility_checkbox = tk.Checkbutton(tab1, text="Show Password", variable=show_password, command=toggle_password_visibility)
visibility_checkbox.grid(row=3, column=2, padx=5, pady=5)

tk.Label(tab1, text="Option 2: Insert Temperature").grid(row=7, column=0, padx=5, pady=5)

tk.Label(tab1, text="Enter the temperature").grid(row=8, column=0, padx=5, pady=5)
temperature_entry = tk.Entry(tab1)
temperature_entry.grid(row=8, column=1, padx=5, pady=5)

tk.Label(tab1, text="Enter the Date and time (DD/MM/YYYY HH:MM): ").grid(row=9, column=0, padx=5, pady=5)
```



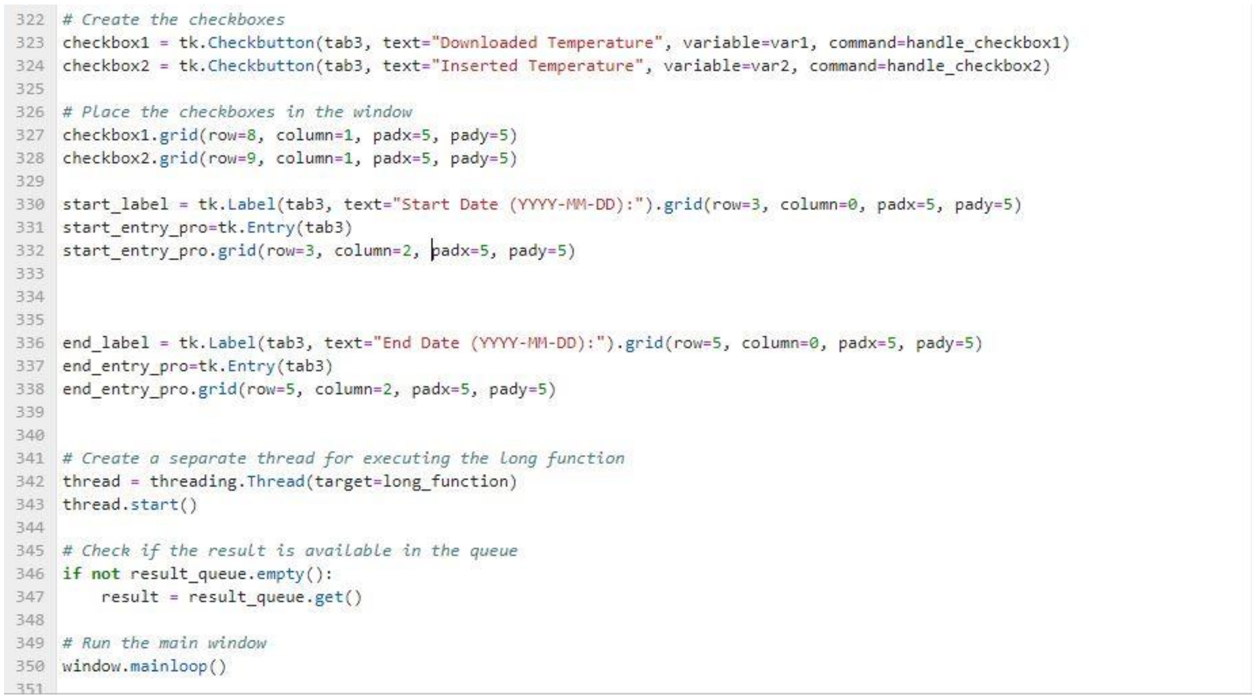
```

tk.Label(tab1, text="Enter the Date and time (DD/MM/YYYY HH:MM): ").grid(row=9, column=0, padx=5, pady=5)
date_time_entry_temp = tk.Entry(tab1)
date_time_entry_temp.grid(row=9, column=1, padx=5, pady=5)

# Create a Run button
save_t_button = tk.Button(tab1, text="Enter", command=insert_temp)
save_t_button.grid(row=11, column=1, padx=5, pady=5)


tk.Label(tab1, text=" Atmospheric Correction Parameters").grid(row=12, column=0, padx=5, pady=5)
tk.Label(tab1, text=" The parameters are computed using the -https://atmcorr.gsfc.nasa.gov/atm_corr.html-").grid(row=13, column=0, padx=5, pady=5)
tk.Label(tab1, text="Lu").grid(row=14, column=0, padx=5, pady=5)
Lu_entry = tk.Entry(tab1)
Lu_entry.grid(row=14, column=1, padx=5, pady=5)

tk.Label(tab1, text="Ld").grid(row=15, column=0, padx=5, pady=5)
Ld_entry = tk.Entry(tab1)
Ld_entry.grid(row=15, column=1, padx=5, pady=5)

tk.Label(tab1, text="t").grid(row=16, column=0, padx=5, pady=5)
t_entry = tk.Entry(tab1)
t_entry.grid(row=16, column=1, padx=5, pady=5)

tk.Label(tab1, text="Enter the Date and time (MM/DD/YYYY HH:MM): ").grid(row=17, column=0, padx=5, pady=5)
date_time_entry = tk.Entry(tab1)
date_time_entry.grid(row=17, column=1, padx=5, pady=5)

# Create a Run button
save_t_button = tk.Button(tab1, text="Enter", command=insert_parameters)
```



```python
# Create a Run button
save_t_button = tk.Button(tab1, text="Enter", command=insert_parameters)
save_t_button.grid(row=19, column=1, padx=5, pady=5)

result_label8 = tk.Label(tab1, text="")
result_label8.grid(row=18, column=0, padx=5, pady=5)



#tab two
tk.Label(tab2, text="NOTE: use the same Username,Password  on -https://scihub.copernicus.eu/dhus/#/home-").grid(row=0, column=0, padx=5, pady=5)
# Create a Run button
run_button = tk.Button(tab2, text="Download S3", command=download_button_click)
run_button.grid(row=10, column=1, padx=5, pady=5)

# Add labels and entry boxes for username and password
tk.Label(tab2, text="Enter The UserName").grid(row=1, column=0, padx=5, pady=5)
username1_entry = tk.Entry(tab2)
username1_entry.grid(row=1, column=2, padx=5, pady=5)

tk.Label(tab2, text="Enter The Password").grid(row=2, column=0, padx=5, pady=5)
#password1_entry = tk.Entry(tab2)
password1_entry = tk.Entry(tab2, show="*", width=20)
password1_entry.grid(row=2, column=2, padx=5, pady=5)

# Create labels

start_label = tk.Label(tab2, text="Start Date (YYYY-MM-DD):").grid(row=3, column=0, padx=5, pady=5)
start_entry=tk.Entry(tab2)
start_entry.grid(row=3, column=2, padx=5, pady=5)
```



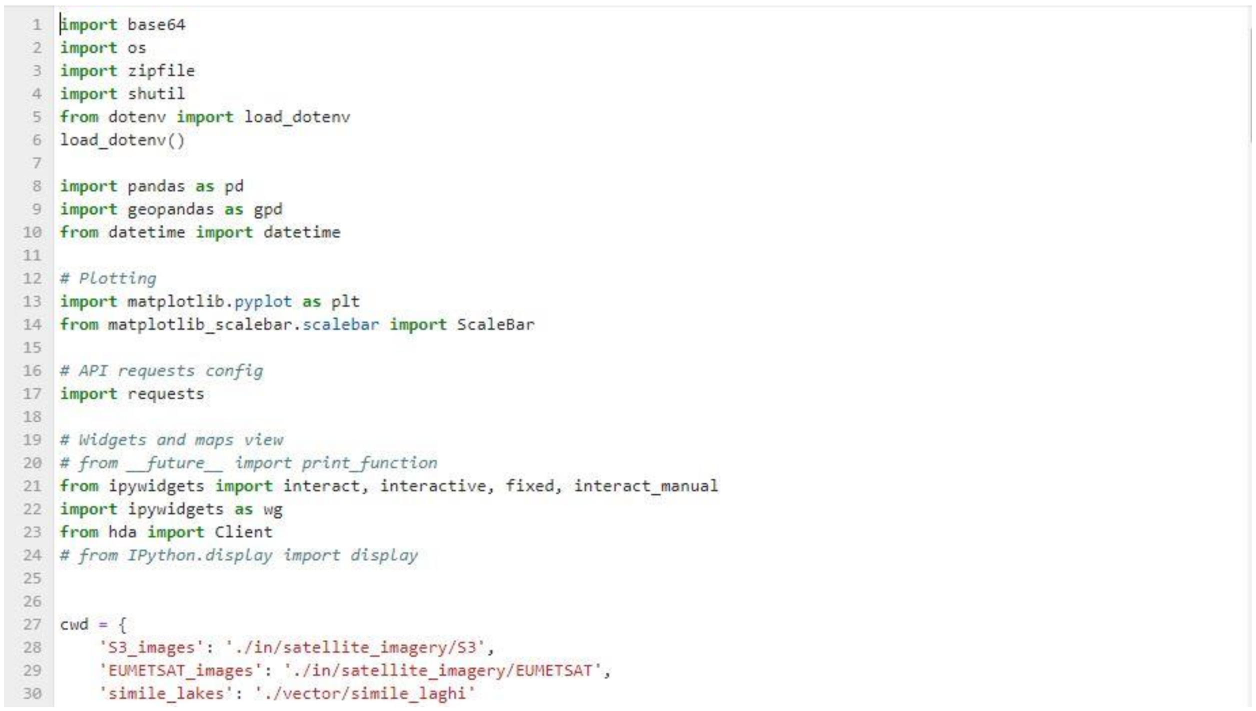
```python
start_entry.grid(row=3, column=2, padx=5, pady=5)



end_label = tk.Label(tab2, text="End Date (YYYY-MM-DD):").grid(row=4, column=0, padx=5, pady=5)
end_entry=tk.Entry(tab2)
end_entry.grid(row=4, column=2, padx=5, pady=5)


result_label3 = tk.Label(tab2, text="")
result_label3.grid(row=5, column=0, padx=5, pady=5)

visibility_checkbox = tk.Checkbutton(tab2, text="Show Password", variable=show_password1, command=toggle_password_visibility_img)
visibility_checkbox.grid(row=2, column=3, padx=5, pady=5)


#Tab Three
run_button = tk.Button(tab3, text="Run", command=long_function)
run_button.grid(row=10, column=1, padx=5, pady=5)
# Create a combobox for selecting the sensor
start_label = tk.Label(tab3, text="Select Sensor:").grid(row=7, column=0, padx=5, pady=5)
combobox = ttk.Combobox(tab3, values=["S3", "Landsat 8"])
combobox.grid(row=7, column=2, padx=5, pady=5)

result_label4 = tk.Label(tab3, text="")
result_label4.grid(row=9, column=0, padx=5, pady=5)

# Create the checkboxes variables
var1 = tk.IntVar()
var2 = tk.IntVar()
```



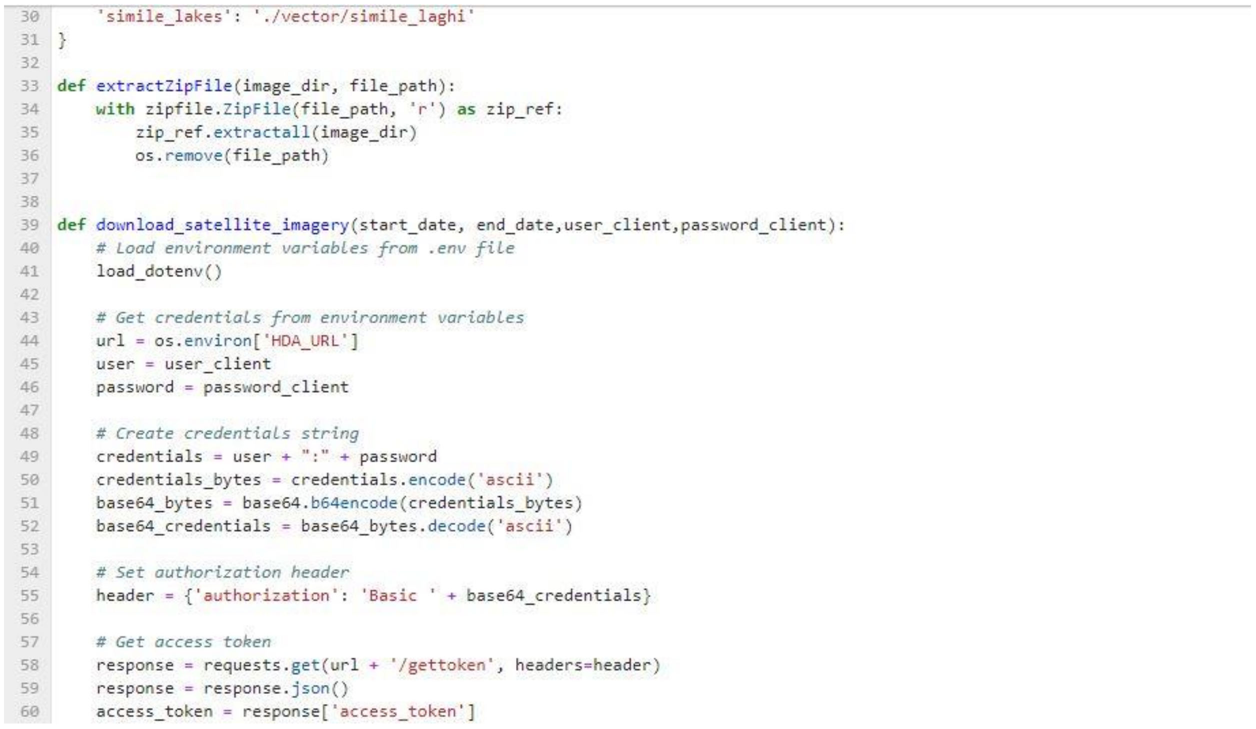
```python
# Create the checkboxes
checkbox1 = tk.Checkbutton(tab3, text="Downloaded Temperature", variable=var1, command=handle_checkbox1)
checkbox2 = tk.Checkbutton(tab3, text="Inserted Temperature", variable=var2, command=handle_checkbox2)

# Place the checkboxes in the window
checkbox1.grid(row=8, column=1, padx=5, pady=5)
checkbox2.grid(row=9, column=1, padx=5, pady=5)

start_label = tk.Label(tab3, text="Start Date (YYYY-MM-DD):").grid(row=3, column=0, padx=5, pady=5)
start_entry_pro=tk.Entry(tab3)
start_entry_pro.grid(row=3, column=2, padx=5, pady=5)



end_label = tk.Label(tab3, text="End Date (YYYY-MM-DD):").grid(row=5, column=0, padx=5, pady=5)
end_entry_pro=tk.Entry(tab3)
end_entry_pro.grid(row=5, column=2, padx=5, pady=5)


# Create a separate thread for executing the long function
thread = threading.Thread(target=long_function)
thread.start()

# Check if the result is available in the queue
if not result_queue.empty():
    result = result_queue.get()

# Run the main window
window.mainloop()

```

Listing 1. Design and Creation of Graphical User Interface (GUI)

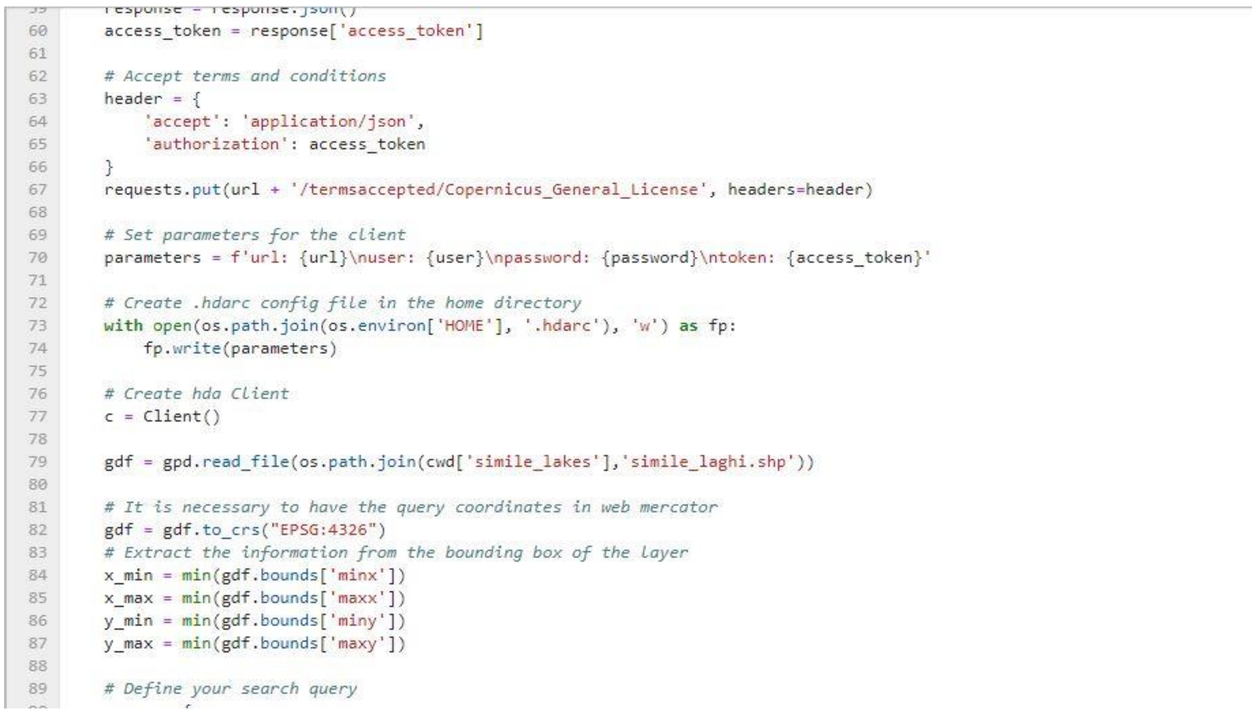
Download Sentinel 3.



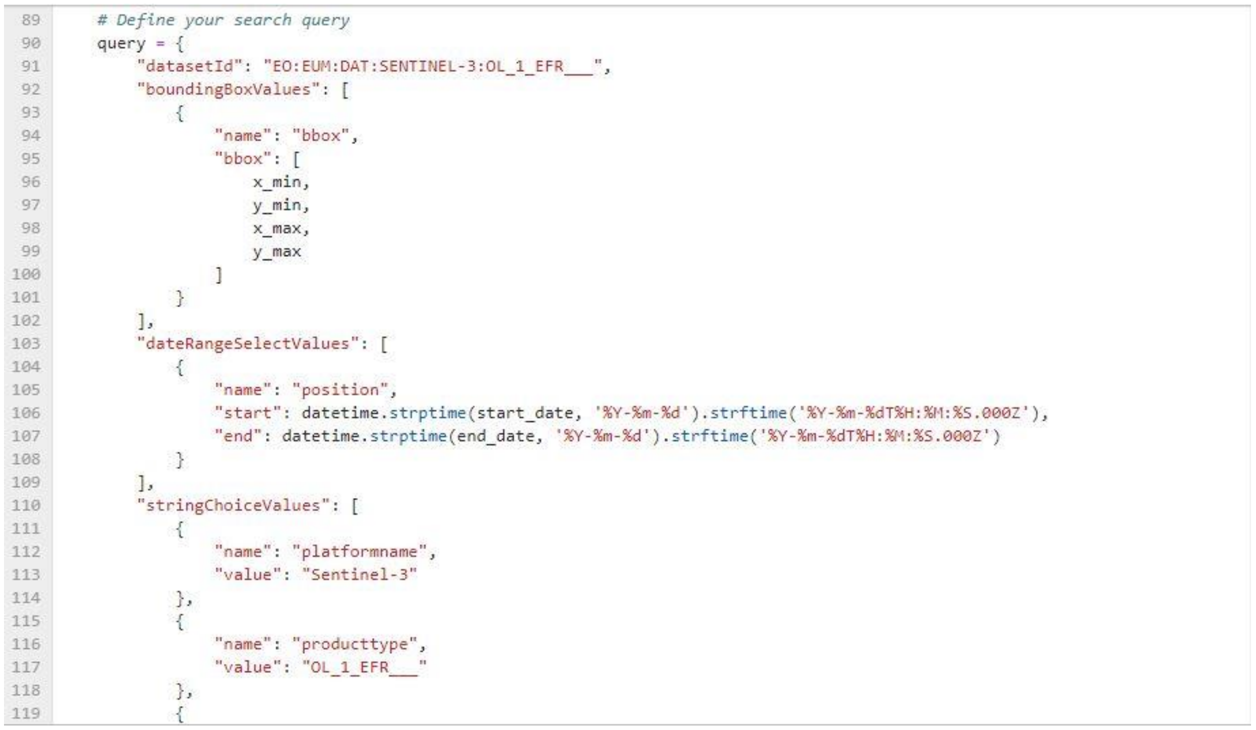
```python
import base64
import os
import zipfile
import shutil
from dotenv import load_dotenv
load_dotenv()

import pandas as pd
import geopandas as gpd
from datetime import datetime

# Plotting
import matplotlib.pyplot as plt
from matplotlib_scalebar.scalebar import ScaleBar

# API requests config
import requests

# Widgets and maps view
# from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as wg
from hda import Client
# from IPython.display import display


cwd = {
    'S3_images': './in/satellite_imagery/S3',
    'EUMETSAT_images': './in/satellite_imagery/EUMETSAT',
    'simile_lakes': './vector/simile_laghi'
```



```python
    'simile_lakes': './vector/simile_laghi'
}

def extractZipFile(image_dir, file_path):
    with zipfile.ZipFile(file_path, 'r') as zip_ref:
        zip_ref.extractall(image_dir)
        os.remove(file_path)


def download_satellite_imagery(start_date, end_date,user_client,password_client):
    # Load environment variables from .env file
    load_dotenv()

    # Get credentials from environment variables
    url = os.environ['HDA_URL']
    user = user_client
    password = password_client

    # Create credentials string
    credentials = user + ":" + password
    credentials_bytes = credentials.encode('ascii')
    base64_bytes = base64.b64encode(credentials_bytes)
    base64_credentials = base64_bytes.decode('ascii')

    # Set authorization header
    header = {'authorization': 'Basic ' + base64_credentials}

    # Get access token
    response = requests.get(url + '/gettoken', headers=header)
    response = response.json()
    access_token = response['access_token']
```



```
    response = response.json()
    access_token = response['access_token']

    # Accept terms and conditions
    header = {
        'accept': 'application/json',
        'authorization': access_token
    }
    requests.put(url + '/termsaccepted/Copernicus_General_License', headers=header)

    # Set parameters for the client
    parameters = f'url: {url}\nuser: {user}\npassword: {password}\ntoken: {access_token}'

    # Create .hdarc config file in the home directory
    with open(os.path.join(os.environ['HOME'], '.hdarc'), 'w') as fp:
        fp.write(parameters)

    # Create hda Client
    c = Client()

    gdf = gpd.read_file(os.path.join(cwd['simile_lakes'],'simile_laghi.shp'))

    # It is necessary to have the query coordinates in web mercator
    gdf = gdf.to_crs("EPSG:4326")
    # Extract the information from the bounding box of the layer
    x_min = min(gdf.bounds['minx'])
    x_max = min(gdf.bounds['maxx'])
    y_min = min(gdf.bounds['miny'])
    y_max = min(gdf.bounds['maxy'])

    # Define your search query
```



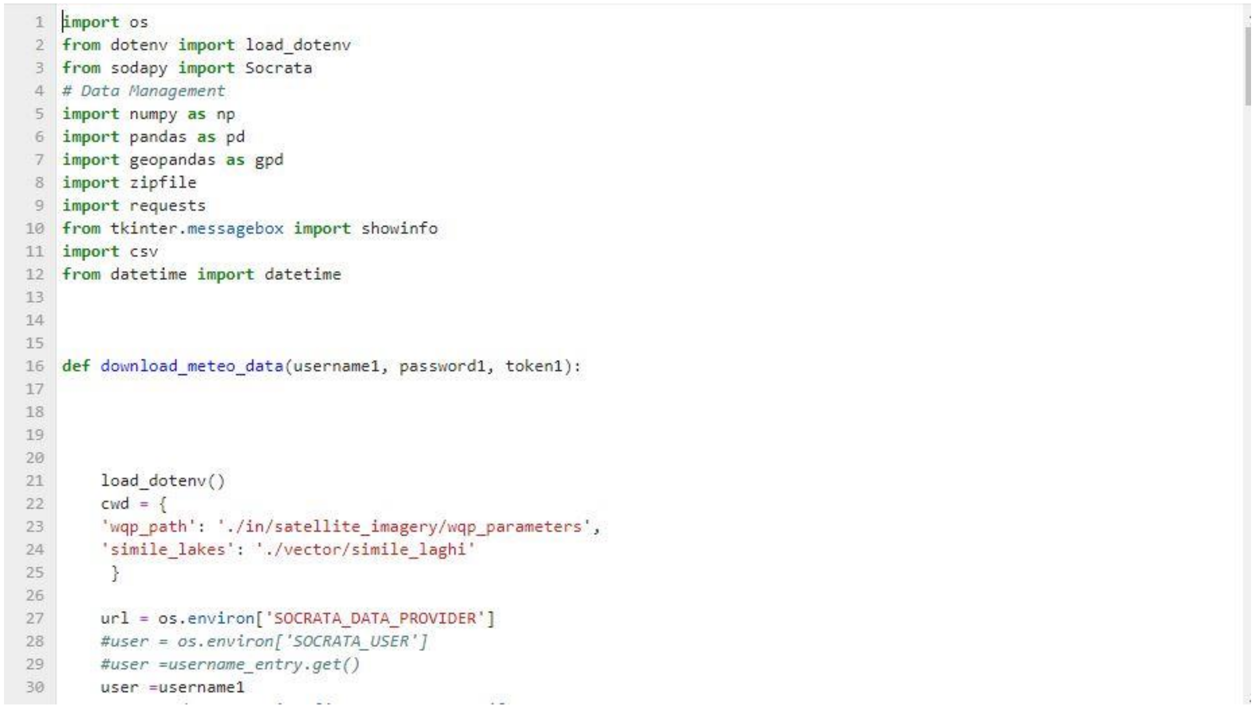
```
    # Define your search query
    query = {
        "datasetId": "EO:EUM:DAT:SENTINEL-3:OL_1_EFR___",
        "boundingBoxValues": [
            {
                "name": "bbox",
                "bbox": [
                    x_min,
                    y_min,
                    x_max,
                    y_max
                ]
            }
        ],
        "dateRangeSelectValues": [
            {
                "name": "position",
                "start": datetime.strptime(start_date, '%Y-%m-%d').strftime('%Y-%m-%dT%H:%M:%S.000Z'),
                "end": datetime.strptime(end_date, '%Y-%m-%d').strftime('%Y-%m-%dT%H:%M:%S.000Z')
            }
        ],
        "stringChoiceValues": [
            {
                "name": "platformname",
                "value": "Sentinel-3"
            },
            {
                "name": "producttype",
                "value": "OL_1_EFR___"
            },
            {
```

```python
            {
                "name": "timeliness",
                "value": "NT"
            }
        ]
    }

    # Run the query and download the products
    matches = c.search(query)
    matches.download()

    for product in matches.__dict__['results']:
        shutil.move(product['filename'], cwd['S3_images'])


    for root, dirs, files in os.walk(cwd['S3_images']):
        for zip_name in files:
            file_path = os.path.join(root,zip_name)
            print(f'Currently extracting: {zip_name}')
        # Exception required for products retrieved through the HDA API. The products download include the .SEN3 extension and not a compressed file format
            if zip_name.endswith('.SEN3'):
                os.rename(os.path.join(cwd['S3_images'],zip_name),os.path.join(cwd['S3_images'],zip_name.split('.')[0]+'.zip'))
                file_path = os.path.join(cwd['S3_images'],zip_name.split('.')[0]+'.zip')
                extractZipFile(cwd['S3_images'], file_path)
            elif zip_name.endswith('.zip'):
                extractZipFile(cwd['S3_images'], file_path)


```

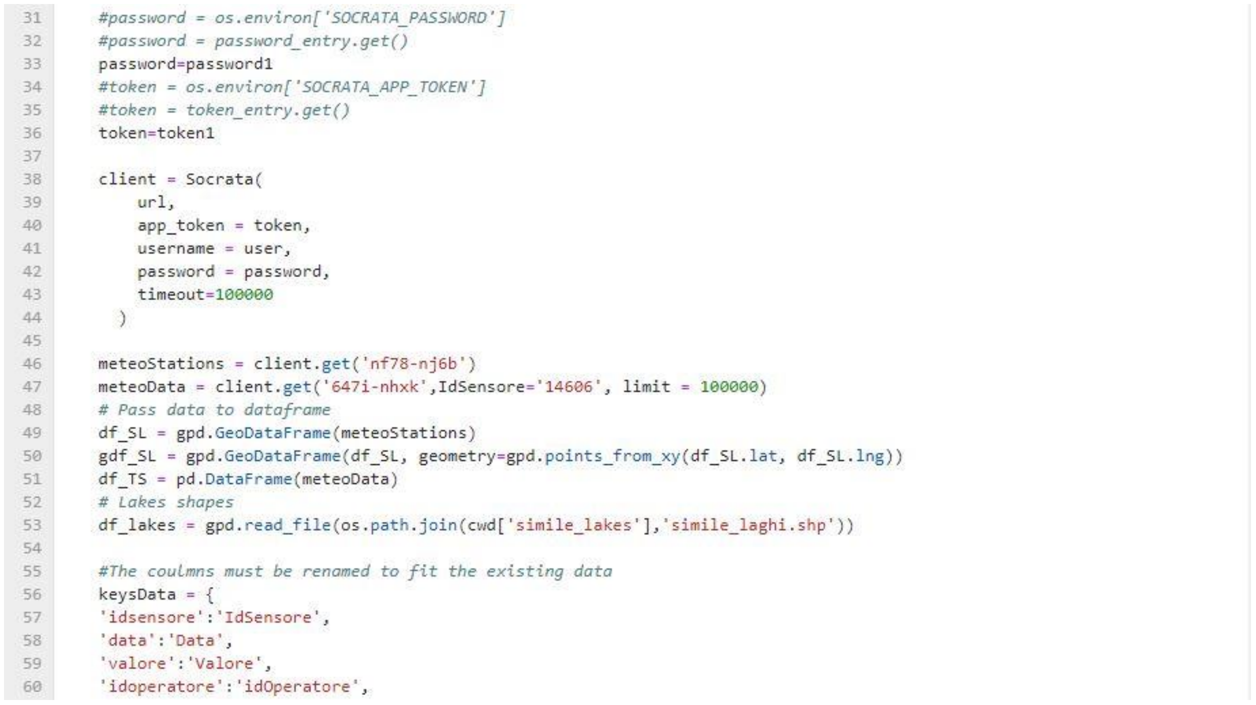
```python
    # Return a result or status message
    return "Satellite imagery downloaded successfully"
```

Listing 2. Function that is used to Download Sentinel-3 imagery.

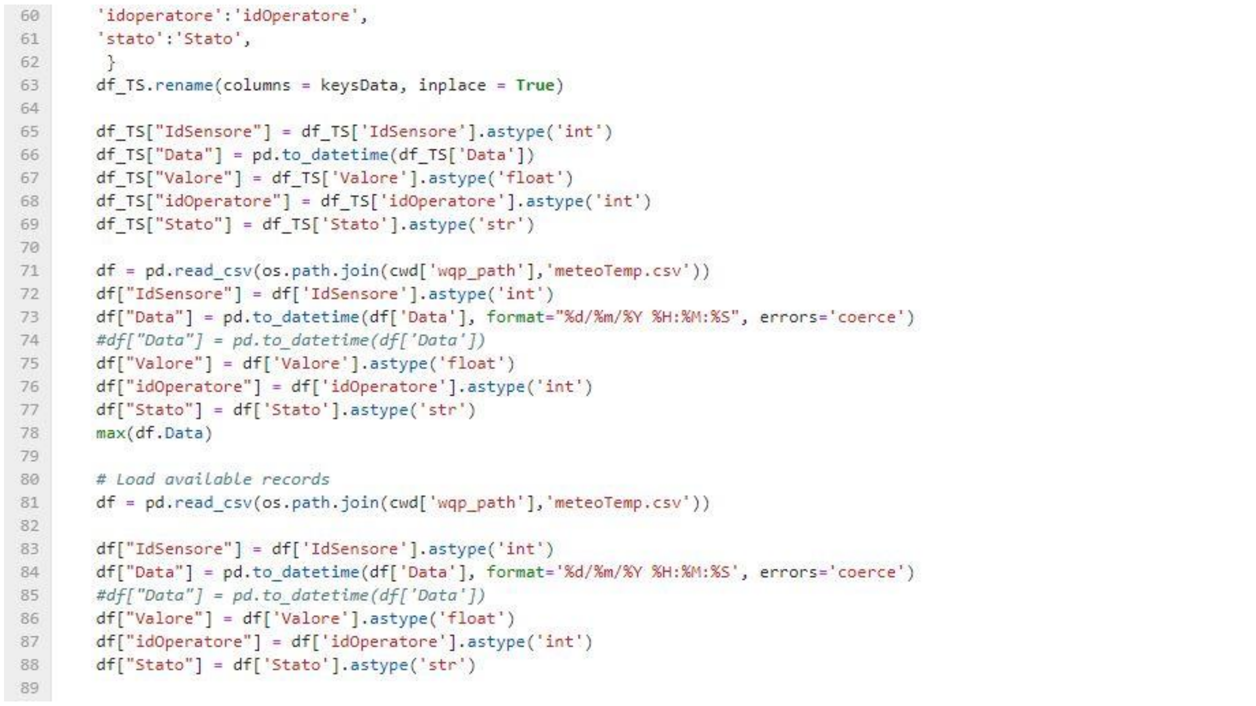
Additional Parameters



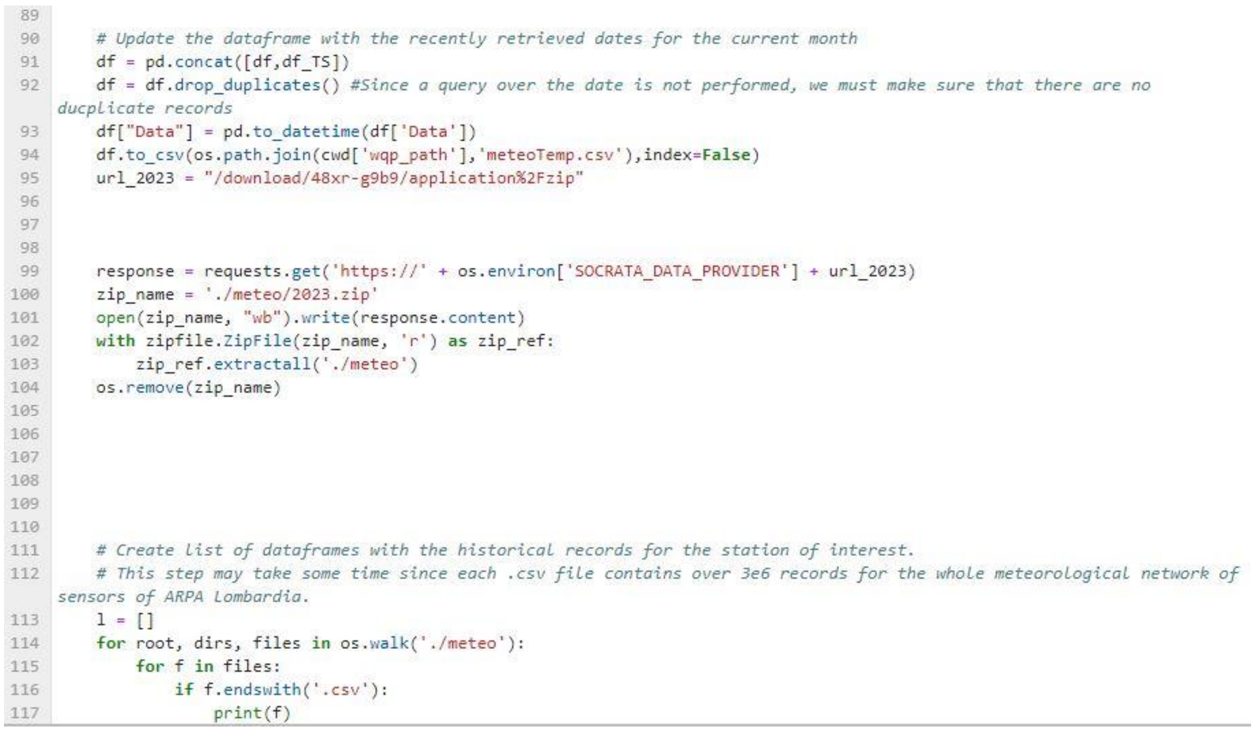
```python
import os
from dotenv import load_dotenv
from sodapy import Socrata
# Data Management
import numpy as np
import pandas as pd
import geopandas as gpd
import zipfile
import requests
from tkinter.messagebox import showinfo
import csv
from datetime import datetime



def download_meteo_data(username1, password1, token1):




    load_dotenv()
    cwd = {
    'wqp_path': './in/satellite_imagery/wqp_parameters',
    'simile_lakes': './vector/simile_laghi'
     }

    url = os.environ['SOCRATA_DATA_PROVIDER']
    #user = os.environ['SOCRATA_USER']
    #user =username_entry.get()
    user =username1
```



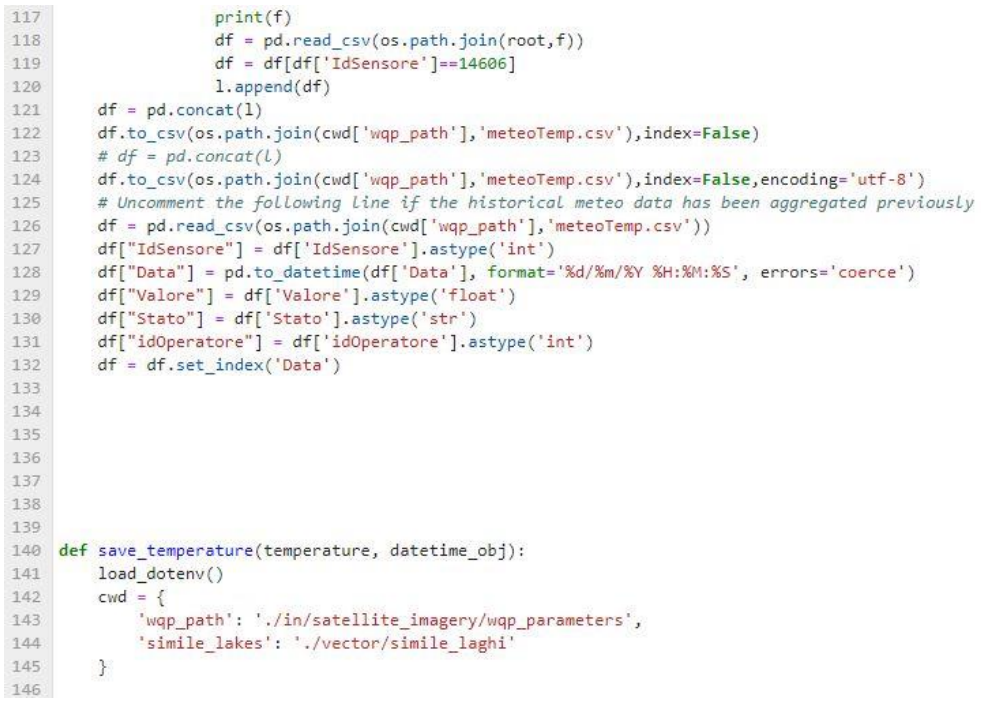
```python
    #password = os.environ['SOCRATA_PASSWORD']
    #password = password_entry.get()
    password=password1
    #token = os.environ['SOCRATA_APP_TOKEN']
    #token = token_entry.get()
    token=token1

    client = Socrata(
        url,
        app_token = token,
        username = user,
        password = password,
        timeout=100000
      )

    meteoStations = client.get('nf78-nj6b')
    meteoData = client.get('647i-nhxk',IdSensore='14606', limit = 100000)
    # Pass data to dataframe
    df_SL = gpd.GeoDataFrame(meteoStations)
    gdf_SL = gpd.GeoDataFrame(df_SL, geometry=gpd.points_from_xy(df_SL.lat, df_SL.lng))
    df_TS = pd.DataFrame(meteoData)
    # Lakes shapes
    df_lakes = gpd.read_file(os.path.join(cwd['simile_lakes'],'simile_laghi.shp'))

    #The coulmns must be renamed to fit the existing data
    keysData = {
    'idsensore':'IdSensore',
    'data':'Data',
    'valore':'Valore',
    'idoperatore':'idOperatore',
```



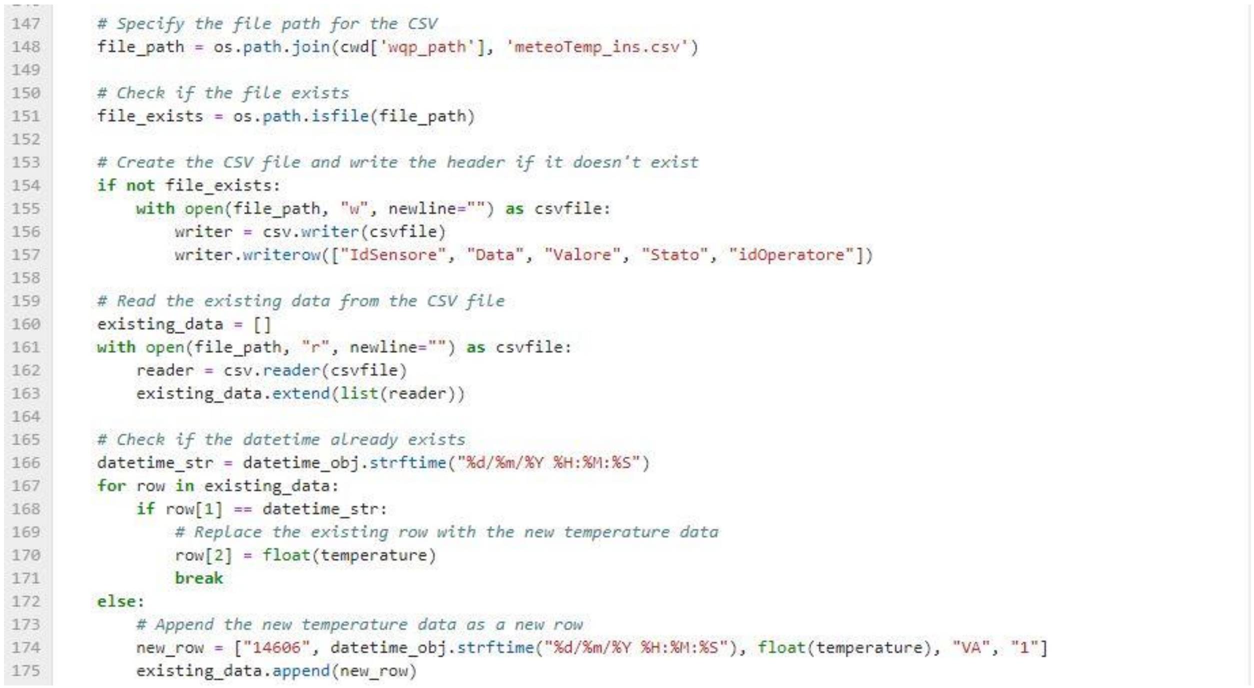
```python
    'idoperatore':'idOperatore',
    'stato':'Stato',
     }
    df_TS.rename(columns = keysData, inplace = True)

    df_TS["IdSensore"] = df_TS['IdSensore'].astype('int')
    df_TS["Data"] = pd.to_datetime(df_TS['Data'])
    df_TS["Valore"] = df_TS['Valore'].astype('float')
    df_TS["idOperatore"] = df_TS['idOperatore'].astype('int')
    df_TS["Stato"] = df_TS['Stato'].astype('str')

    df = pd.read_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'))
    df["IdSensore"] = df['IdSensore'].astype('int')
    df["Data"] = pd.to_datetime(df['Data'], format="%d/%m/%Y %H:%M:%S", errors='coerce')
    #df["Data"] = pd.to_datetime(df['Data'])
    df["Valore"] = df['Valore'].astype('float')
    df["idOperatore"] = df['idOperatore'].astype('int')
    df["Stato"] = df['Stato'].astype('str')
    max(df.Data)

    # Load available records
    df = pd.read_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'))

    df["IdSensore"] = df['IdSensore'].astype('int')
    df["Data"] = pd.to_datetime(df['Data'], format='%d/%m/%Y %H:%M:%S', errors='coerce')
    #df["Data"] = pd.to_datetime(df['Data'])
    df["Valore"] = df['Valore'].astype('float')
    df["idOperatore"] = df['idOperatore'].astype('int')
    df["Stato"] = df['Stato'].astype('str')

```



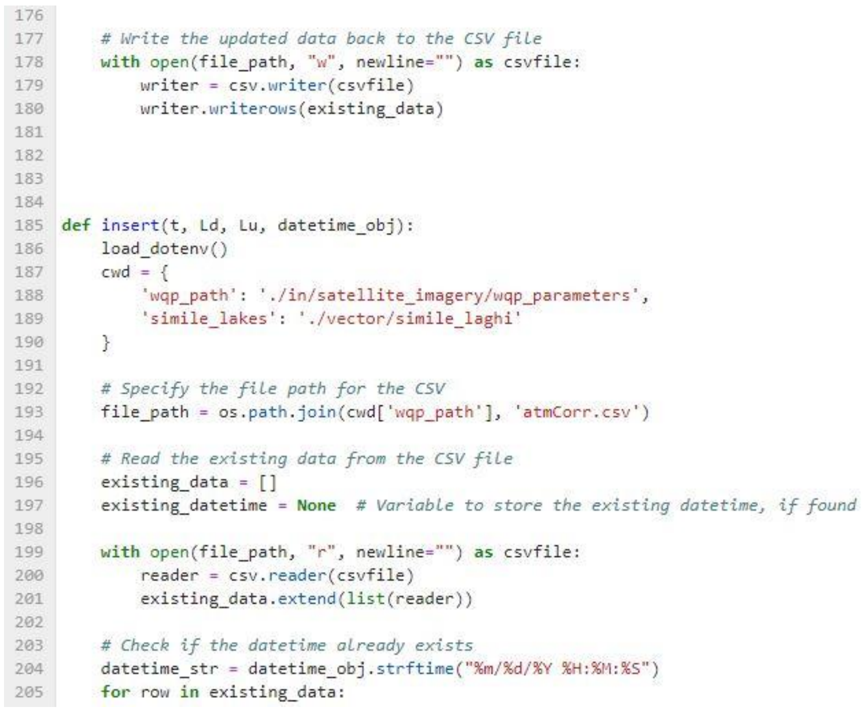
```python

    # Update the dataframe with the recently retrieved dates for the current month
    df = pd.concat([df,df_TS])
    df = df.drop_duplicates() #Since a query over the date is not performed, we must make sure that there are no ducplicate records
    df["Data"] = pd.to_datetime(df['Data'])
    df.to_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'),index=False)
    url_2023 = "/download/48xr-g9b9/application%2Fzip"



    response = requests.get('https://' + os.environ['SOCRATA_DATA_PROVIDER'] + url_2023)
    zip_name = './meteo/2023.zip'
    open(zip_name, "wb").write(response.content)
    with zipfile.ZipFile(zip_name, 'r') as zip_ref:
        zip_ref.extractall('./meteo')
    os.remove(zip_name)






    # Create list of dataframes with the historical records for the station of interest.
    # This step may take some time since each .csv file contains over 3e6 records for the whole meteorological network of sensors of ARPA Lombardia.
    l = []
    for root, dirs, files in os.walk('./meteo'):
        for f in files:
            if f.endswith('.csv'):
                print(f)
```



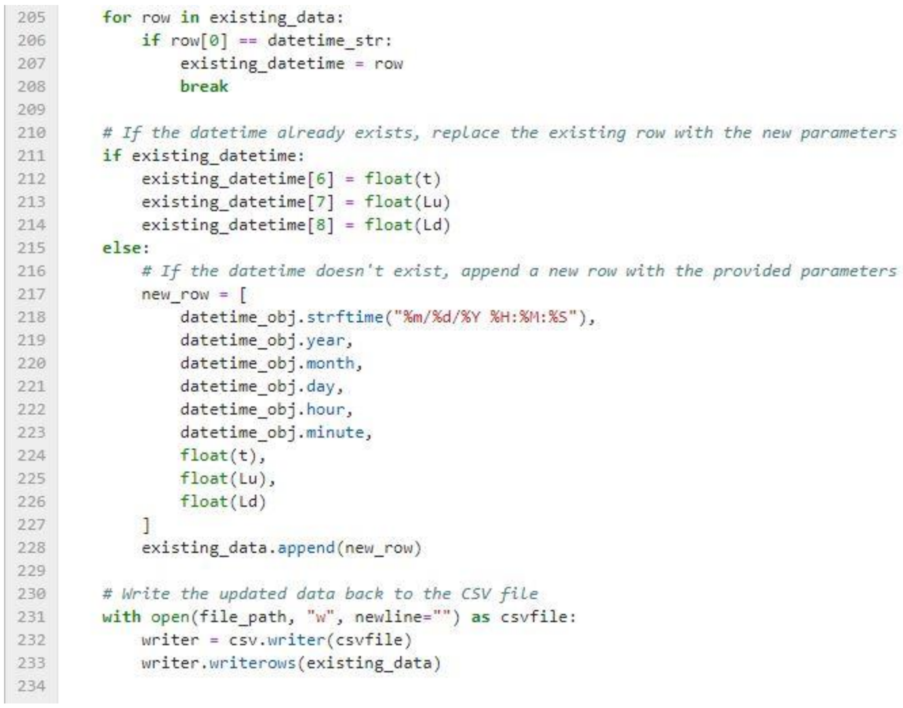
```python
                print(f)
                df = pd.read_csv(os.path.join(root,f))
                df = df[df['IdSensore']==14606]
                l.append(df)
    df = pd.concat(l)
    df.to_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'),index=False)
    # df = pd.concat(l)
    df.to_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'),index=False,encoding='utf-8')
    # Uncomment the following line if the historical meteo data has been aggregated previously
    df = pd.read_csv(os.path.join(cwd['wqp_path'],'meteoTemp.csv'))
    df["IdSensore"] = df['IdSensore'].astype('int')
    df["Data"] = pd.to_datetime(df['Data'], format='%d/%m/%Y %H:%M:%S', errors='coerce')
    df["Valore"] = df['Valore'].astype('float')
    df["Stato"] = df['Stato'].astype('str')
    df["idOperatore"] = df['idOperatore'].astype('int')
    df = df.set_index('Data')







def save_temperature(temperature, datetime_obj):
    load_dotenv()
    cwd = {
        'wqp_path': './in/satellite_imagery/wqp_parameters',
        'simile_lakes': './vector/simile_laghi'
    }

```



```python
    # Specify the file path for the CSV
    file_path = os.path.join(cwd['wqp_path'], 'meteoTemp_ins.csv')

    # Check if the file exists
    file_exists = os.path.isfile(file_path)

    # Create the CSV file and write the header if it doesn't exist
    if not file_exists:
        with open(file_path, "w", newline="") as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(["IdSensore", "Data", "Valore", "Stato", "idOperatore"])

    # Read the existing data from the CSV file
    existing_data = []
    with open(file_path, "r", newline="") as csvfile:
        reader = csv.reader(csvfile)
        existing_data.extend(list(reader))

    # Check if the datetime already exists
    datetime_str = datetime_obj.strftime("%d/%m/%Y %H:%M:%S")
    for row in existing_data:
        if row[1] == datetime_str:
            # Replace the existing row with the new temperature data
            row[2] = float(temperature)
            break
    else:
        # Append the new temperature data as a new row
        new_row = ["14606", datetime_obj.strftime("%d/%m/%Y %H:%M:%S"), float(temperature), "VA", "1"]
        existing_data.append(new_row)
```



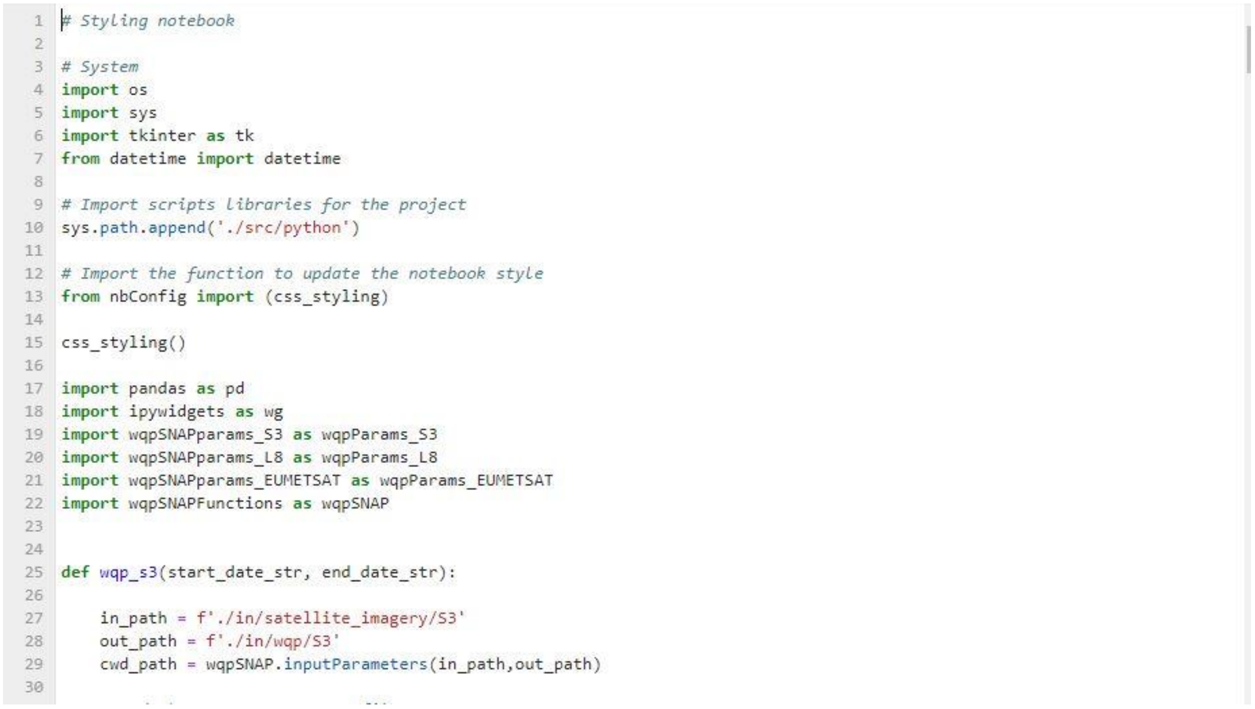
```python

    # Write the updated data back to the CSV file
    with open(file_path, "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(existing_data)




def insert(t, Ld, Lu, datetime_obj):
    load_dotenv()
    cwd = {
        'wqp_path': './in/satellite_imagery/wqp_parameters',
        'simile_lakes': './vector/simile_laghi'
    }

    # Specify the file path for the CSV
    file_path = os.path.join(cwd['wqp_path'], 'atmCorr.csv')

    # Read the existing data from the CSV file
    existing_data = []
    existing_datetime = None  # Variable to store the existing datetime, if found

    with open(file_path, "r", newline="") as csvfile:
        reader = csv.reader(csvfile)
        existing_data.extend(list(reader))

    # Check if the datetime already exists
    datetime_str = datetime_obj.strftime("%m/%d/%Y %H:%M:%S")
    for row in existing_data:
```



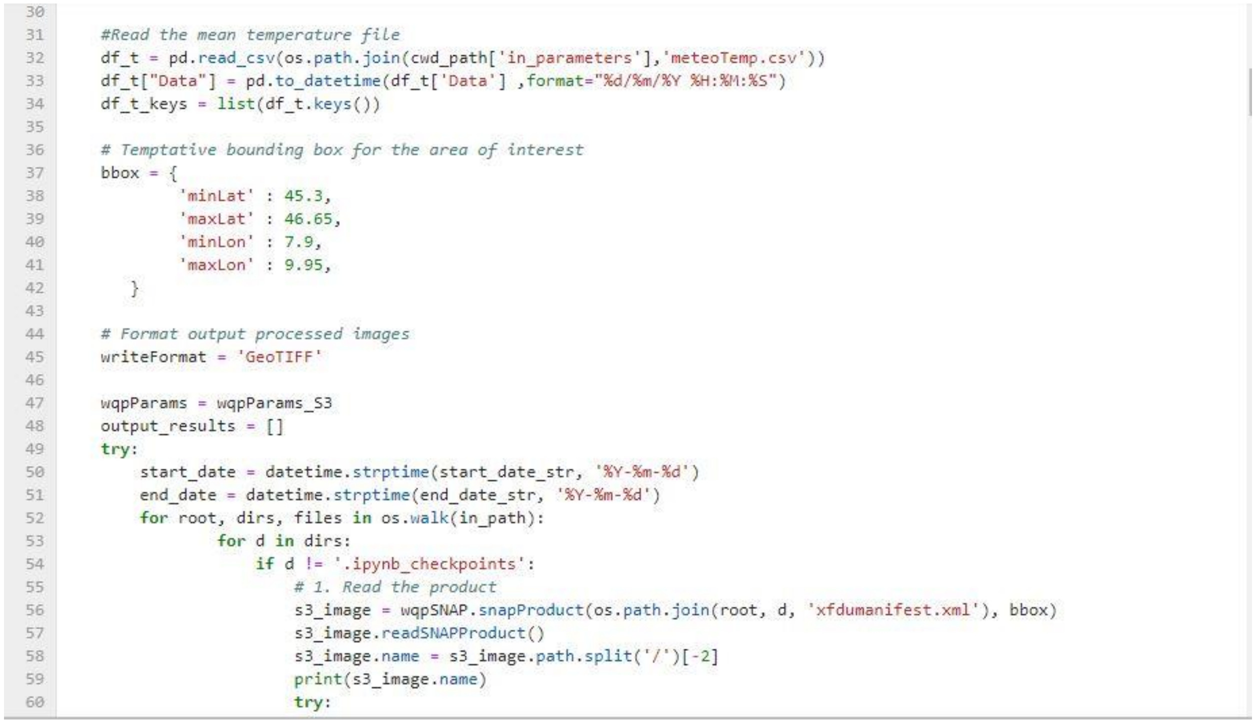
```python
    for row in existing_data:
        if row[0] == datetime_str:
            existing_datetime = row
            break

    # If the datetime already exists, replace the existing row with the new parameters
    if existing_datetime:
        existing_datetime[6] = float(t)
        existing_datetime[7] = float(Lu)
        existing_datetime[8] = float(Ld)
    else:
        # If the datetime doesn't exist, append a new row with the provided parameters
        new_row = [
            datetime_obj.strftime("%m/%d/%Y %H:%M:%S"),
            datetime_obj.year,
            datetime_obj.month,
            datetime_obj.day,
            datetime_obj.hour,
            datetime_obj.minute,
            float(t),
            float(Lu),
            float(Ld)
        ]
        existing_data.append(new_row)

    # Write the updated data back to the CSV file
    with open(file_path, "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(existing_data)

```

Listing 3. Function that is used to Download, Insert the Temperature and Atmospheric correction parameters.

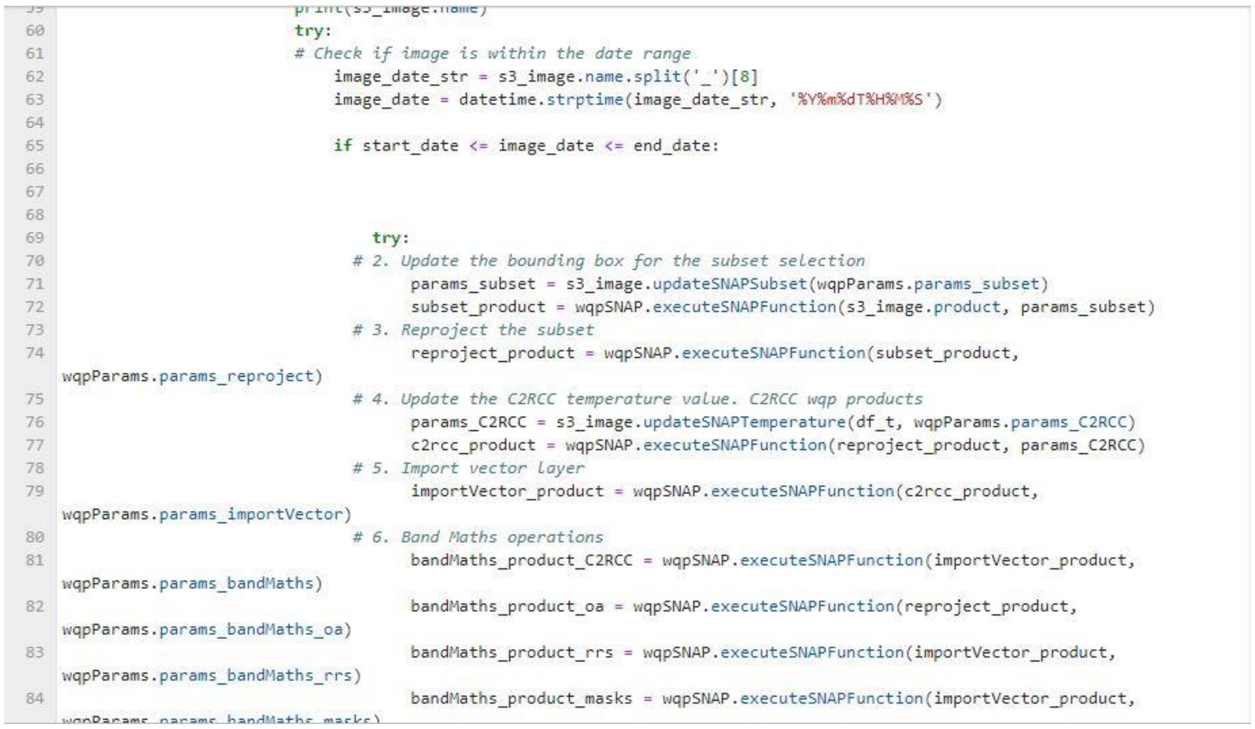
Production of CHL-a and TSM Maps



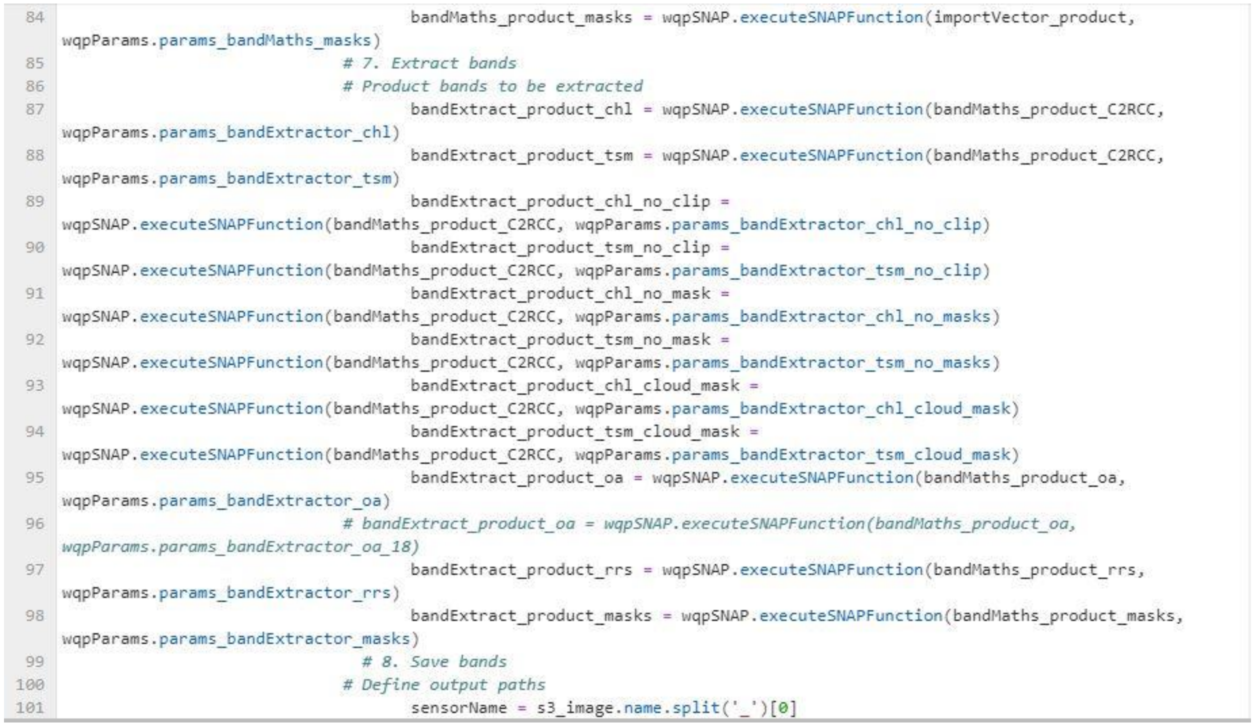
```python
# Styling notebook

# System
import os
import sys
import tkinter as tk
from datetime import datetime

# Import scripts libraries for the project
sys.path.append('./src/python')

# Import the function to update the notebook style
from nbConfig import (css_styling)

css_styling()

import pandas as pd
import ipywidgets as wg
import wqpSNAPparams_S3 as wqpParams_S3
import wqpSNAPparams_L8 as wqpParams_L8
import wqpSNAPparams_EUMETSAT as wqpParams_EUMETSAT
import wqpSNAPFunctions as wqpSNAP


def wqp_s3(start_date_str, end_date_str):

    in_path = f'./in/satellite_imagery/S3'
    out_path = f'./in/wqp/S3'
    cwd_path = wqpSNAP.inputParameters(in_path,out_path)

```



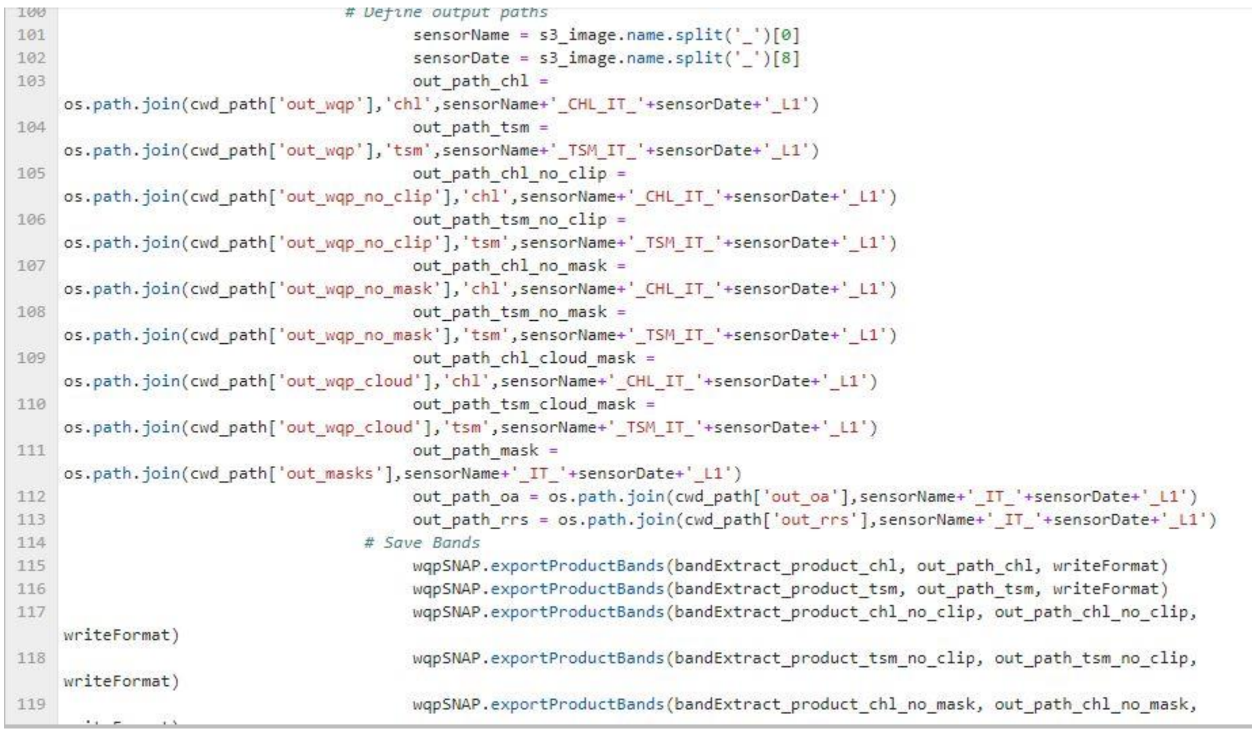
```python

    #Read the mean temperature file
    df_t = pd.read_csv(os.path.join(cwd_path['in_parameters'],'meteoTemp.csv'))
    df_t["Data"] = pd.to_datetime(df_t['Data'] ,format="%d/%m/%Y %H:%M:%S")
    df_t_keys = list(df_t.keys())

    # Temptative bounding box for the area of interest
    bbox = {
            'minLat' : 45.3,
            'maxLat' : 46.65,
            'minLon' : 7.9,
            'maxLon' : 9.95,
        }

    # Format output processed images
    writeFormat = 'GeoTIFF'

    wqpParams = wqpParams_S3
    output_results = []
    try:
        start_date = datetime.strptime(start_date_str, '%Y-%m-%d')
        end_date = datetime.strptime(end_date_str, '%Y-%m-%d')
        for root, dirs, files in os.walk(in_path):
                for d in dirs:
                    if d != '.ipynb_checkpoints':
                        # 1. Read the product
                        s3_image = wqpSNAP.snapProduct(os.path.join(root, d, 'xfdumanifest.xml'), bbox)
                        s3_image.readSNAPProduct()
                        s3_image.name = s3_image.path.split('/')[-2]
                        print(s3_image.name)
                        try:
```



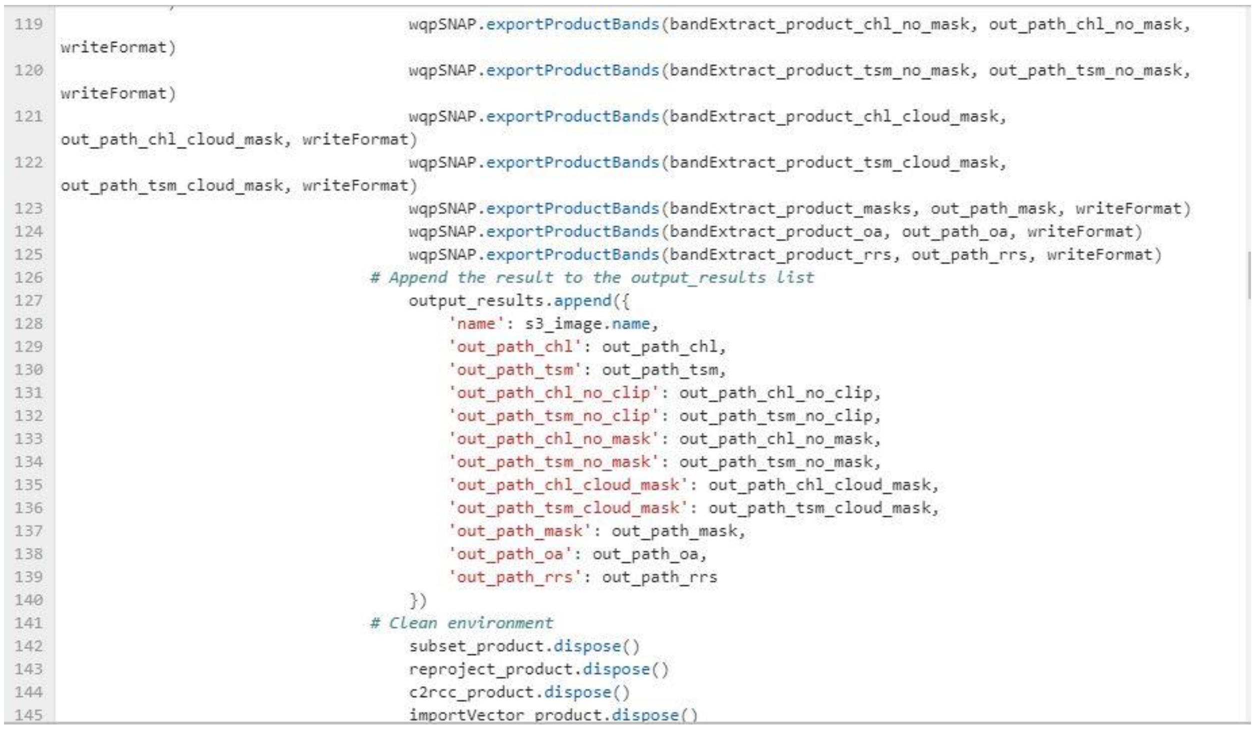
```
                        print(s3_image.name)
                        try:
                        # Check if image is within the date range
                            image_date_str = s3_image.name.split('_')[8]
                            image_date = datetime.strptime(image_date_str, '%Y%m%dT%H%M%S')

                            if start_date <= image_date <= end_date:



                                try:
                              # 2. Update the bounding box for the subset selection
                                    params_subset = s3_image.updateSNAPSubset(wqpParams.params_subset)
                                    subset_product = wqpSNAP.executeSNAPFunction(s3_image.product, params_subset)
                              # 3. Reproject the subset
                                    reproject_product = wqpSNAP.executeSNAPFunction(subset_product,
wqpParams.params_reproject)
                              # 4. Update the C2RCC temperature value. C2RCC wqp products
                                    params_C2RCC = s3_image.updateSNAPTemperature(df_t, wqpParams.params_C2RCC)
                                    c2rcc_product = wqpSNAP.executeSNAPFunction(reproject_product, params_C2RCC)
                              # 5. Import vector layer
                                    importVector_product = wqpSNAP.executeSNAPFunction(c2rcc_product,
wqpParams.params_importVector)
                              # 6. Band Maths operations
                                    bandMaths_product_C2RCC = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths)
                                    bandMaths_product_oa = wqpSNAP.executeSNAPFunction(reproject_product,
wqpParams.params_bandMaths_oa)
                                    bandMaths_product_rrs = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths_rrs)
                                    bandMaths_product_masks = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths_masks)
```



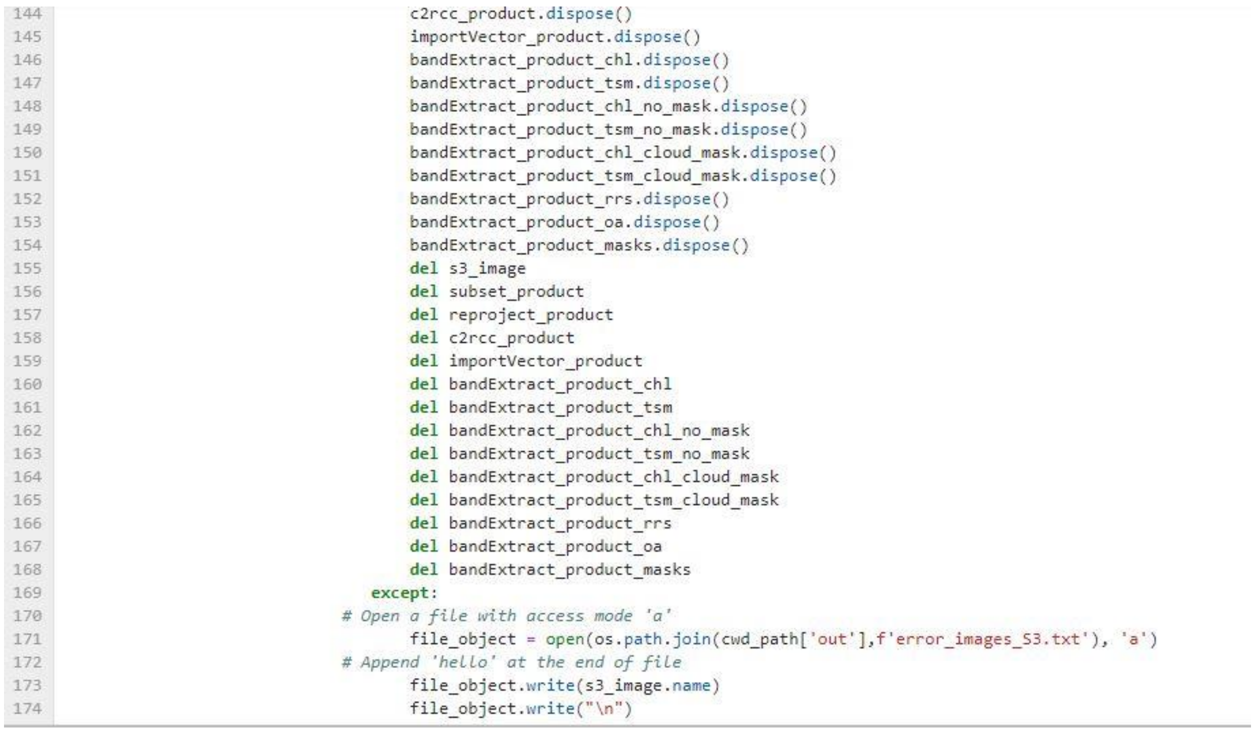
```
                                    bandMaths_product_masks = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths_masks)
                             # 7. Extract bands
                             # Product bands to be extracted
                                    bandExtract_product_chl = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC,
wqpParams.params_bandExtractor_chl)
                                    bandExtract_product_tsm = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC,
wqpParams.params_bandExtractor_tsm)
                                    bandExtract_product_chl_no_clip =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_no_clip)
                                    bandExtract_product_tsm_no_clip =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_no_clip)
                                    bandExtract_product_chl_no_mask =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_no_masks)
                                    bandExtract_product_tsm_no_mask =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_no_masks)
                                    bandExtract_product_chl_cloud_mask =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_cloud_mask)
                                    bandExtract_product_tsm_cloud_mask =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_cloud_mask)
                                    bandExtract_product_oa = wqpSNAP.executeSNAPFunction(bandMaths_product_oa,
wqpParams.params_bandExtractor_oa)
                             # bandExtract_product_oa = wqpSNAP.executeSNAPFunction(bandMaths_product_oa,
wqpParams.params_bandExtractor_oa_18)
                                    bandExtract_product_rrs = wqpSNAP.executeSNAPFunction(bandMaths_product_rrs,
wqpParams.params_bandExtractor_rrs)
                                    bandExtract_product_masks = wqpSNAP.executeSNAPFunction(bandMaths_product_masks,
wqpParams.params_bandExtractor_masks)
                               # 8. Save bands
                             # Define output paths
                                    sensorName = s3_image.name.split('_')[0]
```



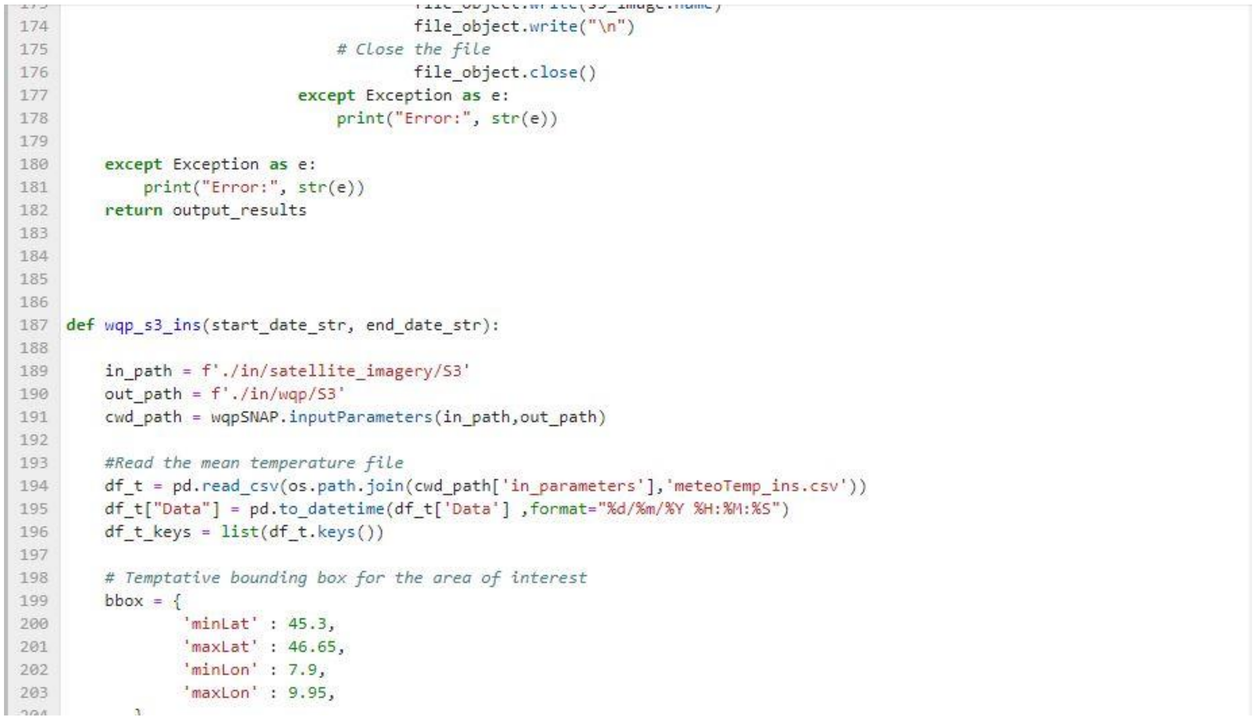
```
                             # Define output paths
                                    sensorName = s3_image.name.split('_')[0]
                                    sensorDate = s3_image.name.split('_')[8]
                                    out_path_chl =
os.path.join(cwd_path['out_wqp'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                                    out_path_tsm =
os.path.join(cwd_path['out_wqp'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                                    out_path_chl_no_clip =
os.path.join(cwd_path['out_wqp_no_clip'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                                    out_path_tsm_no_clip =
os.path.join(cwd_path['out_wqp_no_clip'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                                    out_path_chl_no_mask =
os.path.join(cwd_path['out_wqp_no_mask'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                                    out_path_tsm_no_mask =
os.path.join(cwd_path['out_wqp_no_mask'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                                    out_path_chl_cloud_mask =
os.path.join(cwd_path['out_wqp_cloud'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                                    out_path_tsm_cloud_mask =
os.path.join(cwd_path['out_wqp_cloud'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                                    out_path_mask =
os.path.join(cwd_path['out_masks'],sensorName+'_IT_'+sensorDate+'_L1')
                                    out_path_oa = os.path.join(cwd_path['out_oa'],sensorName+'_IT_'+sensorDate+'_L1')
                                    out_path_rrs = os.path.join(cwd_path['out_rrs'],sensorName+'_IT_'+sensorDate+'_L1')
                               # Save Bands
                                    wqpSNAP.exportProductBands(bandExtract_product_chl, out_path_chl, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_tsm, out_path_tsm, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_chl_no_clip, out_path_chl_no_clip,
writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_tsm_no_clip, out_path_tsm_no_clip,
writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_chl_no_mask, out_path_chl_no_mask,
```



```
                                    wqpSNAP.exportProductBands(bandExtract_product_chl_no_mask, out_path_chl_no_mask,
writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_tsm_no_mask, out_path_tsm_no_mask,
writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_chl_cloud_mask,
out_path_chl_cloud_mask, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_tsm_cloud_mask,
out_path_tsm_cloud_mask, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_masks, out_path_mask, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_oa, out_path_oa, writeFormat)
                                    wqpSNAP.exportProductBands(bandExtract_product_rrs, out_path_rrs, writeFormat)
                                # Append the result to the output_results list
                                    output_results.append({
                                        'name': s3_image.name,
                                        'out_path_chl': out_path_chl,
                                        'out_path_tsm': out_path_tsm,
                                        'out_path_chl_no_clip': out_path_chl_no_clip,
                                        'out_path_tsm_no_clip': out_path_tsm_no_clip,
                                        'out_path_chl_no_mask': out_path_chl_no_mask,
                                        'out_path_tsm_no_mask': out_path_tsm_no_mask,
                                        'out_path_chl_cloud_mask': out_path_chl_cloud_mask,
                                        'out_path_tsm_cloud_mask': out_path_tsm_cloud_mask,
                                        'out_path_mask': out_path_mask,
                                        'out_path_oa': out_path_oa,
                                        'out_path_rrs': out_path_rrs
                                    })
                                # Clean environment
                                    subset_product.dispose()
                                    reproject_product.dispose()
                                    c2rcc_product.dispose()
                                    importVector_product.dispose()
```

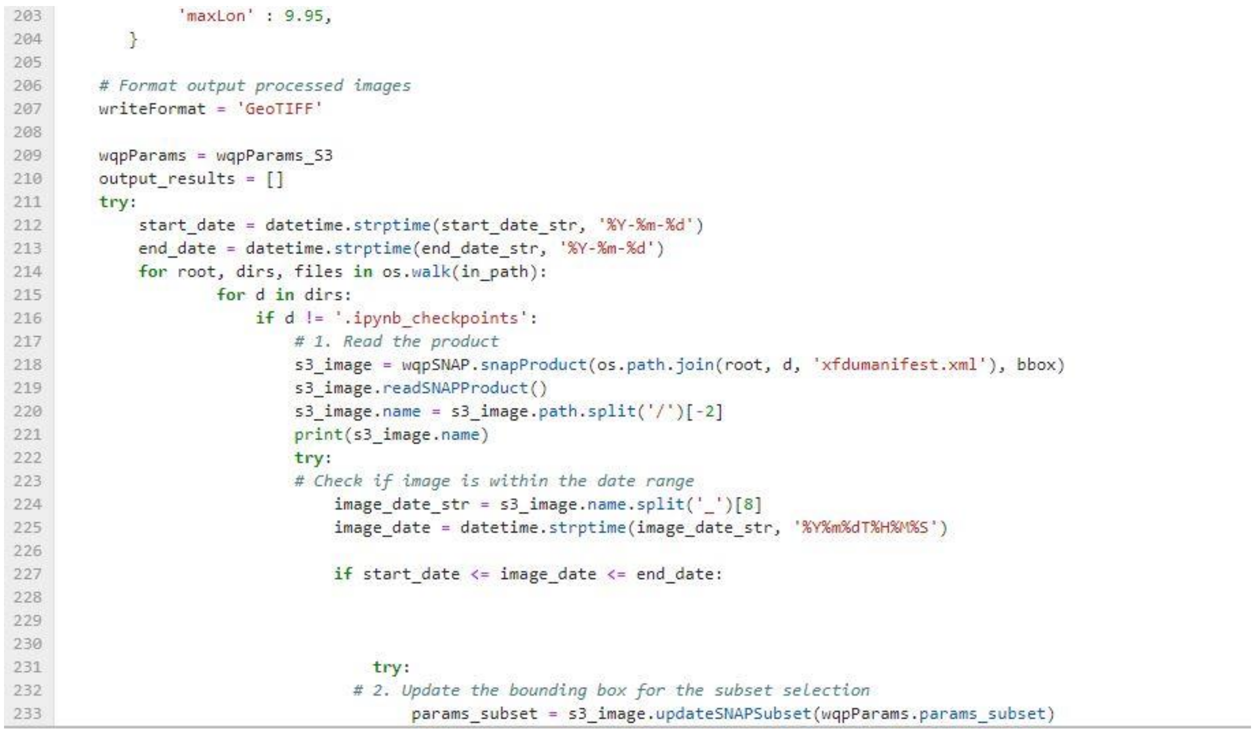


```python
                                    c2rcc_product.dispose()
                                    importVector_product.dispose()
                                    bandExtract_product_chl.dispose()
                                    bandExtract_product_tsm.dispose()
                                    bandExtract_product_chl_no_mask.dispose()
                                    bandExtract_product_tsm_no_mask.dispose()
                                    bandExtract_product_chl_cloud_mask.dispose()
                                    bandExtract_product_tsm_cloud_mask.dispose()
                                    bandExtract_product_rrs.dispose()
                                    bandExtract_product_oa.dispose()
                                    bandExtract_product_masks.dispose()
                                    del s3_image
                                    del subset_product
                                    del reproject_product
                                    del c2rcc_product
                                    del importVector_product
                                    del bandExtract_product_chl
                                    del bandExtract_product_tsm
                                    del bandExtract_product_chl_no_mask
                                    del bandExtract_product_tsm_no_mask
                                    del bandExtract_product_chl_cloud_mask
                                    del bandExtract_product_tsm_cloud_mask
                                    del bandExtract_product_rrs
                                    del bandExtract_product_oa
                                    del bandExtract_product_masks
                                except:
                            # Open a file with access mode 'a'
                                    file_object = open(os.path.join(cwd_path['out'],f'error_images_S3.txt'), 'a')
                            # Append 'hello' at the end of file
                                    file_object.write(s3_image.name)
                                    file_object.write("\n")
```

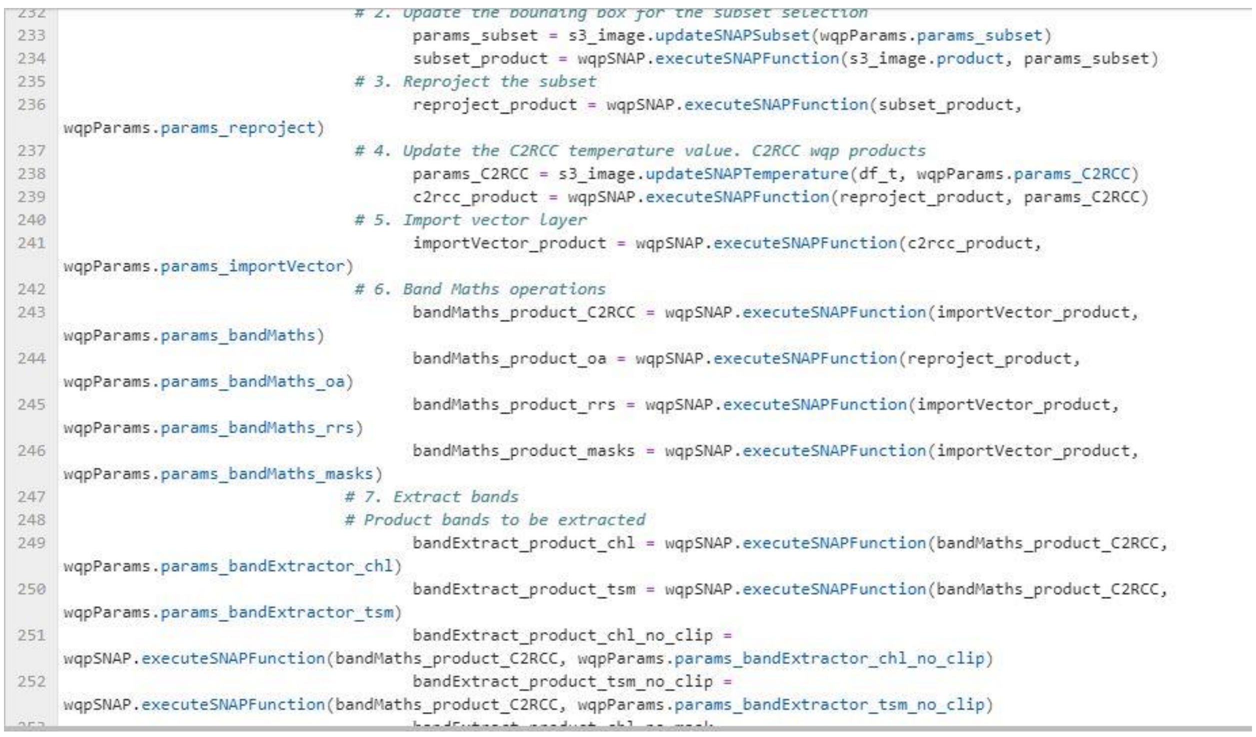


```python
                                    file_object.write("\n")
```

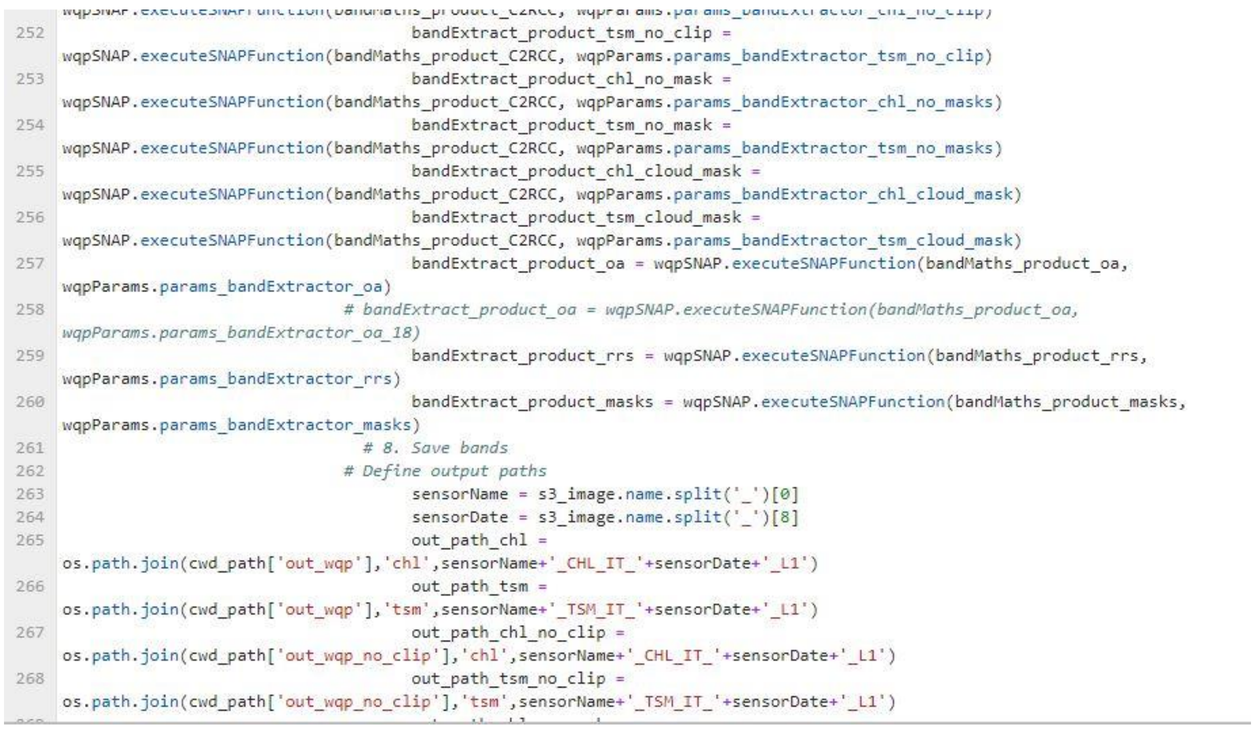
```python
                            # Close the file
                                    file_object.close()
                        except Exception as e:
                            print("Error:", str(e))

    except Exception as e:
        print("Error:", str(e))
    return output_results



def wqp_s3_ins(start_date_str, end_date_str):

    in_path = f'./in/satellite_imagery/S3'
    out_path = f'./in/wqp/S3'
    cwd_path = wqpSNAP.inputParameters(in_path,out_path)

    #Read the mean temperature file
    df_t = pd.read_csv(os.path.join(cwd_path['in_parameters'],'meteoTemp_ins.csv'))
    df_t["Data"] = pd.to_datetime(df_t['Data'] ,format="%d/%m/%Y %H:%M:%S")
    df_t_keys = list(df_t.keys())

    # Temptative bounding box for the area of interest
    bbox = {
            'minLat' : 45.3,
            'maxLat' : 46.65,
            'minLon' : 7.9,
            'maxLon' : 9.95,
        }
```



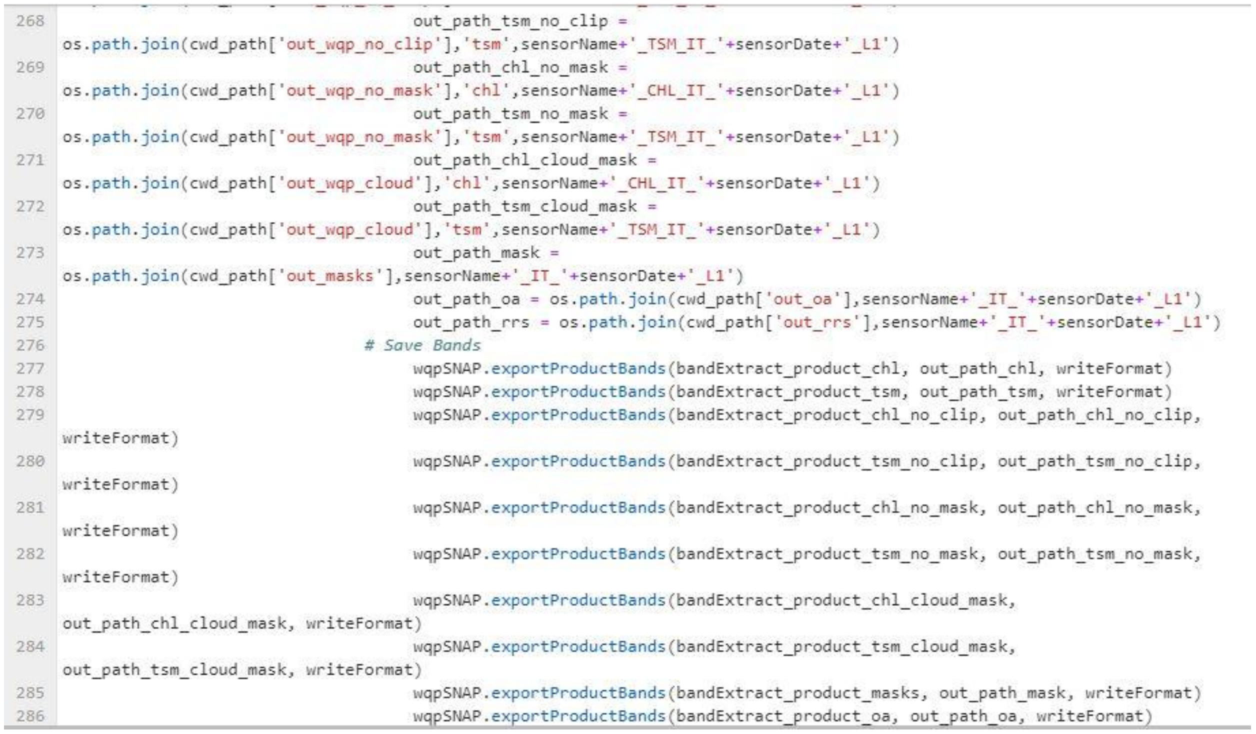
```
            'maxLon' : 9.95,
       }

    # Format output processed images
    writeFormat = 'GeoTIFF'

    wqpParams = wqpParams_S3
    output_results = []
    try:
        start_date = datetime.strptime(start_date_str, '%Y-%m-%d')
        end_date = datetime.strptime(end_date_str, '%Y-%m-%d')
        for root, dirs, files in os.walk(in_path):
                for d in dirs:
                    if d != '.ipynb_checkpoints':
                        # 1. Read the product
                        s3_image = wqpSNAP.snapProduct(os.path.join(root, d, 'xfdumanifest.xml'), bbox)
                        s3_image.readSNAPProduct()
                        s3_image.name = s3_image.path.split('/')[-2]
                        print(s3_image.name)
                        try:
                        # Check if image is within the date range
                            image_date_str = s3_image.name.split('_')[8]
                            image_date = datetime.strptime(image_date_str, '%Y%m%dT%H%M%S')

                            if start_date <= image_date <= end_date:



                                try:
                              # 2. Update the bounding box for the subset selection
                                    params_subset = s3_image.updateSNAPSubset(wqpParams.params_subset)
```



```
                              # 2. Update the bounding box for the subset selection
                                    params_subset = s3_image.updateSNAPSubset(wqpParams.params_subset)
                                    subset_product = wqpSNAP.executeSNAPFunction(s3_image.product, params_subset)
                              # 3. Reproject the subset
                                    reproject_product = wqpSNAP.executeSNAPFunction(subset_product,
wqpParams.params_reproject)
                              # 4. Update the C2RCC temperature value. C2RCC wqp products
                                    params_C2RCC = s3_image.updateSNAPTemperature(df_t, wqpParams.params_C2RCC)
                                    c2rcc_product = wqpSNAP.executeSNAPFunction(reproject_product, params_C2RCC)
                              # 5. Import vector layer
                                    importVector_product = wqpSNAP.executeSNAPFunction(c2rcc_product,
wqpParams.params_importVector)
                              # 6. Band Maths operations
                                    bandMaths_product_C2RCC = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths)
                                    bandMaths_product_oa = wqpSNAP.executeSNAPFunction(reproject_product,
wqpParams.params_bandMaths_oa)
                                    bandMaths_product_rrs = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths_rrs)
                                    bandMaths_product_masks = wqpSNAP.executeSNAPFunction(importVector_product,
wqpParams.params_bandMaths_masks)
                             # 7. Extract bands
                             # Product bands to be extracted
                                    bandExtract_product_chl = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC,
wqpParams.params_bandExtractor_chl)
                                    bandExtract_product_tsm = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC,
wqpParams.params_bandExtractor_tsm)
                                    bandExtract_product_chl_no_clip =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_no_clip)
                                    bandExtract_product_tsm_no_clip =
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_no_clip)
```

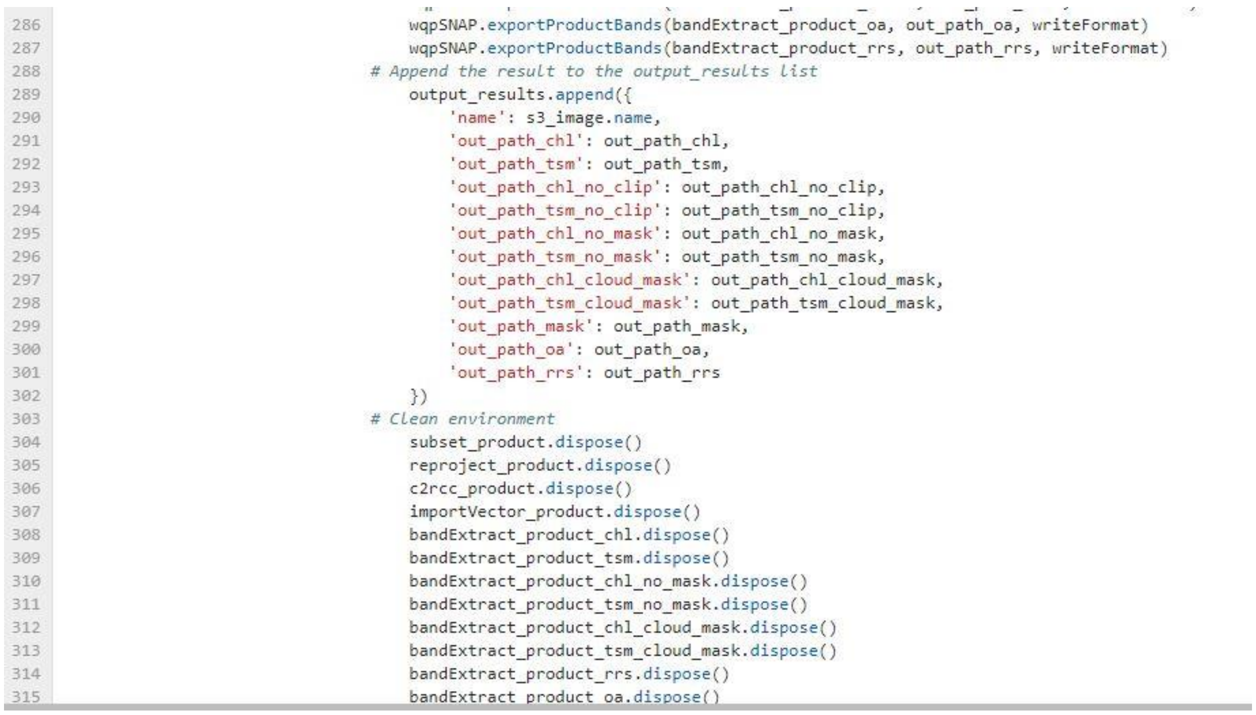


```
wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_no_clip)
                bandExtract_product_tsm_no_clip = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_no_clip)
                bandExtract_product_chl_no_mask = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_no_masks)
                bandExtract_product_tsm_no_mask = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_no_masks)
                bandExtract_product_chl_cloud_mask = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_chl_cloud_mask)
                bandExtract_product_tsm_cloud_mask = wqpSNAP.executeSNAPFunction(bandMaths_product_C2RCC, wqpParams.params_bandExtractor_tsm_cloud_mask)
                bandExtract_product_oa = wqpSNAP.executeSNAPFunction(bandMaths_product_oa, wqpParams.params_bandExtractor_oa)
         # bandExtract_product_oa = wqpSNAP.executeSNAPFunction(bandMaths_product_oa, wqpParams.params_bandExtractor_oa_18)
                bandExtract_product_rrs = wqpSNAP.executeSNAPFunction(bandMaths_product_rrs, wqpParams.params_bandExtractor_rrs)
                bandExtract_product_masks = wqpSNAP.executeSNAPFunction(bandMaths_product_masks, wqpParams.params_bandExtractor_masks)
           # 8. Save bands
         # Define output paths
                sensorName = s3_image.name.split('_')[0]
                sensorDate = s3_image.name.split('_')[8]
                out_path_chl = os.path.join(cwd_path['out_wqp'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                out_path_tsm = os.path.join(cwd_path['out_wqp'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                out_path_chl_no_clip = os.path.join(cwd_path['out_wqp_no_clip'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                out_path_tsm_no_clip = os.path.join(cwd_path['out_wqp_no_clip'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
```

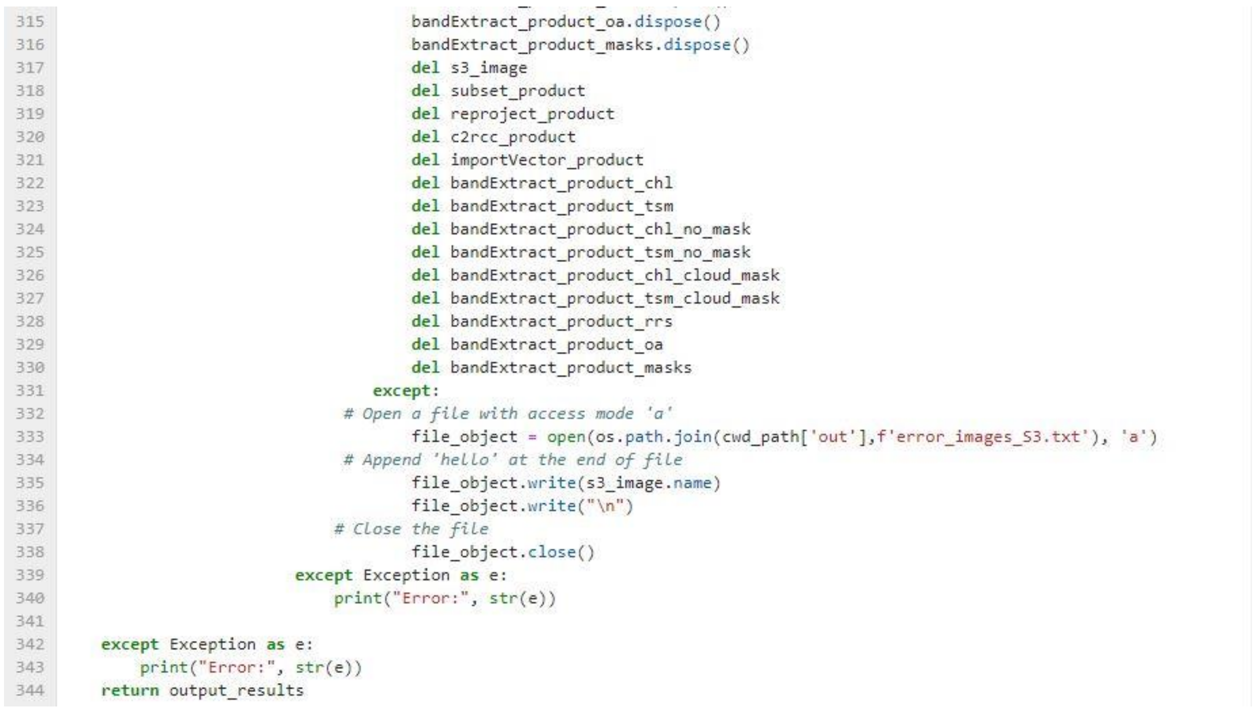


```
                out_path_tsm_no_clip = os.path.join(cwd_path['out_wqp_no_clip'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
```

```
                out_path_chl_no_mask = os.path.join(cwd_path['out_wqp_no_mask'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                out_path_tsm_no_mask = os.path.join(cwd_path['out_wqp_no_mask'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                out_path_chl_cloud_mask = os.path.join(cwd_path['out_wqp_cloud'],'chl',sensorName+'_CHL_IT_'+sensorDate+'_L1')
                out_path_tsm_cloud_mask = os.path.join(cwd_path['out_wqp_cloud'],'tsm',sensorName+'_TSM_IT_'+sensorDate+'_L1')
                out_path_mask = os.path.join(cwd_path['out_masks'],sensorName+'_IT_'+sensorDate+'_L1')
                out_path_oa = os.path.join(cwd_path['out_oa'],sensorName+'_IT_'+sensorDate+'_L1')
                out_path_rrs = os.path.join(cwd_path['out_rrs'],sensorName+'_IT_'+sensorDate+'_L1')
           # Save Bands
                wqpSNAP.exportProductBands(bandExtract_product_chl, out_path_chl, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_tsm, out_path_tsm, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_chl_no_clip, out_path_chl_no_clip, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_tsm_no_clip, out_path_tsm_no_clip, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_chl_no_mask, out_path_chl_no_mask, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_tsm_no_mask, out_path_tsm_no_mask, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_chl_cloud_mask, out_path_chl_cloud_mask, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_tsm_cloud_mask, out_path_tsm_cloud_mask, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_masks, out_path_mask, writeFormat)
                wqpSNAP.exportProductBands(bandExtract_product_oa, out_path_oa, writeFormat)
```



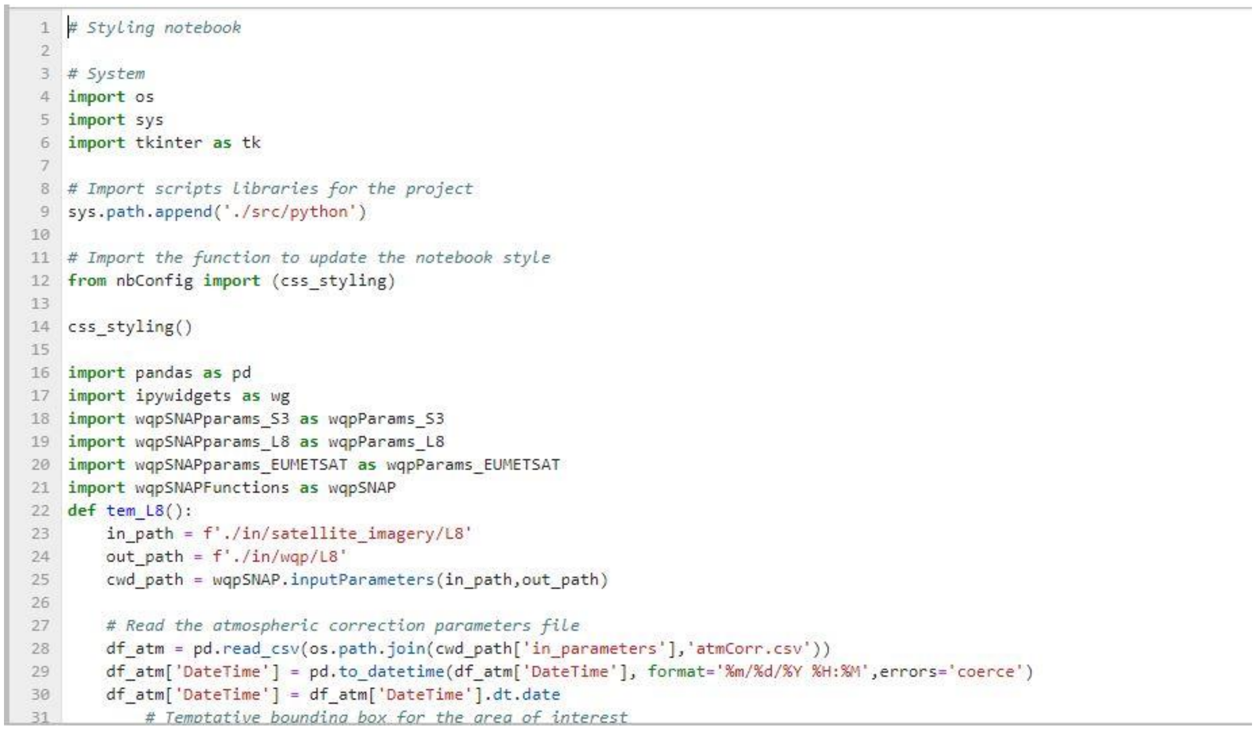
```python
                                wqpSNAP.exportProductBands(bandExtract_product_oa, out_path_oa, writeFormat)
                                wqpSNAP.exportProductBands(bandExtract_product_rrs, out_path_rrs, writeFormat)
                            # Append the result to the output_results list
                                output_results.append({
                                    'name': s3_image.name,
                                    'out_path_chl': out_path_chl,
                                    'out_path_tsm': out_path_tsm,
                                    'out_path_chl_no_clip': out_path_chl_no_clip,
                                    'out_path_tsm_no_clip': out_path_tsm_no_clip,
                                    'out_path_chl_no_mask': out_path_chl_no_mask,
                                    'out_path_tsm_no_mask': out_path_tsm_no_mask,
                                    'out_path_chl_cloud_mask': out_path_chl_cloud_mask,
                                    'out_path_tsm_cloud_mask': out_path_tsm_cloud_mask,
                                    'out_path_mask': out_path_mask,
                                    'out_path_oa': out_path_oa,
                                    'out_path_rrs': out_path_rrs
                                })
                            # Clean environment
                                subset_product.dispose()
                                reproject_product.dispose()
                                c2rcc_product.dispose()
                                importVector_product.dispose()
                                bandExtract_product_chl.dispose()
                                bandExtract_product_tsm.dispose()
                                bandExtract_product_chl_no_mask.dispose()
                                bandExtract_product_tsm_no_mask.dispose()
                                bandExtract_product_chl_cloud_mask.dispose()
                                bandExtract_product_tsm_cloud_mask.dispose()
                                bandExtract_product_rrs.dispose()
                                bandExtract_product_oa.dispose()
```



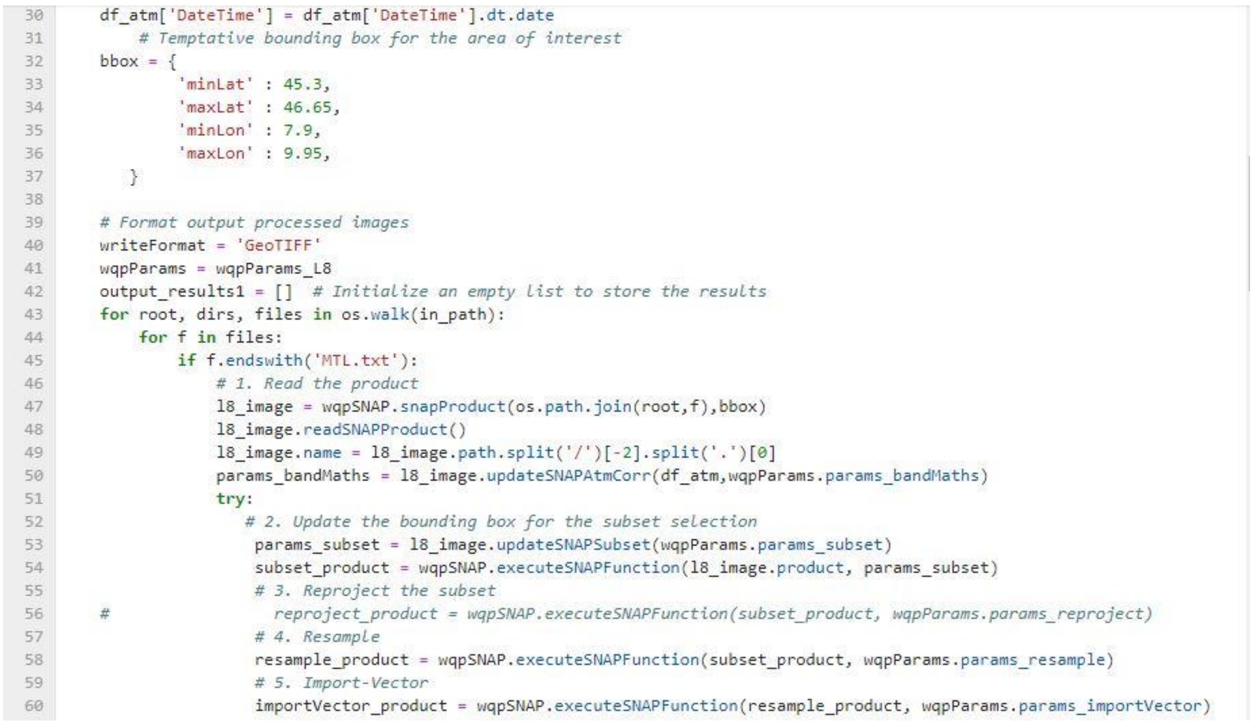
```python
                                bandExtract_product_oa.dispose()
                                bandExtract_product_masks.dispose()
                                del s3_image
                                del subset_product
                                del reproject_product
                                del c2rcc_product
                                del importVector_product
                                del bandExtract_product_chl
                                del bandExtract_product_tsm
                                del bandExtract_product_chl_no_mask
                                del bandExtract_product_tsm_no_mask
                                del bandExtract_product_chl_cloud_mask
                                del bandExtract_product_tsm_cloud_mask
                                del bandExtract_product_rrs
                                del bandExtract_product_oa
                                del bandExtract_product_masks
                            except:
                         # Open a file with access mode 'a'
                                file_object = open(os.path.join(cwd_path['out'],f'error_images_S3.txt'), 'a')
                         # Append 'hello' at the end of file
                                file_object.write(s3_image.name)
                                file_object.write("\n")
                        # Close the file
                                file_object.close()
                    except Exception as e:
                        print("Error:", str(e))

except Exception as e:
    print("Error:", str(e))
return output_results
```

Listing 4. Function that used to generate CHL-a and TSM Maps

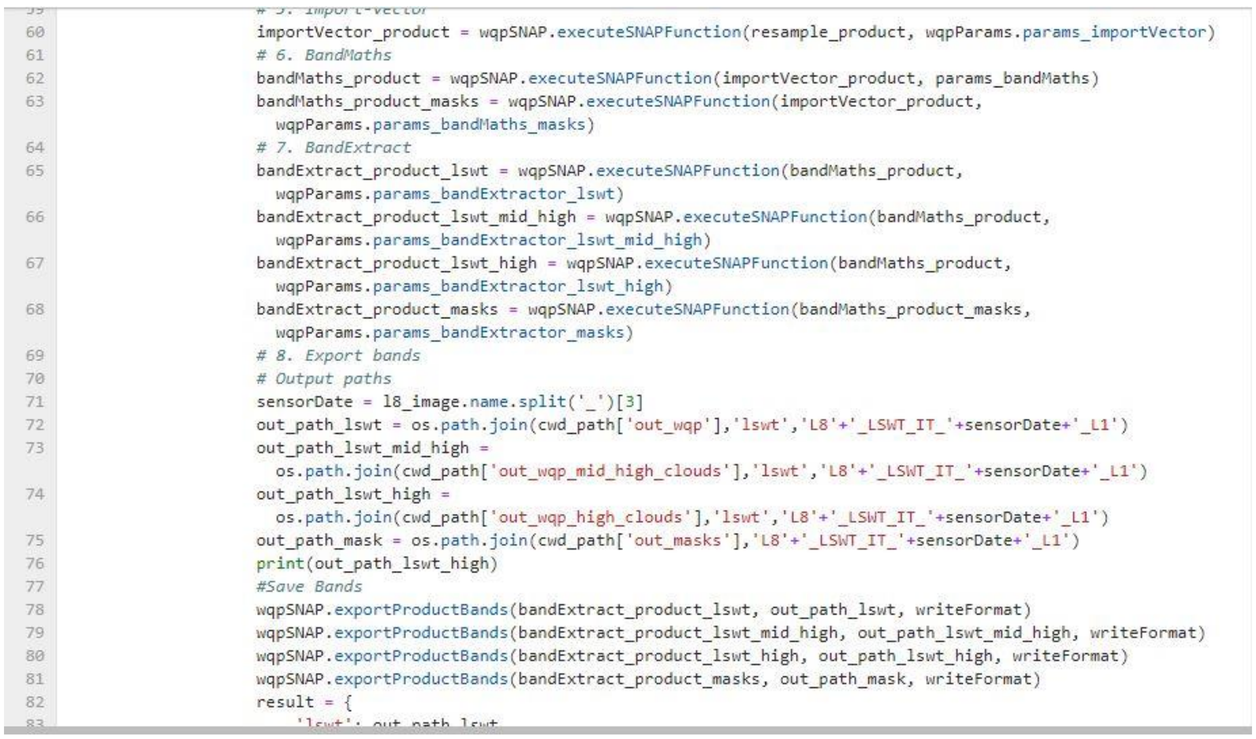
Production of LWST



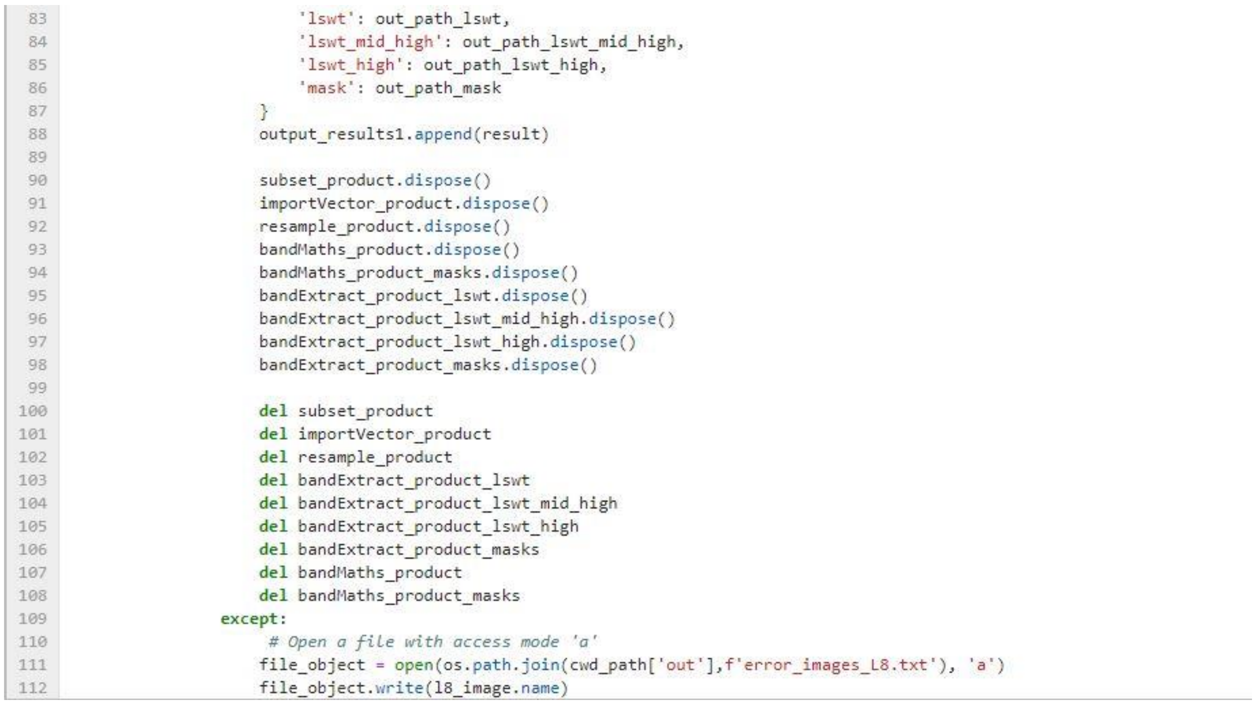
```python
# Styling notebook

# System
import os
import sys
import tkinter as tk

# Import scripts libraries for the project
sys.path.append('./src/python')

# Import the function to update the notebook style
from nbConfig import (css_styling)

css_styling()

import pandas as pd
import ipywidgets as wg
import wqpSNAPparams_S3 as wqpParams_S3
import wqpSNAPparams_L8 as wqpParams_L8
import wqpSNAPparams_EUMETSAT as wqpParams_EUMETSAT
import wqpSNAPFunctions as wqpSNAP
def tem_L8():
    in_path = f'./in/satellite_imagery/L8'
    out_path = f'./in/wqp/L8'
    cwd_path = wqpSNAP.inputParameters(in_path,out_path)

    # Read the atmospheric correction parameters file
    df_atm = pd.read_csv(os.path.join(cwd_path['in_parameters'],'atmCorr.csv'))
    df_atm['DateTime'] = pd.to_datetime(df_atm['DateTime'], format='%m/%d/%Y %H:%M',errors='coerce')
    df_atm['DateTime'] = df_atm['DateTime'].dt.date
        # Temptative bounding box for the area of interest
```



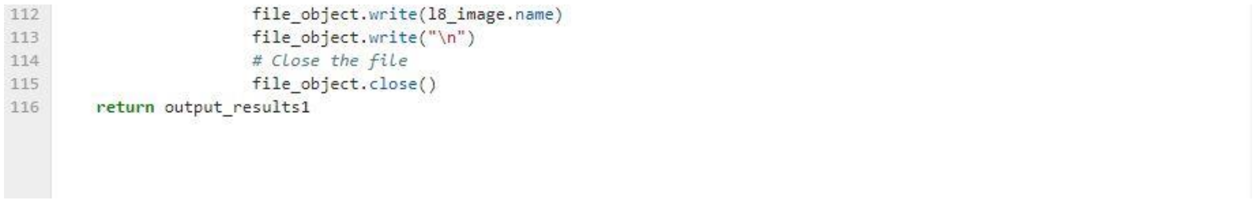
```python
    df_atm['DateTime'] = df_atm['DateTime'].dt.date
        # Temptative bounding box for the area of interest
    bbox = {
            'minLat' : 45.3,
            'maxLat' : 46.65,
            'minLon' : 7.9,
            'maxLon' : 9.95,
       }

    # Format output processed images
    writeFormat = 'GeoTIFF'
    wqpParams = wqpParams_L8
    output_results1 = []  # Initialize an empty list to store the results
    for root, dirs, files in os.walk(in_path):
        for f in files:
            if f.endswith('MTL.txt'):
                # 1. Read the product
                l8_image = wqpSNAP.snapProduct(os.path.join(root,f),bbox)
                l8_image.readSNAPProduct()
                l8_image.name = l8_image.path.split('/')[-2].split('.')[0]
                params_bandMaths = l8_image.updateSNAPAtmCorr(df_atm,wqpParams.params_bandMaths)
                try:
                   # 2. Update the bounding box for the subset selection
                    params_subset = l8_image.updateSNAPSubset(wqpParams.params_subset)
                    subset_product = wqpSNAP.executeSNAPFunction(l8_image.product, params_subset)
                    # 3. Reproject the subset
    #                 reproject_product = wqpSNAP.executeSNAPFunction(subset_product, wqpParams.params_reproject)
                    # 4. Resample
                    resample_product = wqpSNAP.executeSNAPFunction(subset_product, wqpParams.params_resample)
                    # 5. Import-Vector
                    importVector_product = wqpSNAP.executeSNAPFunction(resample_product, wqpParams.params_importVector)
```



```
                    importVector_product = wqpSNAP.executeSNAPFunction(resample_product, wqpParams.params_importVector)
                    # 6. BandMaths
                    bandMaths_product = wqpSNAP.executeSNAPFunction(importVector_product, params_bandMaths)
                    bandMaths_product_masks = wqpSNAP.executeSNAPFunction(importVector_product,
                      wqpParams.params_bandMaths_masks)
                    # 7. BandExtract
                    bandExtract_product_lswt = wqpSNAP.executeSNAPFunction(bandMaths_product,
                      wqpParams.params_bandExtractor_lswt)
                    bandExtract_product_lswt_mid_high = wqpSNAP.executeSNAPFunction(bandMaths_product,
                      wqpParams.params_bandExtractor_lswt_mid_high)
                    bandExtract_product_lswt_high = wqpSNAP.executeSNAPFunction(bandMaths_product,
                      wqpParams.params_bandExtractor_lswt_high)
                    bandExtract_product_masks = wqpSNAP.executeSNAPFunction(bandMaths_product_masks,
                      wqpParams.params_bandExtractor_masks)
                    # 8. Export bands
                    # Output paths
                    sensorDate = l8_image.name.split('_')[3]
                    out_path_lswt = os.path.join(cwd_path['out_wqp'],'lswt','L8'+'_LSWT_IT_'+sensorDate+'_L1')
                    out_path_lswt_mid_high =
                      os.path.join(cwd_path['out_wqp_mid_high_clouds'],'lswt','L8'+'_LSWT_IT_'+sensorDate+'_L1')
                    out_path_lswt_high =
                      os.path.join(cwd_path['out_wqp_high_clouds'],'lswt','L8'+'_LSWT_IT_'+sensorDate+'_L1')
                    out_path_mask = os.path.join(cwd_path['out_masks'],'L8'+'_LSWT_IT_'+sensorDate+'_L1')
                    print(out_path_lswt_high)
                    #Save Bands
                    wqpSNAP.exportProductBands(bandExtract_product_lswt, out_path_lswt, writeFormat)
                    wqpSNAP.exportProductBands(bandExtract_product_lswt_mid_high, out_path_lswt_mid_high, writeFormat)
                    wqpSNAP.exportProductBands(bandExtract_product_lswt_high, out_path_lswt_high, writeFormat)
                    wqpSNAP.exportProductBands(bandExtract_product_masks, out_path_mask, writeFormat)
                    result = {
```



```
                        'lswt': out_path_lswt,
                        'lswt_mid_high': out_path_lswt_mid_high,
                        'lswt_high': out_path_lswt_high,
                        'mask': out_path_mask
                    }
                    output_results1.append(result)

                    subset_product.dispose()
                    importVector_product.dispose()
                    resample_product.dispose()
                    bandMaths_product.dispose()
                    bandMaths_product_masks.dispose()
                    bandExtract_product_lswt.dispose()
                    bandExtract_product_lswt_mid_high.dispose()
                    bandExtract_product_lswt_high.dispose()
                    bandExtract_product_masks.dispose()

                    del subset_product
                    del importVector_product
                    del resample_product
                    del bandExtract_product_lswt
                    del bandExtract_product_lswt_mid_high
                    del bandExtract_product_lswt_high
                    del bandExtract_product_masks
                    del bandMaths_product
                    del bandMaths_product_masks
                except:
                     # Open a file with access mode 'a'
                    file_object = open(os.path.join(cwd_path['out'],f'error_images_L8.txt'), 'a')
                    file_object.write(l8_image.name)
```



```
                    file_object.write(l8_image.name)
                    file_object.write("\n")
                    # Close the file
                    file_object.close()
    return output_results1
```

Listing 5. Function that used to generate LWST Maps

Acknowledgements

We would like to express our deepest gratitude to Prof. Maria Antonia Brovelli and Dr. Lorenzo Amici for their invaluable guidance, mentorship, and support throughout our journey. Their expertise and dedication have been instrumental in shaping our understanding and pushing us to achieve our best. Prof. Maria Antonia Brovelli's profound knowledge and insights in the field have expanded our horizons, and we are truly grateful for the opportunity to learn from her. Dr. Lorenzo Amici's constant encouragement and feedback have played a vital role in our growth, and we appreciate his unwavering support.

In addition, we extend our heartfelt appreciation to our families and friends who have stood by us every step of the way. Their unwavering love, encouragement, and understanding have been a source of strength and motivation. We are grateful for their belief in us, their patience during challenging times, and their constant support. Their presence in our lives has made this journey all the more meaningful and memorable. Their unwavering support has been the foundation upon which we have built our success, and we are forever grateful for their presence in our lives.