Università degli Studi Mediterranea di Reggio Calabria
Dipartimento di Agraria

Corso di Dottorato di ricerca
Scienze Agrarie Alimentari e Forestali XXXIII ciclo

SSD AGR/10 – Costruzioni rurali e territorio agroforestale

Web-Based Spatial Decision Support System for Precision Agriculture: a Support Tool for delineating Dynamic Management Unit Zones.

PhD Thesis

Candidato (PhD Candidate):

Simone Luca Leonardo Lanucara

Coordinatore del Corso di Dottorato (Coordinator of PhD Course): **Prof. Marco Poiana**

Tutor (Supervisor): **Prof. Giuseppe Modica**

Co-Tutor (Co-Supervisor): **Dr. Alessandro Oggioni**

Anno Accademico 2019/2020

Index

Abstract

Precision agriculture (PA) takes advantage of digital technologies to collect and analyze a large amount of data collected in the field or from external sources (e.g. Earth Observation data, Weather data, Open data) for use in site-specific crop management. The main issues in the implementation and dissemination of PA include the harmonization of data from different sources, the interpretation of the huge amount of data collected, the understanding of the causes of variability and the ability to propose robust strategies for the management of inter- and intra-field variability. Different approaches propose Service Oriented Architectures (SOA) to ease integration of heterogeneous and distributed sources for PA; as implemented in other domains (geologic, oceanographic, and ecological), SOA enable web sharing of data and related metadata. Further development of PA requires technological facilities to process data efficiently, and to translate the data to better decisions and actions in a field. While the SOA principle is mainly "find and bind", in PA also analysis is needed. Web-based decision support systems can provide users with tools that are simple to use while at the same time addresses the factors mentioned above.

Management Unit Zones (MUZ) are the sub-division of fields featuring an inter-zonal variation, delineated by agronomists for effective PA operations (till, sow, fertilize, harvest) in the fields. To develop MUZ, normally three factors need to be considered: multi-dimensional data to be used as a basis for creating zones, procedure to be used to process the information, and the optimal number of zones that a field should be divided into. Efficient and easy-to-use tools that address all these factors are needed to provide a technology delivery mechanism, the lack of which has been identified as the major obstacle to the wide adoption of PA.

The aim of this study was to create a Web-Based Spatial Decision Support System (WBSD) for delineating Dynamic Management Unit Zones. The WBSD is based, from a technological perspective, on a SOA developed by Free and Open Source Software (FOSS) and international Open Geospatial Consortium (OGC) standards for data sharing. The system is tested on a case study of 500 hectares of durum wheat crops sown in November 2018 and harvested in June 2019 in Italy.

Riassunto

L'agricoltura di precisione (PA) sfrutta le tecnologie digitali per raccogliere e analizzare una grande quantità di dati raccolti sul campo o da fonti esterne (ad esempio, dati di osservazione della terra, dati meteorologici, dati aperti), tali dati vengono utilizzati per la gestione sito-specifica delle colture. Le difficoltà principali nell'utilizzo e diffusione della PA includono l'armonizzazione dei dati provenienti da fonti diverse, l'interpretazione dell'enorme quantità di dati raccolti, la comprensione delle cause della variabilità e la capacità di proporre strategie robuste per la gestione della variabilità inter e intra-campo. Diversi approcci tecnologici propongono Architetture a Servizi (SOA) per facilitare l'integrazione di fonti eterogenee e distribuite per la PA; come implementato in altri domini (geologico, oceanografico, ecologico), le SOA consentono la condivisione nel web dei dati e dei relativi metadati. Ulteriori sviluppi dalla PA richiedono infrastrutture tecnologiche per elaborare i dati in modo efficiente e per convertire i dati in decisioni ed azioni sui campi. Mentre il principio SOA è principalmente "trova e lega", in PA è necessaria anche l'analisi. I sistemi di supporto decisionale basati sul web possono fornire agli utenti strumenti semplici da usare e allo stesso tempo affrontare i fattori sopra menzionati.

Le Zone Gestionali Omogenee (MUZ) sono suddivisioni dei campi, con una variazione inter-zonale, tracciate dagli agronomi per effettuare le operazioni agronomiche di PA (coltivazione, semina, concimazione, raccolta) nei campi. Per sviluppare le MUZ, normalmente occorre considerare tre fattori: i dati multidimensionali da utilizzare come base per la creazione delle zone, la procedura da utilizzare per elaborare le informazioni e il numero ottimale di zone in cui un campo deve essere suddiviso. Strumenti efficienti e facili da usare che affrontino tutti questi fattori sono necessari per fornire un meccanismo di abilitazione tecnologica, la cui mancanza è stata identificata come il principale ostacolo all'adozione su larga scala della PA.

Scopo di questo studio è stato quello di sviluppare un Sistema di Supporto alle Decisioni Spaziali basato sul Web (WBSD) per delineare le zone di unità di gestione dinamica. Il WBSD si basa, da un punto di vista tecnologico, su un SOA sviluppato attraverso l'utilizzo di Software Gratuito e con Codice Sorgente Aperto (FOSS) e l'adozione di standard internazionali dell'Open Geospatial Consortium (OGC) per la condivisione dei dati. Il sistema è testato su un caso di studio di 500 ettari di colture di grano duro seminate a novembre 2018 e raccolte a giugno 2019 in Italia.

Keywords

Precision Agriculture; Service Oriented Architecture; Durum Wheat, Earth Observation, dynamic Management Unit Zones (dMUZ).

Organization

The first chapter of this paper is given an overview of PA and ICT technologies. In section 1.1, after a general introduction is analyzed, the issues in the diffusion of AP, the possible solutions thanks to the use of ICT technologies, and, finally, this research's objectives are explained. Section 1.2 provides an overview of the PA. Sections 1.3 and 1.4 describe the space-time variability in agricultural production and MUZ, respectively. In section 1.5, the monitoring methodologies of agricultural production systems are analyzed: environmental, crop, operational. In section 1.6, an overview of ICT systems for agriculture is made. A special look is given to Free and Open Source solutions and technologies, SOA architectures, and interoperability obtained thanks to web services and international standards also adopted for the agricultural sector.

In Chapter 2, we describe a case study for the development and implementation of SOA for PA and the delineation of MUZ on 512 hectares of durum wheat in Italy. The study area is identified and described in section 2.1, and base data acquisition is described in section 2.2. Sections 2.3 and 2.4 describe the SOA and its implementation. In section 2.5, the methodologies of base data processing and MUZ delineation are described.

The 3rd chapter is wholly dedicated to presenting the results concerning the development of the SOA, the sharing of the data, and the analysis methodologies for the delineation of the MUZ.

The fourth and last chapter discusses the results of the research, the research perspectives, the possible evolutions taking into account the needs of end-users, the evolution of technologies in PA, and the development of analytical methodologies in a perspective of environmental sustainability of agriculture.

1 Introduction

1.1 General Introduction and aims of the research work

With an expected global population of over nine billion, food production represents a significant challenge that will be further exacerbated by climate change, reduced water supply, and the environmental impacts of intensive plant and livestock production. The Food and Agriculture Organization of the United Nations (FAO) recommends adopting digital technologies to increase productivity and address food security risk (FAO, 2017). Besides, FAO indicates that if each region of the world could maximize its efficiency in agricultural land management practices, there should be adequate food supplies to support the world's population. This is in line with the "Zero Hunger Goal" of the United Nations Agenda 2030, which aims to: 1) ensure sustainable food production systems, 2) implement resilient agricultural practices that increase productivity, and 3) improve the agricultural productivity of small-scale producers.

To address the problems and objectives mentioned above, precision agriculture (PA) can be a solution. PA is a relatively new management concept introduced in the mid-1980s, which not only attracted considerable interest but also started a revolution in resource management (Robert, 1999) by addressing the following three key issues:

- agronomic, improve the effectiveness of inputs concerning yield;
- economic, increase productivity and competitiveness through more efficient practices;
- environmental, reduce the ecological impact of agriculture by optimizing the use of inputs.

The relationship between agriculture and the environment must consider the biophysical nature of agricultural production processes and their close dependence on local ecosystems' characteristics. Therefore, investigating the effects of agricultural activity on the environment requires a series of detailed information regarding the use of inputs, land use, and agricultural management practices.

Implementing management strategies for sustainable agriculture requires a multidisciplinary approach and a synergy between farmers, technical assistance services, industry, and researchers to transfer research and experimentation results in different business realities. The complexity and the lack of information flow between scientific, professional and policymakers increase the difficulties instead of focusing on evidence-based policies.

Therefore, this work's purpose was to create an environment of shared knowledge between the different actors operating in the PA sector through the development of a Free and Open Source Software Web Platform. This proposed web platform can support advanced research in the PA sector, promote the transfer of research know-how to operational applications and finally make the information available directly in the field for immediate operational applications and/or decision-making support.

The rapid and intense transformation of information, both in data acquisition technologies and in the methods of access and sharing, offers exceptional opportunities for the agricultural sector to improve the quality of production, the environmental sustainability of agriculture, and the effectiveness in the use of natural resources. The new technologies developed for the agricultural sector, in conjunction with the rapid evolution of Information Communication Technologies (ICT), Geographic Information Systems (GIS), and Geospatial Technologies (GT), now offer enormous potential for the development and optimization of information sharing solutions to support PA. ICT and GT are now mandatory to implement PA as they can manage the high amount of data, the intensity of knowledge, and the spatial and temporal aspects related to the management of agricultural practices.

Although PA techniques and equipment are increasing, the adoption rate is now slower than in the mid and late 1990s. There is evidence that further expansion of PA is lagging for several reasons (Griffin et al., 2004), many of which are educational in nature or are related to difficulties in dealing with technology. These authors highlighted how *"time to learn equipment and software, lack of electronic skills, lack of training for manufacturers and industry, link between data collection and decision-making, lack of technical assistance, lack of local experts, work with data of different formats, difficulties in maintaining data quality, basic research on yield and soil relations, and need for equipment, techniques, software products.*" (Griffin et al., 2004).

Among the several issues that must be overcome before PA technologies can be widely and rapidly implemented, one is the management of the massive data flow required for PA (Zhang et al., 2002): a) a large amount of data can be acquired electronically in the field by sensors; b) a large amount of data can be acquired by satellites; c) variable rate applications (VRA) of fertilizers, herbicides, and other agricultural chemicals can be practiced. Although there is much space for improvement in existing electronic technology for data acquisition and VRA, these activities do not seem to be the main bottleneck at the moment. Instead, the

main problems seem to be: a) the harmonization of data from different sources; b) the interpretation of the massive amount of data; c) the understanding of the causes of variability; d) the possibility to propose robust strategies for the management of intra-field variability.

Several authors (Saraiva et al., 1997; Lutticken, 2000; Fountas et al., 2002; Sørensen et al., 2002; Backes et al., 2003; Korduan, 2003; Pedersen et al., 2003; Adrian et al., 2005) addressed the issue of the most critical technological requirements for the diffusion of PA. They include the following:

- Data management systems and decision support systems should be designed to meet the specific needs of farmers.
- Systems should have a simple user interface that allows customization to different user profiles.
- Automated and easy-to-use methods are required for data processing. Systems should allow for the inclusion and programming of new automated methods based on user-defined rules.
- The user should have complete control, when they wish, with access to parameters for processing and analysis functions.
- The introduction of specialist knowledge (e.g., rule-based knowledge) should be possible. This could offer the opportunity to fine-tune systems according to local conditions and to include user skills, practices, and preferences (such as risk profile).
- More integrated and better-standardized information systems are needed. This could reduce technical investment, the learning curve and the need for technical support.
- Support for integration and interoperability with different software packages (including simulation packages), data sources (such as meteorological data, market data), locally or remotely via the Internet, using open standards, interfaces and data protocols. This is particularly important for both legacy and distributed systems.

A single proprietary system is unlikely to meet all these requirements due to its complexity and completeness. That is why an open software platform is a more appropriate solution to the problem (Saraiva et al., 1998; Lutticken, 2000; Murakami et al., 2002). An important point that favors an open, well-structured, component-based solution is that the practice of PA still has many uncertainties that are the subject of ongoing research. As a result, new processing methods and techniques may need to be incorporated into information systems in the near future as they become available and tested.

The experimental work carried out in the case study aims to obtain architectural and ICT solutions for interoperability between acquisition systems, business information systems, and mobile platforms by developing a Web platform dedicated to PA. The proposed solution allows overcoming the critical issues for access to environmental and production information to the actors involved (farmers, technicians, researchers, and policymakers).

Among the specific objectives of the work, we can therefore indicate the following:

- Implement a shared knowledge environment by developing a FOSS-based Web platform to access and analyze geospatial data acquired from heterogeneous sources. The system is based on SOA's architectural concept (Korotkiy, 2005) that provides fundamental support to interoperability and distributed systems.
- Apply OGC standards for interoperable data sharing.
- Develop automatic procedures to acquire data flows from satellite, meteorological, pedological, and other different acquisition methods for operational and integrated research applications.
- Characterize intra- and inter-field spatial variability for the planning of site-specific management techniques.
- Develop new site-specific and multi-scale analysis methodologies for the advancement of environmental and productive knowledge.

The challenge was also to push the research on the analysis of vegetation indexes (VIs), derived from high-resolution multispectral images acquired from satellite platforms, aimed at more rational fertilizer use.

On the whole, the approach towards the sharing of such a wealth of data by research institutions and operators in the sector can encourage the development of new site-specific analysis methodologies in order to expand knowledge for more rational use of natural resources and, at the same time, the enhancement of quality products that can meet the needs of the market in a context of intense competition at a global level.

1.2 Precision Agriculture (PA)

The world of agricultural production has changed radically in recent decades and many changes are still approaching. Economic pressure and technologies have profoundly changed the profile of farmers and farm structure. Nowadays, most of the farms are managed by large companies (Gliessman, 2000).

The capacity to manage the capital invested in equipment, land, and technology matters more than agronomic knowledge and farming practices in determining companies' economic performance. The influence of economies of scale has been the significant factor of the dominant management strategy with the consequent, progressive affirmation of large monoculture extensions that do not consider the variability of soil characteristics and heterogeneous environmental conditions. This type of homogeneous management of treatments causes either an overdose or underestimating the inputs necessary to crop production (Schnug et al., 1998).

The challenge of keeping the price of products at a competitive level in the face of the progressive increase in production costs is an issue that farm management has faced by adopting new strategies based on the targeted application of inputs using new technologies. These technologies concern the use of remote sensing, Global Navigation Satellite System (GNSS), advanced mechanization, mathematical models of crop growth, GIS, statistical analysis techniques, and all those solutions that could support farm managers in maintaining the economic balance of companies within certain set limits.

This context was the origin of precision farming. This term indicates the use of new technologies in the management and control of farm activities and agricultural production to optimize inputs' distribution to increase production yields, quality, and profits.

Definitions of PA abound in the scientific literature (Zhang et al., 2002; Lowenberg-DeBoer and Swinton, 1997), but the most recognized is the official definition provided by the International Society for Precision Agriculture (ISPA - https://www.ispag.org/about/definition, last access 10 October 2020): "*Precision Agriculture is a management strategy that gathers, processes and analyses temporal, spatial and individual data and combines it with other information to support management decisions according to estimated variability for improved resource use efficiency, productivity, quality, profitability and sustainability of agricultural production*."

This definition considers the information to be optimized concerning the spatial scale and evolves towards the importance of the decisions to be taken in space and time. Moreover, it does not take into account any particular technology or a set of technologies. Decisions can be made with the simple support of electronic sensors, GPS, GIS, etc., or by the operator alone without the aid of technology.

However, it is not easy to define what "net benefits" are indicated in such a definition. In a simple way, one can speak of "*a concomitant increase in the quantity and/or quality of production and/or the environment using the same or lower quantities of inputs*" (McBratney et al., 2005).

Even if such a definition is abstract in terms of quantity of inputs to be used, it is undoubtedly the one that makes the concept of PA evolving towards objectives more focused on socially and environmentally sustainable development without neglecting the traditional objective of economic sustainability of productions.

This last hypothesis can lead to an even broader global debate on whether the adoption of management strategies based on precision agriculture can be a solution for sustainable development in agriculture. The hypothesis formulated by McBratney et al. (2005) is that this choice can certainly be an opportunity and a well-defined challenge, then it is necessary to make this hypothesis plausible and convincing.

A broad overview of PA techniques' diffusion is provided by Zhang et al. (2002), who also proposes spatial and environmental indexes about each country's potential to adopt PA. They reflect the data reported for Italy, whose spatial index (hectares of crops/workers employed in the sector) is rather low (9.1), while the environmental index (159.4), related to the use of fertilizers (kg per ha of a crop), is well above countries like the United States (103.4), and whose agricultural production has always been characterized by intensive monoculture production.

1.3 Spatial and temporal variability

The PA, as we have seen in the previous section, consists of the application of principles and technologies for the management of space-time variability associated with all aspects of agricultural production. Without variability, the concepts of PA would have little meaning and probably would never have developed.

The environment variability has both a spatial and temporal dimension, such as infestations, climatic parameters, vegetation indices. The variabilities that influence agricultural production can be summarized in the following categories:

- variability of productions (historical and current distribution of crops);
- field variability morphology, proximity to watercourses, and edge effects);
- soil variability (physical and chemical properties of the soils, depth);
- crop variability (plant density, height, water stress, biophysical properties);
- variability of abnormal factors such as disease, insect pests, hail, and wind damage;
- variability due to management such as agronomic practices (sowing, irrigation, cultivation techniques, applications of pesticides and fertilizers, etc.)

Variability management can be done using different approaches:

- cartography-based;
- sensor-based;
- zoning-based.

With GNSS technology availability, remote sensing, yield monitoring, and soil sensing technologies, the cartography-based approach is much easier to apply. However, we must not neglect the procedures to be adopted (methodological and data processing) for applying these technologies and the transferability of the maps to the devices used for the variable rate application of inputs. The sensor-based approach, i.e., the measurement of soil or plant properties, uses a wide range of instruments that can range from real-time measurement installed on mobile devices and operating machines that perform variable rate field operations.

Many experimental systems in PA are based on the cartographic approach. GIS and geospatial databases, which manage maps derived from various acquisition platforms, and field sensor data in a single relational environment, are now widely used to integrate acquisition systems. Advanced geostatistical methods are applied with an integrated environment to analyze agricultural production's spatial and temporal variability. Moreover, crop modeling for the generation of yield potential maps for fertilizers' prescription should not be overlooked. These maps can be used to study the variability of crop growth and disease spread with the integration of weather forecasts.

Zoning management provides for the subdivision of fields into homogeneous zones, by homogeneous property characteristics concerning the entire field, for site-specific applications in the distribution of inputs (Zhang et al., 2010). The management of homogeneous zones is efficient to realize for much more expeditious applications, i.e., spatially filtering the information to reduce the disturbance effects due to the measurement of countless variability factors. Removing excessive detail within the same plot of land simplifies zoning operations and reduces variable rate devices' demands.

1.4 Management Unit Zones (MUZ)

The space-time variation of soil properties and weather conditions can significantly affect the crop's growth, the fruit's development, and the final quality of the harvest. In order to increase farmers' income and, at the same time, improve environmental protection, it is necessary to adopt agronomic techniques to specific local conditions, which is precisely the main objective of PA. For this purpose, the research has focused on determining management unit zones (MUZ), which are defined as those sub-regions of the field within which the effects on the crop induced by seasonal differences in climate, soil and management, are more or less uniform (Lark, 1998). MUZs are the basis for prescription maps, which allow the VRT of agronomic inputs such as fertilizers or irrigation. Generally, the definition of these classes starts from a multi-variety set of spatio-temporal data, which include those properties considered influential on yields such as topographic attributes, soil properties (e.g., texture and apparent volume mass), and chemical properties (e.g., pH, electrical conductivity, organic content) (Moore et al., 1993; Gessler et al., 2000). To define a class, i.e., a group of elements more similar to each other than with individuals from other classes, different approaches have been developed, and at the moment, there is no a universally accepted methodology. To identify such classes, it is necessary to group similar individuals based on suitably chosen properties.

Several types of information have been used for the delineation of the MUZ and include biotic and abiotic factors, climatic, topographic and all those considered influential on crop yield (Carr et al., 1991; McCann et al., 1996; Lark, 1998; Nolan et al., 2000). Photographic images of bare soil (Fleming et al., 2000), remotely sensed radio-metric images (Mulla, 2013) or geophysical sensors on the ground in proximal sensing have been used (Castrignanò et al., 2015; Morari et al., 2009). Another approach uses yield maps (Blackmore, 2000; Basso et al., 2007) and, more recently, aerial drone images (Matese et al., 2015) recorded at

different times during a crop's growing season. The use of crop remotely sensed images during its different growth phases is based on the assumption that there is a strong correlation between them and the yield data, recorded in the same year so that these images can be considered representative of the production potential for that year. Various yield predictive models have been formulated according to multi and hyper-spectral vegetative indexes. Whatever approach is used, the delineation of the field in homogeneous areas must consider both soil and crop properties and the effects that meteorological variables can have on yield. Once the optimal subset of the factors considered influential on the production, both in quantitative and qualitative terms, has been defined, the next step will be the field classification.

There is no single or universal methodology for this purpose. Cluster analysis, widely used in the past (Stafford et al., 1998), is based on the principle that similar individuals are grouped into discrete classes or "clusters" based on the properties measured on each individual (Lark, 1998). There are several clustering procedures, but none is universally accepted to derive MUZ. Most traditional clustering techniques aim to improve knowledge about the intrinsic structure of data and group them in the space of variables without reference to the observations' geographical position. According to such a paradigm, clusters are hyper volumes, multi-dimensional, discrete (crisp), and not overlapping. An observation can be assigned univocally (unequivocally) to one and only one cluster and remains linked to the attributes and the centroid's corresponding standard deviations. In the passage from a cluster to its contiguous, it is expected that their properties differ much higher than between the points inside a single cluster. Therefore, so most of the total variation is represented by the variance between the clusters. A residual variation within a class is still recognized in a smaller proportion than that between clusters, which are assumed to be homogeneous and spatially unrelated. Such an approach does not consider the spatial correlations between the observations and the gradual variation, which can occur both between clusters and within each cluster. For most agricultural soils, such a variation pattern does not correspond to reality, as soil and crop vary continuously and not discreetly, so it will be challenging to draw the boundaries of the MUZ unambiguously.

As mentioned earlier, most properties in an agricultural field show spatial dependence at various scales, which is why geostatistics is generally preferred to describe spatial variation. According to the geostatistical paradigm, each soil or plant property is considered a regionalized random variable that varies continuously according to a spatial dependency

function. Therefore, geostatistics does not use a partition in discrete classes to represent spatial variability. However, for practical reasons, in the PA, it can only make sense to divide the field into a small number of MUZs of sufficient size and contiguous to allow the execution of agricultural operations. Too many MUZs do not make sense in an operational context, as they would lead to areas too smaller and without agronomic meaning.

1.5 Monitoring of agricultural production systems

Rational use of natural resources, energy-saving, and pollution reduction are just some of the keywords that today's environmental sustainability logic requires the agricultural production sector. New technologies available in agriculture for environmental, operational and production monitoring can now easily combine with ICT to expand knowledge of sustainable production systems.

The types of monitoring can be schematically divided into three areas (Vieri et al., 2010):

- environmental monitoring that involves the acquisition of various physical parameters such as temperature, humidity, soil texture, solar radiation, and/or chemical parameters such as pH, the organic substance present, connected with the environment in which the crop develops;
- crop monitoring that involves the acquisition of information on phenological, nutritional, phytosanitary, and productive stages;
- operational monitoring that provides for the acquisition of all those data related to production activities.

1.5.1 Environmental Monitoring

In crop development, the environment assumes considerable importance in conditioning the qualitative/quantitative expression and defining delimitation production areas. With the term environment, in the framework of the present research work, we mean the various factors related to climate and soil, also influenced by geomorphological components such as latitude, altitude, exposure.

In this regards, the agro-meteorology provides useful indications for the best management of the agricultural activity. Once only limited to meteorological monitoring networks, weather stations are now increasingly widespread at the farm level and often managed by

farmers for direct crop monitoring (e.g., water stress, frosts, intervention thresholds for parasite and fungal attacks).

We are also witnessing a significant improvement in the management and acquisition software of field data through a direct connection with the acquisition unit, the download of data and an analytical and graphical pre-processing of the measured data with increasingly shorter acquisition intervals. In addition to the daily visualization of microclimatic data, it is of fundamental importance the possibility to create a historical database on the climatic trends of the different agricultural years. This database is beneficial both from the company's organizational point of view and for applications aimed at product and process traceability protocols (Vieri et al., 2010).

The agro-meteorological monitoring shows interesting applications in the field of rationalization of water resources (irrigation and control of irrigation shifts through the monitoring of crop water stress).

The evolution of sensors has also contributed over the years to the evolution from simple weather stations (limited to the measurement of meteorological parameters such as precipitation, wind, air humidity), to agrometeorological stations, capable of measuring more complex environmental parameters such as soil temperature and humidity, leaf and fruit temperature, evapotranspiration. Besides, the sensor technology supported by wireless technology also allows continuous monitoring and networking of sensors connected to a single remote control unit. In this way, procedures for remote control of the interventions to be carried out in the field and automation of some automatic irrigation processes have become established.

Not to be neglected the use of meteorological parameters for the application of simulation models of crop growth, disease development and recent studies on the impact of climate change on agriculture.

The micrometeorological environmental control is also at the center of continuous experimentation, aimed at understanding the effects of different crop management to expand knowledge in agriculture's sustainability.

1.5.2 Crop Monitoring

Farmers have well known the variability of crop production. The knowledge of the internal variability of individual plots has always been the basis of small farmers' choices for the

planning of their cultivation operations. Modern agricultural enterprises and the increase in farm size need innovative tools to make available to all operators in the sector considerable amounts of data to reach the same degree of specific knowledge on a large scale. The tools that the market and research makes available to us for the study of variability and thus quantify the parameters that determine the heterogeneity of crops for site-specific interventions, based on sustainability, eco-compatibility, and production traceability processes belong precisely to the precision agriculture sector.

Among the most widespread instruments for the observation and monitoring of cultivation parameters at different scales, we can remember:

- Remote sensing using different and platforms ranging from satellites to aircraft to remote-controlled drones.
- Proximal Sensing, using sensors in the vicinity of the crop.

The principles behind the two techniques are almost identical: an imaging system is used to detect and store the reflection of sunlight from the surface of a target on the ground, which, in the case of crops, includes plant cover and soil. The amount of sunlight reflected from the target is described in terms of the spectral reflectance profile (or spectral signature). Remote sensing images primarily provide information about the surface characteristics of crops concerning the horizontal development zone (canopy).

The digital technology of multispectral imaging systems allows the analysis of crops using more than one wavelength. The most common systems use four separate image sensors (one per color) that analyze reflected sunlight in blue, green, red and near-infrared (NIR) wavelengths. Starting from this prerogative, i.e., to associate to specific wavelengths of the electromagnetic spectrum of specific elements, it can be understood how the possible applications can be increasingly complex and diversified according to the specific monitoring purposes.

Earth Observation (EO) systems have been introduced in global projects over the last decade to track climate, atmosphere, land cover, land use, vegetation cycle, variables and changes and to exchange data. It is worth noting Copernicus (Aschbacher et al., 2017), Global Earth Observation System of Systems (Lautenbacher et al., 2006), NASA's Earth observation system (Esfandiari et al., 2007), among the numerous initiatives on a global scale. The advancements in methods of data collection also facilitated an increase in geospatial data

acquisition capabilities. EO data can now be searched, acquired, and processed in technical infrastructures on an increasingly broad scale and with a time series period that is increasing (Giuliani et al., 2020).

In recent years, parallel to remote sensing survey technologies, an interesting series of sensors for investigating crops called proximal sensing systems or side-looking sensors has been developing. These proximal sensing systems are able to provide georeferenced data with high accuracy and are currently divided into image analysis sensors and spectral reflectance analysis sensors. The choice of technology to be used and the most appropriate sensor must be evaluated according to the type of culture to be monitored and the objectives to be set according to the expected investment's economic availability and the available benefits. The acquisition of multispectral data, both from remote and proximal sensing, requires appropriate geostatistical elaborations to produce thematic maps capable of visually describing the crops' vegetative state. To this end, a series of vegetation indices have been developed over the years to compare analytically the biomass of crops measured in different wavelengths.

The Normalized Difference Vegetation Index (NDVI) (Rouse et al., 1973) and the Modified Soil-Adjusted Vegetation Index (MSAVI) (Qi et al., 1994) are worth noting among the many VIs proposed by scholars in the last forty years. For historical trend analysis and vegetation monitoring studies for land surfaces, the NDVI provides vegetation data. Besides, according to the current research context, the NDVI is recognized as directly related to the phenology and yield of wheat (Moriondo et al., 2007, Benedetti et al., 1993, Lopresti et al., 2015). The MSAVI and its revision, MSAVI2, aim to resolve some of the shortcomings of NDVI for areas with a high degree of soil surface exposure (Ahmad, 2012).

1.5.3 Operational Monitoring

The technologies used in agriculture significantly use mechanization for the entire cultivation cycle's farm operations, from tillage, sowing, and plant protection treatments to harvesting. Reducing the impact due to mechanization, both in terms of reducing polluting emissions and compacting effects on the soil, are objectives that the world of agricultural mechanical engineering is facing with the introduction of new monitoring technologies integrated with advanced devices. Emerging technology is telemetry, a computer technology that allows the measurement and transcription of interest information to the system designer or operator (Vieri et al., 2010).

Telemetry consists of collecting and evaluating a large amount of mechanical and productive data, using a series of sensors strategically applied in crucial points of a machine (for agriculture, it goes from the tractor to the grape harvester, etc.) so as not to hinder its operation. The number and type of sensors vary according to the needs of those who control the machine's work. Control systems coupled to the machines are found on tractors, combine harvesters, machines for the mechanical harvesting of products, or variable rate machines to apply plant protection products. These systems allow integrated control of crop productivity.

1.6 Information and Communication Technologies for Precision Agriculture

It is now well known how much agricultural production systems have benefited considerably from the inclusion of advanced technologies already developed by other industrial sectors. In agriculture, we have seen how the industrial era has brought mechanization and the use of fertilizers. The technological era has been characterized by the development of genetic engineering and process automation. We can currently say that the information age has offered great potential for the integration of advanced technologies in PA.

The evaluation of the spatial and temporal variability of production has always been taken into account over the centuries, and today is recognized as a fundamental PA element. Before the mechanization era, the small size of the fields allowed farmers to vary crop treatments manually. Later, with the increase in field size and intensive mechanization, it became more difficult to manage the variability within plots without a revolutionary development of technologies (Stafford, 2000).

The PA is based on the reorganization of the entire agricultural system with low input use, but a high level of efficiency and sustainability (Stafford, 2000). This new approach mainly benefits from the diffusion and convergence of some innovative technologies, including the GNSS, GIS, miniaturized computer components, automatic control systems, remote sensing and proximity sensing, advanced telecommunications information processing systems.

The technology used in agriculture can now collect much more comprehensible data on production variability both in space and time. The challenge to react to this variability at an increasingly detailed scale has thus become the objective of the PA (Whelan,1997).

After more than twenty years of development, the PA has reached a crossroads, consisting of much more technology needed and available than the economic and environmental benefits found, not yet fully demonstrated and evident (Stafford, 2000). Over the years,

improvements resulting from technological innovation have been widely used. However, the development of agronomic and ecological principles for recommendations aimed at a rational and localized use of inputs has generally been much slower.

As a result, even farmers are currently very uncertain about how to use PA on their farms. In this regard, a boost to PA use could come from more restrictive environmental legislation, public interest awareness of the excessive use of agrochemicals, and the economic advantage in reducing the use of inputs and finally from improving the efficiency of farm management. After all, downstream of this evolution, the success of PA technologies can be measured by the real economic and environmental benefits.

The technological evolution of sensors, computers, and communication devices profoundly influences agriculture changes (Kitchen, 2008). Modern agriculture has been based for decades on information-driven management as long as the decisions to be made wide-ranging and straightforward. In the last twenty years, the ability to acquire differentiated and massive information at different spatial and temporal scales has grown almost immeasurably, so much so that the processes needed to transform information into decisions have to be accelerated. Therefore, researchers, professionals, and farmers have filled their activities with images, maps, data, and tables to the point that we are drowning in information but lack knowledge.

In PA, the information is the driving force of agricultural management. The time and capital invested in collecting production data and the actions necessary to transform them into decisions often need to be compensated and balanced by the improvements, even in the immediate future, that can be achieved. If these expectations are not realized in time, the producer tends to return to a traditional or previously consolidated management. Clearly, in a market economy, the first test to evaluate the improvements achieved through such detailed information is profitability. Thus, economic evaluation has a stronger effect on the agricultural producer's decision on whether or not to adopt long-term practices (Griffin et al., 2005).

In the last decade, geospatial analysis techniques and methodologies have become widespread in agriculture, making it easier for end-users to visualize geospatial data. Today the interpretation of geospatial data has become easier and easier to understand. However, not everyone in the industry has the expertise to use GIS, or perform geospatial analysis, correctly and efficiently. The sustainability of GIS applications, based on commercial

products, is strictly dependent on individual companies' economic availability. In this situation, the ability to use Free and Open Source GIS tools (FOSS4G) allows for the development and management of customized applications at relatively low cost. Thanks to their flexibility of use, such systems can effectively become the basis for building a new information culture no longer conditioned by data availability, but preferably oriented to users' needs.

There are many tools available to developers that are used for the implementation of web platforms and, in principle, we can divide them into two main categories:

- Proprietary
- OpenSource

These tools consist of development libraries, applications for data creation and editing, geospatial databases, platforms for publishing web applications, etc.

The choice of tools for the development of web platforms is a step that requires a careful evaluation of both the components that will constitute the geographic information system distributed on the Internet and the sustainability of the service and its interoperability in a more global system of spatial data infrastructures.

1.6.1 Proprietary Tools

The Proprietary Tools are characterized by a well maintained and tested software package, technical support provided directly by the companies that produce these tools. For this reason, proprietary tools are often easier to use, requiring less basic computer knowledge. All this, however, in the face of often very high costs both for the purchase of rights to use these objects (eg: software), and for their maintenance and updating. For this reason, these proprietary tools are widely used in the development of applications for large institutions both public and private that need to make their data available to a large number of users within their institution or outside their structure, thus being able to amortize the costs of commercial products and reducing investment in maintenance and development. Among the main proprietary tools used for the development of spatial data infrastructures we can mention:

- Microsoft Windows OS (Operative System)
- ESRI ArcGIS (GIS Desktop Software)
- ESRI ArcIMS (GIS Server)

- Microsoft VisualBasic, VisualStudio, .NET (Programming Language)

1.6.2 Free and Open Source Software

Open Source tools are born from the collaboration of developers and designers around the world. Open Source is not only a set of rules and technologies but also an exact philosophy. Most of the Open Source tools, libraries, software and operating systems are freely distributable without restrictive copyright constraints. They are often released together with the source code so that other developers can freely modify and customize the code to meet their needs; doing that also allowing to self-feed the Open Source community with software always evolving and always updated. This type of approach finds a broad scope in all those situations in which there is limited economic availability (non-profit institutions, public research institutes, etc.) or needs to develop custom and/or advanced features compared to closed commercial packages.

However, the use of Open Source tools requires in-depth computer knowledge and greater familiarity with software design and development techniques. Developers must also have a strong sense for finding solutions to the problems that may arise during development and be available to the global sharing of code and know-how to face and promptly solve the problems encountered during an application's design.

Between the leading Open Source tools used for the development of informative systems on the web we can enumerate the following ones:

- Linux OS (Operative System)
- Apache (Web Server)
- TomCat, Jetty (Application Server)
- Quantum GIS (GIS Desktop Sofware)
- MapServer, GDAL, Geoserver (GIS Server)
- PHP, Java, Jsp, Ajax (Programming Language).

1.6.3 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA), Web services, mash-ups, Ajax, Web 2.0, and some of their implementation schemes such as SOAP (simple object access protocol), UDDI (universal description, discovery, and integration), XML (extensible markup language), and many other approaches to computer systems architectures have become the keywords in the information systems community.

Several formal definitions of SOA exist at the time of this work, from sources such as: Organization for the Advancement of Structured Information Standards (OASIS), Open Group, Object Management Group (OMG), World Wide Web Consortium (W3C), Webopedia, TechEncyclopedia, etc.

The Open Group defines SOA as "*an architectural style that supports service orientation*". The definition continues with the description of the architectural style and service orientation. OASIS defines SOA as "*a paradigm for the organization and use of distributed functionality that can be under the control of different proprietary domains*". OMG defines SOA as "*an architectural style for a community of service providers and consumers to achieve mutual benefit*". OMG adds that SOA allows technical independence among community members, specifies the standards that community members must agree to adhere to, provides business and process value for members, and finally, allows a variety of technologies to facilitate community interactions. W3C defines SOA as "a form of distributed system architecture typically characterized by logic, message orientation, descriptive orientation, granularity, and platform neutrality.

Several authors (Papazoglou, 2003; Erl, 2005; Papazoglou et al., 2007) have proposed a three-level hierarchical perspective of SOA where the first level includes the application front-end, the service, the service repository and the service bus (SB). The second level includes the service interfaces. Finally, the last level of the proposed hierarchy is composed of business logic and data.

Yoon & Jeong (2018) in a recent work have identified the main advantages of SOA that can be summarized as follows:

a) decoupling, the infrastructure and architecture are divided into various services and as a result, the software can be freely coupled (or not dependent on each other);

b) flexibility, i.e. one can develop the components of the architecture in any language and platform. For example, one can write the client-side in a dynamic language like Python / Ruby / Javascript and write performance-critical components in lower-level languages like C or Java;

c) simplicity of debugging, having components isolated in various services make it easy to test and debug them individually;

d) scalability, the presence of separate components makes it much easier to scale architectures, i.e. one can scale a particular component without affecting the others;

e) reuse, as the various components are built separately, it becomes much easier to reuse them later.

1.6.4 Web service and data interoperability

Geospatial data analysis plays a central role in PA. The complexity of the software and the high costs in terms of time and money required to manage the large data prove to be obstacles to the diffusion and adoption of PA (Fountas et al., 2005; McBratney et al., 2005; Jarfe & Werner, 2000). In particular, the lack of interoperability between different software is identified as a primary issue (Kitchen et al., 2005). Precisely this problem of interoperability of geospatial data between software and systems has been at the center of work in the geospatial information community over the last 15 years. Many standardization initiatives are now taking shape around the work of the OGC, a global body with over 300 members from industry, government and academia, and the ISO/TC211 Geographic Information/Geomatics Technical Committee, ensuring ISO compatibility for OGC specifications. These standards can be applied in many fields, including PA. Of particular interest for the PA are the standards for the transfer of geospatial data using web services, which allow the exchange of information on-demand between distributed systems.

Web services provide the functionality of computer systems through a network interface. It is important to distinguish Web services from Web pages and Web-based applications; the latter two are intended to be used by a user using a browser, while Web services are accessible to software called clients. These clients can, in turn, be part of a web-based application or even a "cascading" web service, or they can be desktop applications. Web services are usually implemented to be self-describing, i.e. a client can automatically determine what functionality and/or data is available from a particular service. In a service-oriented architecture, each service provides specialized data or functionality, creating networks of applications that can be managed by one or many organizations. Web services are supported by accepted standards and pervasive network technologies that connect every component (including servers, desktop clients and mobile clients such as laptops, cell phones or onboard computers) to the Internet via a wired or wireless connection (Curbera et al., 2003). The application of web services to the agricultural sector is not new. Some authors (Spilke and Zürnstein, 2005; Casadesus et al., 2007) highlight the potential of web services for data transfer between partners in agriculture and for application integration, including external service providers, and also consider the architectural requirements for a sensor-controlled precision irrigation SOA.

1.6.5 OpenGis Standard

OpenGIS standards, developed and proposed by OGC, are generic standards that can be used in many domains. Of particular interest for agricultural data are the standards for Web services and transfer formats for geospatial information. It should be noted that most web service interface specifications, OpenGIS allows the client to specify the format and coordinate reference system (CRS), from those supported by the server, in which the data must be returned. Many problems with integrating data from heterogeneous sources are therefore greatly simplified. However, because data can be stored internally in a different format, there may still be problems transforming the reference system and thus spatial misalignments. Although there may be disadvantages to using OpenGIS standards for agricultural data flows since they are not explicitly designed for this task, they have several advantages:

- standardized multi-purpose interfaces and many implementations (both proprietary and open-source);
- the interfaces comply with existing ISO standards and are already used in a wide range of applications;
- many basic data sets such as topographic maps, digital terrain models, environmental data, geological data and others have already been made available using these standards as part of national and regional spatial data infrastructure initiatives such as INSPIRE (EC, 2007).

1.6.6 Geography Markup Language

Geography Markup Language (GML) is an XML grammar, i.e. a schema for modeling, transporting and storing geographic information; it is currently in version 3.2.1, which is identical to the ISO 19136: 2007 standard. The GML offers a variety of object types to describe geography, including features, coordinate reference systems, geometry, topology, time, units of measurement and generalized values. (Cox et al., 2003). To apply GML, it is necessary to create an application schema (implemented as an XML schema) that defines the characteristics of interest within the application domain. Such application schemes have been defined, for example, for city models (Gröger et al., 2006) and transport models (Cambridge Systematics, Inc. et al., 2006). GML would also be a good basis for standardizing data formats for PA (Korduan & Nash, 2005). A feature could correspond to a farm, a field, a soil sample, or a yield measurement at a single point or in a field in this context.

1.6.7 Web Map Service

The Web Map Service Interface (WMS) produces spatially referenced maps from geospatial information, both vector and raster. The WMS specifies the operations to retrieve a map description offered by a server and retrieve a map and query a server about the features displayed on a map (De la Beaujadiere, 2006). WMS version 1.3.0 was approved as ISO 19128:2005 standard. In the context of PA data flows, it is likely that the WMS is used to recover background images (topographic mapping, orthophotos, satellite images) or for the production of human-readable data summaries.

1.6.8 Web Feature Service

The Web Feature Service (WFS) allows clients to retrieve and update geospatial data encoded in Geography Markup Language " (Vretanos, 2005). Basic WFS enables data recovery, including vector geometry, with transactional extensions for insertion/update/erase operations defined as WFS-T. While a WFS must offer data in GML format, other specific formats can also be offered. A typical use of WFS in an agricultural context can be the retrieval of yield data from an agricultural process (Steinberger et al., 2006) or soil sampling results from a contractor's server.

1.6.9 Web Coverage Service

The Web Coverage Service (WCS) provides the functionality of basic WFS, but for raster data, effectively extending the WMS to provide a pictorial portrait of the data (i.e. images) actual values of the data itself, such as GeoTIFF or ASCII/Grid. The WCS is a suitable interface for delivering remote sensing data or interpolated maps (yield data, satellite indexes). Real data values are required and not just a simple graphical representation.

1.6.10 Web Processing Service

The Web Processing Service (WPS) offers any geospatial data processing functionality, typical of GIS Desktop software, to clients over a network, including access to pre-programmed calculations and/or calculation models (Schut, 2007). The WPS v1.0.0 standard was released in 2008 and defines an interface through which distributed processing capabilities can be provided. These functionalities can be typical of GIS such as map algebra (Kiehle et al., 2006), buffering (Heier & Kiehle, 2006) or spatial join (Stollberg et al., 2007) or more specialized functions such as the generation of fertilizer application maps or management unit zones (Nash et al., 2007).

1.6.11 Sensor Web Enablement

OGC's Sensor Web Enablement (SWE) initiative is focused on the development of standards to enable discovery, exchange and processing of sensor observations and sensor system functionality (Botts et al., 2006). Almost all aspects of the use of networked sensors are covered in the SWE, e.g., Sensor Model Language (Sensor ML) for sensor description and tasking, Sensor Observation Service (SOS) for recovery of observation data. In addition, there are also alert services (SAS), planning (SPS), and notification services (SNS). However, the communication behind the service interfaces is not defined, i.e. how the data transfer between sensors and an SWE service gateway is handled is not standardized, thus allowing any range of proprietary standards or protocols to be used for field communication, control of sensor nodes or transfer of measurements to the gateway (Walter & Nash, 2009). Given the current interest in wireless sensors (Kim & Evans, 2009; Lokhorst et al., 2008; Morais et al., 2008; Pierce & Elliot, 2008; Vellidis et al., 2007; Wang et al., 2006), the SWE standard can play an increasingly important role in the management and integration of real-time and continuous data collection for precision agriculture.

1.6.12 ISOBUS

Standardization initiatives in the agricultural machinery sector such as LBS ('Landwirtschaftliches Bus-System'/'Agricultural Bus System') and its successor ISOBUS - ISO11783, 'Tractors and machinery for agriculture and forestry-serial control and communications data network', have primarily focused on hardware compatibility. Although ISOBUS includes aspects of software-level communication between enterprise management information systems (FMIS) and on-board computers, there are currently no international standards for software-level communication between different FMIS. The problem of incompatibility between systems has been consistently cited as an obstacle to PA adoption (Pedersen et al., 2004; Reichardt & Juergens, 2006). In particular, when using specialized models and software, it is likely that these work as stand-alone software. Using a standard format for this data, which allows direct file-based transfer, would significantly improve this aspect of the data flow. Current initiatives define such formats, such as agroXML (Kunisch et al., 2009;), with extensions that include complex data on precision agriculture (Steinberger et al., 2007). Most of these initiatives aim to produce an XML schema that defines an encoding of agricultural data using structured text, computerized and human-readable markings (tags) that indicate each piece of data's meaning. Where data is held by multiple organizations and/or in a distributed system, such file-based data transfer is

sub-optimal because often only a particular value of data or information relating to a particular object is required, and it would therefore be preferable to be able to retrieve only this information on demand, rather than an entire file, particularly because the exact set of information to be retrieved may not be known in advance. An SOA architecture through which data can be exchanged on-demand using Web services with standardized interfaces and data transfer formats play a primary role in optimizing PA data flow.

1.6.13 Service-Oriented Architecture for Precision Agriculture

PA applications are based on detailed spatial information. Field trials are planned and conducted using geospatial data. Most of the information can be displayed on maps. The increasing number of specific sensors, satellite data, and VRT technologies increase the amount of data collected. The integration of data from different sources leads to a better understanding of processes and interactions between site characteristics, meteorological influence and management practices and helps to improve crop management decisions. The analysis of data from different sensors and external data providers requires structured collection and storage. Data must be transferred between data providers and used in different programs. For the interoperability of components, standardized interfaces are needed. As seen in the previous sections, in agribusiness, some initiatives create standards for business processes (AgXML), hardware compatibility (LBS; ISOBUS). The standardization of geospatial data formats is the goal of the Open Geospatial Consortium OGC (OGC, 2007) in collaboration with the ISO / TC 211 working group.

We describe an interoperable system based on OGC, ISO/TC 211 and ISO 11783 standards for recording, managing, and using precision agricultural data in the following sections. The system is described in a case study on a case study of 500 hectares of durum wheat crops sown in November 2018 and harvested in June 2019 in Italy.

2 Materials and Methods

2.1 Study Area

The study area is a portion of territory within the Municipality of Jolanda di Savoia (Province of Ferrara), a broad plain located in Italy's northern part (Fig. 1). The study area of about 40 km^2 has a flat morphology, with an average altitude of about - 1 m a.s.l..

It was a territory covered by water and marshes in the nineteenth century and reclaimed in the twentieth century. Within the study area, we collected data on 27 fields planted with durum wheat for a total of 512 hectares (Fig.1).

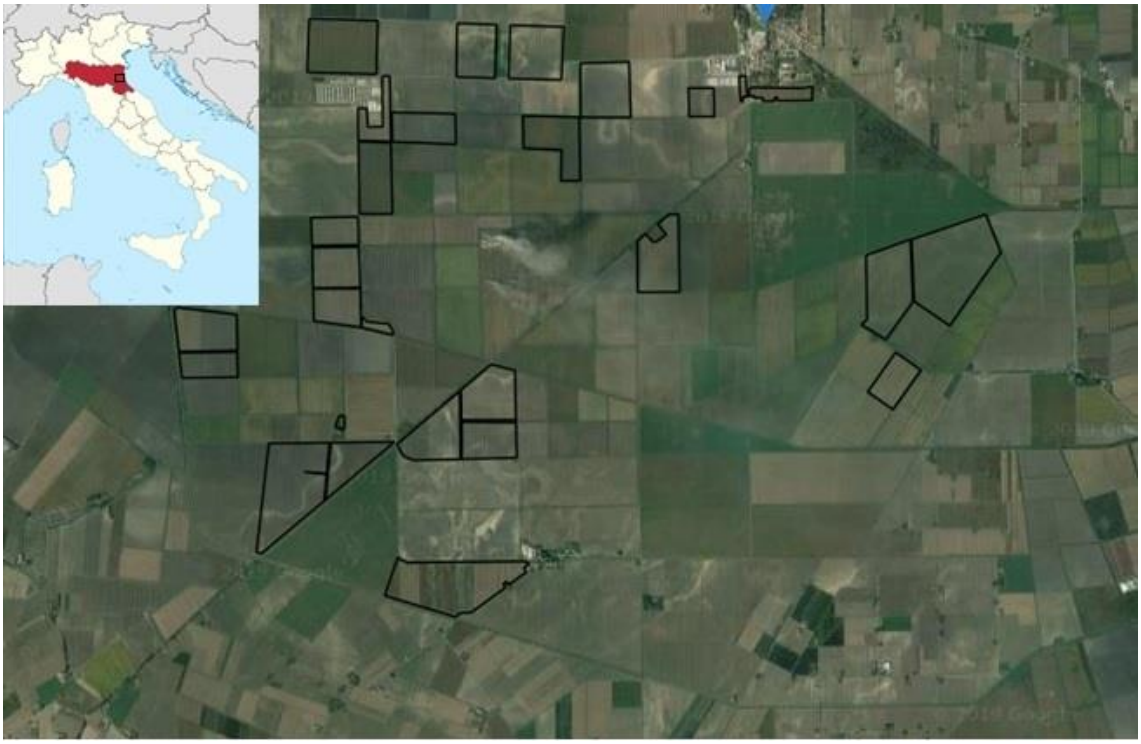

Figure 1: Geographical localization of the study area, the municipality of Jolanda di Savoia, upper left of the figure; fields with durum wheat cultivar, black polygons, the center of the figure.

2.2 Dataset

This thesis project framework has collected different datasets concerning 27 fields cultivated with durum wheat (Fig. 2-4) sown in November 2018 and harvested in June 2019.

Figure 2: Photo of a field cultivated with durum wheat in Jolanda di Savoia. The photo was taken on 2019/06/04.

Figure 3: Photo of a field cultivated with durum wheat in Jolanda di Savoia. The photo was taken on 2019/06/02.

Figure 4: Photo of a field cultivated with durum wheat in Jolanda di Savoia. The photo was taken on 2019/06/19.

The multi-temporal data, collected from different sources and processed using ICT via geomatics techniques, are relative to a) fields' boundaries; b) crop management plans; c) agronomic operations (sowing and harvest); d) soil properties d) meteo-climatic variables;

f) satellite data. The collected data allowed us to test the prototype's interoperability and analyze the space-time variability intra-fields and inter-fields.

The complete list of data types, file formats, and sources are shown in Table 1.

Table 1: Datatypes, files format, and source of the data collected in the case-study.

Data Type	Format	Source
Fields	Shapefile	Digitalization
Crop management plans Plan	Shapefile	Digitalization
Soil chemical and physical properties	Shapefile	SoilGrid WCS
Agronomic Operations	Shapefile	Agricultural Machines
Meteo-Climatic	JSON	MeteoBlu API
Satellite data	GeoTIFF	Sentinel open access hub Planetscope API

We collected the soil data from SoilGrids (SG - https://www.isric.org/explore/soilgrids). SG is a global digital soil mapping system that uses global soil profile information and artificial intelligence to model the spatial distribution of soil chemical and physical properties across the globe (Table 2) (Batjes et al., 2020).

SG generated and makes available, for free, soil maps for the world produced using machine learning at 250 m of spatial resolution and six standard depths – i.e., intervals – (Table 3) (Grunwald et al., 2011). SG share soil data by WMS, WCS, and Web-based Distributed Authoring and Versioning (WebDAV) standards.

We exploited the WCS to acquire soil data and use it as input to processing pipelines for our research aims. The acquired datasets are one raster, in GeoTIFF format, for each soil parameter and depth (Table 2). Each soil parameter values are expressed in the mapping unit and converted into conventional or international units.

The complete list of soil parameters, mapping unit, conversion factor, and conventional units is shown in Table 2, while the six standard depths (intervals), in Table 3.

Table 2: Data types, files format, and source of the collected soils' chemical and physical properties.

Description	Mapping units	Conversion factor	Conventional units
Bulk density of the fine earth fraction	cg/cm³	100	kg/dm³
Cation Exchange Capacity of the soil	mmol(c)/kg	10	cmol(c)/kg
Volumetric fraction of coarse fragments (> 2 mm)	cm^3/dm^3 (vol‰)	10	$cm^3/100cm^3$ (vol%)
Proportion of clay particles (< 0.002 mm) in the fine earth fraction	g/kg	10	g/100g (%)
Total nitrogen (N)	cg/kg	100	g/kg
Soil pH	pHx10	10	pH
Proportion of sand particles (> 0.05 mm) in the fine earth fraction	g/kg	10	g/100g (%)
Proportion of silt particles (≥ 0.002 mm and ≤ 0.05 mm) in the fine earth fraction	g/kg	10	g/100g (%)
Soil organic carbon content in the fine earth fraction	dg/kg	10	g/kg
Organic carbon density	hg/dm³	10	kg/dm³
Organic carbon stocks	t/ha	10	kg/m²

Table 3: the standard depths corresponding to the six intervals of soil parameters obtained from SoilGrids (SG) mapping system.

Depth	Interval I	Interval II	Interval III	Interval IV	Interval V	Interval VI
Top depth (cm)	0	5	15	30	60	100
Bottom depth (cm)	5	15	30	60	100	200

After acquiring the soil data, we harmonized it in one raster for each soil parameter and in conventional units. The processing pipeline for soil parameters acquisition and harmonization is developed by geospatial libraries in Python scripting language, illustrated in the Python code in Appendix Section A.

It is worth notice that we have acquired and harmonized the soil data, for each parameter, for all the Italian territory. Figure 5 shows the total nitrogen distribution in the Italian territory in 2019, superimposed on a satellite base-map.

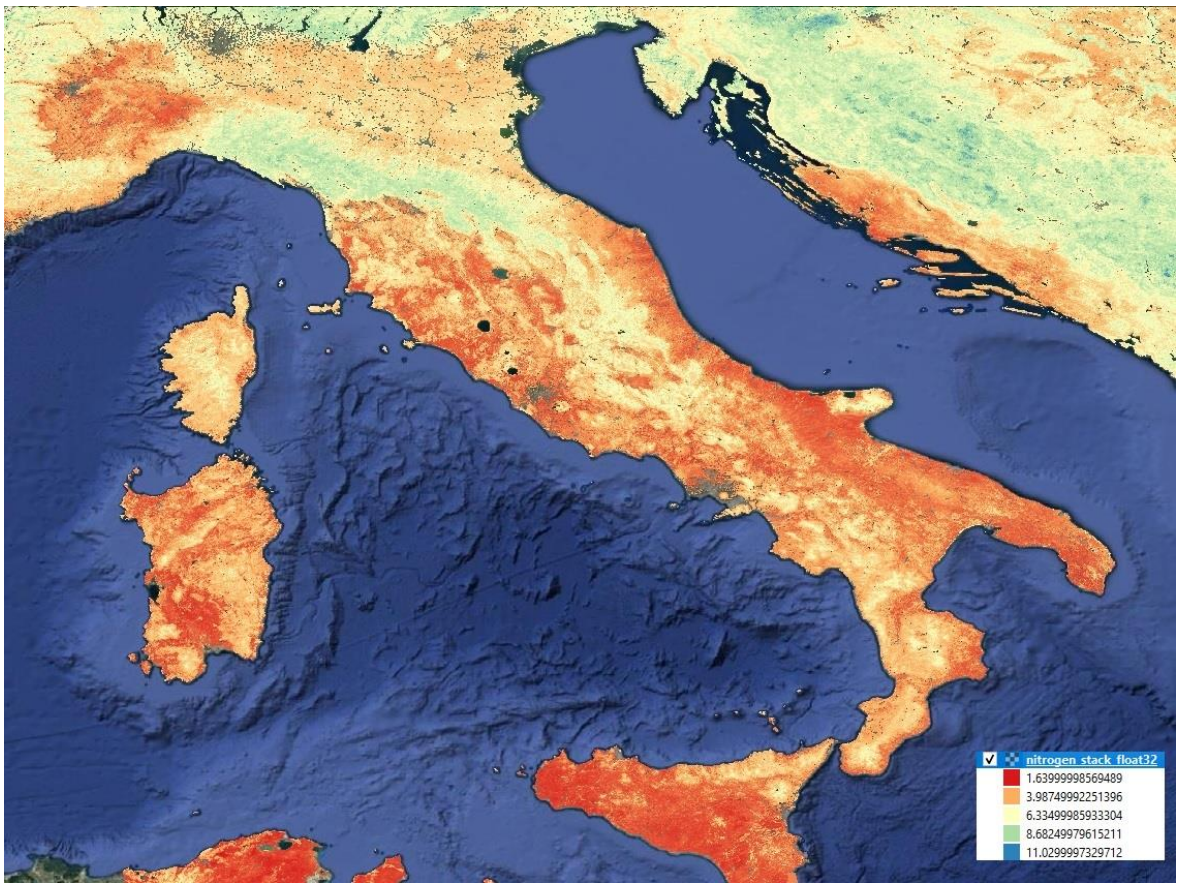

Figure 5: Total nitrogen (N) distribution in the Italian territory (reference year 2019), according to different five classes, superimposed on a satellite base-map.

After the acquisition of soil data, we collected multi-temporal satellite data, from November 2018 to June 2019, from two distinct sources: a) Copernicus Open Access Hub (https://scihub.copernicus.eu/); b) Planet (Planet, 2017).

The Copernicus Open Access Hub, developed in the Framework of Copernicus Programme, provides complete access to Sentinel-1, Sentinel-2 A and B (S2), Sentinel-3 and Sentinel-5P data, through a) Open Hub, i.e., access via interactive Graphical User Interface (GUI – Fig. 6), and b) API Hub that provides access by Application Programming Interfaces (APIs).

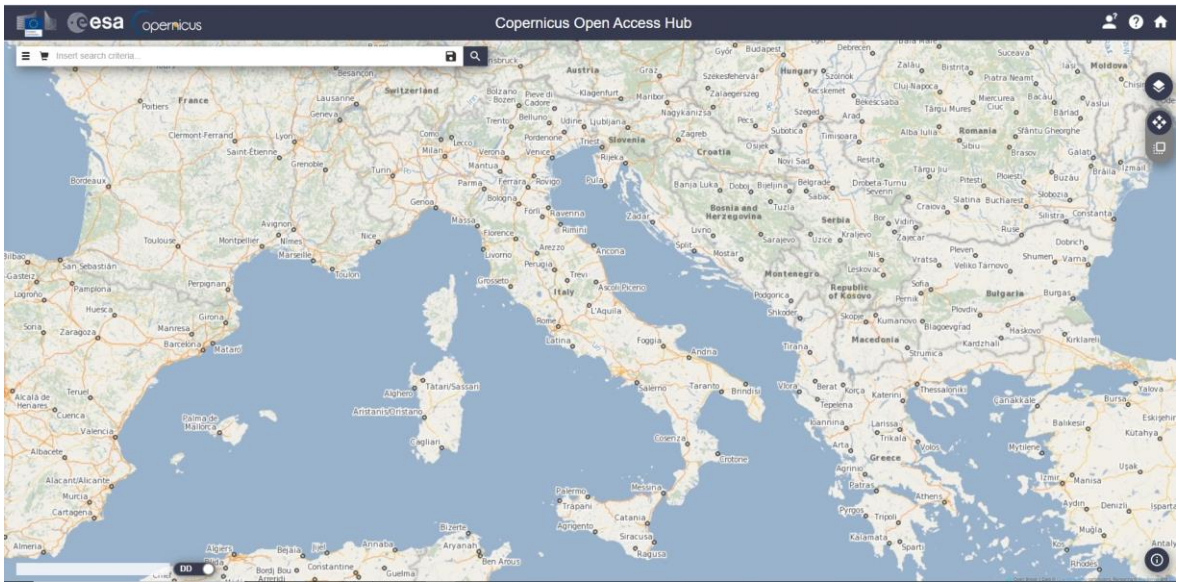

Figure 6: Graphical User Interface (GUI) of the Copernicus Open Hub. Users can select an area of interest through the graphical interface, add filter parameters such as date range, cloud cover, satellite, and download the relevant satellite images.

Planet is a private company that provides access to different satellite constellations: PlanetScope (PS), RapidEye, SkySat, through: a) Planet Explorer, i.e., access via interactive GUI (Fig.7); b) Planet API that provides access by RESTful interface to Planet's imagery archive. For our research aims, we exploited the Copernicus API Hub to acquire S2 data and Planet API to acquire PS data.

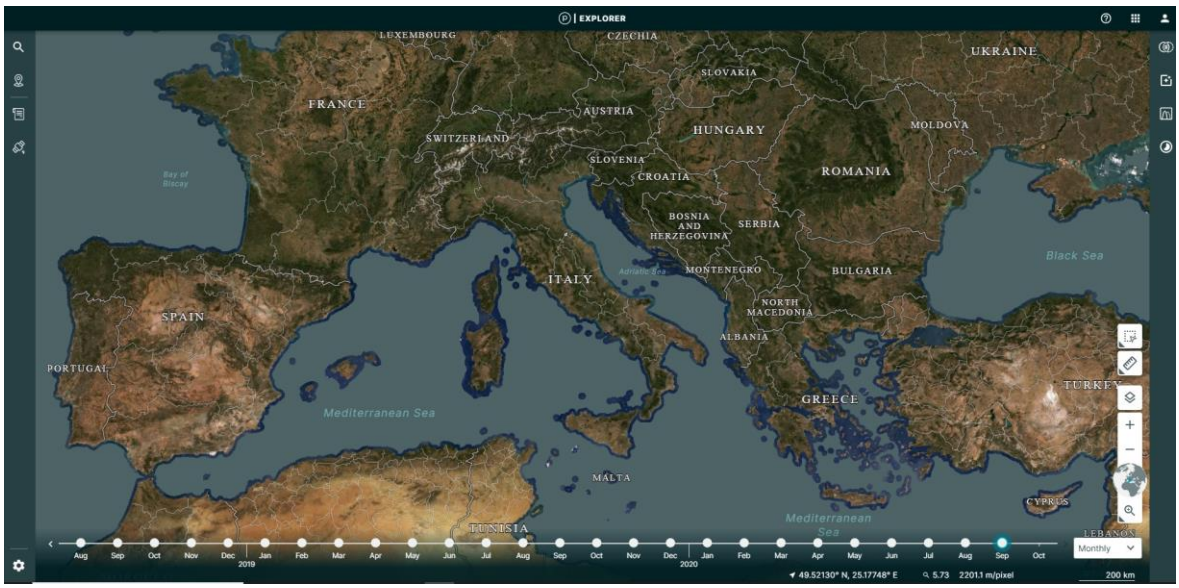

Figure 7: Graphical User Interface (GUI) of Planet Explorer. Through the graphical interface, users can select an area of interest, add filter parameters such as date range, cloud cover, satellite, and download the relevant satellite images

The processing pipeline for Satellite data acquisition and harmonization is developed by geospatial libraries in Python scripting language, illustrated in the Python code in Appendix Section B and C.

The S2 sensors acquire twelve bands, with a revisit time of two-three days for European countries and a ground sample distance (GSD) ranging between 10 and 60 m. The PS sensors acquire four bands, with a revisit time of one day and a GSD resampled to 3 m. The complete list of satellites' platforms, revisiting time, bands acquired, and GSD is shown in the following Table 4.

Table 4: Main characteristics of the used satellite data: revisiting time [days], band, bandwidth, and ground sample distance (GSD) [m].

Satellite	Revisiting Time [days]	Band	Bandwidth [nm]	ground sample distance (GSD) [m]
Sentinel2 A-B	2-3	1 - Coastal aerosol	421 – 457	60
		2 - Blue	439 – 535	10
		3 - Green	537 – 582	10
		4 - Red	646 – 685	10
		5 - Vegetation Red Edge	694 – 714	20
		6 - Vegetation Red Edge	731 – 749	20
		7 - Vegetation Red Edge	768 – 796	20
		8 - NIR	767 – 908	10
		8A - Vegetation Red Edge	848 – 881	20
		9 - Water vapour	931 – 958	60
		10 - SWIR - Cirrus	1.338 – 1.414	60
		11 - SWIR	1.539 – 1.681	20
		12 - SWIR	2.072 – 2.312	20
PlanetScope	1	1 – Blue	455 - 515	3,7
		2 - Green	500 - 590	3,7
		3 – Red	590 – 670	3,7
		4 - NIR	780 - 860	3,7

After acquiring the multi-temporal satellite data, we acquired multi-temporal Meteo Climatic Data (from November 2018 to June 2019) from the MeteoBlue services (https://www.meteoblue.com, last access 10 October 2020).

MeteoBlue is a private company that elaborates and shares worldwide meteorological, forecast, and historical data by FTP, API, email. We exploited the API to download the historical data for every analyzed field in the JSON file format for our research. The processing pipeline of meteorological data acquisition and harmonization is developed by

geospatial libraries in Python scripting language. The details of the Python code are illustrated in Appendix Section D.

The complete list of meteorological data parameters and the measuring unit acquired are shown in the following Table 5.

Table 5: Meteorological data parameters provided by MeteoBlue and acquired in the JSON file format through the implemented API.

Parameter	Measuring units
Time	Day
Max Temperature	°C
Min Temperature	°C
Mean Temperature	°C
Accumulated Precipitation	mm
Max Windspeed	ms-1
Min Windspeed	ms-1
Mean Windspeed	ms-1
Max Relativehumidity	percent
Min Relativehumidity	percent
Mean Relativehumidity	percent

All the data acquired are stored in the Data Layer of the Service Oriented Architecture (SOA) described in the following section.

2.3 The Web-Based Spatial Decision Support System (Web-SDSS)

The proposed Web-Based Spatial Decision Support System (Web-SDSS) has been developed basing on the SOA concept. SOA can be defined as multi-tier architecture comprised of services: a) independent from software; b) interoperable; c) discoverable and d) reusable (Erl, 2005).

Several authors (Papazoglou, 2003; Erl, 2005; Papazoglou et al., 2007) proposed SOA composed of three tiers: a) storage, for data, b) service, for business logic and service interoperability c) front-end, for graphical user interfaces (GUI). The same architecture is envisaged by INSPIRE EU Directive specifications (European Commission, 2007).

Based on these premises and previous researches (Lanucara et al., 2020; Lanucara et al., 2021; Modica et al., 2016; Pepe et al., 2019), we designed a three-tier SOA composed by different modules (Fig. 8) independent from software, as follows:

- Data archive: it allows the storage of data in the geospatial database and structured file system. Data can be provided from (i) local sensors such as, for example, meteorological sheds, humidity sensors, multi-parametric sensors; (ii) machinery such as yield data, agronomic application data; (iii) earth observation analysis from satellites, such as NDVI, MSAVI, and other vegetation indexes (VIs); (iv) other geospatial data such as field and parcel boundaries, soil data, VRT prescription maps; (v) provider of data as service (i.e., REST API, OGC services) such as meteorological data, weather forecasts, and soil data.
- Semantic: enables the creation and management of code-lists, controlled vocabularies, and thesauri used by meta-dating systems for a multilingual semantic enabling data description.
- Metadata: enables the meta-dating of data according to standard profiles, using the Semantic module for data description and the conversion between metadata schemes.
- Catalog: performs the collection and cataloging of data and relevant metadata, allowing univocal and detailed research of web services and data.
- Services: Transforms data into interoperable web services.
- Geo-Processing: Enables the execution of geospatial processing and algorithms for the production of digital data from knowledge, such as, for example, delineation of management zones, prescription maps for agronomic operation (i.e., VRT maps for soil treatment, sowing, fertilization, plant defense, differential harvest)
- Authorization and authentication: Enable authentication and authorization of users, data, and applications.
- Front-End: the access portal allows the users to insert, view, query, process, and download data.

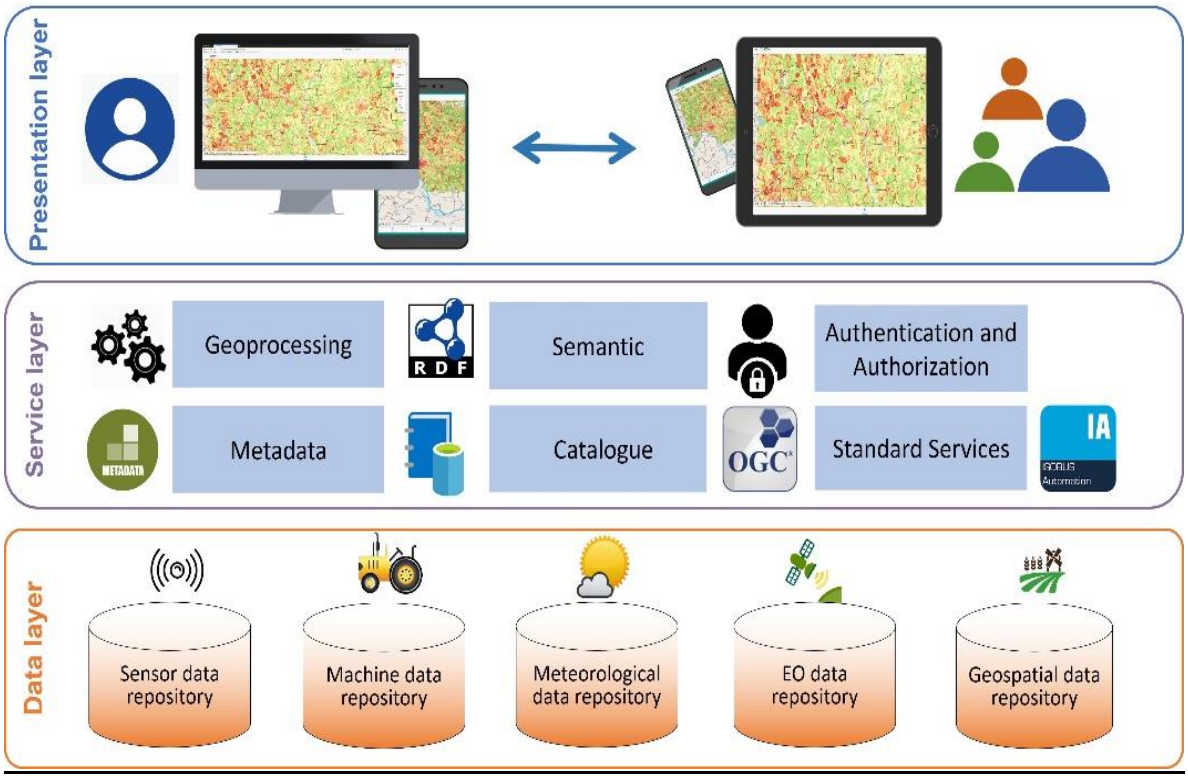

Figure 8: General scheme of the proposed modular, three-tier Service Oriented Architecture (SOA) based on free and open-source software (FOSS).

The interoperability of services and related data is achieved by adopting the OGC and ISOBUS standard specifications. In particular, for:

- Data from sensors and meteorological data, the standard used is the SWE, which in addition to allowing the description of sensors and their implementation, allows to search and download data such as weather time series, soil moisture data, solar radiation data, all in real-time;
- Data from agricultural machinery such as data of application of fertilization, seeding, weeding and yield the standard to refer to is ISOBUS which allows not only to download such data from machinery but also to provide the machinery with prescription maps for the different agronomic operations;
- Data from earth observation such as satellite data and/or data collected by unmanned aerial vehicles (UAVs) and their related products such as NDVI, Modified Soil-Adjusted Vegetation Index (MSAVI), the most appropriate standards are the WMS, with multi-temporal enabling, for their graphic representation, the WCS standard for the extraction of the related data;

- Geospatial data such as field boundaries, management unit zones, campaign notebooks, company master data, the most appropriate standards are the WMS for their pictorial representation, the WFS standard for their download and update.
- Soil data from SoilGrids, the most appropriate standards, are the WMS for their pictorial representation and the WCS for their download.

2.4 Implementation Methodology

The technology platform on which the prototype was implemented is a private-cloud managed by vCloud Director @ VMware software. This software allows the creation of virtual data centers and their use simply and intuitively (Fig. 9). It includes integrated services such as data protection, disaster recovery, data center backup systems, and multi-site data center management. vCloud Director also enables cloud service providers to offer differentiated cloud services on their VMware cloud infrastructure.

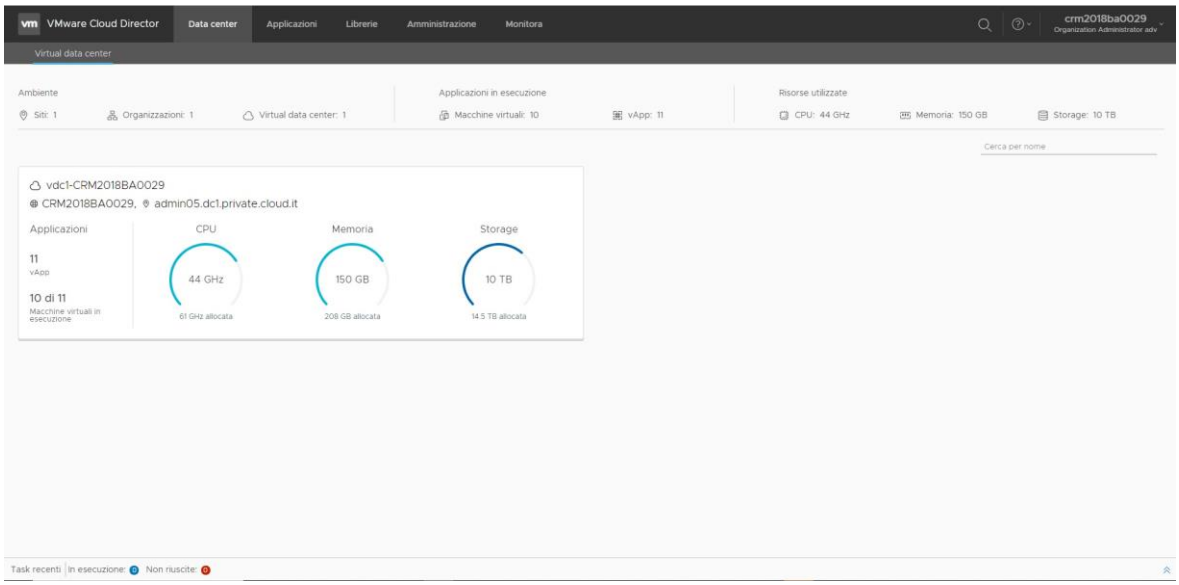

Figure 9: Graphical user interface (GUI) of the vCloud Director implemented on VMware software.

Inside the private-cloud, for the development of the logical functionality described in section 2.3, we have implemented several virtual machines based on the open-source operating system Ubuntu Server version 16.04.

We have developed and installed different software in the virtual machines that implement different logical functionality. The software choice was based on both their open-source availability and their actual compatibility and support with OGC and ISOBUS standards. It should be noted that the implementation of the software was carried out through the docker

system (https://docs.docker.com/) and, in particular, docker-compose that allows instant horizontal scalability. The docker-compose script, in YAML programming language, is illustrated in Appendix Section E. The YAML is a human-readable programming language for data serialization.

The complete list of software and their logical functions is summarized in Table 6.

Table 6: List of logical functions and software implemented in the Service Oriented Architecture (SOA).

Logical Function	**Software**
Database for master, agricultural machinery, satellite, soil meteorological, and geospatial data.	PostGIS (https://postgis.net/)
Processing	52°North WPS (https://52north.org/software/software-projects/wps/)
Metadating	EDI (http://edidemo.get-it.it/)
Catalog of data and metadata	GeoNetwork OpenSource (https://geonetwork-opensource.org/)
Semantic	Apache Jena (https://jena.apache.org/)
Authentication and authorization	Keycloak (https://www.keycloak.org/)
Enabling interoperability for OGC services (WMS, WFS, WCS)	Geoserver (http://geoserver.org/)
Enabling interoperability for SOS services	52°North SOS (https://52north.org/software/software-projects/sos/)
Enabling interoperability for ISOBUS service	ISOAgLib (http://www.isoaglib.com/en/home)
Front End	Heron MC (https://heron-mc.org/)

It is worthy of notice that concerning the interoperability enablement of ISOBUS services, not pre-compiled open source software, but the ISOAgLib (https://www.isoaglib.com/en/home) programming library was detected during this study. This, as part of an ISOBUS system, detects all functions embedded in an electronic communication system according to ISO 11783, such as the display of user interfaces on a virtual terminal (VT) or an activity controller (TC). All functions, according to the standard and established machine interfaces, are already implemented in ISOAgLib. This facilitates access to software development for ISOBUS.

A Front End interface, commonly referred to as WebGIS, has been developed for user access. The development of software for implementing the WebGIS framework and applications has been stimulated by the publication of geospatial data on the web. WebGIS is not strictly an extension of the GIS Desktop Suite, but it belongs to a broader web-based software group (Anderson, et al., 2003). The network is how the Web browser communicates data, and communication is based on a client-server architecture in which two different modules communicate to perform a task. In this way, it is possible to create geospatial web-oriented data publishing and research tools.

The principal benefits of using a WebGIS client are:

- Efficient usability (WebGIS applications are accessible through popular internet browsers);
- Spreading on the network and the opportunity to meet a broader user audience.

Therefore, the position of a similar approach/architecture in the Web-SDSS as a support tool is fundamental. The WebGIS client was developed using the MC (http:/heron-mc.org) heron mapping client, a GeoExt-based free and open-source application (http:/geoext.org). GeoExt is a powerful JavaScript toolkit that combines the OpenLayers web mapping library with the Ext JS user interface to help create powerful web-based desktop geospatial apps based on the JavaScript language. Heron MC offers developers pre-built widgets for browsing, querying, printing, as well as advanced data editing and filtering tools. Heron MC developers are also increasingly generating new functions, so new functions and menus can be introduced to the web interface. OpenLayers is a JavaScript open-source library used in web browsers to visualize interactive maps: It provides an API that allows access to various Web mapping sources, such as OGC protocols, commercial maps (i.e., Google and Bing Maps, etc.), various GIS formats, OpenStreetMap (OSM) project maps, etc. Ext JS is a JavaScript application for desktops, tablets, and smartphones to create feature-rich, cross-platform web apps. In modern browsers, Ext JS leverages HTML5 features while retaining legacy browser compatibility and functionality. It includes hundreds of high-performance UI widgets built to meet the needs of the most complicated web apps.

The WebGIS allows users to get the maps and work with them (Fig. 10-11), according to various features, to gather a substantial degree of interactivity. First of all, the collection of functions available includes the basic ones: the classic pan and zoom on a map, which includes the image scaling and the identification of geospatial objects in the archive and

relevant to the particular request. Then, the selection mode, which can be graphical and geospatial, is another capability. In reality, a collection of geospatial objects characterized by a shared spatial relationship (e.g. contiguity, adjacency, intersection, etc.) and unique descriptive attributes (qualitative/quantitative) are selected in such a WebGIS environment. Objects may be selected depending on the request made in the standard WFS and WCS. One of the common ways to invoke a WPS method is to exploit a web-based program to carry out the elaborations: this is another unique capability of the current implementation of WebGIS. Thus, the WebGIS provides an interface mechanism for (a) load / identifying / selecting the data needed by the procedure, (b) elaborate the data by the chosen procedure, and (c) make the result of the elaboration it accessible to the users.

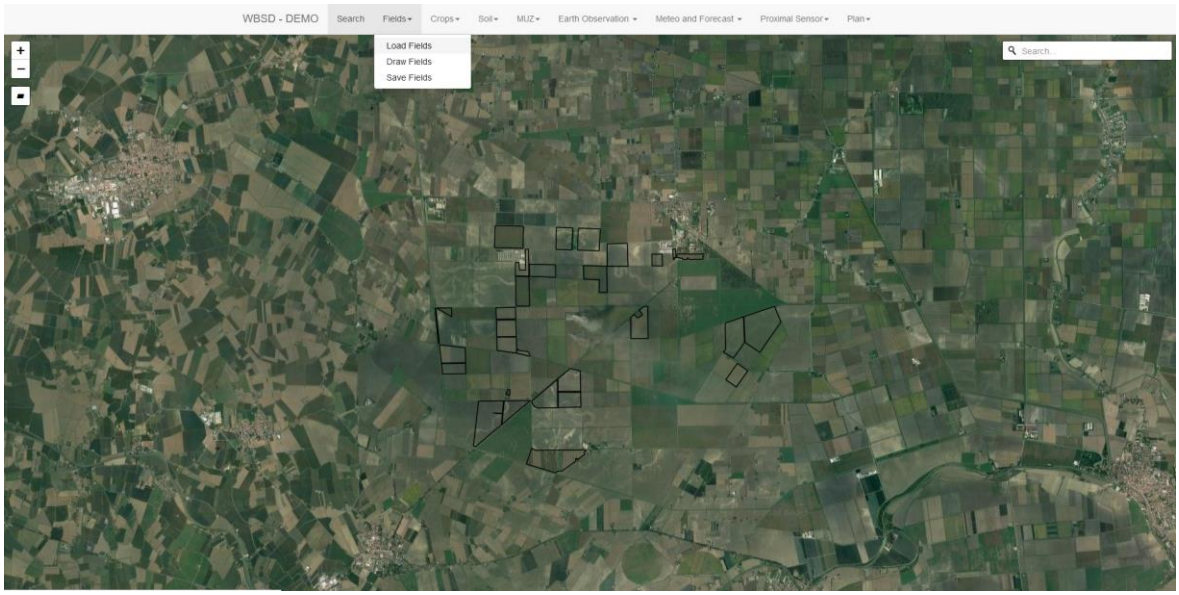

Figure 10: WebGIS interface showing the 27 fields interested by the case study superimposed on a satellite base-map.

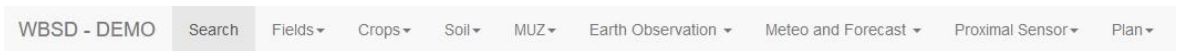

Figure 11: TopBar of the WebGIS that enable the different functionalities of the interface.

2.5 Dynamic delineation of Management Zones (dMUZ)

In the present research, MUZs are not static but dynamically delineated following the cultivated crop's life cycle. To dynamically delineate MUZ, therefore analyzing intra-field

and inter-field variability and sharing the results for each farm, we have developed a methodology that uses ICT and geospatial data processing techniques.

The following workflow summarizes the methodology: a) acquisition and correction of S2 and PS multi-temporal Satellite data; b) elaboration of NDVI and MSAVI2 time-series; c) masking of NDVI and MSAVI2 time-series for each field boundaries; d) storage of all masked data in a multi-dimensional Data Cube (Lewis et al., 2016) in the Data Layer of the Service-Oriented Architecture (SOA) described in section 2.3; e) elaboration of the Maximum Value Composite (MVC); f) Clustering of the MVC for each field; f) sharing of the data via the web application.

To process the NDVI and MSAVI2 from the Satellite data, we used two separate downloads and processing pipelines for the two Satellite constellations, S2, and PS (Fig. 12). The pipeline for S2 uses the Copernicus API Hub to download data, Level 2A (L2A). L2A products provide the bottom of atmosphere reflectance projected to a cartographic projection. The pipeline for PS uses Planet API to download Level 3B (L3B) data from PlanetScope Satellites. L3B data provide top of atmosphere radiance projected to a cartographic projection, so after the download, we applied atmospheric and radiometric corrections using the parameters provided by PlanetScope XML metadata. Once the Satellite data has been downloaded and corrected, the pipeline calculates the NDVI, MSAVI2, the relative time-series and store them.

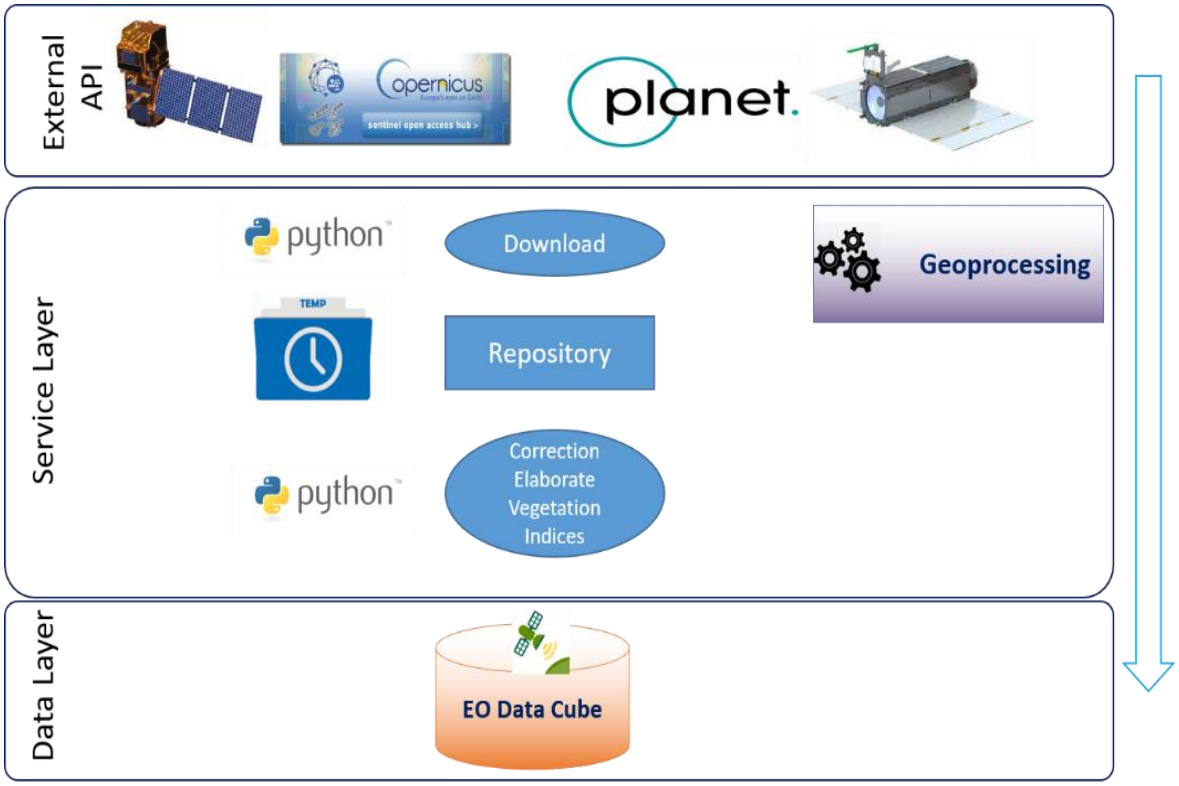

Figure 12: Satellite data acquisition, processing, and storage pipeline.

It is worth to notice that the pipeline for S2 download and calculate NDVI and MSAVI2 for whole Italy (Fig. 13), while the PS pipeline download, correct and calculate NDVI and MSAVI2 for the area of interest (Fig. 14).

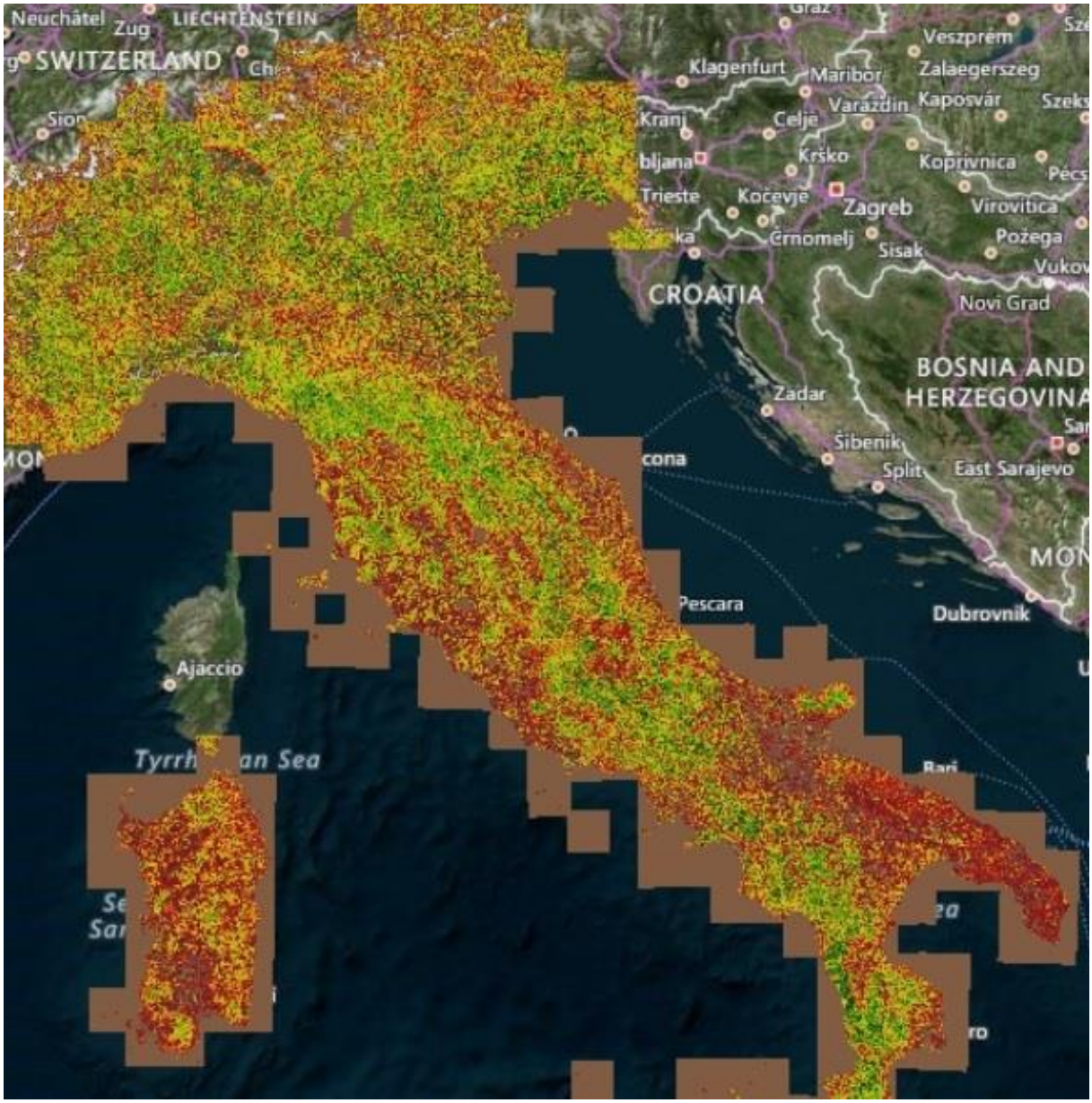

Figure 13: NDVI calculated for the whole territory of Italy from Sentinel-2 pipeline, superimposed on a Satellite basemap.

Figure 14: NDVI calculated for the Area of Interest from the PlanetScope pipeline (Data acquired on 01 June 2019).

The entire pipelines of acquisition, correction, processing, storage of satellite data, and VI has been implemented via Python programming language in the Geoprocessing Service Layer of the Prototype of SOA for PA described in Section 2.3. The Python code of the pipelines is attached in Appendix Section B-C, F-G. Both pipelines allowed us to download and process data throughout the phenological cycle of durum wheat in the season and area of interest with different time intervals that depend on the satellite's revisit time.

After the elaboration of NDVI and MSAVI2, we masked the NDVI and MSAVI2 time-series for each field (Fig. 15) and archived all masked data in a multi-dimensional Data Cube. A Data Cube is a multi-dimensional ("n-D") array of values. The cube is used alongside some measure of interest to represent data. Each dimension represents some attribute in the database and the cells in the data cube represent the measure of interest. For example, they could contain a count of the number of times an attribute combination occurs in a database or the minimum, maximum, quantity, or average value of an attribute. Masking was carried out for all the images that, within every single field, have a cloud cover of 5% or less, the other images of the time series were not taken into account. The masking procedure, in Python programming language, is illustrated in Appendix Section H.

Figure 15: NDVI calculated for 27 fields, starting from PlanetScope (PS) data acquired on 15 may 2019, and superimposed on a Satellite base map.

To identify the inter- and intra-field variability, we processed the MVC for each field. The MVC procedure examine a series of multi-temporal satellite data (compositing period), and, analyzing on a pixel-by-pixel basis each value, maintains only the highest value for each pixel location (Brent 1986). After all, pixels have been evaluated, the procedure creates a new image, the MVC image (Fig. 16). The compositing period, in this case, is the phenological cycle of durum wheat.

The MVC procedure has been implemented via Python programming language in the Geoprocessing Service Layer of SOA Prototype for PA described in Section 2.3. The Python code of the MVC procedure is attached in Appendix Section I.

Figure 16: MVC calculated for 27 fields, starting from PlanetScope (PS data), superimposed on a satellite base map. The MVC is calculated in the time period of the phenological cycle of durum wheat.

After the processing and storage of the MVC, we applied a clustering procedure to delineate the MUZ for each field. One of the main issues facing PA is the evaluation of different algorithms for the delineation of MUZ. Clustering techniques may be a basis for delineating zones (Tagarakis, 2013), but there is no widely accepted method. Cluster analysis is an explorative method which characterizes data in various combination of numerous variables in discrete classes. It is separated into two main categories, non-hierarchical and hierarchical. The most significant, non-hierarchical clustering is k-means, where multi-dimensional information is characterized into k classes. The centroid has a minimum Euclidean distance from each data point in each class. Fuzzy c-means is an extension of clustering of k-means that represents uncertainties related to class boundaries and membership (Nayak et al., 2015). Different authors have demonstrated the usefulness of k-means and fuzzy c-mean clustering techniques for MUZ delineation: Ping et al. (2005) used k-means cluster analysis of yield and soil properties to delineate MUZ in cotton. Molin & Castro 2008 applied fuzzy c-mean on soil data. Yan et al. (2007) delineated MUZ using fuzzy c-means on NDVI, and yield data in cotton. Kyaw et al. (2008) selected NDVI to delineate MUZ in maize and soybean using management zone analyst software (MZA 1.0.1, University of Missouri-Columbia, USA) which performs fuzzy c-means clustering.

The procedure we used to exploit the Fuzzy c-means algorithm on MVC. First of all, the procedure performs a pre-processing on the data that returns an array with standardized values. For many machine learning estimators, standardization of datasets is a common requirement; they could misbehave if the individual characteristics do not look like standard normally distributed data: Gaussian with zero mean and unit variance. We often ignore the shape of the distribution in practice and simply convert the data to center it by removing each function's mean value, then scale it by dividing non-constant characteristics by their standard deviation. For example, several elements used in a learning algorithm's objective function assume that all characteristics are focused around zero and have variance in the same order. Suppose a feature has a variance that is greater than others in order of magnitude. In that case, it might dominate the objective function and render the estimator unable to learn from other features as expected correctly.

Obtained the standardized values, the Fuzzy c-means algorithm is executed that returns an array classified according to the number of clusters chosen. The algorithm clusters the multidimensional data by assigning each point to a membership in each cluster center from 0 to 100 percent. The SciKit-Fuzzy python framework (https://pythonhosted.org/scikit-fuzzy/overview.html) achieves the clustering of Fuzzy c-means via skfuzzy.cmeans class, and the output from this function is used via skfuzzy.cmeans_predict class to identify new data according to the measured clusters (also known as prediction).

Once obtained, the array is classified according to the number of clusters chosen. This is archived both in GeoTIFF and vector format (Fig.17) in the data layer of the architecture described in Section 2.3. The complete procedure of clustering in Python programming language is illustrated in Appendix Section L. The workflow, implemented in the SOA to obtain the MUZ, is summarized in Figure 18.

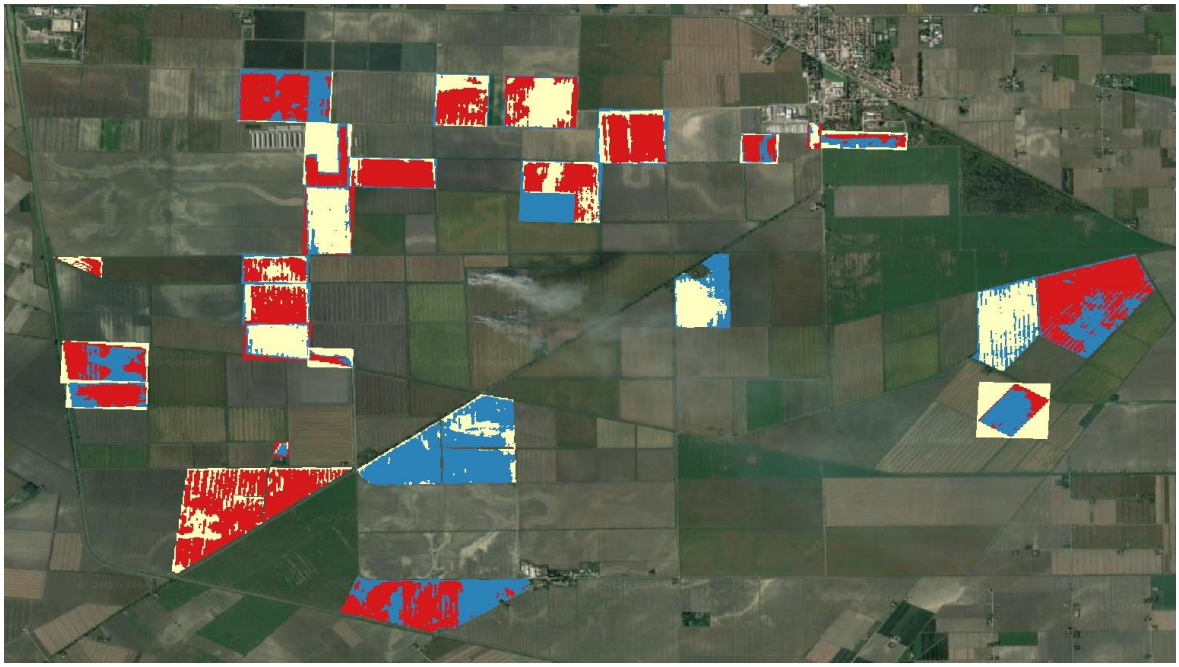

Figure 17: Management unit zones for 27 fields calculated by Fuzzy c-means algorithm on Maximum Value Composite data, superimposed on a Satellite base map.

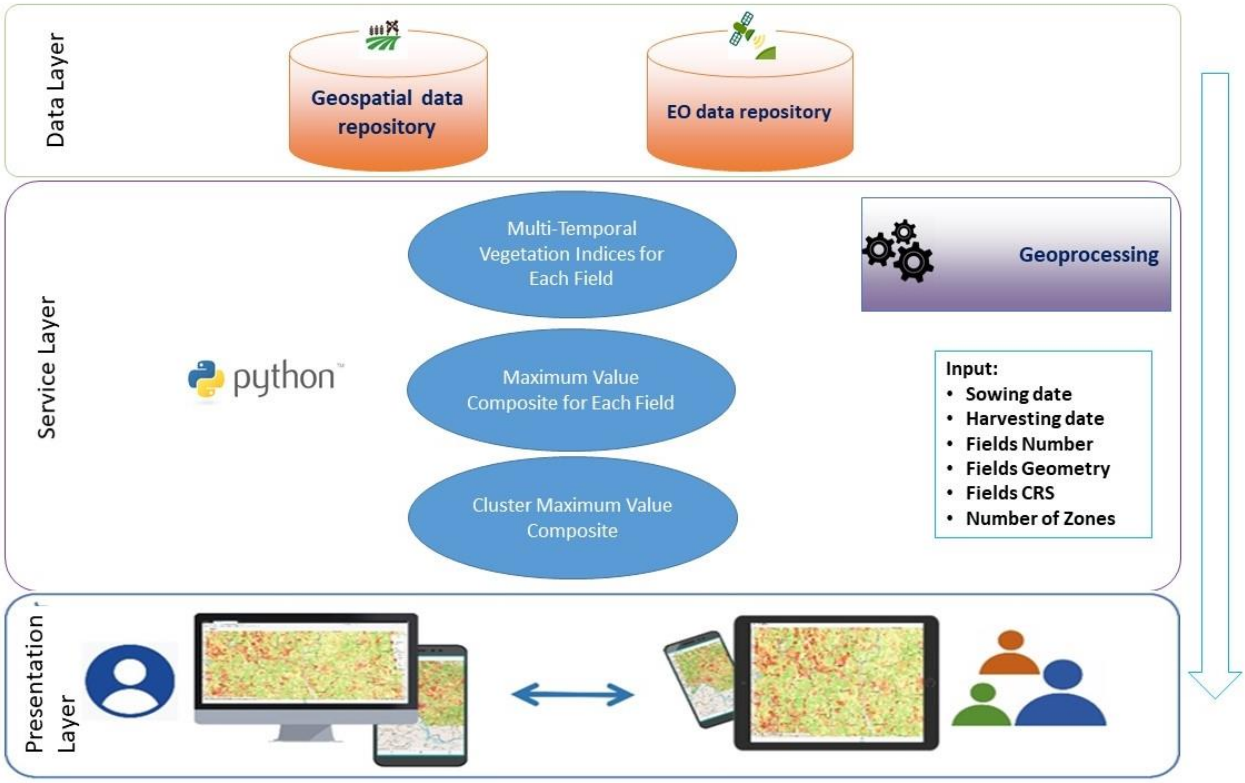

Figure 18: Management unit zones workflow implemented in the Service Oriented Architecture.

3 Results and Discussion

The results of this complex research path, which for its multidisciplinary characteristic has seen the survey embrace many areas that could open as many lines of specialized research, can be summarized as follows:

- Solutions for the acquisition, harmonization and sharing of large amounts of data from heterogeneous sources.
- Solutions for the analysis and interpretation of space-time variability.
- Solutions for the delineation of dynamic management unit zones (dMUZ).

The results are also presented in a perspective of operational relapse of the products, derived from the methodological and cyclical approach of AP, for a validation of the effectiveness of site-specific practices and the adoption of new management proposals that combine economic benefits, quality and environment.

In the present research, an overview of precision agriculture and ICT technologies in favor of PA is first of all given in Chapter 1. In particular, the methodologies of environmental monitoring, space-time variability, methodologies for the dynamic delineation of MUZ are examined. It is also analyzed the problems in the diffusion of AP, and the possible solutions thanks to the use of ICT technologies: SOA and interoperability through the use of international standards. The rapid and intense transformations in data acquisition technologies and in the methods of access, sharing and use of information, offer extraordinary opportunities for the PA in order to improve the quality of production, the degree of environmental sustainability and the efficiency in the use of resources. SOA facilitate the interconnection between service providers and consumers and provide business and process value for the members belonging to the technological infrastructure. In particular, in section 1.6.3 the advantages of using SOA are highlighted, which can be summarized as: decoupling of components, flexibility, debugging simplicity, scalability and reuse. In sections 1.6.4 to 1.6.13 the topic of interoperability in PA is discussed and in particular the OGC and ISOBUS standard web services for access, management (visualization, download, update and creation) and search of PA data are identified. WMS services are identified for the pictorial representation of data such as plots, management unit areas, soil maps, application and yield maps; while WFS services for access, download and update of such data. WCS services for access and download of raster data such as satellite or UAV (NDVI, MSAVI) data and indexes. SOS services to access and download

meteorological or sensor data such as soil moisture and solar radiation data. ISOBUS services for access and download of agricultural machinery data such as yield or application maps, but also to specify to the machinery the application operations or application maps also with VRT technology. Finally, the WPS services for data processing.

In Chapter 2 we describe a case study for the development and implementation of SOA, and the delineation of MUZ on 512 hectares of durum wheat in Italy. The study area is identified and described in section 2.1, and base data acquisition is described in section 2.2. The data acquired are: Fields Boundaries, Crop management plans Plan, Soil chemical and physical properties, Agronomic Operations, Meteo-Climatic, Satellite. All data acquired come from heterogeneous sources and are in heterogeneous formats. The Satellite data, Sentinel 2 and PlanetScope, were acquired by Copernicus API Hub and Planet API in SAFE and geotiff formats respectively. The pedological data from SoilGrid WCS services. Meteo-Climatic data from MeteoBlue services in Json format.

All data have been harmonized and stored in the SOA data layer. It is worth mentioning that the Soil, Satellite and Meteo data have been acquired for the whole Italy for a storage space of about 10 Terabyte. If, on the one hand, the technological and research evolution allows today to identify the inter- and intra-field variability in a more and more pushed way, the processes downstream of the information collection are not to be neglected, that is, the management of huge amounts of data, their integration, the accessibility in real time to this information. Sections 2.3 and 2.4 describe the SOA and its implementation. The SOA is composed by three layers (data, service, presentation) within which the different modules are developed: data archive, semantics, metadata, catalog, services, geo-processing, authentication and authorization, front-end. The implementation was carried out by developing a horizontally scalable container system and the exclusive use of FOSS. In recent years, the gradual emergence of FOSS solutions for the sharing and management of data through the World Wide Web has favored the development of cyber-infrastructures based on geospatial technologies, allowing easy access to data and information to an increasingly wide audience of end users and reducing the costs of sharing and management of computer applications for companies. The usability of information has also become easier thanks to the spread of very different devices. From the desktop personal computer, we quickly moved to the use of laptops, notebooks, smartphones, and tablets to increase flexibility of access to shared information. In this context today the new frontier is that of interactivity, i.e. the

development of applications for pocket devices to be used in open field that allow operators to feed the information flow of corporate information systems.

In section 2.5 are described the methodologies of base data processing and dMUZ. First of all, we have processed the base satellite data to obtain NDVI and MSAVI time-series for each field investigated. The NDVI and MSAVI were masked by field boundaries and produced for each date of the time series. This procedure has allowed us to analyze: the inter and intra-field variability for all 27 fields cultivated with durum wheat; the variation of VI indices over time throughout the phenological cycle. To perform the analysis, we calculated the average value of the VI for each date of the archived-time series. The result of the analysis, shown in Figures 19-23 showed an excellent correspondence with the phenological cycle of durum wheat, as already demonstrated by several authors (Liu & Si 2011, Damian et al., 2020, Zhou et al., 2020). It should be noted, however, in many fields there is a significant lowering in the value of NDVI or MSAVI. Analyzing in detail the image of each single date it was found that this lowering is not related to a particular stress, but is related to cloud cover (Fig. 24). The analysis, in Python programming language, is illustrated at Appendix Section M-N. Once obtained the time-series of VI for each field we applied the MVC procedure, with the composition period of the phenological cycle of durum wheat, to minimize the influence of cloud cover, and identify both intra-field and inter-field variability for all the analyzed fields. The variability can be identified both in spatial (Fig. 25) and analytical terms (Fig. 26-27).

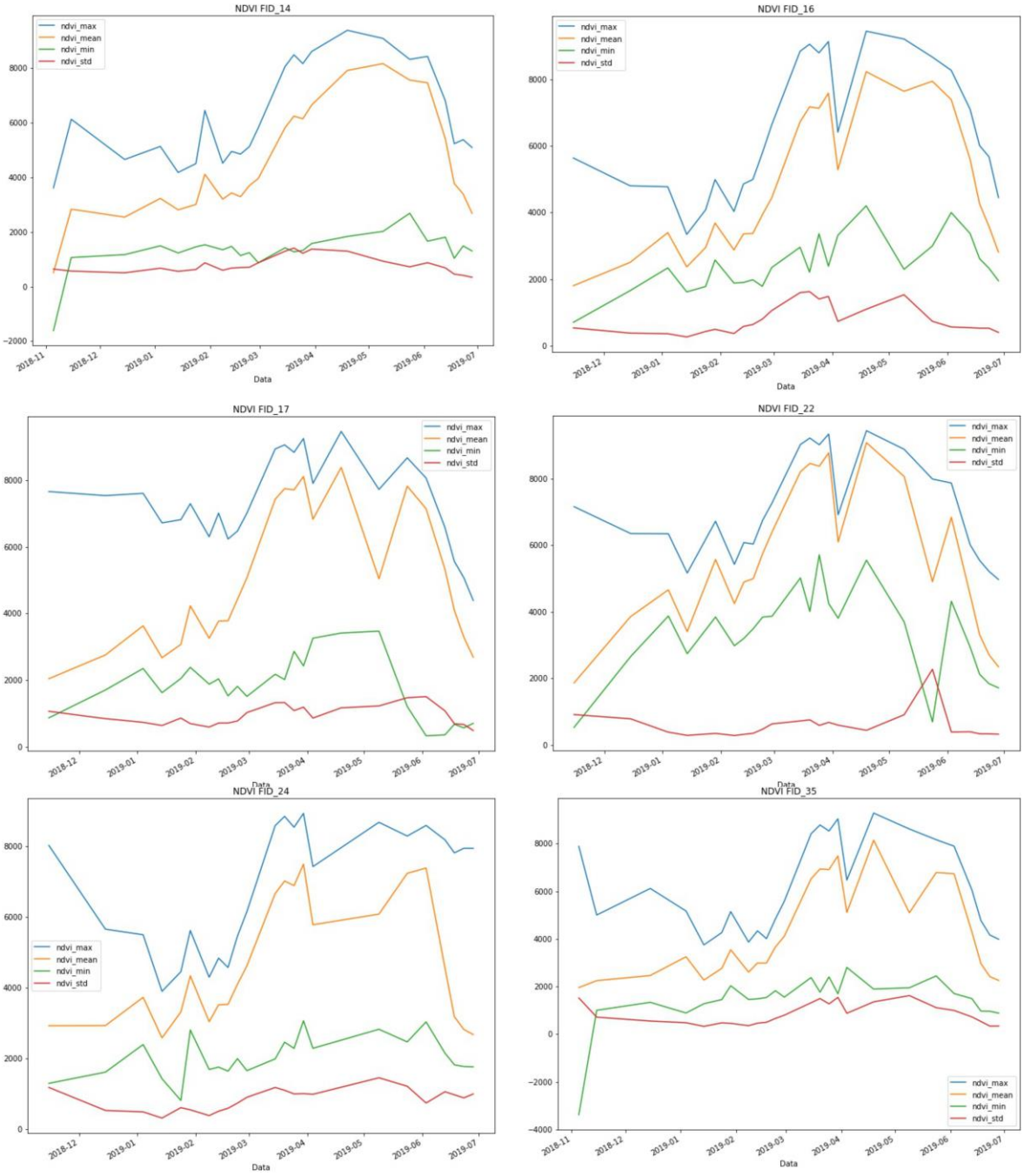

Figure 19: Histograms derived from the NDVI value calculation for each field, during the phenological cycle of durum wheat. The field number is identified above each histogram.

Figure 20: Histograms derived from the NDVI value calculation for each field, during the phenological cycle of durum wheat. The field number is identified above each histogram.

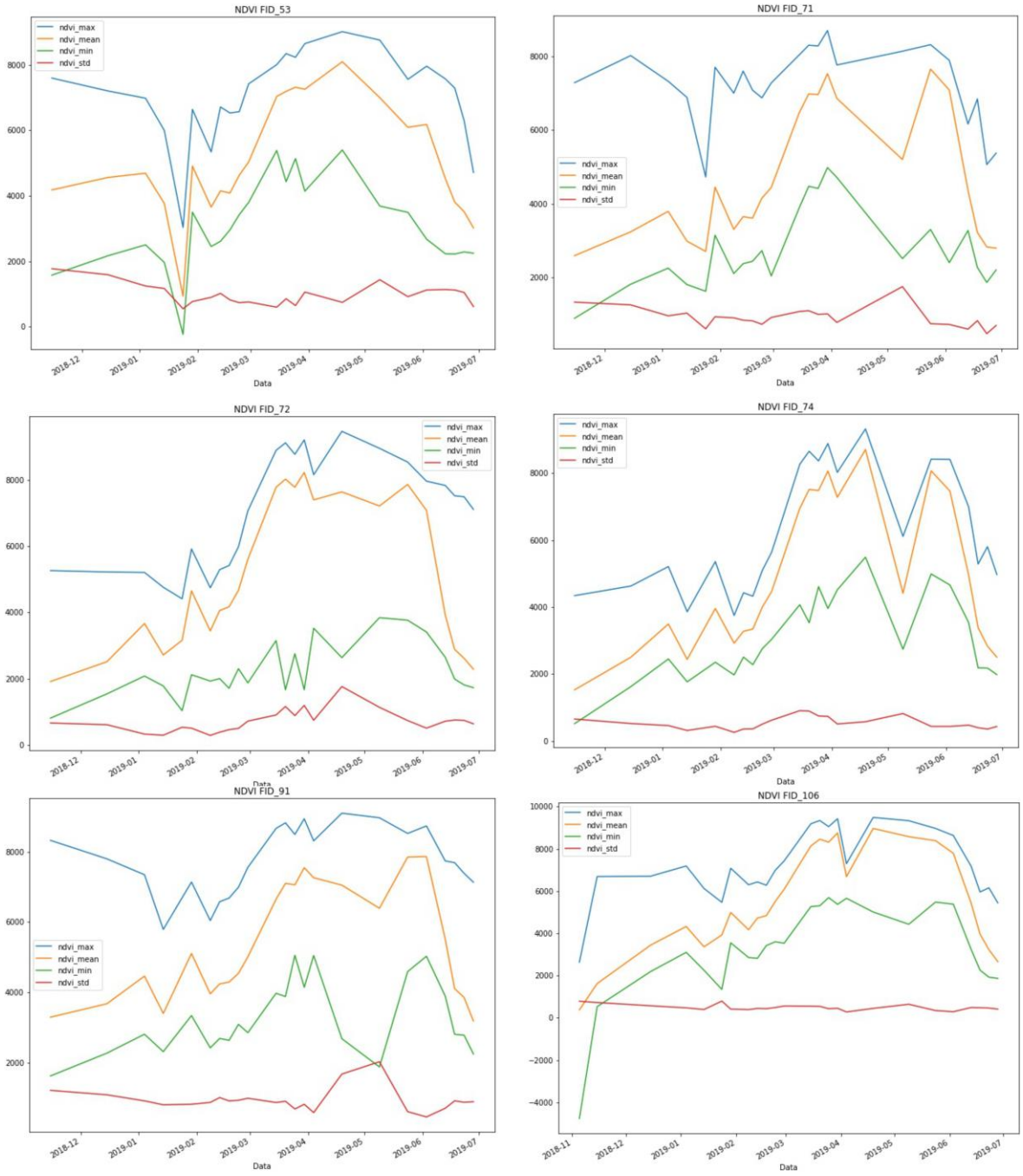

Figure 21: Histograms derived from the NDVI value calculation for each field, during the phenological cycle of durum wheat. The field number is identified above each histogram

Figure 22: Histograms derived from the NDVI value calculation for each field, during the phenological cycle of durum wheat. The field number is identified above each histogram

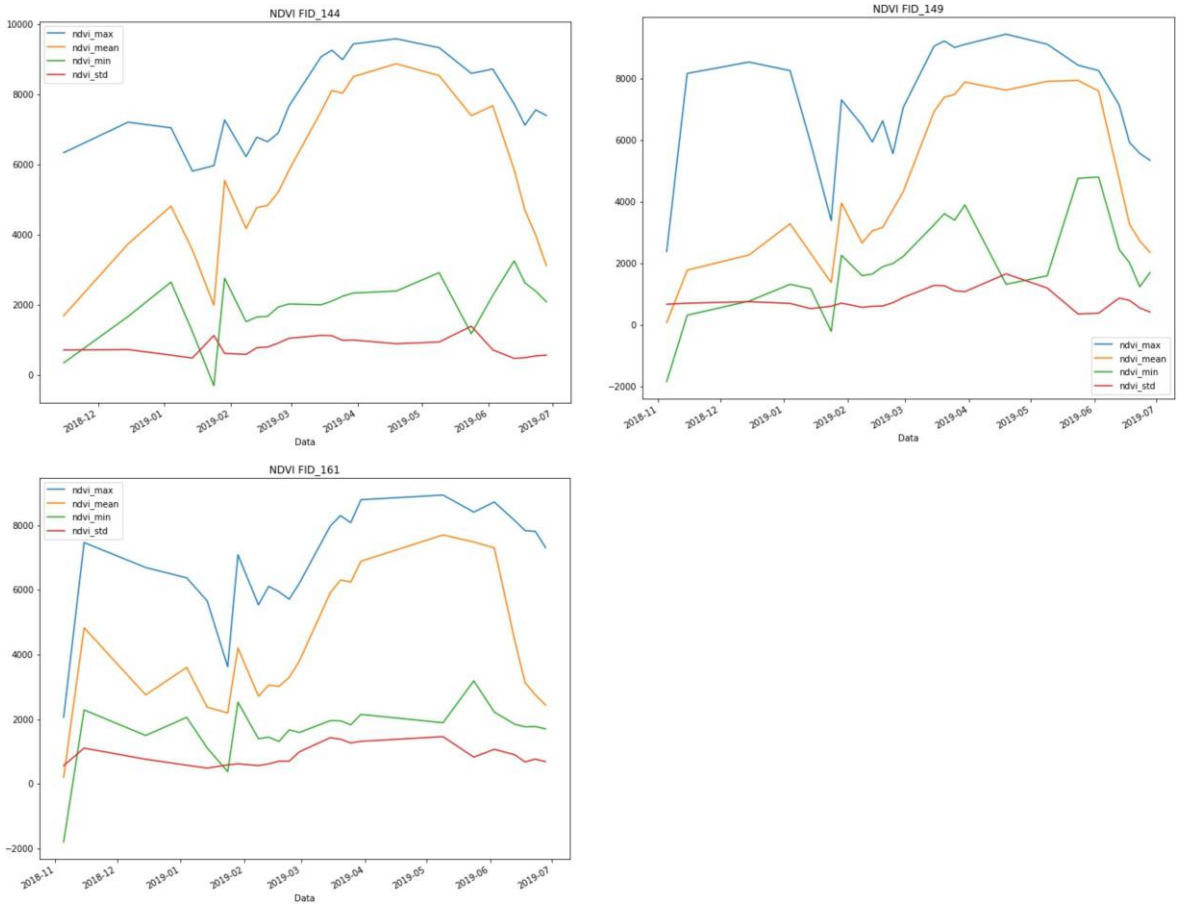

Figure 23: Histograms derived from the NDVI value calculation for each field, during the phenological cycle of durum wheat. The field number is identified above each histogram

Figure 24: Modified soil-adjusted vegetation index (MSAVI) (date 11.04.2019) of the area of interest superimposed on a satellite base-map. The areas in wich the MSAVI is not showed are cloud cover.

Figure 25: Maximum Value Composite (MVC) of 27 fields superimposed on a satellite base-map.

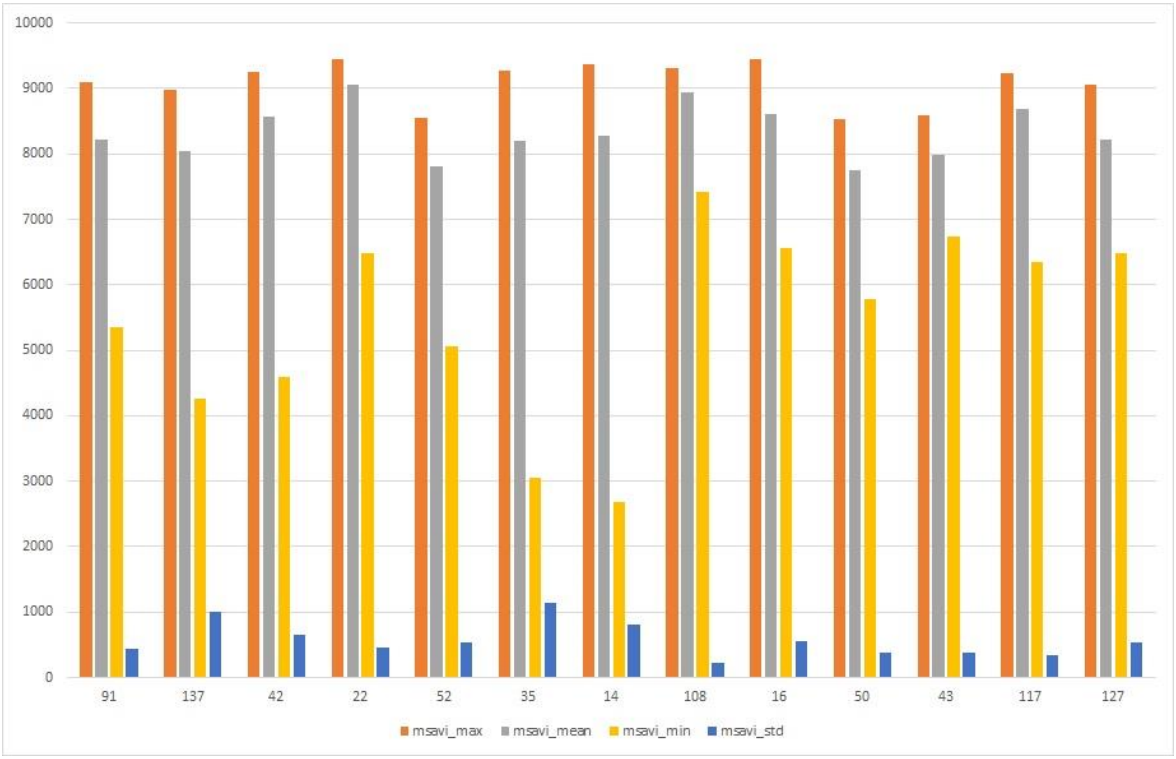

Figure 26: Chart representing, for each field, the maximum, average, minimum and standard deviation values of the MVC. Fields number is identified at the bottom of the graph.

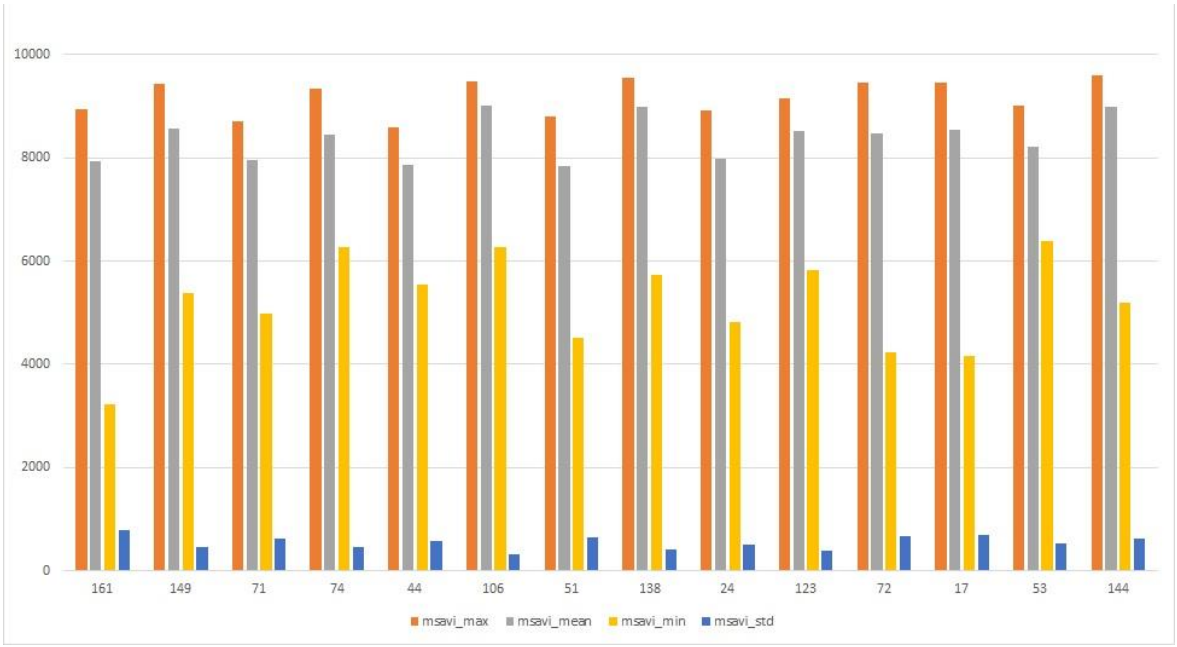

Figure 27: Chart representing, for each field, the maximum, average, minimum and standard deviation values of the MVC. Fields number is identified at the bottom of the graph.

After the elaboration of MVC for all fields, we delineated the MUZ. MUZ are the sub-division of fields featuring an inter-zonal variation, delineated by agronomists for effective PA operations (till, sow, fertilize, harvest) in the fields. To develop MUZ, normally three factors need to be considered: multi-dimensional data to be used as a basis for creating zones, procedure to be used to process the information, and the optimal number of zones that a field should be divided into. Efficient and easy-to-use tools that address all these factors are needed to provide a technology delivery mechanism, the lack of which has been identified as the major obstacle to the wide adoption of PA. A multitude of spatial-temporal information can be used to delineate the MUZ in a field: the physical and chemical properties of the soil; the spatial variability in yields; spectral reflectance elaborated by remote sensing techniques. In recent years, an interest in using VI has emerged. The NDVI, calculated from satellite images, has been widely used globally (Tarnavsky et al., 2008), as it is closely correlated with crop productivity (Sultana et al., 2014; Lopresti et al., 2015; Peralta et al., 2016). Furthermore, satellite images can be freely acquired, enabling large areas monitoring. Based on this premise we exploited the MVC of VI as base data for clustering. Delimitation of MUZ was determined through fuzzy c-means clustering algorithms subdividing each field in three homogeneous zone (Fig. 28).

Figure 28: Management Unit Zones elaborated by fuzzy c-mean on maximum value composite of VI for the phenological cycle of durum wheat.

The integration of geospatial technologies, ICT was the basis for the realization of the WBSD for PA developed during this work. The system is designed for the storage, management, access and sharing of data and information via Web for advanced applications and research. It is aimed at the distributed and integrated use of meteorological, satellite, soil and morphological data and has been tested in the case study described in Chapter 2. The objective was to facilitate the integration of information in a low-cost system through the use of FOSS and thus facilitate the creation of an environment of shared knowledge to support new methods of analysis and the transfer of results for site-specific operational applications. The system also aims to test the flow of information that can be used by operators in the field such as agronomists field researchers also for the processing of MUZ. From a first phase of testing and experimentation, it emerges that the criticality of data and information flows are related to the speed of network connection available to the user. Instead, the usability of information is very dependent on the design and planning of web interfaces, which must be immediate and intuitive even for an operator not expert in the use

of geographic information systems, and accompanied by specific functions that meet the needs of end users.

The design and development phase of the system has led to the definition of a SOA and related WebGIS application functional to the research and experimentation in progress on PA. The customization of the graphical interface of the viewer, the general framework of the system, has allowed to create an integrated and easy to use WebGIS application.

During the research activities, from 2017 to 2020, the following data were stored, managed and processed in the SOA:

- Field boundaries and crop plans: for 512 hectares of durum wheat cultivated in 27 fields. Durum wheat was sown in November 2018 and harvested in June 2019.
- Historical meteorological parameters: Air temperature (maximum, minimum, average), cumulative rainfall, wind speed (maximum, minimum, average) and daily relative humidity (maximum, minimum, average) in the period from November 2018 to June 2019. For a total of about 6,500 archived records.
- Soil data: Bulk density of the fine earth fraction, Cation Exchange Capacity of the soil, Volumetric fraction of coarse fragments (> 2 mm), Proportion of clay particles (< 0.002 mm) in the fine earth fraction, Total nitrogen (N), Soil pH, Proportion of sand particles (> 0. 05 mm) in the fine earth fraction, Proportion of silt particles (≥ 0.002 mm and ≤ 0.05 mm) in the fine earth fraction, Soil organic carbon content in the fine earth fraction, Organic carbon density, Organic carbon stocks. Soil data have been obtained for all Italy for a total of 5 GB of archived data.
- Satellite data: derived from Sentinel-2 satellites for all Italy and from PlanetScope satellites for the area of interest. For a total of about 10 TB of data.

All data and processing results are available both through interoperable web services with OGC standards and through the SOA Front End.

WebGIS or Front End is the component of the system through which you can view and analyze the data. The functions identified and implemented in the WebGIS are currently the following:

- Upload and editing of field boundaries, crop plans, and soil data.
- Visualization and consultation of data with basic functions such as Pan, Zoom, Identify and other advanced functions such as spatial queries.

- Multi-Temporal and Multi-Satellite query and visualization of the different VIs.
- Query and visualization of meteo and soil data.
- Access to all data stored in the database through GIS software by means of OGC standards.
- Download of geospatial data.
- Processing of MUZs according to the previously described methodology.

The user can easily load and draw on the map, field boundaries (Fig. 29), and recall, through a spatial query, information related to a particular field; he can also check meteorological, pedological (Fig. 32), and satellite data (Fig. 30). In addition to the functions of visualization (pan, zoom), interrogation, location search (Fig. 31), it is also possible to recall processing operations for MUZ (Fig.33).

Figure 29: WebGIS interface showing the loading, editing and saving function for fields. In this case we have loaded the 27 fields investigated in the case study. The colour of the polygon represent the variety of wheat cultivated in each field. Description of the colour/variety are show in the legend (upper left of figure)

Figure 30: WebGIS interface showing the monitoring of the fields by vegetation index. In this case we have loaded Normalized Difference Vegetation Index for 15 November calculated from Sentinel 2 satellite.

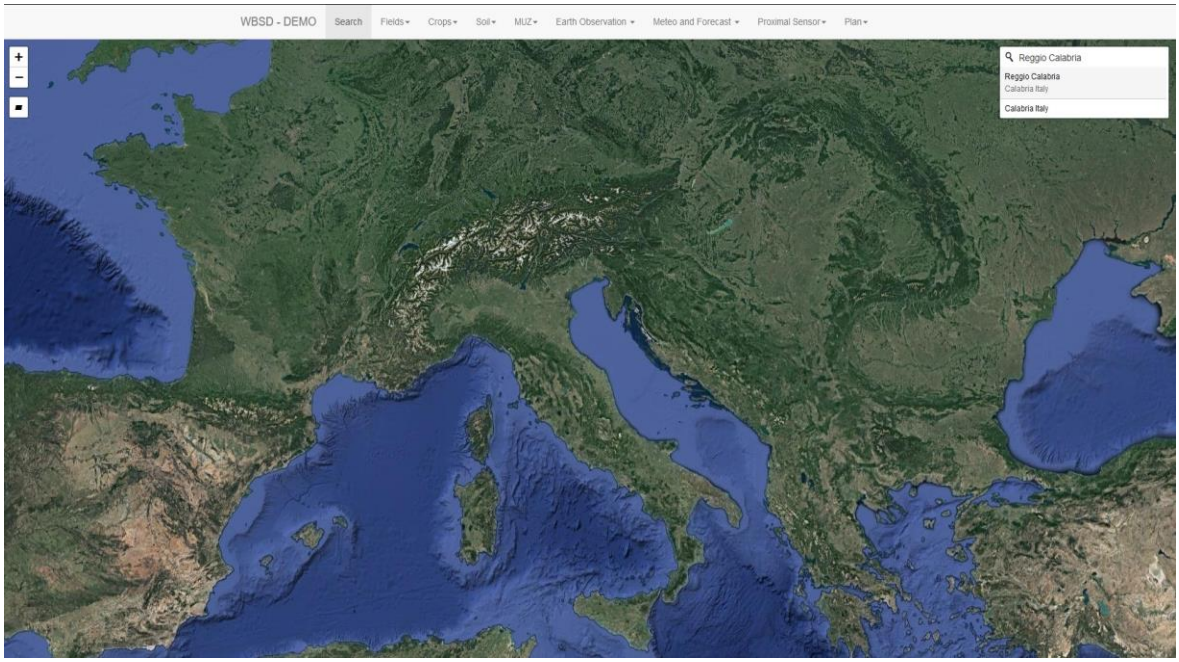

Figure 31: Location search function (upper left of the figure) of the WebGIS.

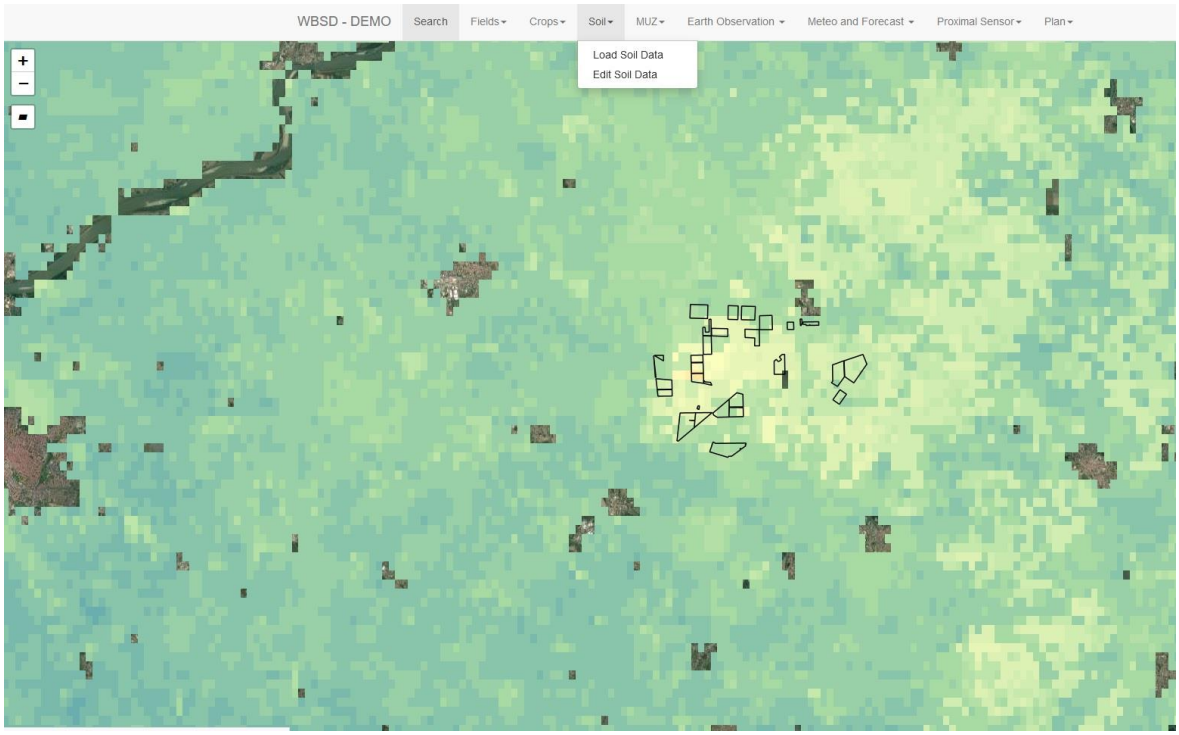

Figure 32: WebGIS interface showing load and edit Soil data.

Figure 33: WebGIS interfaces showing the elaboration function for management unit zones.

Thus, the application provides a tool for data analysis and querying by researchers, agronomists, and farms manager, thus becoming the gateway to the vast wealth of information collected and processed during the research activity to share and efficient use of the knowledge acquired.

The rapid and extreme developments in data collection technology and ways of accessing, sharing, and using information provide the PA sector with excellent opportunities to increase output quality, the degree of environmental protection, and resource usage efficiency. The technologies used in PA are continually expanding for enterprise-level applications. If, on the one hand, the technological and research evolution allows today to identify the inter and intra-field variability in an ever-increasing way, on the other hand, the processes related to the collection of information should not be neglected: integration, real-time accessibility to this information and its portability to devices on agricultural machinery. The accessibility of information has also become more comfortable thanks to the spread of very different devices. From the desktop personal computer, we have rapidly used laptops, notebooks, PDAs, smartphones, and tablets to increase the flexibility of access to shared information. In this context today, the new frontier is that of interactivity or the development of applications for pocket devices to be used in the field that enable operators to feed the flow of information flow of enterprise information systems.

The developed system thus opens new ways to use Web applications in the PA by encouraging the development of new functions of server-side analysis accessible through mobile devices equipped with GPS (Smartphones and PDAs) and usable in the open field for site-specific management. According to interoperability standards, access to the information distributed on the web also sees considerable potential in the direct transfer of prescription maps to devices and software supplied with agricultural machinery used for VRT applications.

4 Conclusions and Future Perspectives

Food production and its sustainability represent a major global challenge for the future. FAO recommends using digital technologies to increase food productivity and sustainability to achieve the "Zero Hunger Goal" objective of the United Nations Agenda 2030. The PA, which by its nature is linked to ICT, can represent a solution to address the challenges and objectives mentioned above.

With each passing year, new generations of EO satellites are delivering increasingly large volumes of data with such extensive global coverage that data limitations are no longer a limiting factor for many applications. New data applications have been delivered through comprehensive research and development activities that provide the tremendous potential to affect significantly critical environmental, economic and social issues, including at local, regional, and global levels. EO's importance is emphasized by such applications, although the challenge is to provide the proper links between data, applications, and users. Despite modern machine and research infrastructures, much archived EO satellite data is underutilized even today. It is difficult for advanced economies to overcome this problem and even more challenging for developing countries interested in using EO satellite data. In many economies, considering conventional local processing and data sharing methods (e.g., scene-based file uploading over the internet), to overcome this "scaling" problem is not technically feasible or financially affordable, as the scale of the data and difficulties in planning, handling, storage, and analysis remain significant obstacles. Luckily, just as the technology for satellite EO has significantly improved, so has ICT. New computing infrastructures, technologies, and data architectures, such as the 'Data Cube,' will solve the data processing and analysis problems emerging from the massive rise in free and open data volumes. Such a solution has tremendous potential to streamline the delivery and management of data for providers while also reducing technological obstacles to consumers' maximum potential to leverage the data.

The PA studies the inter and intra-field space-time variability to propose concrete agronomic management strategies that reduce inputs and increase yields to view environmental sustainability. MUZs are the sub-division of fields featuring an inter-zonal variation, delineated by agronomists for effective PA operations (till, sow, fertilize, harvest) in the fields. The delineation of MUZ is a critical approach that makes it possible, among other benefits, to reduce the cost of production while reducing the environmental impact of

agricultural activities. Usually, three factors need to be considered in order to create MUZ: multi-dimensional data to be used as a basis for creating zones, the method to be used for processing the details, and the optimal number of zones to be divided into a field. Efficient and easy-to-use tools that address all these factors are needed to provide a technology delivery mechanism, the lack of which has been identified as the major obstacle to the wide adoption of PA. A multitude of spatial-temporal information can be used to delineate the MUZ in a field: the physical and chemical properties of the soil (Santi et al., 2016); topographical components (Fraisse et al., 2001); the intrinsic parameters of each crop (El Nahry et al., 2011); crop productivity (Buttafuoco et al., 2010); spectral reflectance elaborated by EO techniques, i.e., vegetation indexes (Zhang et al., 2010, Damian et al., 2020). Different classification algorithms can be used to divide a field into MUZs, and many GIS packages contain the functions necessary for creating MUZ from spatial data, by they can be cumbersome to use and require considerable time to learn (Fridgen et al., 2004). Among the various methodologies developed, it is worth mentioning: fuzzy c-means clustering algorithms to partition spatial observations into clusters (Damian et al., 2020, Li et al., 2007); Multiresolution Segmentation (Karydas et al., 2020); Principal component analysis (PCA) and k-means clustering. (Nogueira et al., 2020).

We have developed a WBSD for the: a) acquisition, storage, and sharing of PA data; b) delineation of MUZ using fuzzy clustering of MVC of VI (NDVI and MSAVI) by GUI.

The results presented here could be the basis for developing a cyber-infrastructure for PA data management and MUZ delineation. It is assumed that MUZ can be used to implement VRA applications (irrigation, fertilization, etc.), which leads to more efficient crop management. It is useful to point out the need for future studies: a) to identify and/or develop open-source software for ISOBUS interoperability; b) to define simplified user interfaces for different PA functionalities to be assigned to users with different profiles (e.g., farmer, agronomist specialist). The implemented system could be expanded through an application, WebAPP, for pocket devices and reach the operators in the field by providing complementary tools and support to the operational and monitoring strategies. The WebApp would allow users to enter geo-referenced information related to a single plant, row, or area and store it in the system.

The development of a shared knowledge environment that can support advanced research for economic and environmental sustainability is a concept that apparently seems very

simple to apply in contexts where the actors involved operate in the same sector. The realization of common paths and the construction of an infrastructure based on the potential offered by the integration of geospatial sciences and ICT, through which converge data acquired from advanced technologies and information, and arrive at new knowledge frameworks for the reduction of the impacts of agriculture on the environment, in practice presents many difficulties. These criticalities are not only related to the search for appropriate technological solutions to create the connection between data and devices but instead are represented by the ability to involve actors in the development processes of diversified applications or cross-cutting services.

Following, we trace the efforts that should be made to improve the frameworks of knowledge and information flows:

a) Technical and semantic interoperability

Considering the wide range of data formats managed by the system, we can affirm that in light of the IT tools available in the world of geomatic applications, we have now reached good levels of technical data interoperability. However, in order to facilitate the semantic understanding of the data, in multidisciplinary working groups, many efforts should still be made to overcome those difficulties related to the use of languages that characterize the different research disciplines and the different professional and productive sectors. An important contribution in this direction has been given by the development of the FAO AGROVOC thesaurus. This thesaurus can be a standard reference for all PA and farm management systems for all software applications to understand the data and obtain semantic interoperability.

b) Data management and modeling in precision agriculture

The methodologies adopted in PA require, as already seen, the collection and management of a large number of data, especially if collected at very close time intervals or in real-time. Therefore, the management and processing of data require the adoption of computer tools much more complex than traditional spreadsheets or personal databases. Moreover, understanding the relationships between different environmental variables is not only a problem of integrating data formats. The adoption of new analysis methodologies requires data modeling that must be tested and experimented with according to the objectives of the investigation. The adoption of relational databases and data modeling tools can help better

manage and explore multi-scale and multi-temporal data for the development of new site-specific analysis methodologies. In this work, the adoption of Open Source databases allows many users to use a common platform to query the system, extract the data they are interested in and allow further manipulation and processing of the database at no cost. Nevertheless, it is not so evident that in the scientific and professional world, there is such widespread knowledge on the use of these tools that allow more complex analysis of the information heritage that today the new technologies of data acquisition make available. To solve these critical issues, the best investment to make is in increasing the skills of human resources, encouraging advanced training of technical and scientific personnel, and contributing at the same time to the preparation of new professionals needed for the transfer of innovation in the agricultural sector and beyond.

c) Data and information flow

Data flow is one of the main bottlenecks for the experimentation and operational adoption of products and results coming from the adoption of new technologies and the results of advanced research in agriculture. The data flow is often limited to the transmission between the acquisition system and the related management software (e.g., between meteorological sensors and field sensor management software) and, at most, in very advanced cases, with the same integration into the farm management software. The use of interoperable web services with internationally recognized standards, for the access and distribution via the web of spatial data, and not only, but thanks also to the availability of standard web services, such as those of OGC, is a solution adopted today to receive, visualize and distribute in real-time data via the web. In any case, it is necessary for the development and dissemination of specific standards for precision agriculture, already addressed in international research, for interoperability in agriculture and mechanical engineering.

d) Research perspectives in precision agriculture

The research needs funds to be carried out and to reach developments and results that see the experiments' real operational impact. Also, this work has required resources, at least in the realization of the application part, to be tested and evaluated regarding the technical and methodological choices adopted in implementing a distributed service. The criticality of the stages of research progress is due to the often-uncertain prospects of continuity of investigations, especially in research paths so long and complex and that require, among other things, extensive use of technology. A fundamental role for higher investment in

research and development is undoubtedly that of the agricultural sector's development and innovation policies. Moreover, the European directives and new European agricultural policy on research and the environmental sustainability of agriculture can undoubtedly give hope on innovative paths to follow.

In a period of financial crisis, such as the one that Europe, particularly Italy, is going through, it is hoped that we will begin to invest in research and innovation as a driving force for a resumption of economic growth. Furthermore, this also in the perspective of creating new models of development for better environmental sustainability of agriculture.

References

Aschbacher, J. (2017). ESA's earth observation strategy and Copernicus. In Satellite earth observations and their impact on society and policy, pp. 81–86. Springer, Singapore

Batjes, N. H., Ribeiro, E., & Van Oostrum, A. (2020). Standardised soil profile data to support global mapping and modelling (WoSIS snapshot 2019). Earth System Science Data, 12(1), 299-320.

Brent N. Holben (1986) Characteristics of maximum-value composite images from temporal AVHRR data, International Journal of Remote Sensing, 7:11, 1417-1434

FAO, (2017). Information and Communication Technology (ICT) in Agriculture, Rome: s.n.

Adrian, A.M.; Norwood, S.H.; Mask, P.L. (2005). Producers' perceptions and attitudes toward precision agriculture techologies. Computers and Electronics in Agriculture 48: 256-271.

Ahmad, F.(2012). Spectral vegetation indices performance evaluated for Cholistan Desert. J. Geogr. Reg. Plann. 5(6), 165–172

Anderson, G., Moreno-Sanchez, R.: Building Web-based spatial information solutions around open specifications and open source software. Trans. GIS 7, 447–466 (2003)

Backes, M.M Dorshalg, D.; Plumer, L. (2003). A metadata profile for precision agriculture based on ISO 19115. In: Proceedings of the 4th European Conference on Precision Agriculture, pp.41-46.

Basso B., Bertocco M., Sartori L., Martin E.C. (2007). Analyzing the effects of climate variability on spatial pattern of yield in a maize-wheat-soybean rotation. European Journal of Agronomy 26: 82-91.

Blackmoore S. (2000). The interpretation of trends from multiple yield maps. Computers and Electronics in Agriculture 26: 37-51.

Botts, M., Percivall, G., Reed, C., & Davidson, J. (Eds.). (2006). OGC ® sensor web enablement: Overview and high level architecture. Wayland, MA, USA: Open Geospatial Consortium, Inc.

Cambridge Systematics, Inc. with Bentley Systems, Inc., Info Tech, Inc., Michael Baker Jr. Inc., & Campbell, C. E. (2006). XML schemas for exchange of transportation data. NCHRP 20-64 final report.

Carr P.M., Carlson G.R., Jacobsen J.S., Nielseng G.A., Skogley E.O. (1991). Farming soils, not fields. A strategy for increasing fertilizer profitability. Journal of Production Agriculture 4: 57-61.

Casadesus, J., Biel, C., & Bonany, J. (2007). Architecture and requirements for sensor-controlled irrigation. In C. Parker (Ed.), Proceedings of EFITA/WCCA 2007, Glasgow Caledonian University.

Castrignanò A., Landrum C., De Benedetto D. (2015). Delineation of Management Zones in Precision Agriculture by Integration of Proximal Sensing withg Multivariate Geostatistics. Examples of Sensor Data Fusion. Agricolturae Conspectus Scientificus 80: 39-45

Cox, S., Daisey, P., Lake, R., Portele, C., & Whiteside, A. (Eds.). (2003). OpenGIS ® geography markup language (GML) implementation specification, Version 3.1.1.

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., & Weerawarana, S. (2003). The next step in web services. Communications of the ACM, 46(10), 29–34.

De la Beaujadiere, J. (Ed.). (2006). OpenGIS ® web map server implementation specification. Wayland, MA, USA: Open Geospatial Consortium, Inc.

Damian, J. M., Pias, O. H. D. C., Cherubin, M. R., Fonseca, A. Z. D., Fornari, E. Z., & Santi, A. L. (2020). Applying the NDVI from satellite images in delimiting management zones for annual crops. Scientia Agricola, 77(1).

EC (European Commission). (2007). Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE).

El Nahry, A. H., Ali, R. R., & El Baroudy, A. A. (2011). An approach for precision farming under pivot irrigation system using remote sensing and GIS techniques. Agricultural Water Management, 98(4), 517-531.

Erl, T. (2005). Service-oriented architecture (Vol. 8). Pearson India.

Esfandiari, M., Ramapriyan, H., Behnke, J., Sofinowski, E. (2007). Earth observing system (EOS) data and information system (EOSDIS) evolution update and future. In 2007 IEEE International Geoscience and Remote Sensing Symposium, pp. 4005–4008, IEEE

Fleming K.L., Westfall D.G., Wiens D.W., Brodahl M.C. (2000). Evaluating farmer defined management zone mapos foir variable rate fertilizer application. Precision Agriculture 2: 201-215.

Fountas, S., Blackmore, S., Ess, D., Hawkins, S., Blumhoff, G., Lowenberg-Deboer, J., et al. (2005). Farmer experience with precision agriculture in Denmark and the US eastern corn belt. Precision Agriculture, 6(2), 121–141.

Fountas, S.M Blackmore, S.B.; Petersen, S.M. (2002). A new methodology for decision analysis on precision farming based on user's experience. In: Proceedings of the 6th International Conference on Precision Agriculture, pp. 1640-1654.

Fraisse, C. W., Sudduth, K. A., & Kitchen, N. R. (2001). Delineation of site-specific management zones by unsupervised classification of topographic attributes and soil electrical conductivity. Transactions of the ASAE, 44(1), 155.

Fridgen, J.J.; Kitchen, N.R.; Sudduth, K.A.; Drummond, S.T.; Wiebold, W.J.; Fraisse, C.W. 2004. Management zone analyst (MZA): Software for subfield management zone delineation.Agronomy Journal 96: 100-108.

Giuliani, G., Chatenoux, B., Piller, T., Moser, F., Lacroix, P. (2020). Data Cube on Demand (DCoD): generating an earth observation data cube anywhere in the world. Int. J. Appl. Earth Obs. Geoinf. 87, 102035

Gessler P.E., Chadwick O.A., Chamaran F., Althouse L., Holmes K. (2000). Modeling soil-landscape and ecosystem properties using terrain attributes. Soil Science Society of America Journal 64: 2046 - 2056.

Gliessman, S. T. 2000. Agroecology: Ecological Processes in Sustainable Agriculture. Lewis Publishers, an imprint of CRC Press, Boca Raton, FL.

Griffin, T.W.; Lowenberg-DeBoer, J; Lambert, D.M.; Peone, J; Payne, T.; Daberkow, S.G. (2004). Adoption, Profitability and Making Better Use of Precision Farming Data. Staff Paper #04-06. Department of Agricultural Economics, Purdue University.

Griffin, T.W., Lowenberg-DeBoer, J., 2005. Worldwide adoption and profitability of precision agriculture: implications for Brazil. Revista de Politica Agricola 14 (4), 20–38.

Gröger, G., Kolbe, T. H., & Czerwinski, A. (Eds.). (2006). Candidate OpenGIS ® CityGML implementation specification. Wayland, MA., USA: Open Geospatial Consortium, Inc.

Grunwald, S., Thompson, J. A., & Boettinger, J. L. (2011). Digital soil mapping and modeling at continental scales: Finding solutions for global issues. Soil Science Society of America Journal, 75(4), 1201-1213.

Heier, C., & Kiehle, C. (2006). Automatisierte Liegenschaftsauskunft mittels OGC Web Processing Service (Automated cadastre disclosure using the OGC web processing service). Geo-Informationssysteme, 19(7), 12–16.

Jarfe, A., & Werner, A. (2000). Development of a GIS-based management system for precision agriculture. In H. H. Tok (Ed.), Agroenviron 2000: Proceedings of the 2nd international symposium on new technologies for environmental monitoring and agro-applications (Tekirdağ University, Turkey, pp. 121–125). ISBN 975-374-29-8.

Karydas, C., Iatrou, M., Iatrou, G., & Mourelatos, S. (2020). Management Zone Delineation for Site-Specific Fertilization in Rice Crop Using Multi-Temporal RapidEye Imagery. Remote Sensing, 12(16), 2604.

Kiehle, C., Greve, K., & Heier, C. (2006). Standardized geoprocessing—taking spatial data infrastructures one step further. In J. Suárez & B. Márkus (Eds.), Proceedings of the 9th AGILE conference on geographic information science (Visegrád, Hungary, pp. 273–282).

Kim, Y., & Evans, R. G. (2009). Software design for wireless sensor-based site-specific irrigation. Computers and Electronics in Agriculture, 65(1), 159–165.

Kitchen, N. R., Snyder, C. J., Franzen, D. W., & Wiebold, W. J. (2005). Educational needs of precision agriculture. Precision Agriculture, 3(4), 341–351.

Kitchen, N. R. (2008). Emerging technologies for real-time and integrated agriculture decisions. Computers and electronics in agriculture (61) 1-3

Kyaw, T., Ferguson, R. B., Adamchuk, V. I., Marx, D. B., Tarkalson, D. D., & McCallister, D. L. (2008). Delineating site-specific management zones for pH-induced iron chlorosis. Precision Agriculture, 9, 71–84.

Korduan, P. (2003). Standardization in data managment to increase interoperability of spatial precision agriculture data. In: Proceedings of the 4th European Conference on Precision Agriculture, pp.323-328.

Korduan, P., & Nash, E. (2005). Integration von ISO- und agroXML in GML (Integration of ISO- and agroXML in GML). In A. B. Cremers, R. Manthey, P. Martini, & V. Steinhage (Eds.), Proceedings of the 35th annual meeting of the Gesellschaft für Informatik e.V., Bonn (pp. 375–379).

Korotkiy, M. (2005). Towards an ontology-enabled service-oriented architecture. ICSOC 2005, 1.

Kunisch, M., Frisch, J., Martini, D., Schmitz, M., & Böttinger, S. (2009). Stand der Entwicklung von agroXML (State of development of agroXML). In R. Bill, P. Korduan, & L. M. Theuvsen (Eds.), Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung (Challenges for agricultural informatics through globalisation and climate change), proceedings of the 29th GIL conference, 9–10 March 2009 (pp. 89–92), Rostock. Gesellschaft für Informatik, Bonn, Germany.

Lanucara S., Oggioni A., Di Fazio S., Modica G. (2020) A Prototype of Service Oriented Architecture for Precision Agriculture. In: Coppola A., Di Renzo G., Altieri G., D'Antonio P. (eds) Innovative Biosystems Engineering for Sustainable Agriculture, Forestry and Food Production. MID-TERM AIIA 2019. Lecture Notes in Civil Engineering, vol 67. Springer, Cham.

Lanucara S., Modica G. (2021) Detection and Sharing of Anomalies in the Vegetative Vigor of Durum Wheat in Italy. In: Bevilacqua C., Calabrò F., Della Spina L. (eds) New Metropolitan Perspectives. NMP 2020. Smart Innovation, Systems and Technologies, vol 178. Springer, Cham.

Lark, R.M. 1998. Forming spatially coherent regions by classification of multivariate data: An example from the analysis of maps of crop yield. Int. J. Geogr. Inf. Sci. 12:83–98.

Lautenbacher, C.C. (2006). The global earth observation system of systems: Science serving society. Space Policy 22(1), 8–11

Lewis, A., Lymburner, L., Purss, M.B., Brooke, B., Evans, B., Ip, A., Oliver, S. (2016) Rapid, highresolution detection of environmental change over continental scales from satellite data the Earth Observation Data Cube. Int. J. Digital Earth 9(1), 106–111

Li, Y.; Zhou, S.; Feng, L.; Houng Yi, L. 2007. Delineation of sitespecific management zones using fuzzy clustering analysis in a coastal saline land. Computers and Electronics in Agriculture 56: 174-186

Liu, J., & Si, W. (2011, July). Using NDVI and air temperature to monitoring winter-wheat phenology in Xingtai, Hebei, China. In 2011 International Conference on Control, Automation and Systems Engineering (CASE) (pp. 1-4). IEEE.

Lowenberg-DeBoer, J. and Swinton, S. 1997. Economics of site-specific management in agronomic crops. In: The State of Site-Specific Management for Agriculture USA. edited by F. Pierce and E. Sadler, (ASA-CSSA-SSSA, Madison, Wisconsin, USA), pp. 369–396.

Lokhorst, K., Goense, D., Ipema, B., de Vries, H., & van Grootheest, J.-E. (2008). Agriculture benefit from the LOFAR infrastructure. In L. Kooistra & A. Ligtenberg (Eds.), Proceedings workshop sensing a changing world, 19–21/11/2008 (pp. 96–99).

Lopresti, M.F.; Di Bella, C.M.; Degioanni, A.J. 2015. Relationship between MODIS-NDVI data and wheat yield: a case study in northern Buenos Aires province, Argentina. Information Processing in Agriculture 2: 73-84.

Lutticken, R.E. (2000). Development of an Internet-based communication and information network to progress the implementation of precision agriculture. In Proceedings of the 5th International Conference on Precision Agriculture, 14 p.

Matese A., Toscano P., Di Gennaro S.F., Genesio L., Vaccari F.P., Primicerio J., Belli C., Zaldei A., Bianconi R., Gioli B. (2015). Intercomparison of UAV, Aircraft and Satellite Remote Sensing Platforms for Precision Viticulture. Remote Sensing 7: 2971-2990.

McBratney, A., Whelan, B., Ancev, T., & Bouma, J. (2005). Future directions of precision agriculture. Precision Agriculture, 6(1), 7–23.

McCann B.L., Pennock D.J., Vann Kessel C., Walley F.L. (1996). The development of management units for site specific farming. In P.C. Robert et al (Eds.), Proceedings of the 3rd International conference on precision agriculture, Madison, WI: ASA, CSSA, and SSSA, pp. 295-302.

Modica G. et al. (2016) Land Suitability Evaluation for Agro-forestry: Definition of a Web-Based Multi-Criteria Spatial Decision Support System (MC-SDSS): Preliminary Results. In:

Gervasi O. et al. (eds) Computational Science and Its Applications -- ICCSA 2016. ICCSA 2016. Lecture Notes in Computer Science, vol 9788. Springer, Cham

Moore I.D. Gessler P.E., Peterson G.A. (1993) – Soil attribute prediction using terrain analysis. Soil Science Society of America Journal 57: 443 -452.

Molin, J. P., & Castro, C. N. (2008). Establishing management zones using soil electrical conductivity and other soil properties by the fuzzy clustering technique. Scientia Agricola, 65(6), 567–573.

Morais, R., Fernandes, M. A., Matos, S. G., Serôdio, C., Ferreira, P. J. S. G., & Reis, M. J. C. S. (2008). A ZigBee multi-powered wireless acquisition device for remote sensing applications in precision viticulture. Computers and Electronics in Agriculture, 62(2), 94–106.

Morari F., Castrignanò A., Pagliarin C. (2009). Application of Multivariate Geostatistics in Delineating Managment Zones within a gravelly vineyard using geo-electrical sensors. Computers and Electronics in Agriculture 68: 97-107.

Mulla D.J. (2013) Twenty five years of remote sensing in precision agriculture: Key adavances and remaining knowledge gaps. Biosystems Engineering 114: 358-371.

Murakami, E.; Ribeiro, L.C.M.; Saraiva, A.M.; Cugnasca, C.E. (2002). An infrastrucutre for development of information systems for precision agriculture. In: Proceedings of the 6th International Conference on Precision Agriculture, pp. 1712-1722.

Nash, E., Bobert, J., Wenkel, K.-O., Mirschel, W., & Wieland, R. (2007). Geocomputing made simple: Service-chain based automated geoprocessing for precision agriculture. In U. Demšar (Ed.), Proceedings of the 9th international conference on geocomputation, National University of Ireland, Maynooth.

Nayak J., Naik B., Behera H.S. (2015) Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014. In: Jain L., Behera H., Mandal J., Mohapatra D. (eds) Computational Intelligence in Data Mining - Volume 2. Smart Innovation, Systems and Technologies, vol 32. Springer, New Delhi.

Nogueira Martins, R., Magalhaes Valente, D. S., Fim Rosas, J. T., Souza Santos, F., Lima Dos Santos, F. F., Nascimento, M., & Campana Nascimento, A. C. (2020). Site-specific Nutrient Management Zones in Soybean Field Using Multivariate Analysis: An Approach Based on Variable Rate Fertilization. Communications in Soil Science and Plant Analysis, 51(5), 687-700.

Nolan S.C, Goddard T.W., Lohstraeter G. (2000). Assessing management units on rolling topography. In: Proceedings of the 5th International Conference on precision and other resource management. Madison, WI: ASA, CSSA, and SSSA.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on (pp. 3-12). IEEE.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. Computer, 40(11).

Pedersen, S.M.; Fountas, S.; Blackmore, S.; Pedersen, J.L.; Pedersen, H.H. (2003). Adoption of precision farming in Denmark. In: Proceedings of the 4th European Conference on Precision Agriculture, pp.533-538.

Pedersen, S. M., Fountas, S., Blackmore, B. S., Gylling, M., & Pedersen, J. L. (2004). Adoption and perspective of precision farming in Denmark. Acta Agriculturae Scandinavica Section B, Soil and Plant Science, 54(1), 2–6.

Pepe M., Candiani G., Pavesi F., Lanucara S., Guarneri T., Caceffo D. (2019) SDI and Smart Technologies for the Dissemination of EO-Derived Information on a Rural District. In: Calabrò F., Della Spina L., Bevilacqua C. (eds) New Metropolitan Perspectives. ISHT 2018. Smart Innovation, Systems and Technologies, vol 100. Springer, Cham.

Peralta, N.R.; Assefa, Y.; Du, J.; Barden, C.J.; Ciampitti, I.A. 2016. Mid-season high-resolution satellite imagery for forecasting site-specific corn yield. Remote Sensing 8: 1-16.

Pierce, F. J., & Elliot, T. V. (2008). Regional and on-farm wireless sensor networks for agricultural systems in Eastern Washington. Computers and Electronics in Agriculture, 61(1), 32–43.

Ping, J. L., Green, C. J., Bronson, K. F., Zartman, R. E., & Dobermann, A. (2005). Delineating potential management zones for cotton based on yields and soil properties. Soil Science, 170(5), 371–385.

Planet Team. Planet Application Program Interface: In Space for Life on Earth. San Francisco, CA (2017). https://api.planet.com

Qi, J., Kerr, Y., Chehbouni, A. (1994). External factor consideration in vegetation index development. In: Proceedings of Physical Measurements and Signatures in Remote Sensing, ISPRS, pp. 723–730

Reichardt, M., & Juergens, C. (2006). The farmers view on the usability of precision farming in Germany—results of a multi-temporal survey. In Agricultural engineering for a better world: Proceedings of XVI CIGR world congress (VDI Verlag GmbH Düsseldorf).

Rouse, J.W., Haas, R.H., Schell, J.A., Deering, D., Deering, W (1973). Monitoring vegetation systems in the Great Plains with ERTS. In: ERTS Third Symposium, NASA SP-351 I, pp. 309–317

Robert, P.C. (1999) Precision agriculture: research needs and status in the USA. In: Proceedings of the 2nd European Conference on Precision Agriculture, pp. 19-33.

Santi, A. L., Damian, J. U. M., Cherubin, M. I. R., Amado, T. J. C., Eitelwein, M. T., Vian, A. E. L., & Herrera, W. F. B. (2016). Soil physical and hydraulic changes in different yielding zones under no-tillage in Brazil. African Journal of Agricultural Research, 11(15), 1326-1335.

Saraiva, A.M.; Massola, A.M.A.; Cugnasca, C.E. (1998). An object model for field information systems. In: Proceedings of the 4th European Conference on Precision Agriculture, pp.1355-1366.

Saraiva, A.M.; Massola, A.M.A.; Paz, S.M. (1997). Object oriented approach to the development of a field information system. In: Annual International Meeting of the American Society of Agricultural Engineers, ASAE Paper 97-3015.

Schut, P. (Ed.). (2007). OpenGIS ® Web Processing Service. Wayland, MA, USA: Open Geospatial Consortium, Inc.

Schnug, E., Panten, K. and Haneklaus, S. 1998. Soil sampling and nutrient recommendations – the future. Communications in Soil Science and Plant Analysis 29(11–14): 1455–1462.

Sørensen, C.G.; Fountas, S.; Blackmore, S.; Pedersen, H.H.; (2002). Information sources and decision making on precision farming. In: Proceedings of the 6th International Conference of Precision Agriculture. pp.1683-1695.

Spilke, J., & Zürnstein, K. (2005). Webservices—Beschreibung eines Ansatzes zur Anwendungskopplung und von Nutzensmöglichkeiten im Agrarbereich (Web services—description of an approach for application integration and of possibilities for usage in the agricultural sector). Zeitschrift für Agrarinformatik, 13(2), 33–40.

Stafford J.V., Lark R.M., Bolam H.C. (1998). Using yield maps to regionalize fields into potential management units. In P.C. Robert et al. (Eds.), Proceedings of the 4th international conference on precision agriculture, Madison, WI: ASA, CSSA, and SSSA, pp. 225-237.

Stafford, J.V., 2000. Implementing precision agriculture in the 21st century. Journal of Agricultural Engineering Research 76, 267/275.

Steinberger, G., Rothmund, M., & Auernhammer H. (2006). Agricultural Process Data Service (APDS). In Agricultural engineering for a better world: Proceedings of XVI CIGR world congress (VDI Verlag GmbH Düsseldorf).

Steinberger, G., Rothmund, M., Martini, D., Spietz, C., Mallon, D., & Nash, E. (2007). Integrating agroXML into an agricultural spatial data infrastructure. Landtechnik, 62(2), 114–115.

Stollberg, B., Lutz, M., Ostländer, N., & Bernard, L. (2007). Geoprozessierung in Geodateninfrastrukturen—Aufgaben für die nächste Generation.(Geoprocessing in spatial data infrastructures—challenges for the next generation). GIS Zeitschrift für Geoinformatik 4/2007, pp. 22–27.

Sultana, S.R.; Ali, A.; Ashfaq, A.; Mubeen, M.; Haq, M.; Ahmad, S.; Ercisli, S.; Jaafar, H.Z.R. 2014. Normalized difference vegetation index as a tool for wheat yield estimation: a case study from Faisalabad, Pakistan. The Scientific World Journal 14: 1-8

Tagarakis, A., Liakos, V., Fountas, S. et al. Management zones delineation using fuzzy clustering techniques in grapevines. Precision Agric 14, 18–39 (2013)

Tarnavsky, E.; Garrigues, S.; Brown, M.E. 2008. Multiscale geostatistical analysis of AVHRR, SPOT-VGT, and MODIS global NDVI products. Remote Sensing of Environment 112: 535-549.

Vieri M., Spezia, G., Pagni, Paolo, Frutti-viticoltura, I., Cattolica, U., & Parmense, E. (2010). Ingegneria delle produzioni viticole: stato dell'arte e future applicazioni. Review n.11 Italus Hortus 17(1), 33-57.

Vellidis, G., Garrick, V., Pocknee, S., Perry, C., Kvien, C., & Tucker, M. (2007). How wireless will change agriculture. In J. Stafford (Ed.), Precision agriculture'07 proceedings of the 6th European conference on precision agriculture (pp. 57–67).

Vretanos, P. (Ed.). (2005). Web feature service implementation specification. Wayland, MA, USA: Open Geospatial Consortium, Inc.

Walter, K., & Nash, E. (2009). Coupling wireless sensor networks and the Sensor Observation Service—bridging the interoperability gap. In J.-H. Haunert, B. Kieler, & J. Milde (Eds.), Proceedings of the 12th AGILE international conference on geographic information science, 2–5 June 2009, Hannover, Germany.

Wang, N., Zhang, N., & Wang, M. (2006). Wireless sensors in agriculture and food industry: Recent development and future perspective. Computers and Electronics in Agriculture, 50(1), 1–14.

Whelan, B.M., McBratney, A.B., Boydell, B.C., 1997. The Impact of PrecisionAgriculture. Proceedings of the ABARE Outlook Conference, 'The Future of Cropping in NW NSW', Moree, UK, July 1997, p. 5.

Yan, L., Zhou, S., Cifang, W., Hongyi, L., & Feng, L. (2007). Classification of management zones for precision farming in saline soil based on multi-data sources to characterize spatial variability of soil properties. Transactions of the Chinese Society of Agricultural Engineering, 23(8), 84–89.

Yoon, T., & Jeong, B. K. (2018). Service Oriented Architecture (SOA) Implementation: Success Factors and Realized Benefits. International Journal of Information Systems in the Service Sector (IJISSS), 10(2), 1-21

Zhang, N.; Wang, M.; Wang, N. (2002). Precision Agriculture - a worldwide overview. Computers and Electronics in Agriculture 36; 113-132.

Zhang, X., Shi, L., Jia, X., Seielstad, G., & Helgason, C. (2010). Zone mapping application for precision-farming: a decision support tool for variable rate application. Precision agriculture, 11(2), 103-114.

Zhou, M., Ma, X., Wang, K., Cheng, T., Tian, Y., Wang, J., ... & Yue, C. (2020). Detection of phenology using an improved shape model on time-series vegetation index in wheat. Computers and Electronics in Agriculture, 173, 105398.

Appendix

A SOIL DATA ACQUISITION AND PROCESSING

In [ ]:
```
import os
import gdal
import numpy as np
import rasterio
import glob
from owslib.wcs import WebCoverageService
```

In [ ]:
```
#lista layers di interesse
layers = ["all_layers"]
```

In [ ]:
```
print(layers)
```

In [ ]:
```
#chiamata OGC WCS al server per ottenere elenco layer del coverage
L = []
for i in layers:
    wcs = WebCoverageService('http://maps.isric.org/mapserv?map=/map/' + i + '.map', version='1.0.0')
    #print(list(wcs.contents))
    names = [k for k in wcs.contents.keys() if k.endswith("_mean")]
    L.append (names)
    print(names)
```

In [ ]:
```
#to test
test = wcs.contents['all_layers']
#ec_100-200cm_medio.supportedCRS
test.supportedCRS
```

In [ ]:
```
#to test
test.supportedFormats
```

In [ ]:
```
#to test
test.boundingboxes
```

In [ ]:
```
#bbox italia EPSG 152160 = min y, min x , max y ,  max x
bbox = (1396469, 4095551, 2526221, 5102736)
```

In [ ]:
```
for i in L:
    for jelementi in i:
        response = wcs.getCoverage(
            identifier= jelementi,
            crs='urn:ogc:def:crs:EPSG::152160',
            bbox=bbox,
            resx=250, resy=250,
            format='GEOTIFF_INT16')
        with open('/home/jovyan/work/soil/' + jelementi +'.tif', 'wb') as file:
            file.write(response.read())
```

In [ ]:
```
for soil in os.listdir('/home/jovyan/work/soil/'):
    if soil.endswith('.tif'):
        rastername = soil [:-4]
        print(rastername)
```

In [ ]:
```
for soil in os.listdir('/home/jovyan/work/soil/'):
    if soil.endswith('.tif'):
        rastername = soil [:-4]
        outraster = ("/home/jovyan/work/soil/outputfolder/" + rastername + "_wgs84.tif")
        os.system("gdalwarp -t_srs EPSG:4326 -dstnodata 0" + " " + soil + " " + outraster)
```

In [ ]:
```
directory_to_check = '/home/jovyan/work/soil/outputfolder/'
Lista = []
for soil in os.listdir(directory_to_check):
    if soil.endswith('.tif') and soil.startswith('ocd'):
        mergerastername = soil [:4]
        tifs = (os.path.join(directory_to_check, soil))
        outvrt = ("/home/jovyan/work/soil/mergedsoil/" + mergerastername + "_stack.vrt")
        outtif = ("/home/jovyan/work/soil/mergedsoil/" + mergerastername + "_stack.tif")
        Lista.append(tifs)
```

In [ ]:

```
#ordina bande
lista1 = [
        directory_to_check + mergerastername + '0-5cm_mean_wgs84.tif',
        directory_to_check + mergerastername + '5-15cm_mean_wgs84.tif',
        directory_to_check + mergerastername + '15-30cm_mean_wgs84.tif',
        directory_to_check + mergerastername + '30-60cm_mean_wgs84.tif',
        directory_to_check + mergerastername + '60-100cm_mean_wgs84.tif',
        directory_to_check + mergerastername + '100-200cm_mean_wgs84.tif'
]
```

In [ ]:

```
lista1
```

In [ ]:

```
file_list = lista1

# Read metadata of first file
with rasterio.open(file_list[0]) as src0:
    meta = src0.meta

# Update meta to reflect the number of layers
meta.update(count = len(file_list))

# Read each layer and write it to stack
with rasterio.open('/home/jovyan/work/soil/mergedsoil/' + mergerastername + 'stack.tif', 'w', **meta) as dst:
    for id, layer in enumerate(file_list, start=1):
        with rasterio.open(layer) as src1:
            dst.write_band(id, src1.read(1))
```

In [ ]:

```
with rasterio.open('/home/jovyan/work/soil/mergedsoil/' + mergerastername + 'stack.tif', 'r') as ds:
    arr = ds.read()/100
    arr_meta = ds.profile

arr_meta.update(dtype=rasterio.float32)

with rasterio.open('/home/jovyan/work/soil/mergedsoil/' + mergerastername + 'stack_float32.tif', 'w', **arr_meta) as src1:
    src1.write(arr.astype(rasterio.float32))
```

B SENTINEL-2 DATA ACQUISITION

In [ ]:

```
import geopandas as gpd
from shapely.geometry import MultiPolygon, Polygon
import sentinelsat
from sentinelsat import SentinelAPI
import folium
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
#Bisogna essere registrati sul sito di copernicus per poter utilizzare questa API
user = '###########' #nome utente
password = '############' #password

api = SentinelAPI(user, password, 'https://scihub.copernicus.eu/dhus')
```

In [ ]:

```
#SHP da utilizzare come bbox
bbox=gpd.read_file('campi_bbox.shp')
```

In [ ]:

```
#mostra il bbox sulla mappa
map = folium.Map([44.53,11.9], zoom_start=8)
folium.GeoJson(bbox).add_to(map)
map
```

In [ ]:

```
#prendi la geometria del bbox
footprint = None
for i in bbox['geometry']:
    footprint = i
```

In [ ]:

```
#trova le immigini passando i parametri desiderati
products = api.query(footprint,
                     date = ('20160101', '20201010'), #range di date da cercare
                     platformname = 'Sentinel-2',
                     processinglevel = 'Level-2A',
                     cloudcoverpercentage = (0, 5)) #percentuale di nuvolosità desidera
ta, 5% in qusto caso
```

In [ ]:

```
#stampa i tile trovati
tile = api.to_geodataframe(products)
ax = tile.plot(column='uuid',figsize=(17, 17))
tile.apply(lambda x: ax.annotate(text=x.uuid, xy=x.geometry.centroid.coords[0],size=20,
ha='center', fontsize='14'),axis=1)
```

In [ ]:

```
#tile + bbox
f, ax = plt.subplots(1,figsize=(15,15))
tile.plot(ax=ax,column='uuid',cmap=None)
bbox.plot(ax=ax,color='red')
```

In [ ]:

```
#crea un GDF e lo ordina in base alla % di nuvolosità
tile_gdf = api.to_geodataframe(products)
tile_gdf_sorted = granule_gdf.sort_values(['cloudcoverpercentage'], ascending=[True])
tile_gdf_sorted
```

In [ ]:

```
len(granule_gdf_sorted)
```

In [ ]:

```
#scarica i  tile
api.download(['tile_gdf_sorted'])
```

In [ ]:

```
import zipfile

#estrae la cartella della zip e la espota nel path desiderato
with zipfile.ZipFile('C:/Users/Utente/tile_gdf_sorted.zip', 'r') as zip_ref:
    zip_ref.extractall('C:/Users/Utente/Desktop/tile_gdf_sorted/')
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

C PLANETSCOPE DATA ACQUISITION

In [1]:
```
## add JOLANDA DI SAVOIA bounding box, AREA OF INTEREST (AOI)
geojson_geometry = {
  "type": "Polygon",
  "coordinates": [[[8.67898,44.656839],[8.67898,44.706894],[8.739678,44.706894],[8.739678,44.656839],[8.67898,44.656839]]]
}
```

In [2]:
```
# get images that overlap with our AOI
geometry_filter = {
  "type": "GeometryFilter",
  "field_name": "geometry",
  "config": geojson_geometry
}
# get images acquired within a date range
date_range_filter = {
  "type": "DateRangeFilter",
  "field_name": "acquired",
  "config": {
    "gte": "2018-10-201T00:00:00.000Z",
    "lte": "2019-11-27T00:00:00.000Z"
  }
}
# only get images which have <5% cloud coverage
cloud_cover_filter = {
  "type": "RangeFilter",
  "field_name": "cloud_cover",
  "config": {
    "lte": 0.10
  }
# combine our geo, date, cloud filters
combined_filter = {
  "type": "AndFilter",
  "config": [geometry_filter, date_range_filter, cloud_cover_filter, grid_cell_filter, usable_data_filter]
}
```

In [ ]:
```
#ENABLE ENVIRONMENT AND AUTHORIZATION
import os
import json
import requests
from requests.auth import HTTPBasicAuth

# API Key stored as variable
PLANET_API_KEY = '################'

#select item type
item_type = "PSOrthoTile"
#select asset type
#asset_type ="analytic"

# API request object
search_request = {
  "interval": "day",
  "item_types": [item_type],
#  "asset_types": [asset_type],
  "filter": combined_filter
}

# fire off the POST request
search_result = \
  requests.post(
    'https://api.planet.com/data/v1/quick-search',
    auth=HTTPBasicAuth(PLANET_API_KEY,''),
    json=search_request)

print(json.dumps(search_result.json(), indent=1))
```

In [ ]:
```
# extract image IDs only
image_ids = [feature['id'] for feature in search_result.json()['features']]
print(image_ids)
```

In [ ]:
```
# grab the image ID
id0 = 'Image'
id0_url = 'https://api.planet.com/data/v1/item-types/{}/items/{}/assets'.format(item_type, id0)

# Returns JSON metadata for assets in this ID. Learn more: planet.com/docs/reference/data-api/items-assets/#asset
result = \
  requests.get(
    id0_url,
    auth=HTTPBasicAuth(PLANET_API_KEY, '')
  )

# List of asset types available for this particular satellite image
print(result.json().keys())
```

In [ ]:
```
# This is "inactive" if the "analytic" asset has not yet been activated; otherwise 'active'
print(result.json()['analytic']['status'])
```

In [ ]:
```
# Parse out useful links
links = result.json()[u"analytic"]["_links"]
self_link = links["_self"]
activation_link = links["activate"]

# Request activation of the 'analytic' asset:
activate_result = \
  requests.get(
    activation_link,
    auth=HTTPBasicAuth(PLANET_API_KEY, '')
  )
```

In [ ]:
```
activation_status_result = \
  requests.get(
    self_link,
    auth=HTTPBasicAuth(PLANET_API_KEY, '')
  )

print(activation_status_result.json()["status"])
```

In [ ]:
```
# Image download:
download_link = activation_status_result.json()["location"]
print(download_link)
```

In [ ]:
```
# This is "inactive" if the "analytic" asset has not yet been activated; otherwise 'active'
print(result.json()['analytic_xml']['status'])
```

In [ ]:
```
# Parse out useful links
links = result.json()[u"analytic_xml"]["_links"]
self_link = links["_self"]
activation_link = links["activate"]

# Request activation of the 'analytic' asset:
activate_result = \
  requests.get(
    activation_link,
    auth=HTTPBasicAuth(PLANET_API_KEY, '')
  )
```

In [ ]:
```
activation_status_result = \
  requests.get(
    self_link,
    auth=HTTPBasicAuth(PLANET_API_KEY, '')
  )

print(activation_status_result.json()["status"])
```

In [ ]:
```
# XML download:
download_link = activation_status_result.json()["location"]
print(download_link)
```

In [ ]:
```

```

D METEOROLOGICAL DATA ACQUISITION

In [ ]:

```
# Import helper modules
import os
import json
import requests
import fiona
from fiona import collection
import glob
import numpy as np
import pandas as pd
import geopandas as gpd
from osgeo import gdal, ogr
import rasterio
import regex as re
from rasterio.shutil import delete
from rasterio import mask
from rasterio.merge import merge
import datetime
import matplotlib.pyplot as plt
import shapely
from datetime import date, timedelta
```

In [ ]:

```
# enter your API Key
os.environ['apikey']='##################'
```

In [ ]:

```
# Setup the API Key from the environment variable
API_KEY = os.getenv('apikey')
```

In [ ]:

```
#test the aAPI Key
API_KEY
```

In [ ]:

```
#costruisci i parametri di richiesta
url='http://my.meteoblue.com/packages/historybasic-day'
lat='$var'
lon='$var'
startdate='$var'
enddate='$var'
apikey=API_KEY
timeformat='iso8601'
format='json'
```

In [ ]:

```
#costruisci la richiesta
htmlrequest = url+'?'+'lat='+lat+'&'+'lon='+lon+'&'+'startdate='+startdate+'&'+'enddate
='+enddate+'&'+'apikey='+apikey+'&'+'timeformat='+timeformat+'&'+'format='+format
```

In [ ]:

```
#stampa la richiesta
print(htmlrequest)
```

In [ ]:

```
# Setup the session
session = requests.Session()
# Authenticate
session.auth = (API_KEY, "")
```

In [ ]:

```
# Make a GET request to the Planet Data API
res = session.get(htmlrequest)
```

In [ ]:

```
# Response status code
res.status_code
```

In [ ]:

```
polygonPath='/home/jovyan/work/jolanda_grano_duro/output/NDVI/'
polygons = glob.glob(polygonPath +'/FID/*.shp')
prova =p.split('/')[7]
campo = prova.replace('.shp' , ' campo')
print(campo)
```

In [ ]:

```
for p in polygons:
    centroid = gpd.read_file(p).centroid
    coordinate = centroid.crs
    centroid2 = centroid.to_crs(epsg=4326)
    #lista_coordinate =((centroid2.geometry.map(lambda p: p.x)[0],centroid2.geometry.ma
p(lambda p: p.y)[0]))
    latitudine=(centroid2.geometry.map(lambda p: p.x)[0])
    longitudine=(centroid2.geometry.map(lambda p: p.y)[0])
    nomeshape=p.split('/')[7]
    campo=nomeshape.replace('.shp' , ' FID')
    #costruisci la richiesta
    htmlrequest = url+'?'+'lat='+str(latitudine)+'&'+'lon='+str(longitudine)+'&'+'start
date='+startdate+'&'+'enddate='+enddate+'&'+'apikey='+apikey+'&'+'timeformat='+timeform
at+'&'+'format='+format
    DF=pd.read_csv(htmlrequest)
    df=DF[['time', 'temperature_mean', 'precipitation', 'windspeed_mean']]
    df.set_index('time',inplace=True)
    df.plot(figsize=(11,9),title='Meteo Data '+ campo)
```

In [ ]:

```
pd.read_json
```

E DOCKER-COMPOSE YAML FILE FOR SERVICE ORIENTED ARCHITECTURE IMPLEMENTATION

In [ ]:

```
version: '2'
services:
  nginx:
    container_name: nginx_loadbalance
    restart: always
    image: nginx
    logging:
        driver: "json-file"
        options:
            max-size: "10m"
            max-file: "10"
    ports:
     - "81:8080"
    volumes:
     - /opt/sdi/nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    links:
     - geoserver_8085
     - geoserver_8086
  db:
    image: simonelanucara/postgis:10.0-2.4
    volumes:
     - /opt/sdi/pg/postgres_data:/var/lib/postgresql
    ports:
     - "25433:5432"
    environment:
      - USERNAME=#######
      - PASS=########
      - ALLOW_IP_RANGE=0.0.0.0/0
    restart: always
  geoserver_8085:
    container_name: geoserver_8085
    restart: always
    image: simonelanucara/geoserver-docker:2.15
    logging:
        driver: "json-file"
        options:
            max-size: "10m"
            max-file: "10"
    hostname: geoserver_8085
    ports:
     - "8085:8080"
    environment:
     - GEOSERVER_LOG_LOCATION=/opt/sdi/geoserver_data/logs/geoserver_8085.log
    volumes:
     - /opt/sdi/geoserver_data:/opt/geoserver/data_dir
     - /opt/sdi/tomcat_settings/setenv.sh:/usr/local/tomcat/bin/setenv.sh
     - /opt/sdi/tomcat_settings/logs_8085:/usr/local/tomcat/logs
     - /opt/sdi/Rasters:/root/Raster
    links:
     - db
  geoserver_8086:
    container_name: geoserver_8086
    restart: always
    image: simonelanucara/geoserver-docker:2.15
    logging:
        driver: "json-file"
        options:
            max-size: "10m"
            max-file: "10"
```

```
    hostname: geoserver_8086
    ports:
     - "8086:8080"
    environment:
     - GEOSERVER_LOG_LOCATION=/opt/sdi/geoserver_data/logs/geoserver_8086.log
    volumes:
     - /opt/sdi/geoserver_data:/opt/geoserver/data_dir
     - /opt/sdi/tomcat_settings/setenv.sh:/usr/local/tomcat/bin/setenv.sh
     - /opt/sdi/tomcat_settings/logs_8086:/usr/local/tomcat/logs
     - /opt/sdi/Rasters:/root/Raster
    links:
     - db
  geonetwork:
    container_name: geonetwork
    image: simonelanucara/geonetwork
    restart: always
    hostname: geonetwork
    ports:
      - 8087:8080
    environment:
     DATA_DIR: /opt/sdi/geonetwork_data
     POSTGRES_DB_USERNAME: ##########
     POSTGRES_DB_PASSWORD: ########
     POSTGRES_DB_HOST: 172.17.0.1
     POSTGRES_DB_PORT: 25433
    volumes:
     - /opt/sdi/geonetwork_data:/opt/sdi/geonetwork_data
    links:
     - db
  keycloak:
      image: simonelanucara/keycloak
      environment:
        #DB_VENDOR: POSTGRES
        #DB_ADDR: db
        DB_DATABASE: keycloak
        DB_USERNAME: #######
        DB_PASSWORD: ######
        DB_HOST: 172.17.0.1
        DB_PORT: 25433
        KEYCLOAK_USER: ######
        KEYCLOAK_PASSWORD: #####
        # Uncomment the line below if you want to specify JDBC parameters. The paramete
r below is just an example, and it shouldn't be used in production without knowledge. I
t is highly recommended that you read the PostgreSQL JDBC driver documentation in order
to use it.
        #JDBC_PARAMS: "ssl=false"
      ports:
        - 8443:8080
      links:
        - db
  wps:
      image: lanucara/northwps
      environment:
        DB_DATABASE: wps
        DB_USERNAME: #####
        DB_PASSWORD: #####
        DB_HOST: 172.17.0.1
        DB_PORT: 25433
       WPS_USER: #######
      ports:
        - 8493:8080
```

```
    links:
      - db
edi:
    image: lanucara/EDI
    environment:
      DB_DATABASE: edi
      DB_USERNAME: #####
      DB_PASSWORD: #####
      DB_HOST: 172.17.0.1
      DB_PORT: 25433
     WPS_USER: #######
    ports:
      - 8494:8080
    links:
      - db
sos:
    image: lanucara/northsos
    environment:
      DB_DATABASE: sos
      DB_USERNAME: #####
      DB_PASSWORD: #####
      DB_HOST: 172.17.0.1
      DB_PORT: 25433
     WPS_USER: #######
    ports:
      - 8494:8080
    links:
      - db
jena:
    image: lanucara/ajena
    environment:
      DB_DATABASE: jena
      DB_USERNAME: #####
      DB_PASSWORD: #####
      DB_HOST: 172.17.0.1
      DB_PORT: 25433
     WPS_USER: #######
    ports:
      - 8495:8080
    links:
      - db
```

F SATELLITE DATA CORRECTION AND ELABORATION OF MODIFIED SOIL-ADJUSTED VEGETATION INDEX

In [ ]:

Edit Attachments

```
import rasterio
import numpy as np

filename = "/home/jovyan/work/planet_data/psorthotile/"

# Load red and NIR bands - note all PlanetScope 4-band images have band order BGRN
with rasterio.open(filename) as src:
    band_red = src.read(3)

with rasterio.open(filename) as src:
    band_nir = src.read(4)
```

In [ ]:

Edit Attachments

```
from xml.dom import minidom

xmldoc = minidom.parse("/home/jovyan/work/planet_data/psorthotile/Analytic_metadata.xml")
nodes = xmldoc.getElementsByTagName("ps:bandSpecificMetadata")

# XML parser refers to bands by numbers 1-4
coeffs = {}
for node in nodes:
    bn = node.getElementsByTagName("ps:bandNumber")[0].firstChild.data
    if bn in ['1', '2', '3', '4']:
        i = int(bn)
        value = node.getElementsByTagName("ps:reflectanceCoefficient")[0].firstChild.data
        coeffs[i] = float(value)

# Multiply the Digital Number (DN) values in each band by the TOA reflectance coefficients
band_red = band_red * coeffs[3]
band_nir = band_nir * coeffs[4]
```

In [ ]:

Edit Attachments

```
# Allow division by zero
#np.seterr(divide='ignore', invalid='ignore')

# Calculate MSAVI. This is the equation at the top of this guide expressed in code
msavi2 = (2 * (band_nir.astype(float) + 1) - np.sqrt((2 * band_nir.astype(float) + 1) * (2 * band_nir.astype(float) + 1) - 8
#ndvi = (band_nir.astype(float) - band_red.astype(float)) / (band_nir + band_red)
```

In [ ]:

Edit Attachments

```
# check range MSAVI values, excluding NaN
np.nanmin(msavi2), np.nanmax(msavi2)
```

In [ ]:

Edit Attachments

```
# Set spatial characteristics of the output object to mirror the input
kwargs = src.meta
kwargs.update(
    dtype=rasterio.float32,
    count = 1)

# Write band calculations to a new raster file
with rasterio.open('/home/jovyan/work/planet_data/psorthotileMSAVI/.tif', 'w', **kwargs) as dst:
        dst.write_band(1, msavi2.astype(rasterio.float32))
```

G SATELLITE DATA CORRECTION AND ELABORATION OF NORMALIZED DIFFERENCE VEGETATION INDEX

In [ ]:

Edit Attachments

```
import rasterio
import numpy as np
import gdal

filename = "/home/jovyan/work/planet_data/psorthotile/"

# Load red and NIR bands - note all PlanetScope 4-band images have band order BGRN
with rasterio.open(filename) as src:
    band_red = src.read(3)

with rasterio.open(filename) as src:
    band_nir = src.read(4)
```

In [ ]:

Edit Attachments

```
from xml.dom import minidom
# image correction by xml metadata
xmldoc = minidom.parse("/home/jovyan/work/planet_data/psorthotile/Analytic_metadata.xml")
nodes = xmldoc.getElementsByTagName("ps:bandSpecificMetadata")

# XML parser refers to bands by numbers 1-4
coeffs = {}
for node in nodes:
    bn = node.getElementsByTagName("ps:bandNumber")[0].firstChild.data
    if bn in ['1', '2', '3', '4']:
        i = int(bn)
        value = node.getElementsByTagName("ps:reflectanceCoefficient")[0].firstChild.data
        coeffs[i] = float(value)

# Multiply the Digital Number (DN) values in each band by the TOA reflectance coefficients
band_red = band_red * coeffs[3]
band_nir = band_nir * coeffs[4]
```

H MASKING PROCEDURE

In [ ]:

```
#importa le librerie
import os
import fiona
import glob
import numpy as np
import pandas as pd
from osgeo import gdal, ogr
import rasterio
import regex as re
from rasterio.shutil import delete
from rasterio import mask
from rasterio.merge import merge
import datetime
import matplotlib.pyplot as plt
from datetime import date, timedelta
```

In [ ]:

```
#setta i percorsi di input ed output
root = '/home/jovyan/satellitedata/it/master/NDVI/'
shpfiles='/home/jovyan/work/jolanda_grano_duro/input/shp_jolanda_grano_duro'
outputPath='/home/jovyan/work/jolanda_grano_duro/output/NDVI/'
```

In [ ]:

```
#prendi tutti i .tif
rasterfiles = [os.path.join(root, file) for root, dirs, files in os.walk(root)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i .tif totali
len(rasterfiles)
```

In [ ]:

```
#imposta le date di inizio e fine analisi
sdate = date(2018, 11, 1)   # start date
edate = date(2019, 6, 30)   # end date
delta = edate - sdate       # as timedelta

DATE = []

for i in range(delta.days + 1):
    day = sdate + timedelta(days=i)
    DATE.append(day.strftime('%Y%m%d'))
```

In [ ]:

```
#crea la lista dei tif ricadenti in quelle date
ndvi2mesi=[]
for s in rasterfiles:
    for l in DATE:
        if l in s:
            ndvi2mesi.append(s)
```

In [ ]:

```
#crea le cartelle dal fid dello shape e suddividi gli shape
os.makedirs(outputPath + "/FID/", exist_ok=True)
with fiona.open(shpfiles) as source:
    meta = source.meta
    for f in source:
        ID=f['properties']['fid']
        #ID=f['id']
        outfile = os.path.join(outputPath + "/FID/{}.shp".format(int(ID)))
        with fiona.open(outfile, 'w', **meta) as sink:
            sink.write(f)
```

In [ ]:

```
#leggi ogni singolo shape
polygons = glob.glob(outputPath +'/FID/*.shp')
```

In [ ]:

```
len(polygons)
```

In [ ]:

```
# crea cartella dove mandare raster non validi
os.makedirs(outputPath + "/FID/empty_raster", exist_ok=True)
```

In [ ]:

```
#crea i singoli tif per campo e per data con filtraggio in base a percentuale no data
rasterintersection = []
for polygon in polygons:
    # Estrai il nome dell'ID per aggiungerlo ai path
    subdirshp,shp=os.path.split(str(polygon))
    shp= os.path.splitext(shp)

    # legge il .shp di input e la sua geometria
    with fiona.open(polygon,'r')as SHP:
        fid_geometry=[feature["geometry"] for feature in SHP]

        for raster_path in ndvi2mesi:

            # Trova il FID che si sovrappone al tile
            raster = gdal.Open(raster_path)
            vector = ogr.Open(polygon)
            transform = raster.GetGeoTransform()
            pixelWidth = transform[1]
            pixelHeight = transform[5]
            cols = raster.RasterXSize
            rows = raster.RasterYSize
            xLeft = transform[0]
            yTop = transform[3]
            xRight = xLeft+cols*pixelWidth
            yBottom = yTop+rows*pixelHeight
            ring = ogr.Geometry(ogr.wkbLinearRing)
            ring.AddPoint(xLeft, yTop)
            ring.AddPoint(xLeft, yBottom)
            ring.AddPoint(xRight, yBottom)
            ring.AddPoint(xRight, yTop)
            ring.AddPoint(xLeft, yTop)
            rasterGeometry = ogr.Geometry(ogr.wkbPolygon)
            rasterGeometry.AddGeometry(ring)
            layer = vector.GetLayer()
            feature = layer.GetFeature(0)
            vectorGeometry = feature.GetGeometryRef()
            overlap=rasterGeometry.Intersect(vectorGeometry)

            if overlap == True:

                # Estrai il nome del tile per aggiungerlo ai path
                subdir,file=os.path.split(str(raster_path))
                file= os.path.splitext(file)
                file=file[0].split('_')
                rasterintersection.append(raster_path)
                #print(shp[0],'  ',raster_path)
                #print(raster_path)

                  # Crea una cartella con il nome dell'ID e del tile
                os.makedirs(outputPath + "/FID_{}".format(shp[0]),exist_ok=True)

                 # Crea il path di uscita del tile ritagliato
                output_path =outputPath + "/FID_{}".format(shp[0]) + "/{}_{}".format(fi
le[0],file[1]) + ".tif"

                with rasterio.open(raster_path) as tile:
                    out_image, out_transform = rasterio.mask.mask(tile, fid_geometry, c
rop=True)
                    out_meta = tile.meta
```

```
                    #controlla la % di nodata = '-32768'
                    nodata_value = -32768
                    nodata_count = np.count_nonzero(out_image == nodata_value)
                    total_count = out_image.shape[1] * out_image.shape[2]
                    pct_valid = nodata_count/total_count * 100

                    count_nan = np.count_nonzero(np.isnan(out_image))
                    val = count_nan/total_count * 100

                    #se non ci sono nodata a '-32768' pct_valid = 0 e val indica la % d
i 'nan' nella clip
                    if pct_valid <= 95 and val < (10 /total_count *100):
                        #i = i + 1
#                             Aggiorna i metadati nel sul .tif ritagliato
                        out_meta.update({"driver": "GTiff","height": out_image.shape[1
],"width": out_image.shape[2],"transform": out_transform,"nodata":-32768})

#                            Salva il nuovo .tif
                        with rasterio.open(output_path, "w", **out_meta) as dest:
                            dest.write(out_image)
```

In [ ]:

```
rasterfiles = [os.path.join(outputPath, file) for outputPath, dirs, files in os.walk(ou
tputPath)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
len(rasterfiles)
```

In [ ]:

```
valide = []
nonvalide = []
for i in rasterfiles:
    with rasterio.open(i) as immagine:
        arr = immagine.read()
        nodata_value = -32768
        pixeltutti = arr.shape[1] * arr.shape[2]
        nodata_count = np.count_nonzero(arr == nodata_value)
        pct_invalid = nodata_count/pixeltutti * 100

        if pct_invalid <= 50.0:
            print(i, 'immagine valida ' + str(pct_valid))
            valide.append((i, pct_invalid))
        else:
            print(i, 'immagine non valida ' + str(pct_valid))
            nonvalide.append((i, pct_invalid))
```

In [ ]:

```
nonvalide
```

In [ ]:

```
#crea i singoli tif per campo e per data con filtraggio in base a percentuale no data
rasterintersection = []
for polygon in polygons:
    # Estrai il nome dell'ID per aggiungerlo ai path
    subdirshp,shp=os.path.split(str(polygon))
    shp= os.path.splitext(shp)

    # legge il .shp di input e la sua geometria
    with fiona.open(polygon,'r')as SHP:
        fid_geometry=[feature["geometry"] for feature in SHP]

        for raster_path in ndvi2mesi:

            # Trova il FID che si sovrappone al tile
            raster = gdal.Open(raster_path)
            vector = ogr.Open(polygon)
            transform = raster.GetGeoTransform()
            pixelWidth = transform[1]
            pixelHeight = transform[5]
            cols = raster.RasterXSize
            rows = raster.RasterYSize
            xLeft = transform[0]
            yTop = transform[3]
            xRight = xLeft+cols*pixelWidth
            yBottom = yTop+rows*pixelHeight
            ring = ogr.Geometry(ogr.wkbLinearRing)
            ring.AddPoint(xLeft, yTop)
            ring.AddPoint(xLeft, yBottom)
            ring.AddPoint(xRight, yBottom)
            ring.AddPoint(xRight, yTop)
            ring.AddPoint(xLeft, yTop)
            rasterGeometry = ogr.Geometry(ogr.wkbPolygon)
            rasterGeometry.AddGeometry(ring)
            layer = vector.GetLayer()
            feature = layer.GetFeature(0)
            vectorGeometry = feature.GetGeometryRef()
            overlap=rasterGeometry.Intersect(vectorGeometry)

            if overlap == True:

                # Estrai il nome del tile per aggiungerlo ai path
                subdir,file=os.path.split(str(raster_path))
                file= os.path.splitext(file)
                file=file[0].split('_')
                rasterintersection.append(raster_path)
                #print(shp[0],'  ',raster_path)
                #print(raster_path)

                  # Crea una cartella con il nome dell'ID e del tile
                os.makedirs(outputPath + "/FID_{}".format(shp[0]),exist_ok=True)

                 # Crea il path di uscita del tile ritagliato
                output_path =outputPath + "/FID_{}".format(shp[0]) + "/{}_{}".format(fi
le[0],file[1]) + ".tif"

                #os.system("gdalwarp -dstnodata -32768 -crop_to_cutline -cutline  " + p
olygon + " " + raster_path + " " + output_path)
                 # Legge il .tif e ritaglia in base al .shp
                with rasterio.open(raster_path) as tile:
```

```
                    out_image, out_transform = rasterio.mask.mask(tile, fid_geometry, c
rop=True)
                    out_meta = tile.meta
                    #controlla la % di nodata = '-32768'
                    nodata_value = -32768
                    nodata_count = np.count_nonzero(out_image == nodata_value)
                    total_count = out_image.shape[1] * out_image.shape[2]
                    pct_invalid = nodata_count/total_count * 100

                    if pct_invalid <= 95.0:
                        count_nan = np.count_nonzero(np.isnan(out_image))
                        total = out_image.shape[1] * out_image.shape[2]
                        invalid = count_nan/total * 100
                         #Se la % di 'nan' sia < 10 % dei pixel della clip salvala
                        if (invalid < (10 /total_count *100)):
                             #i = i + 1
#                             # Aggiorna i metadati nel sul .tif ritagliato
                            out_meta.update({"driver": "GTiff","height": out_image.shap
e[1],"width": out_image.shape[2],"transform": out_transform,"nodata": -32768})
#                             # Salva il nuovo .tif
                            with rasterio.open(output_path, "w", **out_meta) as dest:
                                dest.write(out_image)

                             #print (i,' Nuova clip: %s'%output_path,'\nFID: %s'% polyg
on,' ','Tile: %s'% raster_path,'\n',pct_invalid,' ',invalid, out_image.min(),out_image.
max(),'\n')

            else:
                pass
```

In [ ]:

```
invalid < (5 /total_count *100)
```

In [ ]:

```
#leggi i tif precedentemente creati
rasterfiles = [os.path.join(outputPath, file) for outputPath, dirs, files in os.walk(ou
tputPath)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i tif precedentemente creati
len(rasterfiles)
```

I MAXIMUM VALUE COMPOSITE (MVC) ELABORATION

In [ ]:

```
import glob
import os

import gdal
import numpy as np
from pathlib import Path

product='MSAVI'#'NDVI'

directory_to_check = ("/home/jovyan/work/jolanda_grano_duro/output/"+product) # Which d
irectory do you want to start with?

def my_function(directory):

# get rasters' file names
    fnames = glob.glob('*.tif')
#controlla che ci siano file .tif
    if len(fnames) == 0:
        pass
    else:
# read general properties of the first raster (assuming all the rasters share these pro
perties)
        ds = gdal.Open(fnames[0], 0)
        gt = ds.GetGeoTransform()
        sr = ds.GetProjection()
        xsize = ds.RasterXSize
        ysize = ds.RasterYSize
        nd = ds.GetRasterBand(1).GetNoDataValue()
        del ds

# read each raster and create a 3D array
        arrays = []
        for fn in fnames:
            ds = gdal.Open(fn, 0)
            arr = ds.ReadAsArray()  # 2D array (rows by columns)
            arrays.append(arr)
            del ds
        arr = np.stack(arrays)  # 3D array (date by rows by columns)

# mask the array to exclude values outside a defined range - IF SENTINEL 2 CNR DATA - m
ask = (arr < 10000) & (arr > 10000)
        mask = (arr < 10000) & (arr > 10000)
        arr = np.ma.array(arr, mask=mask)

# get maximum value for each pixel
        arr = np.max(arr, axis=0)

# fill the array with the NoData value
        arr = arr.filled(-32768)

# create the output raster
        cwd = os.getcwd()
        numerocampo = os.path.basename(cwd)
        out_fn = (str(numerocampo) + '_MVC_'+product+'.tif')
        driver = gdal.GetDriverByName('GTiff')
        out_ds = driver.Create(out_fn, xsize, ysize, 1, gdal.GDT_Float32)  # you might
 want to change the pixel type
        out_ds.SetGeoTransform(gt)
        out_ds.SetProjection(sr)
```

```
        out_band = out_ds.GetRasterBand(1)
        out_band.SetNoDataValue(-32768)
        out_band.WriteArray(arr)

# # save and close output file
        out_band.FlushCache()
        del out_ds, out_band


# Get all the subdirectories of directory_to_check recursively and store them in a lis
t:
directories = [os.path.abspath(x[0]) for x in os.walk(directory_to_check)]
directories.remove(os.path.abspath(directory_to_check)) # If you don't want your main d
irectory included

for i in directories:
        os.chdir(i)         # Change working Directory
        my_function(i)      # Run your function
```

In [ ]:

```
import glob
import os
import gdal
import numpy as np
from pathlib import Path
import rasterio
import regex as re
import datetime
import pandas as pd
```

In [ ]:

```
product='NDVI'#'NDVI'

directory_to_check = ("/home/jovyan/work/jolanda_grano_duro/output/"+product) # Which d
irectory do you want to start with?

T=[]
def my_function(directory):
# get rasters' file names
    fnames = glob.glob('*_MVC_'+product+'.tif')
#controlla che ci siano file .tif
    if len(fnames) == 0:
        pass
    else:

        for i in fnames:
            with rasterio.open(i) as tile:
                arr=tile.read()
                arr[arr == tile.nodata] = -32768
                meta = tile.meta.copy()
                r=arr[arr>-10000]
                #datesearch = re.search("([0-9]{8})", i)
                #date = datetime.datetime.strptime(datesearch.group(), '%Y%m%d').date()
                #stringadate = date.strftime("%Y/%m/%d")
                campo = i.split('_')[1]
                #file = i.split('_')[2]
                T.append((campo,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min(),'%.2f'%r.
std()))
                dataframe = pd.DataFrame(T,columns=['Campo','msavi_max','msavi_mean','m
savi_min', 'msavi_std'])
            print(campo,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min(),'%.2f'%r.std())
# Get all the subdirectories of directory_to_check recursively and store them in a lis
t:
directories = [os.path.abspath(x[0]) for x in os.walk(directory_to_check)]
directories.remove(os.path.abspath(directory_to_check)) # If you don't want your main d
irectory included

for i in directories:
        os.chdir(i)         # Change working Directory
        my_function(i)      # Run your function
```

In [ ]:

```
dataframe = pd.DataFrame(T,columns=['Campo','msavi_max','msavi_mean','msavi_min', 'msav
i_std'])
```

In [ ]:

```
dataframe
```

In [ ]:

```
dataframe.to_csv ('/home/jovyan/work/jolanda_grano_duro/output/NDVI/NDVI_MVC_analysis.c
sv')
```

L FUZZY C-MEAN CLUSTERING ELABORATION

In [ ]:

```
import os
import glob
import numpy as np
from osgeo import gdal, ogr
import rasterio
from sklearn.preprocessing import scale
#from sklearn.cluster import KMeans
#from k_means_constrained import KMeansConstrained
from sklearn import cluster
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
# get rasters' file names
fnames = '/home/jovyan/work/jolanda_grano_duro/output/MSAVI/$.tif'
#controlla che ci siano file .tif
if len(fnames) == 0:
    pass
else:
    with rasterio.open(fnames) as tile:
        arr=tile.read()
        out_meta = tile.meta
        dst_meta = out_meta.copy()
        dst_meta['nodata'] = -32768.
arr
```

In [ ]:

```
# array, value 0 and nan, metadata
def read_img(raster):
    with rasterio.open(raster)as ds:
        profile = ds.profile
        n_bands = profile['count']
        arr=ds.read()
        arr = arr[0,:, :]
        arr[arr==-32768.] = np.nan
    return arr, profile, n_bands

# Preprocessing standardization
def pre_processing(arr):
    flattened_img = arr.reshape(n_bands, -1)
    flattened_img = flattened_img.T
    n_pixels = flattened_img.shape[0]
    clustered = np.ones((n_pixels, 1)) * -1
    null_mask = np.isnan(flattened_img).all(axis=1)
    notnull_array = flattened_img[~null_mask]
    std_array = scale(notnull_array, axis=0)
    return clustered, null_mask,std_array

# Array clustering
def clustering(std_array,n_clusters, max_iter):
    K = cluster.KMeans(n_clusters = n_clusters, max_iter = max_iter)
    K.fit(std_array)
    X_clustered = K.labels_
    point = X_clustered.shape[0]
    X_clustered = X_clustered.reshape(point,1)
    return X_clustered


# update nodata
def write_clustering(img,clustered,out_clusters_path=''):
    arr, profile, n_bands = read_img(img)
    dst_meta = profile.copy()
    dst_meta['nodata'] = -32768.
    with rasterio.open(out_clusters_path + os.path.basename(img),'w',**dst_meta)as dst:
        dst.write(clustered.astype(rasterio.float32))
```

In [ ]:

```
dst_meta = profile.copy()
dst_meta['nodata'] = -1
with rasterio.open('/home/jovyan/work/jolanda_grano_duro/output/MSAVI/$_clustered.tif',
'w',**dst_meta)as dst:
    dst.write(clustered.astype(rasterio.float32))
```

M NORMALIZED DIFFERENCE VEGETATION INDEX ANALYSIS

In [ ]:

```
#importa le librerie
import os
import fiona
import glob
import numpy as np
import pandas as pd
from osgeo import gdal, ogr
import rasterio
import regex as re
from rasterio.shutil import delete
from rasterio import mask
from rasterio.merge import merge
import datetime
import matplotlib.pyplot as plt
from datetime import date, timedelta
```

In [ ]:

```
#setta i percorsi di input ed output
root = '/home/jovyan/satellitedata/it/master/NDVI/'
shpfiles='/home/jovyan/work/jolanda_grano_duro/input/shp_jolanda_grano_duro'
outputPath='/home/jovyan/work/jolanda_grano_duro/output/NDVI/'
```

In [ ]:

```
#prendi tutti i .tif
rasterfiles = [os.path.join(root, file) for root, dirs, files in os.walk(root)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i .tif totali
len(rasterfiles)
```

In [ ]:

```
#imposta le date di inizio e fine analisi
sdate = date(2018, 11, 1)   # start date
edate = date(2019, 6, 30)   # end date
delta = edate - sdate       # as timedelta

DATE = []

for i in range(delta.days + 1):
    day = sdate + timedelta(days=i)
    DATE.append(day.strftime('%Y%m%d'))
```

In [ ]:

```
#crea la lista dei tif ricadenti in quelle date
ndvi2mesi=[]
for s in rasterfiles:
    for l in DATE:
        if l in s:
            ndvi2mesi.append(s)
```

In [ ]:

```
#crea le cartelle dal fid dello shape e suddividi gli shape
os.makedirs(outputPath + "/FID/", exist_ok=True)
with fiona.open(shpfiles) as source:
    meta = source.meta
    for f in source:
        ID=f['properties']['fid']
        #ID=f['id']
        outfile = os.path.join(outputPath + "/FID/{}.shp".format(int(ID)))
        with fiona.open(outfile, 'w', **meta) as sink:
            sink.write(f)
```

In [ ]:

```
#leggi ogni singolo shape
polygons = glob.glob(outputPath +'/FID/*.shp')
```

In [ ]:

```
len(polygons)
```

In [ ]:

```
# crea cartella dove mandare raster non validi
os.makedirs(outputPath + "/FID/empty_raster", exist_ok=True)
```

In [ ]:

```
#crea i singoli tif per campo e per data con filtraggio in base a percentuale no data
rasterintersection = []
for polygon in polygons:
    # Estrai il nome dell'ID per aggiungerlo ai path
    subdirshp,shp=os.path.split(str(polygon))
    shp= os.path.splitext(shp)

    # legge il .shp di input e la sua geometria
    with fiona.open(polygon,'r')as SHP:
        fid_geometry=[feature["geometry"] for feature in SHP]

        for raster_path in ndvi2mesi:

            # Trova il FID che si sovrappone al tile
            raster = gdal.Open(raster_path)
            vector = ogr.Open(polygon)
            transform = raster.GetGeoTransform()
            pixelWidth = transform[1]
            pixelHeight = transform[5]
            cols = raster.RasterXSize
            rows = raster.RasterYSize
            xLeft = transform[0]
            yTop = transform[3]
            xRight = xLeft+cols*pixelWidth
            yBottom = yTop+rows*pixelHeight
            ring = ogr.Geometry(ogr.wkbLinearRing)
            ring.AddPoint(xLeft, yTop)
            ring.AddPoint(xLeft, yBottom)
            ring.AddPoint(xRight, yBottom)
            ring.AddPoint(xRight, yTop)
            ring.AddPoint(xLeft, yTop)
            rasterGeometry = ogr.Geometry(ogr.wkbPolygon)
            rasterGeometry.AddGeometry(ring)
            layer = vector.GetLayer()
            feature = layer.GetFeature(0)
            vectorGeometry = feature.GetGeometryRef()
            overlap=rasterGeometry.Intersect(vectorGeometry)

            if overlap == True:

                # Estrai il nome del tile per aggiungerlo ai path
                subdir,file=os.path.split(str(raster_path))
                file= os.path.splitext(file)
                file=file[0].split('_')
                rasterintersection.append(raster_path)
                #print(shp[0],'  ',raster_path)
                #print(raster_path)

                  # Crea una cartella con il nome dell'ID e del tile
                os.makedirs(outputPath + "/FID_{}".format(shp[0]),exist_ok=True)

                 # Crea il path di uscita del tile ritagliato
                output_path =outputPath + "/FID_{}".format(shp[0]) + "/{}_{}".format(fi
le[0],file[1]) + ".tif"

                with rasterio.open(raster_path) as tile:
                    out_image, out_transform = rasterio.mask.mask(tile, fid_geometry, c
rop=True)
                    out_meta = tile.meta
```

```
                    #controlla la % di nodata = '-32768'
                    nodata_value = -32768
                    nodata_count = np.count_nonzero(out_image == nodata_value)
                    total_count = out_image.shape[1] * out_image.shape[2]
                    pct_valid = nodata_count/total_count * 100

                    count_nan = np.count_nonzero(np.isnan(out_image))
                    val = count_nan/total_count * 100

                    #se non ci sono nodata a '-32768' pct_valid = 0 e val indica la % d
i 'nan' nella clip
                    if pct_valid <= 95 and val < (10 /total_count *100):
                        #i = i + 1
#                           Aggiorna i metadati nel sul .tif ritagliato
                        out_meta.update({"driver": "GTiff","height": out_image.shape[1
],"width": out_image.shape[2],"transform": out_transform,"nodata":-32768})

#                           Salva il nuovo .tif
                        with rasterio.open(output_path, "w", **out_meta) as dest:
                            dest.write(out_image)
```

In [ ]:

```
rasterfiles = [os.path.join(outputPath, file) for outputPath, dirs, files in os.walk(ou
tputPath)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
len(rasterfiles)
```

In [ ]:

```
valide = []
nonvalide = []
for i in rasterfiles:
    with rasterio.open(i) as immagine:
        arr = immagine.read()
        nodata_value = -32768
        pixeltutti = arr.shape[1] * arr.shape[2]
        nodata_count = np.count_nonzero(arr == nodata_value)
        pct_invalid = nodata_count/pixeltutti * 100

        if pct_invalid <= 50.0:
            print(i, 'immagine valida ' + str(pct_valid))
            valide.append((i, pct_invalid))
        else:
            print(i, 'immagine non valida ' + str(pct_valid))
            nonvalide.append((i, pct_invalid))
```

In [ ]:

```
nonvalide
```

In [ ]:

```
#crea i singoli tif per campo e per data con filtraggio in base a percentuale no data
rasterintersection = []
for polygon in polygons:
    # Estrai il nome dell'ID per aggiungerlo ai path
    subdirshp,shp=os.path.split(str(polygon))
    shp= os.path.splitext(shp)

    # legge il .shp di input e la sua geometria
    with fiona.open(polygon,'r')as SHP:
        fid_geometry=[feature["geometry"] for feature in SHP]

        for raster_path in ndvi2mesi:

            # Trova il FID che si sovrappone al tile
            raster = gdal.Open(raster_path)
            vector = ogr.Open(polygon)
            transform = raster.GetGeoTransform()
            pixelWidth = transform[1]
            pixelHeight = transform[5]
            cols = raster.RasterXSize
            rows = raster.RasterYSize
            xLeft = transform[0]
            yTop = transform[3]
            xRight = xLeft+cols*pixelWidth
            yBottom = yTop+rows*pixelHeight
            ring = ogr.Geometry(ogr.wkbLinearRing)
            ring.AddPoint(xLeft, yTop)
            ring.AddPoint(xLeft, yBottom)
            ring.AddPoint(xRight, yBottom)
            ring.AddPoint(xRight, yTop)
            ring.AddPoint(xLeft, yTop)
            rasterGeometry = ogr.Geometry(ogr.wkbPolygon)
            rasterGeometry.AddGeometry(ring)
            layer = vector.GetLayer()
            feature = layer.GetFeature(0)
            vectorGeometry = feature.GetGeometryRef()
            overlap=rasterGeometry.Intersect(vectorGeometry)

            if overlap == True:

                # Estrai il nome del tile per aggiungerlo ai path
                subdir,file=os.path.split(str(raster_path))
                file= os.path.splitext(file)
                file=file[0].split('_')
                rasterintersection.append(raster_path)
                #print(shp[0],'  ',raster_path)
                #print(raster_path)

                  # Crea una cartella con il nome dell'ID e del tile
                os.makedirs(outputPath + "/FID_{}".format(shp[0]),exist_ok=True)

                 # Crea il path di uscita del tile ritagliato
                output_path =outputPath + "/FID_{}".format(shp[0]) + "/{}_{}".format(fi
le[0],file[1]) + ".tif"

                #os.system("gdalwarp -dstnodata -32768 -crop_to_cutline -cutline  " + p
olygon + " " + raster_path + " " + output_path)
                 # Legge il .tif e ritaglia in base al .shp
                with rasterio.open(raster_path) as tile:
```

```
                    out_image, out_transform = rasterio.mask.mask(tile, fid_geometry, c
rop=True)
                    out_meta = tile.meta
                    #controlla la % di nodata = '-32768'
                    nodata_value = -32768
                    nodata_count = np.count_nonzero(out_image == nodata_value)
                    total_count = out_image.shape[1] * out_image.shape[2]
                    pct_invalid = nodata_count/total_count * 100

                    if pct_invalid <= 95.0:
                        count_nan = np.count_nonzero(np.isnan(out_image))
                        total = out_image.shape[1] * out_image.shape[2]
                        invalid = count_nan/total * 100
                         #Se la % di 'nan' sia < 10 % dei pixel della clip salvala
                        if (invalid < (10 /total_count *100)):
                             #i = i + 1
#                              # Aggiorna i metadati nel sul .tif ritagliato
                            out_meta.update({"driver": "GTiff","height": out_image.shap
e[1],"width": out_image.shape[2],"transform": out_transform,"nodata": -32768})
#                              # Salva il nuovo .tif
                            with rasterio.open(output_path, "w", **out_meta) as dest:
                                dest.write(out_image)

                            #print (i,' Nuova clip: %s'%output_path,'\nFID: %s'% polyg
on,' ','Tile: %s'% raster_path,'\n',pct_invalid,' ',invalid, out_image.min(),out_image.
max(),'\n')

            else:
                pass
```

In [ ]:

```
invalid < (5 /total_count *100)
```

In [ ]:

```
#Leggi i tif precedentemente creati
rasterfiles = [os.path.join(outputPath, file) for outputPath, dirs, files in os.walk(ou
tputPath)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i tif precedentemente creati
len(rasterfiles)
```

In [ ]:

```
#crea il dataframe dall'analisi dei tif per ogni campo
T=[]
for i in rasterfiles:
    with rasterio.open(i) as tile:
        arr=tile.read()
        arr[arr == tile.nodata] = -32768
        meta = tile.meta.copy()
        #media = arr.mean()
        r=arr[arr>-10000]
        datesearch = re.search("([0-9]{8})", i)
        date = datetime.datetime.strptime(datesearch.group(), '%Y%m%d').date()
        stringadate = date.strftime("%Y/%m/%d")
        campo = i.split('/')[7]
        file = i.split('/')[8]
        T.append((campo, file, stringadate,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min
(),'%.2f'%r.std()))
        dataframe = pd.DataFrame(T,columns=['Campo','File','Data','ndvi_max','ndvi_mea
n','ndvi_min', 'ndvi_std'])
        print(campo, file, stringadate,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min
(),'%.2f'%r.std())
```

In [ ]:

```
#salva il dataframe in un tif
dataframe.to_csv ('/home/jovyan/work/jolanda_grano_duro/output/NDVI/ndvi_analisi.csv')
```

In [ ]:

```
#controlla il dataframe
dataframe
```

In [ ]:

```
#ristruttura il dataframe
df = dataframe[['Data','Campo','File','ndvi_max','ndvi_mean','ndvi_min','ndvi_std']]
df[['ndvi_max','ndvi_mean','ndvi_min','ndvi_std']] = df[['ndvi_max','ndvi_mean','ndvi_m
in','ndvi_std']].astype(float)
df['Data'] = df['Data'].astype('datetime64[ns]')
#df['Data'] = df['Data'].str.replace('/W', '-')
df.set_index('Data',inplace=True)
df.sort_values('Data')
```

In [ ]:

```
len(df['Campo'].unique())
```

In [ ]:

```
# Salva grafici dell'analisi NDVI
os.makedirs('/home/jovyan/work/jolanda_grano_duro/output/grafici_NDVI/',exist_ok=True)
for i, g in df.groupby(['Campo']):
    DF= g[['Campo',     'ndvi_max',     'ndvi_mean','ndvi_min', 'ndvi_std']]
    DF.plot(figsize=(11,9),title='NDVI {}'.format(i))
    plt.savefig('/home/jovyan/work/jolanda_grano_duro/output/grafici_NDVI/NDVI_{}.jpeg'
.format(i))
```

N MODIFIED SOIL-ADJUSTED VEGETATION INDEX (MSAVI) ANALYSIS

In [ ]:

```
#importa le librerie
import os
import fiona
import glob
import numpy as np
import pandas as pd
from osgeo import gdal, ogr
import rasterio
import regex as re
from rasterio.shutil import delete
from rasterio import mask
from rasterio.merge import merge
import datetime
import matplotlib.pyplot as plt
from datetime import date, timedelta
```

In [ ]:

```
#setta i percorsi di input ed output
root = '/home/jovyan/satellitedata/it/master/MSAVI/'
shpfiles='/home/jovyan/work/jolanda_grano_duro/input/shp_jolanda_grano_duro'
outputPath='/home/jovyan/work/jolanda_grano_duro/output/MSAVI/'
```

In [ ]:

```
#prendi tutti i .tif
rasterfiles = [os.path.join(root, file) for root, dirs, files in os.walk(root)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i .tif totali
len(rasterfiles)
```

In [ ]:

```
#imposta le date di inizio e fine analisi
sdate = date(2018, 11, 1)   # start date
edate = date(2019, 6, 30)   # end date
delta = edate - sdate       # as timedelta

DATE = []

for i in range(delta.days + 1):
    day = sdate + timedelta(days=i)
    DATE.append(day.strftime('%Y%m%d'))
```

In [ ]:

```
#crea la lista dei tif ricadenti in quelle date
ndvi2mesi=[]
for s in rasterfiles:
    for l in DATE:
        if l in s:
            ndvi2mesi.append(s)
```

In [ ]:

```
#crea le cartelle dal fid dello shape e suddividi gli shape
os.makedirs(outputPath + "/FID/", exist_ok=True)
with fiona.open(shpfiles) as source:
    meta = source.meta
    for f in source:
        ID=f['properties']['fid']
        #ID=f['id']
        outfile = os.path.join(outputPath + "/FID/{}.shp".format(int(ID)))
        with fiona.open(outfile, 'w', **meta) as sink:
            sink.write(f)
```

In [ ]:

```
#leggi ogni singolo shape
polygons = glob.glob(outputPath +'/FID/*.shp')
```

In [ ]:

```
len(polygons)
```

In [ ]:

```
# crea cartella dove mandare raster non validi
os.makedirs(outputPath + "/FID/empty_raster", exist_ok=True)
```

In [ ]:

```
#crea i singoli tif per campo e per data con filtraggio in base a percentuale no data
rasterintersection = []
for polygon in polygons:
    # Estrai il nome dell'ID per aggiungerlo ai path
    subdirshp,shp=os.path.split(str(polygon))
    shp= os.path.splitext(shp)

    # legge il .shp di input e la sua geometria
    with fiona.open(polygon,'r')as SHP:
        fid_geometry=[feature["geometry"] for feature in SHP]

        for raster_path in ndvi2mesi:

            # Trova il FID che si sovrappone al tile
            raster = gdal.Open(raster_path)
            vector = ogr.Open(polygon)
            transform = raster.GetGeoTransform()
            pixelWidth = transform[1]
            pixelHeight = transform[5]
            cols = raster.RasterXSize
            rows = raster.RasterYSize
            xLeft = transform[0]
            yTop = transform[3]
            xRight = xLeft+cols*pixelWidth
            yBottom = yTop+rows*pixelHeight
            ring = ogr.Geometry(ogr.wkbLinearRing)
            ring.AddPoint(xLeft, yTop)
            ring.AddPoint(xLeft, yBottom)
            ring.AddPoint(xRight, yBottom)
            ring.AddPoint(xRight, yTop)
            ring.AddPoint(xLeft, yTop)
            rasterGeometry = ogr.Geometry(ogr.wkbPolygon)
            rasterGeometry.AddGeometry(ring)
            layer = vector.GetLayer()
            feature = layer.GetFeature(0)
            vectorGeometry = feature.GetGeometryRef()
            overlap=rasterGeometry.Intersect(vectorGeometry)

            if overlap == True:

                # Estrai il nome del tile per aggiungerlo ai path
                subdir,file=os.path.split(str(raster_path))
                file= os.path.splitext(file)
                file=file[0].split('_')
                rasterintersection.append(raster_path)
                #print(shp[0],'  ',raster_path)
                #print(raster_path)

                  # Crea una cartella con il nome dell'ID e del tile
                os.makedirs(outputPath + "/FID_{}".format(shp[0]),exist_ok=True)

                 # Crea il path di uscita del tile ritagliato
                output_path =outputPath + "/FID_{}".format(shp[0]) + "/{}_{}".format(fi
le[0],file[1]) + ".tif"

                #os.system("gdalwarp -dstnodata -32768 -crop_to_cutline -cutline  " + p
olygon + " " + raster_path + " " + output_path)
                 # Legge il .tif e ritaglia in base al .shp
                with rasterio.open(raster_path) as tile:
```

```
                    out_image, out_transform = rasterio.mask.mask(tile, fid_geometry, c
rop=True)
                    out_meta = tile.meta
                    #controlla la % di nodata = '-32768'
                    nodata_value = -32768
                    nodata_count = np.count_nonzero(out_image == nodata_value)
                    total_count = out_image.shape[1] * out_image.shape[2]
                    pct_invalid = nodata_count/total_count * 100

                    if pct_invalid <= 95.0:
                        count_nan = np.count_nonzero(np.isnan(out_image))
                        total = out_image.shape[1] * out_image.shape[2]
                        invalid = count_nan/total * 100
                         #Se la % di 'nan' sia < 10 % dei pixel della clip salvala
                        if (invalid < (10 /total_count *100)):
                             #i = i + 1
#                             # Aggiorna i metadati nel sul .tif ritagliato
                             out_meta.update({"driver": "GTiff","height": out_image.sha
pe[1],"width": out_image.shape[2],"transform": out_transform,"nodata": -32768})
#                             # Salva il nuovo .tif
                             with rasterio.open(output_path, "w", **out_meta) as dest:
                                 dest.write(out_image)

                             #print (i,' Nuova clip: %s'%output_path,'\nFID: %s'% polyg
on,' ','Tile: %s'% raster_path,'\n',pct_invalid,' ',invalid, out_image.min(),out_image.
max(),'\n')

            else:
                pass
```

In [ ]:

```
#leggi i tif precedentemente creati
rasterfiles = [os.path.join(outputPath, file) for outputPath, dirs, files in os.walk(ou
tputPath)
               for file in files if os.path.splitext(file)[1] == '.tif' ]
```

In [ ]:

```
#conta i tif precedentemente creati
len(rasterfiles)
```

In [ ]:

```
#crea il dataframe dall'analisi dei tif per ogni campo
T=[]
for i in rasterfiles:
    with rasterio.open(i) as tile:
        arr=tile.read()
        arr[arr == tile.nodata] = -32768
        meta = tile.meta.copy()
        #media = arr.mean()
        r=arr[arr>-10000]
        datesearch = re.search("([0-9]{8})", i)
        date = datetime.datetime.strptime(datesearch.group(), '%Y%m%d').date()
        stringadate = date.strftime("%Y/%m/%d")
        campo = i.split('/')[7]
        file = i.split('/')[8]
        T.append((campo, file, stringadate,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min
(),'%.2f'%r.std()))
        dataframe = pd.DataFrame(T,columns=['Campo','File','Data','msavi_max','msavi_me
an','msavi_min', 'msavi_std'])
        print(campo, file, stringadate,'%.2f'%r.max(),'%.2f'%r.mean(),'%.2f'%r.min
(),'%.2f'%r.std())
```

In [ ]:

```
#salva il dataframe in un tif
dataframe.to_csv ('/home/jovyan/work/jolanda_grano_duro/output/MSAVI/msavi_analisi.csv'
)
```

In [ ]:

```
#controlla il dataframe
dataframe
```

In [ ]:

```
#ristruttura il dataframe
df = dataframe[['Data','Campo','File','msavi_max','msavi_mean','msavi_min','msavi_std'
]]
df[['msavi_max','msavi_mean','msavi_min','msavi_std']] = df[['msavi_max','msavi_mean',
'msavi_min','msavi_std']].astype(float)
df['Data'] = df['Data'].astype('datetime64[ns]')
#df['Data'] = df['Data'].str.replace('/W', '-')
df.set_index('Data',inplace=True)
df.sort_values('Data')
```

In [ ]:

```
len(df['Campo'].unique())
```

In [ ]:

```
# Salva grafici dell'analisi MSAVI
os.makedirs('/home/jovyan/work/jolanda_grano_duro/output/MSAVI/grafici_MSAVI/',exist_ok
=True)
for i, g in df.groupby(['Campo']):
    DF= g[['Campo',     'msavi_max',     'msavi_mean','msavi_min',       'msavi_std']]
    DF.plot(figsize=(11,9),title='MSAVI {}'.format(i))
    plt.savefig('/home/jovyan/work/jolanda_grano_duro/output/MSAVI/grafici_MSAVI/MSAVI_
{}.jpeg'.format(i))
```

In [ ]:

```

```

List of figures

List of Tables