*remote sensing*

*Article*

# Hyperfidelis: A Software Toolkit to Empower Precision Agriculture with GeoAI

**Vasit Sagan [1,2,3,*], Roberto Coral [1], Sourav Bhadra [1], Haireti Alifu [3], Omar Al Akkad [3], Aviskar Giri [3] and Flavio Esposito [2]**

[1] Department of Earth and Atmospheric Sciences, Saint Louis University, Saint Louis, MO 63108, USA
[2] Department of Computer Science, Saint Louis University, Saint Louis, MO 63108, USA
[3] Taylor Geospatial Institute, Saint Louis, MO 63108, USA; omar.alakkad@taylorgeospatial.org (O.A.A.)
[*] Correspondence: vasit.sagan@slu.edu; Tel.: +1-314-977-5156

**Abstract:** The potential of artificial intelligence (AI) and machine learning (ML) in agriculture for improving crop yields and reducing the use of water, fertilizers, and pesticides remains a challenge. The goal of this work was to introduce Hyperfidelis, a geospatial software package that provides a comprehensive workflow that includes imagery visualization, feature extraction, zonal statistics, and modeling of key agricultural traits including chlorophyll content, yield, and leaf area index in a ML framework that can be used to improve food security. The platform combines a user-friendly graphical user interface with cutting-edge machine learning techniques, bridging the gap between plant science, agronomy, remote sensing, and data science without requiring users to possess any coding knowledge. Hyperfidelis offers several data engineering and machine learning algorithms that can be employed without scripting, which will prove essential in the plant science community.

**Keywords:** artificial intelligence; data science; multispectral image; hyperspectral image; machine learning; remote sensing; plant phenotyping

**Citation:** Sagan, V.; Coral, R.; Bhadra, S.; Alifu, H.; Al Akkad, O.; Giri, A.; Esposito, F. Hyperfidelis: A Software Toolkit to Empower Precision Agriculture with GeoAI. *Remote Sens.* **2024**, *16*, 1584. https://doi.org/10.3390/rs16091584

Academic Editor: Kun Jia

Received: 26 March 2024
Revised: 13 April 2024
Accepted: 23 April 2024
Published: 29 April 2024

## 1. Introduction

Improving food security is a daunting task for humanity as the world population is projected to surpass nine billion by 2050, necessitating a 70% increase in agricultural production [1]. With limited arable land and water resources, developing technologies that enable effective screening and identifying high yield crops, monitoring growth, and prescribing irrigation and nitrogen applications for proactive farm management are arguably the best strategies to achieve that goal [2–4]. In addition to precision agriculture, these technologies include genetic manipulation [5], nanotechnology [6], genomics [7], droplet irrigation [8], and computerization. These approaches aims to enhance crop breeding and develop resilient varieties capable of thriving under future climate conditions [9].

Geospatial artificial intelligence (GeoAI) facilitated by recent advancements in big data analytics, computing, miniaturized sensors and drones, and satellite-based sensor web has demonstrated its significant potential for improved agricultural production [10]. GeoAI has been used to predict which crops to plant each year and when the best dates to sow and harvest are by analyzing soil, climate, and historic crop performance data [11]. By combining innovations in geospatial science (in particular Earth Observation) with the rapid growth of methods in machine learning, data mining, and high-performance computing, GeoAI has been increasingly used to advance phenotyping [12], yield prediction [13,14], crop breeding [15], and seed composition estimation [16].

The role of GeoAI and digital agriculture in small-scale farming needs careful consideration, particularly as two-thirds of rural populations in emerging nations have access to land plots of less than two hectares [17]. Despite being the backbone to agricultural innovation, smallholder farmers are often slow to adopt new technologies due to limited access, expertise, and financial resources [18]. Family farming, prevalent in Asia and the Pacific, constitutes a significant portion of global agriculture [19]. While smallholder farmers stand to benefit from digital agricultural advancements through enhanced market transparency, increased production, and improved logistics [20–22] challenges persist. There is a risk of widening the digital divide between small and large farms, as technological advancements become more commercialized [23]. Addressing this issue requires strategies to ensure reasonable access to digital technologies for smallholder farmers as the agricultural landscape evolves.

Low cost is a critical factor in allowing farmers to benefit from digital technologies as farming is an expansive operation with thin margin. When choosing software for agricultural monitoring, it is important to consider the cost-effectiveness of open-source software against proprietary software. As remote sensing technology advances, big data of hyperspectral, multispectral, synthetic aperture radar (SAR), thermal, and light detection and ranging (LiDAR) at global scale become increasingly available, necessitting farming to incorporate cutting-edge sensor data into their operations with more complicated analyses. However, there is a knowledge gap in how to convert multi-scale and multi-source data into a set of meaningful deliverables that can be utilized for informed decision-making in agriculture. Furthermore, most of current software tools are limited for widespread use by proprietary nature or requires extensive training and understanding of the industry. As a result, farmers become disadvatanged communities in this rapidly evoling digital era with the lack of tools available to them to handle geospaital big data [24].

Although several studies of GeoAI for agriculture have demonstrated its promise, the primary bottleneck in advancing GeoAI for agriculture lies in the scarcity of labeled training datasets that are "machine learning-ready". This challenge is compounded by the immense size of imagery datasets, especially when considering global coverage, which poses an added challenge for both machine learning algorithms and cloud computing infrastructure. This may be addressed by implementing preprocessing techniques, data standardization procedures, or other measures aimed at optimizing the compatibility of imagery data with machine learning frameworks. Furthermore, there is still a lack of automation in the processing and analysis of remote sensing imagery; to the best of our knowledge, there are no platforms that can cover the entire loop in imagery data processing, including feature extraction, training, and application of advanced AI algorithms. The only related solutions we found that partially cover some of those topics are FIELDimageR [25], an R package that focuses on extracting plot-level results from field crop imagery but requires in-depth knowledge of plant science and coding experience, PlantCV [26], a toolkit that provides a wide array of image processing functions for plant phenotyping analysis that can be used in Python but without providing a graphical user interface (GUI) to guide the users, and Polly [27], an easy to use online data analysis and modeling tool that mainly focuses on data analysis and integration.

In this work we present Hyperfidelis 1.0, a GeoAI software (refer as Hyperfidelis hearafter) that enables farmers that manage farms of all sizes, data scientists and plant scientists to extract valuable information from geospatial data (e.g., UAV or satellite imagery), improving agricultural monitoring and therefore allowing for a better awareness of food supply challenges. By employing Hyperfidelis, agricultural practitioners can obtain reliable information on the health (pre-visual water stress, plant disease, and nutrient deficiency), vitality, and growth of their agricultural fields, and by examining that information, they can comprehend the past and "predict the future", allowing them to make more timely and informed agricultural decisions. Hyperfidelis provides a comprehensive workflow that includes imagery visualization, plot-level zonal statistics, feature extraction, and modeling of key agricultural traits (e.g., yield, protein content) in a completely automated AI framework. The program offers a graphical user interface with state-of-the-art machine learning algorithms not requiring users to possess any coding knowledge. Hyperfidelis provides a variety of data engineering and machine learning algorithms that can be used without scripting, which will prove essential in the plant science community. For instance, in crop breeding operations involving UAVs and satellite remote sensing sensors, Hyperfidelis will significantly improve plant phenotyping efficiency and accuracy. The features of Hyperfidelis were identified by examining the requirements for various tools in agricultural decision-making and remote sensing.

## 2. Materials and Methods

### *2.1. Software Design*

Hyperfidelis offers a user-friendly GUI powered by Python, chosen for its platform independence, simple syntax, and extensive documentation. Python's flexibility and readability make it ideal for beginners and efficient for machine learning tasks, supported by libraries like Keras 2.8.0 [28], TensorFlow 2.8.0 [29], and Scikit-learn 0.24.2 [30]. The frontend utilizes PyQt5 5.15.2 [31], leveraging Qt's capabilities for cross-platform GUI development. Qt Designer streamlines GUI creation with drag-and-drop functionality and supports various styles and resolutions. Hyperfidelis integrates open-source Python libraries such as Rasterio 1.2.10 [32] and GeoPandas 0.10.2 [33] for geospatial data management and manipulation, along with Scikit-learn 0.24.2 and TensorFlow 2.8.0 for machine learning and deep learning tasks. By utilizing open-source modules exclusively, Hyperfidelis ensures accessibility and fosters a collaborative development environment.

Accurate spatial representation is critical in geospatial data, requiring a coordinate system and datum for precise positioning on Earth. Hyperfidelis, focusing on enhancing agricultural output, supports geospatial visualization by accessing image datum and coordinate systems. It reads various formats of raster and vector datasets including satellite imagery, UAV-based RGB imagery, multispectral imagery, and hyperspectral imagery, using libraries like rasterio 1.2.10 for manipulation and Spectral Python 0.22.4 [34] for hyperspectral data 3D spectral hypercube visualization [35] (Figure 1) and processing (Figure 2). Moreover, Hyperfidelis extracts color channels and normalizes them for visualization, facilitated by PyQt5 5.15.2 for frontend display. Hyperfidelis extends its capabilities to analyze an array of remote sensing datasets derived from UAVs and satellites [14]. This encompasses datasets from airborne, spaceborne, and field-level sensors, provided they are in TIFF format and contain both spectral and spatial information.

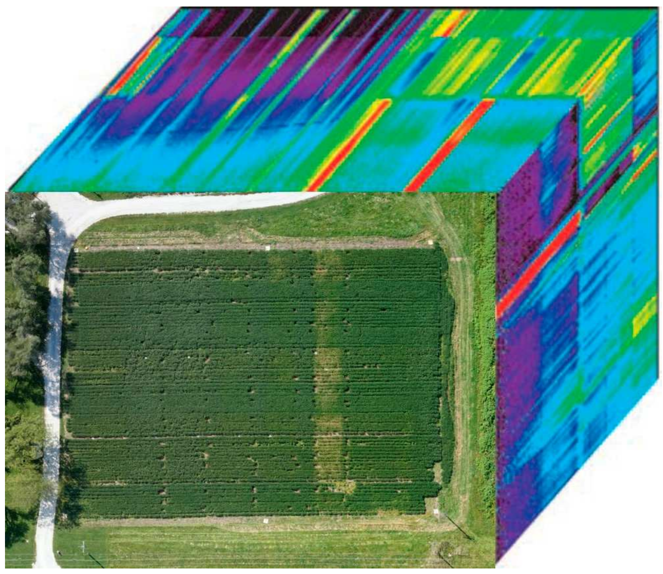


**Figure 1.** Illustration of a 3D spectral hypercube: height and width are the spatial dimensions, while the spectral dimension is represented by the bands composing the image.

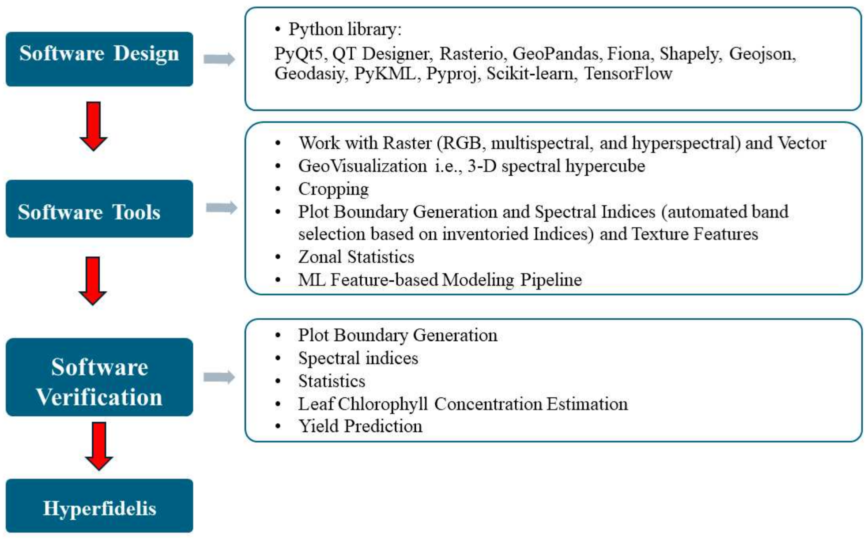

Software Design
• Python library:
PyQt5, QT Designer, Rasterio, GeoPandas, Fiona, Shapely, Geojson, Geodasiy, PyKML, Pyproj, Scikit-learn, TensorFlow
Software Tools
• Work with Raster (RGB, multispectral, and hyperspectral) and Vector
• GeoVisualization i.e., 3-D spectral hypercube
• Cropping
• Plot Boundary Generation and Spectral Indices (automated band selection based on inventoried Indices) and Texture Features
• Zonal Statistics
• ML Feature-based Modeling Pipeline
Software Verification
• Plot Boundary Generation
• Spectral indices
• Statistics
• Leaf Chlorophyll Concentration Estimation
• Yield Prediction
Hyperfidelis

**Figure 2.** Diagram of the development and functionalities of Hyperfiedelis.

## *2.2. Hyperfidelis Funtionality*

When working with large images, it is sometimes important to concentrate solely on a portion of the image, for example, to extract features from a fraction of the original image. Hyperfidelis allows users to crop any sort of image they provide (RGB, multispectral, or hyperspectral). Cropping can be performed in two ways: manually or with the use of a shapefile.

When employing a shapefile to crop an image, the user will load a file with one or more polygons that will be used to crop the original image. The loading of the image is handled by rasterio 1.2.10 and Spectral Python 0.22.4, while the visualization of the original image and the hold–drag–release feature is managed by PyQt5 5.15.2. The coordinate reference systems (CRS) of the file and raster image are the first element Hyperfidelis examines since they must match; if they do not, their projections will be transformed to match. If the shapefile contains multiple polygons, the crop function will be performed separately for each polygon, resulting in multiple cropped images. Another condition to check is whether the shapefile's spatial extent overlaps with the image to be cropped.

The platform supports all of the most common file formats used for representing points, lines, and polygons, including ESRI shapefile, WKT (Well-Known Text), JSON/GeoJSON, and KML (Keyhole Markup Language).

When working with these types of files, there are a few points to keep in mind. WKT, JSON/GeoJSON, and KML do not normally contain information about their coordinate system, although ESRI shapefiles (.shp) do. The former usually use the World Geodetic System (WGS84) as their reference coordinate system, which is the same as the one used by the Global Positioning System (GPS). As a result, when overlapping a WKT, JSON/GeoJSON, or KML file to an image, the platform verifies if the image's CRS is WGS84 as well, and if it is not, it converts the file's coordinates to the format used by the image's CRS.

Furthermore, because the ESRI shapefile format is a proprietary format, the process of creating a shapefile is standardized. Only appropriate software (e.g., ArcGIS, QGIS) and libraries (e.g., fiona 1.8.21) can open, read, and write this type of file. Creating WKT, JSON/GeoJSON, and KML files, on the other hand, is a "less standardized" operation; users can access and edit those types of files with any text editor, potentially introducing errors. The software assumes the following standards for reading a WKT, JSON/GeoJSON, and KML file: Google KML 2.2 Schema (Open Geospatial Consortium, Arlington, VA, USA, 2008) for KML, RFC 7946 [36] for GeoJSON, and OGC standard (Open Geospatial Consortium, Arlington, VA, 2019) for WKT.

Hyperspectral imagery typically comes with a header file containing the image's metadata, as we discussed in the previous section. When cropping hyperspectral images, the platform generates a new header with most of the metadata remaining the same except for the height (number of rows) and width (number of columns) which will reflect the cropped image's dimensions.

Several libraries and packages were required to implement the cropping functionality, which supports all those file formats and performs CRS transformation as necessary. The libraries that have been used are summarized in Table 1. Once the image has been cropped, the platform shows the users a preview of the resulting image, and the user can save the cropped image if they wish.

**Table 1.** List of Python libraries which were used to implement the cropping functionality.

| Library | Source | Use Case |
|---|---|---|
| Rasterio 1.2.10 | Gillies [32] | Open, crop, and reproject imagery raster |
| Fiona 1.8.21 | Gillies [37] | Read ESRI shapefiles |
| GeoPandas 0.10.2 | Geopandas developers [38] | Produce a GeoDataFrame from a shapefile, which will be plotted and visualized on top of the raster image |
| Shapely 1.8.2 | Gillies [39] | Serialize the content of a WKT file into a polygon object |
| Json 0.9.8 | Python's built-in library | Parse the text contained in a JSON/GeoJSON file and convert it to a Python dictionary |
| Geojson 2.5.0 | Butler [36] | Construct a GeoJSON feature from a shapely polygon |
| Geodaisy 0.1.1 | Brochet-Nyugen [40] | Convert GeoJSON to WKT |
| PyKML 0.2.0 | Hengl [41] | Parse a KML file |
| Pyproj 3.1.0 | Megies [42] | Transform the shapefile's CRS from one to another |

## 2.3. Feature Extraction

### 2.3.1. Spectral Indices

Plant water stress, growth conditions, or disease infections can be detected indirectly using spectral vegetation indices (VIs). These indices are frequently used to assess and detect changes in plant physiology and chemistry. They have been developed to extract various plant properties from remote sensing data, such as pigment content, leaf area, water content, and so on, based on information from a few wavelengths [43].

Vegetation indices derived from remote sensing-based overhead images are simple and effective methods for assessing vegetation cover, vigor, and growth dynamics, among other factors. They have been widely used in remote sensing applications employing a variety of airborne and satellite platforms, with recent advancements utilizing UAV [44]. From RGB, multispectral, and hyperspectral images, Hyperfidelis can generate more than 100 vegetation indices.

Imagery collected from various sensors have different band designations, the platform will prompt the user to indicate the band designation (i.e., the band order representing wavelengths) for the image to be opened after the user has loaded it and intends to calculate some vegetation indices. MicaSense Altum sensors, for example, use Band 1 for blue, but Landsat 8 [45] uses Band 2. Only RGB and multispectral images require a band designation; hyperspectral data include a header file to obtain information about the band wavelengths. After the bands have been configured, the user can choose which vegetation indices to calculate through the form depicted in Figure 3a. Hyperfidelis provides 26 RGB indices, 46 multispectral indices, and 69 hyperspectral indices (Appendix A, Tables A1–A3).

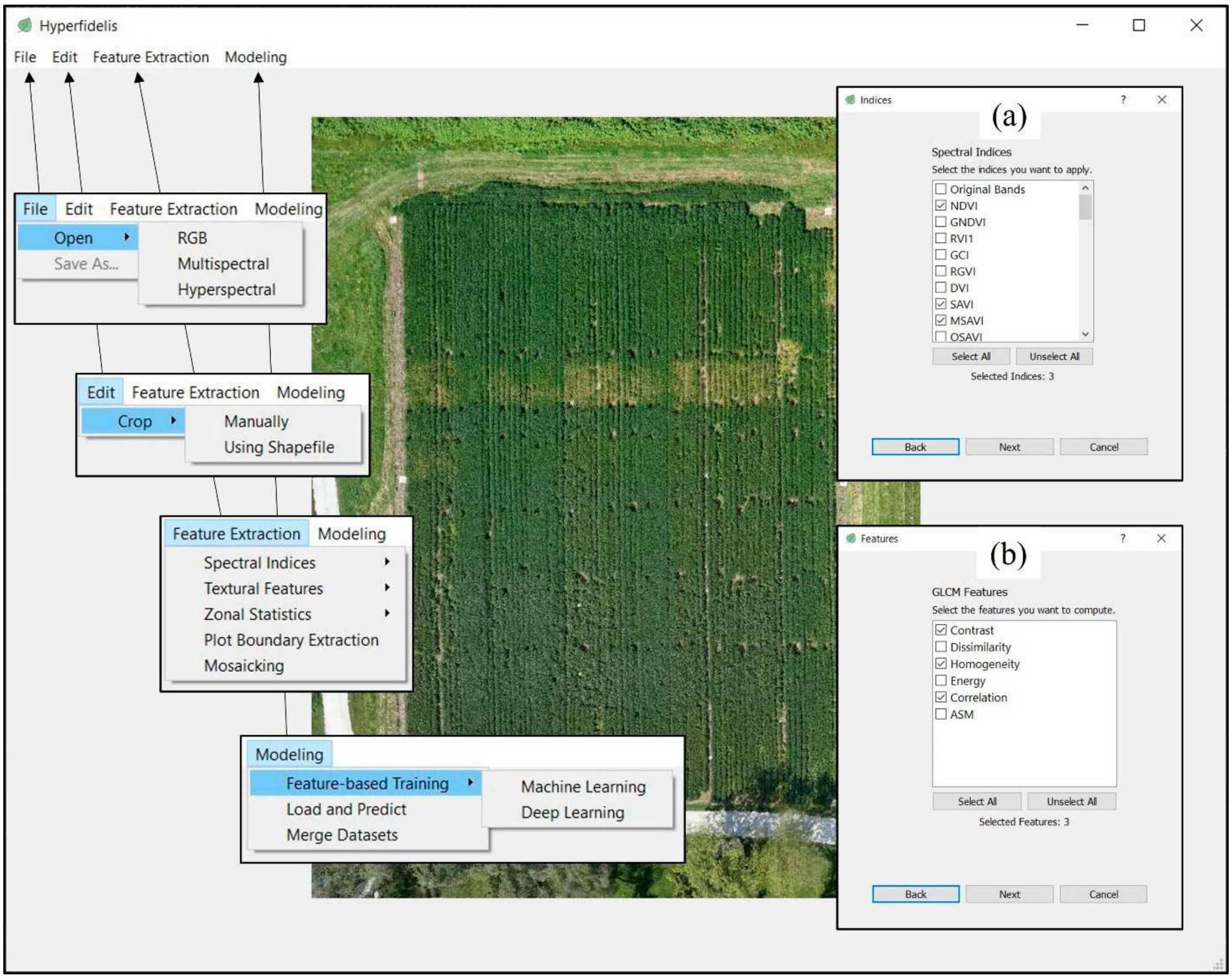


**Figure 3.** Geospatial visualization of an RGB image displayed in the main window of Hyperfidelis. (**a**) User input form for selecting vegetation indices for calculation. (**b**) GLCM features.

If a user selects an index that requires one or more band wavelengths that are not available in the loaded image, the platform will notify the user about the missing band(s) and skip that index. For RGB and multispectral images, individual bands can also be saved alongside the vegetation indices. Each index is calculated pixel by pixel, producing a one-dimensional image. To extract the bands and export the indices to an image file, rasterio 1.2.10 and Spectral Python 0.22.4 are used: rasterio manages RGB and multispectral data, while Spectral Python manages hyperspectral data.

### 2.3.2. Texture Features

A texture is a pattern, tonal or gray-level variations in an image [46]. Because it can provide additional information about features, texture analysis has become more important in remote sensing image processing and interpretation [47]. Textural information, which is defined as spatial variation in pixel intensities within an image, can be used to emphasize structural and geometric characteristics of plant canopies [48] and to prevent saturation in models that are not designed for the landscape's high spatial heterogeneity [49].

Textural information may be extracted from remote sensing imagery using a variety of approaches. Gray level co-occurrence matrix (GLCM) texture measurements have been the most often utilized approach. The GLCM texture feature is a matrix that is used to compute how many times pixel pairs with specified values appear in a picture. It describes the relationship between the intensity of one pixel and the intensity of another pixel within the given neighborhood. For describing image textures, the GLCM is considered an necessary complement to spectral analysis, which characterize the relative frequencies of two pixel brightness values linked by a spatial correlation [46]. Many researchers have determined that GLCM approaches are particularly effective in analyzing and processing remote sensing images and have used the textural information retrieved by the GLCM in a variety of applications [50,51].

Hyperfidelis allows extracting texture features from an image or a portion of an image (Figure 3b). When a user loads an image and wishes to extract texture features from it, the platform allows them to choose which area of the image they want to extract texture features from. Since GLCM extraction is a computationally intensive operation, utilizing only a portion of the image will substantially decrease processing time. If Hyperfidelis is executed on a high performance computer (HPC) equipped with multiple nodes, leveraging parallel processing, it is feasible to efficiently handle the computationally intensive task of calculating a GLCM for the entire image. This allows users with enhanced computational resources to benefit from full-image processing capabilities within a reasonable timeframe.

The user, in the following stage, will choose what features to compute via the form shown in Figure 3b. In fact, after extracting an image's GLCM, several properties such as contrast, dissimilarity, homogeneity, ASM (Angular Solar Moment), energy, and correlation may be obtained. Contrast is a metric for the number of local variations in an image, while dissimilarity is a measure for the distance between pairs of pixels, homogeneity is a metric for how evenly the elements are distributed in the matrix, ASM and energy are measurements for textural uniformity or pixel pair repetitions, and correlation is a criterion for gray-tone linear dependencies in an image. The platform will then compute the GLCM features specified and export each feature as an image file.

Certain optimization strategies are implemented in the platform to tackle computational challenges for extracting GLCM. First, since feature extraction must be performed for each band of the supplied imagery, parallelization is implemented: multiple subprocesses execute feature extraction for multiple bands in parallel. When filling the matrix for two consecutive pixels, the majority of the pixels are the same, with the exception of those on the left and right sides; hence, caching is used to speed up the matrix-filling process. Finally, the user can choose to extract features from a portion to reduce computation time, as previously stated.

For this feature, gdal 3.4.3 [52] reads the image and rasterio 1.2.10 writes the texture features to image files. Scikit-image 0.24.2 [53] is used to construct the GLCM matrix as well as to handle feature extraction. In terms of optimization, the platform uses the Python "multiprocessing 3.12.3" library to parallelize the feature extraction on separate subprocesses and PyTorch 2.0.0 [54] to efficiently extract patches from the image.

### 2.3.3. Zonal Statistics

Zonal statistics refers to computing statistical variables such as mean, max, and min values of a raster data unit defined by a polygon [55]. It is one of the most popular techniques that utilize raster and vector data for extracting information. A raster layer, a vector layer, and an accumulator are the input to the zonal statistics task. The vector layer is made up of a collection of disconnected polygons, such as plot boundaries. A large matrix comprising remote sensing data makes up the raster layer. The accumulator is a user-supplied function that computes the statistics of interest by taking pixel values one at a time. The output is a value for each polygon in the vector layer's accumulator [55]. For instance, if the raster represents the NDVI values of a field, the vector represents all plot boundaries in the field and the accumulator is the average function; the average NDVI for each plot will be the result of this problem.

Zonal statistics is widely used in a multitude of geospatial domains. Raster zonal statistics, for example, have been used to summarize raster data of forest products, agricultural goods, and tourism services in order to assess the economic worth of regions [56]. Raster zonal statistics is also a significant approach for summarizing raster values over a watershed to extract hydrological characteristics [57] and a valuable tool for evaluating the quality of DEM (Digital Elevation Model) data [58]. By integrating imagery of various bands and deleting mixed pixels, zonal statistics have also been utilized to increase the quality of land use and land cover (LULC) classification [59].

To combine raster and vector data, Hyperfidelis allows users to generate zonal statistics. After loading a raster image, the user may load some vector data in the form of polygons or points; in the case of points, buffers will be formed around them to produce the polygons on which the statistics will be calculated.

When a user loads a file with polygons, the platform conducts the same checks it does when trying to crop an image using a shapefile. To summarize, the checks performed are as follows: (1) verifying that the vector and raster data have the same CRS, and if not, transform their projections to match; (2) ensuring that the vector data contain only polygons; and (3) checking that the polygons overlap with the raster image. If those conditions are met, the polygons will be plotted on the image so that the user can see where they are in respect to the image.

If the user chooses to provide vector data containing points, the platform will check that the file supplied only contains points, and it will match the CRS of raster and vector data, just as it does with polygons. The platform will then display a form for the user to specify the radius (in meters) and shape (square or circle) of the buffers that will be formed around each point. If the selected shape is a square, the square buffers will have a side length equal to double the provided radius. The buffers are then generated, and if they overlap the raster image, they are visualized on the image.

Once the polygons are shown on the screen, the user can choose which zonal statistics to calculate. The options are mean, min, max, median, majority, sum, count, and range. Hyperfidelis will compute and export each statistic for each element in the vector dataset into a spreadsheet file and will save one file for each statistic (Figure 4a).

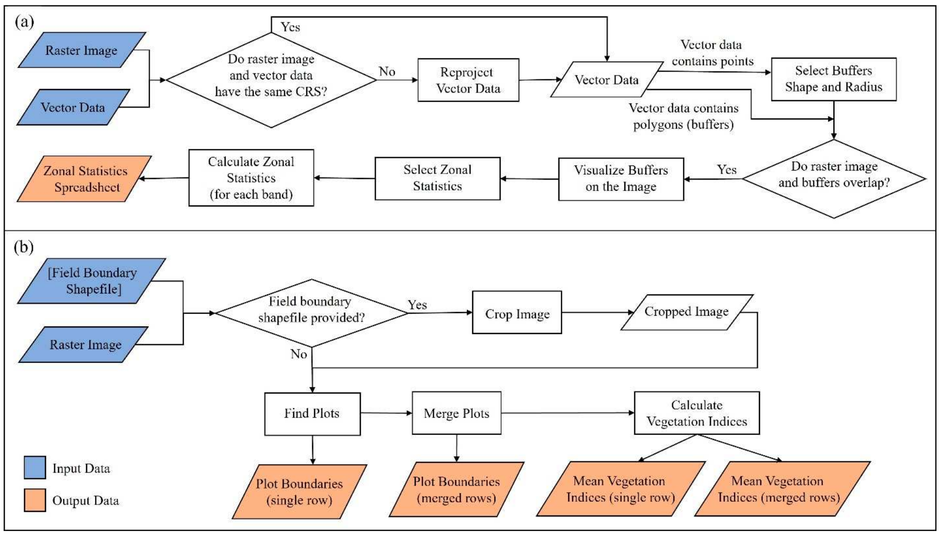

(a)
Raster Image
Vector Data
Do raster image and vector data have the same CRS?
Yes
No
Reproject Vector Data
Vector Data
Vector data contains points
Select Buffers Shape and Radius
Vector data contains polygons (buffers)
Do raster image and buffers overlap?
Yes
Visualize Buffers on the Image
Select Zonal Statistics
Calculate Zonal Statistics (for each band)
Zonal Statistics Spreadsheet
(b)
[Field Boundary Shapefile]
Raster Image
Field boundary shapefile provided?
Yes
No
Crop Image
Cropped Image
Find Plots
Merge Plots
Calculate Vegetation Indices
Plot Boundaries (single row)
Plot Boundaries (merged rows)
Mean Vegetation Indices (single row)
Mean Vegetation Indices (merged rows)
Input Data
Output Data

**Figure 4.** (**a**) Workflow for zonal statistics: the platform takes a raster image and some vector data (e.g., points or polygons). The vector data will be used to generate the buffers on which the zonal statistics will be calculated. (**b**) The plot boundary extraction pipeline takes as input a raster image and an optional field boundary file and outputs two plot boundary shapefiles (single-row and merged rows) and two spreadsheets with mean vegetation indices for each extracted plot (single-row and merged rows).

It is worth noting that most zonal statistics are calculated on two-dimensional images, i.e., images with only one band (e.g., NDVI). However, the platform can generate zonal statistics on imagery with several bands (e.g., RGB) as well, in which case the statistics will be produced for each individual band. The majority of the libraries that are used to implement the image cropping feature are equally utilized to achieve this feature. Hyperfidelis employs rasterstats 0.16.0 [60], a library for collecting information from geospatial data based on vector geometries, to calculate zonal statistics.

#### 2.3.4. Plot Boundary Extraction

In experimental research farms, dozens or hundreds of crop varities are planted in smaller area unites known as "plots". In farming practice, a plot can be many acres of large field grown with a single crop varity. Plant coverage, growth, flowering state, and other phenomena may be explained using traits taken from the plots. Finding the exact position of plots to extract plot (or field) boundary from an orthomosaic image is a crucial preparatory step in obtaining such information. Because plots may not be exactly aligned or evenly dispersed in a field, extraction of plots using techniques that assume uniform spacing is frequently incorrect [61]. Hyperfidelis includes a pipeline for extracting plot boundaries from a raster image and calculating vegetation indices and zonal statistics for those plots.

The pipeline's input (which the user must supply) is made up of the input image (currently, only multispectral images are supported) and the number of rows, columns, and rows per plot in the image. The user can additionally provide a field boundary shapefile from which the plots can be extracted. We recommend providing a custom shapefile, as the model assumes uniform distribution of plots. GeoTIFF and JPEG are supported for image formats, whereas ESRI shapefiles, KML, JSON/GeoJSON, and WKT are supported for boundary files.

The steps in the pipeline are as follows: (1) read the imagery and crop it using the field boundary file (if field boundary file is provided), (2) extract plot boundaries, (3) merge plots, (4) calculate vegetation indices, and (5) for each identified plot, calculate zonal statistics (mean).

If a field boundary shapefile is supplied, the input image will be cropped using the region of interest (ROI) in the file; the cropped image will be stored to disk and used as input for the pipeline's next stages (i.e., plot boundary extraction). The original input image will be utilized if this is not the case. The platform verifies if the image CRS and the file CRS are the same before cropping the image (match). If they do not match, the file is reprojected to match the image CRS before cropping. The platform employs GRID (GReenfield Image Decoder) 1.3.11 [62], a Python module to identify plot grid patterns without the requirement for user interaction, to extract plot boundaries; we customized the module to detect and extract merged plots as well. The plot boundary extraction feature is independent of ground devices providing geographic information, does not need drawing lines or polygons, is tolerant of plot alterations related to plant interaction inside and across plots, and requires no training. A plot boundary shapefile for single rows is generated after the plot extraction phase.

The algorithm will utilize the plot boundary shapefile created in the previous stage to construct a new shapefile of merged rows, depending on the number of rows per plot supplied as input after the plot boundaries have been detected. For merged rows, the output will be a plot boundary shapefile as well. At this stage, 43 vegetation indices are generated, and the mean of each vegetation index is determined for each discovered plot (zonal statistics), with the results exported in two spreadsheets (mean vegetation indices for single-row and merged row). The workflow of the pipeline is depicted in Figure 4b.

### *2.4. Feature-Based Modeling Pipeline*

One of the key features of Hyperfidelis is the ability to provide users with a system for feature-based machine learning regression. The aim is to give non-ML users a tool that would allow them to predict a certain variable on a dataset and identify the optimal model for that dataset without writing a single line of code. With that in mind, users may load any sort of spreadsheet dataset into the platform, including CSV and Excel files; the only condition is that the dataset has a header (in the first row), which is often used to indicate the names of the fields/columns. Aside from the header, the spreadsheet must include numeric data; the number of rows and columns in the file is unrestricted.

When working with large numeric datasets, one typical concern is that various input data or features may have different scales (e.g., one variable ranging from 0 to 10 and another ranging from −100 to 1000), which can cause issues when utilizing some machine learning models (i.e., Linear Regression, Support Vector Machine, and sNeural Networks). Unscaled features can, in fact, slow down the training phase and lower the algorithm's performance. Another difficulty that might arise when dealing with a high number of variables (e.g., hundreds of bands of a hyperspectral data) is that if the ML model is fed with too many features, the training time can exponentially increase, but more critically, the model will likely overfit the data.

To reduce the amount of preprocessing (or any action) performed by the user, even when working with large datasets, there is no need to manually split the dataset into training and test sets, separate features and the target variable, or perform any feature engineering (i.e., feature scaling and dimensionality reduction) on the dataset because this will be handled by the platform before training. As a result, the user may simply import their spreadsheet dataset into the platform as-is, and the platform will walk them through the process if preprocessing is necessary or desired.

Hyperfidelis implements a pipeline to perform feature-based modeling that can be split down into the following key components: input data loading, features and label configuration, selection of machine learning algorithms and hyperparameter tuning, parameters configuration, model training, and performance evaluation.

#### 2.4.1. Input Data Loading

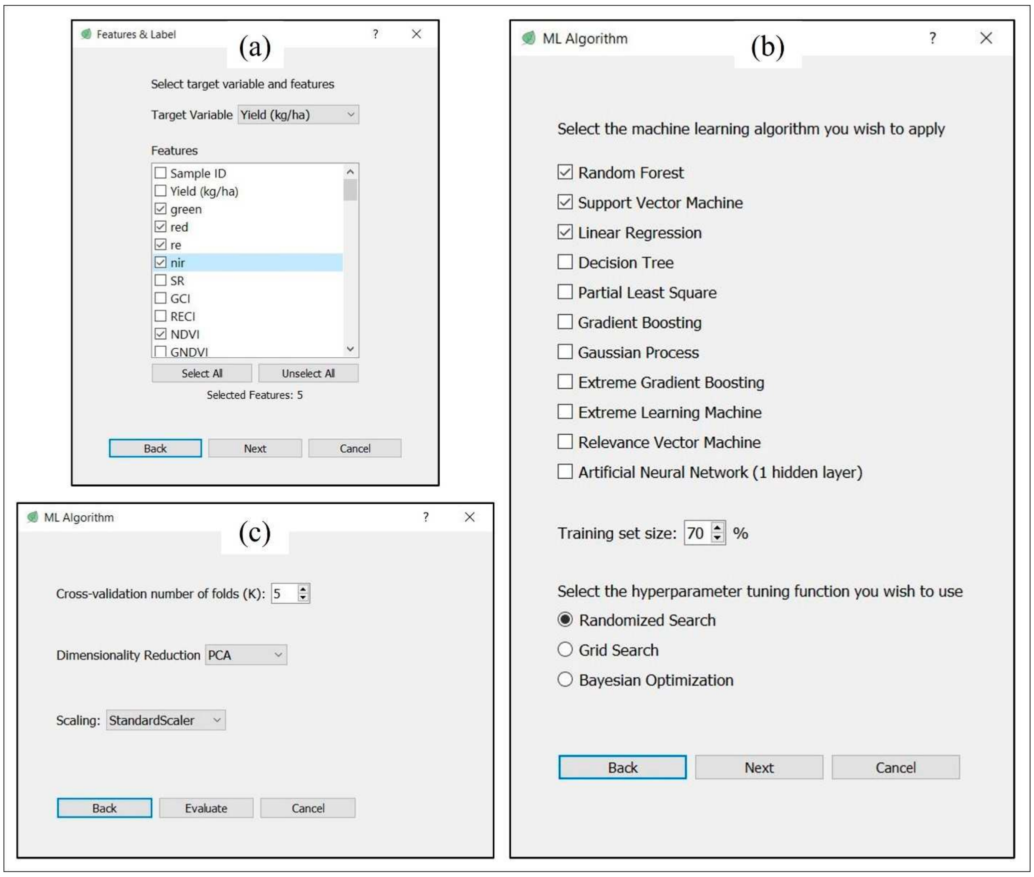
The user first should upload the dataset on which feature-based regression will be applied to predict a certain target variable. The dataset can be in the form of any spreadsheet file (i.e., CSV and Excel files) that has a header and numeric values (i.e., the features and target variable), as indicated in the previous section. The user may browse through their files and folders and pick the dataset. Only once a spreadsheet has been loaded will it be possible to move on to the next phase.

#### 2.4.2. Feature and Label Configuration

Following the selection of a dataset, the user must indicate which features will be utilized for training and what is the target variable to predict. This is made possible by the form shown in Figure 5a, which allows users to pick one target variable from a dropdown list and multiple features from a checkbox list from the spreadsheet's fields. The user is allowed to move to the following step only if the target variable and at least one feature are selected and if the target variable is not one of the features selected. The platform relies on pandas 1.4.2 to load the spreadsheet file and obtain the column names.

#### 2.4.3. Selection of Machine Learning Algorithms and Hyperparameter Tuning

Hyperfilis 1.0 supports up to eleven ML regression models (Table 2) that can be selected from a checkbox list. Except for Extreme Gradient Boosting, Extreme Learning Machine, and Relevance Vector Machine, which were developed using third-party libraries, most of the ML models described above were implemented using scikit-learn 0.24.2. These tools can also be used for classification analyses.



(a) Features & Label

Select target variable and features

Target Variable: Yield (kg/ha)

Features

- [ ] Sample ID
- [ ] Yield (kg/ha)
- [✓] green
- [✓] red
- [✓] re
- [✓] nir
- [ ] SR
- [ ] GCI
- [ ] RECI
- [✓] NDVI
- [ ] GNDVI

Select All | Unselect All

Selected Features: 5

Back | Next | Cancel

(b) ML Algorithm

Select the machine learning algorithm you wish to apply

- [✓] Random Forest
- [✓] Support Vector Machine
- [✓] Linear Regression
- [ ] Decision Tree
- [ ] Partial Least Square
- [ ] Gradient Boosting
- [ ] Gaussian Process
- [ ] Extreme Gradient Boosting
- [ ] Extreme Learning Machine
- [ ] Relevance Vector Machine
- [ ] Artificial Neural Network (1 hidden layer)

Training set size: 70 %

Select the hyperparameter tuning function you wish to use

● Randomized Search
○ Grid Search
○ Bayesian Optimization

Back | Next | Cancel

(c) ML Algorithm

Cross-validation number of folds (K): 5

Dimensionality Reduction: PCA

Scaling: StandardScaler

Back | Evaluate | Cancel

**Figure 5.** Feature-based modeling forms. (**a**) After reading the input dataset, the platform allows the users to specify the features and the target variable to be used for model training. (**b**) Hyperfidelis can run up to eleven ML regression models, with hyperparameters fine-tuned using Randomized Search, Grid Search, or Bayesian Optimization. (**c**) As a last step before training, the user can choose to perform cross-validation, dimensionality reduction, and feature scaling.

**Table 2.** Machine learning regression models.

| ML Model | Description |
|---|---|
| Random Forest | Regression learning algorithm that employs ensemble learning. |
| Decision Tree | Supervised learning method that uses a tree-like structure to make predictions. The tree splits the data based on features (branches) and leads to a final prediction (leaf). |
| Support Vector Machine | Method that aims to find a hyperplane in an N-dimensional space that categorizes data points. |
| Linear Regression | Model in which the input variables and the single-output variable are assumed to have a linear relationship. |
| Partial Least Squares | Strategy for reducing predictors to a smaller collection of uncorrelated components and performing least squares regression on these components rather than the original data. |
| Gradient Boosting | Strategy for turning weak learners into strong learners by gradually, additively, and sequentially training different models. |

**Table 2.** *Cont.*

| ML Model | Description |
|---|---|
| Gaussian Process | Probabilistic model capable of making predictions based on previous information (kernels) and providing uncertainty estimates for those predictions. |
| Extreme Gradient Boosting | Library based on gradient boosting that aims to push machines' computing limitations to their limits in order to produce a scalable, portable, and accurate library. |
| Extreme Learning Machine | Learning method for single hidden layer feedforward neural networks that solves the slow training speed and overfitting issues of standard neural network learning algorithms. |
| Relevance Vector Machine | Regression and classification method that employs Bayesian inference. |
| Artificial Neural Network | Like the human brain, artificial neural networks include neurons linked to one another in multiple levels. Nodes is the term for these neurons. |
| Deep Neural Network | A type of artificial neural network (ANN) with multiple hidden layers between the input and output layers. |

Once the ML models to be evaluated have been selected, the user may indicate how much of the dataset will be used for training and, as a result, how much will be utilized for testing. Scikit-learn 0.22.4 handles the dataset division into training and test sets.

Finally, the user must specify the hyperparameter tuning technique they want to apply to fine-tune the hyperparameters of each selected model. The choices are as follows: (1) Grid Search, an extensive search over a manually defined subset of a learning algorithm's hyperparameter space, (2) Randomized Search, which substitutes exhaustive enumeration of all combinations with a random selection of them, and (3) Bayesian Optimization, which creates a probabilistic model of the function mapping from hyperparameter values to the objective, which is then tested on a validation set.

Scikit-learn 0.22.4 is used to implement Grid Search and Randomized Search, whereas KerasTuner is used to implement Bayesian Optimization. The user can get to the next phase if the training set size is specified, at least one ML model is chosen, and a hyperparameter tuning function is selected. This step's form is shown in Figure 5b.

#### 2.4.4. Parameters Configuration

A set of parameters may be configured in this step to enhance the model's performance, minimize training time, and avoid potential overfitting. There is a risk of overfitting on the test set when investigating different hyperparameters for a model since the parameters can be changed until the model performs ideally. This allows test set information to "leak" into the model, and evaluation measures no longer report on generalization performance. To address this issue, another portion of the dataset can be set aside as a "validation set": training takes place on the training set, followed by evaluation on the validation set, and then, when the experiment appears to be successful, final evaluation on the test set. However, by dividing the available data into three sets, we dramatically restrict the number of samples that can be used to train the model, and the results can be influenced by a random selection of the (train, validation) sets.

Cross-validation (CV) is used to overcome this problem: a test set should still be kept for final evaluation, but the validation set is no longer required when using CV. The training set is divided into k smaller sets in the basic technique, known as k-fold CV. For each of the k "folds", the process is as follows: by utilizing $k - 1$ of the folds as training data, a model is created, and the model is then verified using the remaining data. The average of the values computed in the loop becomes the performance metric supplied by k-fold cross-validation. Cross-validation is used by the platform during training for the reasons stated above, and the number of folds (k) may be customized by the user at this stage.

Because some machine learning algorithms are sensitive to feature scaling while others are nearly invariant to it, feature scaling may be necessary when dealing with big datasets containing features of varying scales. Machine learning techniques that employ gradient descent as an optimization strategy, such as linear regression, logistic regression, neural networks, and others, require data to be scaled. The range of features has the greatest impact on distance algorithms like KNN, K-means, and SVM. This is because they analyze distances between data points to estimate their similarity behind the scenes. Tree-based algorithms, on the other hand, are mostly unaffected by feature scaling.

As a result, the platform provides the user with the option of applying feature scaling to the dataset. The user can choose from three alternatives here: (1) no feature scaling, (2) min–max scaling (normalization), a scaling technique for shifting and rescaling values so that they end up ranging between 0 and 1, and (3) standard scaling (standardization), a method of scaling in which the values are centered around the mean and the standard deviation is one. This indicates that the attribute's mean becomes zero, and the resulting distribution has a standard deviation of one unit.

When the data distribution does not follow a Gaussian distribution, normalization is a good option. This is important in algorithms like K-Nearest Neighbors and Neural Networks, which do not presume any data distribution. In circumstances when the data follow a Gaussian distribution, on the other hand, standardization can be beneficial. This, however, does not have to be the case. Standardization, unlike normalization, does not have a boundary range. As a result, even if the data contain outliers, normalization will have no effect on them.

Dimensionality reduction is another feature engineering/preprocessing approach that comes into play when dealing with large datasets, especially datasets with numerous features. When used, dimensionality reduction (1) reduces the amount of space required to store data as the number of dimensions decreases, (2) reduces computation/training time as the number of dimensions reduces, (3) aids in avoiding overfitting on algorithms that perform poorly on large dimensional datasets, and (4) addresses multicollinearity by removing redundant features.

The user can choose whether or not to reduce the dimensionality of the provided dataset here. They can choose from the following options: (1) no dimensionality reduction, (2) PCA (Principal Component Analysis), a feature extraction approach that combines the input variables in a specified way, allowing the "least important" variables to be deleted and the most useful parts of all the variables to be retained (also, following PCA, all of the "new" variables are independent of one another), and (3) KernelPCA, a PCA extension that employs kernel approaches techniques. The initially linear PCA processes are conducted in a reproducing kernel Hilbert space using a kernel, SVD (Singular Value Decomposition) data having m-columns (features) are projected into a subspace with m or fewer columns, preserving the essence of the original data. With sparse data (i.e., data with many zero values), this method performs better.

Scikit-learn 0.22.4 is employed to implement cross-validation, feature scaling, and dimensionality reduction. The models may be trained and evaluated when the user has set the number of folds for cross-validation, feature scaling, and dimensionality reduction parameters. Figure 5c depicts the parameters configuration form.

### 2.4.5. Model Training

Model training is the most important phase in feature-based regression. The dataset is separated into training and test sets, and the necessary fields are divided into features and the target variable; if selected in previous steps, feature scaling and dimensionality reduction are performed. The platform then constructs a model for each of the chosen machine learning algorithms (e.g., Random Forest, Support Vector Machine, and Decision Tree) and trains them all with different hyperparameter combinations using the tuning technique chosen. This stage generates a list of "best" models (e.g., best Random Forest, best Support Vector Machine, and best Decision Tree), which are the models with the best combinations of hyperparameters for each algorithm. This step is implemented leveraging scikit-learn 0.22.4 pipelines, which are a handy tool for putting together many steps that can be cross-validated while adjusting parameters.

### 2.4.6. Deep Neural Network

Hyperfidelis also makes available a deep neural network (DNN) architecture with six hidden layers and an increasing number of neurons (from 64 to 1024) to perform regression. Batch normalization and dropout are applied after the first three layers and before the output layer. Dropout is a regularization technique for reducing overfitting in artificial neural networks by preventing complex co-adaptations on training data; batch normalization is a method for making artificial neural networks faster and more stable by normalizing the layers' inputs by re-centering and re-scaling.

The *Input Data Loading* and *Features and Label Configuration* steps are unchanged when utilizing the DNN; however, the *Selection of Machine Learning Algorithms and Hyperparameter Tuning* step is removed. Similar to ML models, the user may define the training set size and whether or not to use feature scaling in the *Parameters Configuration* stage. When it comes to neural networks, feature scaling is a step that may drastically improve the network's performance. On the other hand, reducing the dimensionality of the data to be used as an input to a deep neural network would somewhat defeat the purpose of using a deep neural network to begin with, because the network would have to decompress the data only to try to recombine them once more internally to reach its full predictive capacity; for this reason, the dimensionality reduction option is not available when using DNN.

The learning rate, the batch size, and the number of epochs are the most significant parameters to test when training a DNN. The learning rate controls how much the model changes in response to the estimated error each time the model weights are updated; the batch size defines how many samples are processed per iteration before the model is updated; the number of epochs determines how many times the learning algorithm runs through the entire training dataset. In this stage, the user can try a combination of different learning rates, batch sizes, and epoch numbers on the input dataset. Keras 2.8.0 was used to implement the DNN architecture.

### 2.4.7. Performance Evaluation

The evaluation of the models' performance is the final stage in the pipeline. The platform creates a list of the top models after the *Model Training* phase (one for each algorithm selected in the *Selection of Machine Algorithm* step). For each "best model", metrics like $R^2$, RMSE (Root Mean Squared Error), and RRMSE (Relative Root Mean Square Error) are calculated, and each model's performance is represented in the form of metrics ($R^2$, RMSE, RRMSE) and scatter plots comparing actual target variable values against predicted values by the model. The quality of fit of a regression model is represented by $R^2$, a statistical metric. The optimum $R^2$ value is 1; the closer the $R^2$ value is to 1, the better the model fits. The root mean square error (RMSE) is one of the most often used methods for measuring prediction quality; it illustrates how much predictions deviate from the observed true values using Euclidean distance. In the scatter plots, the actual values are plotted on the $X$-axis, while the predicted values are plotted on the $Y$-axis. The plots also display a line that shows a perfect prediction, in which the predicted value matches the actual value perfectly. The distance between a point and this ideal 45° angle line represents how well or poorly the prediction has performed. Those performances are provided to the user so that they can examine the differences between models and then autonomously decide which model to export.

The color of the scatter plots can be changed, and each model can be exported. If the user chooses to store a model for later usage, the platform will handle it for them and export it as a pickle file. When applying a previously saved model to a new dataset, it is critical that any data preparation (e.g., feature scaling, dimensionality reduction, etc.) performed on the dataset used to train the model is also performed on the new dataset. Therefore, when exporting a model, the platform will also export the data preparation objects applied on the training dataset, which will then be applied to the new dataset when the model will be loaded.

When a model is exported, a PDF report with a summary of the evaluation is generated and saved together with the model. The following information is included in the report: the best model's name, target variable, features, number of samples, scatter plots with actual and predicted values, evaluation metrics ($R^2$, RMSE, RRMSE), hyperparameters of the best model, feature importance, tuning method utilized, number of folds used in cross-validation, dimensionality reduction technique used, feature scaling method, and training set size.

Scikit-learn 0.22.4 is used to calculate the metrics, matplotlib 3.3.4 [63] is employed to display the scatter plots, and the report is written in HTML and then converted to PDF using xhtml2pdf 0.2.8 [64].

### *2.5. Plot-Level Prediction Pipeline*

One of Hyperfidelis' novel features is the ability to apply a machine learning model to every plot in crop imagery, allowing users to map plant traits (e.g., chlorophyll, nitrogen, protein content, yield) over an entire field. The goal is to give users a tool that will help them through the process of creating a training dataset for a model that will be used to predict plant characteristics in a field. To perform plot-level mapping, Hyperfidelis provides a pipeline that involves training dataset generation, model training, model application, and mosaicking.

#### 2.5.1. Training Dataset Generation

The initial stage in the pipeline is to create a training dataset from hyperspectral imagery and a plot boundary file with ground truth data (i.e., the plant trait that the user wishes to map to the whole field). This phase will result in a dataset comprising numerous vegetation indices generated by the platform for each plot in the plot boundary file, as well as the ground truth data for each plot in the input boundary file. The training dataset's features will be represented by the vegetation indices, while the ground truth data will be the target variable (i.e., the plant trait that the user wants to predict for the entire field) as depicted in Figure 6a.

Once the user has loaded the raster image and opted to extract plot-level spectral features (i.e., vegetation indices), the platform will request the user to input the plot boundary shapefile containing the ground truth data (e.g., a shapefile containing the protein content of some plots in the field). The user will then be asked to specify which field in the shapefile represents the plot ID, as well as whatever additional fields from the shapefile they want to save in the training dataset (i.e., the target variable). They will also be asked to choose the vegetation indices they want to calculate for each plot.

After that, the platform will clip each plot using the boundary shapefile, keeping just vegetation pixels and setting the other pixels to zero for each clipped plot. To do so, the platform extracts the plot's near-infrared (NIR) band first, because healthy plants exhibit high reflectance in the NIR band, and NIR data may also be used to distinguish different types of rock and soil [65]. Then, using KMeans clustering, two clusters (soil and vegetation) are created, with soil pixels set to 0 and vegetation pixels set to 1, resulting in a vegetation mask. Because the soil pixels in the mask are set to 0 and the vegetation pixels are set to 1, by multiplying a plot raster image with the corresponding mask, the output image will maintain the original vegetation pixels, and the soil pixels will be set to zero.

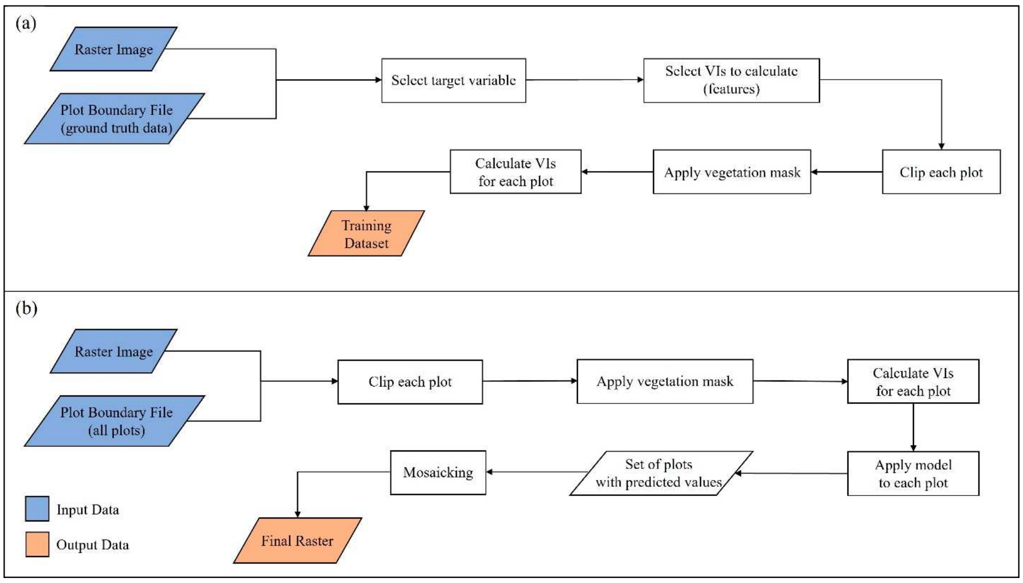


**Figure 6.** (**a**) Training dataset generation workflow. Given a raster image and a plot boundary shapefile containing ground truth data, the platform will generate a dataset that can be used to train a model. (**b**) Model application and mosaicking workflow. In this stage, Hyperfidelis applies the model exported from the model training phase to all plots in the field. By mosaicking the output plots predicted by the model, a plot-level map is generated.

For studying crop residue or other scenarios where RGB data are utilized, alternative methods may be necessary for separating vegetation from non-vegetation areas. These methods could include thresholding techniques based on color intensity or texture analysis to identify areas likely to contain vegetation. Additionally, machine learning algorithms trained on labeled RGB images could be used to classify pixels as vegetation or non-vegetation based on learned features. RGB data lack the NIR band, which is crucial for distinguishing healthy vegetation from soil. Therefore, using KMeans clustering directly on RGB data to create a vegetation/soil mask may not yield accurate results.

The selected vegetation indices are computed for each plot once the vegetation pixels have been extracted, and the training dataset is generated and saved as a CSV file. The following fields will be included in the final dataset: plot ID of each plot, vegetation indices (e.g., NDVI, LAI, etc.) for each plot (i.e., features), ground truth data (e.g., protein content) for each plot (i.e., target variable).

### 2.5.2. Model Training

After the training dataset has been created, the user will be able to utilize Hyperfidelis' feature-based modeling pipeline presented in the previous section to train and export a model that will be used to map the plant trait throughout the whole field. The user will just need to upload the produced training dataset into the platform, and the platform will then allow the user to perform the following steps, as previously discussed: indicate features and the target variable from the dataset, apply multiple ML algorithms to the dataset, perform normalization and dimensionality reduction, if necessary, fine-tune each ML model and evaluate its performance, and export the best model.

As a result, after this phase, a model capable of predicting the target variable in the training dataset generated in the previous step will be exported.

### 2.5.3. Model Application

This step is quite similar to the Training Dataset Generation discussed previously, i.e., the user is prompted to provide a raster image and a plot boundary file here as well. However, in this case, the plot boundary shapefile must contain all of the field's plots (or all of the plots from which the user wishes to predict a plant characteristic), and no ground truth data is necessary. Also, because the platform will automatically extract that information from the metadata associated with the exported model, the user will not be required to specify the vegetation indices to be computed for each plot. The platform will clip each plot, create a vegetation mask on the NIR band using KMeans clustering, then apply the mask to the plot to extract vegetation pixels and set soil pixels to 0. Following that, each plot's vegetation indices are computed.

At this point, the user will be asked to select the model that was exported in the previous step, which will be applied to each clipped plot. The model will be fed the plot's vegetation indices (the same ones used for training) for each plot, and it will predict the target variable (i.e., plant trait) using those features. The model generates a continuous variable for each plot; by multiplying this value with all the pixels in the vegetation mask, the platform can recreate the image of the plot, with each vegetation pixel set to the predicted value of the plant trait (e.g., the plot's mean protein content) and each soil pixel set to zero. At the end of this stage, the platform will predict the plant trait for each plot in the boundary shapefile and export each plot with the predicted values as a GeoTIFF file (see Figure 6b).

#### 2.5.4. Mosaicking

The pipeline's final phase involves mosaicking all the plots exported in the previous stage to create the final raster. When visualized in any GIS application (e.g., ArcGIS and QGIS), the final raster will be composed of all the original plots, each represented with a single value (the projected plant trait for the plot) on the vegetation pixels and soil pixels set to zero, allowing for easy identification and analysis of variations across plots.

## 3. Hyperfidelis Applications and Case Studies

In this section, examples of applications are presented to demonstrate Hyperfidelis' potential. The first example shows how Hyperfidelis can be used to extract plot boundaries from a sorghum field and calculate zonal statistics on each extracted plot, the second example illustrates the use of Hyperfidelis for soybean yield prediction, and the third example represents a use case for plot-level chlorophyll content estimation over an entire cornfield. Each application makes use of different combinations of Hyperfidelis tools.

### *3.1. Plot Boundary Extraction and Zonal Statistics*

Thousands of plants are planted according to experimental designs in field studies to examine genotypes or management approaches, and smaller groups of plants known as "plots" are inspected for numerous forms of phenotyping analysis. Hundreds to thousands of plots might be found in a field trial for plant-breeding research. The evaluation of phenotypes on a plot scale necessitates the examination of responses within individual plots, which must be extracted, but due to the large number of plots in a field, manual plot extraction is often not feasible [66].

The plot extraction feature in Hyperfidelis aims to extract plots from UAV orthomosaic imagery from an early growth stage showing clear plot boundary. We tested this feature on several images. In this section, we present a use case scenario that we performed using a UAV-based multispectral image of a sorghum field collected by our team in Maricopa, Arizona (Figure 7a).

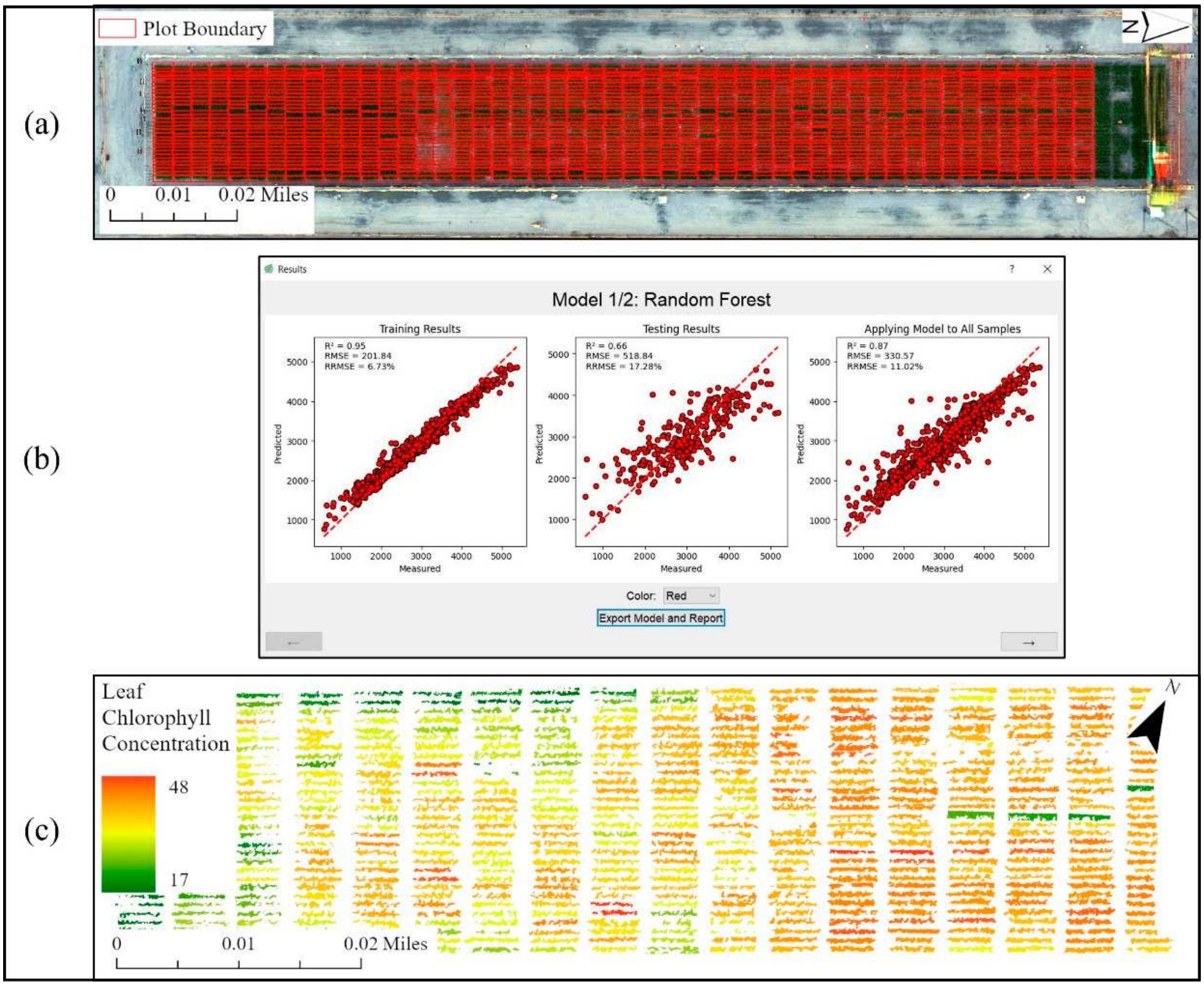


**Figure 7.** (**a**) Single-row plot boundary shapefile generated by the platform and overlayed on the original image: 95% of the plots have been correctly detected. (**b**) Scattr plots of measured and predicted crop yield with Random Forest regression in which the model was trained and tested with canopy spectral, structural, thermal, and textural information as features and yield data as the target variable. (**c**) Shows a color coded map of leaf chlorophyll concentration (LCC) per plot that was generated by Hyperfidelis using the Random Forest regression pipeline. In (**c**), red shows high LCC values and green represesnts lower values.

The purpose was to extract plot boundaries from the field and calculate vegetation indices for each plot so that vegetation health and plant performance under environmental stress could be evaluated for each plot. We loaded the imagery into Hyperfidelis, indicating the number of rows (32) and columns (52) that make up the area of interest, as well as the number of rows per plot (1). The platform detected 95% of single plot boundaries and exported them as a plot boundary shapefile (see Figure 7a). Then the vegetation indices for each plot were extracted as the plot mean (zonal statistics), resulting in a dataset of single-row plots with values of vegetation indices. By analyzing the resulting datasets, it is possible to evaluate vegetation cover, vigor, and the growth of each plot quantitatively and qualitatively [44].

### *3.2. Yield Prediction*

For grain policy and food security, accurate and non-destructive crop yield predictions from early season data is imperative [67,68], especially in the future climate characterized by increasing frequency of extreme weather patters [69,70]. Early assessment of yield at field/farm-scale plays a significant role in crop breeding and management [71–73]. Early-season yield prediction can help increase agricultural productivity and profit while lowering input costs [74–76]. Furthermore, high-accuracy non-destructive crop yield predictions would allow quick and efficient identification of high-yielding genotypes from a wide number of potential genotypes [77,78].

We used Hyperfidelis' feature-based modeling tool to train and compare different machine learning models for predicting yield. We used two datasets containing soybean fields located in North and South Americas (Figure 8). The first dataset was collected at an experimental site located at the University of Missouri Bradford Research Center (see Figure 8(b1)) that includes yield and canopy spectral, structural, thermal, and textural data taken from the field and produced in our previous work [13]. The second dataset consists of multiple fields located in Buenos Aires, Argentina (see Figure 8(b2–b5)). For details on the experimental sites and design, refer to references [13,15].

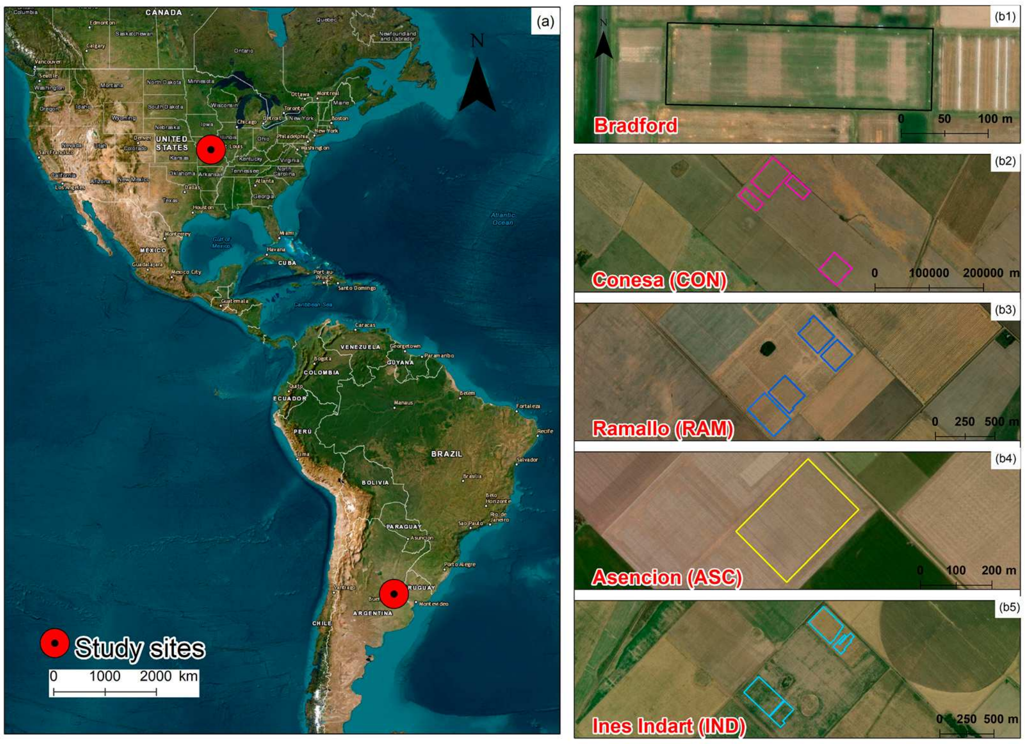


**Figure 8.** (**a**) Map showing test sights we gathered our datasets from. (**b1**) University of Missouri Bradford Research Center field. (**b2**) Conesa field in Argentina dataset. (**b3**) Ramallo field in Argentina dataset. (**b4**) Asencion field in Argentina dataset. (**b5**) Ines Indart field in Argentina dataset.

For the Bradford dataset, the raw bands (green, red, red-edge, and near-infrared) from multispectral orthomosaics of the field, along with a collection of 17 vegetation indices (VI), were utilized as canopy spectral features. Canopy height was combined with vegetation fraction, which is the percentage of plant area per ground surface area, as a canopy structure feature. From UAV thermal-infrared (TIR) data, the normalized relative canopy temperature (NRCT) [79] was computed and utilized as a thermal feature. Multispectral bands including green, red, red-edge, and NIR wavelengths, RGB-based canopy height, and TIR-based NRCT were used to extract texture features. Mean, variance, homogeneity, contrast, dissimilarity, entropy, second moment, and correlation are among the GLCM-based texture measurements.

The Argentinian dataset, on the other hand, contains multispectral UAV imagery with blue, green, red, red edge, and near-infrared. 38 commonly used VIs were extracted from the UAV multispectral images. The VIs were extracted from nine different flight dates to add a temporal resolution to the dataset.

We used the feature-based modeling pipeline outlined in the previous section to indicate the yield as the target variable and all the other information in the datasets as the features used for training. We then applied all of the ML algorithms available in Hyperfidelis to the input datasets (i.e., Random Forest, Support Vector Machine, Linear Regression, Decision Tree, Partial Least Square, Gradient Boosting, Gaussian Process, Extreme Gradient Boosting, Extreme Learning Machine, Relevance Vector Machine, and single-layer Artificial Neural Network (ANN)). We set the training set size to 70% of the dataset resulting in 30% of the dataset being used for testing, and chose Grid Search as the hyperparameter tuning function. For cross-validation, we chose 10 as the number of folds (k), StandardScaler was chosen as the feature scaling function, and no dimensionality reduction was applied.

At the end of the processing, the platform displays the results of all ML models in scatter plots in various layouts. For example, Figure 9 shows Hyperfidelis modeling results ($R^2$ value of 0.68) with the Random Forest Regression for Argentinian dataset; $R^2$ value of 0.62 was achieved with single-layer ANN (Figure 10) for the same dataset. A user then can export the best performing model (i.e., the Random Forest model) with associated accuracy report and hyperparameters. It is worth noting that ANN can be customized to include multiple layers which may improve the prediction power of the algorithm.

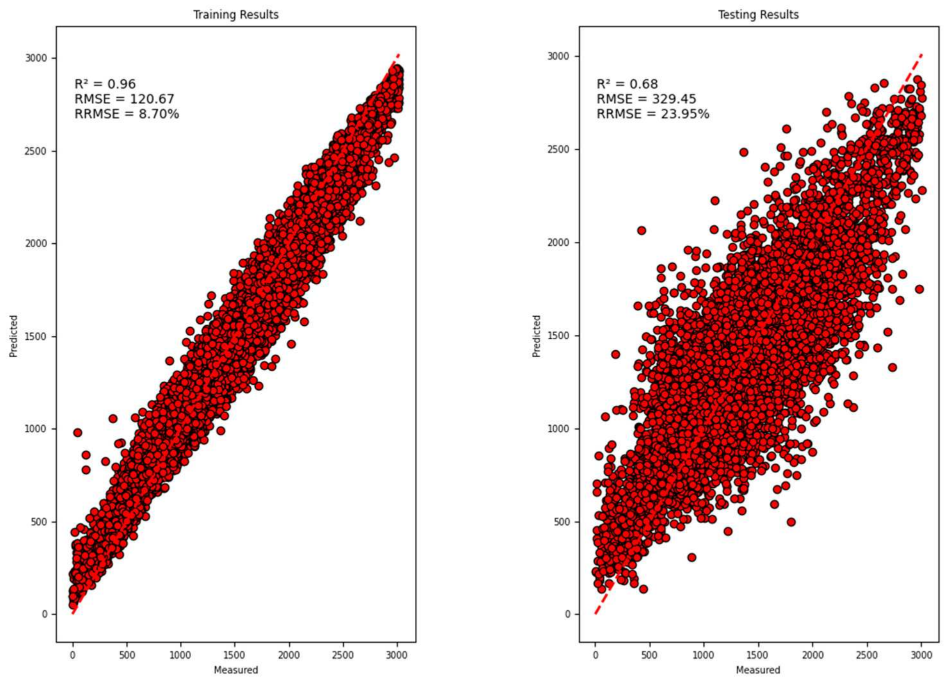


**Figure 9.** Performance evaluation produced by Hyperfidelis after training Random Forest on the Argentina dataset, we achieved an R-square of 0.68.

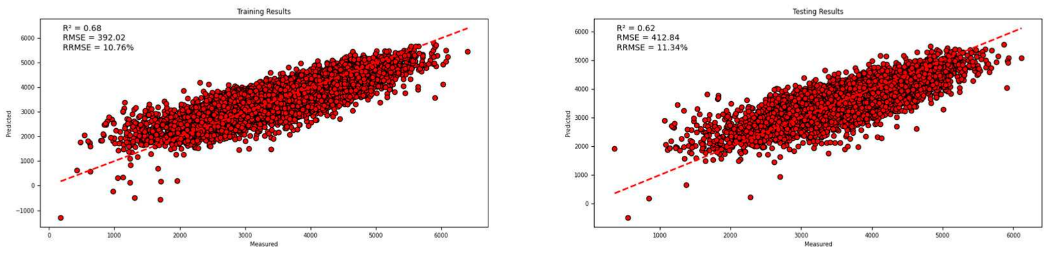


**Figure 10.** Performance evaluation produced by Hyperfidelis after training the single-layer artificial neural network model on the Argentina dataset (R-square of 0.62).

## *3.3. Plot-Level Leaf Chlorophyll Concentration Estimation*

One of the key leaf biochemical characteristics frequently examined in crop phenotyping is leaf chlorophyll concentration (LCC). In precision agriculture, LCC is used to indicate crop growth condition, health, productivity, and nutrient deficiency in addition to genomics-assisted breeding [80,81]. Although laboratory-based chemical analysis of LCC is accurate, it is damaging, labor-intensive, and impractical for large-scale fields [82]. As a result, in plant genetics, physiology, and breeding applications, predicting leaf biochemical attributes non-destructively and rapidly is a primary goal. Hyperfidelis was used to estimate plot-average LCC for an experimental cornfield at Planthaven Farms in O'Fallon, Missouri (Figure 7c). To accomplish this, we fed to the platform a hyperspectral image of the field and a shapefile containing the ground truth LCC data of some of the plots in the field. We specified which attributes in the shapefile reflected the plot ID and the ground truth data of interest (i.e., LCC obtained using Dualex 4 Scientific handheld sensor) when requested, as well as the vegetation indices that the platform would compute for each plot. The platform generated a training dataset that included vegetation indices and ground truth LCC for each plot.

The resulting training dataset was then employed to train several machine learning models using Hyperfidelis' feature-based modeling tool. The vegetation indices are the features needed to train the models, and the LCC is the target variable that the algorithms will be able to predict. We applied all of the ML models available on the platform, utilizing 70% of the dataset for training and 30% for testing and tweaking each model's hyperparameters using Grid Search. We set 10 as the number of folds (k) for cross-validation, and because the features (i.e., the vegetation indices) had varied scales, we used StandardScaler to normalize them. Once the model training phase was completed, we examined each model's performance and chose the best one, which in this case was the Support Vector Machine.

Since the exported model could reliably predict the mean LCC of a plot given a set of vegetation indices (the same used for training), we needed to calculate those vegetation indices for each plot in the study area so that we could feed the model with those data. To do so, we provided the platform with the same hyperspectral image of the field that we had previously used as well as a shapefile containing all of the field's plots. The file in this case did not contain any ground truth data, simply plot boundaries. We also loaded the model, which contained metadata that the platform utilized to determine which vegetation indices should be calculated on each plot (the same used for training the model).

Hyperfidelis computed vegetation indices for each plot and used these as input to the model to estimate the mean LCC of each plot. In this phase, soil pixels were to 0 and vegetation pixels to the mean LCC predicted value, resulting in a collection of raster images of each plot in the field. The last step was the mosaicking of all the raster images (plots), resulting in an LCC plot-level map depicted in Figure 7c for the entire specified field. An indication of the health, vitality, physiological state, production, and nutritional deficits of each plot in the field may all be determined by examining the LCC plot-level map.

## 4. Discussions

### *4.1. State of the Art*

As mentioned in Section 1, the only works that we have found that can be compared to Hyperfidelis are FIELDimageR, PlantCV, and Polly. FIELDimageR is a package that analyzes orthomosaic imagery with a large number of plots. Cropping and rotating images, as well as calculating the number of plants per plot, canopy cover percentage, vegetation indices, and plant height are the functions supported. While FIELDimageR provides geospatial visualization and plot-level feature extraction capabilities, it lacks data engineering and machine learning methods, making it unable to model agricultural features; it also misses a GUI, necessitating R programming proficiency. PlantCV is a community-developed image analysis software package aimed at plant phenotyping, and it is made up of a set of modular and reusable tools for plant image analysis. PlantCV, like FIELDimageR, lacks a GUI, resulting in a plethora of functions for plant image processing and analysis that only Python developers will be able to add to their code as desired after thorough examination and comprehension of the package documentation. PlantCV's most recent version [83] introduced some machine learning capabilities (such as classification), although they are solely for image segmentation and only rely on one machine learning model (the naive Bayes classifier). It is also worth noting that PlantCV does not support georeferenced images. Polly is a web-based data analysis and modeling application that allows users to integrate and analyze data, as well as prototype and test their methodologies. Polly enables users to upload or connect to structured data sources, load data into the Polly system, and perform a variety of data processing operations (e.g., data cleaning, data pre-processing, attribute encoding, regression, and classification analysis). Polly's GUI is intuitive to use, and it may also be utilized as an educational tool. However, its capabilities are restricted; in fact, it can only be used for data analysis and integration, and tabular data (e.g., CSV, Excel files) are the only input type supported. Thus, despite the fact that it incorporates machine learning functionalities, users will have to rely on other tools to generate training datasets and visualize, interpret, and analyze imagery.

### *4.2. Innovation*

We developed a geospatial stand-alone software for improving crop productivity that provides a comprehensive workflow involving imagery visualization, feature extraction, and modeling of key agricultural traits in a ML framework. The software includes a modern user interface that eliminates the need for any coding knowledge, bridging the gap between plant science and advanced data science. Hyperfidelis provides a variety of data engineering and machine learning algorithms that can be used without scripting, which will prove essential in the plant science community; in crop breeding operations involving UAVs and remote sensing sensors, Hyperfidelis will improve plant phenotyping efficiency and accuracy significantly. We used exclusively open-source libraries and did not employ any third-party software or tools, resulting in code freedom and flexibility, license independence, and cost-effectiveness. We tested our platform through several simulations and real-world scenarios, and we were able to demonstrate that by using Hyperfidelis, farmers can obtain accurate information about the health, vigor, and growth of their crop fields, allowing them to proactively take action to boost agricultural productivity.

### *4.3. Limitations*

The software's backend has been implemented in Python, while the frontend has been built using PyQt5 5.15.2. Tkinter [84] and Flask [85] were the other options we considered while deciding between several frontend solutions. While Tkinter is recognized for being lightweight and simple to use and is included in the Python standard library, the style it provides is somewhat obsolete, with a restricted amount and types of widgets. Flask, on the other hand, enables the development of Python web apps in HTML/CSS/JavaScript, resulting in a more modern user interface. PyQt5 5.15.2 was chosen because it does not require the creation of HTML/CSS/JavaScript web pages, allowing for quicker development.

In terms of the backend, Python is versatile, easy to use, and rapid to develop, and it comes with a large number of libraries for managing geospatial data (e.g., rasterio 1.2.10, Spectral Python 0.22.4, GDAL 3.4.3, etc.) and performing machine learning (e.g., scikit-learn 0.22.4, Keras 2.8.0, etc.); however, it has some intrinsic limitations, such as memory consumption and speed. Python's memory usage is high due to the flexibility of the data types, and it is not well suited for memory heavy tasks; also, because Python is an interpreted language, it is slower than C++ or Java [86], and this was the main bottleneck in our platform. During our experiments, the processing time for 1000 × 1000 images reached peaks of 10 min when dealing with computationally intensive operations like grey-level co-occurrence matrix creation. Furthermore, Python does not provide many 3D rendering libraries, and the ones we explored only supported a restricted number of file formats and had extremely slow processing times (more than 10 min to load LiDAR data); consequently, we were unable to handle and visualize LiDAR data in the program. Finally, it is worth noting that when using Hyperfidelis ML features, particularly when working with DNN, the input dataset size plays an important role in the training of the model; if the user provides a dataset with a limited number of observations, that will represent a bottleneck for the model's accuracy.

### *4.4. Future Work*

Deep learning, specifically convolutional neural networks (CNNs), has demonstrated its effectiveness in a variety of remote sensing-related challenges [87–89]. Hyperfidelis' ML capabilities will be expanded with the addition of CNN imagery-based, end-to-end modeling. The key benefit this will bring over to feature-based modeling is that no vegetation indices will be calculated, and no manual tuning or feature engineering will be required [14]. Another upgrade that might be made is to increase the platform's level of automation, which would decrease the need for user interaction to the bare minimum, thereby turning it into an autonomous machine learning system [90]. This might be accomplished, for example, by incorporating a workflow management system into the platform, that automates and streamlines many elements of ML and geospatial work. Further Hyperfidelis development research can focus on how to support LiDAR data visualization, management, and analysis. Future efforts could also integrate Hyperfidelis with on-demand cloud computing resources (e.g., cloud CPUs and cloud GPUs) to overcome Python's performance constraints when dealing with computationally heavy and highly parallel computing activities. Relying on the cloud would ensure location and hardware independence allowing farmers to utilize Hyperfidelis through low-performance portable devices such as tablets. Finally, we plan to explore more Hyperfidelis use cases such as disease detection, seed composition estimation, and nitrogen uptake and use efficiency.

The software currently supports independently exporting the data or results at every step of data processing to various formats. However, it does not support spectral data processing or importing machine learning models or algorithms into the software framework. Researchers, however, can create customized neural networks by adjusting the number of neurons, layers, learning rate, etc. Hyperfidelis does not support LiDAR point clouds in its current release but can ingest LiDAR-derived features in downstream modeling. Spectral data processing, LiDAR data processing, and importing external machine learning models to run within the software architecture is currently being developed and will be available in the next release along with functionalities to support a suite of CNN approaches.

## **5. Conclusions**

The main goal of this work was to create a tool that would allow farmers, data scientists, and plant scientists to extract valuable information from geospatial big data in order to improve crop productivity and ensure that food production keeps up with population growth, ensuring food security. We built a GeoAI software that enables the users implementing machine learning algorithms without having any coding knowledge, bridging the gap between plant science and advanced data science. Example applications demonstrated Hyperfidelis' capabilities, including automated plot boundary extraction, crop yield prediction, and leaf chlorophyll concentration estimation. Each application makes use of a different set of Hyperfidelis tools. Using the software presented in this work, farmers and practitioners will be able to take proactive action to improve crop production, potentially reducing global food insecurity. The main contributions of this work are as follows:

- The geospatial stand-alone software developed provides a workflow for imagery visualization, feature extraction, and modeling of key agricultural traits in a ML framework;
- The graphical user interface eliminates the need for users to code, bridging the gap between plant science, agronomy, and advanced data science;
- The wide range of state-of-the-art data engineering and machine learning algorithms implemented can be employed without scripting;
- The exclusive use of open-source libraries results in code freedom and flexibility, license independence, and cost-effectiveness.

**Author Contributions:** Conceptualization, V.S. and R.C.; methodology, R.C., S.B. and O.A.A.; software, R.C.; validation, R.C., S.B., A.G. and H.A.; formal analysis, R.C.; investigation, R.C.; resources, V.S.; data curation, R.C., S.B., H.A., A.G., O.A.A. and V.S.; writing—original draft preparation, R.C.; writing—review and editing, V.S., R.C., S.B., H.A., A.G., O.A.A. and F.E.; visualization, R.C., H.A. and O.A.A.; supervision, V.S.; project administration, V.S.; funding acquisition, V.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the United Soybean Board and Foundation for Food & Agricultural Research (projects#: 2120-152-0201 and 2331-201-0103), NSF Cyber Physical Systems (CPS award # 2133407), the U.S. Geological Survey under Grant/Cooperative Agreement No. G23AP00683 (GY23-GY27), as well as Saint Louis University Research Advancement Grants.

**Data Availability Statement:** The data used in this study are not publicly available due to collaboration agreements with partner institutes.

**Acknowledgments:** The authors extend their thanks to the current and past members of the Saint Louis University Remote Sensing Lab for offering valuable input at the various stages of this work.

**Conflicts of Interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A

**Table A1.** RGB vegetation indices available in Hyperfidelis.

| RGB Vegetation Index | Formula |
|---|---|
| Normalized red (NR) | $NR = R/(R^{1} + G^{2} + B^{3})$ |
| Normalized green (NG) | $NG = G/(R + G + B)$ |
| Normalized blue (NB) | $NB = B/(R + G + B)$ |
| Excess red (ExR) | $ExR = 1.4 \times (NR - NG)$ |
| Excess green (ExG) | $ExG = 2 \times (NG - NR - NB)$ |
| Excess blue (ExB) | $ExB = 1.4 \times (NB - NG)$ |
| Excess green red (ExGR) | $ExGR = ExG - ExR$ |
| Normalized difference index (NDI) | $NDI = (NR - NG)/(NR + NG + 0.01)$ |
| Normalized green–red difference index (NGRDI) | $NGRDI = (NG - NR)/(NG + NR)$ |
| Color intensity (INT) | $INT = (R + G + B)/3$ |
| Blue–green ratio index (BGRI) | $BGRI = B/G$ |
| Red–green ratio index (RGRI) | $RGRI = R/G$ |

**Table A1.** *Cont.*

| RGB Vegetation Index | Formula |
|---|---|
| Red–blue ratio index (RBRI) | $RBRI = R/B$ |
| Green–red ratio index (GRRI) | $GRRI = G/R$ |
| Normalized red–green ratio index (NRGRI) | $NRGRI = NR/NG$ |
| Normalized green–red ratio index (NGRRI) | $NGRRI = NG/NR$ |
| Normalized blue–green ratio index (NBGRI) | $NBGRI = NB/NG$ |
| Normalized red–blue ratio index (NRGRI) | $NRGRI = NR/NB$ |
| Green–red vegetation index (GRVI) | $GRVI = (G - R)/(G + R)$ |
| Visible atmospherically resistance index (VARI) | $VARI = (G - R)/(G + R - B)$ |
| Principal component analysis index (IPCA) | $IPCA = 0.994 \times \mid R - B \mid + 0.961 \times \mid G - B \mid + 0.914 \times \mid G - R \mid$ |
| Kawashima vegetation index (IKAW) | $IKAW = (R - B)/(R + B)$ |
| Green leaf area index (GLI) | $GLI = (2 \times (G - R - B))/(2 \times (G + R + B))$ |
| Color index of vegetation (CIVE) | $CIVE = 0.441 \times R - 0.881 \times G + 0.385 \times B + 18.78745$ |
| Woebbecke index (WI) | $WI = (G - B)/(R - G)$ |
| Vegetation index (VEG) | $VEG = G/(R^{0.667} \times B^{0.333})$ |

[1] R: red band; [2] G: green band; [3] B: blue band.

**Table A2.** Multispectral vegetation indices available in Hyperfidelis.

| Multispectral Vegetation Index | Formula |
|---|---|
| Normalized difference vegetation index (NDVI) | $NDVI = (NIR\ ^{4} - R)/(NIR + R)$ |
| Green normalized difference vegetation index (GNDVI) | $GNDVI = (NIR - G)/(NIR + G)$ |
| Ratio vegetation index 1 (RVI1) | $RVI1 = NIR/R$ |
| Green chlorophyll index (GCI) | $GCI = (NIR/G) - 1$ |
| Red–green ratio vegetation index (RGVI) | $RGVI = R/G$ |
| Difference vegetation index (DVI) | $DVI = NIR - R$ |
| Soil-adjusted vegetation index (SAVI) | $SAVI = ((NIR - R)/(NIR + R + L)) \times (1 + L)$ <br> $L = 0.5$ |
| Modified soil-adjusted vegetation index (MSAVI) | $MSAVI = 1/2 \times ((2 \times NIR) + 1 - sqrt((2 \times NIR + 1)^2 - 8 \times (NIR - R)))$ |
| Optimized soil-adjusted vegetation index (OSAVI) | $OSAVI = (NIR - R)/(NIR + R + 0.16)$ |
| Renormalized difference vegetation index (RDVI) | $RDVI = sqrt((NIR - R)^2/(NIR + R))$ |
| Triangular vegetation index (TVI) | $TVI = 60 \times (NIR - BG) - 100 \times (R - G)$ |
| Transformed soil-adjusted vegetation index (TSAVI) | $TSAVI = (a \times (NIR - a \times R - b))/(a \times NIR + R - a \times b)$ <br> $a = 0.96916, b = 0.084726$ |
| Perpendicular vegetation index (PVI) | $PVI = (NIR - a \times R - b)/sqrt(1 + a^2)$ <br> $a = 0.96916, b = 0.084726$ |
| Adjusted transformed soil-adjusted vegetation index (ATSAVI) | $ATSAVI = (a \times (NIR - a \times R - b))/(a \times NIR + R - a \times b + x \times (1 + a^2))$ <br> $a = 0.96916, b = 0.084726, x = 0.08$ |
| Normalized difference water index (NDWI) | $NDWI = (G - NIR)/(G + NIR)$ |
| Simple ratio pigment index (SRPI) | $SRPI = B/R$ |

**Table A2.** *Cont.*

| Multispectral Vegetation Index | Formula |
|---|---|
| Ratio vegetation index 2 (RVI2) | $RVI2 = NIR/G$ |
| Modified chlorophyll absorption ratio index 1 (MCARI1) | $MCARI1 = (RE^{5} - R - 0.2 \times (RE - G)) \times (RE/R)$ |
| Modified chlorophyll absorption ratio index 2 (MCARI2) | $MCARI2 = 1.2 \times (2.5 \times (NIR - R) - 1.3 \times (NIR - G))$ |
| Modified chlorophyll absorption ratio index 3 (MCARI3) | $MCARI3 = 1.5 \times (2.5 \times (NIR - R) - 1.3 \times (NIR - G)) \times (2 \times NIR + 1)^2 - (6 \times NIR - 5 \times R) - 0.5$ |
| Modified triangular vegetation index 1 (MTVI1) | $MTVI1 = 1.2 \times (1.2 \times (NIR - G) - 2.5 \times (R - G))$ |
| Modified triangular vegetation index 2 (MTVI2) | $MTVI2 = 1.5 \times (1.2 \times (NIR - G) - 2.5 \times (R - G)) \times (2 \times NIR + 1)^2 - (6 \times NIR - 5 \times R) - 0.5$ |
| Normalized difference cloud index (NDCI) | $NDCI = (RE - G)/(RE + G)$ |
| Plant senescence reflectance index (PSRI) | $PSRI = (R - G)/RE$ |
| Structure insensitive pigment index (SIPI) | $SIPI = (NIR - B)/(NIR + R)$ |
| Spectral polygon vegetation index (SPVI) | $SPVI = 0.4 \times 3.7 \times (NIR - R) - 1.2 \times \mid G - R \mid$ |

[4] NIR: near infrared band; [5] RE: red edge band.

**Table A3.** Hyperspectral vegetation indices available in Hyperfidelis (Raster4ml Package- Raster4ml 0.1.0 Documentation).

| Hyperspectral Vegetation Index | Formula |
|---|---|
| ARI (Anthocyanin Reflectance Index) | $ARI = (1/R_{550}{}^{6}) - (1/R_{700})$ |
| BGI (Blue Green Pigment Index) | $BGI = R_{450}/R_{550}$ |
| BRI (Blue Red Pigment Index) | $BRI = R_{450}/R_{690}$ |
| CAI (Chlorophyll Absorption Index) | $CAI = 0.5 \times (R_{2015} + R_{2195}) - R_{2106}$ |
| CRI1 (Chlorophyll Reflection Index) | $CRI1 = (1/R_{510}) - (1/R_{550})$ |
| CRI2 | $CRI2 = (1/R_{510}) - (1/R_{700})$ |
| CSI1 | $CSI1 = R_{695}/R_{420}$ |
| CSI2 | $CSI2 = R_{695}/R_{760}$ |
| CUR (Curative Index) | $CUR = (R_{675} \times R_{550})/R_{683}{}^2$ |
| DSWI (Disease Water Stress Index) | $DSWI = (R_{802} - R_{547})/(R_{1657} + R_{682})$ |
| DSWI5 | $DSWI5 = (R_{800} - R_{550})/(R_{1660} + R_{680})$ |
| G (Green Index) | $G = R_{554}/R_{677}$ |
| GMI1 | $GMI1 = R_{750}/R_{550}$ |
| GMI2 | $GMI2 = R_{750}/R_{700}$ |
| gNDVI (Blue Normalized Difference Vegetation Index) | $gNDVI = (R_{750} - R_{550})/(R_{750} + R_{550})$ |
| hNDVI | $hNDVI = (R_{827} - R_{668})/(R_{827} + R_{668})$ |
| LAI (MSAVI) (Modified Soil Adjusted Vegetation Index) | $LAI\ (MSAVI) = 0.1663^{4.2731} \times {}^{MSAVI}$ |
| LAI (MTVI2) | $LAI\ (MTVI2) = 0.2227^{3.6566} \times {}^{MTVI2}$ |
| LAI (RDVI) | $LAI\ (RDVI) = 0.0918^{6.0002} \times {}^{RDVI}$ |
| LAI (TVI) | $LAI\ (TVI) = 0.1817^{4.1469} \times {}^{TVI}$ |

**Table A3.** *Cont.*

| Hyperspectral Vegetation Index | Formula |
|---|---|
| LCI (Leaf Chlorophyll Index) | $LCI = (R_{850} - R_{710})/(R_{850} + R_{680})$ |
| LIC1 | $LIC1 = (R_{800} - R_{680})/(R_{800} + R_{680})$ |
| LIC2 | $LIC2 = R_{440}/R_{690}$ |
| LIC3 | $LIC3 = R_{440}/R_{740}$ |
| LWVI1(Leaf Water Vegetation Index) | $LWVI1 = (R_{1094} - R_{983})/(R_{1094} + R_{983})$ |
| LWVI2 | $LWVI2 = (R_{1094} - R_{1205})/(R_{1094} + R_{1205})$ |

[6] $R_{550}$: reflectance at 550 nm.

## References

1. FAO. *FAO's Work on Agricultural Innovation: Sowing the Seeds of Transformation to Achieve the SDG's*; FAO: Rome, Italy, 2018.
2. Pretty, J.; Bharucha, Z.P. Sustainable intensification in agricultural systems. *Ann. Bot.* **2014**, *114*, 1571–1596. [CrossRef]
3. Garnett, T.; Appleby, M.C.; Balmford, A.; Bateman, I.J.; Benton, T.G.; Bloomer, P.; Burlingame, B.; Dawkins, M.; Dolan, L.; Fraser, D. Sustainable intensification in agriculture: Premises and policies. *Science* **2013**, *341*, 33–34. [CrossRef]
4. Godfray, H.C.J.; Garnett, T. Food security and sustainable intensification. *Philos. Trans. R. Soc. B Biol. Sci.* **2014**, *369*, 20120273. [CrossRef]
5. Smith, P. Delivering food security without increasing pressure on land. *Glob. Food Secur.* **2013**, *2*, 18–23. [CrossRef]
6. Fraceto, L.F.; Grillo, R.; de Medeiros, G.A.; Scognamiglio, V.; Rea, G.; Bartolucci, C. Nanotechnology in agriculture: Which innovation potential does it have? *Front. Environ. Sci.* **2016**, *4*, 186737. [CrossRef]
7. Scheben, A.; Yuan, Y.; Edwards, D. Advances in genomics for adapting crops to climate change. *Curr. Plant Biol.* **2016**, *6*, 2–10. [CrossRef]
8. MacDonald, G.K.; D'Odorico, P.; Seekell, D.A. Pathways to sustainable intensification through crop water management. *Environ. Res. Lett.* **2016**, *11*, 091001. [CrossRef]
9. Lang, T.; Barling, D. Food security and food sustainability: Reformulating the debate. *Geogr. J.* **2012**, *178*, 313–326. [CrossRef]
10. Sagan, V.; Maimaitijiang, M.; Paheding, S.; Bhadra, S.; Gosselin, N.; Burnette, M.; Demieville, J.; Hartling, S.; LeBauer, D.; Newcomb, M. Data-driven artificial intelligence for calibration of hyperspectral big data. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1–20. [CrossRef]
11. Eli-Chukwu, N.C. Applications of artificial intelligence in agriculture: A review. *Eng. Technol. Appl. Sci. Res.* **2019**, *9*, 4377–4383. [CrossRef]
12. Bhadra, S.; Sagan, V.; Sarkar, S.; Braud, M.; Mockler, T.C.; Eveland, A.L. PROSAIL-Net: A transfer learning-based dual stream neural network to estimate leaf chlorophyll and leaf angle of crops from UAV hyperspectral images. *ISPRS J. Photogramm.* **2024**, *210*, 1–24. [CrossRef]
13. Maimaitijiang, M.; Sagan, V.; Sidike, P.; Hartling, S.; Esposito, F.; Fritschi, F.B. Soybean yield prediction from UAV using multimodal data fusion and deep learning. *Remote Sens. Environ.* **2020**, *237*, 111599. [CrossRef]
14. Sagan, V.; Maimaitijiang, M.; Bhadra, S.; Maimaitiyiming, M.; Brown, D.R.; Sidike, P.; Fritschi, F.B. Field-scale crop yield prediction using multi-temporal WorldView-3 and PlanetScope satellite data and deep learning. *ISPRS J. Photogramm. Remote Sens.* **2021**, *174*, 265–281. [CrossRef]
15. Skobalski, J.; Sagan, V.; Haireti, A.; Al Akkad, O.; Lopes, F.; Grignola, F. Bridging the gap between crop breeding and GeoAI: Soybean yield prediction from multispectral UAV images with transfer learning. *ISPRS J. Photogramm.* **2024**, *210*, 260–281. [CrossRef]
16. Sarkar, S.; Sagan, V.; Bhadra, S.; Rhodes, K.; Pokharel, M.; Fritschi, F.B. Soybean seed composition prediction from standing crops using PlanetScope satellite imagery and machine learning. *ISPRS J. Photogramm.* **2023**, *204*, 257–274. [CrossRef]
17. Rapsomanikis, G. *The Economic Lives of Smallholder Farmers: An Analysis Based on Household Data from Nine Countries*; Food and Agriculture Organization of the United Nations: Rome, Italy, 2015; pp. 1–39.
18. World Bank. *ICT in Agriculture (Updated Edition): Connecting Smallholders to Knowledge, Networks, and Institutions*; The World Bank: Washington, DC, USA, 2017.
19. Sylvester, G. *Success Stories on Information and Communication Technologies for Agriculture and Rural Development*; RAP Publication: New York, NY, USA, 2015.
20. Lio, M.; Liu, M.C. ICT and agricultural productivity: Evidence from cross-country data. *Agric. Econ.* **2006**, *34*, 221–228. [CrossRef]
21. Jensen, R. The digital provide: Information (technology), market performance, and welfare in the South Indian fisheries sector. *Q. J. Econ.* **2007**, *122*, 879–924. [CrossRef]
22. Deichmann, U.; Goyal, A.; Mishra, D. Will digital technologies transform agriculture in developing countries? *Agric. Econ.* **2016**, *47*, 21–33. [CrossRef]
23. Fabregas, R.; Kremer, M.; Schilbach, F. Realizing the potential of digital development: The case of agricultural advice. *Science* **2019**, *366*, eaay3038. [CrossRef]

24. Bassier, M.; Vincke, S.; de Lima Hernandez, R.; Vergauwen, M. An overview of innovative heritage deliverables based on remote sensing techniques. *Remote Sens.* **2018**, *10*, 1607. [CrossRef]
25. Matias, F.I.; Caraza-Harter, M.V.; Endelman, J.B. FIELDimageR: An R package to analyze orthomosaic images from agricultural field trials. *Plant Phenome J.* **2020**, *3*, e20005. [CrossRef]
26. Fahlgren, N.; Feldman, M.; Gehan, M.A.; Wilson, M.S.; Shyu, C.; Bryant, D.W.; Hill, S.T.; McEntee, C.J.; Warnasooriya, S.N.; Kumar, I. A versatile phenotyping system and analytics platform reveals diverse temporal responses to water availability in Setaria. *Mol. Plant* **2015**, *8*, 1520–1535. [CrossRef] [PubMed]
27. Muhammad, W.; Esposito, F.; Maimaitijiang, M.; Sagan, V.; Bonaiuti, E. Polly: A Tool for Rapid Data Integration and Analysis in Support of Agricultural Research and Education. *Internet Things* **2020**, *9*, 100141. [CrossRef]
28. Gulli, A.; Pal, S. *Deep Learning with Keras*; Packt Publishing Ltd.: Birmingham, UK, 2017.
29. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. {TensorFlow}: A System for {Large-Scale} Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
30. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
31. Blanchette, J.; Summerfield, M. *C++ GUI Programming with Qt 4*; Prentice Hall Professional: Saddle River, NJ, USA, 2006.
32. Gillies, S. *Rasterio Documentation*; MapBox: San Francisco, CA, USA, 2019; Volume 23.
33. GeoPandas Developers. GeoPandas. Available online: https://geopandas.org/en/stable/ (accessed on 1 May 2022).
34. Boggs, T. Spectral Python. Available online: https://www.spectralpython.net/ (accessed on 15 October 2023).
35. Amigo, J.M.; Babamoradi, H.; Elcoroaristizabal, S. Hyperspectral image analysis. *A tutorial. Anal. Chim. Acta* **2015**, *896*, 34–51. [CrossRef] [PubMed]
36. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Schaub, T.; Schmidt, C. GeoJSON. Electronic. Available online: http://geojson.org (accessed on 1 May 2022).
37. Gillies, S. Fiona. Available online: https://pypi.org/project/Fiona/ (accessed on 15 October 2023).
38. Jordahl, K.; Van den Bossche, J.; Wasserman, J.; McBride, J.; Gerard, J.; Tratner, J.; Perry, M.; Farmer, C. *Geopandas/Geopandas: v0. 5.0*; Zenodo: Meyrin, Switzerland, 2021.
39. Gillies, S. The Shapely User Manual. Available online: https://pypi.org/project/shapely (accessed on 15 October 2023).
40. Brochet-Nguyen. Geodaisy. Available online: https://pypi.org/project/geodaisy/ (accessed on 15 March 2024).
41. Hengl, T.; Roudier, P.; Beaudette, D.; Pebesma, E. plotKML: Scientific visualization of spatio-temporal data. *J. Stat. Softw.* **2015**, *63*, 1–25. [CrossRef]
42. Megies, T.; Beyreuther, M.; Barsch, R.; Krischer, L.; Wassermann, J. ObsPy–What can it do for data centers and observatories? *Ann. Geophys.* **2011**, *54*, 47–58.
43. Mahlein, A.K.; Rumpf, T.; Welke, P.; Dehne, H.W.; Plumer, L.; Steiner, U.; Oerke, E.C. Development of spectral indices for detecting and identifying plant diseases. *Remote Sens. Environ.* **2013**, *128*, 21–30. [CrossRef]
44. Xue, J.R.; Su, B.F. Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications. *J. Sens.* **2017**, *2017*, 1353691. [CrossRef]
45. Roy, D.P.; Wulder, M.A.; Loveland, T.R.; Woodcock, C.E.; Allen, R.G.; Anderson, M.C.; Helder, D.; Irons, J.R.; Johnson, D.M.; Kennedy, R. Landsat-8: Science and product vision for terrestrial global change research. *Remote Sens. Environ.* **2014**, *145*, 154–172. [CrossRef]
46. Haralick, R.M.; Shanmugam, K.; Dinstein, I.H. Textural features for image classification. *IEEE Trans. Syst. Man Cybern.* **1973**, *SMC-3*, 610–621. [CrossRef]
47. Duan, M.Q.; Zhang, X.G. Using remote sensing to identify soil types based on multiscale image texture features. *Comput. Electron. Agric.* **2021**, *187*, 106272. [CrossRef]
48. De Grandi, G.D.; Lucas, R.M.; Kropacek, J. Analysis by Wavelet Frames of Spatial Statistics in SAR Data for Characterizing Structural Properties of Forests. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 494–507. [CrossRef]
49. Mutanga, O.; Skidmore, A.K. Narrow band vegetation indices overcome the saturation problem in biomass estimation. *Int. J. Remote. Sens.* **2004**, *25*, 3999–4014. [CrossRef]
50. Ressel, R.; Frost, A.; Lehner, S. A Neural Network-Based Classification for Sea Ice Types on X-Band SAR Images. *IEEE J.-Stars* **2015**, *8*, 3672–3680. [CrossRef]
51. Kupidura, P. The Comparison of Different Methods of Texture Analysis for Their Efficacy for Land Use Classification in Satellite Imagery. *Remote Sens.* **2019**, *11*, 1233. [CrossRef]
52. Warmerdam, F.; Rouault, E. GDAL. Available online: https://gdal.org/index.html (accessed on 1 May 2022).
53. Van der Walt, S.; Schönberger, J.L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J.D.; Yager, N.; Gouillart, E.; Yu, T. scikit-image: Image processing in Python. *PeerJ* **2014**, *2*, e453. [CrossRef]
54. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, 32.
55. Singla, S.; Eldawy, A. Raptor Zonal Statistics: Fully Distributed Zonal Statistics of Big Raster+ Vector Data. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 571–580.

56. Chen, N.W.; Li, H.C.; Wang, L.H. A GIS-based approach for mapping direct use value of ecosystem services at a county scale: Management implications. *Ecol. Econ.* **2009**, *68*, 2768–2776. [CrossRef]
57. Whiteaker, T.L.; Maidment, D.R.; Gopalan, H.; Patino, C.; McKinney, D.C. Raster-network regionalization for watershed data processing. *Int. J. Geogr. Inf. Sci.* **2007**, *21*, 341–353. [CrossRef]
58. Shan, J.; Zaheer, M.; Hussain, E. Study on accuracy of 1-degree DEM versus topographic complexity using GIS zonal analysis. *J. Surv. Eng.* **2003**, *129*, 85–89. [CrossRef]
59. Saadat, H.; Adamowski, J.; Bonnell, R.; Sharifi, F.; Namdar, M.; Ale-Ebrahim, S. Land use and land cover classification over a large area in Iran based on single date analysis of satellite imagery. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, 608–619. [CrossRef]
60. Perry, M.T. Rasterstats. Available online: https://pythonhosted.org/rasterstats/ (accessed on 1 May 2022).
61. Khan, Z.; Miklavcic, S.J. An Automatic Field Plot Extraction Method From Aerial Orthomosaic Images. *Front. Plant Sci.* **2019**, *10*, 683. [CrossRef]
62. Chen, C.J.; Zhang, Z. GRID: A python package for field plot phenotyping using aerial images. *Remote Sens.* **2020**, *12*, 1697. [CrossRef]
63. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
64. Holtwick, D. xhtml2pdf. Available online: https://pypi.org/project/xhtml2pdf/ (accessed on 15 October 2023).
65. Butcher, G. *Tour of the Electromagnetic Spectrum*; National Aeronautics and Space Administration: Washington, DC, USA, 2010.
66. Yang, C.; Baireddy, S.; Cai, E.; Crawford, M.; Delp, E.J. Field-Based Plot Extraction Using UAV RGB Images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 1390–1398.
67. Wang, L.; Tian, Y.; Yao, X.; Zhu, Y.; Cao, W. Predicting grain yield and protein content in wheat by fusing multi-sensor and multi-temporal remote-sensing images. *Field Crop. Res.* **2014**, *164*, 178–188. [CrossRef]
68. Pantazi, X.E.; Moshou, D.; Alexandridis, T.; Whetton, R.; Mouazen, A. Wheat yield prediction using machine learning and advanced sensing techniques. *Comput. Electron. Agric.* **2016**, *121*, 57–65. [CrossRef]
69. Kang, Y.; Khan, S.; Ma, X. Climate change impacts on crop yield, crop water productivity and food security—A review. *Prog. Nat. Sci.* **2009**, *19*, 1665–1674. [CrossRef]
70. Alexandratos, N.; Bruinsma, J. *World Agriculture towards 2030/2050: The 2012 Revision*; FAO: Rome, Italy, 2012.
71. Yang, C.; Everitt, J.; Du, Q.; Bin, L.; Chanussot, J. Using High-Resolution Airborne and Satellite Imagery to Assess Crop Growth and Yield Variability for Precision Agriculture. *Proc. IEEE* **2013**, *101*, 582–592. [CrossRef]
72. Schut, A.G.T.; Traore, P.; Blaes, X.; de By, R. Assessing yield and fertilizer response in heterogeneous smallholder fields with UAVs and satellites. *Field Crop. Res.* **2018**, *221*, 98–107. [CrossRef]
73. Mourtzinis, S.; Arriaga, F.; Balkcom, K.; Ortiz, B. Corn Grain and Stover Yield Prediction at R1 Growth Stage. *Agron. J.* **2013**, *105*, 1045–1050. [CrossRef]
74. Panda, S.; Ames, D.; Panigrahi, S. Application of Vegetation Indices for Agricultural Crop Yield Prediction Using Neural Network Techniques. *Remote Sens.* **2010**, *2*, 673. [CrossRef]
75. McBratney, A.; Whelan, B.; Ancev, T.; Bouma, J. Future Directions of Precision Agriculture. *Precis. Agric.* **2005**, *6*, 7–23. [CrossRef]
76. Bausch, W.C.; Duke, H.R. Remote Sensing of Plant Nitrogen Status in Corn. *Trans. ASABE* **1996**, *39*, 1869–1875. [CrossRef]
77. Elsayed, S.; Elhoweity, M.; Ibrahim, H.; Dewir, H.; Migdadi, H.; Schmidhalter, U. Thermal imaging and passive reflectance sensing to estimate the water status and grain yield of wheat under different irrigation regimes. *Agric. Water Manag.* **2017**, *189*, 98–110. [CrossRef]
78. Yu, N.; Li, L.; Schmitz, N.; Tian, L.F.; Greenberg, J.; Diers, B. Development of methods to improve soybean yield estimation and predict plant maturity with an unmanned aerial vehicle based platform. *Remote Sens. Environ.* **2016**, *187*, 91–101. [CrossRef]
79. Elsayed, S.; Rischbeck, P.; Schmidhalter, U. Comparing the performance of active and passive reflectance sensors to assess the normalized relative canopy temperature and grain yield of drought-stressed barley cultivars. *Field Crops Res.* **2015**, *177*, 148–160. [CrossRef]
80. Malenovsky, Z.; Homolova, L.; Zurita-Milla, R.; Lukes, P.; Kaplan, V.; Hanus, J.; Gastellu-Etchegorry, J.P.; Schaepman, M.E. Retrieval of spruce leaf chlorophyll content from airborne image data using continuum removal and radiative transfer. *Remote Sens. Environ.* **2013**, *131*, 85–102. [CrossRef]
81. Houborg, R.; Fisher, J.B.; Skidmore, A.K. Advances in remote sensing of vegetation function and traits. *Int. J. Appl. Earth Obs.* **2015**, *43*, 1–6. [CrossRef]
82. Sun, J.; Yang, J.; Shi, S.; Chen, B.W.; Du, L.; Gong, W.; Song, S.L. Estimating Rice Leaf Nitrogen Concentration: Influence of Regression Algorithms Based on Passive and Active Leaf Reflectance. *Remote Sens.* **2017**, *9*, 951. [CrossRef]
83. Gehan, M.A.; Fahlgren, N.; Abbasi, A.; Berry, J.C.; Callen, S.T.; Chavez, L.; Doust, A.N.; Feldman, M.J.; Gilbert, K.B.; Hodge, J.G. PlantCV v2: Image analysis software for high-throughput plant phenotyping. *PeerJ* **2017**, *5*, e4088. [CrossRef] [PubMed]
84. Python Software Foundation. tkinter. Available online: https://docs.python.org/3/library/tkinter.html (accessed on 1 October 2023).
85. Grinberg, M. *Flask Web Development: Developing Web Applications with Python*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2018.
86. Prechelt, L. An empirical comparison of c, c++, java, perl, python, rexx and tcl. *IEEE Comput.* **2000**, *33*, 23–29. [CrossRef]
87. Kattenborn, T.; Leitloff, J.; Schiefer, F.; Hinz, S. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS J. Photogramm. Remote Sens.* **2021**, *173*, 24–49. [CrossRef]

88. Zhang, W.; Tang, P.; Zhao, L. Remote sensing image scene classification using CNN-CapsNet. *Remote Sens.* **2019**, *11*, 494. [CrossRef]
89. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.-S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 8–36. [CrossRef]
90. LeDell, E.; Poirier, S. H2O automl: Scalable automatic machine learning. In Proceedings of the AutoML Workshop at ICML, Virtual Workshop, 18 July 2020.