



HAL
open science

Machine Learning for a Smart and Sustainable Agriculture

Christophe Maudoux, Selma Boumerdassi

► **To cite this version:**

Christophe Maudoux, Selma Boumerdassi. Machine Learning for a Smart and Sustainable Agriculture. SSA 2021, Jun 2021, Virtual conference, France. pp.103-121, 10.1007/978-3-030-88259-4_8. hal-03502870

HAL Id: hal-03502870

<https://hal.science/hal-03502870v1>


Submitted on 26 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.


L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Smart & Sustainable Agriculture Machine Learning behind this (R)evolution

Christophe Maudoux*

 0000-0001-5215-9046

Selma Boumerdassi*

 0000-0003-2603-2433

Abstract

Last decade has seen the emerging concept of Smart and Sustainable Agriculture that makes farming more efficient by minimizing environmental impacts. Behind this evolution, we find the scientific concept of Machine Learning. Nowadays, machine learning is everywhere throughout the whole growing and harvesting cycle.

Many algorithms are used for predicting when seeds must be planted. Then, data analyses are conducted to prepare soils and determine seeds breeding and how much water is required. Finally, fully automated harvest is planned and performed by robots or unmanned vehicles with the help of computer vision. To reach these amazing results, many algorithms have been developed and implemented.

This paper presents how machine learning helps farmers to increase performances, reduce costs and limit environmental impacts of human activities. Then, we describe basic concepts and the algorithms that compose the underlying engine of machine learning techniques. In the last parts we explore datasets and tools used in researches to provide cutting-edge solutions.

Index terms – Smart and Sustainable Agriculture, Machine Learning, Datasets, Supervised and Unsupervised Algorithms, Practical applications

I – INTRODUCTION

Smart farming is a concept of agriculture management based on information and communication technologies implemented to increase products quantity and quality [1]. Different kind of technologies can be found behind this concept: *(i)* Sensors to scan soils or scale water, light, humidity or temperature *(ii)* Software for specific farm applications *(iii)* Communication technologies like cellular network *(iv)* Positioning by GPS *(v)* Hardware and software systems that enable IoT-based solutions and automation like drones [2] *(vi)* Data analytics to make decision and predict outcomes.

All these technologies integrated on farms allow farm-holders making farming processes data-driven and data-enabled. This emerging concept makes agriculture more efficient. Next step is now to reach a sustainable agriculture with the help of high-precision algorithms. The mechanism that drives it is *machine learning* which is the scientific approach that provides machines the ability to learn by themselves [3]. It has emerged with the help of data sciences, computing progresses and the ability to collect then store more and more data.

The rest of this article is organized as follows. Section II presents an overview of ML applications in employed by smart agriculture. In section III, we expose main concepts behind machine learning. Sections IV and V describe with details some supervised and unsupervised algorithms respectively. In section VII, we present some widely employed tools and detail **Weka**. Then, some common metrics for evaluating approach performances are defined. Section VIII concludes this paper.

II – APPLICATIONS

From *smart farming* to *sustainable agriculture*, Machine Learning Algorithms or MLAs are mechanisms behind this evolution. The most popular models deployed in agriculture are ANN which stands for Artificial Neural Networks and Support Vector Machines (SVM).

2.1 Crop Management

2.1.1 Yield Prediction

ML can help to determine the best yield mapping or crop type to match supply with demand. Furthermore, an early and reliable estimation of crop yield is essential in quantitative and financial evaluation at the field level for determining strategic plans in agricultural commodities for import-export policies and increasing farmer's incomes [4].

2.1.2 Crop Quality

The accurate detection and classification of crops can increase gain and reduce waste. ML can analyse interconnections to highlight new features biasing crops quality [5].

*Cnam Paris – Cedric/Networks and IoT Systems, 75003 Paris, France. Email: {firstname.name}@cnam.fr

2.1.3 Disease Detection

To control diseases, farmers spread pesticides over the cropping areas. To be effective, this method requires significant amounts. ML can be used here for limiting financial costs and environmental impacts [6].

2.1.4 Weed Fighting

Weeds are harmful for crop production and it can be a boring and difficult task to get rid of this threat. Main problem in weed fighting is that they are difficult to highlight and discriminate from crops. MLAs can overcome these difficulties by minimizing cost and side effects [7].

2.2 Species Management

2.2.1 Species Breeding

Species selection is a tedious process. It requires identifying and selecting specific genes that determine adaptation to climate change, protect plants against diseases, provide a better taste, and so on MLAs take huge amount of data to analyse crops performance and build a model to determine which genes will increase a specific feature to a plant [8].

2.2.2 Species Classification

Classical approach consists in comparing colour and leaves shape. ML can provide more accurate and faster results by analysing the leaf vein morphology which expresses more information [9].

2.2.3 Species Reproduction

Pollination is the process in which pollen is taken from one plant to another so that it can reproduce. This task is essentially done by bees and insects. Problem is that bee number is dramatically decreasing. ML can help to detect the most fertile plants or select the most suitable time period when fertilization can happen and deploy drones to help in reproduction process [10].

2.3 Fields Management

2.3.1 Water Management

Water management in agriculture impacts environment balance. ML-based applications are connected with weather forecasting and evaporation estimations for a more effective use of irrigation systems [11].

2.3.2 Soil Management

Soil is a complex resource with many outside interactions. MLAs study evaporation processes, soil moisture and temperature to understand the dynamics of ecosystems and consequences [12].

2.4 Livestock Management

2.4.1 Livestock Production

ML predicts outcome or evolution to optimize economic efficiency of livestock production allowing farmers to modify feeding or life conditions [13].

2.4.2 Animal Welfare

Diseases, promiscuity, boredom can lead to lethal consequences for cattle. ML classifiers can detect behaviour changes to determine stress, unhappiness or weariness and alert that a problem could occur [14].

III – MAIN CONCEPTS

All applications presented above are based on ML. This part introduces underlying concepts and describes algorithms mainly deployed in agriculture.

3.1 Definitions

MLAs are programs that can learn from data and improve from experience. Learning tasks may include defining the best function that maps input to output, finding out the hidden structure of unlabelled data or grouping samples such that objects within the same cluster are more similar to each other than to the objects from another cluster.

The most common type of machine learning process is to learn the mapping function: $Y = f(X) + e$ to make predictions of Y for new X with an irreducible error e based on the lack of attributes to sufficiently characterize the best mapping. This is called predictive modelling. The goal here is to make the most accurate possible predictions, irrespective of the function form.

3.2 Assumptions

MLAs make different assumptions about the shape and structure of the function and how best to optimize a representation to approximate it. Due to this reason, it is very important to try out different algorithms on a machine learning problem.

Assumptions are used for optimizing learning process, but can restrict what can be learned. Algorithms using strong assumptions are named *parametric* MLAs. Algorithms with weak assumptions about the mapping function form are classified as *non-parametric* MLAs. By not making assumptions, they can learn any functional shape from the training data. *Decision Trees*, *Naive Bayes* or *Neural Networks* are non-parametric MLAs.

To summarize, *parametric* methods make large assumptions about the mapping function and are faster to train, require less data but may not be as powerful. *Non-parametric* methods make few or no assumptions about the target function and in turn require a lot more data, are slower to train and have a higher model complexity, but can result in more powerful models [15, 16].

IV – SUPERVISED ALGORITHMS

Input and output variables are available and an algorithm is used to learn the mapping function from the input to the output. It is called *supervised* learning because the learning process is based on a training dataset where the correct answers are already known. The algorithm iteratively makes predictions on the training data and is corrected by the “teacher”. Learning stops when the algorithm achieves an acceptable level of performance. Supervised learning tasks can be used for resolving *classification* (output variable is a real value) or *regression* (value is a category) problems. The aims of supervised algorithm are make machines learn explicitly, predict outcomes or direct feedback.

4.1 K-Nearest Neighbour

KNN does not require to pre-process the training dataset. KNN predicts new data by searching into all the data for the K most similar means named neighbours and summarizing the output variable for those instances. For *regression* the mean output variable is used, in *classification* it is the most common class value. A distance measurement, commonly the *Euclidean* one, is computed to define which of the K instances in the training dataset are most similar to a new entry.

$$\text{EuclideanDistance}(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2} \quad (1)$$

Other distance measures are existing like *Mahalanobis*, *Hamming*, *...*. The best distance metric to use depends on data properties. *Euclidean* suits if input data are close. Prefer the *Manhattan* one if not. The computational complexity of KNN increases with the amount of the training dataset. For huge dataset, KNN can be optimized by using stochastic approach. A snippet is cut out from the training dataset and the K-most similar instances are computed from it. For regression problems, the prediction is based on the median value. For classification, it is the highest frequency class from the K-most similar instances.

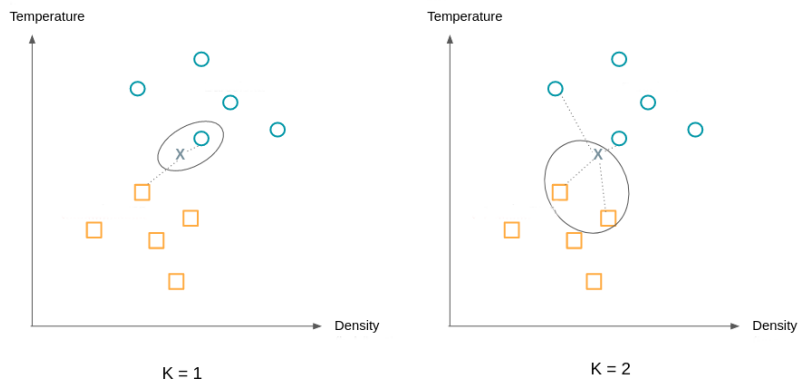


Figure 1: KNN example

4.2 Linear Discriminant Analysis

LDA is a dimensionality reduction algorithm used for multi-class classification. It aims to split the samples in a training dataset by their class value. The algorithm try to find out a linear combination of input variables that achieves the maximum separation for means between classes d^2 and the minimum variation of means within each class, the scatter s^2 . LDA creates an axis and projects the data onto this axis in a way to maximize the separation of categories.

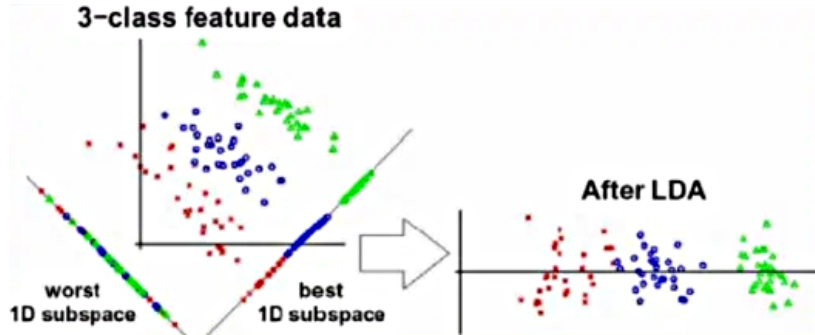


Figure 2: LDA example

4.3 Naive Bayes

NB is based on eponymic theorem equation (2) that provides a way to calculate the probability of a hypothesis given prior knowledge:

$$P(h|d) = \frac{P(d|h) P(h)}{P(d)} \quad (2)$$

Where $P(h|d)$ is the probability of hypothesis h given the data d (posterior probability), $P(d|h)$ the probability of data d given that the hypothesis h was true, $P(h)$ the probability of hypothesis h being true regardless of the data (prior probability of h) and $P(d)$ the probability of the data, regardless of the hypothesis. After calculating the posterior probability for a number of different hypotheses, the one with the highest probability is selected: the Maximum A Posteriori (MAP).

$$\text{MAP}(h) = \max[P(h|d)] \quad (3)$$

$$\Leftrightarrow \text{MAP}(h) = \max \left(\underbrace{\frac{P(d|h) P(h)}{P(d)}}_{\text{Bayes theorem 2}} \right) \quad (4)$$

$$\Leftrightarrow \text{MAP}(h) = \max[P(d|h)] \quad (5)$$

Equation (4) is obtained by substituting in equation (3) the equation (2) result. Then we find equation (5) after removing constants $P(d)$ and $P(h)$ in equation (4) because we are interested in the most probable hypothesis. Furthermore, we have an even number of instances in each class also the probability of each class will be the same.

It is named *Naive Bayes* because probabilities computation for each hypothesis are simplified and assumed to be conditionally independent. It requires a rigid independence assumption between input variables. Therefore, it is more proper to call “Simple Bayes” or “Independence Bayes”. Rather than attempting to calculate the values of each attribute value $P(d1, d2, d3|h)$, they are separately calculated like this: $P(d1|h) P(d2|h)$. Figure 3 depicts a NB classification.

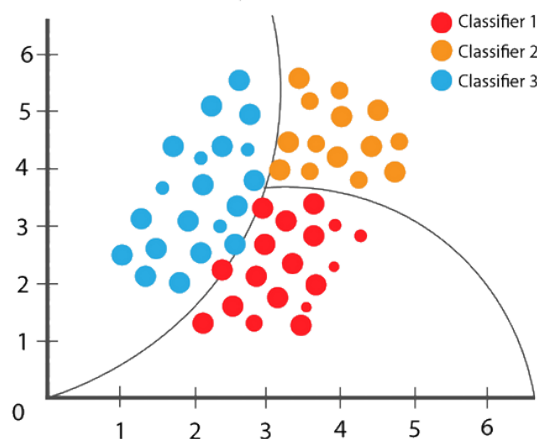


Figure 3: NB example

4.4 Support Vector Machines

SVM is the most popular algorithm. The numeric input variables consist in an n-dimensional space that can be split by a hyperplane. In SVM, this hyperplane is selected to best separate the points in the input variable space by their class. In a two-dimensional space, input data can be completely separated by a line matching this equation: $B_0 + (B_1X_1) + (B_2X_2) = 0$.

Where B_1 and B_2 determine the slope of the line and B_0 the intercept are found by the learning algorithm with X_1 and X_2 are the two input variables. The distance between the line and the closest data points is referred to as the margin. The optimal separation line is the one with the largest margin named the *Maximal-Margin hyperplane*. The margin is defined as the perpendicular distance from the line to only the closest points. Only these points are used for defining the line and building the classifier. These points are called the *support vectors* and defined the hyperplane figure 4.

The hyperplane is learned from training data using an optimization procedure that maximizes the margin. Learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. If the groups can not be easily split, values are transformed by a function to be able to separate them figure 5.

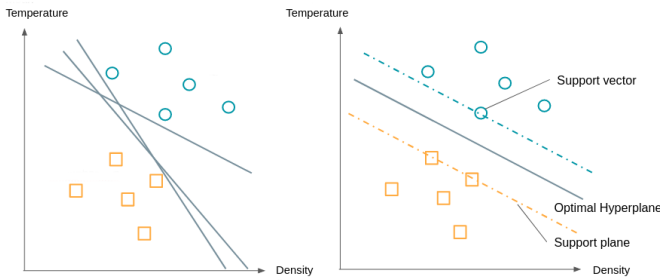


Figure 4: SVM example

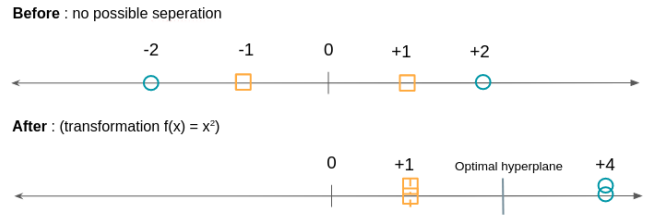


Figure 5: Separation example

4.5 Decision Tree

DT or Classification and Regression Trees (CART) can be used for classification or regression predictive modelling problems but are better at solving classification ones. DT are composed of two nodes: *Decision* and *Leaf* nodes. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches are the decision rules and each leaf node represents the outcome. *Decision nodes* are used for selecting a branch (make a choice) and have multiple branches. *Leaf nodes* are the output of those decisions and do not contain any further branches. Decisions are extracted from the given dataset and depend on its intrinsic features. A CART is a graphical representation for getting all possible solutions to a problem based on given conditions. It is called a DT because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure depicted by figure 6.

A decision tree simply asks a question, and based on the answer, it further splits the tree into subtrees. The representation is a binary tree as shown by figure 6. Each node represents a single input variable and a split point on that variable assuming it is numeric. The leaf nodes of the tree contain an output variable which is used to make a prediction. With the binary tree representation of the CART, making predictions is relatively straightforward. Given a new input, the tree is gone through by evaluating the specific input started at the root node of the tree.

4.6 Random Forest

RF is based on a multitude of DTs as shown in figure 7. It is one of the most popular and powerful MLAs. The problem with DTs is that they are greedy by nature. The variable which will be used to split the dataset is selected by using a greedy algorithm that minimizes error. Built DTs can have many structural similarities and so the same forecasting. By combining predictions from multiple models in ensembles works better if the forecasting from each sub-models are uncorrelated or weakly correlated. *Random Forest* changes the way the sub-trees are learned. With *Decision Trees*, when selecting a split-point, the learning algorithm looks through all variables and all variable values in order to select the most optimal split-point. With *Random Forest*, the learning algorithm is limited to a random sample of features on which to search. Searched features at each decision node have to be passed to the algorithm.

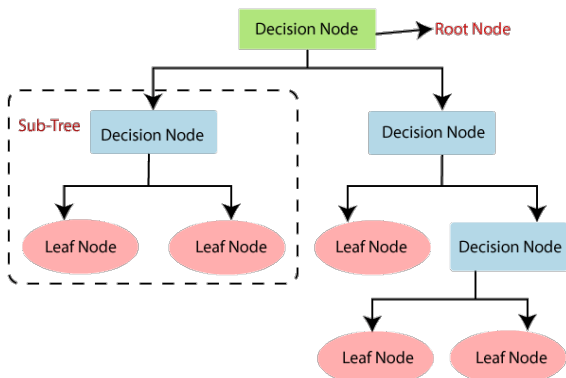


Figure 6: CART diagram

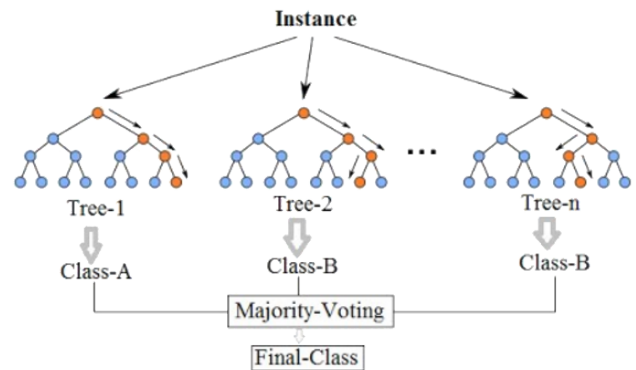


Figure 7: Simplified RF

4.7 Artificial Neural Network

ANN is very often used in agriculture. It represents the structure of a human brain. It consists of neurons and synapses organized into layers (figure 8). ANN can have a huge number of neurons connected together, which makes it extremely efficient at analysing and memorizing various information.

There are different types of ANN but always composed of the same components: neurons, synapses, biases, weights and functions.

Neuron or node (figure 9) is the elementary component that receives information, performs basic computation task, and sends it to the next ones. (i) Input neurons receive information (ii) Hidden neurons process that information (iii) Output neurons perform result.

In large NN, neurons are stacked into layers: (i) An input layer to receive information (ii) Some hidden layers to compute intermediate calculation (iii) An output layer that provides valuable results.

Every neuron performs transformation on the input information. Neurons only operate numbers in the range $[0,1]$ or $[-1,1]$. In order to transpose data into something that a neuron can handle, a normalization function like *Sigmoid* is required. Neurons communicate each other through synapses.

Synapses are links that connect each neuron. Every synapse has a weight. The weights also add to the changes in the input information. The results of the neuron with the higher weight will be dominant in the next neuron, while low important information will not be passed over. The initialization phase consists in randomly assign weights which will be optimized later.

Bias allows more variations of weights to be stored. Biases append more behaviours about the input space to the model's weights. In the case of neural networks, a bias neuron is added to every layer. It plays an important role by making it possible to tune the activation function.

Process aims for each neuron to extract a feature from input data. Each of the neurons has its own weights used for tuning the features. During the training phase, the "teacher" needs to select a weight for each of the neurons in order that the output provided by the whole network would be correct.

To perform transformations and get an output, every neuron has an activation function. This combination of functions performs a transformation described by a mapping function.

There are many types of Neural Networks. Main architectures are *Perceptron*, *Feed Forward*, *Multilayer Perceptron (MLP)*, *Convolutional* or *Radial Basis Function* to name a few. They are trained like any other algorithm. The weights of the ANN are repeatedly optimized until the difference between data and output is zero or close to it. Then the model is correctly able to predict an accurate output.

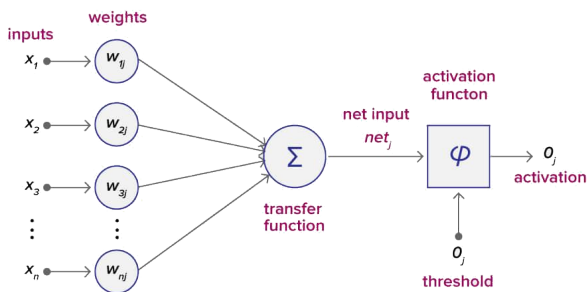


Figure 8: ANN structure

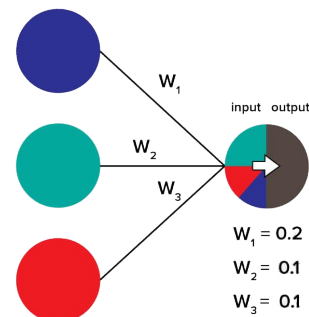


Figure 9: Neuron example

There are many types of ANN available or that might be in the development stage. They can be classified depending on their structure, data flow, neurons used and their density, number of layers and so on... Main architectures are Perceptron, Feed Forward, Multilayer Perceptron, Convolutional or Radial Basis Function for named a few.

Only input data are available and there is none corresponding output variables. The aim here is to model the underlying structure or distribution in the data in order to learn more about the data. This type of algorithms is called *unsupervised* learning because unlike *supervised* described above, there is no correct answers and no more “teacher”.

Algorithms find out themselves and present the interesting structure in the data. Unsupervised learning problems can be split into *clustering problems* if you just want to group data and *association problems* if you try to define rules that describe large amount of data. Unsupervised algorithms can not be used for predicting results but are useful to identify patterns or structures.

5.1 Principle Component Analysis

PCA is a technique to build relevant features through linear or non-linear (kernel) combinations of the original variables. Relevant features are highlighted by linearly transforming correlated variables into a smaller number of uncorrelated variables. To do this, original data are projected into the reduced PCA space using the eigenvectors of the covariance matrix also known as the Principal Components (PCs). The resulting projected data are essentially linear combinations of the original data capturing most of the variance in the data. PCA can be defined as an orthogonal transformation of the data into a series of uncorrelated data.

Goals of PCA is to find linearly independent dimensions which can represent the data points and should allow to predict the original dimensions.

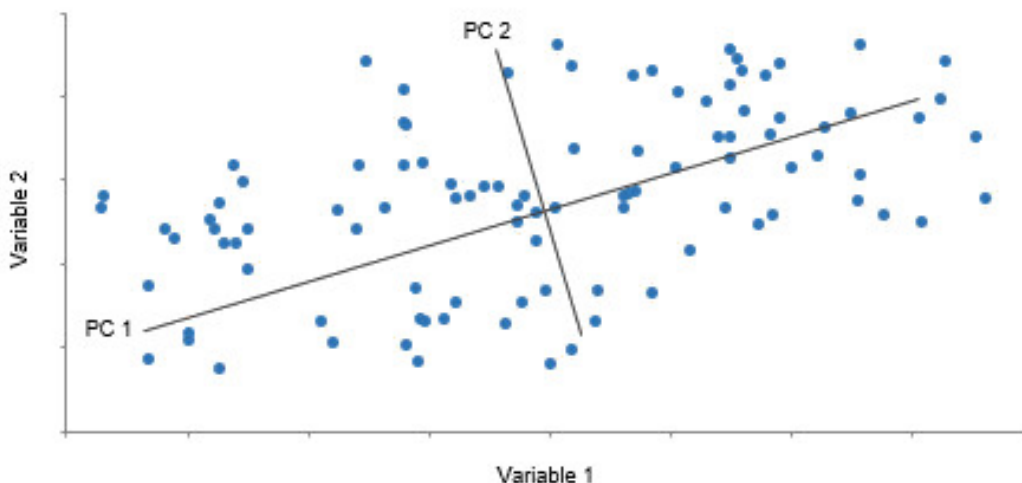


Figure 10: PCA example

5.2 K-Means

Even if we do not know how the clusters will be constituted, the *K-Means* algorithm imposes to provide the expected number of clusters K . Some techniques exist to find the optimal K . Algorithm begins by randomly positioning center points named the *starter means*. Then it associates with the same clusters observations closest to these means. Next it calculates the average of the observations from each cluster and move corresponding means to computed position. After, the algorithm re-assign the observations to the nearest means and so on. To ensure the stability of found groups, the *starter means* selection is repeated several times because some initial draws may give a different configuration from the huge majority of cases.

Gathering the objects into well separated clusters K is performed by *K-Means* in such way that objects within each cluster are as close as possible to each other but as far as possible from objects in other clusters. Each cluster is characterized by its centre point named *centroid* which is the point whose coordinates are the average of the coordinates of points assigned to the clusters. In a dataset, *K-Means* algorithm tries to find the specified K number of clusters and their k centroids.

In following example, we suppose that dataset contains two variables and we expect to get two clusters:

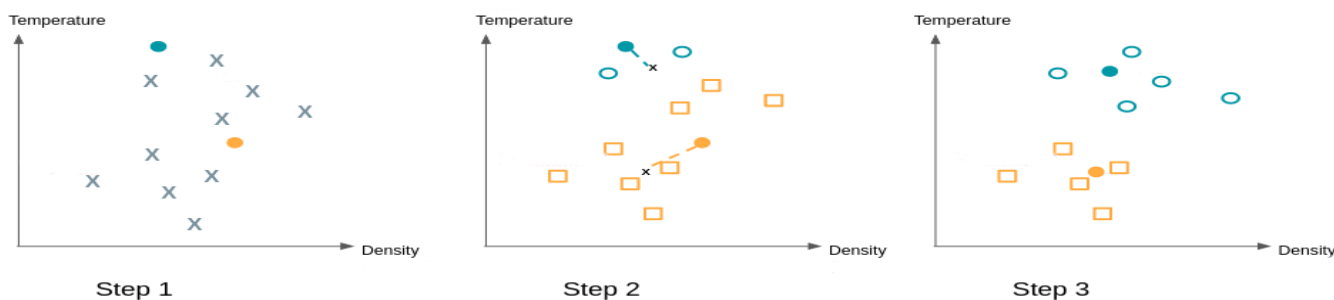


Figure 11: K-Means example

5.3 K-Medoids

This algorithm is a clustering process related to *K-Means* and *medoidshift* algorithms. *K-Means* tries to reduce the total squared error, while *k-medoids* minimizes the sum of dissimilarities between points labelled to be in the same cluster and a point designated as the centre of that cluster. In contrast to *K-Means*, it chooses means as centres also known as *medoids*. A *medoid* is a data point of a dataset with the lowest average dissimilarity. It can be defined as the most centrally located dot in the dataset.

K-Medoids algorithm is less sensitive to noise and outliers than *K-Means* (section 5.2) because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distances. Choice of dissimilarity function is very wide. Main weak-point of the *K-means* algorithm is its sensitivity to outliers, because a mean can be easily biased by out-range values. To avoid this drawback, *K-Medoids* algorithm uses an actual point in the cluster to represent the centre of a cluster instead of the mean point. A *medoid* is the most centrally located object of the cluster, with minimum sum of distances to other points.

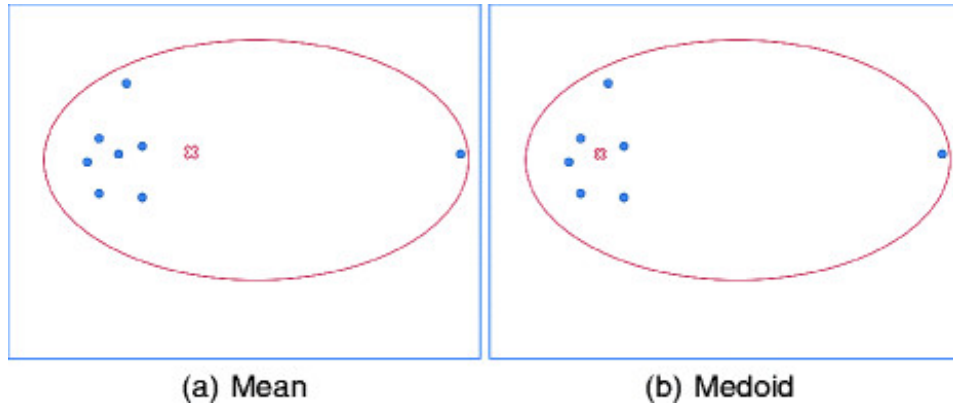


Figure 12: Mean versus Medoid

The group of points forms a cluster, while the rightmost point is an outlier. As mean can be biased, It could not represent the right cluster center. In the other hand, a medoid is robust to outliers and so correctly represents the cluster centre.

The most common implementation of *K-Medoid* clustering is the *Partitioning Around Medoids* algorithm. First, it randomly selects k of the n data points as the medoids. Secondly, the assignment step consists of associate each data point to the closest medoid. Then, for each medoid m and each data point o associated to m , it swaps m and o . Endlessly, the algorithm computes the total cost of the average dissimilarity of o to all the data points associated to m and select the medoid o with the lowest cost of the dissimilarity function. Steps 2 and 3 are repeated until there is no change in the assignments.

5.4 Mean Shift

It assigns the data points to the clusters iteratively by shifting points towards the highest density of data points in the region. Unlike *K-Means* (section 5.2) clustering algorithm, *Mean Shift* does not require specifying the number of clusters in advance because it is determined by the algorithm with respect to the data. Main drawback to *Mean Shift* is its expensive computation time.

Mean Shift clustering algorithm required that data are mathematically represented because it is based on *Kernel Density Estimation*. KDE is a probability function used for estimating the underlying distribution by placing a kernel on each point in the data set. A kernel is a weighting function. Many types of kernels are existing but the most popular is the *Gaussian* one. Adding up all the individual kernels generates a probability surface example density function. The resultant density function will be different depending on the kernel bandwidth parameter.

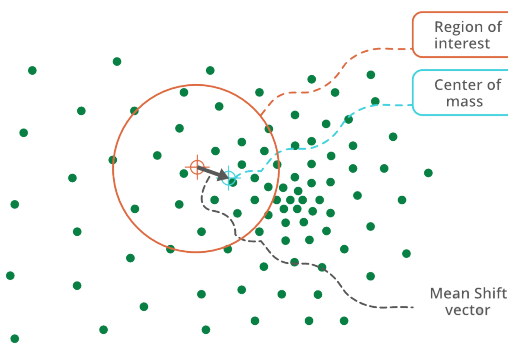


Figure 13: Mean Shift principle

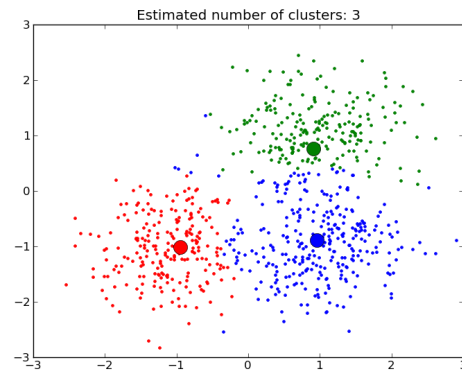


Figure 14: Density example

5.5 Density-based Spatial Clustering of Applications with Noise

DBSCAN groups points that are close to each other based on a distance measurement like Euclidean distance equation (1) and a minimum number of points. Means in low-density regions are considered as outliers. The key idea here is that for each point of a cluster, the neighbourhood of a given radius eps has to contain at least a minimum number of points named $MinPts$. Partitioning methods like *K-Means* (section 5.2) and hierarchical clustering are useful for finding compact and well-separated clusters with spherical or convex shapes.

But, real datasets contain arbitrary shape or noise. *DBSCAN* algorithm suits this kind of clusters better.

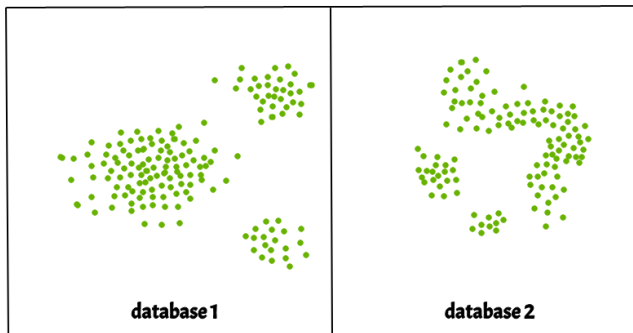


Figure 15: Well-separated clusters

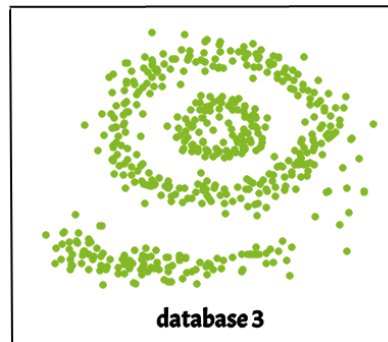


Figure 16: Arbitrary shapes and noise example

VI – DATASETS

Provide a dataset is the main difficulty to deploy ML. Furthermore, labelled datasets are required to implement supervised approaches. Some smart agriculture data can be found but the majority of them are not on digital format. To perform researches, universities or scientists have built and provided a few datasets. This part describes a few ones that can be freely used.

6.1 Rice Leaf Diseases

This dataset is provided by UC Irvine Machine Learning Repository, and it is used for detecting and classifying rice plant diseases [17]. It has been created by manually separating infected leaves into different disease classes. There are three categories of disease: bacterial leaf blight, brown spot, and Leaf smut. Each one is composed of 40 jpg images. They were captured with a white background, in direct sunlight.

6.2 PlantVillage

PlantVillage is a not-for-profit project created by Penn State University in the USA and EPFL in Switzerland [18]. 10,000 images of diseased and healthy crops have been collected. The aim is to develop algorithms that can accurately diagnose a disease based on an image. The dataset is offering 38 classes and 54,305 images of 14 different plant species in total, 12 of which are healthy, 26 of which are diseased. The dataset also has one more class identifying 1143 background. Thus, total number of images is up to 55,000.

6.3 DeepWeeds

It is a multi-class weed species image dataset for deep learning [19]. It consists of 17,509 images capturing eight different weed species native to Australia with neighbouring plants.

6.4 PlantDoc

PlantDoc is a dataset for visual plant disease detection released by researchers at Indian Institute of Technology in 2019 [20]. Early detection of plant diseases is difficult due to the lack availability of sufficiently large-scale dataset. This dataset can be used for benchmarking classification models. It contains 2,598 data points in total across 13 plant species and up to 17 classes of diseases extracted from annotated internet images. Unlike similar dataset like *CropDeep*, this dataset is freely available to download for machine or deep learning.

6.5 CropDeep

It is a species classification and detection dataset, consisting of 31,147 images with over 49,000 annotated instances from 31 different classes. In contrast to existing vision datasets, images were collected with different cameras and equipment in greenhouses, captured in a wide variety of situations. It features visually similar species and periodic changes with more representative annotations, which have supported a stronger benchmark for deep-learning-based classification and detection.

7.1 Machine learning implementation

Many tools and framework are existing to implement MLAs. We can cite four main tools: (i) Python with Jupyter is an interpreted high-level general-purpose programming language with many ML libraries. (ii) R with TensorFlow is a programming language provided with a free software environment named RStudio for statistical computing and graphics. The R language is widely used among statisticians and data miners. (iii) MatLab is a licenced proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. (iv) Weka a free, open-source and multiplatform collection of visualization tools and algorithms for ML.

7.2 The workbench for machine learning

Waikato Environment for Knowledge Analysis is a freely available and commonly used open source software developed at the University of Waikato in New Zealand. The *Weka Toolbox*¹ is written using object-oriented language Java. It provides implementation of state-of-the-art data mining or MLAs [21]. It affords many functionalities as association, filtering, classification, clustering, visualization, regression, and so on.

Overall, data must be converted into an ARFF (Attribute-Relation File Format) file. This is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections: (i) the Header which contains the name of the relation, a list of the attributes and their types, (ii) the Data:

```
% Comment
@RELATION iris
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-virginica}

@DATA
1,5,4,5,Iris-setosa
4,3,1,4,Iris-setosa
4,3,1,6,Iris-setosa
```

7.3 Basic features

Algorithm performances can be evaluated by using multiple metrics defined and explained below. We can define the following four basic features:

True Positives (TP) correctly predicted to be true positives
 \Rightarrow correctly classified as malware

True Negatives (TN) correctly predicted to be false and really negatives
 \Rightarrow correctly classified as benign

False Positives (FP) predicted to be true, but was incorrect, it was actually false
 \Rightarrow incorrectly classified as malware

False Negatives (FN) predicted to be false, but was incorrect., it was actually true
 \Rightarrow incorrectly classified as benign

7.4 Performance metrics

Therefore, main performance metrics are calculated based-on the previous basic features:

Accuracy $\frac{TP+TN}{TP+FN+FP+TN} \Rightarrow$ gives the overall accuracy of the model, meaning the fraction of the total samples that were correctly classified

Misclassification rate or Classification Error $\frac{FP+FN}{TP+FN+FP+TN}$
 \Rightarrow tells what fraction of predictions were incorrect.

Precision $\frac{TP}{TP+FP} \Rightarrow$ expresses what fraction of predictions as a positive class were actually positive.

True-Positive Rate (TPR) or Recall or Sensitivity $\frac{TP}{TP+FN}$
 \Rightarrow fraction of all positive correctly predicted as positive

True-Negative Rate (TNR) or Specificity $\frac{TN}{TN+FP}$
 \Rightarrow fraction of negative cases correctly predicted as negative

False-Positive Rate (FPR) $\frac{FP}{TN+FP} = 1 - \text{TNR}$

False-Negative Rate (FNR) $\frac{FN}{TP+FN} = 1 - \text{TPR}$

F-Measure (FM) or F1 Score $2 \frac{\text{Recall} \cdot \text{Precision}}{\text{Precision} + \text{Recall}}$
 \Rightarrow is the harmonic mean of *Precision* and *Recall* (where 1 is ‘good’ and 0 is ‘bad’).

¹<http://www.cs.waikato.ac.nz/ml/weka>

Matthews Correlation Coefficient (MCC)

$$\frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Sensitivity and Specificity are inversely proportional to each other. So when we increase *Sensitivity*, *Specificity* decreases and vice versa.

VIII – CONCLUSION

Agriculture is more and more connected or automated. A huge amount of data is generated by all these smart and communicating devices. Collect then analyse them by machine learning techniques is probably the most suitable solution to evolve from a smart to a sustainable agriculture.

To provide and perform efficient answers to practical problems like weeds fighting, animals well-fare or diseases prevention, accurate algorithms must be developed. But to reach this goal, real or freely materials are required to be able to test proposed approaches or experiment new applications.

Help farmers to digitalize existing data could be a win-win solution to fill this data lake. Another way is to assist them in every way to deploy smart agriculture and in other hand, compile and broadcast collected information.

A smart and sustainable agriculture is the key to reach a better environment and provide the best one to our descent.

References

- [1] Sciforce. *Smart Farming, or the Future of Agriculture*. Medium. Jan. 14, 2019. URL: <https://medium.com/sciforce/smart-farming-or-the-future-of-agriculture-359f0089df69> (visited on 06/27/2021).
- [2] *5 Top Harvest Automation Startups Impacting Agriculture*. StartUs Insights. Oct. 14, 2019. URL: <https://www.startus-insights.com/innovators-guide/5-top-harvest-automation-startups-impacting-agriculture/> (visited on 06/28/2021).
- [3] *Machine Learning in Agriculture: Applications and Techniques*. KDnuggets. URL: <https://www.kdnuggets.com/machine-learning-in-agriculture-applications-and-techniques.html/> (visited on 06/27/2021).
- [4] Mamunur Rashid et al. “A Comprehensive Review of Crop Yield Prediction Using Machine Learning Approaches With Special Emphasis on Palm Oil Yield Prediction”. In: *IEEE Access* 9 (2021), pp. 63406–63439. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2021.3075159](https://doi.org/10.1109/ACCESS.2021.3075159). URL: <https://ieeexplore.ieee.org/document/9410627/> (visited on 07/05/2021).
- [5] Arnab Kumar Saha et al. “IOT-Based Drone for Improvement of Crop Quality in Agricultural Field”. In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). Las Vegas, NV: IEEE, Jan. 2018, pp. 612–615. ISBN: 978-1-5386-4649-6. DOI: [10.1109/CCWC.2018.8301662](https://doi.org/10.1109/CCWC.2018.8301662). URL: <http://ieeexplore.ieee.org/document/8301662/> (visited on 07/05/2021).
- [6] G. Sambasivam and Geoffrey Duncan Opiyo. “A Predictive Machine Learning Application in Agriculture: Cassava Disease Detection and Classification with Imbalanced Dataset Using Convolutional Neural Networks”. In: *Egyptian Informatics Journal* 22.1 (Mar. 2021), pp. 27–34. ISSN: 11108665. DOI: [10.1016/j.eij.2020.02.007](https://doi.org/10.1016/j.eij.2020.02.007). URL: <https://linkinghub.elsevier.com/retrieve/pii/S1110866520301110> (visited on 07/05/2021).
- [7] Sajad Sabzi and Yousef Abbaspour-Gilandeh. “Using Video Processing to Classify Potato Plant and Three Types of Weed Using Hybrid of Artificial Neural Network and Partinle Swarm Algorithm”. In: *Measurement* 126 (Oct. 2018), pp. 22–36. ISSN: 02632241. DOI: [10.1016/j.measurement.2018.05.037](https://doi.org/10.1016/j.measurement.2018.05.037). URL: <https://linkinghub.elsevier.com/retrieve/pii/S026322411830424X> (visited on 07/05/2021).
- [8] Mahendar Thudi et al. “Genomic Resources in Plant Breeding for Sustainable Agriculture”. In: *Journal of Plant Physiology* 257 (Feb. 2021), p. 153351. ISSN: 01761617. DOI: [10.1016/j.jplph.2020.153351](https://doi.org/10.1016/j.jplph.2020.153351). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0176161720302418> (visited on 07/05/2021).
- [9] Ulrich Weiss et al. “Plant Species Classification Using a 3D LIDAR Sensor and Machine Learning”. In: *2010 Ninth International Conference on Machine Learning and Applications*. 2010 International Conference on Machine Learning and Applications (ICMLA). Washington, DC, USA: IEEE, Dec. 2010, pp. 339–345. ISBN: 978-1-4244-9211-4. DOI: [10.1109/ICMLA.2010.57](https://doi.org/10.1109/ICMLA.2010.57). URL: <http://ieeexplore.ieee.org/document/5708854/> (visited on 07/05/2021).
- [10] Sander Van Goethem et al. “An IoT Solution for Measuring Bee Pollination Efficacy”. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019 IEEE 5th World Forum on Internet of Things (WF-IoT’19). Limerick, Ireland: IEEE, Apr. 2019, pp. 837–841. ISBN: 978-1-5386-4980-0. DOI: [10.1109/WF-IoT.2019.8767298](https://doi.org/10.1109/WF-IoT.2019.8767298). URL: <https://ieeexplore.ieee.org/document/8767298/> (visited on 07/05/2021).

- [11] Joao Cardoso, Andre Gloria, and Pedro Sebastiao. “Improve Irrigation Timing Decision for Agriculture Using Real Time Data and Machine Learning”. In: *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI). Sakheer, Bahrain: IEEE, Oct. 26, 2020, pp. 1–5. ISBN: 978-1-72819-675-6. DOI: [10.1109/ICDABI51230.2020.9325680](https://doi.org/10.1109/ICDABI51230.2020.9325680). URL: <https://ieeexplore.ieee.org/document/9325680/> (visited on 07/05/2021).
- [12] Prachin Jain et al. “Maximising Value of Frugal Soil Moisture Sensors for Precision Agriculture Applications”. In: *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*. 2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G). Geneva, Switzerland: IEEE, Sept. 21, 2020, pp. 63–70. ISBN: 978-1-72817-031-2. DOI: [10.1109/AI4G50087.2020.9311008](https://doi.org/10.1109/AI4G50087.2020.9311008). URL: <https://ieeexplore.ieee.org/document/9311008/> (visited on 07/05/2021).
- [13] Shadi Nayeri, Mehdi Sargolzaei, and Dan Tulpan. “A Review of Traditional and Machine Learning Methods Applied to Animal Breeding”. In: *Anim. Health. Res. Rev.* 20.1 (June 2019), pp. 31–46. ISSN: 1466-2523, 1475-2654. DOI: [10.1017/S1466252319000148](https://doi.org/10.1017/S1466252319000148). URL: https://www.cambridge.org/core/product/identifier/S1466252319000148/type/journal_article (visited on 07/05/2021).
- [14] Yael Salzer et al. “Towards On-Site Automatic Detection of Noxious Events in Dairy Cows”. In: *Applied Animal Behaviour Science* 236 (Mar. 2021), p. 105260. ISSN: 01681591. DOI: [10.1016/j.applanim.2021.105260](https://doi.org/10.1016/j.applanim.2021.105260). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0168159121000472> (visited on 07/05/2021).
- [15] Jason Brownlee. *Master Machine Learning Algorithms: Discover How They Work and Implement Them*. Machine Learning Mastery, Mar. 4, 2016. 162 pp.
- [16] Gaël Bonnardot. *8 Machine Learning Algorithms Explained in Human Language*. Datakeen. Nov. 6, 2017. URL: <https://datakeen.co/en/8-machine-learning-algorithms-explained-in-human-language/> (visited on 06/30/2020).
- [17] Harshadkumar B. Prajapati, Jitesh P. Shah, and Vipul K. Dabhi. “Detection and Classification of Rice Plant Diseases”. In: *IDT* 11.3 (Aug. 29, 2017), pp. 357–373. ISSN: 18724981, 18758843. DOI: [10.3233/IDT-170301](https://doi.org/10.3233/IDT-170301). URL: <https://www.medra.org/servlet/aliasResolver?alias=iiospress&doi=10.3233/IDT-170301> (visited on 06/27/2021).
- [18] David P. Hughes and Marcel Salathe. *An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics*. Apr. 11, 2016. arXiv: [1511.08060](https://arxiv.org/abs/1511.08060) [cs]. URL: <http://arxiv.org/abs/1511.08060> (visited on 06/28/2021).
- [19] Alex Olsen. *AlexOlsen/DeepWeeds*. June 21, 2021. URL: <https://github.com/AlexOlsen/DeepWeeds> (visited on 06/28/2021).
- [20] Pratik Kayal. *PratikKayal/PlantDoc-Dataset*. June 24, 2021. URL: <https://github.com/pratikayal/PlantDoc-Dataset> (visited on 06/27/2021).
- [21] Ian H. Witten et al. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. San Diego, CA, USA: Morgan Kaufmann, Oct. 1, 2016. 655 pp. ISBN: 978-0-12-804357-8.