Classification of wheat and barley fields using high-resolution Sentinel-1 backscatter data

A Degree Thesis Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Alejandro Domínguez Costa

In partial fulfillment

of the requirements for the degree in

TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisors: Núria Duffo, Wolfgang Wagner & Isabella Pfeil

Vienna, June 2020

Abstract

In a world where food demand keeps increasing due to constant population growth, it is very important to ensure food supply to all the inhabitants despite the high arable land demand. To manage that, it is crucial that the yield of winter cereal fields is the highest possible. In the context of crop monitoring and optimizing field management, Earth Observation (EO) methods have been used in different ways. The main goal of this thesis is to develop a reliable algorithm capable of classifying wheat and barley fields from the Netherlands and Austria using statistical methods and high-resolution backscatter data from ESA's Copernicus mission that uses synthetic aperture radars embarked on both Sentinel-1 satellites. To develop this algorithm it was first necessary to study the phenological stages of crops, the backscatter throughout the seasons and inter-annual variability due to different temperature, rainfall and soil moisture conditions.

Research results give three different types of classifications, each one with a different grade of complexity. The most complex and consistent algorithm is able to classify winter cereals with an average success rate of 83.55% for the three studied seasons and up to a maximum success rate of 91.90% for a single season.

In conclusion, this project demonstrated that even though winter wheat and winter barley backscatter response is very similar during large parts of the growing season, significant differences are found during the phases of head formation and bending, which lead to structural effects in the backscatter time series. Furthermore, it could be shown that it is possible to separate these crop types with a high accuracy using only microwave remote sensing observations.

Resum

En un món on la demanda d'aliments continua augmentant a causa del creixement constant de la població, és molt important assegurar el subministrament d'aliments a tots els habitants malgrat l'elevada demanda de terres de cultiu. Per aconseguir-ho, és crucial que el rendiment dels camps de cereals d'hivern sigui el més alt possible. En el context del monitoratge dels cultius i l'optimització de la gestió del camp, l'observació terrestre des de l'espai s'usa de diverses maneres.

L'objectiu principal d'aquesta tesi és desenvolupar un algoritme fiable capaç de classificar els camps de blat i ordi dels Països Baixos i Àustria mitjançant mètodes estadístics i dades de retrodispersió d'alta resolució de la missió Copernicus de l'ESA que utilitza radars d'obertura sintètica embarcats en els dos satèl·lits Sentinel-1. Per desenvolupar aquest algorisme, primerament va ser necessari estudiar les etapes fenològiques dels cultius, les dades de satèl·lit al llarg de les temporades i la variabilitat interanual a causa de les diferències de temperatura, precipitacions i condicions d'humitat del terreny.

Els resultats de la investigació donen tres tipus diferents de classificacions, cadascuna amb un grau de complexitat diferent. L'algoritme més complex i consistent és capaç de classificar els cereals d'hivern amb una taxa d'èxit mitjana d'un 83,55 % per a les tres temporades estudiades, amb una taxa d'èxit màxima de 91,90 % per a una sola temporada.

En conclusió, aquest projecte demostra que, tot i que la retrodispersió de blat i ordi d'hivern observada des de satèl·lit és molt similar durant gran part de l'any, diferències significants són identificades durant les fases de formació i doblegament del cap dels cultius, que provoquen canvis estructurals en les gràfiques temporals de retrodispersió. A més a més, es visualitza la possibilitat de separar ambdós tipus de conreus amb gran precisió utilitzant exclusivament tècniques de teledetecció per microones.

Resumen

En un mundo donde la demanda de alimentos sigue aumentando debido al crecimiento constante de la población, es muy importante asegurar el suministro de alimentos a todos los habitantes a pesar de la elevada demanda de tierras de cultivo. Para ello, es crucial que el rendimiento de los campos de cereales de invierno sea el mas alto posible. En el contexto del monitoreo de cultivos y la optimización de la gestión del campo, la observación de la Tierra desde el espacio se utiliza de distintas maneras.

El objetivo principal de esta tesis es desarrollar un algoritmo fiable capaz de clasificar los campos de trigo y cebada de los Países Bajos y Austria mediante métodos estadísticos y datos de retrodispersión de alta resolución de la misión Copernicus de la ESA que utiliza radares de apertura sintética embarcados en los dos satélites Sentinel-1. Para desarrollar este algoritmo, en primer lugar fue necesario estudiar las etapas fenológicas de los cultivos, los datos de satélite a lo largo de las temporadas y la variabilidad interanual debida a las diferencias de temperatura, precipitaciones y condiciones de humedad del terreno.

Los resultados de la investigación dan tres tipos diferentes de clasificaciones, cada una con un grado de complejidad diferente. El algoritmo más complejo y consistente es capaz de clasificar los cereales de invierno con una tasa de éxito promedio de hasta un 83,55 % para las tres temporadas estudiadas, con un máximo de 91,90 % para una sola temporada.

En conclusión, este proyecto demuestra que, aunque la retrodispersión de trigo y cebada observada desde satélite es muy similar durante gran parte del año, diferencias significativas son identificadas durante las fases de formación y doblamiento de la cabeza de los cultivos, que provocan cambios estructurales en los gráficos temporales de retrodispersión. Finalmente, se visualiza la posibilidad de separar ambos tipos de cultivos con una gran precisión utilizando exclusivamente técnicas de teledetección por microondas.

A mi abuelo Agustín, que me estará observando desde el cielo.

Acknowledgements

I would like to thank Isabella Pfeil and Wolfgang Wagner for the opportunity of doing my thesis at the Geodesy and Geoinformation at Technische Universiät Wien, I am so grateful that they trusted me to perform such an important task at their department. I have to mention that Isabella has especially shown a huge dedication to help me doing a good work, always suggesting new ideas or asking doubts. I am really thankful. Vielen dank, Isabella.

Obviously, I would like to thank Núria Duffo, who was the first to tell me about going to Vienna and has always been aware of my progress and personal situation. Muchas gracias, Núria.

I also want to say thanks to all the amazing people that have shared my stay in Vienna, I have felt like home with them. Muchas gracias por todo Marc, Esther, Adri, María y Chen!

And last but not least, I am so grateful for the amazing family and friends I have. Muchas gracias Pili, Paco y Anna por animarme desde Teià. També agraïr a tots els amics que hem mantigut contacte tot i la distància, no us oblidaré mai. Finalment, a tu també Paula, per recolzar-me en tot moment i per tot el que hem viscut i ens queda per viure, t'estimo.

Contents

List of Figures

List of Tables

1 Introduction

This project has been carried out at the Geodesy and Geoinformation department of Technische Universität Wien (TUW). The development of this project has been realized thanks to the host university and Universitat Politècnica de Catalunya (UPC), in the context of Erasmus + KA103 studies program.

1.1 Statement of purpose

The agriculture industry is facing many challenges in the present and future, such as the reduced availability of arable land due to climate change [2], the high food demand as a result of the constant population growth, and the increasing importance of water use efficiency.

Crop monitoring is key to analyze crop field performance and to increase irrigation profitability. However, a first classification of crop types is needed to get a good analysis of field properties depending on the harvested crops. The purpose of this project is to distinguish barley and wheat fields using high-resolution Sentinel-1 backscatter data. Both crop types behave very similarly in their growing cycle. In order to find differences between them, statistical methods will be applied to Earth Observation (EO) data over numerous fields in the Netherlands.

The main project goals are explained here after. First of all, to describe the temporal behavior of C-band backscatter over wheat and barley fields through the growing season and put it in relation to crop growing stages. Secondly, to evaluate differences between wheat and barley fields and shifts in time series because of weather variability. Thirdly, to develop an automatic classification of wheat and barley fields using decision trees based on previously found thresholds describing the temporal differences between wheat and barley backscatter. Lastly, to test the algorithm using Netherlands and Austria data.

1.2 Requirements and Specifications

On the one hand, to describe accurately how growing patterns of wheat and barley are reflected in Sentinel-1 backscatter and how they differ. On the other hand, to develop an automatic classification algorithm of fields based only on backscatter data. Thereby, co- (VV) and cross-polarized (VH) backscatter and Cross Ratio index (CR) will be used.

$$CR = \frac{VH}{VV}$$

As a self-specification exercise, the final goal of the project would be successfully achieved as long as a decision tree with at least 75% success rate is created.

1.3 Methods and Procedures

This project is linked to the research considered in the following articles: "Sensitivity of Sentinel-1 Backscatter to Vegetation Dynamics: An Austrian Case Study" [3], which was developed at the Department of Geodesy and Geoinformation and "Analyzing Temporal and Spatial Characteristics of Crop Parameters Using Sentinel-1 Backscatter Data" [4], which was realized in Germany.

Up to this moment, researchers at the Geodesy and Geoinformation department at TUW have not succeeded in quantifying differential characteristics (e.g. time lags, absolute backscatter values) of winter cereals that allow their classification. The main goal of this undertaking is to solve that issue by carrying out a detailed investigation of wheat and barley backscatter time series.

The software used in this project is open source.Python programming language (version 3.6) is harnessed in the analysis of backscatter time series for a large number of fields in the Netherlands (almost 3000). For the purpose of testing the scalability of the decision tree, Austrian fields will be applied to test the algorithm as well.

The initial ideas were suggested by the host university supervisors.

1.4 Work Plan

First, some weeks were needed to become accustomed to the Python environment displaying basic plots and starting to work with dataframes. Afterwards, absolute and relative backscatter time series of the available seasons for wheat and barley were plotted and their behavior along the growing stage of fields was identified. Thereafter, the periods where crops differed the most were distinguished and thresholds were defined to implement them in the decision tree. After that, the impact of inter-annual weather variations on backscatter measurements was evaluated. Based on previous results, all the potential thresholds were collected and tested in a first decision tree. Later on, the ones that gave better results were selected to obtain an optimum algorithm that allows us to sort crop types with the minimum error rate possible. Finally, the final step was to transcript all the development, results and conclusions of the thesis into a document and make corrections.

Work Package	Start	End	2-Mar	9-Mar	16-Mar	23-Mar	30-Mar	6-Apr	13-Apr	20-Apr	27-Apr	4-May	11-May	18-May	25-May	1-Jun	8-Jun	15-Jun	22-Jun	29-Jun	6-Jul	13-Jul
Get used to the environment	2-Mar	16-Mar																				
Describe backscatter time series	16-Mar	13-Apr																				
Link results to crop growing stages	13-Apr	27-Apr																				
Analyse weather variability	27-Apr	11-May																				
Automatic classification	11-May	8-Jun																				
Thesis writing and corrections	8-Jun	13-Jul																				

Figure 1: Preliminar Work Plan

1.5 Deviations and Incidences

In order to prevent the spread of the Corona Virus, access to all buildings of the TU Wien was only possible to maintain the absolutely necessary from 19th March on. Therefore, all the work had to be done from home and the communication with the supervisor was exclusively via e-mail and video-chat since then. Consequently, the adaptation to the new situation may have caused subtle delays.

At the end, the timeline of the project was the shown in Figure 2.

Work Package	Start	End	2-Mar	9-Mar	16-Mar	23-Mar	30-Mar	6-Apr	13-Apr	20-Apr	27-Apr	4-May	11-May	18-May	25-May	1-Jun	8-Jun	15-Jun	22-Jun	29-Jun	6-Jul	13-Jul
Get used to the environment	2-Mar	16-Mar																				
Describe backscatter time series	23-Mar	13-Apr																				
Link results to crop growing stages	13-Apr	27-Apr																				
Analyse weather variability	30-Mar	13-Apr																				
Automatic classification	11-May	22-Jun																				
Thesis writing and corrections	8-Jun	13-Jul																				

Figure 2: Final

2 Remote sensing background

Remote Sensing (RS) is the technology that enables data acquisition of an object or phenomena without making physical contact. It has numerous applications, including but not limited to determining the state of physical, biological or geographic parameters such as polluting gas quantities in the atmosphere, the amount of snow cover over a region or the biomass of a specific crop [5].

Since most of this information is not directly accessible, one option is to study the ElectroMagnetic (EM) properties of the region of interest (i. e. maturity of a crop might be linked to its backscatter radiation). Thus, the sensor must be sensitive to the relevant part of the EM spectrum.

The greatest challenge in RS is to relate observable features with natural parameters. If this connection is ambiguous, data behavior can be misunderstood and lead to errors. Nevertheless, conceiving limitations allows algebraic simplifications which may fulfill the confined problem needs.

As a way to collect valuable information, it is very effective to use the strong wavelength-dependence of the scattering properties of media.

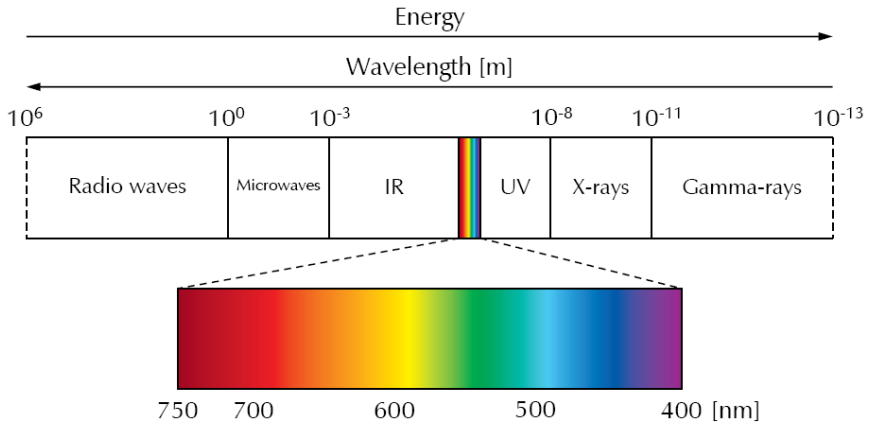


Figure 3: Electromagnetic Spectrum. Retrieved from: https://www.quora.com

Given the great width of the EM spectrum (see in Figure 3) worthwhile knowledge can be extracted from it. Beginning with the longest wavelengths there are radio waves, which are used in submarine communications and some radars. Microwaves (MW) come next. As they are commonly applied to RS of vegetation, they will be explained in more detail further below. InfraRed (IR) and visible groups also play a role in RS, and they are also described in the next section. Finally, UltraViolet (UV), X and γ rays bands are the highest frequencies of the spectrum, and they can determine the presence of moisture in objects.

Furthermore, radiation is divided into three categories based on the origin:

- Self emission: Radioactive substances
- Diffuse scattering of incoherent waves: Reflected sunlight

- Reflection of artificial, coherent waves: Surface reflected radiation originated at a satellite

2.1 Optical remote sensing

In the last 30 years, optical RS technologies have been used for precision agriculture applications [6]. They make use of visible, near infrared and short-wave infrared sensors to form images of the earth's surface by detecting solar radiation reflected from targets on the ground. Taking into account that every material absorbs light differently, the spectral behavior provides substantial information for different elements. The regular wavelength range of usage is compressed between $0.4\mu m$ and $1.7\mu m$. Optical RS systems are classified by the number of spectral bands used for the image processing:

- Panchromatic imaging system: Obtaining black and white images using one single band, e.g. WorldView-1 [7].
- Multispectral imaging system: Up to ten bands employed to get a multilayer image that contains color (spectral information) and brightness. This technology is applied in the French program Satellite Pour l'Observation de la Terre (SPOT) by the High Resolution Visible-XbandSpectral (HRV-XS) instrument.
- Superspectral imaging system: Typically around tens of narrow different bands, it gives a finer spectral resolution. Used in the National Aeronautics and Space Administration's (NASA) Moderate Resolution Imaging Spectroradiometer (MODIS) and ESA's Medium Resolution Imaging Spectrometer (MERIS) instruments.
- Hyperspectral imaging system: It makes use of hundreds of different frequencies that provide a huge number of applications such as phytoplankton monitoring, pollution observation, and moisture status. This is the technology utilized to observe vegetation characteristics. Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) uses that type of technology.

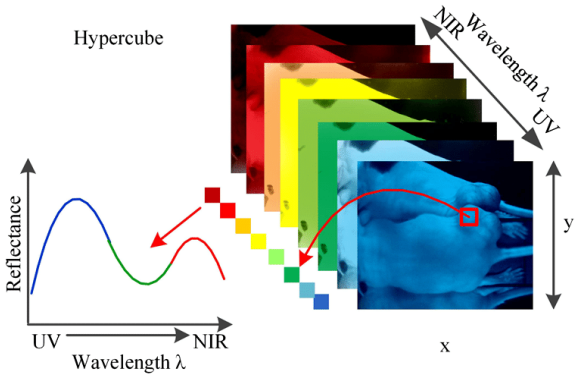


Figure 4: The image above represents a scheme of an image cube of hyperspectral imaging systems. Retrieved from: https://www.researchgate.net

Since optical RS is a passive system, the major drawback is the impact of cloud covering in readings. This is why this technology is only reliable in clear days in the agricultural context [6].

2.2 Microwave remote sensing

The MW domain is defined as wavelengths ranging from 1 mm to 1 m. Microwaves are able to penetrate the atmosphere, especially at longer wavelengths [1] (as it can be seen in Figure 5), and do not require sunlight. This makes MW remote sensing a highly suitable technique for many applications.

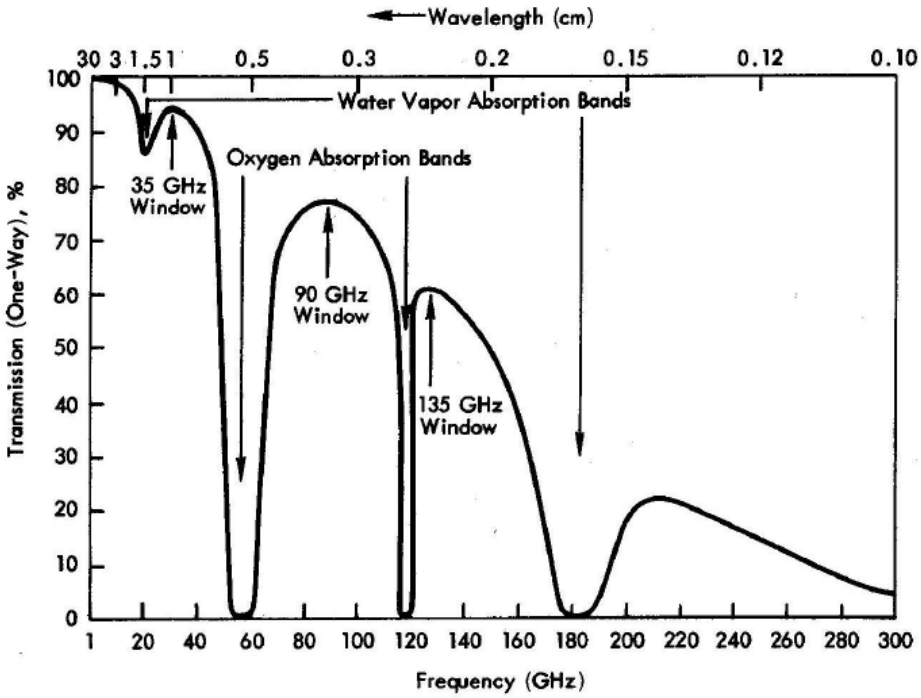
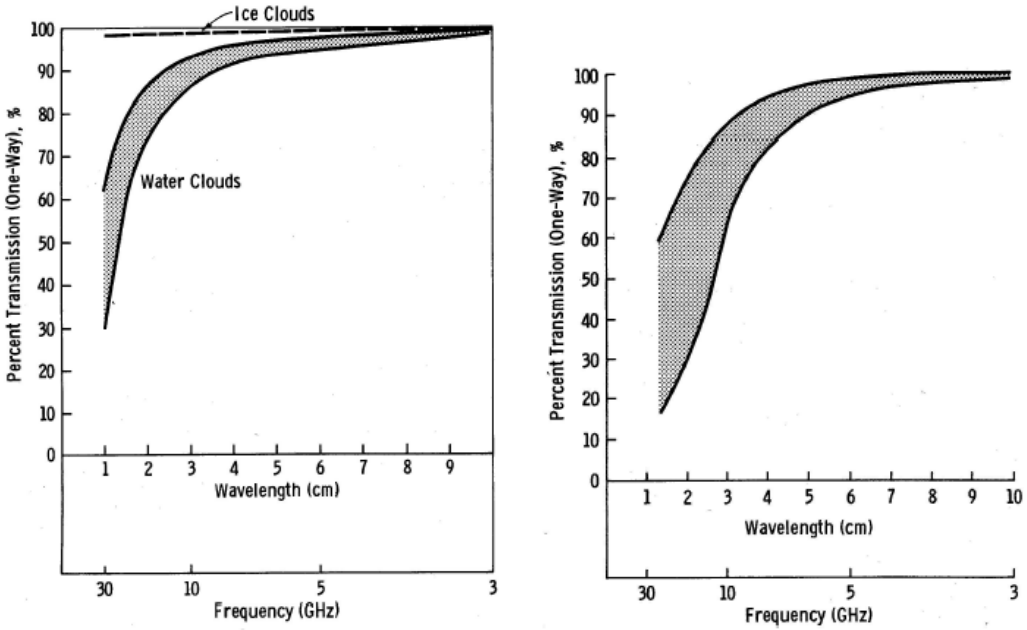


Figure 5: One-way transmission of MW through the atmosphere, along vertical direction and under clear sky conditions. Retrieved from: [1]

Unfortunately, meteorological phenomena have a negative effect in terms of MW propagation (Figure 6). The image on the left (Figure 6a) shows that ice clouds have practically no effect on MW, whereas this circumstance makes it impossible to capture aerial photographs. Nevertheless, water clouds have more influence as wavelength decreases, becoming significant when $\lambda < 2cm$. In Figure 6b, the rain effect displays a similar shape. The impact becomes considerable when $\lambda < 3cm$ and heavy rain.

To summarize, the repercussion of weather events is almost negligible when $\lambda > 4cm$, what reveals a great advantage of MW.



(a) Effect of clouds on MW

(b) Effect of rain on MW

Figure 6: Effect of meteorological phenomena present in the Earth's atmosphere. Retrieved from: [1]

Further to this, MW hold the ability to penetrate deeper into vegetation than optical waves. This capability relies on the wavelengths used, moisture content and vegetation density. Smaller frequencies and drier soil allow radiation to penetrate more easily than higher frequencies and wetter grounds. In any case, penetration is always deeper for MW than optical RS.

The last highlighted property of MW is that it gives additional information about vegetation, which complements visible and near-infrared data when conditions are suitable for three frequency ranges. MW backscatter depends on the dielectric properties, i.e., the water content of the surface or vegetation canopy to study, whereas optical remote sensing reflects molecular resonances. By combining both technologies, a complete understanding of the feature of interest is acquired.

2.2.1 Synthetic Aperture Radar

Synthetic Aperture Radar's (SAR) is based on the motion of the radar antenna over a region in order to enhance significantly spatial resolution in comparison to conventional radars. This allows SAR to create high-resolution images with comparatively small physical antennas [8].

Table 1: Different spectral bands used in SAR

SAR Band	**Frequency range (GHz)**	**Corresponding wavelengths (cm)**
P	0.23-1	130-30
L	1-2	30-15
S	2-4	15-7.5
C	4-8	7.5-3.75
X	8-12.5	3.75-2.4
Ku	12.5-18	2.4-1.67
K	18-26.5	1.67-1.13
Ka	26.5-40	1.13-0.75

Compared to optical RS, SAR has the following advantages [9]:

1. SAR is an active remote sensing method, which makes it possible to collect data despite light and weather conditions. As stated in the previous section and shown in Table 1, C, S, L and P bands can acquire "cloud-free" images in all weather conditions [10].

2. SAR RS is sensitive to the dielectric and geometrical characteristics of the crops. Depending on the frequency, it can obtain information below the vegetation canopy cover.

3. This technology is designed to use different imaging parameters, such as the incident angles and the polarization configurations of the sensor, to obtain further information.

Nevertheless, SAR technology was not thoroughly used until recently because of its low revisit time and scarce quality of spatial resolution, like the $25m \ x \ 25m$ pixel size of the SAR instrument onboard the SeaSat [11]. Since both ESA Copernicus satellites Sentinel-1A and Sentinel-1B (see in Figure 7) are orbiting the Earth, revisit time has hugely improved to 1.5-4 days over Europe and spatial resolution has been upgraded to the order of tens of meters [9, 12]. These specifications fulfill the requirements to enable the extraction of valuable data that can be used by researchers along the whole growing season.

Figure 7: Sentinel-1 mission satellites. Retrieved from: https://sentinel.esa.int/web/sentinel/missions/sentinel-1

The Sentinel-1 C-SAR instrument (see parameters in Table 2) has improved users' capabilities and provides data routinely and systematically for maritime and land monitoring, emergency response, climate change and security. Moreover, it provides a wide range of information for an effective sustainable field management. This is due to the C-Band MW sensibility to vegetation characteristics such as Volumetric soil Water Content (VWC), Leaf Area Index (LAI) and height [3].

Table 2: Key parameters of Sentinel-1 C-SAR instrument. Retrieved from: https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-1-sar/sar-instrument

Parameter	**Value**
Centre frequency	5.405 GHz (corresponding to a wavelength of 5.55 cm)
Bandwidth	0-100 MHz (programmable)
Polarisation	HH+HV, VV+VH, VV, HH
Incidence angle range	20°- 46°
Look direction	Right
Antenna type	Slotted waveguide radiators
Antenna size	12.3 m x 0.821 m
Antenna mass	880 kg (representing 40% of the total launch mass)
Azimuth beam width	0.23°
Azimuth beam steering range	-0.9° to +0.9°
Elevation beam width	3.43°
Elevation beam steering range	-13.0° to +12.3°
RF Peak power	- 4.368 kW, - 4.075 kW (IW, dual polarisations)
Pulse width	5-100 µs (programmable)
Transmit duty cycle	Max 12%, SM 8.5%, IW 9%, EW 5%, WV 0.8%
Receiver noise figure at module input	3 dB
Maximum range bandwidth	100 MHz
PRF (Pulse Repetition Frequency)	1 000 - 3 000 Hz (programmable)
Data compression	FDBAQ (Flexible Dynamic Block Adaptive Quantization)
ADC sampling frequency	300 MHz (real sampling)
Data quantisation	10 bit
Total instrument mass (including antenna)	945 kg
Attitude steering	Zero-Doppler steering and roll steering

2.3 Previous research

As a first instance of previous investigation in the project field, researchers of the Geodesy and Geoinformation department at TUW attempted to assess vegetation variables quantitatively relying on in situ reference data under varying meteorological conditions [3]. Backscatter information supplied by ESA's Copernicus Sentinel-1 mission (similar to Figure 8) and in situ vegetation variables from Austrian fields were examined under different weather scenarios. Linear regression, exponential regression and Machine Learning (ML) models were applied to the aforementioned data, and they were evaluated with the coefficient of determination R^2 and the Root Mean Square Error (RMSE). Regarding the results, VH/VV index resulted to be the best feature to estimate VWC using Random Forest analysis (a flexible, easy to use ML algorithm) for corn and winter cereals. In conclusion, the large potential of MW applied to vegetation monitoring was proved showing satisfactory outcomes.

Additionally, another investigation about the heterogeneity between and within agricultural fields was conducted by three German researchers [4]. In fact, Sentinel-1 satellites SAR performance was tested to detect variability in two test sites in Germany. Besides, the crop types submitted to study were wheat and barley, and in situ measurements during 2017 and 2018 were compared to backscatter. In the analysis, the observed parameters that were related to SAR backscatter parameters were wet and dry biomass, absolute and relative VWC, LAI and plant height. The employed methodology consisted of linear, exponential and multiple regression. In the end, the final results showed that regression performance within fields is especially successful for wheat fields at early stages. For instance, VV backscatter showed R^2 values around 0.7 for all the crop parameters except for relative VWC. Nevertheless, concerning regression between fields, the results were extremely affected by differences in field data quality, incidence angle and precipitation. In conclusion, only when further information in terms of seasons and fields is available will the creation of a global set of valid regression equations to predict crop parameters from backscatter be feasible.

Figure 8: Copernicus image of Flevoland, the Netherlands. This image combines three radar acquisitions from the Copernicus Sentinel-1 mission taken about two months apart to show changes in crop and land conditions over time. The first image from 8th May 2018 is associated with blue, the second from 7 July depicts changes in green, and the third from 5th September has been linked to red. Retrieved from: https://earth.esa.int/web/guest/featured-image/-/article/flevoland-the-netherlands

2.4 Growing cycle of winter cereals

Winter cereals is the term used for wheat, barley and oat. Normally, they are sown in late autumn and harvested in late summer. In general, these crops are set aside for animal feeding and milling in the case of oat.

The principal growth stages using the Feekes scale are described below [13] and illustrated in Figure 9:

1. Tillering: The first phase starts around 40 days after planting the seed and it can last until 120 days [14]. The plant develops a vast underground branching in order to give the crop the necessary number of stalks that lead to an optimum growth. Adequate lighting, temperature and soil moisture will result in active basal vegetative buds, which is crucial for a better crop yield.

2. Stem extension: The next stage begins when the first stem node is visible at the base. The plant performs its greater increase of height and its last leaf gets visible. Generally, it spans until the 140th day after emergence.

3. Heading: The onset of the first spikelet of the head marks the beginning of this period and it ends when the head goes out of the sheath completely. Tipically, this phase's duration is 10 days.

4. Flowering: As the title says, the flowering stage lasts until the bloom is completely visible and the grain is ripe, which tends to be around day 170 after emergence.

5. Ripening: In the final part of the growing stage, the plant achieves the physiological maturity and is ripe for cutting, approximately 180 days after the first leaf appearance.

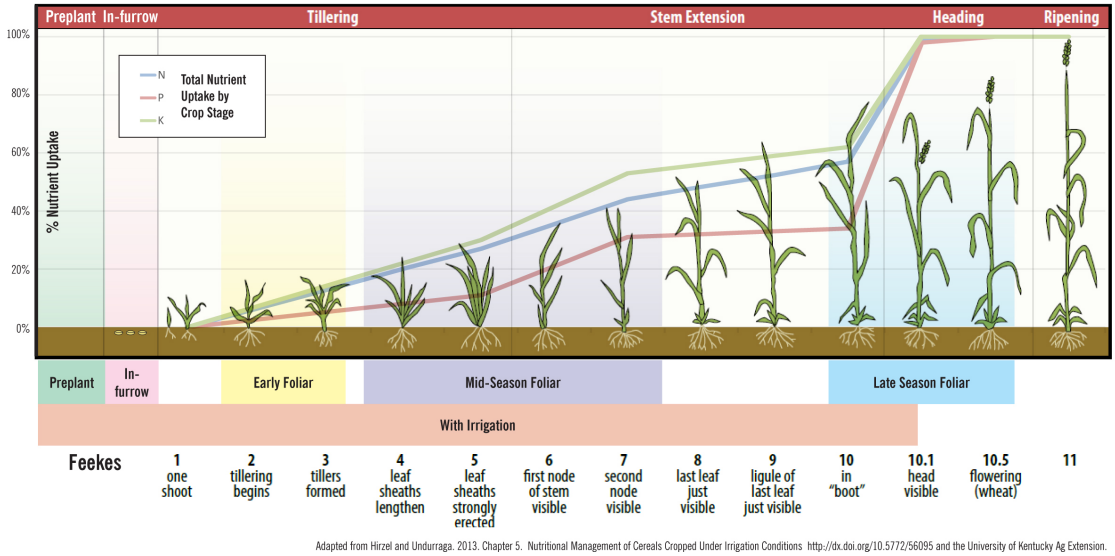


Figure 9: Wheat growth cycle. Retrieved from: University of Kentucky

In this project, the winter cereals submitted to study are wheat and barley. These crops have different sensory and nutritional properties, as it is stated below.

One the one hand, wheat is the second most produced cereal worldwide (see Figure 11). Its grain is a great source of vitamins, protein, and minerals. When it is milled, all nutrients except for the endosperm are removed and flour is obtained. It is the base ingredient of a large variety of food such as bread, cakes or pasta. Also, it is used to elaborate alcoholic beverages and biofuel fermentation. In Figure 10, approximate dates of different stages of winter wheat are illustrated.

Figure 10: Winter wheat growth cycle. Retrieved from:https://balagan.info/

On the other hand, barley is the fourth cereal in the worldwide production ranking (see Figure 11). Nutritionally, it is high in carbohydrates, protein and dietary fiber. It is mainly used for beer and whiskey production as well as human consumption and animal feeding.

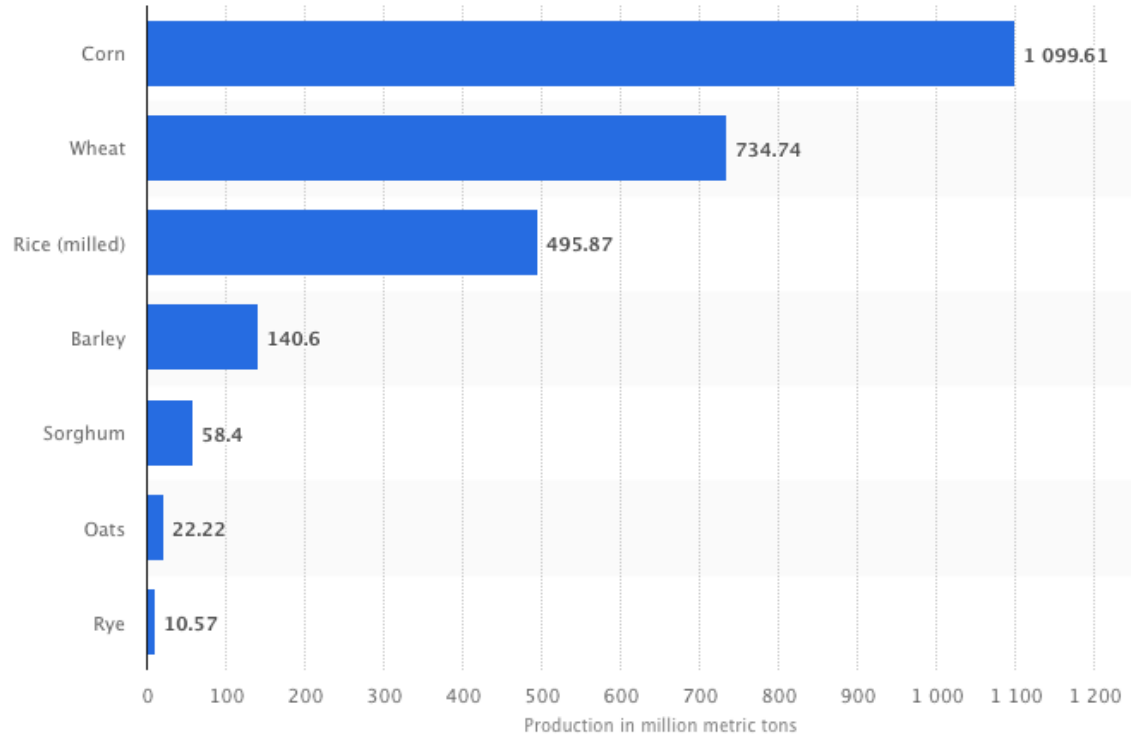


Figure 11: Worldwide production of grain in 2018/19 by type. Retrieved from: https://www.statista.com/statistics/263977/world-grain-production-by-type/

These two winter cereals can seem very similar on the first sight. It is true that their growth stages are pretty alike (Figure 10). Nonetheless, they differ in many aspects that need to be succinctly explained. In the first place, they both belong to *Poaceae* family of plants but, while wheat's genus is *triticum*, barley's is *hordeum*. In the second place, wheat auricles and leaf-sheaths are blunt and hairy whereas barley's are long, slender and hairless [15].

3 Project Methodology

3.1 Data

As mentioned in the previous section, in order to carry out a solid study, it is fundamental to have sufficient information. In this project, the datasets used are Sentinel-1 backscatter, ERA5-Land weather data and Land Parcel Identification System (LPIS) data.

3.1.1 Sentinel-1 data

Sentinel-1 mission offers records of backscatter values (expressed in dB) continuously all over the Earth. To complete the study, information about three complete growing seasons (2015/16, 2016/17 and 2017/18) has been used. The total number of fields of each crop type is different, but the same number of them is always selected to compare them equally. The measurements provided are VV backscatter, (Vertical polarization transmission and reception) VH backscatter (Vertical polarization transmission and Horizontal polarization reception) and CR index (ratio of VH and VV backscatter). Each one of these parameters is divided in two groups depending on the orbit direction of the satellite when measuring: ascending orbit (A) or descending orbit (D).

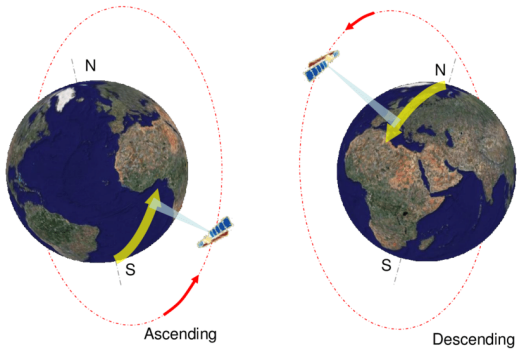


Figure 12: Orbit direction scheme. Retrieved from: https://www.researchgate.net

Moreover, the backscatter data is first averaged over 12-daily periods in order to reduce the effects from different orbits. This is done on the original pixel scale (20 m). Then, statistics are extracted on the field scale in order to reduce spatial effects such as Speckle noise. The following statistics are obtained per field:

- Mean: Average VV and VH backscatter and CR index of every field for a 12-day period.
- Std: The standard deviation of the 12-day period measurements.
- Max: The maximum observed value of the timestamp.
- Min: The minimum observed value of the 12 days.

Nevertheless, min and max were in the end not taken into account as measurements are affected by Speckle noise [16], and did not bring any useful information for the crop classification performed in this project.

3.1.2 LPIS data

To be able to interpret the Sentinel-1 backscatter time series and relate them to growing stages of crop types, reliable reference data is required. Therefore, Land Parcel Identification System (LPIS) data is used, which contains information about fields such as the geospatial reference, the harvested crops of every season or the field geometry. Besides, there is a look up table that links geospatial references to field identification numbers so that the desired fields are selected correctly. The LPIS datasets are freely available for the Netherlands and Austria via the Rijksdienst voor Ondernemend Nederland and the Austrian Open Data portal (https://data.gv.at).

3.1.3 ERA5-Land

In addition, there is land and weather data available using ERA5-Land [17]. This dataset provides hourly high resolution information of surface variables since 1981 at an enhanced resolution with respect to its previous version ERA5. It combines model data with observations from all over the world into a globally complete and consistent dataset. The features used are the following:

- T2M: Average temperature at 2 meters high with respect to the ground for 12-day periods. It is expressed in degrees Celsius.
- TP: Accumulated liquid and frozen water, including rain and snow, that falls to the Earth's surface during 12 days. It is the sum of large-scale precipitation. Precipitation variables do not include fog, dew or the precipitation that evaporates in the atmosphere before it lands at the surface of the Earth. Unit: meters of height filled in a surface of a square meter.
- SWVL1: Average volume of water in soil layer 1 (0 – 7 cm) of the European Centre for Medium-Range Weather Forecasts (ECMWF) Integrated Forecasting System for 12-day periods. The surface is at 0 cm. Unit: cubic meters of water per cubic meter of soil.
- SWVL1: Mean volume of water in soil layer 2 (7 – 28 cm) of the ECMWF Integrated Forecasting System for 12-day periods. Unit: cubic meters of water per cubic meter of soil.

Using these features it is possible to study the effects of weather on backscatter measurements of wheat and barley fields.

3.2 Sentinel-1 backscatter from wheat and barley fields

The first step before elaborating any kind of classification is to know very well the data behavior that might affect the results. For that reason, the first part of the project consists of describing wheat and barley backscatter data time series keeping into account that the crop type is known for every field. Two different graphs (red for wheat and blue for barley) are displayed. The aim of this section is to develop a Python code that makes easier the analysis of the different features' backscatter behavior (absolute and relative VV, VH backscatter and CR index).

For this purpose, the file *Time_series.py* was created (see in the appendix, Page A.1). This code displays backscatter separately in wheat and barley for a whole season by plotting images like Figure 13 :

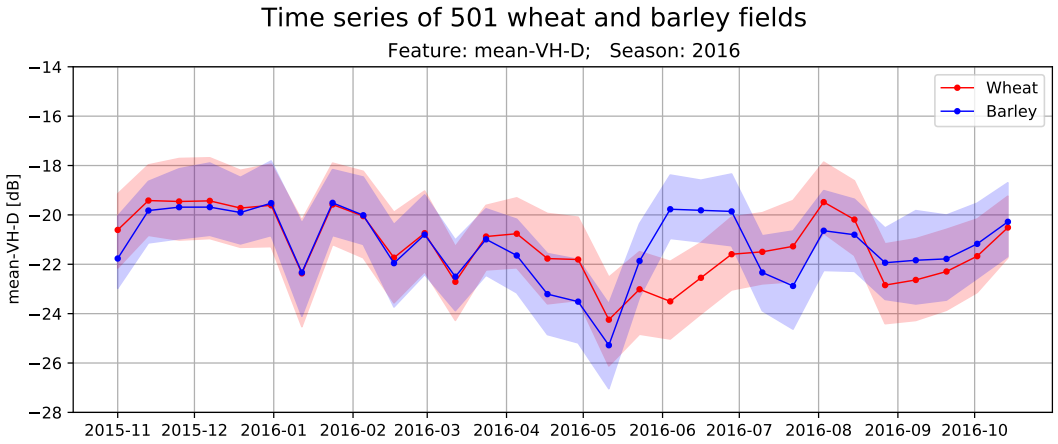


Figure 13: Time series example

Surrounding the curves, there is a shadowed area representing the deviation of the samples, as the half of the total samples lies between the upper and lower limit of each point.

Apart from absolute backscatter values (as it can be seen in Figure 13) it has been developed a normalization option that transforms values of each crop type to a range between 0 and 1, respectively, in order to make easier to compare both crop types. Also, it is possible to normalize a single field time series, where 0 is the absolute minimum and 1 the absolute maximum. It is very clear that applying normalization in that example helps to distinguish both crop types, as it can be seen in Figure 14 during the month of June.

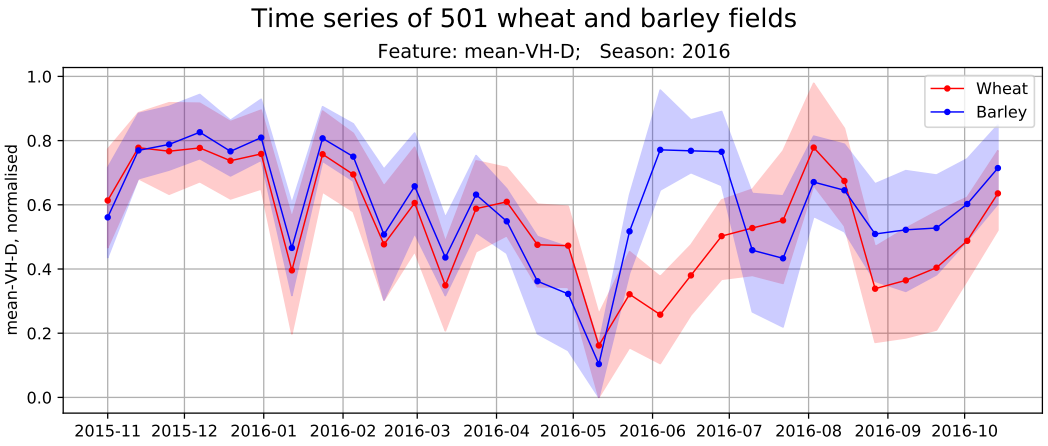


Figure 14: Time series normalization example

Using time series plots like these above, it has been possible to describe VV, VH backscatter and CR index time series tendencies and typical values.

3.3 Weather variability

Once time series were described, it was needed to check if weather affected both crop types in the same way or if it helped to get a better final automatic classification. To study this aspect, the code *Weather.py* (see in the appendix, Page A.2) was written. ERA5-Land features (see Figure 15) were added to the previous time series.

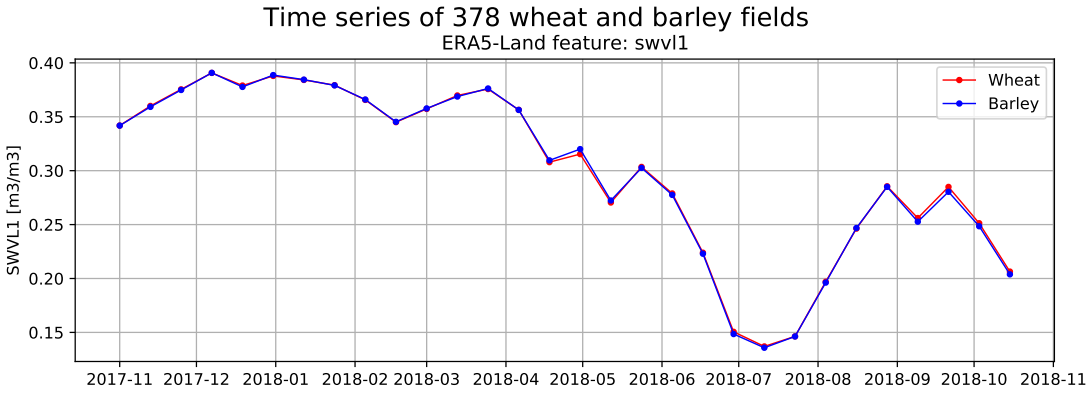


Figure 15: SWVL1 time series for 2018

Also, plots that allowed the comparison of inter-annual variability were used, showing all the available seasons for a feature, as it can be seen in Figure 16

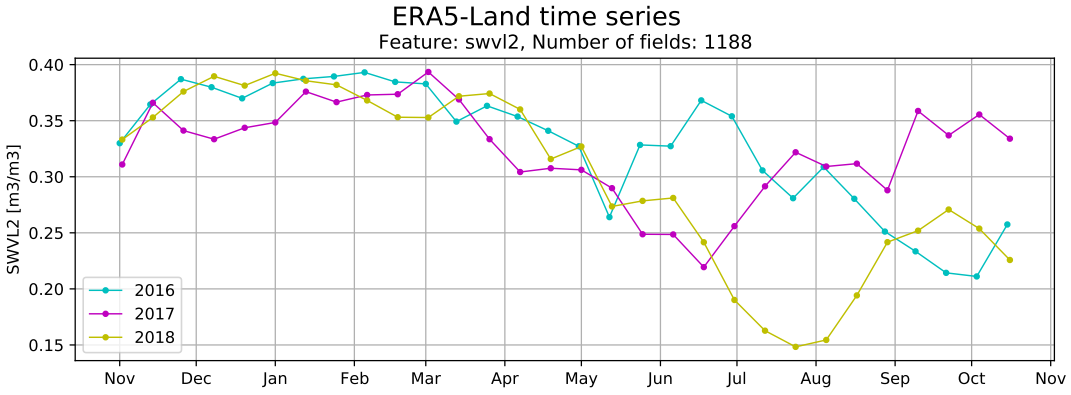


Figure 16: SWVL2 time series for each season

Another type of plots used to visualize Total Precipitation (TP) is bar plots, as shown in Figure 17.

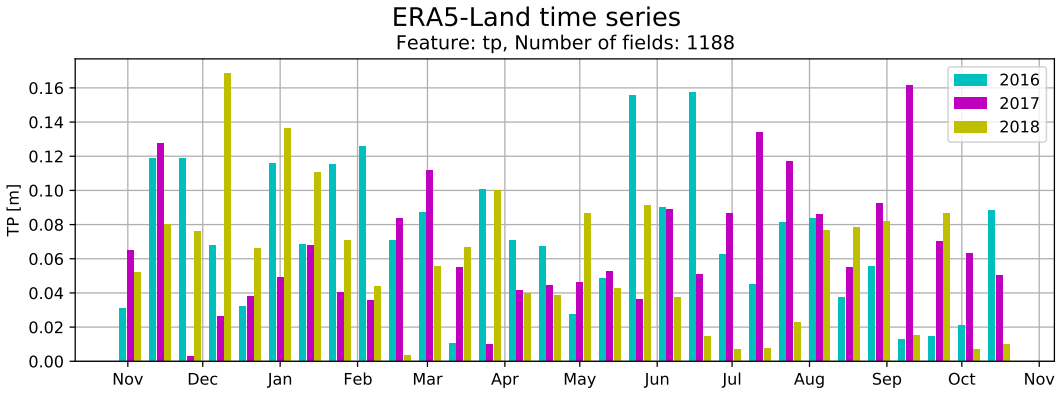


Figure 17: Total precipitation for 2016, 2017 and 2018 seasons

Lastly, scatter plots are used to visualize the relationship between backscatter and weather variables (as can bee seen in Figures 40 and 41).

3.4 Feature identification for the separation of wheat and barley

The suitability of different features used in the separation of wheat and barley was evaluated afterwards. Both are described in the following.

3.4.1 Maximum peak analysis

The first interesting feature that was worth analyzing was the characteristic barley peak in VH and VV backscatter that wheat does not show (see in Figure 14 around June). To be able to visualize better that peak, the code *DoY_analysis.py* (see in the appendix, Page A.3) was created to view in box plots the day of the year when maximum peaks tend to appear given a specific time interval (see in Figure 18). The chosen time interval is from mid May (the 132th day of the year) to mid July (the 192th day of the year). Moreover, If there is not a maximum (timestamp with greater backscatter than previous and next samples), it returns the highest value. This characteristic was called Day of First Maximum (DFM). In the figures that analyze this feature, the box contains half of the total samples and the notch marks the median. Finally, red squares beyond the whiskers are considered outliers due to data distribution.

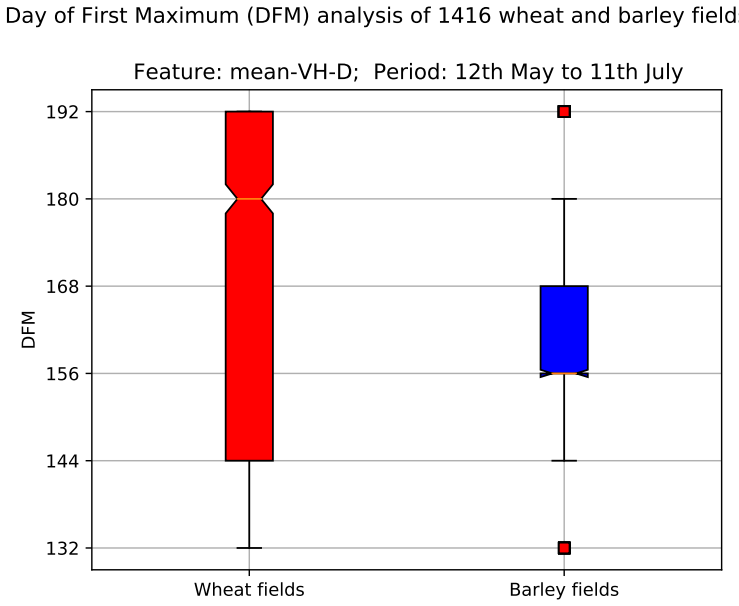


Figure 18: First maximum box plot

In addition, a Python file called *Tseries_max_analysis.py* (see in the appendix, Page A.4) was created to check the procedure and to make the comprehension easier (see Figure 19). To make this analysis, data is always normalized to treat wheat and barley the same way. The diamond markers highlight the detected maximum, which must be higher than the established threshold (e.g. 0.7 in

Figure 19). Eventually, the day when the maximum is detected for every field is the shown data in Figure 18.

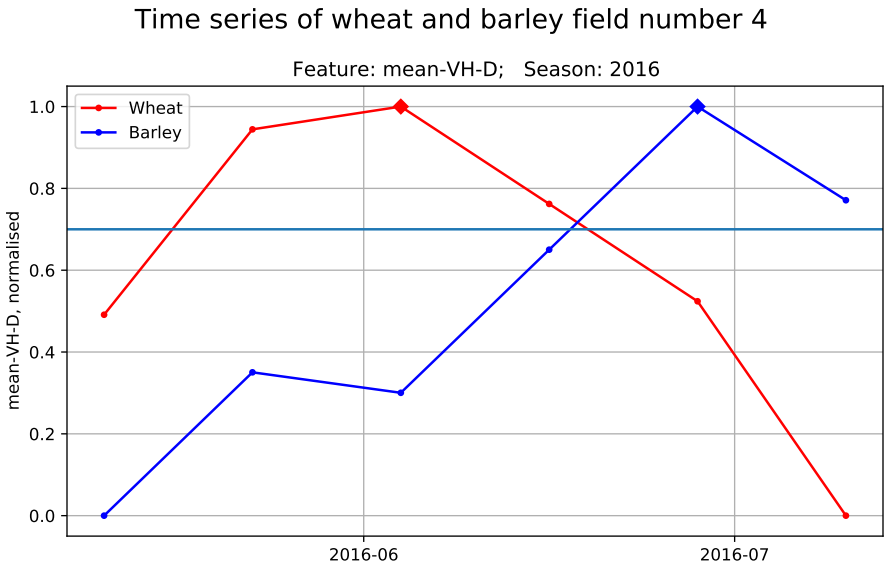


Figure 19: First maximum analysis: Time series of just one wheat and one barley fields. Both maxima are highlighted with a diamond marker.

Another noticeable aspect when looking at VV and VH backscatter time series was the absolute and normalized values difference between wheat and barley maximum peaks (see in Figures 13 and 14). It is clear that the maxima distance between wheat and barley lays in these days. Once the first decision tree was ready, it was proceeded to calculate the ideal thresholds in terms of success rate using loops because the large standard deviation makes it hard to guess the exact values at a glance.

3.4.2 CR index Maximum minus minimum analysis

The other feature that was considered was the difference in seasons 2016 and 2017 in the highest values of the CR index time series, where barley mean is highly variable whereas wheat mean is much more stable, as marked in a green circle in Figure 20.

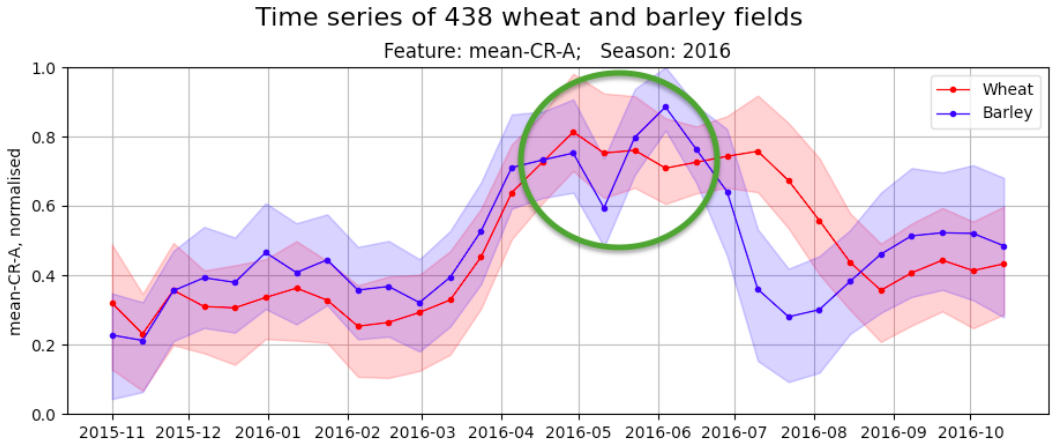


Figure 20: Time series: CR A backscatter 2016

To study the distance between the maximum and the minimum of a time interval for each field, the code *MaxMin_analysis.py* was written (see in the appendix, Page A.5). The data displayed in the box plot in Figure 21 shows the difference in dB between the highest and lowest values given a time interval (from 12th May to 23rd July). In this box plot it is possible to use absolute or normalized values.

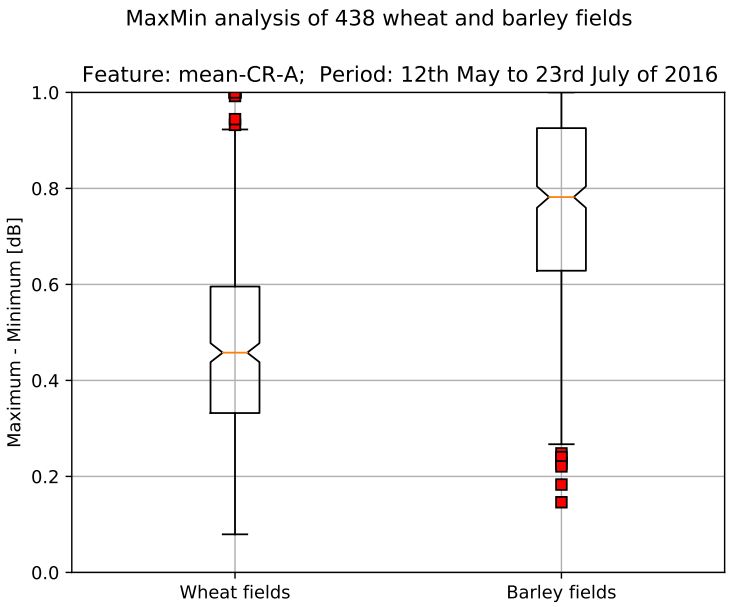


Figure 21: Maximum - minimum example

3.4.3 Automatic classification evaluation

Once the decision trees worked, its performance was evaluated using two metrics:

- Evaluation Rate (ER): It is needed to leave an indecision interval due to the high standard deviation of backscatter that provokes that wheat and barley mixes in some ranges and this it must be evaluated somehow. ER is calculated as the number of wheat and barley fields that can be classified divided by the total number of wheat and barley fields. It is expressed in percentage.

$$ER(\%) = \frac{ClassifiedFields}{TotalFields} * 100$$

- Success Rate (SR): It is the number of wheat and barley fields that are classified correctly divided by the number of wheat and barley classified fields. SR is needed to calculate the error rate, and consequently, the accuracy of the classification. It is expressed in percentage.

$$SR(\%) = \frac{CorrectlyClassifiedFields}{ClassifiedFields} * 100$$

Then, the aim of the project is to maximize those two rates by managing features and thresholds.

4 Results

4.1 Description and interpretation of typical yearly backscatter time series

In this section, the most representative shapes and values of the VV, VH backscatter and CR index are described.

4.1.1 VV backscatter

As it can be seen in Figure 22, high VV backscatter values around -14 dB for both crop types occur in the first part of the growing season (until March), and then a constant drop until May to -20 dB appears. In this drop, wheat keeps 1 dB to 2 dB higher than barley. Again, barley grows steeper than wheat during May and they are paired together once more from July on around -15.5 dB.

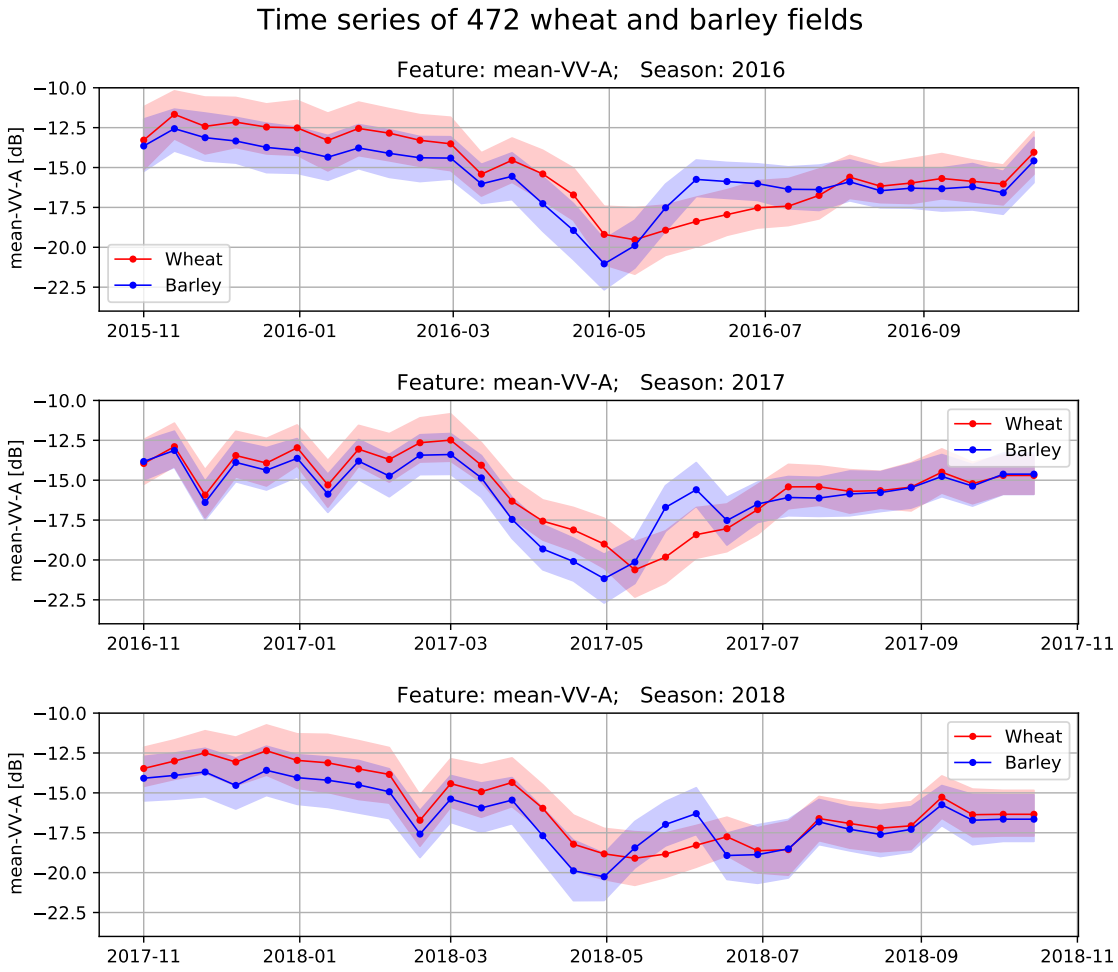


Figure 22: Time series: VV A backscatter for seasons 2016 to 2018 of Dutch fields

As it can be seen in Figure 23, similar shapes to in Figure 22 appear. VV D Backscatter values around -13 dB for the first third of the season and then a great drop until mid May, when both crop types get their minimum at -20 dB. Once again, wheat is 2 dB higher for the first half of the season.

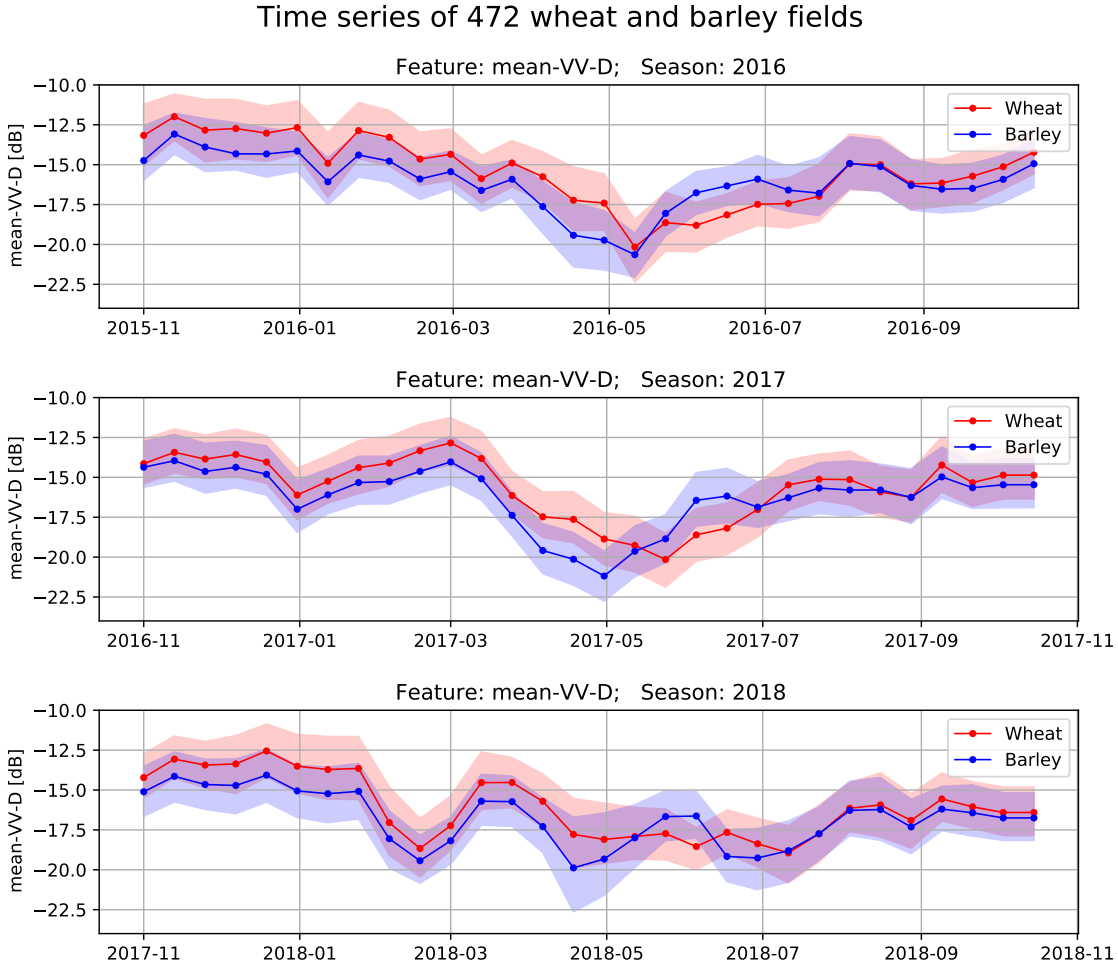


Figure 23: Time series: VV D backscatter for seasons 2016 to 2018 of Dutch fields

In general terms, VV A and VV D backscatter are similar in terms of shape and absolute values. Moreover, they have not several great peaks during the season and a wide "V" shape is seen for both orbit directions.

4.1.2 VH backscatter

However, in VH A backscatter in Figure 24 there is not a clear shape along the whole season. It starts around -20 dB until mid March. Then, it descends during two months until getting its minimum in mid May (-24 dB) and shows a rise until July when it stabilizes, like in VV backscatter (see Figures

22 and 23). The absolute values of wheat and barley are very similar except between April and June, when the barley peak (-19 dB) appears.

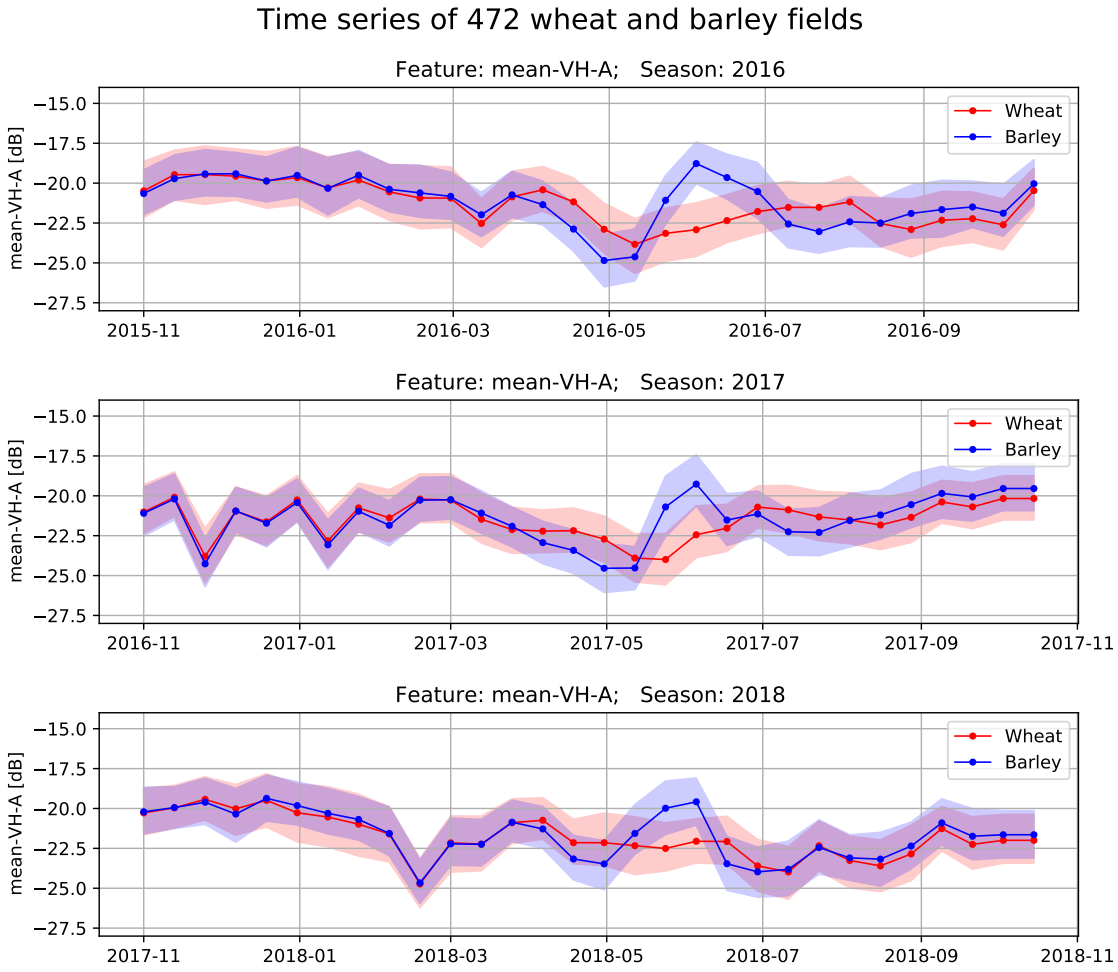


Figure 24: Time series: VH A backscatter for seasons 2016 to 2018 of Dutch fields

For VH D backscatter (see Figure 25), more stable values around -20 dB can be seen until February, but the descent until mid May (-24 dB for wheat and -25 dB for barley) is not as smooth as in VH A backscatter (see Figure 24). After that, there is a barley rise until June (-20 dB) and wheat shows the same peak two months later. Again, both curves get together in the last part of the season around -22dB.

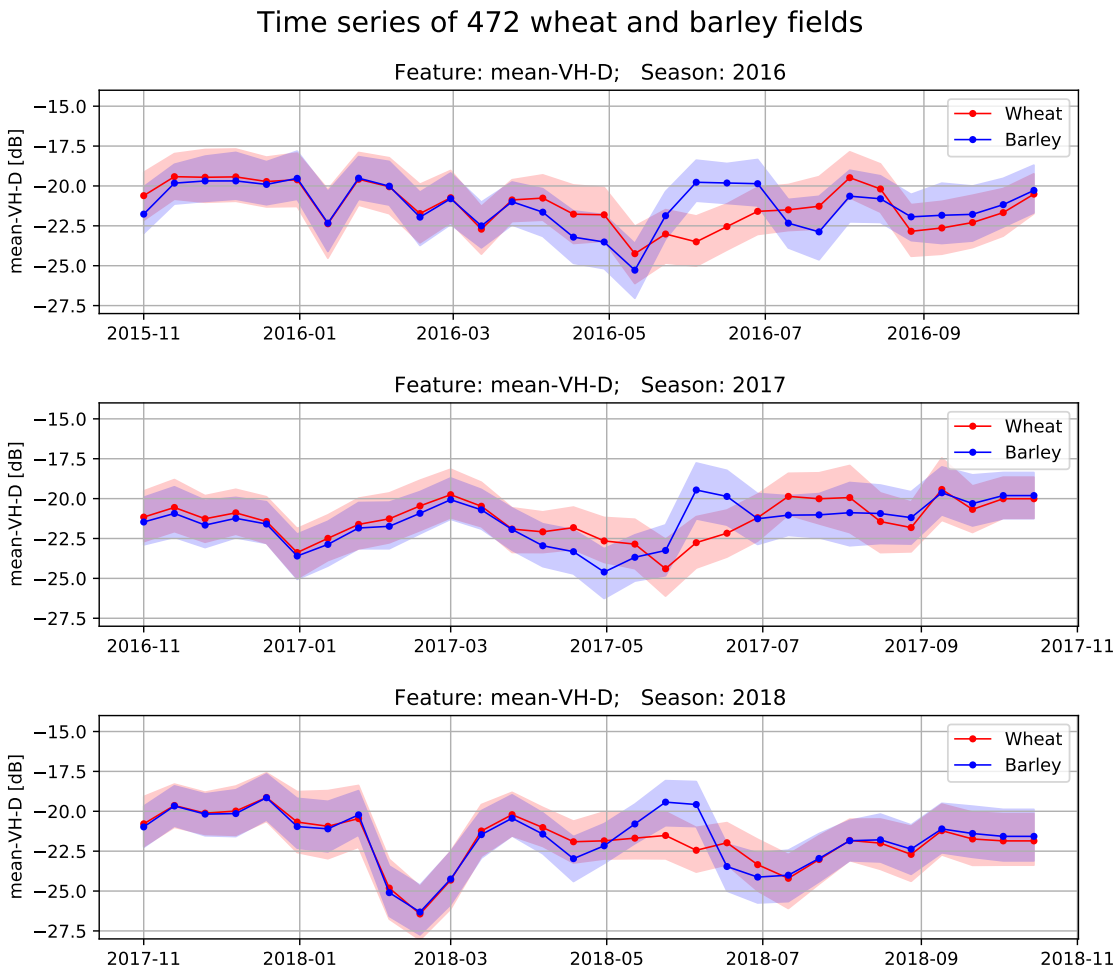


Figure 25: Time series: VH D backscatter for seasons 2016 to 2018 of Dutch fields

Both VH A and VH D show similar absolute values and shape. In general, there is not a clear tendency except for the maxima during summer in which there is a clear difference between wheat and barley.

4.1.3 Cross ratio CR (VH/VV)

When looking at CR A index graphs (like in Figure 26) it is generally observed that wheat starts constantly around -8 dB until the start of the spring, when it rises until -5 dB. From the beginning of this season until May, barley tends to be 2 dB higher and it has the same shape. On the one hand, wheat keeps flat until mid July, when it goes down during 90 days when it stabilizes again in -7dB. On the other hand, barley shows an "M-shape" from May to July. In fall, both crop type recover the same shape and values as in the first part of the season.

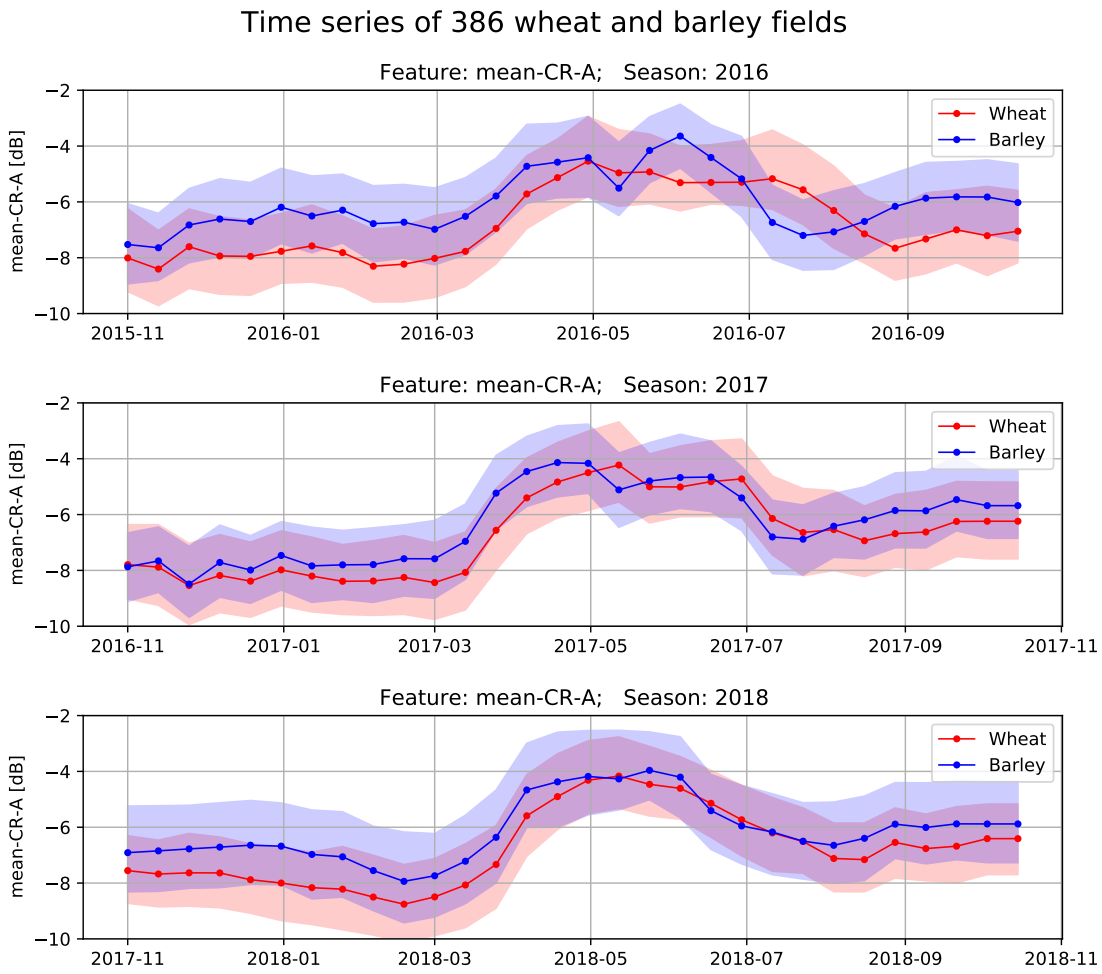


Figure 26: Time series: CR A index for seasons 2016 to 2018 of Dutch fields

In Figure 27, the same shape as in Figure 26 can be seen. In this case, there is a minimum of both crop types around February (-8 dB) and then they both go up during the springtime again, as in CR A index orbit direction. However, this feature generally shows flatter curves during May (wheat at -5 dB and barley at -4 dB). After, both crop types show a smooth drop until August, and they stabilize at -6 dB.

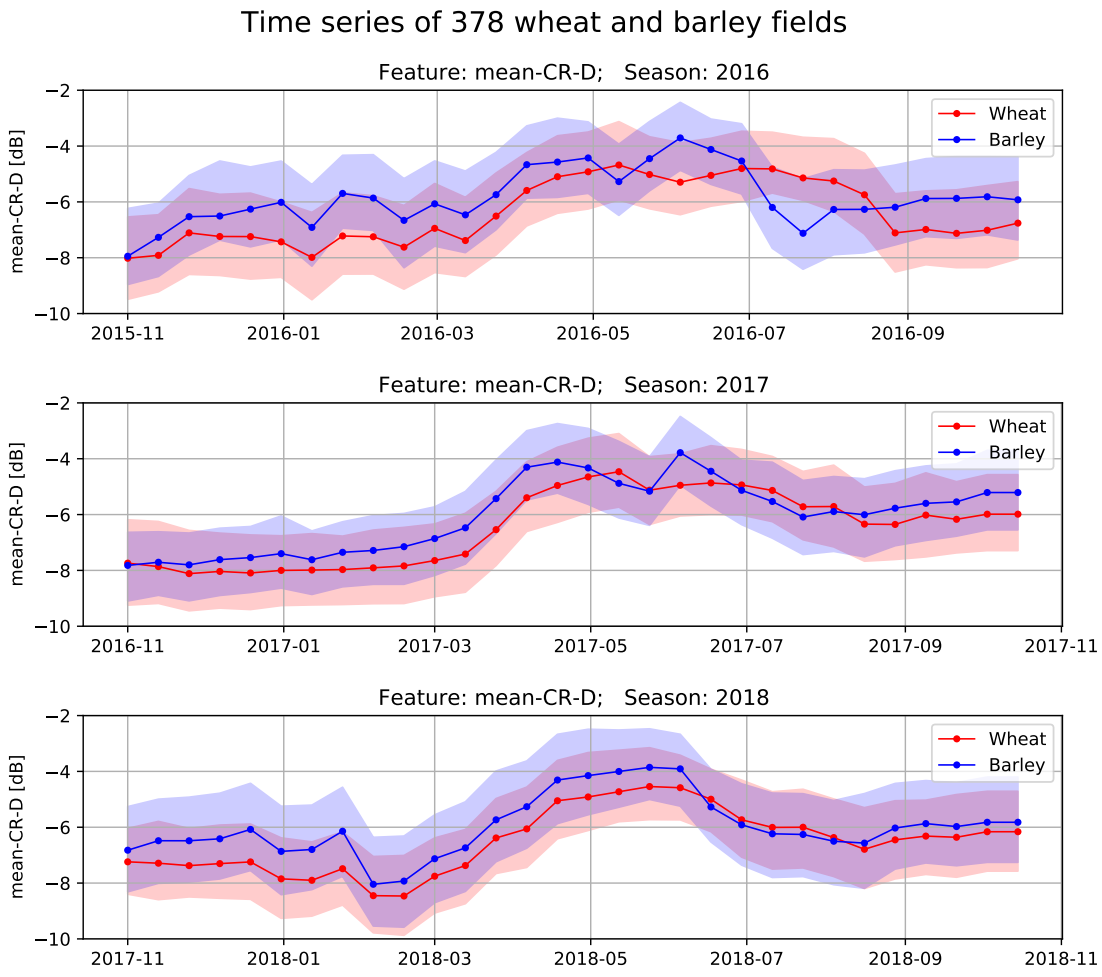


Figure 27: Time series: CR D index for seasons 2016 to 2018 of Dutch fields

In general, both orbit directions increase until May and then decrease after July, as it is the result for $VH[dB] - VV[dB]$. The shape is very similar for all seasons. Both standard deviations are high, they are overlapped for nearly the whole growing stage, which is a great drawback to make a reliable decision tree.

4.2 Differentiation and interpretation between seasons

The point of this section is to find out relationships between ERA5-Land data and backscatter time series. This will provide us valuable information for the future design of the automatic decision tree.

As it can be seen in Figure 28, both crop types are randomly distributed all over the Netherlands without following any pattern, there are no specific zones for wheat neither barley. This fact tells us that weather features will probably be very similar for both crop types, and this is confirmed in Figure 29.

Figure 28: Image of Zeeland, the Netherlands. Wheat fields of 2016 in red and barley fields of 2016 in blue. Retrieved from: Google Earth

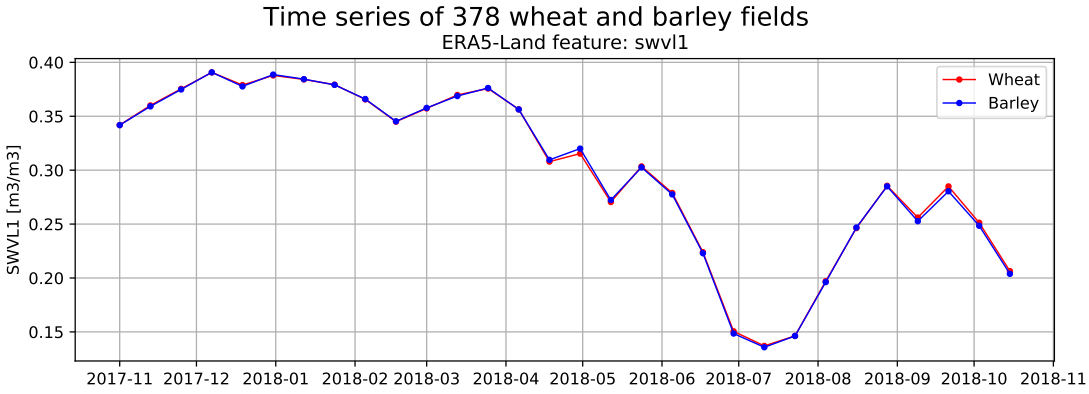


Figure 29: Time series: SWVL1 2018 of Dutch fields

Regarding the identical ERA5-Land values for wheat and barley, it has been considered proper just to analyze one of them for each season and feature.

4.2.1 Temperature at 2 meters high

The recorded temperatures for all the available seasons (see in Figure 30) were generally alike although 2016 winter was warmer than usual and 2018 showed some interesting peaks in February, April and July. Therefore, they were analyzed in more detail.

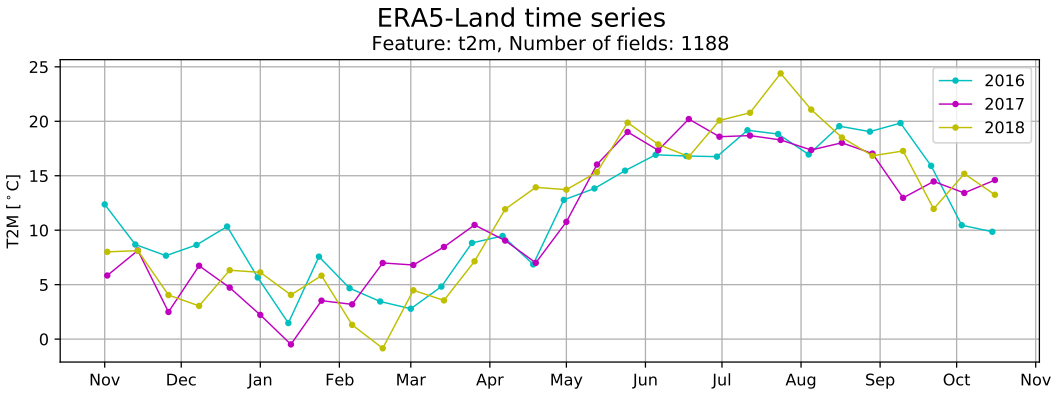


Figure 30: T2M time series for each season of Dutch fields

After looking at time series of all seasons, some variations that might be linked were distinguished. The first one is the sudden drop of backscatter for both wheat and barley that appears in mid 2018-02 (see in Figure 32) and meets a drop of temperature that does not appear in 2016-02 neither 2017-02 (see Figures 22 and 30). Moreover, VV barley backscatter rises accordingly with temperature increases, and wheat takes at least a month to catch barley levels, as it is shown in the big rise in mid May in VV backscatter and VH backscatter Figures 22, 23, 24 and 25.

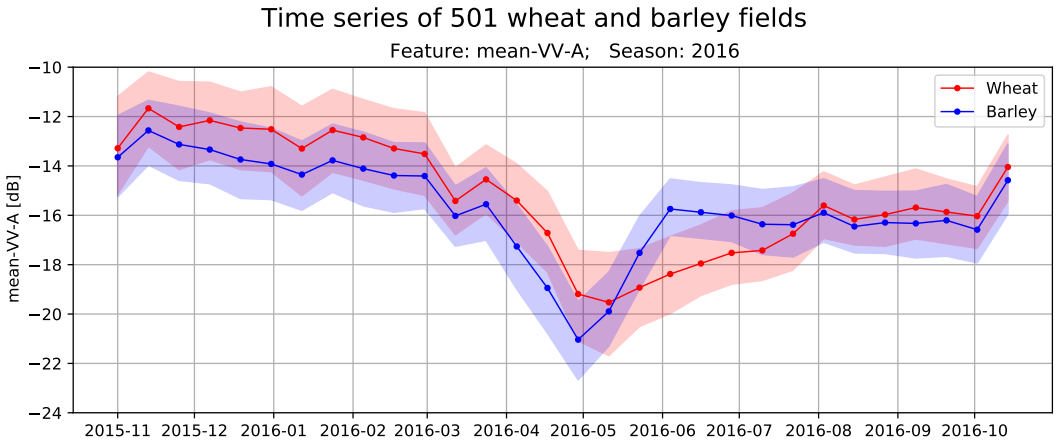


Figure 31: Time series: VV A backscatter 2016 of Dutch fields

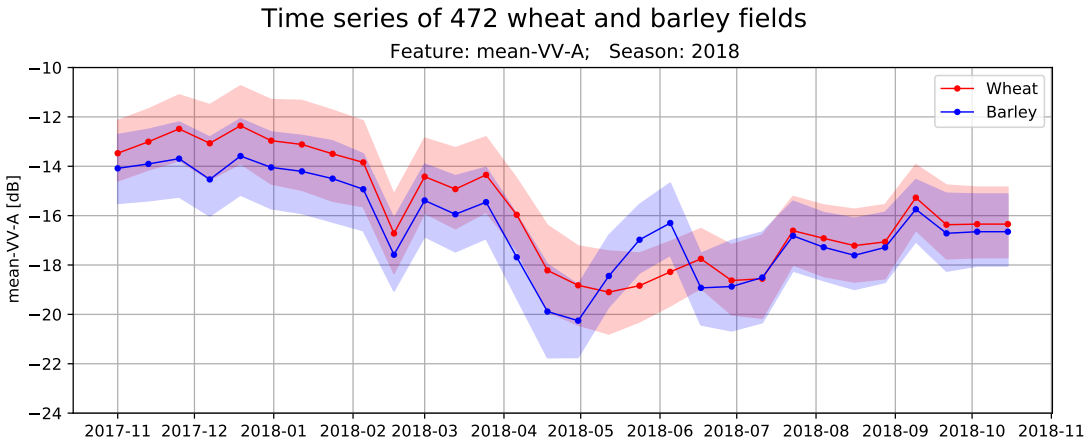


Figure 32: Time series: VV A backscatter 2018 of Dutch fields

4.2.2 Total precipitation

Having look at total precipitations graphic along the seasons in Figure 33, there is a great variation between years except for April, when the three seasons show low values of precipitations. All features had been visualized to find relationships between weather features and backscatter.

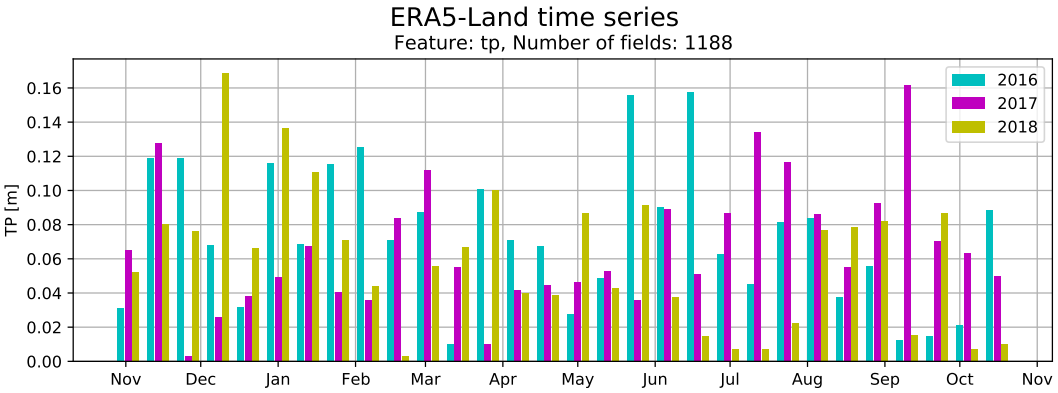


Figure 33: TP time series for each season of Dutch fields

During December and January 2017, when precipitation values were really low, VH and VV backscatter values in early 2017 were really steep and showed some minima (until -24 dB) when compared to other seasons (example in Figures 35 and 34). Moreover, in February 2018 there was almost no rain and minima showed again (see in Figure 36).

In addition, the higher total precipitation was during May and June, the higher was the difference between wheat and barley.

The last remarkable aspect was that VV and VH backscatter showed slightly lower values for 2018 summer because that time of the year total precipitations were very low.

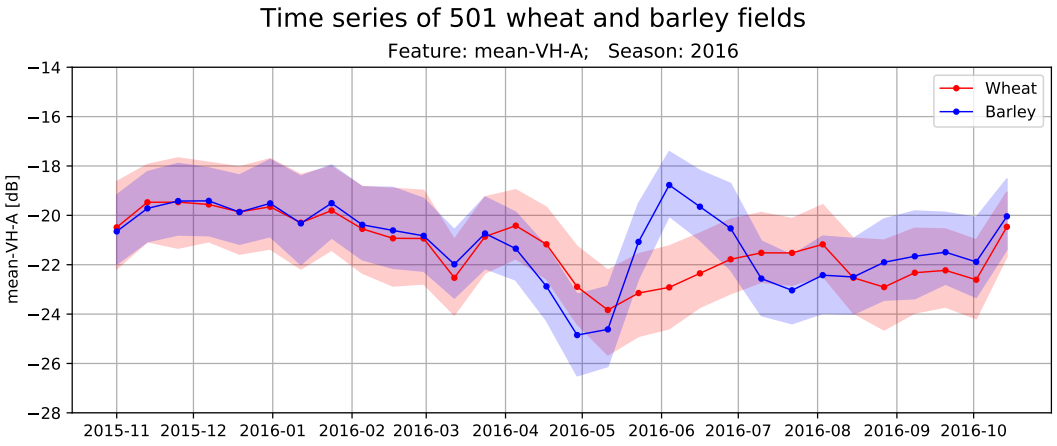


Figure 34: Time series: VH A backscatter 2016 of Dutch fields

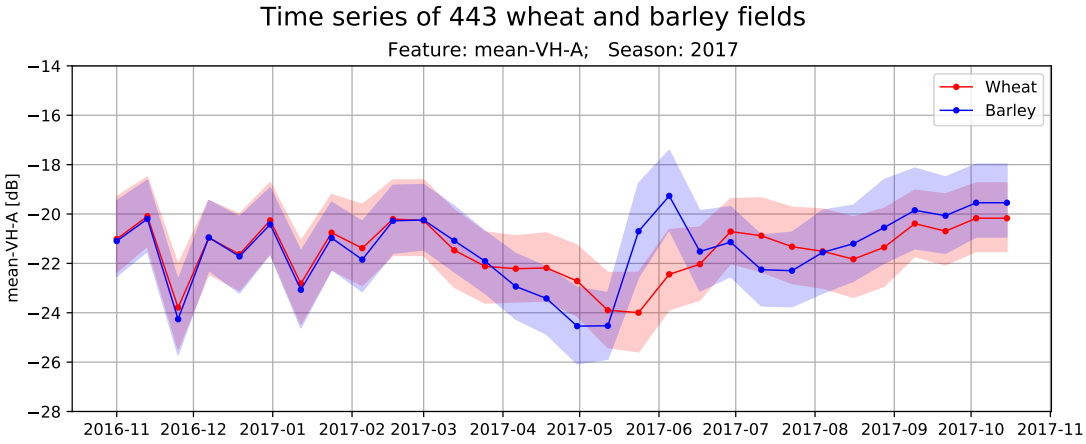


Figure 35: Time series: VH A backscatter 2017 of Dutch fields

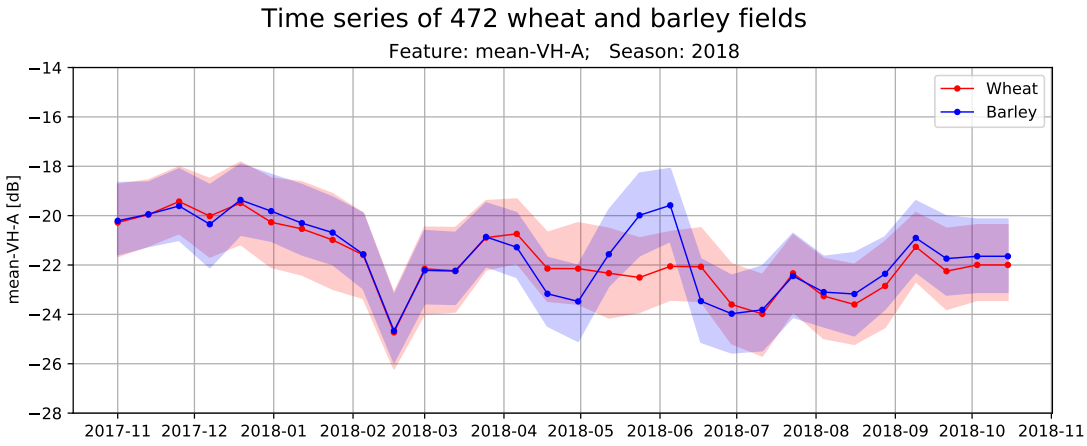


Figure 36: Time series: VH A backscatter 2018 of Dutch fields

After observing those time series, VV and VH backscatter values increases are linked to total precipitation rises. Apart from that, the higher the rain growth is, the bigger the barley VV and VH backscatter peak is during May.

4.2.3 Soil water volume

The first noticeable aspect when looking at both layers of SWV (Figures 37 and 38) is that they were almost identical. Therefore, it was agreed to only focus on SWVL1. Also, what is seen in the first half of both features is that all season have similar values, whereas after May these similarities are lost.

When looking at Figure 37 generally, it is observed a smooth "V" shape already seen in VV and VH backscatter time series, except for summer 2018, in which the lack of precipitation (see Figure 33) provokes a drop in SWV.

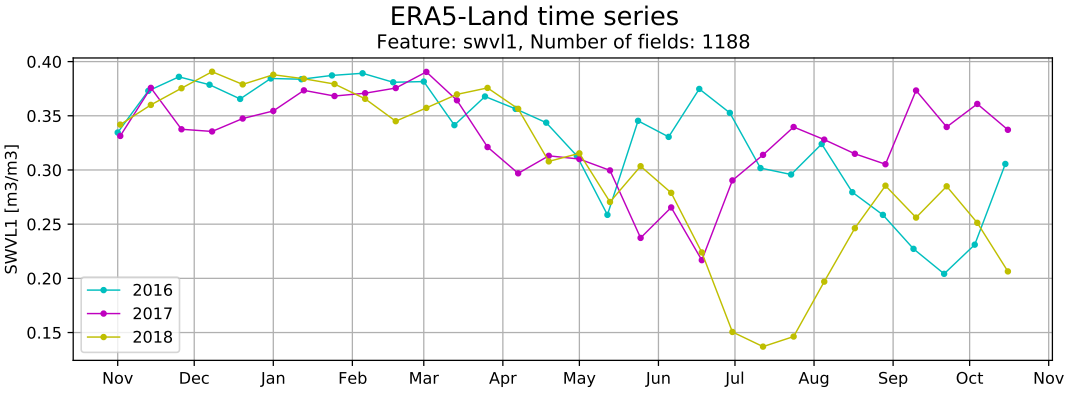


Figure 37: SWVL1 time series for each season of Dutch fields

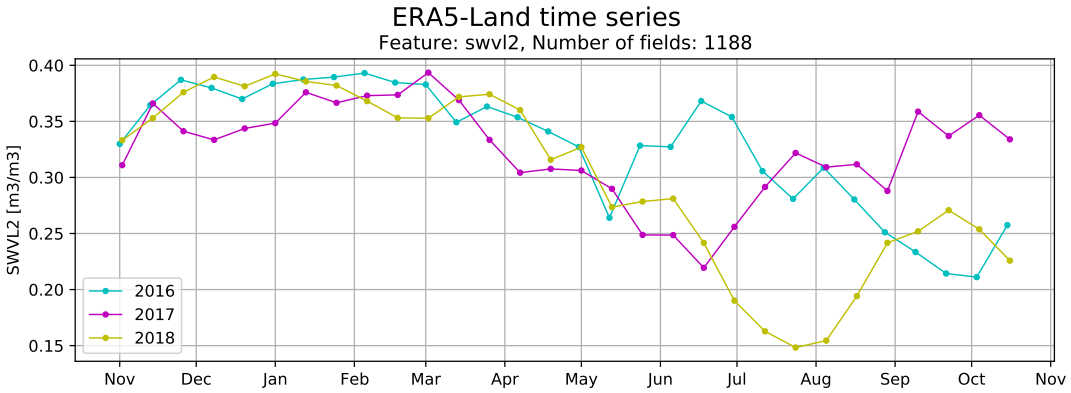


Figure 38: SWVL2 time series for each season of Dutch fields

Once all seasons were studied, it was noticeable that VV and VH backscatter (especially VH D for 2017) is very dependent on SWV, as stated in Figure 39. First, in general shape they are very similar and if focusing in more detail, it is seen that drops in March and June and peaks in July and September match almost exactly.

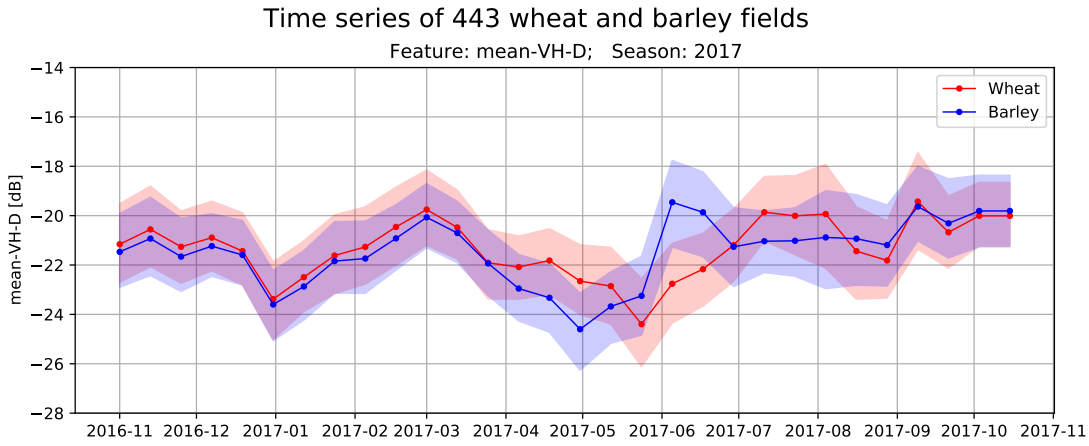


Figure 39: Time series: VH A backscatter 2017 of Dutch fields

After analyzing all the ERA5-Land features and seeing the behavior throughout the seasons, it is clear that VV and VH backscatter is strongly dependent on the environment situations, especially SWV at the last part of the season. However, in order to provide key information for deciding between wheat and barley, this type of data does not show a huge variation between crop types, except for the fact explained below Figure 36

Below (Figures 40 and 41) there are two scatter plots of wheat and barley that summarizes the relationships between VV, VH backscatter and CR index with ERA5-Land data.

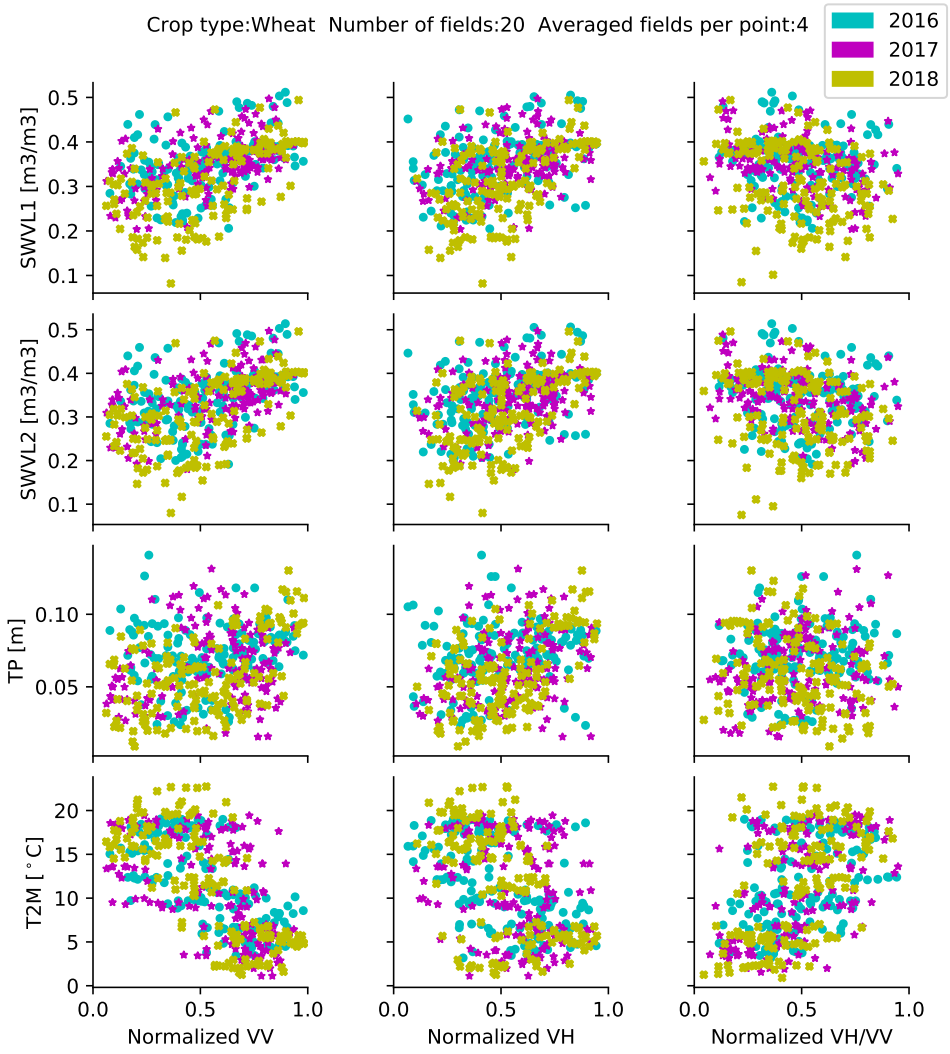


Figure 40: Scatter plot of normalized wheat backscatter with respect to weather features of Dutch fields

Some tendencies can be seen in Figure 40. There are positive linear relationships between VV and VH with soil moisture and negative with T2M. When looking at CR, it is possible to see a positive linear relationship with T2M.

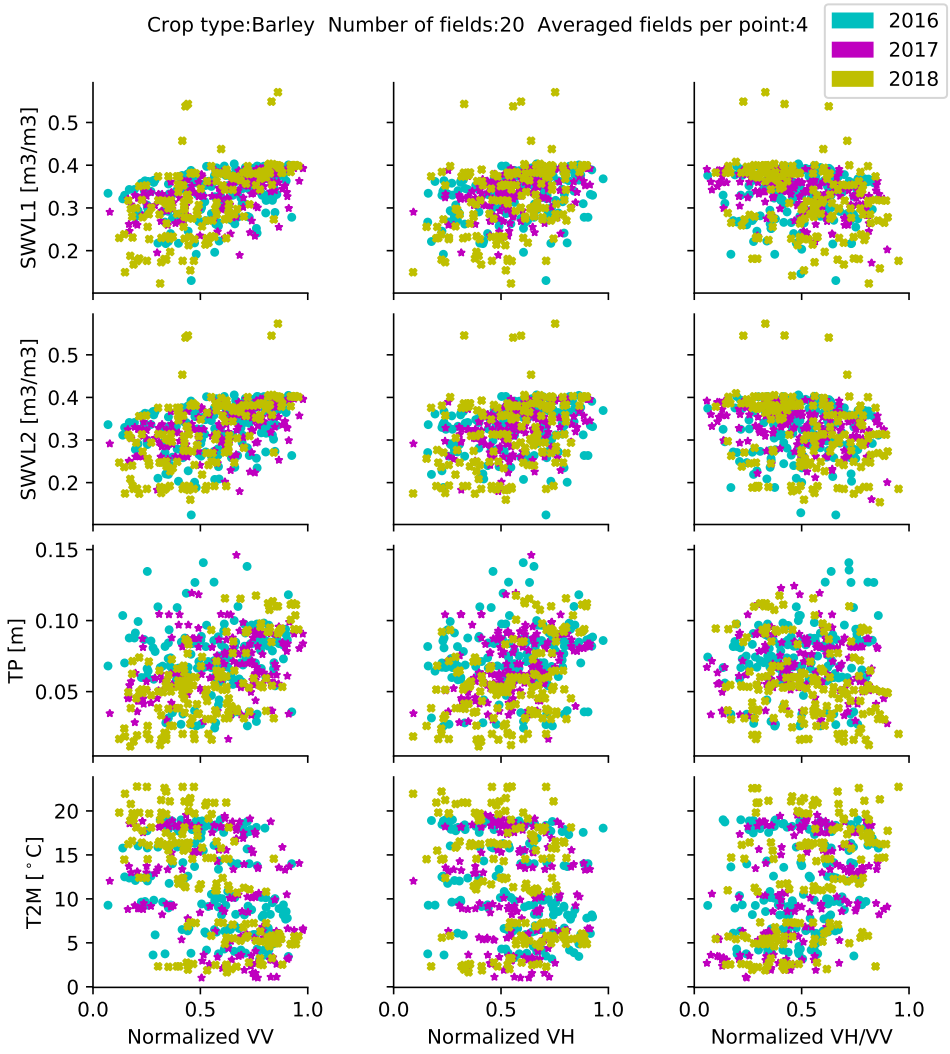


Figure 41: Scatter plot of normalized barley backscatter with respect to weather features of Dutch fields

However, when looking at barley scatter plot 41, those relationships fade away or do not appear as clear as in wheat.

4.3 Physical interpretation of backscatter time series

The observed VV and VH backscatter variation and consequently CR index changes can be explained keeping into account vegetation structural variations and water content. Until March, wheat and barley crops are a few centimeters high (as it is shown in Figure 10), because crops are developing multiple tillers in this stage [4]. Therefore, soil is the main factor affecting backscatter. From March until May, crops start to grow up, and their leaves attenuate the signal significantly [18], as it is clearly shown in all VV and VH backscatter figures, especially in Figure 22. Even though both VV and VH get attenuated by the vertical vegetation elements, VH backscatter decreases less strong because volume scattering (the scattering occurring when EM radiation changes of medium) is stronger, and that is what provokes the rise in CR index in Figures 26 and 27 By mid May, the plant has already developed its flag leaves that cause the maximum attenuation. At this time, all types of backscatter are dominated by plants and not by soil conditions.

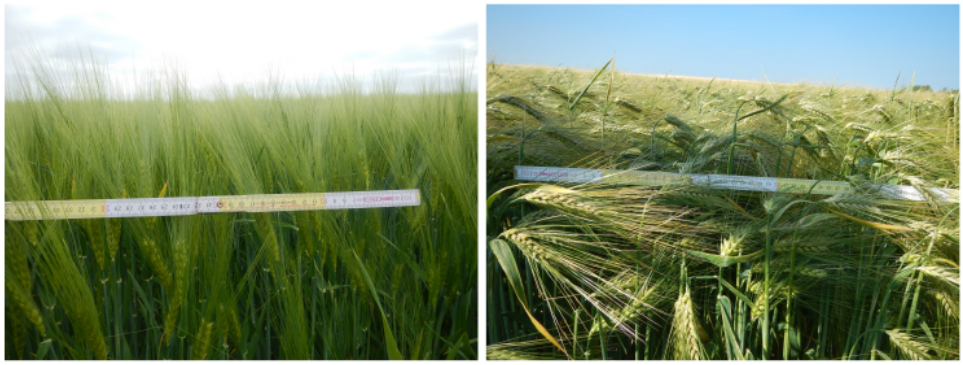


Figure 42: Barley field at 18th of May 2018 with vertical awns (left) and barley field at 6th of June 2018 with horizontal awns (right). Retrieved from: [4]

Around mid May, **the sudden barley rise appears after the ending of the heading stage (mid May), and it is caused by the ears and awns growth and its posterior bend from vertical to horizontal position [4]**, which reduces attenuation significantly. Wheat crops, however, tend to bend after the late ripening stage (around July), when VV and VH wheat backscatter achieves barley backscatter levels.

After that, VV and VH wheat backscatter keeps growing slightly because of the bending and maturity of heads until the beginning of August, whereas barley starts to dry and get a golden color, and consequently, its ground contributions increase. As they are both winter cereals, soil properties are alike, and that is why measurements resemble from mid August until the end of the seasons. After that, VWC gets crucial because is strongly linked to VV and VH backscatter of the last part of the growing season for both crop types, as it is noticeable looking at Figures 22, 23, 24, 25 and 37, when the higher fields water content was, VV and VH backscatter was higher.

4.4 Identification of features based on characteristic differences between wheat and barley

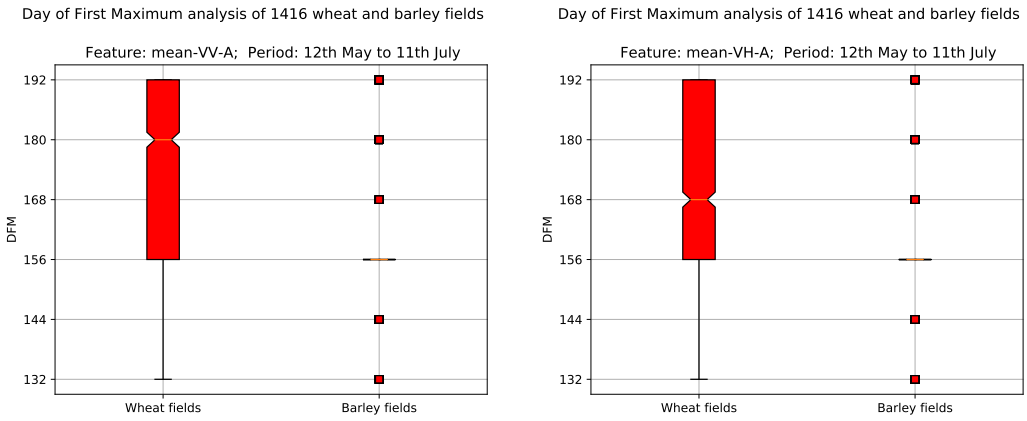
After making a full physical interpretation of wheat and barley backscatter time series and the weather impact, it was proceeded to study the potential regions of difference to select for the decision tree. The candidate regions are explained below.

4.4.1 Day of year of the maximum peak in VV and VH backscatter

When studying backscatter time series, the most obvious difference between wheat and barley that was observed was the barley peak around May-June. The characteristic differences were the amplitude distance and the time differences between maxima (as wheat also achieves a maximum but around a month later). The appropriate way to analyze this was using the time variations, as the backscatter standard deviation is not good.

What was done for visualizing this characteristic was to develop a code that lets one see the difference of time when the first maximum of the selected time interval in box plots. Besides, it was considered that using normalized data was suitable to choose a common threshold to consider maxima equally for both crop types. The data in box plots are the timestamps of the highlighted diamonds for each field and every season, as it can be seen in Figure 19.

As it is clear in Figure 43, for VV and VH backscatter ascending orbit direction, barley tends to have its peak almost always on the 156th day of the year (first week of June), whereas wheat has a more spread distribution and tends to have its maxima later around day number 180 (last days of June) which is deductible regarding VV A and VH A backscatter time series (Figures 22 and 24).



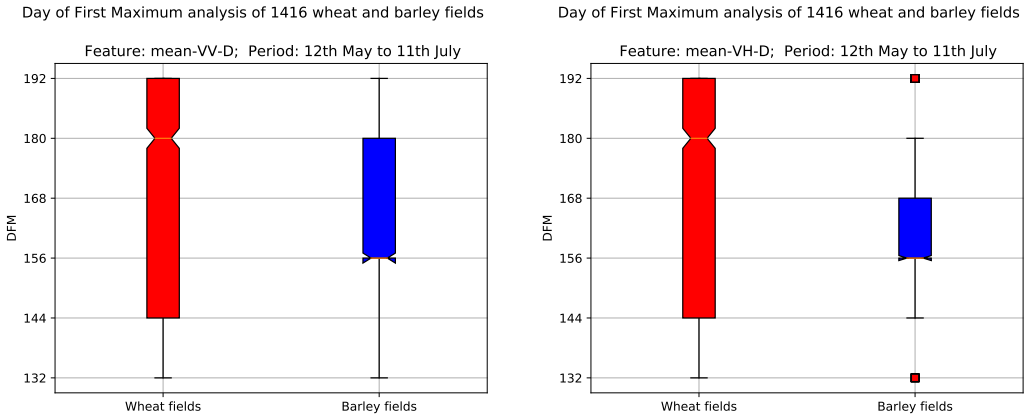
(a) VV A backscatter

(b) VH A backscatter

Figure 43: Maximum peak day box plots of Dutch fields

However, for descending orbit direction (Figure 44), barley maxima come a bit later and wheat

maxima show even before than in both images above. Unfortunately, wheat results are worse than expected due to the high standard deviation. Even though wheat and barley show a very different distribution, they are coincident in time.



(a) VV D backscatter

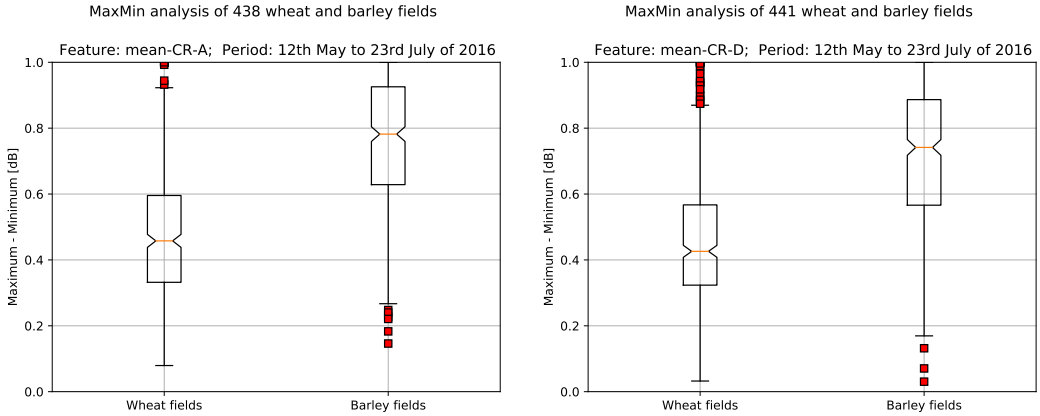
(b) VH D backscatter

Figure 44: Maximum peak day box plots of Dutch fields

4.4.2 CR index maximum minus minimum

To study this feature it was interesting to visualize data in box plots and try to set thresholds to classify wheat and barley. Moreover, it was decided to use normalized data because it helped to separate crop types, and the time interval was extended until mid July to include the great drop that barley presents and wheat does not.

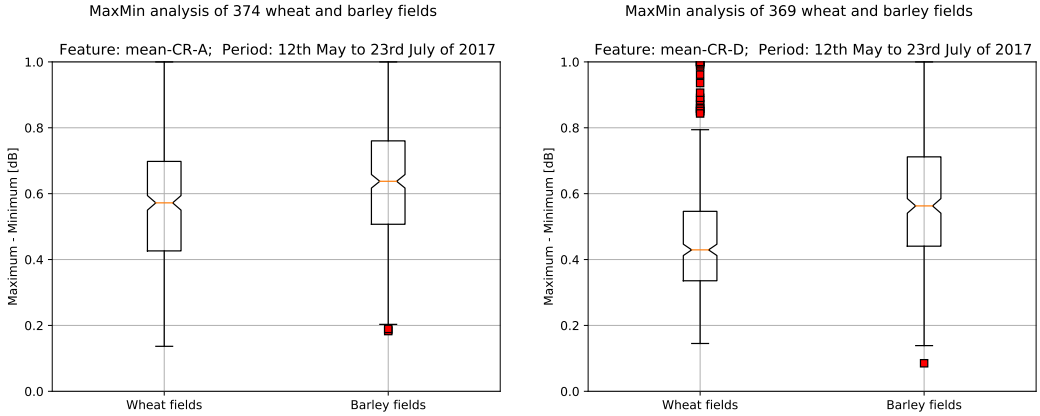
In Figure 45, it is seen a significant difference because a great part of barley fields present their highest and lowest values in this interval, whereas barley fields get more stable values. Approximately 75% of wheat fields present a max-min distance of less than 0.6, whereas the same amount of barley fields are above this value.



(a) CR A index (b) CR D index

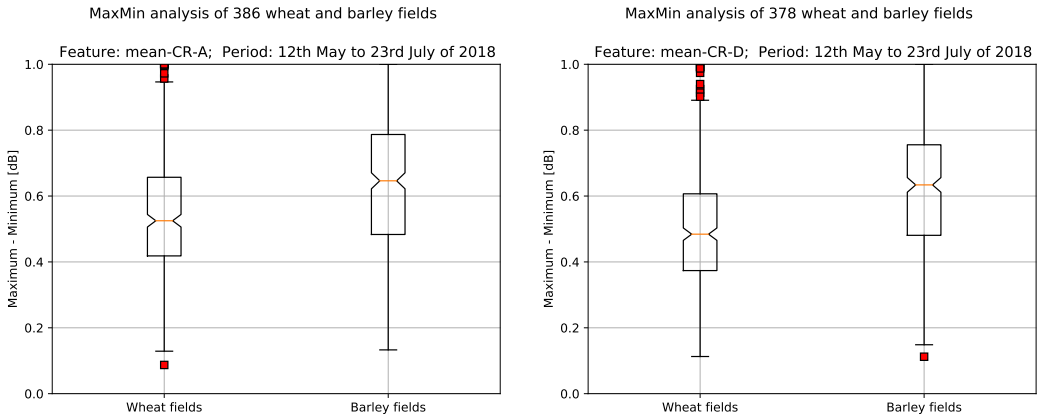
Figure 45: Box plot: CR index 2016 of Dutch fields

However, as it can be seen in Figures 46 and 47, 2017 and 2018 seasons are very similar, and they do not show that clear difference seen in Figure 45. In this case, wheat keeps giving similar values within the time interval and barley does not show those intense variations anymore. Nevertheless, barley always present higher values than wheat for every season and feature using this technique and it is a feature that can be implemented in the decision tree.



(a) CR A index (b) CR D index

Figure 46: Box plot: CR index 2017 of Dutch fields



(a) CR A index

(b) CR D index

Figure 47: Box plot: CR index 2018 of Dutch fields

4.5 Implementation of the decision tree

Once potential features and illustrative thresholds to use in the decision tree were selected, an automatic classification system was developed that permitted checking the reliability of the used features. Simple schemes of possible decision trees were drawn to test, using different features and limits. Afterwards, the usage of other techniques was applied to give consistency and extra accuracy.

4.5.1 First test

Figure 48 shows the first option that was considered. This decision tree only uses the day of year of the maximum peak in VV and VH backscatter between mid May and mid July and decides only using a threshold at 159th day of the year based on the box plot in Figure 43. Notice that available timestamps are calculated every 12 days and in this case, the threshold day could be between the 156th and 167th day and the results would remain the same.

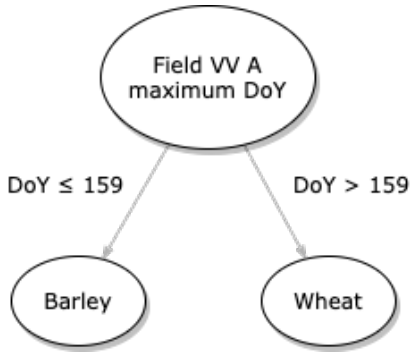


Figure 48: Decision tree: First option

The performance of the decision tree for field from the Netherlands is described in Table 3. These results show a good performance in terms of accuracy in 2017 and 2018, especially in 2017 VV A backscatter and 2017 VH A backscatter features (86.34% and 85.33%, respectively). VV D backscatter is the feature that gives the worst percentages.

Table 3: Success rate of decision tree in Figure 48: Dutch fields

Feature	**2016**	**2017**	**2018**
VV A	61.22%	86.34%	76.42%
VV D	48.62%	64.22%	68.13%
VH A	74.70%	85.33%	68.13%
VH D	62.40%	72.57%	68.87%

Then, Austrian fields were submitted to this first option. The results are shown in Table 4. Again, the best results are in features VV A and VH A backscatter, this time in season 2016 with values of 85.01% and 90.15%, respectively.

Generally, ascending orbit direction gives better results than descending orbit direction. However, it was tried to find a more consistent decision tree that worked for every season and taking profit from other potential features.

Table 4: Success rate of decision tree in Figure 48: Austrian fields

Feature	2016	2017	2018
VV A	85.01%	81.55%	72.60%
VV D	73.15%	59.26%	83.13%
VH A	90.15%	82.04%	70.76%
VH D	83.11%	65.88%	71.37%

4.5.2 Second test

After that first test, VV and VH backscatter absolute values were introduced to see if they improved the success rate. As the first test worked properly, it was tried to improve it by adding an indecision range in the first feature that could be solved using VV and VH backscatter thresholds shown in Table 5. To set these limits, a loop with some candidate values was made, and they were tested in the decision tree to see which was the set of VV and VH backscatter thresholds that gave better success rate.

Table 5: Backscatter thresholds

Feature	Upper limit	Lower limit
VV A	-14.6 dB	-16.2 dB
VV D	-15 dB	-15.7 dB
VH A	-16.8 dB	-21.2 dB
VH D	-18.6 dB	-20.2 dB

Once the limits were set in day of year of maximum peak and backscatter values, the scheme in Figure 49 was tested and delivered the results of the fields from the Netherlands in Table 6.

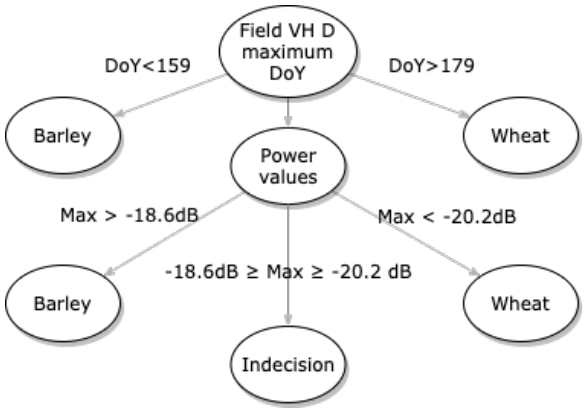


Figure 49: Decision tree: Second option

First, the reason why season 2016 presents a perfect E.R (100%) is because this season lacks the timestamp between 159th and 179th days of year and a linear interpolation is applied. Since, it is impossible to get backscatter maxima from this timestamp.

Apart from this, it is important to mention that this way, 2017 improves significantly in SR for VV D backscatter (from 59.26% to 74.97%) and VH D backscatter (from 65.88% to 84.96%). However, in 2018 season the results are worse than using the first decision tree option.

Table 6: Performance of decision tree in Figure 49: Dutch fields

Feature	**ER 2016**	**SR 2016**	**ER 2017**	**SR 2017**	ER 2018	**SR 2018**
VV A	100%	61.22%	99.32%	86.93%	95.49%	74.53%
VV D	100%	48.62%	97.40%	74.97%	98.01%	65.24%
VH A	100%	74.70%	95.37%	86.98%	91.09%	66.28%
VH D	100%	62.40%	94.58%	84.96%	97.59%	66.92%

After, Austrian fields were tested and the results are shown in Table 7. In general, SR is better and more consistent. For the first time, there is less than 10% error, in VH A backscatter 2016, when the SR gets the value of 90.15%.

Table 7: Performance of decision tree in Figure 49: Austrian fields

Feature	**ER 2016**	**SR 2016**	**ER 2017**	**SR 2017**	ER 2018	**SR 2018**
VV A	100%	85.01%	95.54%	85.65%	96.49%	71.86%
VV D	100%	73.15%	90.77%	76.87%	99.73%	83.28%
VH A	100%	90.15%	99.85%	82.10%	99.84%	70.74%
VH D	100%	83.11%	98.59%	67.29%	99.92%	71.37%

4.5.3 Third test

The last automatic classification that was tested (code in *Classification.py* in the Appendix) is based on the previous decision trees although some features are introduced. The main idea is to submit every field to the second decision tree (see in Figure 49) for VV A, VV D, VH A, VH D backscatter with their corresponding thresholds (see in Table 5)and also to study the CR index maximum minus minimum for ascending and descending orbits shown in Figure 50, which is based on Figures 45, 46 and 47.

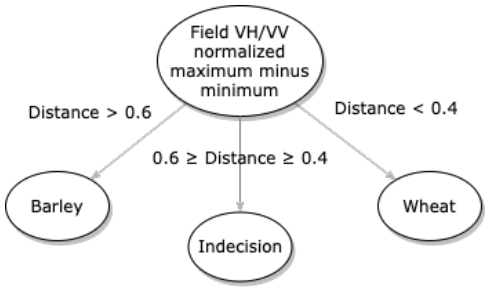


Figure 50: CR index decision tree used in final automatic classification

Every field is submitted to six different decision trees, and it gets six decisions. Then, the mode of the six decisions is used to get to a final decision. Table 8 shows the applied methodology in detail.

Table 8: Final decision depending on mode of single decisions

Mode	**Final decision**
Wheat	Wheat
Barley	Barley
Wheat & undefined	Wheat
Barley & undefined	Barley
Wheat & barley	Undefined
Wheat, barley & undefined	Undefined

And lastly, the final SR and ER results are described in Table 9 for fields from the Netherlands and Austria.

Table 9: Performance of final automatic classification for fields in the Netherlands and Austria

Country	**ER 2016**	**SR 2016**	**ER 2017**	**SR 2017**	ER 2018	**SR 2018**
NLD	86.14%	79.45%	91.68%	91.90%	90.69%	64.53%
AUT	94.75%	89.73%	87.33%	89.30%	90.87%	71.61%

The results show a great improvement in terms of general SR between years, which is a crucial aspect for a reliable algorithm. In this case the mean SR for the three seasons in the Netherlands is 78.63% and for Austria is 83.55%, which is a great result if compared with the previous tests. The ER descends a bit in general with respect to the previous test, but it is never below 86%. However, the results for 2018 were disappointing. When looking deeply in the procedure, very bad results were seen for 2018 in wheat detection when using decision trees such as in Figure 49. That year, VV and VH wheat backscatter was much higher and its maxima were not as clear as in 2016 and 2017 seasons (see in Figure 25), and this provoked that wheat was classified as barley.

5 Discussion

This research demonstrates that it is possible to separate wheat and barley fields by developing an automatic classification based on decision trees with a success rate up to 91.90% and an evaluation rate of 91.68%, which reveals a great performance. The algorithm applied uses VV, VH backscatter and CR index high-resolution Sentinel-1 data from mid May to the end of July, when barley fields tend to present a huge VV and VH backscatter peak that wheat fields do not.

In line with previous works in the subject [4], VV and VH backscatter, as well as CR index basically depend on crops structural changes (growth and bending) from mid May until August, and on SWV for the rest of the growing season. Moreover, VV backscatter is more affected by the early growing of crops than VH backscatter, as it is clear in 2016 time series of Figure 22 in comparison with the same season of Figure 24.

As stated in previous sections, during the last days of stem extension and at the beginning of heading stage is where VV and VH backscatter for wheat and barley differ the most. In the other stages both crop types have similar absolute values, as well as in CR during the whole season.

The "M-shape" that is possible to see around May and June in CR index Figures 26 and 27 for 2016 and 2017 is caused by the abrupt VH backscatter barley peak in comparison to wheat. In Figure 25, for 2018 the barley peak is not very far from the wheat peak because wheat shows higher values than in previous seasons and this is what makes 2018 CR index show a smoother shape. This is caused by the low values of precipitation during these months for 2018.

The first decision tree test is a very simple decision tree that decides barley if the late spring maximum is located before 159th day of year or wheat otherwise. Although it does not leave an indecision zone, it gives good success rate results for VH A backscatter, especially for Austrian fields (2016 gives 90.15%, 2017 gives 82.04% and 2018 gives 70.76%) although the extreme simplicity of the procedure. The advantages of this methodology are its computational simplicity and its complete classification of all fields.

The second test introduces a second level of decision based on absolute VV and VH backscatter values and an indecision range. In this case, the field needs to present its maximum after 179th day of year to decide wheat. Then, the fields that have their maximum between 159th and 179th days of year are classified by the absolute value of the maximum depending on the feature, as stated above in Table 5. There are fields that remain not classified due to their typical values of both wheat and barley. The results for this algorithm are better in success rate for VV D and VH D backscatter features (almost 20% improve in 2017) and Austrian fields show excellent rates (up to 90%). With a simple second level of decision, the results were slightly better, although the improvement margin was still great.

The third and final option of the decision tree is the most solid algorithm when looking generally to all years. It is based on a set of decisions of the second test and CR index maximum minus minimum analysis stated in the previous section and gives very high success rates for 2016 and 2017 seasons (up

to 91.9%).

In conclusion, the performance of all three classification methods satisfy the proposed needs and it uses only backscatter. Obviously, the consistency of the procedure depends on the complexity of the algorithm, in this case, the third option is the most consistent.

However, there are some aspects that can be improved. The first improvable is to find out numerically the relationship between VV, VH backscatter and their ratio CR with ERA5-Land data to modify thresholds depending on yearly temperatures or precipitations. It could help significantly the results in atypical seasons with respect to weather. Also, it would be nice to check how geographical position of fields affect the classification.

Moreover, this automatic classification should be tested with many more seasons to see the average performance, because three seasons are not enough to test the final result given the standard deviation present in backscatter data and the variations in mean from one year to another (as seen in 2018) in the crucial months of May and June.

Another interesting future research should be to try to distinguish wheat and barley the sooner the better in the growing season. This will allow to apply the results into field management practices sooner and consequently improve fields performance.

As it is mentioned in the conclusions of [4], a larger variety of data is required to see general behaviors and the same aspect is seen in this project.

6 Conclusion

By analyzing high-resolution Sentinel-1 data from a subset of fields in the Netherlands and Austria and studying the phenological stages of wheat and barley crops, this thesis has shown how to define a simple, automatic classification method that is able to separate wheat and barley fields using only Sentinel-1 backscatter data with a success rate up to 91.9%.

This project clearly demonstrates that it is possible to make a good separation, although it is clearly seen that the algorithm performance depends greatly on each season's weather conditions that have an effect on the crops and consequently satellite observations. Because of this, this study also included an extensive analysis of weather conditions and year-to-year variability of the EO datasets including the obtained findings in the decision tree would lead to a more robust classification in multiple years, and should be considered in the future.

Also, further research should consider trying to get a classification suitable to be applied as soon as possible in the growing stage to be able to act sooner and consequently to get practical information before.

In summary, the output of the study carried out within this thesis is more detailed knowledge of structural effects of winter wheat and winter barley on backscatter throughout the growing season, how the backscatter is affected by weather conditions, and a simple but well-performing decision tree for the automatic classification of wheat and barley fields using only Sentinel-1 backscatter data.

References

[1] E. Schanda, *Physical Fundamentals of Remote Sensing.* Springer Verlag, 1986.

[2] X. Zhang and X. Cai, "Climate change impacts on global agricultural land availability," *Environmental Research Letters*, vol. 6, no. 1, p. 014014, Jan 2011. [Online]. Available: http://dx.doi.org/10.1088/1748-9326/6/1/014014

[3] M. Vreugdenhil, W. Wagner, B. Bauer-Marschallinger, I. Pfeil, I. Teubner, C. Rüdiger, and P. Strauss, "Sensitivity of sentinel-1 backscatter to vegetation dynamics: An austrian case study," *Remote Sensing*, vol. 10, no. 9, p. 1396, Sep 2018. [Online]. Available: http://dx.doi.org/10.3390/rs10091396

[4] K. Harfenmeister, D. Spengler, and C. Weltzien, "Analyzing temporal and spatial characteristics of crop parameters using sentinel-1 backscatter data," *Remote Sensing*, vol. 11, no. 13, p. 1569, Jul 2019. [Online]. Available: http://dx.doi.org/10.3390/rs11131569

[5] F. T. Ulaby, R. K. Moore, and A. K. Fung, *Microwave Remote Sensing: Active and Passive.* Artech House, 1981, 1982 and 1986, vol. 1, 2 and 3.

[6] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture:key advances and remaining knowledge gaps," *El Sevier*, September 2012.

[7] [Online]. Available: https://apollomapping.com/worldview-1-satellite-imagery

[8] M. Kirscht and C. Rinke, "3d reconstruction of buildings and vegetation from synthetic aperture radar (sar) images," November 1998.

[9] C. A. Liu, Z. X. Chen, Y. Shao, J. S. Chen, T. Hasi, and H. Z. Pan, "Research advances of sar remote sensing for agriculture applications: A review," *Journal of Integrative Agriculture*, vol. 18, no. 3, pp. 506–525, Mar 2019. [Online]. Available: http://dx.doi.org/10.1016/S2095-3119(18)62016-7

[10] Ying, W. Li, and C. Shen, "Removal of optically thick clouds from high-resolution satellite imagery using dictionary group learning and interdictionary nonlocal joint sparse coding." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 5, pp. 1870–1882, 2017.

[11] [Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/s/seasat

[12] [Online]. Available: https://sentinel.esa.int/web/sentinel/missions/sentinel-1

[13] E. C. Large, *Growth stages in cereals*, December 1954, vol. 3.

[14] A. B. Fernando Santos and C. Caldas, *Sugarcane: Agricultural Production, Bioenergy and Ethanol*. Elsevier, 2015.

[15] [Online]. Available: http://agriculture.vic.gov.au/agriculture/grains-and-other-crops/crop-production/identification-of-cereal-seedlings

[16] P. Singh and R. Shree, "Analysis and effects of speckle noise in sar images," *2016 2nd International Conference on Advances in Computing, Communication and Automation (ICACCA) (Fall)*, Sep 2016. [Online]. Available: http://dx.doi.org/10.1109/ICACCAF.2016.7748978

[17] [Online]. Available: https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-land?tab=overview

[18] F. Mattia, T. Le Toan, G. Picard, F. Posa, A. D'Alessio, C. Notarnicola, A. Gatti, M. Rinaldi, G. Satalino, and G. Pasquariello, "Multitemporal c-band radar measurements on wheat fields," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, July 2003.

A Python Code

A.1 *Time_series.py*

```
### Time_series.py: File used to observe time series

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt
from sklearn import preprocessing
months = mdates.MonthLocator()


def normalise(ww, wb):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb


def plot_features(feature='mean-VH-A', idate=None,
                  fdate=None, scale_0_1=True):
    '''Function that plots the time series'''

    '''Crop IDs: Wheat = 1 and Barley = 2'''
    crop1 = 1
    crop2 = 2

    '''Create figure and plot data'''
    fig, axes = plt.subplots(3, 1, figsize=(9, 7.8))

    for year, ax in zip(range(2016, 2019), axes):
        '''Data loading setup'''
        seas = 1
        if year == 2017:
            seas = 2
        elif year == 2018:
            seas = 3

        path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
                   "Datasets/zonal_stats_NLD_s{}.npy".format(seas)
        lut = pd.read_csv('/Users/annadominguezcosta/Desktop/Documentos/'
                          'Erasmus/TFG/Datasets/ERA5-Land/temporary_Freigabe'
                          '/lookup_FID_fieldID.csv',
```

```
                          names=['FeatID', 'FieldID'], delimiter=';', header=None)

        LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
                   "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
        classes_all = gp.read_file(LPIS_all)
        classes_all.index = lut['FieldID'].values
        classes = classes_all.copy()

        np_array = np.load(path_npy)
        np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
                              dt.datetime(int(year), 10, 20), freq='12D')
        np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
                    u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
                    u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
                    u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
                    u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
                    u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
                    u'tm2', u'tp', u'swvl1', u'swvl2']
        np_idx = np.where(np.array(np_feats) == feature)[0][0]
        np_data = np_array[:, :, np_idx]

        '''Dataframe preparation and get rid of NaN values'''
        df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
                                   index=classes_all.index)
        df_from_npy[df_from_npy >= 0] = np.nan
        df_from_npy['class'] = classes[str(year)].values
        df_from_npy = df_from_npy.dropna(axis=0)

        '''Select the desired timestamps'''
        labels = []
        if idate is not None:
            if year == 2016:
                idate -= dt.timedelta(days=1)
            idate -= dt.timedelta(days=12)
            labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
                                   idate, freq='12D')
        if fdate is not None:
            if year == 2016:
                fdate -= dt.timedelta(days=1)
            fdate += dt.timedelta(days=12)
            labels = labels.append(
                pd.date_range(fdate, dt.datetime(int(year), 10, 20), freq='12D'))
        df_from_npy = df_from_npy.drop(labels, axis=1)

        '''Subset of dataframe with certain crop type'''
        ww = df_from_npy[df_from_npy['class'] == crop1].copy().transpose()
        wb = df_from_npy[df_from_npy['class'] == crop2].copy().transpose()

        '''Pick the same number of fields to study'''
        ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
        wb = wb[wb.columns[:min(ww.shape[1], wb.shape[1])]]
        ww.drop('class', inplace=True)
        wb.drop('class', inplace=True)
```

```
103
104         '''Normalize time series (scale between 0 and 1) if asked'''
105         if scale_0_1:
106             ww, wb = normalise(ww, wb)
107
108         '''Time series figure preparation'''
109         ww_plot = ww.copy()[ww.columns[:]]
110         wb_plot = wb.copy()[wb.columns[:]]
111         ax.grid()
112         ax.plot(ww_plot.mean(axis=1), color='r', linewidth=0.9,
113                 marker='.', label='Wheat')
114         ax.plot(wb_plot.mean(axis=1), color='b', linewidth=0.9,
115                 marker='.', label='Barley')
116         ax.fill_between(ww_plot.index, ww_plot.quantile(0.25, axis=1),
117                         ww_plot.quantile(0.75, axis=1), color='r', alpha=0.2)
118         ax.fill_between(wb_plot.index, wb_plot.quantile(0.25, axis=1),
119                         wb_plot.quantile(0.75, axis=1), color='b', alpha=0.2)
120         ax.legend()
121
122         if scale_0_1:
123             ax.set_ylabel('{}, normalised'.format(feature))
124         else:
125             ax.set_ylabel('{} [dB]'.format(feature))
126
127         if scale_0_1 is True:
128             ax.set_ylim(top=1, bottom=0)
129         elif feature.__contains__('VV'):
130             ax.set_ylim(top=-10, bottom=-24)
131         elif feature.__contains__('VH'):
132             ax.set_ylim(top=-14, bottom=-28)
133         else:
134             ax.set_ylim(top=-2, bottom=-10)
135
136         ax.set_title('Feature: {};   Season: {}'.format(feature, year))
137
138     plt.suptitle('Time series of {} wheat and barley fields'.format
139                  (len(ww_plot.columns)), fontsize=16)
140
141     plt.subplots_adjust(left=0.1, bottom=0.05, right=0.95, top=0.9,
142                         hspace=0.4)
143     plt.show()
144
145
146 '''Main function'''
147 if __name__ == "__main__":
148
149     '''Select feature'''
150     feat = 'mean-CR-D'
151
152     '''Set true to activate normalization'''
153     scale_0_1 = False
154
155     plot_features(feature=feat, scale_0_1=scale_0_1)
```

```
156
157     print('Done')
```

A.2 *Weather.py*

```
### Weather.py: File used to analyse the weather impact on measurements

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt
from sklearn import preprocessing
months = mdates.MonthLocator()


def read_era(var, seas):
    '''Function that puts ERA5-Land data from csv to dict'''
    data_dict = {}
    df = pd.read_csv('/Users/annadominguezcosta/Desktop/Documentos/Erasmus'
                     '/TFG/Datasets/ERA5-Land/temporary_Freigabe'
                     '/ERA_NL_{}_S{}.csv'.format(var, seas), index_col=0)
    data_dict[var] = df.sort_index()
    return data_dict


def normalise(ww=None, wb=None):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb


def plot_weather(feature='mean-VH-A', era_var='swvl1', idate=None,
                 fdate=None, scale_0_1=True, plot_type='bar plot'):
    '''Function that plots the time series
    and the weather feature along the season'''

    '''Crop IDs: Wheat = 1 and Barley = 2'''
    crop1 = 1
    crop2 = 2

    '''Data loading setup'''
    era_data = {}
    num_fields = 0
    for year in range(2016,2019):

        seas = 1
        if year == 2017:
```

```
52              seas = 2
53          elif year == 2018:
54              seas = 3
55
56          '''Read the era variable for the desired season'''
57          era_dict = read_era(era_var, seas)
58
59          path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
60                 "Datasets/zonal_stats_NLD_s{}.npy".format(seas)
61          lut = pd.read_csv('/Users/annadominguezcosta/Desktop/Documentos/'
62                        'Erasmus/TFG/Datasets/ERA5-Land/temporary_Freigabe'
63                        '/lookup_FID_fieldID.csv',
64                        names=['FeatID', 'FieldID'], delimiter=';', header=None)
65
66          LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
67                 "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
68          classes_all = gp.read_file(LPIS_all)
69          classes_all.index = lut['FieldID'].values
70          classes = classes_all.copy()
71
72          np_array = np.load(path_npy)
73          np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
74                                dt.datetime(int(year), 10, 20), freq='12D')
75          np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
76                  u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
77                  u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
78                  u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
79                  u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
80                  u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
81                  u'tm2', u'tp', u'swvl1', u'swvl2']
82          np_idx = np.where(np.array(np_feats) == feature)[0][0]
83          np_data = np_array[:, :, np_idx]
84
85          '''Dataframe preparation and get rid of NaN values'''
86          df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
87                                     index=classes_all.index)
88          df_from_npy[df_from_npy >= 0] = np.nan
89          df_from_npy['class'] = classes[str(year)].values
90          df_from_npy = df_from_npy.dropna(axis=0)
91
92          '''Select the desired timestamps'''
93          labels = []
94          if idate is not None:
95              if year == 2016:
96                  idate -= dt.timedelta(days=1)
97              idate -= dt.timedelta(days=12)
98              labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
99                                     idate, freq='12D')
100         if fdate is not None:
101             if year == 2016:
102                 fdate -= dt.timedelta(days=1)
103             fdate += dt.timedelta(days=12)
104             labels = labels.append(
```

```
                pd.date_range(fdate, dt.datetime(int(year), 10, 20), freq='12D'))
        df_from_npy = df_from_npy.drop(labels, axis=1)

        '''Subset of df with certain crop type'''
        ww = df_from_npy[df_from_npy['class'] == crop1].copy().transpose()
        wb = df_from_npy[df_from_npy['class'] == crop2].copy().transpose()
        ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
        ww.drop('class', inplace=True)

        '''Normalize time series (scale between 0 and 1)'''
        if scale_0_1:
            ww = normalise(ww)

        '''Time series preparation'''
        ww_plot = ww.copy()[ww.columns[:]]

        '''Average in situ data over chosen fields'''
        ww_cols = ww_plot.columns
        ww_var_df = era_dict[era_var]
        ww_era = ww_var_df.loc[ww_cols].transpose()
        ww_era.index = np_dt - dt.timedelta(days=seas*365)
        if year == 2016:
            ww_era.index = ww_era.index + dt.timedelta(days=1)
        ww_era.drop(labels, inplace=True)

        era_data['ww_{}_{}'.format(era_var, year)] = ww_era.mean(axis=1)
        num_fields += len(ww.columns)


    '''Create figure and plot data'''
    fig, ax = plt.subplots(1, 1, figsize=(10, 3.5))
    ax.grid()

    '''Plot settings dependig on plot type'''
    if plot_type == 'time series':
        ax.plot(era_data['ww_{}_{}'.format(era_var, 2016)], color='c',
                linewidth=0.9, marker='.', label='2016')
        ax.plot(era_data['ww_{}_{}'.format(era_var, 2017)], color='m',
                linewidth=0.9, marker='.', label='2017')
        ax.plot(era_data['ww_{}_{}'.format(era_var, 2018)], color='y',
                 linewidth=0.9, marker='.', label='2018')
    elif plot_type == 'bar plot':
        width = 2.5
        ax.bar(era_data['ww_{}_{}'.format(era_var, 2016)].keys() -
        dt.timedelta(days=3), era_data[
            'ww_{}_{}'.format(era_var,2016)].values,color='c', width=width,
               label='2016')
        ax.bar(era_data['ww_{}_{}'.format(era_var, 2017)].keys(), era_data[
            'ww_{}_{}'.format(era_var,2017)].values,color='m', width=width,
               label='2017')
        ax.bar(era_data['ww_{}_{}'.format(era_var, 2018)].keys() +
        dt.timedelta(days=3), era_data[
            'ww_{}_{}'.format(era_var,2018)].values,color='y', width=width,
```

```
                label='2018')
    else:
        print('ERROR')

    # Set the locator
    locator = mdates.MonthLocator()  # every month
    # Specify the format - %b gives us Jan, Feb...
    fmt = mdates.DateFormatter('%b')
    X = plt.gca().xaxis
    X.set_major_locator(locator)
    # Specify formatter
    X.set_major_formatter(fmt)
    ax.set_axisbelow(True)
    ax.legend()

    if era_var.__contains__('swvl'):
        ax.set_ylabel(era_var.upper() + ' [m3/m3]')
    elif era_var.__contains__('t2m'):
        ax.set_ylabel(era_var.upper() + ' [$^\circ$C]')
    else:
        ax.set_ylabel(era_var.upper() + ' [m]')

    plt.suptitle('ERA5-Land time series', fontsize=16)
    ax.set_title('Feature: {}, Number of fields: {}'.format(era_var,
                                                            num_fields))

    plt.subplots_adjust(left=0.1, bottom=0.1, right=0.95, top=0.85,
                        hspace=0.4)
    plt.show()


'''Main function'''
if __name__ == "__main__":

    '''Select feaures, season and plot type'''
    feat = 'mean-CR-D'
    season = 2018
    era_var = 'tp'

    #plot_type = 'time series'
    plot_type = 'bar plot'

    '''Timestamps to plot'''
    initial_date = dt.datetime(season, 5, 12)
    final_date = dt.datetime(season, 7, 11)

    '''Uncomment to plot the whole season'''
    initial_date = None
    final_date = None

    '''Set true to activate normalization'''
    scale_0_1 = False

```

```
211     plot_weather(feature=feat, era_var=era_var, idate=initial_date,
212                  fdate=final_date, scale_0_1=scale_0_1, plot_type=plot_type)
213
214     print('done')
```

A.3 *DoY_analysis.py*

```
### DoY_analysis.py: File used to analyze the Day of Year of the first
# maximum given a time interval

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
from scipy import signal as sgn
from sklearn import preprocessing
from os import path

def get_maximum(column=None, height=0.7):
    try:
        return sgn.find_peaks(x=column, height=height)[0][0]
    except:
        return np.argmax(column)


def normalise(ww, wb):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb



def doy_analysis(feature='mean-VH-A', year=2017, idate=None, fdate=None,
                 height=0.7, scale_0_1=False):
    '''Function that shows a boxplot  with respect to DoY'''

    '''Wheat = 1, Barley = 2'''
    crop1 = 1
    crop2 = 2

    '''Data loading setup'''
    seas = 1
    if year == 2017:
        seas = 2
    elif year == 2018:
        seas = 3

    path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
               "Datasets/zonal_stats_NLD_s{}.npy".format(seas)
    lut = pd.read_csv('/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/'
```

```
52                      'Datasets/ERA5-Land/temporary_Freigabe/lookup_FID_fieldID.csv',
53                      names=['FeatID', 'FieldID'], delimiter=';', header=None)
54
55      LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
56                 "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
57      classes_all = gp.read_file(LPIS_all)
58      classes_all.index = lut['FieldID'].values
59      classes = classes_all.copy()
60
61      np_array = np.load(path_npy)
62      np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
63                            dt.datetime(int(year), 10, 20), freq='12D')
64      np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
65                  u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
66                  u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
67                  u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
68                  u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
69                  u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
70                  u'tm2', u'tp', u'swvl1', u'swvl2']
71      np_idx = np.where(np.array(np_feats) == feature)[0][0]
72      np_data = np_array[:, :, np_idx]
73
74      '''Dataframe preparation and get rid of NaN values'''
75      df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
76                                 index=classes_all.index)
77      df_from_npy[df_from_npy >= 0] = np.nan
78      df_from_npy['class'] = classes[str(year)].values
79      df_from_npy = df_from_npy.dropna(axis=0)
80
81      '''Select the desired timestamps'''
82      labels = []
83      if idate is not None:
84          if year == 2016:
85              idate -= dt.timedelta(days=1)
86          idate -= dt.timedelta(days=12)
87          labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
88                                 idate, freq='12D')
89      if fdate is not None:
90          if year == 2016:
91              fdate -= dt.timedelta(days=1)
92          fdate += dt.timedelta(days=12)
93          labels = labels.append(
94              pd.date_range(fdate, dt.datetime(int(year), 10, 20), freq='12D'))
95      df_from_npy = df_from_npy.drop(labels, axis=1)
96
97
98      '''Subset of df with certain crop type'''
99      ww = df_from_npy[df_from_npy['class'] == crop1].copy().transpose()
100     wb = df_from_npy[df_from_npy['class'] == crop2].copy().transpose()
101
102     '''Keep the lowest number of fields and equal for both crop types'''
103     ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
104     wb = wb[wb.columns[:min(ww.shape[1], wb.shape[1])]]
```

```
105     ww.drop('class', inplace=True)
106     wb.drop('class', inplace=True)
107 
108     '''Normalize time series (scale between 0 and 1)'''
109     if scale_0_1:
110         ww, wb = normalise(ww, wb)
111 
112     '''Look for the first maximum of each field and keep the index'''
113     ww_plot_maxIndex = ww.apply(lambda column: get_maximum(column, height)).values
114     wb_plot_maxIndex = wb.apply(lambda column: get_maximum(column, height)).values
115 
116     '''Remove the rows that have not a maximum'''
117     ww_plot_maxIndex = ww_plot_maxIndex[~np.isnan(ww_plot_maxIndex)]
118     wb_plot_maxIndex = wb_plot_maxIndex[~np.isnan(wb_plot_maxIndex)]
119 
120     '''Time index in datetime'''
121     timeindex = np_dt.drop(labels)
122 
123     '''Plot preparation'''
124     data = [timeindex[ww_plot_maxIndex.astype(int)].dayofyear,
125             timeindex[wb_plot_maxIndex.astype(int)].dayofyear]
126 
127     '''Comment blocks to write or read accumulated data'''
128 
129     '''Write data and copy to csv'''
130     '''
131     df_wmax = pd.DataFrame({labels[0]: data[0]})
132     df_bmax = pd.DataFrame({labels[1]: data[1]})
133 
134     if path.exists('wheat_max_VH_A_analysis.csv' and 'wheat_max_VH_D_analysis.csv'):
135         if feature.__contains__('VH-A'):
136             df_wmax.to_csv('wheat_max_VH_A_analysis.csv', index=False,
137             mode='a', header=False)
138             df_bmax.to_csv('barley_max_VH_A_analysis.csv', index=False,
139             mode='a', header=False)
140         else:
141             df_wmax.to_csv('wheat_max_VH_D_analysis.csv', index=False,
142             mode='a', header=False)
143             df_bmax.to_csv('barley_max_VH_D_analysis.csv', index=False,
144             mode='a', header=False)
145     else:
146         if feature.__contains__('VH-A'):
147             df_wmax.to_csv('wheat_max_VH_A_analysis.csv', index=False,
148             header=True)
149             df_bmax.to_csv('barley_max_VH_A_analysis.csv', index=False,
150             header=True)
151         else:
152             df_wmax.to_csv('wheat_max_VH_D_analysis.csv', index=False,
153             header=True)
154             df_bmax.to_csv('barley_max_VH_D_analysis.csv', index=False,
155             header=True)
156     '''
157 
```

```
    '''Read data'''

    if feature.__contains__('VH-A'):
            df_wmax = pd.read_csv('wheat_max_VH_A_analysis.csv')
            df_bmax = pd.read_csv('barley_max_VH_A_analysis.csv')
    else:
            df_wmax = pd.read_csv('wheat_max_VH_D_analysis.csv')
            df_bmax = pd.read_csv('barley_max_VH_D_analysis.csv')

    data = [df_wmax.values.squeeze(), df_bmax.values.squeeze()]


    '''Create figure and plot data'''
    fig, ax = plt.subplots(1, 1, figsize=(6, 5))
    labels = ['Wheat fields', 'Barley fields']
    red_square = dict(markerfacecolor='r', marker='s')
    bplot = ax.boxplot(x=data, labels=labels, notch=True,
                   flierprops=red_square,
               patch_artist=True, whis=1.25)
    ax.set_ylabel('DFM')
    ax.grid()

    '''Fill with colors'''
    colors = ['red', 'blue']
    for patch, color in zip(bplot['boxes'], colors):
        patch.set_facecolor(color)

    ticks = np.arange(min(min(data[0]), min(data[1])),
                      max(max(data[0]),max(data[1]))+12, step=12)
    plt.yticks(ticks)
    plt.suptitle('Day of First Maximum (DFM) analysis of {} wheat and '
                 'barley fields'.format(data[0].size))
    plt.title('Feature: {};  Period: 12th May to 11th July'.format(
        feature))
    plt.subplots_adjust(left=0.13, bottom=0.1, right=0.95, top=0.85)
    plt.show()


'''Main function'''
if __name__ == "__main__":
    feat = 'mean-VH-D'

    '''Minimum relative height to consider a maximum'''
    height = 0.7

    season = 2016

    '''Timestamps to plot'''
    initial_date = dt.datetime(season, 5, 12)
    final_date = dt.datetime(season, 7, 11)

    '''Uncomment to plot the whole season'''
    '''initial_date = None'''
```

```
211     '''final_date = None'''
212
213     '''Set true to activate normalization'''
214     scale_0_1 = True
215
216     doy_analysis(feature=feat, year=season, idate=initial_date,
217                  fdate=final_date, height=height, scale_0_1=scale_0_1)
218     print('done')
```

A.4 *Tseries_max_analysis.py*

```
### Tseries_max_analysis.py: File used to check the DoY analysis performance

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
from sklearn import preprocessing
from scipy import signal as sgn
import matplotlib.dates as mdates


def get_maximum(column=None, height=0.7):
    '''Function tht gets the maximum index of a column. Otherwise it
    gives the maximum'''
    try:
        return sgn.find_peaks(x=column, height=height)[0][0]
    except:
        return np.argmax(column)


def normalise(ww, wb):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb



def check_max(feature='mean-VH-A', field_number=0, height=0, year=2017,
              idate=None, fdate=None, scale_0_1=True):
    '''Function that plots the time series and it highlights the
    desired maximum'''

    '''Wheat = 1, Barley = 2'''
    crop1 = 1
    crop2 = 2

    '''Data loading setup'''
    seas = 1
    if year == 2017:
        seas = 2
    elif year == 2018:
        seas = 3

```

```
52      path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
53                 "Datasets/zonal_stats_NLD_s{}.npy".format(seas)
54      lut = pd.read_csv(
55          '/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/'
56          'Datasets/ERA5-Land/temporary_Freigabe/lookup_FID_fieldID.csv',
57          names=['FeatID', 'FieldID'], delimiter=';', header=None)
58
59      LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
60                 "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
61      classes_all = gp.read_file(LPIS_all)
62      classes_all.index = lut['FieldID'].values
63      classes = classes_all.copy()
64
65      np_array = np.load(path_npy)
66      np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
67                            dt.datetime(int(year), 10, 20), freq='12D')
68      np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
69                  u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
70                  u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
71                  u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
72                  u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
73                  u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
74                  u'tm2', u'tp', u'swvl1', u'swvl2']
75      np_idx = np.where(np.array(np_feats) == feature)[0][0]
76      np_data = np_array[:, :, np_idx]
77
78
79      '''Dataframe preparation and get rid of NaN values'''
80      df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
81                                 index=classes_all.index)
82      df_from_npy[df_from_npy >= 0] = np.nan
83      df_from_npy['class'] = classes[str(year)].values
84      df_from_npy = df_from_npy.dropna(axis=0)
85
86      '''Select the desired timestamps'''
87      labels = []
88      if idate is not None:
89          if year == 2016:
90              idate -= dt.timedelta(days=1)
91          idate -= dt.timedelta(days=12)
92          labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
93                                 idate, freq='12D')
94      if fdate is not None:
95          if year == 2016:
96              fdate -= dt.timedelta(days=1)
97          fdate += dt.timedelta(days=12)
98          labels = labels.append(
99              pd.date_range(fdate, dt.datetime(int(year), 10, 20), freq='12D'))
100     df_from_npy = df_from_npy.drop(labels, axis=1)
101
102     '''Subset of df with certain crop type'''
103     ww = df_from_npy[df_from_npy['class'] == crop1].copy().transpose()
104     wb = df_from_npy[df_from_npy['class'] == crop2].copy().transpose()
```

```
105     ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
106     wb = wb[wb.columns[:min(ww.shape[1], wb.shape[1])]]
107     ww.drop('class', inplace=True)
108     wb.drop('class', inplace=True)
109 
110     '''Normalize time series (scale between 0 and 1)'''
111     if scale_0_1:
112         ww, wb = normalise(ww, wb)
113 
114     '''Time series preparation'''
115     ww_plot = ww.copy()[ww.columns[:]]
116     wb_plot = wb.copy()[wb.columns[:]]
117 
118     '''Create figure and plot data'''
119     fig, ax = plt.subplots(1, 1, figsize=(8, 5))
120     ax.grid()
121 
122     '''Plot the field measurement and mark the maximum calculated'''
123     ax.plot(ww_plot[ww_plot.columns[field_number]], color='r', marker='.',
124             label='Wheat')
125     ax.plot(wb_plot[wb_plot.columns[field_number]], color='b', marker='.',
126             label='Barley')
127 
128     '''Mark the first maximum'''
129     posMaxW = get_maximum(ww_plot[ww_plot.columns[field_number]], height)
130     plt.plot(ww_plot.index[posMaxW],
131              ww_plot[ww_plot.columns[field_number]][posMaxW], 'rD')
132 
133     posMaxB = get_maximum(wb_plot[wb_plot.columns[field_number]], height)
134     plt.plot(ww_plot.index[posMaxB],
135              wb_plot[wb_plot.columns[field_number]][posMaxB], 'bD')
136 
137     ax.axhline(y=height)
138     ax.set_axisbelow(True)
139     ax.xaxis.set_major_locator(mdates.MonthLocator())
140     ax.legend()
141     if scale_0_1:
142         ax.set_ylabel('{}, normalised'.format(feature))
143     else:
144         ax.set_ylabel('{} [dB]'.format(feature))
145 
146     plt.suptitle('Time series of wheat and barley field number {}'.format
147                  (field_number), fontsize=16)
148     ax.set_title('Feature: {};   Season: {}'.format(feature, year))
149 
150     plt.subplots_adjust(left=0.1, bottom=0.1, right=0.95, top=0.85)
151     plt.show()
152 
153 
154 '''Main function'''
155 if __name__ == "__main__":
156 
157     feat = 'mean-VH-D'
```

```
158     season = 2016
159
160     '''Only plot the wheat and barley field with number...'''
161     field_number = 4
162
163     '''Minimum relative height to consider a maximum'''
164     height = 0.7
165
166     '''Timestamps to plot'''
167     initial_date = dt.datetime(season, 5, 12)
168     final_date = dt.datetime(season, 7, 11)
169
170     '''Uncomment to plot the whole season'''
171     #initial_date = None
172     #final_date = None
173
174     '''Set true to activate normalization'''
175     scale_0_1 = True
176
177     check_max(feature=feat, field_number=field_number, height=height,
178               year=season, idate=initial_date, fdate=final_date,
179               scale_0_1=True)
180
181     print('done')
```

A.5 *MaxMin_analysis.py*

```
### Tseries_max_analysis.py: File used to check the DoY analysis performance

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
from sklearn import preprocessing
from scipy import signal as sgn
import matplotlib.dates as mdates


def get_maximum(column=None, height=0.7):
    '''Function tht gets the maximum index of a column. Otherwise it
    gives the maximum'''
    try:
        return sgn.find_peaks(x=column, height=height)[0][0]
    except:
        return np.argmax(column)


def normalise(ww, wb):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb



def check_max(feature='mean-VH-A', field_number=0, height=0, year=2017,
              idate=None, fdate=None, scale_0_1=True):
    '''Function that plots the time series and it highlights the
    desired maximum'''

    '''Wheat = 1, Barley = 2'''
    crop1 = 1
    crop2 = 2

    '''Data loading setup'''
    seas = 1
    if year == 2017:
        seas = 2
    elif year == 2018:
        seas = 3

```

```
52      path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
53                 "Datasets/zonal_stats_NLD_s{}.npy".format(seas)
54      lut = pd.read_csv(
55          '/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/'
56          'Datasets/ERA5-Land/temporary_Freigabe/lookup_FID_fieldID.csv',
57          names=['FeatID', 'FieldID'], delimiter=';', header=None)
58
59      LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
60                 "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
61      classes_all = gp.read_file(LPIS_all)
62      classes_all.index = lut['FieldID'].values
63      classes = classes_all.copy()
64
65      np_array = np.load(path_npy)
66      np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
67                            dt.datetime(int(year), 10, 20), freq='12D')
68      np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
69                  u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
70                  u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
71                  u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
72                  u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
73                  u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
74                  u'tm2', u'tp', u'swvl1', u'swvl2']
75      np_idx = np.where(np.array(np_feats) == feature)[0][0]
76      np_data = np_array[:, :, np_idx]
77
78
79      '''Dataframe preparation and get rid of NaN values'''
80      df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
81                                 index=classes_all.index)
82      df_from_npy[df_from_npy >= 0] = np.nan
83      df_from_npy['class'] = classes[str(year)].values
84      df_from_npy = df_from_npy.dropna(axis=0)
85
86      '''Select the desired timestamps'''
87      labels = []
88      if idate is not None:
89          if year == 2016:
90              idate -= dt.timedelta(days=1)
91          idate -= dt.timedelta(days=12)
92          labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
93                                 idate, freq='12D')
94      if fdate is not None:
95          if year == 2016:
96              fdate -= dt.timedelta(days=1)
97          fdate += dt.timedelta(days=12)
98          labels = labels.append(
99              pd.date_range(fdate, dt.datetime(int(year), 10, 20), freq='12D'))
100     df_from_npy = df_from_npy.drop(labels, axis=1)
101
102     '''Subset of df with certain crop type'''
103     ww = df_from_npy[df_from_npy['class'] == crop1].copy().transpose()
104     wb = df_from_npy[df_from_npy['class'] == crop2].copy().transpose()
```

```
105     ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
106     wb = wb[wb.columns[:min(ww.shape[1], wb.shape[1])]]
107     ww.drop('class', inplace=True)
108     wb.drop('class', inplace=True)
109 
110     '''Normalize time series (scale between 0 and 1)'''
111     if scale_0_1:
112         ww, wb = normalise(ww, wb)
113 
114     '''Time series preparation'''
115     ww_plot = ww.copy()[ww.columns[:]]
116     wb_plot = wb.copy()[wb.columns[:]]
117 
118     '''Create figure and plot data'''
119     fig, ax = plt.subplots(1, 1, figsize=(8, 5))
120     ax.grid()
121 
122     '''Plot the field measurement and mark the maximum calculated'''
123     ax.plot(ww_plot[ww_plot.columns[field_number]], color='r', marker='.',
124             label='Wheat')
125     ax.plot(wb_plot[wb_plot.columns[field_number]], color='b', marker='.',
126             label='Barley')
127 
128     '''Mark the first maximum'''
129     posMaxW = get_maximum(ww_plot[ww_plot.columns[field_number]], height)
130     plt.plot(ww_plot.index[posMaxW],
131              ww_plot[ww_plot.columns[field_number]][posMaxW], 'rD')
132 
133     posMaxB = get_maximum(wb_plot[wb_plot.columns[field_number]], height)
134     plt.plot(ww_plot.index[posMaxB],
135              wb_plot[wb_plot.columns[field_number]][posMaxB], 'bD')
136 
137     ax.axhline(y=height)
138     ax.set_axisbelow(True)
139     ax.xaxis.set_major_locator(mdates.MonthLocator())
140     ax.legend()
141     if scale_0_1:
142         ax.set_ylabel('{}, normalised'.format(feature))
143     else:
144         ax.set_ylabel('{} [dB]'.format(feature))
145 
146     plt.suptitle('Time series of wheat and barley field number {}'.format
147                  (field_number), fontsize=16)
148     ax.set_title('Feature: {};   Season: {}'.format(feature, year))
149 
150     plt.subplots_adjust(left=0.1, bottom=0.1, right=0.95, top=0.85)
151     plt.show()
152 
153 
154 '''Main function'''
155 if __name__ == "__main__":
156 
157     feat = 'mean-VH-D'
```

```
158     season = 2016
159
160     '''Only plot the wheat and barley field with number...'''
161     field_number = 4
162
163     '''Minimum relative height to consider a maximum'''
164     height = 0.7
165
166     '''Timestamps to plot'''
167     initial_date = dt.datetime(season, 5, 12)
168     final_date = dt.datetime(season, 7, 11)
169
170     '''Uncomment to plot the whole season'''
171     #initial_date = None
172     #final_date = None
173
174     '''Set true to activate normalization'''
175     scale_0_1 = True
176
177     check_max(feature=feat, field_number=field_number, height=height,
178               year=season, idate=initial_date, fdate=final_date,
179               scale_0_1=True)
180
181     print('done')
```

A.6 *Classification.py*

```
### Classification.py: File used to make the decision tree and evaluate
# algorithm performance

'''Imports and setup'''
import pandas as pd
import geopandas as gp
import numpy as np
import datetime as dt
from sklearn import preprocessing
from scipy import signal as sgn


def classificationDOY(row, decision_DoY_min, decision_DoY_max, val_min,
                      val_max):
    # Conditions wheat
    if row['Dates_max'].dayofyear > decision_DoY_max:
        return 1
    # Conditions barley
    elif row['Dates_max'].dayofyear < decision_DoY_min:
        return 2
    # Backscatter values conditions
    if row['Value_max'] < val_min:
        return 1
    elif row['Value_max'] > val_max:
        return 2
    return 0

def classificationCR(row, limit_CR_min, limit_CR_max):
    # Max-min normalized distance
    if row['Max-min'] < limit_CR_min:
        return 1
    elif row['Max-min'] > limit_CR_max:
        return 2
    return 0


def normalise(ww, wb):
    '''Function that sets the power values between 0 and 1'''
    min_max_scaler = preprocessing.MinMaxScaler()
    x = ww.values
    y = wb.values
    x_scaled = min_max_scaler.fit_transform(x)
    y_scaled = min_max_scaler.fit_transform(y)
    ww = pd.DataFrame(x_scaled, index=ww.index, columns=ww.columns).copy()
    wb = pd.DataFrame(y_scaled, index=wb.index, columns=wb.columns).copy()
    return ww, wb


def get_maximumIndex(column=None, height=0.7):
    '''Function tht gets the maximum index of a column. Otherwise it
    gives the maximum'''
```

```
52      try:
53          return sgn.find_peaks(x=column, height=height)[0][0]
54      except:
55          return np.argmax(column)
56
57
58  def error(row, crop_type):
59      if row == crop_type:
60          return 1
61      return 0
62
63
64  def choose(row):
65      '''Selection depending on modes'''
66      if row[row.isin(['1'])].any() and row[row.isin(['2'])].any():
67          return 0
68      elif row[row.isin(['0'])].any() and row[row.isin(['1'])].any() :
69          return 1
70      elif row[row.isin(['0'])].any() and row[row.isin(['2'])].any() :
71          return 2
72      elif row[row.isin(['1'])].any() :
73          return 1
74      elif row[row.isin(['2'])].any() :
75          return 2
76      return 0
77
78  def decision_tree(country='NLD'):
79      '''Wheat = 1, Barley = 2'''
80      wheatID = 1
81      barleyID = 2
82      decision_DoY_min = 159
83      decision_DoY_max = 179
84      limit_CR_min = 0.4
85      limit_CR_max = 0.8
86
87      np_feats = [u'mean-VH-A', u'mean-VH-D', u'mean-VV-A', u'mean-VV-D',
88                  u'mean-CR-A', u'mean-CR-D', u'std-VH-A', u'std-VH-D',
89                  u'std-VV-A', u'std-VV-D', u'std-CR-A', u'std-CR-D',
90                  u'min-VH-A', u'min-VH-D', u'min-VV-A', u'min-VV-D',
91                  u'min-CR-A', u'min-CR-D', u'max-VH-A', u'max-VH-D',
92                  u'max-VV-A', u'max-VV-D', u'max-CR-A', u'max-CR-D',
93                  u'tm2', u'tp', u'swvl1', u'swvl2']
94
95      decisions_w = pd.DataFrame()
96      decisions_b = pd.DataFrame()
97
98      features_used = ['mean-VV-A', 'mean-VV-D', 'mean-VH-A',
99                           'mean-VH-D', 'mean-CR-A', 'mean-CR-D']
100     years = [2016, 2017, 2018]
101
102     for year in years:
103         for features in features_used:
104
```

```
105             '''Power values that limit the indecision zone'''
106             val_min = -16.2
107             val_max = -14.6
108             if features.__contains__('VV-D'):
109                 val_min = -15.7
110                 val_max = -15
111             elif features.__contains__('VH-A'):
112                 val_min = -21.2
113                 val_max = -16.8
114             else:
115                 val_min = -20.2
116                 val_max = -18.6
117
118             '''Data loading'''
119             seas = 1
120             if year == 2017:
121                 seas = 2
122             elif year == 2018:
123                 seas = 3
124             idate = dt.datetime(year, 5, 12)
125             fdate = dt.datetime(year, 7, 11)
126             if features.__contains__('CR'):
127                 fdate = dt.datetime(year, 7, 23)
128
129             '''Data file'''
130             path_npy = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
131                        "Datasets/zonal_stats_{}_s{}.npy".format(country, seas)
132
133             if country is 'AUT':
134                 LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
135                        "Datasets/LPIS/ATNOEs_2016_2019_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
136                 lut = pd.read_csv("/Users/annadominguezcosta/Desktop/Documentos"
137                                   "/Erasmus/TFG/Datasets/LPIS/ATNOEs_LUT_FID_fieldID.csv",
138                                   names=['FeatID', 'FieldID'],  header=0,
139                                   delimiter=',')
140             elif country is 'NLD':
141                 LPIS_all = "/Users/annadominguezcosta/Desktop/Documentos/Erasmus/TFG/" \
142                        "Datasets/LPIS/NL----_2015_2018_E4326_MAJ_NOSMA_--BUF_DOWN.dbf"
143                 lut = pd.read_csv("/Users/annadominguezcosta/Desktop/Documentos"
144                                   "/Erasmus/TFG/Datasets/LPIS/NL_LUT_FID_fieldID.csv",
145                                   names=['FeatID', 'FieldID'], header=0,
146                                   delimiter=',')
147             else:
148                 print('FAIL')
149
150             classes_all = gp.read_file(LPIS_all)
151             classes_all = classes_all[classes_all['Field_ID'].isin(lut['FieldID'].values)]
152             classes_all.index = lut['FieldID'].values
153             classes = classes_all.copy()
154
155             np_array = np.load(path_npy)
156             np_dt = pd.date_range(dt.datetime(int(year) - 1, 11, 1),
157                                   dt.datetime(int(year), 10, 20), freq='12D')
```

```
158
159             np_idx = np.where(np.array(np_feats) == features)[0][0]
160             np_data = np_array[:, :, np_idx]
161
162             '''Dataframe preparation and get rid of NaN values'''
163             df_from_npy = pd.DataFrame(data=np_data.squeeze(), columns=np_dt,
164                                        index=classes_all.index)
165             df_from_npy[df_from_npy >= 0] = np.nan
166             df_from_npy['class'] = classes[str(year)].values
167             df_from_npy = df_from_npy.dropna(axis=0)
168
169             '''Select the desired timestamps for DOY'''
170             labels = []
171             if idate is not None:
172                 if year == 2016:
173                     idate -= dt.timedelta(days=1)
174                 idate -= dt.timedelta(days=12)
175                 labels = pd.date_range((dt.datetime(int(year) - 1, 11, 1)),
176                                        idate, freq='12D')
177             if fdate is not None:
178                 if year == 2016:
179                     fdate -= dt.timedelta(days=1)
180                 fdate += dt.timedelta(days=12)
181                 labels = labels.append(
182                     pd.date_range(fdate, dt.datetime(int(year), 10, 20),
183                                   freq='12D'))
184
185
186             '''Subset of df with certain crop type'''
187             ww = df_from_npy[df_from_npy['class'] == wheatID].copy().transpose()
188             wb = df_from_npy[df_from_npy['class'] == barleyID].copy().transpose()
189             ww = ww[ww.columns[:min(ww.shape[1], wb.shape[1])]]
190             wb = wb[wb.columns[:min(ww.shape[1], wb.shape[1])]]
191             ww.drop('class', inplace=True)
192             wb.drop('class', inplace=True)
193
194             '''Normalize time series (scale between 0 and 1)'''
195             ww_norm, wb_norm = normalise(ww, wb)
196             ww_norm = ww_norm.drop(labels)
197             wb_norm = wb_norm.drop(labels)
198
199             indexes = ww_norm.columns
200             df_w_feature = pd.DataFrame(index=indexes)
201             indexes = wb_norm.columns
202             df_b_feature = pd.DataFrame(index=indexes)
203
204             '''Apply feature'''
205             if features.__contains__('CR'):
206                 '''Calculate maximum minus minimum'''
207                 ww_feat = ww_norm.max() - ww_norm.min()
208                 wb_feat = wb_norm.max() - wb_norm.min()
209
210                 '''Create dataframe'''
```

```
211                 dict_wheat = {'Max-min': ww_feat, 'Class': np.ones(len(
212                     ww_feat))}
213                 dict_barley = {'Max-min': wb_feat, 'Class': 2*np.ones(
214                     len(wb_feat))}
215                 decwheat_df = pd.DataFrame(data=dict_wheat,
216                                            index=ww_norm.columns)
217                 decbarley_df = pd.DataFrame(data=dict_barley,
218                                             index=wb_norm.columns)
219
220                 '''Make decision of wheat, barley or indecision(0)'''
221                 decwheat_df['Dec_class'] = decwheat_df.apply(lambda
222                 row: classificationCR(row, limit_CR_min, limit_CR_max),
223                                                              axis=1)
224
225                 decbarley_df['Dec_class'] = decbarley_df.apply(lambda
226                 row: classificationCR(row, limit_CR_min, limit_CR_max),
227                                                              axis=1)
228
229             else:
230                 '''Look for the first maximum of each field and keep the datetime index'''
231                 ww_dateMaxPos = ww_norm.apply(lambda column: ww_norm.index[
232                     get_maximumIndex(column)]).values
233                 wb_dateMaxPos = wb_norm.apply(lambda column: wb_norm.index[
234                     get_maximumIndex(column)]).values
235
236                 '''Create dataframe'''
237                 dict_wheat = {'Dates_max': ww_dateMaxPos,
238                               'Class': np.ones(len(
239                                   ww_dateMaxPos))}
240                 dict_barley = {'Dates_max': wb_dateMaxPos,
241                                'Class': 2 * np.ones(len(
242                                    wb_dateMaxPos))}
243                 decwheat_df = pd.DataFrame(data=dict_wheat,
244                                            index=ww.columns)
245                 decbarley_df = pd.DataFrame(data=dict_barley,
246                                             index=wb.columns)
247
248                 '''Insert a column with maximum power values in dB'''
249                 values_max_ww = np.array([])
250                 values_max_wb = np.array([])
251
252                 i = 0
253                 for index, column in ww.iteritems():
254                     values_max_ww = np.append(values_max_ww,
255                                               column[ww_dateMaxPos[i]])
256                     i = i + 1
257
258                 i = 0
259                 for index, column in wb.iteritems():
260                     values_max_wb = np.append(values_max_wb,
261                                               column[wb_dateMaxPos[i]])
262                     i = i + 1
263
```

```
                decwheat_df['Value_max'] = values_max_ww
                decbarley_df['Value_max'] = values_max_wb

                '''Make decision of wheat, barley or indecision(0)'''
                decwheat_df['Dec_class'] = decwheat_df.apply(
                    lambda row: classificationDOY
                    (row, decision_DoY_min, decision_DoY_max, val_min,
                     val_max), axis=1)

                decbarley_df['Dec_class'] = decbarley_df.apply(
                    lambda row: classificationDOY
                    (row, decision_DoY_min, decision_DoY_max, val_min,
                     val_max), axis=1)

            df_w_feature['{}_{}'.format(features, year)] = decwheat_df[
                'Dec_class']
            df_b_feature['{}_{}'.format(features, year)] = decbarley_df[
                'Dec_class']
            decisions_w = pd.concat([decisions_w,df_w_feature], axis=1,
                                    sort=True)
            decisions_b = pd.concat([decisions_b,df_b_feature], axis=1,
                                    sort=True)
            print('Done: Year={}, Feature={}'.format(year, features))

    '''Separe by seasons'''
    decisions_w_2016 = decisions_w.filter(regex='2016').dropna(how='all')
    decisions_w_2017 = decisions_w.filter(regex='2017').dropna(how='all')
    decisions_w_2018 = decisions_w.filter(regex='2018').dropna(how='all')
    decisions_b_2016 = decisions_b.filter(regex='2016').dropna(how='all')
    decisions_b_2017 = decisions_b.filter(regex='2017').dropna(how='all')
    decisions_b_2018 = decisions_b.filter(regex='2018').dropna(how='all')

    '''Calculate the most appearing simple decisions'''
    df_modes_w_2016 = decisions_w_2016.mode(axis=1)
    df_modes_w_2016 = df_modes_w_2016.apply(lambda row: choose(row),
                                            axis=1)
    df_modes_w_2017 = decisions_w_2017.mode(axis=1)
    df_modes_w_2017 = df_modes_w_2017.apply(lambda row: choose(row),
                                            axis=1)
    df_modes_w_2018 = decisions_w_2018.mode(axis=1)
    df_modes_w_2018 = df_modes_w_2018.apply(lambda row: choose(row),
                                            axis=1)

    '''Choose among the simple decisions'''
    df_modes_b_2016 = decisions_b_2016.mode(axis=1).apply(lambda row: choose(row),
                                            axis=1)
    df_modes_b_2017 = decisions_b_2017.mode(axis=1).apply(lambda row: choose(row),
                                            axis=1)
    df_modes_b_2018 = decisions_b_2018.mode(axis=1).apply(lambda row: choose(row),
                                            axis=1)

    '''Eliminate the indecision values'''
    df_w_2016 = df_modes_w_2016[df_modes_w_2016 != 0].apply(lambda
```

```
                                                                    row:error(row, wheatID))
    df_w_2017 = df_modes_w_2017[df_modes_w_2017 != 0].apply(lambda
                                                                    row:error(row, wheatID))
    df_w_2018 = df_modes_w_2018[df_modes_w_2018 != 0].apply(lambda
                                                                    row:error(row, wheatID))
    df_b_2016 = df_modes_b_2016[df_modes_b_2016 != 0].apply(lambda
                                                                    row:error(row, barleyID))
    df_b_2017 = df_modes_b_2017[df_modes_b_2017 != 0].apply(lambda
                                                                    row:error(row, barleyID))
    df_b_2018 = df_modes_b_2018[df_modes_b_2018 != 0].apply(lambda
                                                                    row:error(row, barleyID))

    '''Join by year and calculate ER'''
    df_w_2016 = df_w_2016.append(df_b_2016, ignore_index=True)
    er_2016 = len(df_w_2016)/(len(df_modes_w_2016)+len(df_modes_b_2016))
    df_w_2017 = df_w_2017.append(df_b_2017, ignore_index=True)
    er_2017 = len(df_w_2017)/(len(df_modes_w_2017)+len(df_modes_b_2017))
    df_w_2018 = df_w_2018.append(df_b_2018, ignore_index=True)
    er_2018 = len(df_w_2018)/(len(df_modes_w_2018)+len(df_modes_b_2018))

    data = {'Success Rate': [df_w_2016.mean()*100, df_w_2017.mean()*100,
                             df_w_2018.mean()*100], 'Evaluation Rate': [er_2016*100,
                                                    er_2017*100, er_2018*100]}

    return pd.DataFrame(data=data, index=years)


if __name__ == "__main__":
    '''Country selection'''
    country = 'NLD'

    final_df = decision_tree(country=country)
    print('')
    print(final_df)
    print('')
    print('Done')
```

Glossary

List of abreviations and their meanings

- A: Ascending orbit
- AVIRIS: Airborne Visible InfraRed Imaging Spectrometer
- CR: Cross Ratio
- D: Descending orbit
- DFM: Day of First Maximum
- ECMWF: European Centre for Medium-range Weather Forecasts
- EM: Electromagnetic
- EO: Earth Observation
- ER: Evaluation Rate
- ESA: European Space Agency
- HRV-XS: High Resolution Visible - XbandSpectral
- IR: InfraRed
- LAI: Leaf Area Index
- LPIS: Land Parcel Identification System
- MERIS: Medium Resolution Imaging Spectrometer
- ML: Machine Learning
- MODIS: Moderate Resolution Imaging Spectroradiometer
- MW: Microwaves
- NASA: National Aeronautics and Space Administration
- RMSE: Root Mean Square Error
- RS: Remote Sensing
- SAR: Synthetic Aperture Radar
- SPOT: Satellite Pour l'Observation de la Terre

- SR: Success Rate
- SWV: Soil Water Volume
- SWVL1: Soil Water Volume content in Layer 1
- SWVL2: Soil Water Volume content in Layer 2
- TP: Total Precipitation
- TUW: Technische Universität Wien
- T2M: Temperature at 2 Meters high
- UPC: Universitat Politècnica de Catalunya
- UV: UltraViolet
- VH: Vertical polarization transmission and Horizontal polarization reception
- VV: Vertical polarization transmission and reception
- VWC: Volumetric soil Water Content