João Luís Rodrigues Fernandes

Analysis of Classification Algorithms for Crop Detection using LANDSAT 8 images

Dissertação para obtenção do Grau de
Mestre em Engenharia Informática

Orientadores: Carlos Viegas Damásio, Prof. Associado,
Universidade Nova de Lisboa
Susana Nascimento, Prof[a] Auxiliar,
Universidade Nova de Lisboa
Adélia Sousa, Prof[a] Auxiliar,
Universidade de Évora

Júri:

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

September, 2015

Analysis of Classification Algorithms for Crop Detection using LANDSAT 8 images

Acknowledgements

To my advisers, thank you all for the help, support and feedback throughout the year. To my friends and family, thank you for the continued support and availability. All of you were essential in helping me do this work.

Abstract

Remote sensing - the acquisition of information about an object or phenomenon without making physical contact with the object - is applied in a multitude of different areas, ranging from agriculture, forestry, cartography, hydrology, geology, meteorology, aerial traffic control, among many others. Regarding agriculture, an example of application of this information is regarding crop detection, to monitor existing crops easily and help in the region's strategic planning.

In any of these areas, there is always an ongoing search for better methods that allow us to obtain better results. For over forty years, the Landsat program has utilized satellites to collect spectral information from Earth's surface, creating a historical archive unmatched in quality, detail, coverage, and length. The most recent one was launched on February 11, 2013, having a number of improvements regarding its predecessors.

This project aims to compare classification methods in Portugal's Ribatejo region, specifically regarding crop detection. The state of the art algorithms will be used in this region and their performance will be analyzed.

Keywords: land cover, remote sensing, landsat, classification, crop detection

RESUMO

Detecção remota - a aquisição de informação sobre um objecto ou acontecimento sem efectuar contacto físico com o objecto - é aplicada numa grande variedade de áreas, desde agricultura, engenharia florestal, cartografia, hidrologia, geologia, meteorologia, controlo de tráfego aéreo, entre muitas outras. Dentro da agricultura, um exemplo de aplicação destes dados é relativo à detecção de culturas, para monitorizar as culturas existentes de forma fácil e ajudar no planeamento estratégico de regiões.

Em qualquer destas áreas, existe sempre uma busca por melhores métodos que por sua vez permitam obter maiores resultados. Por mais de quarenta anos, o programa Landsat utilizou satélites para recolher informação espectral da superfície terrestre, criando um arquivo histórico inigualável em qualidade, detalhe, cobertura e duração. O mais recente satélite foi lançado em 11 de Fevereiro de 2013, tendo algumas melhorias relativamente aos seus antecessores.

Este projecto tem como objectivo comparar métodos de classificação na área do Ribatejo, mais especificamente relativo à detecção de culturas, utilizando os algoritmos mais avançados e analisando o seu desempenho.

Palavras-chave: cobertura terreste, detecção remota, landsat, classificação, detecção de culturas.

CONTENTS

CHAPTER 1

INTRODUCTION

Remote Sensing can be defined as the process in which information is recorded/observed/perceived about an object, area, phenomenon or event without being in actual contact with them - the art or science of telling something about an object without touching it (Fischer et al. (1976)).

In modern usage, the term usually refers to the use of aerial sensor technology to detect and classify objects both on Earth itself and its atmosphere, using electromagnetic radiation reflected or emitted from the earth's surface (Campbell (2002)). It may be split into active remote sensing - when the energy measured is provided by the sensors themselves (e.g. RADAR systems), and passive, measuring ambient levels of existing sources of energy, such as sunlight.

The majority (but not all) of remote sensing is done with passive sensors, in which the sun is the major energy source (Eastman (2003)). Photography is the earliest example of this, capturing the reflection of light off earth features. Although photography is still a major part of remote sensing, newer technologies have been developed to capture wavelengths outside of the visible spectrum, such as ultraviolet and infra-red.

Land Cover is the physical material at the surface of the earth. Land covers include asphalt, grass, trees, water, soil, among many others. Land cover is inherently subject to indeterminacy and relativism, since the meaning of certain classifications can be defined in different ways. For example, in the United Kingdom, areas without trees may be sometimes classified as forest if there is intention to replant, while in Scandinavia areas with slow-growing trees might not be considered forest at all (Comber et al. (2005)).

There have been multiple attempts to define a standardized classification system, a system able to describe the complete range of land cover features independent of the scale or means used to map, such as the Land Cover Classification System (Di Gregorio (2005)) or the various MODIS - Moderate Resolution Imaging Spectroradiometer - datasets (*MODIS*), among others.

Crop Detection can be defined as the detection/monitorization of crops and crop types in agricultural land. This process can greatly benefit land owners by providing information about issues such as drought, pest infestation, detecting water stress in crops, among others, and this information may help in strategic planning regarding the area.

Remote Sensing has a growing relevance in the modern information society. It represents a key technology used in a multitude of different areas. The technological world evolves every day, both in theoretical and practical applications - the development of better algorithms, software and hardware allows us to obtain better and faster results for whichever tasks we do.

Examples of these improvements applied to remote sensing include the development of sensors that capture more information (e.g. resolution, bit depth) than before, the development of hardware that can process that information faster, and the development of classification algorithms that allow us to identify the captured information correctly.

An area that can benefit from the use of remote sensing is crop detection - identifying/monitoring agricultural crops - can greatly benefit their owners. A number of applications for crop detection exist, such as:

Crop Rotation

Crop rotation is the practice of growing a series of different types of crops in the same area in sequential seasons. Detecting appropriate crop rotation practices can be facilitated by crop detection.

Pest Prevention

Pest infestations are an important issue in agriculture, by having information about crops, these can be prevented or at least contained in an easier way.

Loan Control

Loans can be provided for farmers planning to grow certain crops. Crop detection can also be used by the loan providers to confirm farmers applied the money as agreed.

Strategic Planning

A number of strategic decisions can be improved by crop detection, such as estimating areas, future crop productions, scheduling the collection of those crops, among others.

By applying remote sensing techniques, improvements such as lower costs, faster results and higher result accuracy can be achieved. Figure 1.1 shows an example of the final results of this, with landsat 5 images. The land is classified and the results of this are displayed to the user.

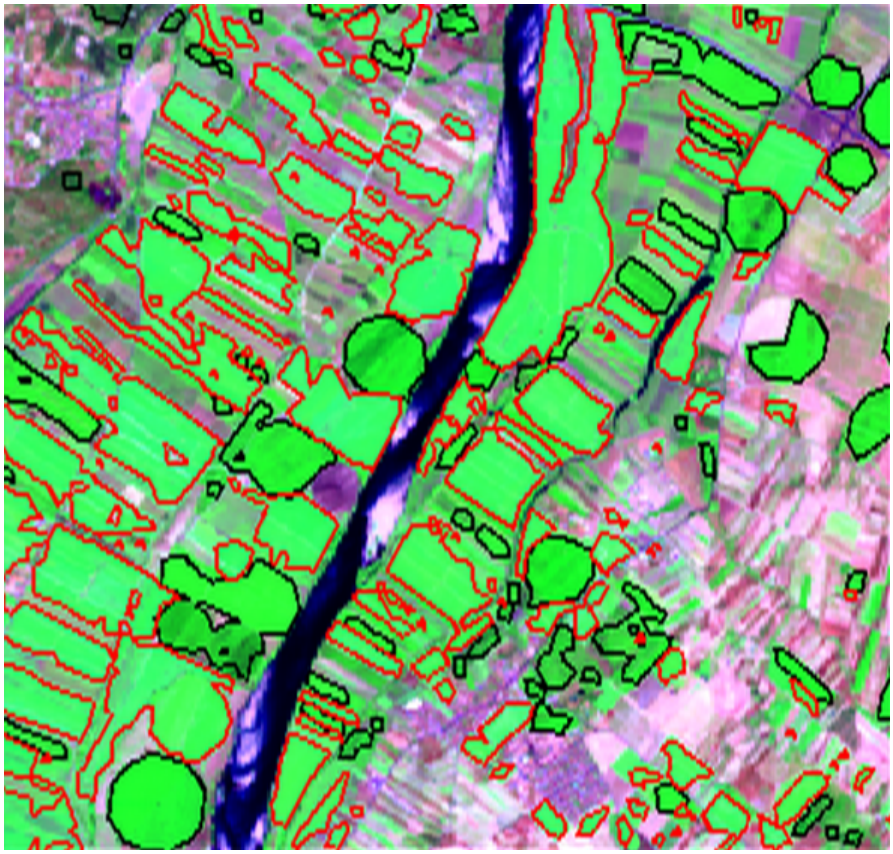


Figure 1.1: Simple crop identification. Red border is tomato, Black border is maize
Source Prof. Adélia Sousa

Region of study The Worldwide Reference System (WRS) is a global notation used in cataloging Landsat data. It enables a user to inquire about satellite imagery over any portion of the world by specifying a nominal scene center designated by PATH and ROW numbers. The WRS has proven valuable for the cataloging, referencing, and day-to-day use of imagery transmitted from the Landsat sensors. Both Landsat 8 and 5 (and others) follow the WRS-2. The combination of a Path number and a Row number uniquely identifies a nominal scene center. The Path number is always given first, followed by the Row number. The notation 127-043, for example, relates to Path number 127 and Row number 043.

The chosen study area chosen has a Path value of 204 and Row value of 33 in the WRS. The image data is made free by USGS and will be downloaded from GLOVIS, USGS Global Visualization Viewer. The study area can be seen in Figure 1.2, near the center of the image the greenest zones of the figure around the river are the agricultural land which will be the focus of our study.

Figure 1.2: The study area. Image generated by combining Landsat 8 bands 2, 3 and 4 (Blue, Green and Red).

The study will be done in the Ribatejo Province area, the most central of the traditional provinces of Portugal, with both no coastline or border with Spain. This region is crossed by the Tagus River, and contains some of the nation's richest agricultural land, making it a relevant area to study. Existing crop types in the area include potato, vineyard, rice, tomato, maize, carrot, among others.

1.1 Problem Description

In any scientific area, there is always a search for methods that can improve the accuracy of results or facilitate their acquisition, and remote sensing is no different. Currently, detecting crops is a time-consuming procedure that is achieved by ground visits that cover a smaller area than necessary and as such, may deliver inconsistent results.

Previous work in the crop detection field, in the same study area as this work, was done by Prof. Adélia Sousa to classify small land portions. This was done with a single classification method and Landsat 5 imagery. We aim to extend this work by using Landsat 8 images, more classes and multiple classification methods, as described briefly in the following section and extensively in Section 3.2.

1.2 Approach

As mentioned before, remotely sensed data can be used to significantly lower costs, speed up results and improve the reliability of these results. A system was developed where the currently best regarded techniques of land cover classification could be compared, including tuning certain parameters of each algorithm. This system provides information regarding classification accuracy, and allows for full images of classified classes to be generated. It also allows for using images from different dates in the training and testing process. Although this is a brief description, a detailed section regarding the specifics of this approach is available on Section 3.2.

1.3 Main Contributions

The proposed solution aims to contribute to the field of remote sensing as a whole, specifically regarding land cover classification, using state of the art algorithms. Regarding algorithms, we aim to provide comparisons, not only between them, but also in themselves, regarding certain parameter values that can improve the results, we also aim to provide information about how some certain small changes to the initial data, such as performing image values correction (Section 2.4) or using only a percentage of the available data influence the results. We also aim to analyze how date difference between ground truth visits and input images influence the results.

1.4 Document Organization

In addition to this introductory chapter, the rest of the document is organized as follows:

Related Work

In chapter 2 we address some topics such as the mechanisms behind remote sensing, the classification methods, some classification evaluation measures, the technologies used in this work and the relevant literature.

System Development

In chapter 3 we describe extensively the approach taken, development of the created system for classifying images, the preliminary work done, how results can be replicated, and how the data was prepared.

Experimental Study

In chapter 4 the work phases performed are described, and the obtained results for the performed work are presented, along with discussion of these results.

Conclusion and Further Work

In chapter 5 the conclusion for this work is presented, and further work that could be done as an extension of the developed approach is described.

CHAPTER 2

RELATED WORK

In this chapter, some concepts related to the mechanisms behind remote sensing are explained, the classification methods are described, along with other relevant concepts to the classification evaluation. There is also mention of the technologies used in this work, and of the relevant literature.

2.1 Spectral Signatures

When electromagnetic energy strikes a material, the interaction that follows is reflection, absorption and/or transmission. Remote sensing uses the reflected portion as data, since this is the portion returned to the sensor system. Exactly how much is reflected will vary and will depend upon many factors such as the nature of the material and where in the electromagnetic spectrum the measurement is being taken.

If we look at the measurements of this reflection over a range of wavelengths, we get a spectral response pattern. A spectral response pattern, often called a signature, is a description of the degree to which energy is reflected in different regions of the electromagnetic spectrum, displayed often in the form of a graph.

Figure 2.1 shows idealized spectral response patterns for some colors in the visible portion of the electromagnetic spectrum. A high reflectance value in the red (R) portion suggests the existence of a material that absorbed both blue (B) and green (G) wavelengths and reflected the red ones, such as a bright red piece of paper. The low value of the green wavelength in the second graph suggests the existence of a dark green material.

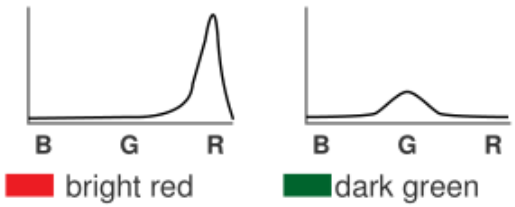


Figure 2.1: Idealized spectral response patterns for red and green. **Source** Eastman (2003)

Although the usage of the visual spectrum to generate spectral response patterns may be enough in some cases, this is not always the case. For example, figure 2.2 shows a signature for vegetation along water and dry soil. It can be seen that in the visible spectrum (approximately 0.4 to 0.8 µm) the reflected values are different but relatively close, and when the spectrum contains more information, in this case regarding infra-red wavelengths (>0.8 µm), it can be seen that the patterns are even more distinct in this region, facilitating the identification of the different classes. Analyzing and classifying spectral response patterns is the basis of land cover classification.

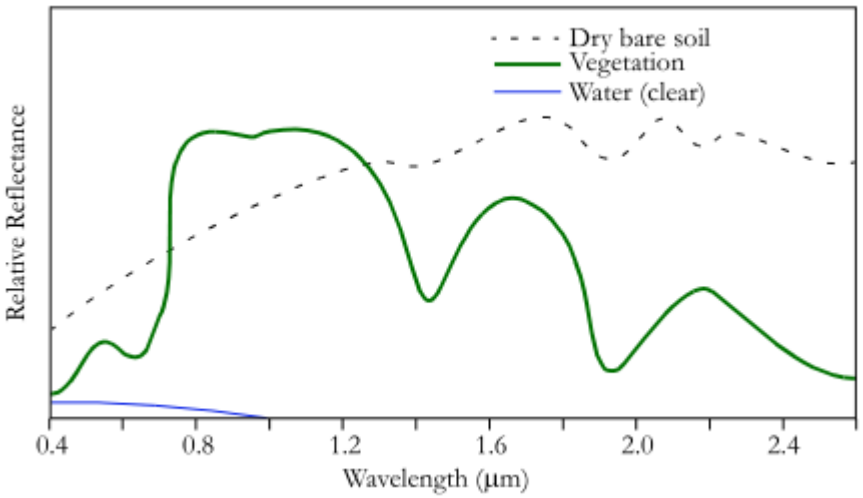


Figure 2.2: idealized spectral response pattern for vegetation along with those of water and dry bare soil. **Source** Eastman (2003)

Given the importance of bands capturing not only the visible spectrum, but other regions of the electromagnetic spectrum, it is not surprising sensor systems came to include multiple bands capturing this information. For example, Landsat 8 has eleven bands, five of which are dedicated to the infra-red zone of the spectrum, as seen in table 2.1 in page 15.

In addition to multi spectral imagery, some systems such as AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) and MODIS (Moderate Resolution Imaging Spectroradiometer) are capable of capturing hyperspectral data. These systems cover virtually the same wavelengths as the multi spectral ones, but capture data in much narrower bands. This can increase the number of bands and precision available for image classification, as well as the computing power and time necessary for them.

2.2 Sensor/Platform Systems

A variety of platforms are available for the capture of remotely sensed data. Of those, the most important ones when it comes to land cover classification, are aerial photography and satellite-based systems.

2.2.1 Photography

Aerial photography is the oldest and most widely used method of remote sensing. Cameras are mounted in aircraft flying between 200 and 15,000 m and capture a large quantity of information. Aerial photos provide an instant visual representation of the earth's surface and can be used to create detailed maps. Camera and platform configurations can be grouped as oblique or vertical.

Oblique aerial photography is taken at an angle to the ground. The resulting images give a view similar to an observer looking out of an airplane window. These are easier to interpret than vertical photographs, but difficult to use for mapping purposes.

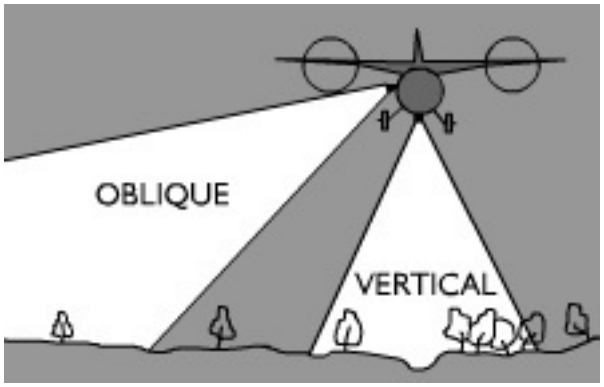


Figure 2.3: the difference between oblique and vertical photography **Source** *utexas.edu - Survey Methods*

Vertical aerial photography is taken with the camera pointed straight down (it is considered vertical photography when the angle made by the camera and the nadir (the line connecting the lens frontal point and the point on the ground that is exactly beneath the aircraft) is less than 3 degrees. Figure 2.3 illustrates the difference between both. The resulting images depict ground features in plan form and are easily comparable with maps. These are particularly useful as resources for areas where no maps are available. Aerial photos depict features often omitted in maps such as field patterns and vegetation. Comparison of old and new photos can also capture changes within an area over time.

2.2.2 Satellite-Based Systems

Although aerial photography has proven to have an important part in the field of remote sensing, the development of satellite platforms, the need to transmit imagery in digital form and the desire for highly consistent imagery have given rise to the development of satellite-based scanning systems as a major format for the capture of remotely sensed data (Eastman (2003)).

The basic logic of a scanning sensor is the use of a mechanism to sweep a small field of view - known as an instantaneous field of view (IFOV) in a west to east direction at the same time the satellite is moving in a north to south direction. Together, this movement provides the means to produce a complete raster image of the environment. Figure 2.4 demonstrates how this system works. A rotating mirror captures the energy contained in the IFOV and separates it into its spectral components. Photoelectric detectors then provide the electrical measurements of the amount of energy detected in each of its defined spectrum, for example, one such detector measures the visible red reflectance, another one measures the infra-red one, etc.

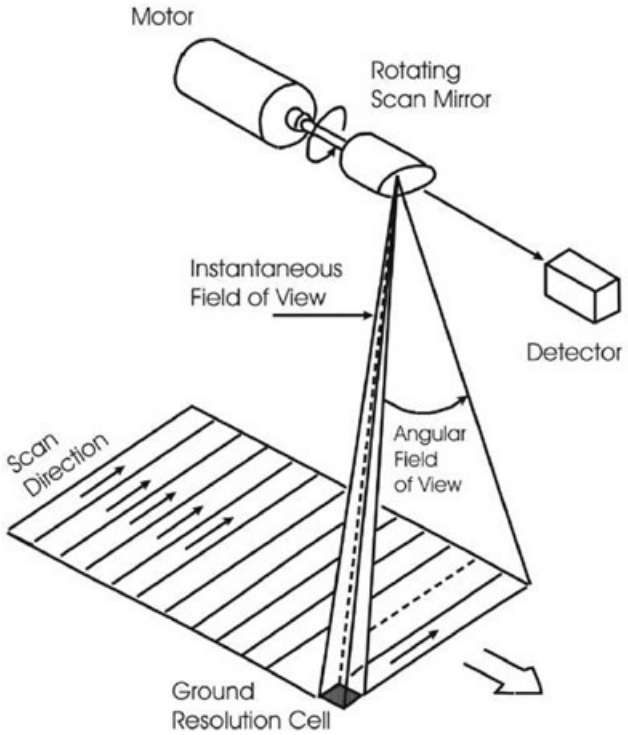


Figure 2.4: Simple mechanism of satellite-based systems. **Source** *what-when-how.com - Imaging System Types (Visible Imagery) (Remote Sensing)*

Characteristics relevant in each system are the spatial resolution, spectral resolution and temporal resolution. Spatial Resolution refers to the size of the area on the ground summarized by one pixel on the image - for example, most Landsat bands have 30 meters of spatial resolution (i.e. one pixel corresponds to a 30 by 30 meter area). Spectral resolution refers to the number and width of the spectral bands that the satellite sensor uses. Temporal resolution in this context means the rate at which images from the same location are captured with the same observation angle.

There are multiple satellite systems in operation today collecting imagery, each has one or more actual satellites. Examples of these systems are Landsat, which will be the one providing data for this study, the *Système Pour L'Observation de la Terre* (SPOT), the Advanced Very High Resolution Radiometer (AVHRR), the Moderate Resolution Imaging Spectroradiometer (MODIS), among others.

2.2.3 Landsat

The Landsat program is the longest running enterprise for acquisition of satellite imagery of Earth. For four decades this program, a joint initiative between the United States Geological Survey (USGS) and the National Aeronautics and Space Administration (NASA), has provided unique resources for those who work in various areas such as agriculture, geology, forestry, regional planning, education, mapping, and global change research. Landsat images are also invaluable for emergency response and disaster relief.

As of this writing there have been eight Landsat satellites, the first of which was launched in 1972, while the most recent one - Landsat 8 - was launched in February 11, 2013. In the context of this work, the satellite providing data will be the Landsat 8.

Landsat 8

The most recent Landsat satellite, Landsat 8 was launched by NASA on February 11, 2013, from the Vanderberg Air Force Base, California. It joins the Landsat 7 on orbit, providing increased coverage of the Earth's surface. It currently has a planned mission duration of five to ten years.

Landsat 8 has also two sensors, the first is the Operational Land Imager (OLI) capturing imagery from nine different spectral bands, seven of which are consistent with the Thematic Mapper and Enhanced Thematic Mapper Plus (ETM+) sensors

found on earlier Landsat satellites, providing some degree of compatibility with historical Landsat data. There are two new spectral bands, one for coastal water and aerosol studies (band 1), and a band for cirrus cloud detection (band 9).

The second one is the Thermal InfraRed Sensor (TIRS), built by the NASA Goddard Space Flight Center. It was added to continue thermal imaging and to support emerging applications such as modeling evapotranspiration for monitoring water use consumption over irrigated lands. The TIRS collects data in two long wavelength thermal infrared bands and has a 3-year expected life.

Table 2.1: Landsat 8 Sensor Bands. **Source** U.S.G.S. (2013)

Sensor	**Band**	**Spectral range (μm)**	**Pixel Size (m)**
OLI	1 - Coastal/Aerosol	0.43 - 0.45	30 x 30
	2 - Blue	0.45 - 0.51	30 x 30
	3 - Green	0.53 - 0.59	30 x 30
	4 - Red	0.64 - 0.67	30 x 30
	5 - Near Infrared	0.85 - 0.88	30 x 30
	6 - Short-wave Infrared 1	1.57 - 1.65	30 x 30
	7 - Short-wave Infrared 2	2.11 - 2.29	30 x 30
	8 - Panchromatic	0.50 - 0.68	15 x 15
	9 - Cirrus	1.36 - 1.38	30 x 30
TIRS	10 - Thermal Infrared 1	10.60 - 11.19	100 x 100 *
	11 - Thermal Infrared 2	11.50 - 12.51	100 x 100 *

*TIRS bands are acquired at 100 meter resolution, but are re-sampled to 30 meter in the delivered data product.

Table 2.1 describes the bands used in the OLI and TIRS Landsat 8 sensors. Like its predecessors, the Landsat 8 has a temporal resolution of 16 days, and the scenes captured have also an area of 170 km by 185 km. However, one noticeable change the Landsat 8 brings is the fact Landsat 8 sensors provide improved signal-to-noise radiometric performance quantized over a 12-bit dynamic range. This translates into 4096 potential grey levels, compared with only 256 grey levels in previous 8-bit instruments. Improved signal to noise performance enables improved characterization of land cover state and condition. The 12-bit data is scaled to 16-bit integers and delivered as such to the public.

(a) Bands 4,3,2 combined (Red, green and blue)

(b) Bands 5,4,3 combined (Infrared, red and green)

Figure 2.5: Example of images obtained by combining various bands

Figure 2.5 shows an example of different information obtained by combining different bands, and how the infrared band in the second image provides information about the crops around the Tagus river (our region of study) by being distinctively more red in those.

2.3 NDVI

Live green plants absorb solar radiation in the spectral region of about 400 to 700 nanometers, which they use as a source of energy in the process of photosynthesis. This spectral region roughly corresponds with the range of light visible to the naked eye. Plants also scatter solar radiation in the near-infrared region of the electromagnetic spectrum. Thus, green plants appear darker in this 400 to 700 nanometer region, and brigther in the near-infrared. Unlike this, other captured types of data, such as snow, clouds, pavement, dry soil, do not exhibit these properties.

As such, using this information was a natural way to identify areas containing large vegetation areas and their condition (Eastman (2003)). The normalized difference vegetation index (NDVI) was an index calculated from these individual measurements, near infra-red (NIR) and visible (red is used for the formula), in this index, which ranges from -1 to 1, areas with dense vegetation will tend to

have medium to high positive values, where other areas will have small positive values, or even negative. The index is defined as follows:

$$NDVI = \frac{(NIR - RED)}{(NIR + RED)}$$

Since after the image correction, both bands will have values between 0 and 1, the NDVI band will have values between -1 and 1. This index is used instead of other possible ones, such as simply (NIR/RED), which, despite having advantages (such as being always positive), can also have disadvantages, mainly the possibility of having a mathematically infinite range, from 0 to infinity.

Since the NDVI band is used in this field, mainly for visual interpretation of an image, it was interesting to see how relevant it would be in the actual classification process of this work, since it did not really offer any new values by itself, being a result of the above equation, only a new relation between bands.

2.4 Image Correction

There was some image preprocessing applied to all the images used in this work, these are composed of 16-bit unsigned integers with values 0 to 65536 that were converted to top of atmosphere (TOA) reflectance values, which are the actual real values recorded by the satellite - these are more sensible to small changes in land cover which can help the goal of this work. This process was performed by prof. Adélia Sousa, and is the Conversion to TOA reflectance provided by USGS on *USGS - Using the USGS Landsat 8 Product*.

2.5 Land Cover Classification Methods

There are two general approaches used to classify land cover into classes, supervised and unsupervised classification (Eastman (2003)). They differ on how the classification is actually performed. In supervised classification the software system identifies specific land cover types from the input data, through a series of known examples in the image, known as training data. With unsupervised classification, the system separates the image points into clusters that will be later classified manually.

Supervised Classification
There are three steps to supervised classification. First, a set of training points is selected for each class. This may be done using information collected by a variety of methods such as ground surveys, aerial photography, or any other source of reference data. The system is then used to develop a statistical characterization of the reflectances for each class. Finally, the image is then classified by examining the reflectances of each point and deciding which of the classes it resembles most.

Unsupervised Classification
In contrast to supervised classification, unsupervised classification requires no advance knowledge about the classes of interest. It examines the data and breaks it into the most prevalent spectral groupings, or clusters, present in the data. After this clustering procedure is done, it is then the analyst's job to identify those classes by associating a sample of pixels in each class with available reference data.

Most studies in this area, including those in Section 2.5.1 and 2.5.2 prefer a supervised approach to the unsupervised one.

The two previous concepts are not only related to land cover classification, they are fundamental to the area of machine learning as a whole (Han et al. (2006)). However, there are two other concepts more specific to land cover classification: pixel-based classification and object-based classification.

Pixel-based Classification
As the name implies, pixel-based classification is done by trying to classify every pixel of the original image independently of each other. This approach has some issues such as the fact pixels might contain more than one class, which might contribute to the misestimation of the land being higher than expected (Foody (2002)).

Object-based Classification
A recently new concept in remote sensing, object-based classification has its first step as image segmentation - grouping the pixels in the image into objects.

This approach uses an algorithm that begins with pixel sizes objects which are iteratively grown through pair-wise merging of neighboring objects based on several user-defined parameters (scale, color , shape, smoothness, compactness, etc.) that are weighted together to define a homogeneity criterion; together, these parameters define a "stopping threshold" of within-object homogeneity based on underlying input layers, and thus define the size and shape of resulting image objects (Duro et al. (2012)). After this separation into objects, this approach classifies each object by themselves, instead of each pixel.

A number of studies have been done in these areas, both comparing classification algorithms themselves and comparing pixel and object-based classification methods, they will be discussed in sections 2.5.1 and 2.5.2, respectively.

2.5.1 Algorithm Comparisons for Land Cover Applications

This section presents an overview of the different classification algorithms used for Land Cover applications. A summary is presented in table 2.2. Regarding algorithm comparison, according to the studies there is a lot of variance in the results provided.

Huang et al. (2002) compared four classification algorithms: support vector machines (SVMs), decision trees (DTs), a neural network classifier and the maximum likelihood classifier (MLC), using pixel-based image analysis of Landsat Thematic Mapper data. Their results suggested a general better performance of the SVM-based classification versus the other three algorithms.
Pal and Mather (2003) compared artificial neural networks (ANNs), DT and MLC approaches using pixel-based analysis for multi and hyperspectral data. They found that DT performs better on multispectral data, but the MLC procedure performs better on hyperspectral data. Gislason et al. (2006) investigated Random Forests (RF) as classification method of a Landsat MultiSpectral Scanner data set. They found that the RF classifier performed better than the DT classifier and was comparable to accuracies obtained by ensemble methods such as bagging and boosting. but considerably faster than these.
Carreiras et al. (2006) used SPOT-4 imagery to assess the extent of agriculture/-pasture and secondary succession forest in the Brazilian Amazon. They used four classification algorithms: quadratic discriminant analysis (QDA), simple classification trees (SCT), probability-bagging classification trees (PBCT), and k-nearest neighbors (K-NN). The study showed that PBCT and K-NN performed better than QDA and SCT.
Laliberte et al. (2006) used an object-based approach on Quickbird imagery to compare K-NN with DT. They found that DTs produced better overall classification accuracies than the K-NN algorithm. Brenning (2009) compared eleven classification algorithms using pixel-based image analysis and Landsat data in automatic rock glacier detection and found that penalized linear discriminant analysis (PLDA) yielded better results than both SVM and RF methods.
Otukei and Blaschke (2010) used Landsat data and a pixel-based approach to assess land cover changes by comparing DTs, SVMs and MLC algorithms and found that DTs performed marginally better than both others. Rodriguez-Galiano et al. (2012) used Landsat 5 imagery and a pixel-based approach to compare a RF approach with a Decision Tree one. The RF approach yielded better results.

2.5.2 Pixel-based Classification versus Object-based Classification

A mention of studies comparing pixel-based and object-based classification approaches is provided below. These comparisons suggest the latter outperforms the former when comparing classification accuracy.

Oruc et al. (2004) used Landsat 7 imagery to compare pixel and object based approaches in Zonguldak, Turkey. They found object-oriented classification (using a fuzzy classifcation method) produced more accurate results than the pixel-based one.
Whiteside and Ahmad (2005) compared the results of an object-based classification and a pixel-based one for mapping land cover. Their results showed the overall accuracy of the object-based classification (using a fuzzy classifcation method) to be marginally better than the pixel-based one. Yan et al. (2006) compared pixel-based classification (with a MLC algorithm) with object-based classification with a k-NN classifier on Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) imagery to identify potential coal fire areas. They found the accuracy of the object-based classification drastically outperformed the pixel-based one by 36.77%.
Platt and Rapoza (2008) compared k-NN and MLC for both pixel-based and object-based classifications with and without expert-based knowledge, using IKONOS imagery. Their results found that object-based classification were not better by themselves, but with the addition of expert knowledge had a better overall classification.
Zhou et al. (2008) used quickbird imagery to compare pixel and object-based classification methods for land cover change assessment and found the latter obtained better results than the former.
Cleve et al. (2008) compared pixel and object based classifications using high-resolution aerial photography to plan wildfire mitigation and achieved better results with the object-based classification (using a fuzzy classifcation method).
Mohan et al. (2009) used artificial neural networks to compare object-based classification with pixel-based classification and found object-based classification to have better results.
Castillejo-González et al. (2009) used QuickBird imagery to compare pixel-based and object-based classification in agricultural environments. Five classification methods were used (Parallelepiped, Minimum Distance,Mahalanobis Classifier Distance,Spectral Angle Mapper and MLC) They found both methods achieve

similar accuracy higher than 80%, both using the MLC classifier.
Weih Jr and Riggan Jr (2010) compared SPOT-5 satellite imagery on both object-based and pixel-based classification and found the former to have better results.
Myint et al. (2011) too used QuickBird imagery to classify urban land cover, and found better results using the object-based classifier using k-NN.
Dingle Robertson and King (2011) compared pixel-based and object-based image analysis for classifying agricultural land cover types and assessing change over time periods (1995 and 2005) using Landsat 5 imagery. They found both approaches were very similar regarding accuracy, although object-based classification had problems such as the absorption of small rare classes into larger objects. However, a post-classification intensive visual analysis suggested the object-based classification (using k-NN) depicted change more accurately than the pixel-based classification using MLC.
Devadas et al. (2012) found a distinct advantage of object-based methods over pixel-based ones using Landsat 5 and 7 data with SVM classification.
Duro et al. (2012) compared SVM, DT and Random Forest approaches to both classification methods and found object-based ones to be always better than their pixel-based counterparts and SVM to be the better overall approach.
Jebur et al. (2013) compared pixel-based (with DT and SVM approaches) and object-based (SVM) classification and found the object-based approach to be better overall.

A summary of the articles and their results is presented in table 2.2, in short, SVM, RF, DT, MLC and K-NN classification are the most frequent best results. The results table will be followed by a brief introduction to each classification method that will be used in this work.

Table 2.2: Article results summary for algorithm comparisons

Article	Research Goal	Best Classifiers
Huang et al. (2002)	General comparison	SVM
Pal and Mather (2003)	General comparison	DT, MLC
Gislason et al. (2006)	General comparison	RF
Carreiras et al. (2006)	Assess extent of agriculture/pasture and forest	PBCT, K-NN
Laliberte et al. (2006)	General comparison	DT
Brenning (2009)	Rock glacier detection	PLDA
Otukei and Blaschke (2010)	Assess land cover changes	DT
Rodriguez-Galiano et al. (2012)	General comparison	RF
Oruc et al. (2004)	General comparison	Fuzzy Classifier
Whiteside and Ahmad (2005)	General comparison	Fuzzy Classifier
Yan et al. (2006)	Identify potential coal fire areas	K-NN
Platt and Rapoza (2008)	General comparison	K-NN, MLC
Zhou et al. (2008)	Assess land-cover changes	Rule-based classification
Cleve et al. (2008)	Wildfire mitigation	Fuzzy Classifier
Mohan et al. (2009)	General comparison	Neural Networks
Castillejo-González et al. (2009)	Crop detection	MLC
Weih Jr and Riggan Jr (2010)	General comparison	Principal Component Analysis
Myint et al. (2011)	General comparison	K-NN
Dingle Robertson and King (2011)	Crop detection, Assess land cover changes	K-NN
Devadas et al. (2012)	General comparison	SVM
Duro et al. (2012)	General comparison	SVM
Jebur et al. (2013)	General comparison	SVM

2.5.3 K-Nearest Neighbor

The K-nearest neighbor (K-NN) method is one of the most fundamental and simple classification methods, it was introduced by Fix and Hodges Jr (1951). It is a very simple algorithm where every point is classified based on the pre-specified k value, the number of closest training samples in the feature space by a majority vote. For example, if $k = 1$, the object is simply assigned the class of its nearest neighbor. To prevent ties regarding voting, an odd value of k is usually chosen. Different distance metrics exist, such as Euclidean distance, Manhattan distance, Hamming distance, among others, depending on its usefulness for specific problems. The Euclidean distance is the most common approach on general problems. A simple algorithm description is as follows:

For each point p, having a known set L of data points, the distance between p and every point of L is calculated. The distances are sorted in increasing numerical order and the first k elements are picked. Finally a majority vote is done to decide the classification of p.

Figure 2.6 shows an example of a simple 2-Dimensional K-NN classification and the relevance of k values, the green point should be classified in either red triangle or blue square classes. If $k = 3$ it's assigned to the red triangle class (solid line circle), if $k = 5$ (dashed line circle) it's assigned to the blue square class.

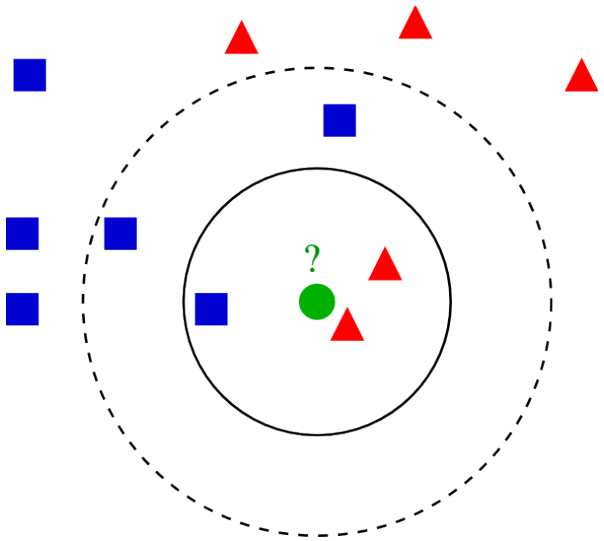


Figure 2.6: K-NN classification example **Source** *wikipedia.org - K-nearest neighbors algorithm*

2.5.4 Decision Trees

Decision tree classification (Quinlan (1986)) is a common method used for classification. Unlike other methods that use all available features at once by making a single complex decision, decision trees work by partitioning the problem into smaller subproblems and defining a set of rules to be followed sequentially down a tree-like structure (Pal and Mather (2003)).

The algorithm works by initially constructing the actual tree, by splitting the training data into subsets based on tests on the features in a process called recursive partitioning. This process is then repeated on each derived subset. This recursion is stopped once splitting subsets no longer adds value to the predictions. Labels are then assigned to the terminal (leaf) nodes.

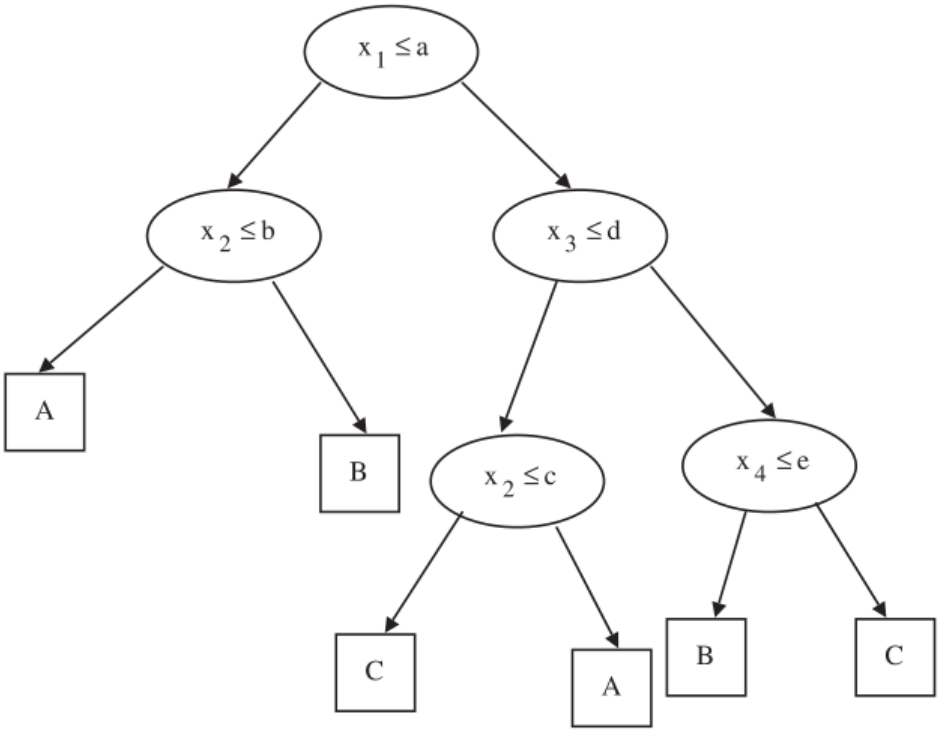


Figure 2.7: decision tree example **Source** Pal and Mather (2003)

For the actual classification, the tree structure is simply followed sequentially from the root until a leaf node is reached by analyzing the test data features. Figure 2.7 shows a simple example of a classification tree. The x_i are the feature values, A, B and C are the class values. Assuming we follow the left child if the tests are positive and right nodes if not, a point where $x_1 \leq a$ and $x_2 > b$ will be classified as the class B.

A number of parameters can be relevant to the tree construction, with the most relevant (Atkinson and Therneau (2000)) being `minsplit` and `cp` in R (Section 2.7). `minsplit` is the number of minimum observations that must exist in a node for which a split will be attempted. `cp` is the cost complexity factor, for example, a value of 0.001 defines that a split must decrease the overall lack of fit by a factor of 0.001 before being attempted. Of these, this study will use only `cp`, since it is the only one available through the caret package (Subsection 2.7.1).

2.5.5 Random Forests

The basis of Random Forests are Decision Trees, these are mentioned in the previous subsection. Decision trees can be used multiple times, such as in bagging or boosting. In boosting, successive trees give extra weight to points predicted incorrectly by earlier trees. In the end, there is a weighted vote taken for prediction. In bagging, each tree is independently constructed using a bootstrap sample of the data set, then a majority vote is taking for prediction. Liaw and Wiener (2002)

An additional layer of randomness is added by random forests, proposed by Breiman (2001). In a random forest model, during training, a number of decision trees are created, each having as basis a different training data subset by resampling randomly the original training dataset with replacement. Along with this random subset of the training data, the features each tree receives are themselves a random subset of the original features. This allows for greater classifier stability and increases classification accuracy (Breiman (2001)).

For classification, each point is simultaneously "pushed" through all trees until it reaches the corresponding leaves, predictions then are made based on a majority vote done by each tree. This method has two pre-specified parameters - the number of variables in the random subset at each node - `mtry`, and the total number of trees. Of these, this study will use only `mtry`, since it is the only one available through the caret package (subsection 2.7.1).

2.5.6 Support Vector Machines

Support vector machines (SVMs) is a supervised learning technique used in classification introduced by Cortes and Vapnik (1995). In its simplest form, SVMs are linear binary classifiers that assign a given test sample a class from one of two possible labels by separating the data through a hyperplane (line in 2 dimensions, plane in 3 dimensions, etc). An instance of the data to be labeled in the case of remote sensing is for example, an individual pixel. Figure 2.8 identifies an example of this scenario in 2 dimensions.

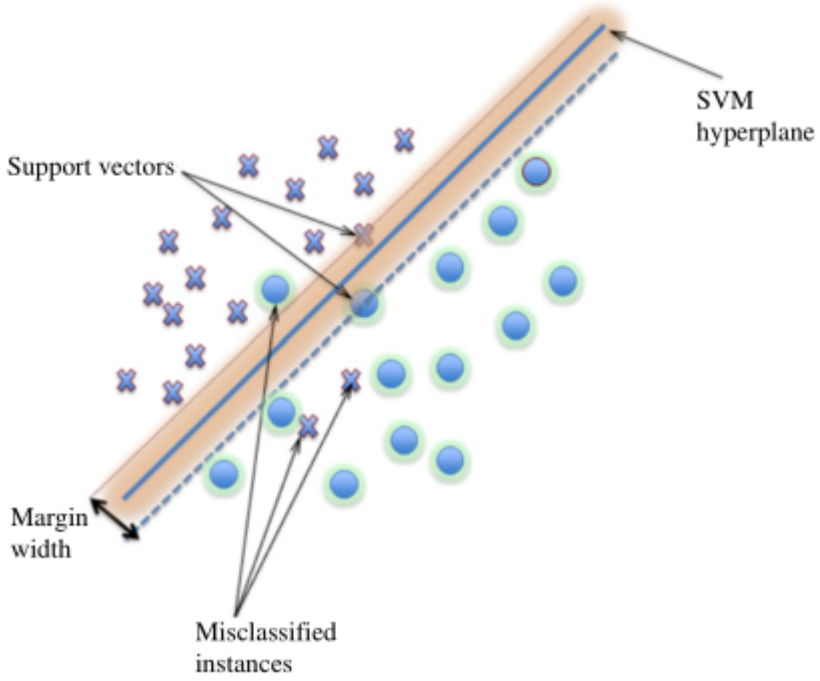


Figure 2.8: simple support vector machine example. **Source** Mountrakis et al. (2011)

However, there are situations where the data isn't separable in the current dimensions. Figure 2.9 exemplifies this, on the left, in 1-D the blue and orange points cannot be separated, however, by applying the transformation $x \rightarrow (x, x^2)$, we get the data set shown on the right, from which a hyperplane can be found that separates the data correctly. One such example is the line shown in red. This is called a kernel - a transformation of the input data that allows algorithms such as SVMs to process it more easily. There are a number of existing kernel functions. In this work the one used will be the (Gaussian) radial basis function kernel.

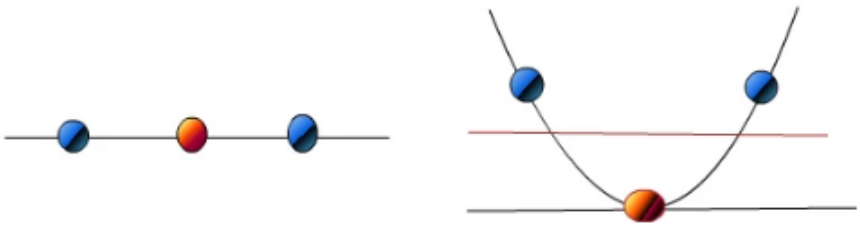


Figure 2.9: a harder case **Source** *quora.com - What are Kernels in Machine Learning and SVM?*

SVMs have proven a very reliable method in processing of remote sensing data (Mountrakis et al. (2011)). Since remote sensing will almost always deal with multiple classes and not two, adjustments are made for the SVM algorithm to operate as a multi-class classifier. The dominant approaches for this are constructing a number of binary classifiers, such as the one-vs-all approach, where the classifier with the highest output function assigns the class, and the one-vs-one approach, where every pair of classes is tested against each other, and classification is done by a voting strategy.

The parameters for this Gaussian kernel in particular are C - Cost and γ - gamma, C controls the cost of misclassification on the training data. The gamma parameter defines the influence of a single training point, with low values meaning high influence, and high values meaning low influence.

2.5.7 Maximum Likelihood

The maximum likelihood classification method is a very simple and common method, ubiquitous in every geographic information systems software, such as ESRI's ArcGIS suite (*Esri*) and Exelis' ENVI suite (*Exelis Visual Information Solutions*). This method is described more extensively by Richards and Richards (1999).

The algorithm works by taking into account the different classes and the training data, considering both the variances and covariances of the class signatures. A class can then be characterized by the mean vector and the covariance matrix. After that, each pixel of the image is then assigned to the class to which it has the highest probability of belonging.

2.6 Evaluation Measures for Data Classification

2.6.1 Cross-validation

Cross-validation is a model validation technique for assessing how the results of a model will perform in a new data set. In a prediction problem, a model is given a dataset of known data in which to train (training dataset), and a dataset of data against which the model is tested (testing dataset). The model "learns" with the training dataset, and then receives the testing dataset, outputting the predictions, which are then compared with the real classes of this dataset, yielding results such as accuracy.

The goal of cross-validation is to define a "test" dataset to test the model in the training phase, in order to try to limit issues such as overfitting, and give an insight on how the model will probably perform in the real testing dataset, and, if the model has parameters, cross-validation also allows for these to be tuned, so the best ones can be used in the testing phase. For example, in this work, cross-validation is performed for k-NN (and others), where different values of k are tested using cross-validation and the best of these is chosen as the "true" k value to use for the actual testing phase. These k's do not mean the same thing, in k-NN, k means the number of neighbors and regarding cross-validation, it means the number of folds to split data in.

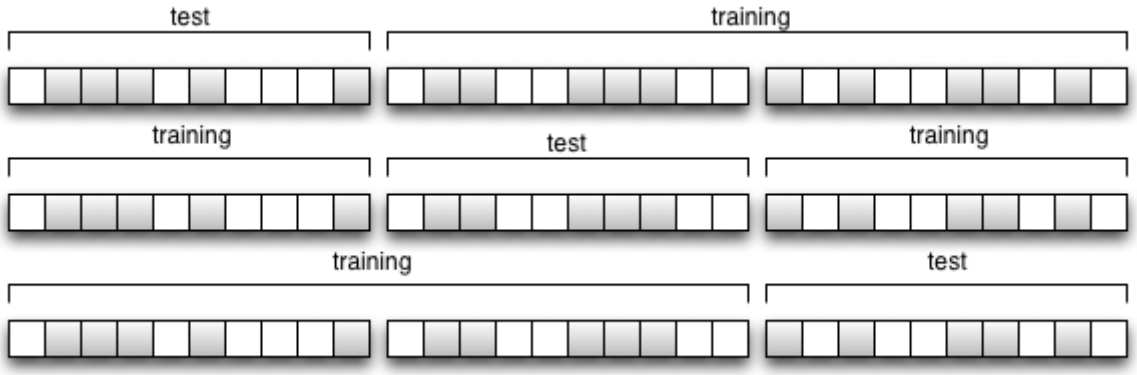


Figure 2.10: Example of 3-fold cross validation. **Source** *in.ed.ac.uk - Dispel Tutorial 0.8 documentation - k-fold cross validation*

There are many different ways to partition the training dataset to perform this analysis. In this work, 3-fold ($k = 3$) cross validation will be used. In k-fold cross validation, the original training data is split into k equal subsample sets, where one is retained as the validation data for testing, and $k - 1$ used as training data. This is repeated k times, with each of the subsamples being used exactly once as validation data. The k results from this analysis can then be averaged to produce a single estimation. Figure 2.10 shows an example of 3-fold cross validation in a dataset containing originally 30 training samples.

2.6.2 Confusion Matrix

A confusion matrix is a specific table layout that allows for visualization of an algorithm's performance, by comparing the results of the predicted classes with the true class labels - each column represents the instances of a predicted class while rows represent the instances in an actual class, or vice-versa. This work uses the former whenever a confusion matrix is available. The name confusion matrix stems from the fact these tables allow for easy interpretation of whether two classes are being misclassified as one another (confusing the classes).

Table 2.3: simple three-class confusion matrix

	Cats	Dogs	Rabbits	Total
Cats	5	3	0	8
Dogs	2	3	1	6
Rabbits	0	2	11	13
Total	7	8	12	27

Table 2.3 shows an example of a confusion matrix for a system trying to classify animals into cats, dogs or rabbits, there are 27 samples, 8 cats, 6 dogs and 13 rabbits. It can be seen that from the 8 actual cats, the system predicted 5 correctly and 3 to be dogs, for example, also it predicted one dog to be a rabbit, among other predictions. All correct guesses are located in the diagonal of a table making it easy to inspect the table for errors, which are the values outside of the diagonal.

There are a number of values that can be obtained from a confusion matrix, this work will make use of precision and recall. Precision is the fraction of predicted instances that are relevant (correct). For example, precision for the cat class would be about 0.71, or $5/7$. Recall is the fraction of relevant instances that were retrieved, which, for the cat class would be 0.625, or $5/8$.

2.6.3 Cohen's Kappa coefficient

Cohen's Kappa coefficient (Cohen (1960)) is a statistical measure of inter-rater agreement for qualitative (categorical) items. It is generally thought to be more robust than simple percent agreement calculation, since it takes into account the agreement occurring by chance. The equation for kappa is

$$k = \frac{(p_o - p_e)}{(1 - p_e)}$$

where p_o is the relative observed agreement and p_e the hypothetical probability of chance agreement. If the raters are in complete agreement then $k = 1$, while if there is no agreement among the raters other than what would be expected by chance, $k < 0$. A simple two class example follows, with a confusion matrix in a example system trying to classify cats and dogs.

Table 2.4: simple two-class confusion matrix

	Cats	Dogs	Total
Cats	10	5	15
Dogs	7	8	15
Total	17	13	30

From the confusion matrix, we can see there is a total of 30 classified instances, where 15 (10 + 5) were cats, 15 were dogs (7+8), and there were 17 instances classified as cats, and 13 as dogs. p_o in this case is simply the total number of correctly classified instances in the entire matrix, divided by the total number of instances, as such: 0.6 $((10 + 8)/30)$.

To calculate p_e, we first multiply the marginal frequency of cats for one "rater", by the marginal frequency of cats for the second "rater", and divide by the total number of instances. The marginal frequency for a certain class by a certain "rater" is just the sum of all instances the "rater" indicated were that class. In this case, we have 15 cats and 17 classified cats, resulting in a value of 8.5 $((15 * 17)/30)$, this is also done for the second class, yielding a result of 6.5 $((15 * 13)/30)$. The final step is to add these values together and divide them by the total number of instances, yielding a p_e value of 0.5 $((8.5 + 6.5)/30)$.

Kappa is then calculated, yielding a value of 0.2, $(0.60 - 0.50)/(1 - 0.50)$. This measure was included since it was, along with simple accuracy, used in the great majority of studies in table 2.2.

2.7 Used Technologies

We will make use of R, a multi-platform open-source language and software for statistical computing (R Core Team (2013)). It is a highly used tool in this field and has several add-on packages that will be helpful in this work. Esri's ArcGIS suite (*Esri*) will also be central to this work.

2.7.1 R Packages

The main packages used in this work were: *rgdal*, *caret*, *raster*, *randomForest*, *rpart*, *kernlab*

Additional packages not mentioned might have been used, possibly as dependencies of these mentioned, in any way, a comprehensive list of all the packages installed in the system is provided for clarity, and is available in appendix A.

Caret

A fundamental package for this work was the *caret* package (Kuhn (2015)), developed mainly by Max Kuhn, the *caret* package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for data splitting, preprocessing, feature selection, model tuning using resampling, among others. Since R has many different packages with different functions that have different syntax for their arguments, this package provided a uniform interface of the functions themselves, facilitating a number of tasks important for this work.

2.7.2 Shapefiles

The shapefile format is a highly popular geospatial vector data format used in geographic information system software. This format was developed and is maintained by ESRI, as a specification for data interoperability between ESRI and other GIS products (*Esri*). This format can describe points, lines and polygons, to represent any kind of information wanted as items.

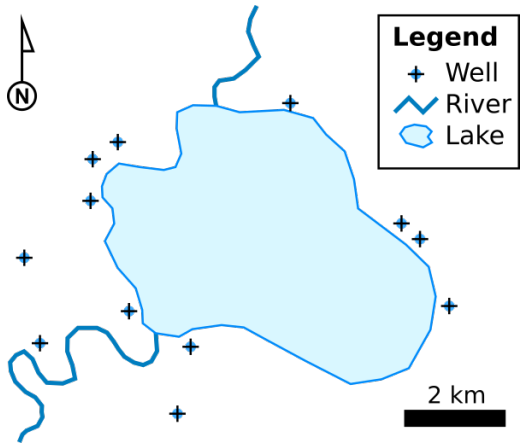


Figure 2.11: Example of information represented by shapefiles. **Source** *wikipedia.org - Shapefile*

Figure 2.11 shows an example of information represented by a shapefile. In it, it can be seen that points represent wells, lines represent rivers and a polygon represents a lake. Each item usually has attributes that have some information about it, such as name or temperature. The shapefile format does not consist of only one file, but a set of files, some optional and some mandatory that have a common prefix, stored in the same directory.

This is a fundamental format used for this work, it is the format in which information is saved, to be used by the R scripts (for this work, polygons were used, where all the points inside were defined as one of the classes used). From it, and the image layers, the necessary data for this work is obtained, by extracting, for each point inside of the polygon the correspondent values in each of the image layers.

2.8 Summary

It can be seen that remote sensing is a large field where a number of studies have been done. Regarding classifiers, the best results for land cover classification usually arise from Support Vector Machines, Decision Trees, Random Forests, Maximum Likelihood Classifiers and the K-Nearest Neighbor algorithm, as can be seen in table 2.2, both when using pixel-based and object-based approaches. Regarding the comparison between these approaches, it can be seen that generally the classification results are better when using the object-based classification. However, the pixel-based approaches still manage to obtain high accuracy percentages. For this work, those classification methods were chosen, and a pixel-based approach was used, since object-based ones introduce unnecessary layers of complexity to this work.

Regarding the tools, ArcGis and R (with packages) were chosen, since they're both widely used fields in this work, ArcGis would allow for a proper environment for preparing the data - both the shapefile and the image layers, and R would then be fit for processing this data as wanted, containing a number of packages, such as the previously mentioned *caret*, which allowed for a uniform training environment for all the classification methods.

CHAPTER 3

SYSTEM DEVELOPMENT

In this chapter, the development of the created system will be described. Initially, information will be provided about the environment where the study was conducted, as well as how the results can be reproduced. After that, the data preparation performed, where raw data is transformed into acceptable input to the study, will be extensively described. Finally, the work performed in R will also be described, the main work phases separated into single and multiple image parts.

3.1 Preliminary Work

As of this document's writing, the preliminary work done on this work was a ground survey made on July 2013 by Prof. Adélia Sousa and Prof. José Rafael Silva, that assigned a classification to some points of the study area. These points are the fundamental basis of our work since they would then be used as training/testing data.

3.2 Approach

1. Landsat 8 imagery of our study area from July 2013 (the same month as the previously done ground survey) was acquired from online archives. A crop of the original images was done, since they are much larger than necessary for this study and it would considerably slow the following steps.

2. About nine thousand points of ground truth data were obtained based on a previously done ground survey around that time, that assigned a classification to some points in the area. The bands used are bands 2-7 according to table 2.1, capturing the visible and infrared zones of the electromagnetic spectrum, since those are the best bands to use for land cover classification (*USGS - What are the best spectral bands to use for my study?*).

3. The main crop classes used were:

   - Potato
   - Vineyard
   - Rice
   - Tomato
   - Maize

   Other classes were also be used to facilitate the distinction between land cover types, described in Subsection 3.4.1.

4. The classification was done using the R as mentioned in Subsection 2.7, since it is a notorious tool in this field that provided us with implementations of the classification methods chosen. In R, several packages were used to classify the image data, five classification algorithms were used:

   - Support Vector Machines
   - Random Forests
   - Maximum Likelihood Classifier
   - K-Nearest Neighbor
   - Decision Tree

   For each of these methods, whenever possible, their parameters were tuned in order to discover which ones provide the best results in the study area. Analysis of the results was then performed to compare algorithm performance, both regarding overall accuracy percentage and Cohen's kappa coefficient.

3.3 System Information/Results Replication

The largest portion of this work was done using the R language, since, as mentioned in Section 2.7, it is a very appropriate language to perform this kind of analysis on, it is widely used in this field, and it contains a number of useful packages (e.g. loading the data from the shapefile, classification algorithms). In this section the packages and folder structure used will be explained, along with a brief description of each file and its function. The version used was R version 3.1.2 (2014-10-31) under Windows 7 x64, RStudio version 0.98.1102 was used as an IDE since it provides a helpful interface compared to running R in a simple command line interface.

3.3.1 Folder structure

A simple folder structure is as follows:

```
thesis
├── inputs
│   ├── shapefiles
│   └── <all layer folders>
├── mlc
└── outputs
```

The *thesis* folder is the main folder, containing the *inputs*, *mlc* and *outputs* folders. It has the main R scripts, the *mlc* folder has the Maximum Likelihood Classifier implementation, the *inputs* folder contains the *shapefiles* and the image layers' folders for each date. The *shapefiles* folder contains the shapefiles, and the each layer folder contains the seven layers (blue, green, red, near-infrared, short-wave infrared 1, short-wave infrared 2 and NDVI) for that date. The *outputs* folder contains the generated images. These were mostly created for the sake of organization, and as such, there is no issue in editing the R scripts to load files from different locations, if necessary.

A brief description of the R scripts contained in the main folder follows, for more information, all the scripts are highly commented and available in the appendix.

loader.R

Loads the shapefile and image layer values, extracts the values of the points inside the shapefiles for those images, and joins them in a data frame ready to be used by the rest of the scripts.

plotHistogramas.R

Plots the imagery seen in Subsection 4.2.1 into box-plots.

si_decisionTree.R

The decision tree algorithm is applied, before this, loader.R is ran, data is split into training and test data, then split for cross-validation, then the grid is set up. After, results are collected and the full image is generated.

si_kNN.R, si_MLC.R, si_randomForest.R, si_SVM.R

Same as above for the other algorithms.

percentages.R

Yields the results of Subsection 4.2.3.1 - the study on how important is using 100% of the data versus 25,50 and 75%.

multi_loader.R

Same as loader.R, to be ran in the following scripts, loads the new image data for testing (so all the data from loader.R is for the training set).

multi_randomForest.R

Runs loader.R, sets the data as training set, trains it, runs multi_loader.R, sets the new data as test set, collects results and generates the new image.

multi_SVM.R

Same as above, for SVM.

mlc.R

This file is in the *mlc* folder, it is merely an adaptation of another package's algorithm, containing a method for training and testing. Details on this file are below.

Maximum likelihood classifier adaptation
Unlike the four other methods, the Maximum Likelihood Classification algorithm was not available on the *caret* package, this would not be a problem in itself since this method has no parameters to actually perform tuning on (which was the main reason for using the *caret* package), but there were no other implementations of this method found that would work with the same data format as used by our scripts.

The *rasclass* package contained one method that supported a different input format - however, changing the data to this format just to use it on this classification algorithm was deemed an unnecessary step, more convoluted than simply trying to modify the existing algorithm to fit our needs, and, as such, its maximum likelihood classification function was extracted and adapted to be used with our data formats, the *rasclass* package licenses are the GNU General Public License version 2 and 3 so there is absolutely no issue regarding modification of the code.

3.3.2 Result replication

The *plotHistogramas*, *si_** and *percentages* scripts can all be ran independently. This was intended and the reason there is some similar code in these. There are no changes necessary to be made to reproduce the Section 4.2.1, Section 4.2.2 and 4.2.3.1 results, simply running the scripts is enough. To reproduce the results from the other complementary studies (sections 4.2.3.2 and 4.2.3.3, the loader.R file is modified to receive the original bands with no image correction in the former, and to ignore the last band (NDVI) in the latter, and then the *si_** scripts can be ran.

For the multi-image phase, it's merely a matter of modifying the *loader.R* and *multi_loader.R* to set up what image dates are wanted from the available ones, and then running the *multi_** files.

3.4 Data Preparation

There is a need for some data preparation before the actual classification is done in R. This work can be divided into two parts, regarding the shapefile and the image layers used.

3.4.1 Shapefile Preparation

Regarding the shapefile, the original shapefile was compiled by Prof. Adélia Sousa as mentioned in Section 3.1, it was the result of ground visits within days of the satellite images were acquired, to obtain the best accuracy possible. This shapefile, however, needed some modifications because it contained samples from classes not relevant to this work (i.e. not the five main classes from 3.2), mostly because their sample size was too low both in the collected data and in the study region.

These classes were deleted from the original shapefile. This was the first step regarding the shapefile modification. All the shapefile modifications mentioned in the following paragraphs were done in ArcGIS' training sample manager - this creates a specific shapefile format that the R scripts process (for example, the R scripts assume the class name column is called *Classname* in the shapefile, and the spectral values are present on columns numbered 7 to 12, or 13 with NDVI), however, if necessary, the R scripts can be very easily edited to adapt to other custom shapefile formats. When classes were added it means there were a number of polygons created that contained enough points to be useful in classification, these were then merged and classified as one class.

After this removal, there was a need to add extra classes aside from the five "main" crop classes, to avoid classifying large areas wrongly. As mentioned in 3.2, one that was certainly necessary was the "River" class, since the main area of study is crossed by the Tagus River, and it could also help identify smaller bodies of water. Two other classes were added in this second step. The first was "Stover", added because there were a lot of samples collected from this class, and while not exactly a crop class by itself, it can be very useful in this field. The second one was an "Other" class, mainly used to identify buildings, houses and roads existent closer to the boundaries of the study area.

This modified shapefile was used during some time for the classification (with roughly the same results as the final one), but ultimately it was necessary to make some changes for a better visual interpretation of the image, discovered after the generation of the classified images in R. There were two main issues:

Misclassification of large parts of vegetation near the image boundaries

Large parts of vegetation that had no specific class assigned were being classified as "Vineyard", this had no impact on the classification accuracies in itself, and as such was not an issue that could be detected during the result analysis of the classification algorithms. It was only after visual interpretation of the image that this became noticed as an issue. To fix this issue another class was created to classify this extra vegetation, that fixed most of these visual issues.

Misclassification of small roads between crops

Small roads between crops were also being misclassified, another issue only uncovered after visual interpretation of the results. Again, this did not impact the classification accuracy, but for the sake of visual interpretation it was a problem that could be addressed. These were also mostly being misclassified as "Vineyard". This was a harder problem to solve, mainly because adding training samples of these small roads was very hard, since the image pixels corresponded to 30x30m areas, these roads were barely visible in some areas. Some small polygons were created and temporarily made an extra "Small Roads" class, but eventually it was merged with the "Other" class since it provided roughly the same information.

The final shapefile contained, then, information about nine classes, the "main" five - Maize, Vineyard, Rice, Tomato, Potato, and four extra classes - Stover, Vegetation, River and Other.

3.4.2 Layers Preparation

The first step in processing the layers was to apply image correction to each layer. This was done by Prof. Adélia Sousa.

After that, an important step was to delete most of the image, the original image obtained from USGS was too big and most of it was of no relevance to this work, compared to the area of which we have training samples. Classifying the whole image would, then, be an unnecessary process that could be avoided, because it would provide no relevant information, and mainly because it would dramatically increase the generation time of the final images.

A rectangle-shaped crop was applied to all the image layers, making sure it covered all the training sample polygons but not much more than that. This was done in ArcGIS by creating a rectangle-shaped shapefile, and then using the Spatial Analyst Toolbox - Extract by Mask option, with each layer and this rectangle shapefile as inputs, receiving the cropped layer as output.

Finally, the NDVI (Section 2.3) band was generated, this was also performed in ArcGIS using its Raster Calculator option. Since this band was "artificially" generated there was some curiosity regarding its impact in the classification procedure, a simple study of the relevance of the NDVI band was performed.

These were the steps taken to generate the input files that R uses for its classification. The shapefile modifications are described above, and the layers procedure was initially applied to the layers of one image (the one from July 22, 2013), and then for the temporal analysis of this work (Section 4.1.2) were applied to the layers of the new images.

CHAPTER 4

EXPERIMENTAL STUDY

In this chapter, the work phases performed are described, and the obtained results for these are presented, along with discussion of these results.

4.1 Work Phases

4.1.1 Single-image phase

4.1.1.1 Pre-Classification

Before the classification was actually done, a simple script was created to plot the box-plots for each different class, for each band (file *w0_plotHistograms.R*). This was not a part of the actual classification procedure, but instead, provides a first look at how different the spectral signatures for the different classes were, and a way to visually extract some preliminary information from these. The results of this script are nine box-plots (or, if applied to other shapefiles, N box-plots, N being the number of different classes).

4.1.1.2 Classification

In the first main part of this work, the five classification algorithms were applied to our data. This data contains about nine thousand samples, a very large number compared to most works in the related work (table 2.2). This was achieved by creating five scripts, one for each algorithm. The main steps that describe the action taken by these scripts follows:

Load Data

The *loader.R* file is executed. This file loads a shapefile, a number of layers, and for each point of the shapefile the correspondent layer values are extracted and put into an entry with the point's class.

Split Data

The data is split into training and test sets, 66% for the training set, 34% for the test set. This is done with stratified sampling (i.e. the proportion of each class' samples in the original data is maintained for both these sets).

Split Data (2)

The data is split once more to create three folds for cross validation during the training phase.

Create Grid

The values of the parameters to be tuned are defined, this is called creating a grid by the *caret* package. For example, for the k-NN algorithm, k is defined as the sequence from 3 to 21 with an interval of 2.

Train/Tuning

The algorithm is ran for each possible grid combination while performing 3-fold cross-validation. The best result is then chosen to be used for the test set. For MLC the whole training data is simply fed to the algorithm.

Test

The test set is fed to the algorithm and predicted classes are received. Those are compared to the actual classes and results such as accuracy/Kappa and confusion matrices are obtained.

Predict Image

The whole image is fed to the algorithm and predicted classes are received, those create then a new image with each pixel having a value from 1 to 9 (or the number of different classes in the shapefile), although it's obviously not guaranteed it will have all of them, some classes might not be predicted at all, in some extreme cases.

4.1.1.3 Complementary studies

A number of studies was performed to analyze variables that could have influenced the previous results:

Reduction of the initial data

Since a high number of data points was available, it was interesting to study how the algorithms performed with percentages of this original data. This was done by using 5, 10, 25, 50, 75 and 100% of the original data and comparing the results.

Classification without image correction applied

The importance of the image correction process was also something to study, this was performed by loading the original image layers, instead of those where image correction was performed.

Not utilizing the NDVI band

It was also interesting to study how much the NDVI band contributed to the results, this was simply done by ignoring the NDVI band layer on the *loader.R* file

4.1.2 Multi-image phase

After applying the classification methods to the same image from which the data was extracted, they were applied to a multitude of different images, about two and four weeks around the original image's date, to analyze how these methods were performing over time. The classifiers were trained using the ground truth data used in the single-image phase, and then the new image was fed to the program to be classified. The accuracy results were then calculated by classifying the same data points existing in the ground truth and comparing the classes of those points with the classified values. After that, the complete image was generated. For this section, only RF and SVM algorithms were used since they were ones where better results were had in the previous section.

This section does not separate the same date's data into training/test sets, the training set is the original images' values, and the test set is the new images' ones, about nine thousand values each, around three times more than the number of test values used in the single-image phase. Since this process used the same ground truth data as in the initial phase, the results are expected to be "worse" - for example, a point that was Maize on the original image and its crops would be collected before the next image was taken, could be "correctly" classified as Stover but in fact it would be marked incorrectly since the original data had the Maize class on that point. The further the image dates are from the original image's date, the worse their classification accuracies are expected to be.

This process was then extended to use other dates' imagery to classify these other dates, mediocre results were also expected here, based on the difference between the dates. Finally, two images from 2015 were used as test data, with the July 22, 2013 images serving as training data, to obtain classified images for this year, which were used to compare the results with actual ground truth data, as mentioned in 4.5.

4.2 Single-image phase

In this section the results of the study performed on the image from July 22, 2013 will be presented, the spectral signatures for each class, the classification results, and other studies performed.

4.2.1 Pre-Classification

Spectral signatures for the nine classes will be presented, all the plots' y limits are $(0, 1)$ except for the River class, which contains negative NDVI values and as such has different limits. This was hard-coded in R for better visualization.

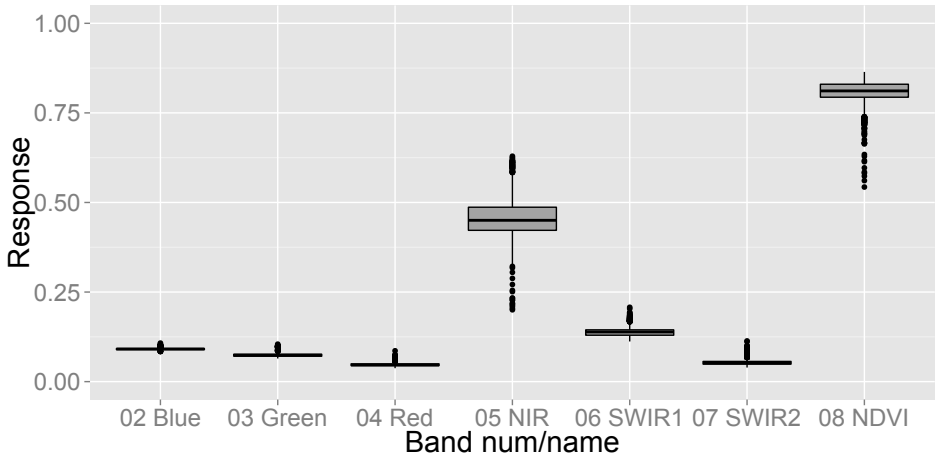


Figure 4.1: Spectral Responses For Maize Class

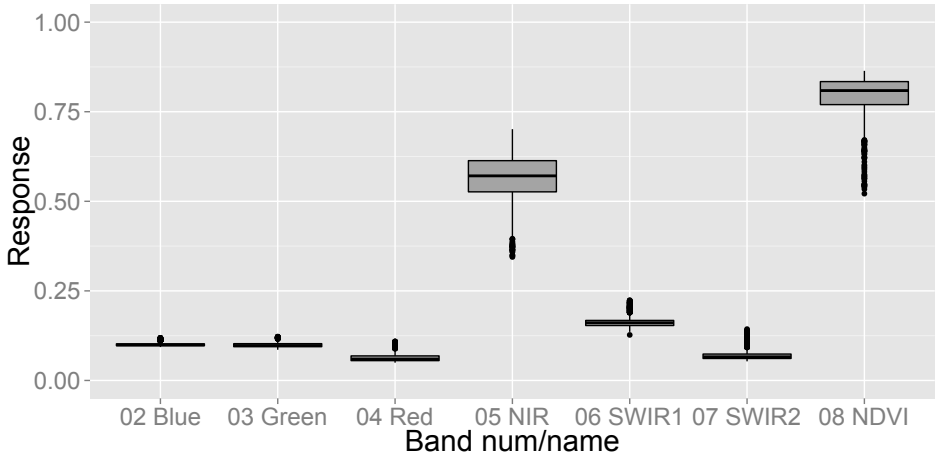


Figure 4.2: Spectral Responses For Tomato Class

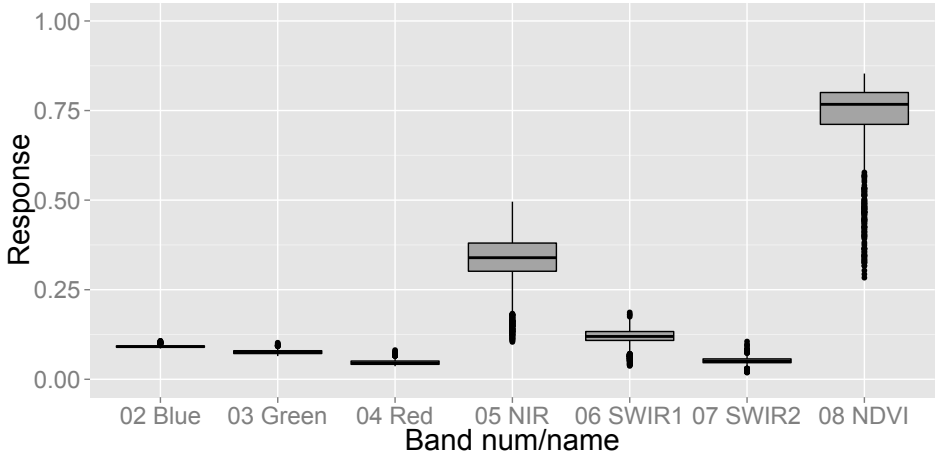


Figure 4.3: Spectral Responses For Rice Class

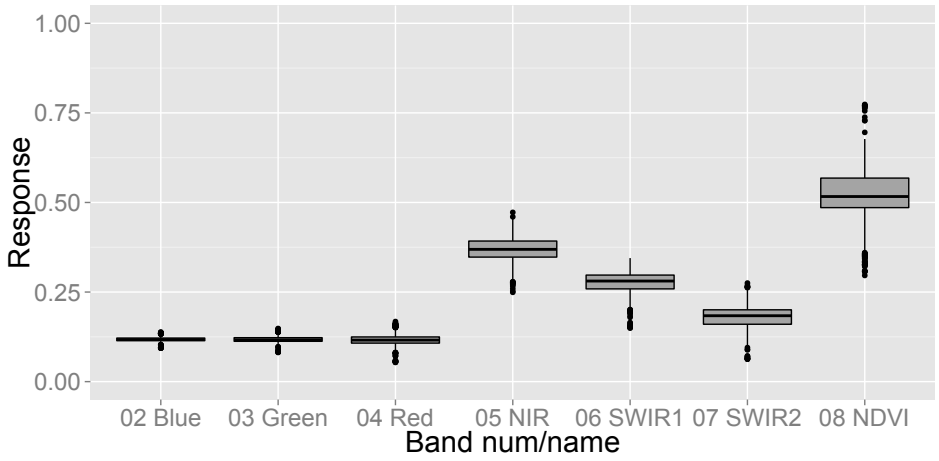


Figure 4.4: Spectral Responses For Vineyard Class

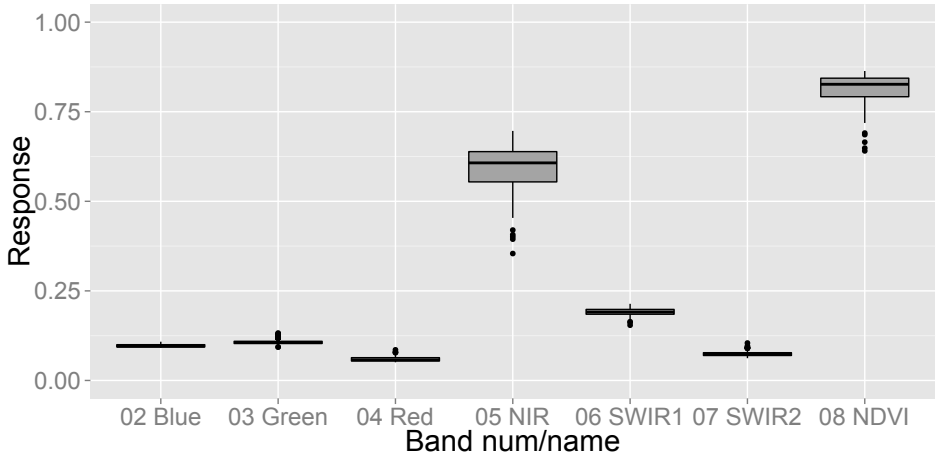


Figure 4.5: Spectral Responses For Potato Class

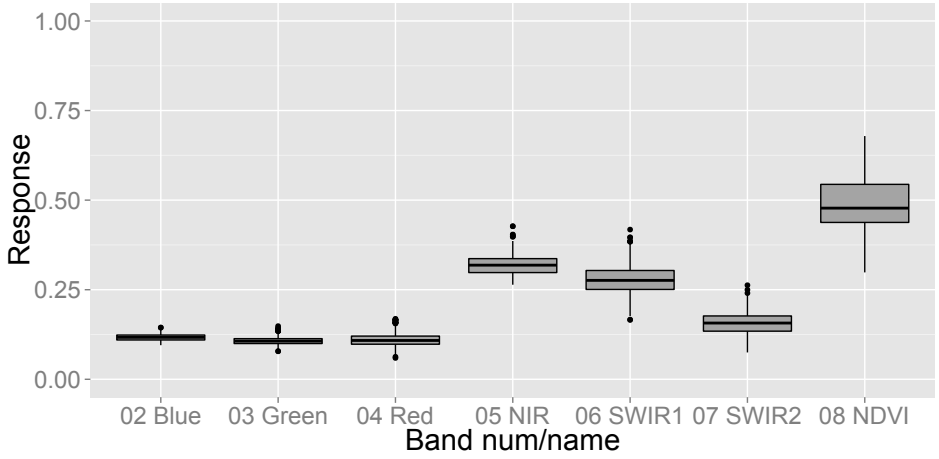


Figure 4.6: Spectral Responses For Vegetation Class

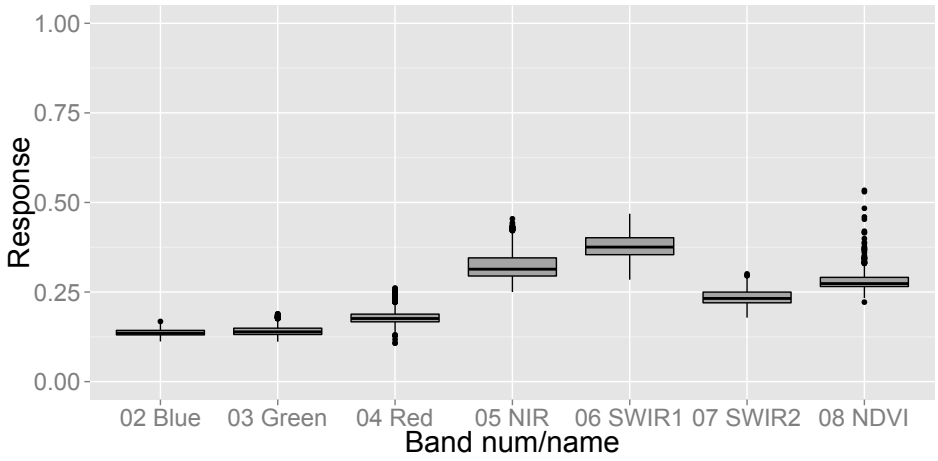


Figure 4.7: Spectral Responses For Stover Class

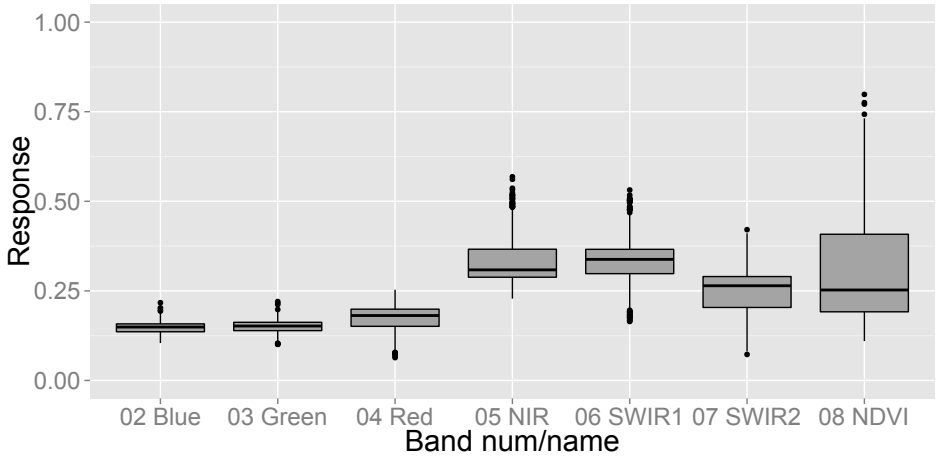


Figure 4.8: Spectral Responses For Other Class

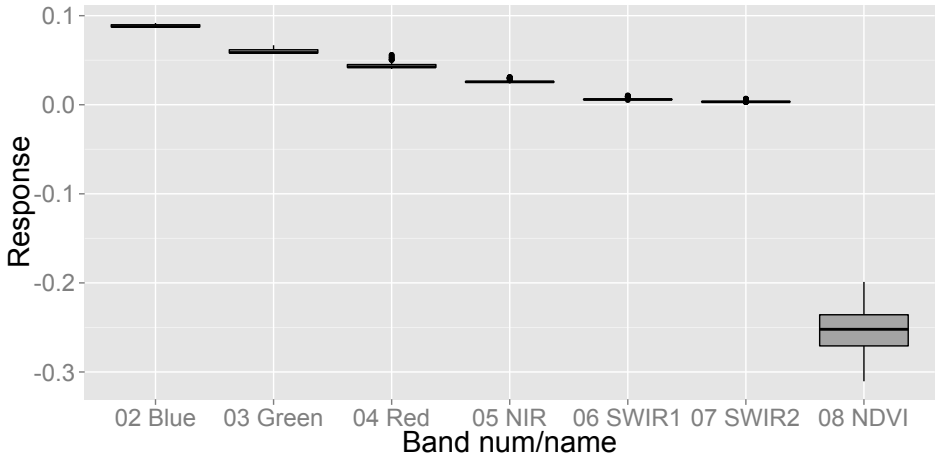


Figure 4.9: Spectral Responses For River Class

4.2.2 Classification

Table 4.1: k-Nearest Neighbor confusion matrix

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	325	0	0	22	0	0	0	0	0	347	0.937
Stover	0	286	0	0	0	0	1	1	4	292	0.979
River	0	0	131	0	0	0	0	0	0	131	1
Maize	12	0	0	861	0	6	2	0	0	881	0.977
Potato	0	0	0	0	200	8	1	1	0	210	0.952
Tomato	0	0	0	6	3	665	0	2	0	676	0.984
Vineyard	0	1	0	1	0	3	172	3	5	185	0.93
Other	0	9	0	0	0	9	12	193	2	225	0.858
Vegetation	1	4	0	0	0	0	7	2	285	299	0.953
Total	338	300	131	890	203	691	195	202	296	3246	
Precision	0.962	0.953	1	0.967	0.985	0.962	0.882	0.955	0.963		

Accuracy: 0.961 **Kappa**: 0.953 **95% Confidence Interval**: (0.953,0.967)

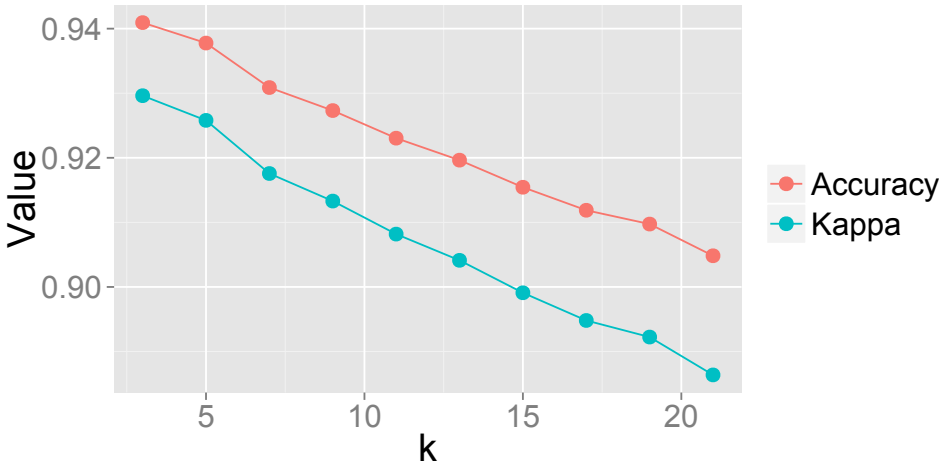


Figure 4.10: kNN accuracy/kappa results during training, $k = 3$ was chosen

Table 4.2: Decision Tree confusion matrix

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	327	0	0	19	0	1	0	0	0	347	0.942
Stover	0	274	0	0	0	0	2	15	1	292	0.938
River	0	0	131	0	0	0	0	0	0	131	1
Maize	22	0	0	847	1	9	2	0	0	881	0.961
Potato	0	0	0	0	202	8	0	0	0	210	0.962
Tomato	0	0	0	0	9	664	1	2	0	676	0.982
Vineyard	0	0	0	2	1	2	150	11	19	185	0.811
Other	0	13	0	0	1	7	3	195	6	225	0.867
Vegetation	0	6	0	0	1	0	10	7	275	299	0.92
Total	349	293	131	868	215	691	168	230	301	3246	
Precision	0.937	0.935	1	0.976	0.94	0.961	0.893	0.848	0.914		

Accuracy: 0.944 **Kappa**: 0.934 **95% Confidence Interval**: (0.936,0.952)

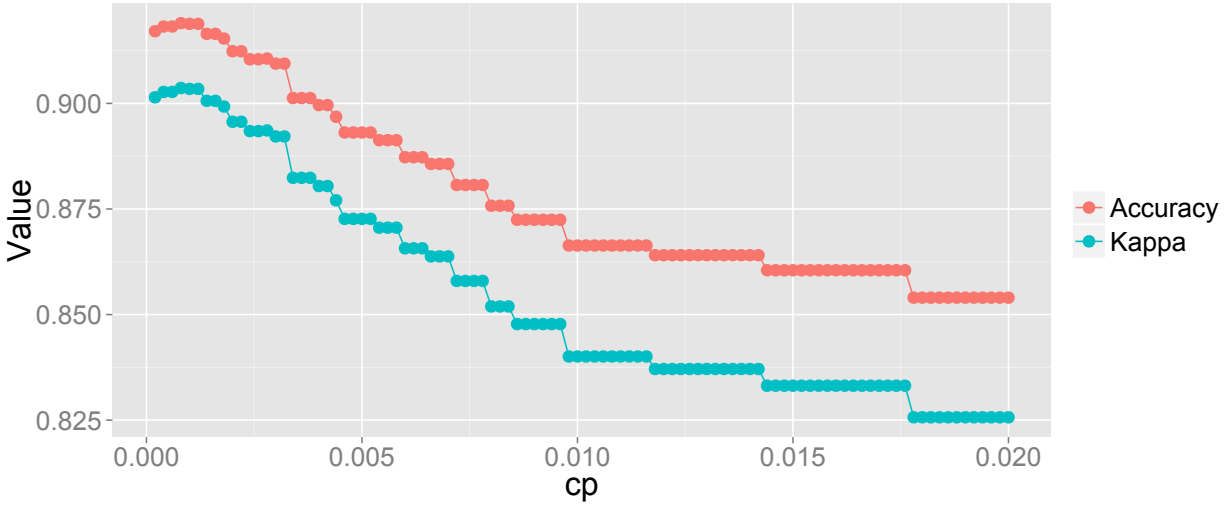


Figure 4.11: Decision Tree accuracy/kappa results during training, $cp = 0.0008$ was chosen

Table 4.3: Random Forest confusion matrix

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	340	0	0	6	0	1	0	0	0	347	0.98
Stover	0	284	0	0	0	0	1	4	3	292	0.973
River	0	0	131	0	0	0	0	0	0	131	1
Maize	3	0	0	872	0	6	0	0	0	881	0.99
Potato	0	0	0	0	203	7	0	0	0	210	0.967
Tomato	0	0	0	1	1	672	0	2	0	676	0.994
Vineyard	0	0	0	1	0	3	174	1	6	185	0.941
Other	0	9	0	0	0	9	2	201	4	225	0.893
Vegetation	1	2	0	0	0	0	6	6	284	299	0.95
Total	344	295	131	880	204	698	183	214	297	3246	
Precision	0.988	0.963	1	0.991	0.995	0.963	0.951	0.939	0.956		

Accuracy: 0.974 **Kappa**: 0.969 **95% Confidence Interval**: (0.968,0.979)

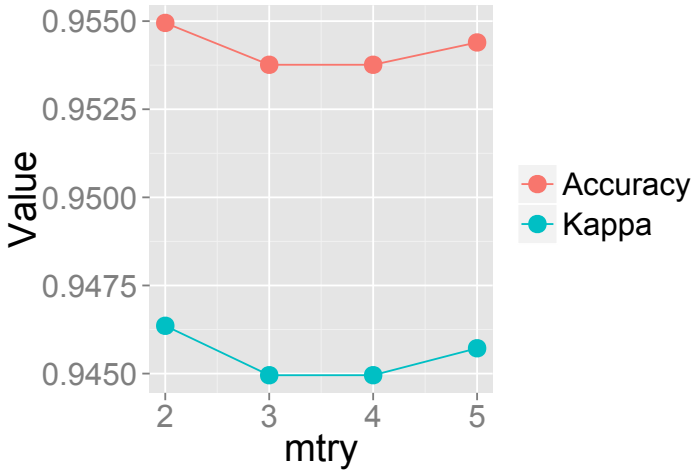


Figure 4.12: Random Forest accuracy/kappa results during training, $mtry = 2$ was chosen

Table 4.4: Support Vector Machines confusion matrix

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	345	0	0	2	0	0	0	0	0	347	0.994
Stover	0	288	0	0	0	0	1	1	2	292	0.986
River	0	0	131	0	0	0	0	0	0	131	1
Maize	2	0	0	869	0	9	1	0	0	881	0.986
Potato	0	0	0	0	205	5	0	0	0	210	0.976
Tomato	0	0	0	0	1	673	0	2	0	676	0.996
Vineyard	0	0	0	2	0	3	180	0	0	185	0.973
Other	0	11	0	0	0	8	2	201	3	225	0.893
Vegetation	0	3	0	0	0	0	0	5	291	299	0.973
Total	347	302	131	873	206	698	184	209	296	3246	
Precision	0.994	0.954	1	0.995	0.995	0.964	0.978	0.962	0.983		

Accuracy: 0.981 **Kappa**: 0.977 **95% Confidence Interval**: (0.975,0.985)

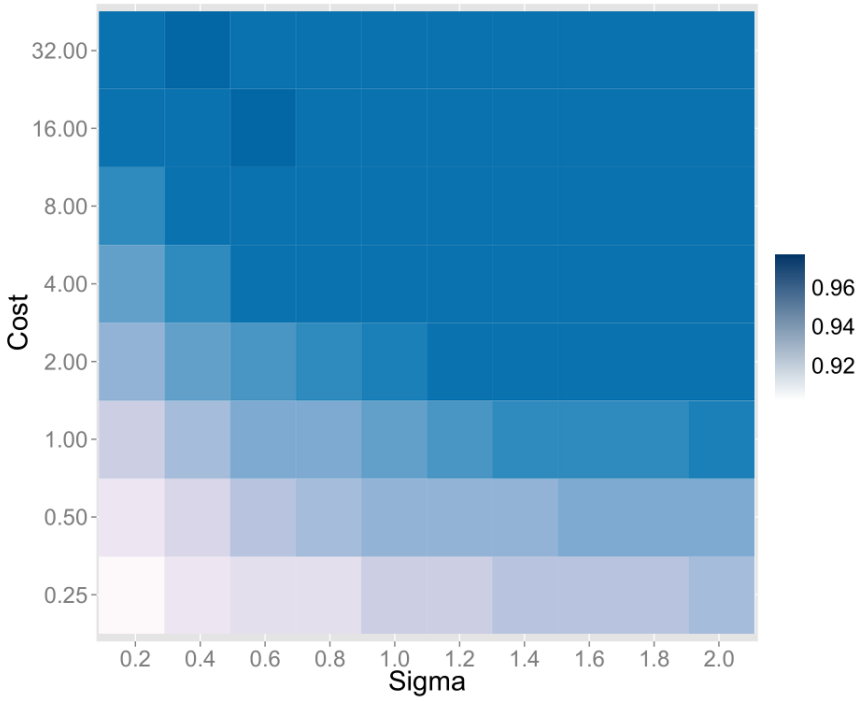


Figure 4.13: SVM accuracy/kappa results during training, $Cost = 32$, $Sigma = 0.4$ were chosen

Table 4.5: Maximum Likelihood Classification confusion matrix

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	345	0	0	0	0	1	1	0	0	347	0.994
Stover	0	278	0	0	0	0	1	13	0	292	0.952
River	0	0	131	0	0	0	0	0	0	131	1
Maize	6	0	0	858	0	9	4	0	4	881	0.974
Potato	0	0	0	0	203	5	0	2	0	210	0.967
Tomato	0	0	0	0	1	662	9	4	0	676	0.979
Vineyard	0	0	0	2	0	4	177	2	0	185	0.957
Other	0	9	0	0	0	9	2	204	1	225	0.907
Vegetation	0	8	0	0	0	0	3	5	283	299	0.946
Total	351	295	131	860	204	690	197	230	288	3246	
Precision	0.983	0.942	1	0.998	0.995	0.959	0.898	0.887	0.983		

Accuracy: 0.968 **Kappa**: 0.962 **95% Confidence Interval**: (0.961,0.973)

Table 4.6: Average execution times for the single image phase algorithms

Algorithm	Loading Data (mins)	Training/Testing Time (secs)	Full Classification time (mins)
KNN		4.104234	1.463784
DT		3.357192	0.325752
RF	1.1597	14.91685	0.9372203
SVM		181.175	1.996981
MLC		0.9300539	5.567268

4.2.3 Complementary studies

4.2.3.1 Reduction of available data

Table 4.7: Results of using only a percentage of the available initial data on the accuracy/kappa values

Algorithm	Percentage	Accuracy	Kappa
KNN	5%	0.835	0.804
	10%	0.9	0.881
	25%	0.942	0.931
	50%	0.957	0.949
	75%	0.954	0.945
	100%	0.96	0.952
DT	5%	0.842	0.811
	10%	0.888	0.866
	25%	0.886	0.864
	50%	0.938	0.926
	75%	0.924	0.909
	100%	0.93	0.917
RF	5%	0.88	0.856
	10%	0.931	0.918
	25%	0.952	0.942
	50%	0.971	0.965
	75%	0.97	0.964
	100%	0.97	0.964
SVM	5%	0.937	0.924
	10%	0.975	0.97
	25%	0.974	0.969
	50%	0.981	0.978
	75%	0.978	0.974
	100%	0.98	0.976
MLC	5%	0.93	0.917
	10%	0.972	0.967
	25%	0.959	0.951
	50%	0.967	0.961
	75%	0.967	0.961
	100%	0.963	0.956

4.2.3.2 No image correction performed

Table 4.8: k-Nearest Neighbor confusion matrix with no image correction

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	323	0	0	23	0	0	1	0	0	347	0.931
Stover	0	279	0	0	0	2	6	0	5	292	0.955
River	0	0	131	0	0	0	0	0	0	131	1
Maize	9	0	0	866	0	14	2	0	0	891	0.972
Potato	0	0	0	1	191	14	0	0	0	206	0.927
Tomato	2	0	0	1	4	667	1	5	0	680	0.981
Vineyard	0	1	0	2	0	4	168	4	5	184	0.913
Other	0	5	0	0	0	12	9	195	3	224	0.871
Vegetation	0	4	0	0	0	0	4	0	289	297	0.973
Total	334	289	131	893	195	713	191	204	302	3252	
Precision	0.967	0.965	1	0.97	0.979	0.935	0.88	0.956	0.957		

Accuracy: 0.956 **Kappa**: 0.948 **95% Confidence Interval**: (0.948,0.963)

Table 4.9: Decision Tree confusion matrix with no image correction

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	331	0	0	14	0	1	0	0	1	347	0.954
Stover	0	264	0	0	1	2	2	8	15	292	0.904
River	0	0	131	0	0	0	0	0	0	131	1
Maize	12	0	0	869	1	8	0	0	1	891	0.975
Potato	1	0	0	0	188	15	2	0	0	206	0.913
Tomato	2	0	0	4	13	653	1	7	0	680	0.96
Vineyard	0	2	0	2	0	6	156	5	13	184	0.848
Other	0	5	0	0	0	13	10	189	7	224	0.844
Vegetation	0	6	0	0	0	0	13	2	276	297	0.929
Total	346	277	131	889	203	698	184	211	313	3252	
Precision	0.957	0.953	1	0.978	0.926	0.936	0.848	0.896	0.882		

Accuracy: 0.940 **Kappa**: 0.929 **95% Confidence Interval**: (0.931,0.948)

Table 4.10: Random Forest confusion matrix with no image correction

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	335	0	0	12	0	0	0	0	0	347	0.965
Stover	0	279	0	0	0	2	3	2	6	292	0.955
River	0	0	131	0	0	0	0	0	0	131	1
Maize	1	0	0	879	0	9	2	0	0	891	0.987
Potato	0	0	0	1	193	9	1	2	0	206	0.937
Tomato	1	0	0	0	2	670	1	6	0	680	0.985
Vineyard	0	2	0	2	0	5	169	2	4	184	0.918
Other	0	4	0	0	0	11	8	198	3	224	0.884
Vegetation	0	5	0	0	0	0	4	0	288	297	0.97
Total	337	290	131	894	195	706	188	210	301	3252	
Precision	0.994	0.962	1	0.983	0.99	0.949	0.899	0.943	0.957		

Accuracy: 0.966 **Kappa**: 0.960 **95% Confidence Interval**: (0.959,0.972)

Table 4.11: Support Vector Machines confusion matrix with no image correction

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	345	0	0	1	0	0	1	0	0	347	0.994
Stover	0	279	0	0	0	2	7	1	3	292	0.955
River	0	0	131	0	0	0	0	0	0	131	1
Maize	3	0	0	874	0	13	1	0	0	891	0.981
Potato	1	0	0	0	198	7	0	0	0	206	0.961
Tomato	1	0	0	1	3	671	0	4	0	680	0.987
Vineyard	0	1	0	2	0	5	174	2	0	184	0.946
Other	0	5	0	0	0	13	9	194	3	224	0.866
Vegetation	0	3	0	0	0	0	0	0	294	297	0.99
Total	350	288	131	878	201	711	192	201	300	3252	
Precision	0.986	0.969	1	0.995	0.985	0.944	0.906	0.965	0.98		

Accuracy: 0.972 **Kappa**: 0.966 **95% Confidence Interval**: (0.965,0.977)

Table 4.12: Maximum Likelihood Classification confusion matrix with no image correction

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	343	0	0	1	0	2	1	0	0	347	0.988
Stover	0	277	0	0	0	2	4	8	1	292	0.949
River	0	0	131	0	0	0	0	0	0	131	1
Maize	6	0	0	864	0	12	6	1	2	891	0.97
Potato	1	0	0	0	196	7	0	2	0	206	0.951
Tomato	0	0	0	1	2	646	10	21	0	680	0.95
Vineyard	0	3	0	2	0	5	168	5	1	184	0.913
Other	0	6	0	0	0	11	4	202	1	224	0.902
Vegetation	0	13	0	0	0	0	2	4	278	297	0.936
Total	350	299	131	868	198	685	195	243	283	3252	
Precision	0.98	0.926	1	0.995	0.99	0.943	0.862	0.831	0.982		

Accuracy: 0.955 **Kappa**: 0.946 **95% Confidence Interval**: (0.947,0.962)

4.2.3.3 Without NDVI band

Table 4.13: k-Nearest Neighbor confusion matrix without NDVI band

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	318	0	0	29	0	0	0	0	0	347	0.916
Stover	0	288	0	0	0	0	1	1	2	292	0.986
River	0	0	131	0	0	0	0	0	0	131	1
Maize	14	0	0	855	0	7	5	0	0	881	0.97
Potato	1	0	0	0	202	6	0	1	0	210	0.962
Tomato	0	0	0	2	3	669	1	1	0	676	0.99
Vineyard	0	0	0	1	0	1	177	3	3	185	0.957
Other	0	6	0	0	0	9	11	196	3	225	0.871
Vegetation	0	1	0	0	0	0	5	4	289	299	0.967
Total	333	295	131	887	205	692	200	206	297	3246	
Precision	0.955	0.976	1	0.964	0.985	0.967	0.885	0.951	0.973		

Accuracy: 0.963 **Kappa**: 0.956 **95% Confidence Interval**: (0.956,0.969)

Table 4.14: Decision Tree confusion matrix without NDVI band

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	327	0	0	19	0	1	0	0	0	347	0.942
Stover	0	273	0	0	0	0	0	11	8	292	0.935
River	0	0	131	0	0	0	0	0	0	131	1
Maize	22	0	0	847	1	9	2	0	0	881	0.961
Potato	0	0	0	0	202	8	0	0	0	210	0.962
Tomato	0	0	0	0	6	667	1	2	0	676	0.987
Vineyard	0	0	0	2	0	6	155	3	19	185	0.838
Other	0	16	0	0	1	8	1	192	7	225	0.853
Vegetation	0	10	0	0	0	1	12	2	274	299	0.916
Total	349	299	131	868	210	700	171	210	308	3246	
Precision	0.937	0.913	1	0.976	0.962	0.953	0.906	0.914	0.89		

Accuracy: 0.945 **Kappa**: 0.935 **95% Confidence Interval**: (0.937,0.953)

Table 4.15: Random Forest confusion matrix without NDVI band

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	342	0	0	4	0	1	0	0	0	347	0.986
Stover	0	283	0	0	0	0	1	6	2	292	0.969
River	0	0	131	0	0	0	0	0	0	131	1
Maize	5	0	0	868	0	7	1	0	0	881	0.985
Potato	0	0	0	0	203	7	0	0	0	210	0.967
Tomato	0	0	0	1	0	673	0	2	0	676	0.996
Vineyard	0	0	0	2	0	3	175	1	4	185	0.946
Other	0	8	0	0	0	9	0	204	4	225	0.907
Vegetation	0	3	0	0	0	0	5	6	285	299	0.953
Total	347	294	131	875	203	700	182	219	295	3246	
Precision	0.986	0.963	1	0.992	1	0.961	0.962	0.932	0.966		

Accuracy: 0.975 **Kappa**: 0.970 **95% Confidence Interval**: (0.969,0.980)

Table 4.16: Support Vector Machines confusion matrix without NDVI band

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	347	0	0	0	0	0	0	0	0	347	1
Stover	0	287	0	0	0	0	1	2	2	292	0.983
River	0	0	131	0	0	0	0	0	0	131	1
Maize	2	0	0	869	0	9	1	0	0	881	0.986
Potato	0	0	0	0	205	5	0	0	0	210	0.976
Tomato	0	0	0	0	1	673	0	2	0	676	0.996
Vineyard	0	0	0	2	0	4	178	1	0	185	0.962
Other	0	10	0	0	0	8	1	203	3	225	0.902
Vegetation	0	3	0	0	0	0	0	4	292	299	0.977
Total	349	300	131	871	206	699	181	212	297	3246	
Precision	0.994	0.957	1	0.998	0.995	0.963	0.983	0.958	0.983		

Accuracy: 0.981 **Kappa**: 0.978 **95% Confidence Interval**: (0.976,0.986)

Table 4.17: Maximum Likelihood Classification confusion matrix without NDVI band

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	342	0	0	0	0	1	4	0	0	347	0.986
Stover	0	278	0	0	0	0	1	13	0	292	0.952
River	0	0	131	0	0	0	0	0	0	131	1
Maize	6	0	0	855	0	9	6	2	3	881	0.97
Potato	0	0	0	0	204	5	0	1	0	210	0.971
Tomato	0	0	0	0	1	654	10	11	0	676	0.967
Vineyard	0	0	0	2	0	3	175	3	2	185	0.946
Other	0	10	0	0	0	7	1	206	1	225	0.916
Vegetation	0	7	0	0	0	0	2	6	284	299	0.95
Total	348	295	131	857	205	679	199	242	290	3246	
Precision	0.983	0.942	1	0.998	0.995	0.963	0.879	0.851	0.979		

Accuracy: 0.964 **Kappa**: 0.957 **95% Confidence Interval**: (0.957,0.970)

4.3 Multi-image phase

In this section the results for the multi-image phase study will be presented. Since two algorithms were used, learning from five different images and each trying to classify four others, this yields forty different results, for the sake of simplicity, only eight confusion matrices (the ones where the image from July 22, 2013 is used as training data will be presented. Although the other matrices will not be shown, their results will be presented in the form of a table.

4.3.1 Random Forest Classification

Table 4.18: Random Forest confusion matrix - 22 Jul 13 as train, 20 Jun 13 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	968	0	14	34	0	0	4	0	0	1020	0.949
Stover	0	281	0	0	0	0	403	76	99	859	0.327
River	379	0	0	0	0	0	0	0	0	379	0
Maize	3	274	0	512	9	0	1090	483	246	2617	0.196
Potato	0	0	0	4	547	2	51	0	6	610	0.897
Tomato	0	106	0	0	186	356	1258	81	1	1988	0.179
Vineyard	0	4	0	0	6	3	497	11	30	551	0.902
Other	0	70	0	0	1	0	127	421	28	647	0.651
Vegetation	0	26	0	14	0	0	21	9	813	883	0.921
Total	1350	761	14	564	749	361	3451	1081	1223	9554	
Precision	0.717	0.369	0	0.908	0.73	0.986	0.144	0.389	0.665		

Accuracy: 0.460 **Kappa**: 0.406 **95% Confidence Interval**: (0.450,0.470)

Table 4.19: Random Forest confusion matrix - July 22, 2013 as train, July 06, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	1011	0	0	0	0	5	4	0	0	1020	0.991
Stover	0	747	0	0	2	14	17	37	42	859	0.87
River	1	0	378	0	0	0	0	0	0	379	0.997
Maize	605	4	0	1483	21	54	125	24	301	2617	0.567
Potato	3	0	0	0	545	44	15	3	0	610	0.893
Tomato	1	0	0	3	82	1506	266	127	3	1988	0.758
Vineyard	8	6	0	4	0	7	482	37	7	551	0.875
Other	0	64	0	0	0	8	13	548	14	647	0.847
Vegetation	0	23	0	0	0	0	14	9	837	883	0.948
Total	1629	844	378	1490	650	1638	936	785	1204	9554	
Precision	0.621	0.885	1	0.995	0.838	0.919	0.515	0.698	0.695		

Accuracy: 0.789 **Kappa**: 0.756 **95% Confidence Interval**: (0.781,0.797)

Table 4.20: Random Forest confusion matrix - July 22, 2013 as train, August 07, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	627	0	0	392	0	0	1	0	0	1020	0.615
Stover	0	796	0	7	2	5	8	18	23	859	0.927
River	237	0	142	0	0	0	0	0	0	379	0.375
Maize	19	0	0	2551	2	35	4	0	6	2617	0.975
Potato	1	0	0	52	154	359	31	4	9	610	0.252
Tomato	42	0	0	247	0	1689	10	0	0	1988	0.85
Vineyard	0	6	0	19	0	10	342	11	163	551	0.621
Other	0	54	0	0	0	32	17	524	20	647	0.81
Vegetation	0	15	0	6	0	0	15	3	844	883	0.956
Total	926	871	142	3274	158	2130	428	560	1065	9554	
Precision	0.677	0.914	1	0.779	0.975	0.793	0.799	0.936	0.792		

Accuracy: 0.803 **Kappa**: 0.760 **95% Confidence Interval**: (0.795,0.811)

Table 4.21: Random Forest confusion matrix - July 22, 2013 as train, August 21, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	385	0	0	635	0	0	0	0	0	1020	0.377
Stover	0	774	0	0	1	3	9	40	32	859	0.901
River	241	0	138	0	0	0	0	0	0	379	0.364
Maize	42	1	0	2480	6	33	19	0	36	2617	0.948
Potato	143	185	0	11	14	83	104	24	46	610	0.023
Tomato	71	292	0	18	6	1422	53	3	123	1988	0.715
Vineyard	6	17	0	9	0	4	272	4	239	551	0.494
Other	0	99	0	0	0	23	14	486	25	647	0.751
Vegetation	0	18	0	2	0	0	56	6	801	883	0.907
Total	888	1386	138	3155	27	1568	527	563	1302	9554	
Precision	0.434	0.558	1	0.786	0.519	0.907	0.516	0.863	0.615		

Accuracy: 0.709 **Kappa**: 0.650 **95% Confidence Interval**: (0.700,0.718)

In the following tables, training is defined by the row, and test by the column, for example, the middle row of table 4.22 shows the results when training set was the July 22 data, test data being each of the cells' column title. The diagonal (grey) cells are the single-image phase scripts applied to different dates (so the center cell is the same result as the one from Table 4.3).

Table 4.22: Accuracies for the RF algorithm applied to different dates in training and testing

	20 Jun	06 Jul	22 Jul	07 Aug	21 Aug
20 Jun	0.9121	0.6921	0.5806	0.5597	0.4973
06 Jul	0.5918	0.9488	0.8611	0.7045	0.6538
22 Jul	0.46	0.789	0.974	0.803	0.709
07 Aug	0.3586	0.5895	0.8969	0.9568	0.807
21 Aug	0.2833	0.6212	0.793	0.8138	0.9608

Table 4.23: Kappa values for the RF algorithm applied to different dates in training and testing

	20 Jun	06 Jul	22 Jul	07 Aug	21 Aug
20 Jun	0.8947	0.6428	0.5124	0.486	0.4098
06 Jul	0.5312	0.939	0.8319	0.6332	0.5756
22 Jul	0.406	0.756	0.969	0.760	0.650
07 Aug	0.3074	0.5357	0.8775	0.9485	0.7716
21 Aug	0.2027	0.5533	0.7502	0.7805	0.9533

4.3.2 Support Vector Machine Classification

Table 4.24: SVM confusion matrix - July 22, 2013 as train, June 20, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	817	0	0	43	0	0	0	160	0	1020	0.801
Stover	0	222	0	73	0	0	47	517	0	859	0.258
River	0	0	291	0	0	0	0	88	0	379	0.768
Maize	0	485	0	876	1	28	1013	214	0	2617	0.335
Potato	0	34	0	27	197	351	1	0	0	610	0.323
Tomato	0	213	0	53	0	712	1009	1	0	1988	0.358
Vineyard	0	351	0	63	0	11	122	4	0	551	0.221
Other	0	93	0	2	2	0	356	194	0	647	0.3
Vegetation	0	456	0	322	0	1	25	77	2	883	0.002
Total	817	1854	291	1459	200	1103	2573	1255	2	9554	
Precision	1	0.12	1	0.6	0.985	0.646	0.047	0.155	1		

Accuracy: 0.359 **Kappa**: 0.272 **95% Confidence Interval**: (0.350,0.369)

Table 4.25: SVM confusion matrix - July 22, 2013 as train, July 06, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	967	0	0	19	2	1	4	27	0	1020	0.948
Stover	0	488	0	0	3	3	29	254	82	859	0.568
River	0	0	379	0	0	0	0	0	0	379	1
Maize	250	4	0	2029	1	31	158	12	132	2617	0.775
Potato	9	0	0	0	562	16	16	7	0	610	0.921
Tomato	10	0	0	1	40	1500	228	209	0	1988	0.755
Vineyard	4	0	0	12	0	7	480	42	6	551	0.871
Other	0	56	0	0	0	10	8	546	27	647	0.844
Vegetation	0	12	0	0	0	0	6	4	861	883	0.975
Total	1240	560	379	2061	608	1568	929	1101	1108	9554	
Precision	0.78	0.871	1	0.984	0.924	0.957	0.517	0.496	0.777		

Accuracy: 0.818 **Kappa**: 0.787 **95% Confidence Interval**: (0.810,0.825)

Table 4.26: SVM confusion matrix - July 22, 2013 as train, August 07, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	879	0	0	141	0	0	0	0	0	1020	0.862
Stover	0	815	0	9	1	4	13	6	11	859	0.949
River	0	0	379	0	0	0	0	0	0	379	1
Maize	2	0	0	2570	4	33	2	0	6	2617	0.982
Potato	0	0	0	40	243	252	74	1	0	610	0.398
Tomato	4	0	0	52	0	1921	11	0	0	1988	0.966
Vineyard	0	6	0	51	0	7	468	2	17	551	0.849
Other	0	60	0	0	0	40	12	528	7	647	0.816
Vegetation	0	11	0	4	0	0	40	8	820	883	0.929
Total	885	892	379	2867	248	2257	620	545	861	9554	
Precision	0.993	0.914	1	0.896	0.98	0.851	0.755	0.969	0.952		

Accuracy: 0.903 **Kappa**: 0.883 **95% Confidence Interval**: (0.896,0.908)

Table 4.27: SVM confusion matrix - July 22, 2013 as train, August 21, 2013 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total	Recall
Rice	763	0	0	257	0	0	0	0	0	1020	0.748
Stover	0	808	0	1	0	4	19	15	12	859	0.941
River	0	0	378	0	0	0	0	1	0	379	0.997
Maize	6	0	0	2527	5	31	19	1	28	2617	0.966
Potato	23	7	0	92	146	14	120	173	35	610	0.239
Tomato	13	0	0	63	10	1405	56	275	166	1988	0.707
Vineyard	0	7	0	22	0	2	414	6	100	551	0.751
Other	0	66	0	0	0	26	15	515	25	647	0.796
Vegetation	1	30	0	0	0	0	140	44	668	883	0.757
Total	806	918	378	2962	161	1482	783	1030	1034	9554	
Precision	0.947	0.88	1	0.853	0.907	0.948	0.529	0.5	0.646		

Accuracy: 0.798 **Kappa**: 0.760 **95% Confidence Interval**: (0.790,0.806)

In the following tables, training is defined by the row, and test by the column, for example, the middle row of table 4.28 shows the results when training set was the July 22 data, test data being each of the cells' column title. The diagonal (grey) cells are the single-image phase scripts applied to different dates (so the center cell is the same result as the one from Table 4.4).

Table 4.28: Accuracies for the SVM algorithm applied to different dates in training and testing

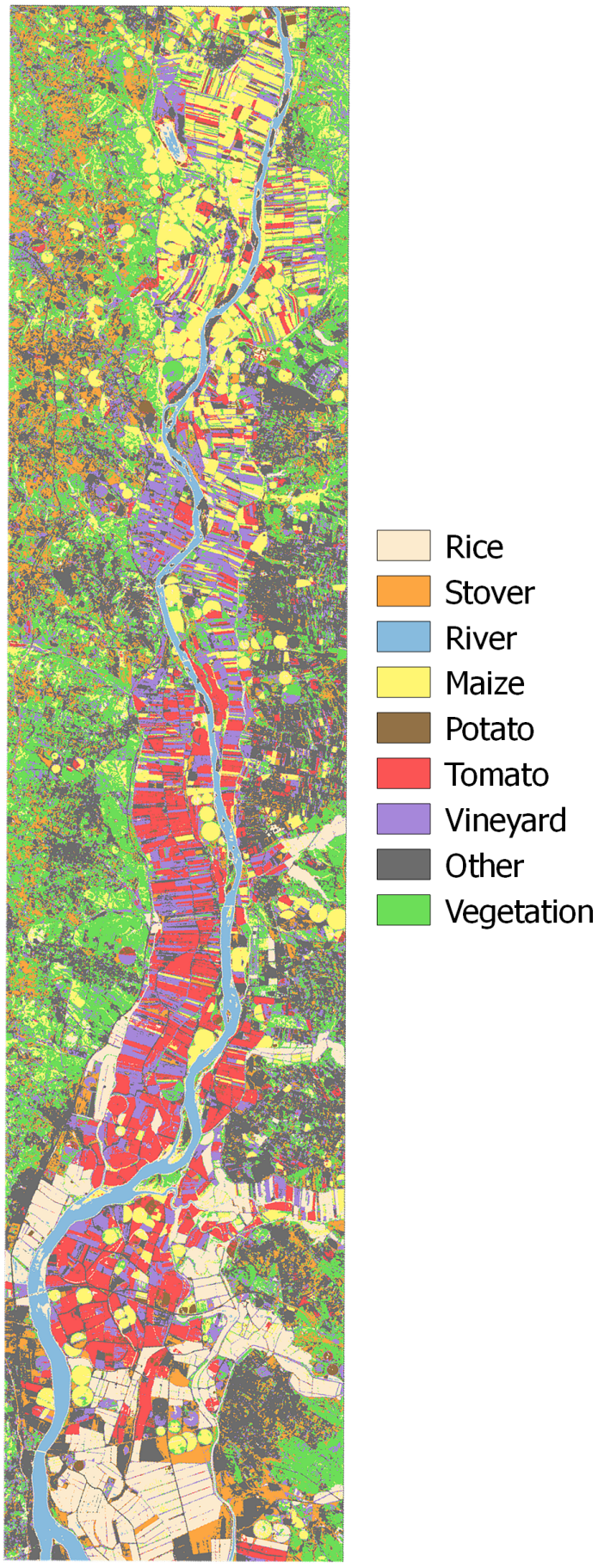
	20 Jun	06 Jul	22 Jul	07 Aug	21 Aug
20 Jun	0.9497	0.3023	0.2309	0.2576	0.2151
06 Jul	0.2839	0.9633	0.861	0.753	0.6439
22 Jul	0.3593	0.8177	0.982	0.9026	0.798
07 Aug	0.1992	0.4677	0.8957	0.9747	0.7849
21 Aug	0.1863	0.588	0.836	0.8922	0.9707

Table 4.29: Kappa values for the SVM algorithm applied to different dates in training and testing

	20 Jun	06 Jul	22 Jul	07 Aug	21 Aug
20 Jun	0.94	0.237	0.1675	0.1978	0.1502
06 Jul	0.1565	0.9562	0.8315	0.6969	0.567
22 Jul	0.2724	0.7875	0.978	0.8828	0.7596
07 Aug	0.1289	0.3873	0.8757	0.9699	0.7475
21 Aug	0.0586	0.5233	0.8045	0.8717	0.9651

4.3.3 Images

Images for all the results of this chapter can also be generated, an example of such an image is provided (in custom colors) in the next page, these images not only provided meaningful analysis during the initial phase of this work, but are the intended output of a real-world application to this work.


Rice
Stover
River
Maize
Potato
Tomato
Vineyard
Other
Vegetation

4.4 Single-image phase discussion

Regarding visual interpretation of the spectral signatures for all the classes (Section 4.2.2), slightly distinct patterns can be found in them, although there obviously some outliers are present in all classes. The most similar classes are arguably Maize/Rice and Vegetation/Vineyard, and some misclassification is expected between those groups.

Table 4.30: Summary of the accuracy results for the single-image phase.

Algorithm	Initial (4.2.2)	No Correction (4.2.3.2)	No NDVI (4.2.3.3)	Percentage (4.2.3.1)	Accuracy
KNN	0.961	0.956	0.963	5%	0.835
				10%	0.9
				25%	0.942
				50%	0.957
				75%	0.954
				100%	0.96
DT	0.944	0.940	0.945	5%	0.842
				10%	0.888
				25%	0.886
				50%	0.938
				75%	0.924
				100%	0.93
RF	0.974	0.966	0.975	5%	0.88
				10%	0.931
				25%	0.952
				50%	0.971
				75%	0.97
				100%	0.97
SVM	0.981	0.972	0.981	5%	0.937
				10%	0.975
				25%	0.974
				50%	0.981
				75%	0.978
				100%	0.98
MLC	0.968	0.955	0.964	5%	0.93
				10%	0.972
				25%	0.959
				50%	0.967
				75%	0.967
				100%	0.963

Regarding the classification itself (Section 4.2.2), very good results come from all the algorithms, regarding both accuracy and Kappa values. There seems to be a slight edge in Support Vector Machines and Random Forest algorithms, followed by both k-Nearest Neighbor and the Maximum Likelihood Classifier, followed by the Decision Tree algorithm.

Regarding the tuning of the classifiers' parameters the following can be stated: for k-NN, smaller k values seem to provide the best results, with $k = 3$ being the best. For Decision Tree, again, smaller values of cp provided the best results, although not the smallest value used of 0.0002, but 0.0008. In the Random Forest algorithm, $mtry$ value of 2 was chosen, however, there is very little difference between any of the $mtry$ values used in training, as seen in figure 4.12. For Support Vector Machines, figure 4.13 shows the grid results, where the best were $Cost = 32$, $Sigma = 0.4$, followed by $Cost = 16$, $Sigma = 0.6$.

For the complementary studies performed, regarding the study about influence of a percentage of the training data, it seems that generally most of the algorithms benefit from having more data, but the accuracy loss is more noticeable only when using 5/10% of the original data, results can worsen by slightly over 10%. SVM and MLC are the only ones to maintain an accuracy rate of over 90% when using only 5% of data. Regarding the images with no image correction, it can be seen that results worsen when using these images, but this worsening is very slight, from around 0.4% in the least worst case (Decision Tree) to 1.3% in the worst case (Maximum Likelihood Classifier). Regarding the lack of usage of the NDVI band, there is rarely any difference from the original results, and so it seems this band is quite irrelevant.

Regarding the main classification itself, as mentioned previously, very good results were achieved, following are a number of reasons for this:

The new satellite

None of the previous studies used the Landsat 8 satellite, since this is a very recent operating satellite. The fact 16 bit images are produced instead of the usual 8 bit ones may have some influence on the results, by providing more accurate measurements. Also, the electromagnetic spectrum zones captured by the satellite may be new or slightly different than the ones captured by previous satellites, resulting in slightly different measures, which may also be a factor.

Easier image

The image in itself can also be slightly easier to classify than others, although this seems doubtful, since most of the mentioned studies try to classify classes that are presumably more distinct than the ones in this study.

Class-correct ground truth data

Some studies mention that the ground truth data was defined by visual analysis of the images, instead of actual field visits, like in this work. This can lead to worse results, since learning from wrong classes is not, naturally, adequate.

Time-correct ground truth data

By far, the main reason for these results seems to be the fact the ground visits (mentioned on Section 3.1) were performed only a few days after the satellite image was taken. This lack of temporal difference, along with the actual visits as mentioned above, allowed this study to have ground truth data as correct as possible. As the multi-image phase has shown, simply using ground truth data two weeks apart from the date the image was taken, already worsens the accuracies by at least 8/9%.

Regarding execution times for the single image phase, table 4.6 presents the average running time for the various algorithms. The times can be split into three phases: the loading data phase, which is equal for every algorithm, the training/testing phase, where the algorithms are trained and tested, and the image classification phase, where the full image is classified. In all of the algorithms except for the SVM, the training/testing phase is the shortest phase of the process. It is, however, relevant to note the SVM classifier is one of the two algorithms (the other being DT) that trains with a large number of parameters (80 in total, 8 C values and 10 sigma values), which might justify its running time. It might also be relevant to note that MLC takes an unusually long time to classify the full image compared to the other algorithms.

4.5 Multi-image phase discussion

Regarding the multi-image phase, for the 2013 images, as mentioned in Section 4.1.2, the results are expected to be worse and can't really be analyzed directly, these can be divided into two groups:

Images where training data comes from the July 22 image
: Regarding these images, although the results also worsen over time, the algorithms have learned with the "right" data, it is expected that these worsened results are mostly a product of the actual land cover changing while ground truth stays exactly the same (such as, a crop changing to another development stage from which there were no good training samples, or the crop itself actually being harvested).

Other images where training data comes from different images
: As seen in Tables 4.22, 4.23, 4.28 and 4.29, again, the furthest the dates from training and test data are, the worse the results are, leading to very bad results in some cases, not much information can be extracted from these results, learning from the "wrong" date is already detrimental, classifying a distinct date from this one simply worsens the results.

Regarding the two algorithms used in this phase, SVM seems to have an advantage when the training and test data have closer dates, but also gets very worse for dates highly apart, while Random Forest has slightly worse results for the former, but doesn't fall off so hard on very distinct dates.

For the 2015 imagery section (appendix B), all the results are very similar for both images, for both algorithms, all hovering around 59 to 64% accuracy. Again, this is still a very good result considering some points are being misclassified wrongly, for example, a newly created crop area for Maize that was Vegetation in the 2013 images, can possibly be classified as Maize but in the end be marked as wrong, since the actual class (from the original 2013 data) in that point was Vegetation.

Ground truth visits were performed in August 2015, by Prof. Carlos Damásio on a portion of the fields, where the actual crop types were compared to the images' classes. The feedback from this seems to be very positive, with the great majority of the points being actually well classified - around 107 out of 116, yielding an accuracy of about 92%.

CHAPTER 5

CONCLUSION AND FURTHER WORK

Five different classification algorithms (k-NN, DT, RF, SVM, MLC) were applied to our data - a number of points each containing information regarding its spectral measurement and its class. Initially, this was done by training and testing the algorithms with the satellite image with closer date to when the ground truth was defined. This yielded very good results - both accuracy and kappa values above 0.9, for all of the algorithms, especially to the RF (0.974 accuracy, 0.969 Kappa) and SVM (0.981 accuracy, 0.977 Kappa) ones.

These results appear to be better than the typical results from the works in this field. Regarding the two works more focused on crop detection, Castillejo-González et al. (2009) achieved accuracies of over 80% with the MLC classifier and pixel-based classification, however, while no other algorithm used in this work was used. Dingle Robertson and King (2011) compared pixel-based and object-based image analysis for classifying agricultural land cover, achieving accuracies of around 75%, however, only around 200 test samples were used, meaning that this work used over ten times more samples for testing (and training), which was also a factor.

In summary, these results seem to be explained by the key factors mentioned previously, especially by the correctness of our data, which was collected mere days after the satellite image was taken, this, along with the high number of samples available, seem to be the two main factors for such good results.

A number of complementary studies was also done to analyze the influence of certain variables - having only a percentage of the initial data available, classify the images without performing image correction, and not using the artificially obtained NDVI band. Of these, the most relevant results came from using a small percentage of the original data (5/10%), where results decreased by over 10% in some cases, showing that having a good amount of data also contributes to the results.

After that, the work was extended to include data from other dates, in this section the algorithms were set to use the best parameters collected in the previous part, used data from one date as the training set, and from other as test set. These results, as expected, are worse than the initial ones and worsen the further the different dates are from one another, since the algorithm is trying to classify data based on a different date's ground truth, and as such the actual accuracy results are secondary. This was done more to analyze the generated images visually than its numeric results.

Finally, images from 2015 were also classified and ground truth visits were performed to assess their accuracies, on which very positive results were also achieved.

From this study, there are a number of improvements/additions that could possibly be made, although not pursued in this work, the main ones found were:

More classes

Simply having five crop classes and four extra ones limits this work, crop types from which there was no class assigned will obviously be classified as one of the existing classes, more classes, both crop ones and additional ones could be added to the shapefile and possibly create a more realistic output. Even the river class is not enough to classify bodies of water, the Tagus river has some sandbanks in it which are mostly being classified as rice - visible in the images generated, a sandbanks class could be added to the shapefile with a few samples to solve this problem.

More classes from same crop types in different development stages

There could also be added classes regarding different evolution states of each crop, for example, having classes for both developing tomato and fully-developed tomato, and do this for multiple or all classes. This could help create a more real-world scenario where classes would not be misclassified simply by being in a different development stage.

Use information from multiple dates to train/test classes

It could also be possible to have the algorithm learn from multiple dates, and use the information of same bands of different dates to classify the classes better, instead of having a set of values (blue, green, red, near infrared, short-wave infrared 1 and 2 and NDVI) for a date, we would have multiple sets providing information. Both bands could be used (e.g. blue from date 1 and blue from date 2), or some kind of mathematical formulas could be applied to generate information similar to how NDVI was created.

Regardless of these possible improvements, the objective of this work - make a comparative study of various classification algorithms and their parameters, specifically regarding crop classification - was successfully achieved.

BIBLIOGRAPHY

Atkinson, E. J. and T. M. Therneau (2000). "An introduction to recursive partitioning using the RPART routines". In: *Rochester: Mayo Foundation*.

Breiman, L. (2001). "Random forests". In: *Machine learning* 45(1), pp. 5–32.

Brenning, A. (2009). "Benchmarking classifiers to optimally integrate terrain analysis and multispectral remote sensing in automatic rock glacier detection". In: *Remote Sensing of Environment* 113(1), pp. 239–247.

Campbell, J. B. (2002). *Introduction to remote sensing*. CRC Press.

Carreiras, J., J. Pereira, M. L. Campagnolo, and Y. E. Shimabukuro (2006). "Assessing the extent of agriculture/pasture and secondary succession forest in the Brazilian Legal Amazon using SPOT VEGETATION data". In: *Remote Sensing of Environment* 101(3), pp. 283–298.

Castillejo-González, I. L., F. López-Granados, A. García-Ferrer, J. M. Peña-Barragán, M. Jurado-Expósito, M. S. de la Orden, and M. González-Audicana (2009). "Object-and pixel-based analysis for mapping crops and their agro environmental associated measures using QuickBird imagery". In: *Computers and Electronics in Agriculture* 68(2), pp. 207–215.

Cleve, C., M. Kelly, F. R. Kearns, and M. Moritz (2008). "Classification of the wildland–urban interface: A comparison of pixel-and object-based classifications using high-resolution aerial photography". In: *Computers, Environment and Urban Systems* 32(4), pp. 317–326.

Cohen, J. (1960). "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20(1), pp. 37–46. ISSN: 0013-1644. DOI: 10.1177/001316446002000104. URL: http://epm.sagepub.com/cgi/content/refs/20/1/37.

Comber, A., P. Fisher, and R. Wadsworth (2005). "What is land cover". In: *Environment and Planning B: Planning and Design* 32(1), pp. 199–209.

Cortes, C. and V. Vapnik (1995). "Support-vector networks". In: *Machine learning* 20(3), pp. 273–297.

Devadas, R, R. Denham, and M Pringle (2012). "Support Vector Machine Classification of Object-based Data for Crop Mapping, Using Multi-temporal Landsat Imagery". In: *ISPRS Archives* 39, B7.

Di Gregorio, A. (2005). *Land cover classification system: classification concepts and user manual: LCCS*. 8. Food & Agriculture Org.

Dingle Robertson, L. and D. J. King (2011). "Comparison of pixel-and object-based classification in land cover change mapping". In: *International Journal of Remote Sensing* 32(6), pp. 1505–1529.

Duro, D. C., S. E. Franklin, and M. G. Dubé (2012). "A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using SPOT-5 HRG imagery". In: *Remote Sensing of Environment* 118, pp. 259–272.

Eastman, J. R. (2003). *IDRISI Kilimanjaro: guide to GIS and image processing*. Clark Labs, Clark University Worcester, MA.

Esri. *Esri*. URL: `http://www.esri.com/software/arcgis/arcgis-for-desktop`.

Esri. *Esri*. URL: `http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`.

Fischer, W. A., W. Hemphill, and A. Kover (1976). "Progress in remote sensing (1972–1976)". In: *Photogrammetria* 32(2), pp. 33–72.

Fix, E. and J. L. Hodges Jr (1951). *Discriminatory analysis-nonparametric discrimination: consistency properties*. Tech. rep. DTIC Document.

Foody, G. M. (2002). "Status of land cover classification accuracy assessment". In: *Remote sensing of environment* 80(1), pp. 185–201.

Gislason, P. O., J. A. Benediktsson, and J. R. Sveinsson (2006). "Random forests for land cover classification". In: *Pattern Recognition Letters* 27(4), pp. 294–300.

Han, J., M. Kamber, and J. Pei (2006). *Data mining, southeast asia edition: Concepts and techniques*. Morgan kaufmann.

how.com, what-when. *what-when-how.com - Imaging System Types (Visible Imagery) (Remote Sensing)*. URL: `http://what-when-how.com/remote-sensing-from-air-and-space/imaging-system-types-visible-imagery-remote-sensing/`.

Huang, C, L. Davis, and J. Townshend (2002). "An assessment of support vector machines for land cover classification". In: *International Journal of Remote Sensing* 23(4), pp. 725–749.

Jebur, M. N., H. Z. Mohd Shafri, B. Pradhan, and M. S. Tehrany (2013). "Per-pixel and object-oriented classification methods for mapping urban land cover

extraction using SPOT 5 imagery". In: *Geocarto International*(ahead-of-print), pp. 1–15.

Kuhn, M. (2015). *caret: Classification and Regression Training*. R package version 6.0-41. URL: http://CRAN.R-project.org/package=caret.

Laliberte, A. S., J. Koppa, E. L. Fredrickson, and A. Rango (2006). "Comparison of nearest neighbor and rule-based decision tree classification in an object-oriented environment". In: *IEEE International Geoscience and Remote Sensing Symposium and 27th Canadian Symposium on Remote Sensing, July*.

Liaw, A. and M. Wiener (2002). "Classification and regression by randomForest". In: *R news* 2(3), pp. 18–22.

Martin, P. *in.ed.ac.uk - Dispel Tutorial 0.8 documentation - k-fold cross validation*. URL: http://homepages.inf.ed.ac.uk/pmartin/tutorial/case_studies.html.

MODIS. *MODIS*. URL: http://webmap.ornl.gov/wcsdown/dataset.jsp?ds_id=10004.

Mohan, B. K., S. Ladha, and I. CSRE (2009). "Comparison of object based and pixel based classification of high resolution satellite images using artificial neural networks". In: *IIT Bombay, Mumbai*.

Mountrakis, G., J. Im, and C. Ogole (2011). "Support vector machines in remote sensing: A review". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66(3), pp. 247–259.

Myint, S. W., P. Gober, A. Brazel, S. Grossman-Clarke, and Q. Weng (2011). "Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery". In: *Remote Sensing of Environment* 115(5), pp. 1145–1161.

Oruc, M, A. Marangoz, and G Buyuksalih (2004). "Comparison of pixel-based and object-oriented classification approaches using Landsat-7 ETM spectral bands". In: *Proceedings of the ISRPS 2004 Annual Conference*, pp. 19–23.

Otukei, J. and T Blaschke (2010). "Land cover change assessment using decision trees, support vector machines and maximum likelihood classification algorithms". In: *International Journal of Applied Earth Observation and Geoinformation* 12, S27–S31.

Pal, M. and P. M. Mather (2003). "An assessment of the effectiveness of decision tree methods for land cover classification". In: *Remote sensing of environment* 86(4), pp. 554–565.

Platt, R. V. and L. Rapoza (2008). "An Evaluation of an Object-Oriented Paradigm for Land Use/Land Cover Classification". In: *The Professional Geographer* 60(1), pp. 87–100.

Quinlan, J. R. (1986). "Induction of decision trees". In: *Machine learning* 1(1), pp. 81–106.

quora.com. *quora.com - What are Kernels in Machine Learning and SVM?* URL: https://www.quora.com/What-are-Kernels-in-Machine-Learning-and-SVM.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org/.

Richards, J. A. and J. Richards (1999). *Remote sensing digital image analysis*. Vol. 3. Springer.

Rodriguez-Galiano, V., B Ghimire, J Rogan, M Chica-Olmo, and J. Rigol-Sanchez (2012). "An assessment of the effectiveness of a random forest classifier for land-cover classification". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 67, pp. 93–104.

Solutions, E. V. I. *Exelis Visual Information Solutions*. URL: http://www.exelisvis.com/ProductsServices/ENVIProducts/ENVI.aspx.

USGS. *USGS - Using the USGS Landsat 8 Product*. URL: http://landsat.usgs.gov/Landsat8_Using_Product.php.

USGS. *USGS - What are the best spectral bands to use for my study?* URL: http://landsat.usgs.gov/best_spectral_bands_to_use.php.

U.S.G.S. (2013). "Landsat 8: U.S. Geological Survey Fact Sheet". In:

utexas.edu. *utexas.edu - Survey Methods*. URL: https://www.utexas.edu/courses/denbow/labs/survey2.htm.

Weih Jr, R. C. and N. D. Riggan Jr (2010). "Object-Based Classification vs. Pixel-Based Classification: Comparitive Importance of Multi-Resolution Imagery". In: *Proceedings of GEOBIA 2010: Geographic Object-Based Image Analysis* 38, p. 6.

Whiteside, T. and W. Ahmad (2005). "A comparison of object-oriented and pixel-based classification methods for mapping land cover in northern Australia". In: *Proceedings of SSC2005 Spatial intelligence, innovation and praxis: The national biennial Conference of the Spatial Sciences Institute*, pp. 1225–1231.

wikipedia. *wikipedia.org - K-nearest neighbors algorithm*. URL: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

Wikipedia. *wikipedia.org - Shapefile*. URL: https://upload.wikimedia.org/wikipedia/commons/3/38/Simple_vector_map.svg.

Yan, G., J.-F. Mas, B. Maathuis, Z. Xiangmin, and P. Van Dijk (2006). "Comparison of pixel-based and object-oriented image classification approaches—a case study in a coal fire area, Wuda, Inner Mongolia, China". In: *International Journal of Remote Sensing* 27(18), pp. 4039–4055.

Zhou, W., A. Troy, and M. Grove (2008). "Object-based land cover classification and change analysis in the Baltimore metropolitan area using multitemporal high resolution remote sensing data". In: *Sensors* 8(3), pp. 1613–1636.

APPENDIX A

LIST OF R PACKAGES IN THE SYSTEM

Package Version
1 BradleyTerry2 1.0-6
2 brglm 0.5-9
3 car 2.0-25
4 caret 6.0-41
5 colorspace 1.2-6
6 combinat 0.0-8
7 dichromat 2.0-0
8 digest 0.6.8
9 dismo 1.0-12
10 e1071 1.6-4
11 filehash 2.3
12 foreach 1.4.2
13 formatR 1.2
14 gbm 2.1.1
15 ggplot2 1.0.0
16 gridExtra 2.0.0
17 gtable 0.1.2
18 gtools 3.4.1
19 iterators 1.0.7
20 kernlab 0.9-20
21 klaR 0.6-12
22 labeling 0.3
23 lme4 1.1-7
24 manipulate 0.98.1102
25 maptools 0.8-34
26 minqa 1.2.4
27 mlbench 2.1-1

28 munsell 0.4.2
29 nloptr 1.0.4
30 pbkrtest 0.4-2
31 pls 2.4-3
32 plyr 1.8.1
33 pROC 1.8
34 profileModel 0.5-9
35 proto 0.3-10
36 quantreg 5.11
37 randomForest 4.6-10
38 rasclass 0.2.1
39 raster 2.3-24
40 rattle 3.4.1
41 RColorBrewer 1.1-2
42 Rcpp 0.11.5
43 RcppEigen 0.3.2.4.0
44 reshape 0.8.5
45 reshape2 1.4.1
46 rgdal 0.9-1
47 rjson 0.2.15
48 rpart.plot 1.5.2
49 RSNNS 0.4-6
50 rstudio 0.98.1102
51 scales 0.2.4
52 sp 1.0-17
53 SparseM 1.6
54 stringr 0.6.2
55 tikzDevice 0.8.1
56 XML 3.98-1.1
57 base 3.1.2
58 boot 1.3-13
59 class 7.3-11
60 cluster 1.15.3
61 codetools 0.2-9
62 compiler 3.1.2
63 datasets 3.1.2
64 foreign 0.8-61
65 graphics 3.1.2
66 grDevices 3.1.2
67 grid 3.1.2
68 KernSmooth 2.23-13
69 lattice 0.20-29
70 MASS 7.3-35
71 Matrix 1.1-4
72 methods 3.1.2

73 mgcv 1.8-3
74 nlme 3.1-118
75 nnet 7.3-8
76 parallel 3.1.2
77 rpart 4.1-8
78 spatial 7.3-8
79 splines 3.1.2
80 stats 3.1.2
81 stats4 3.1.2
82 survival 2.37-7
83 tcltk 3.1.2
84 tools 3.1.2
85 translations 3.1.2
86 utils 3.1.2

APPENDIX B

2015 IMAGES' CONFUSION MATRICES

B.0.0.1 Random Forest

Table B.1: Random Forest confusion matrix - July 22, 2013 as train, June 26, 2015 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total
Rice	979	0	2	37	0	0	0	0	4	1022
Stover	39	118	0	225	0	25	22	413	18	860
River	0	0	387	0	0	0	0	0	0	387
Maize	66	165	0	1292	31	576	189	212	92	2623
Potato	5	13	0	368	52	99	29	32	8	606
Tomato	9	0	0	302	13	1382	212	38	46	2002
Vineyard	0	3	0	14	0	30	395	14	87	543
Other	0	26	0	1	0	64	27	526	15	659
Vegetation	0	13	0	31	0	0	58	15	757	874
Total	1098	338	389	2270	96	2176	932	1250	1027	9576

Accuracy: 0.615 **Kappa**: 0.545 **95% Confidence Interval**: (0.605,0.625)

Table B.2: Random Forest confusion matrix - July 22, 2013 as train, 12 Jul 15 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total
Rice	819	0	0	178	3	21	1	0	0	1022
Stover	8	121	0	257	0	47	23	393	11	860
River	367	0	20	0	0	0	0	0	0	387
Maize	49	43	0	1385	3	677	179	262	25	2623
Potato	0	22	0	371	34	83	64	22	10	606
Tomato	3	0	0	345	1	1591	23	33	6	2002
Vineyard	4	3	0	5	0	32	403	17	79	543
Other	1	28	0	0	0	59	9	550	12	659
Vegetation	0	25	0	3	0	0	99	15	732	874
Total	1251	242	20	2544	41	2510	801	1292	875	9576

Accuracy: 0.591 **Kappa**: 0.509 **95% Confidence Interval**: (0.581,0.600)

B.0.0.2 Support Vector Machines

Table B.3: SVM confusion matrix - July 22, 2013 as train, June 26, 2015 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total
Rice	936	0	5	45	0	0	0	35	1	1022
Stover	51	100	0	229	0	25	46	409	0	860
River	0	0	387	0	0	0	0	0	0	387
Maize	38	216	0	1364	59	539	231	146	30	2623
Potato	0	19	0	367	86	64	44	18	8	606
Tomato	11	0	0	316	3	1460	157	42	13	2002
Vineyard	0	18	0	18	0	34	468	3	2	543
Other	0	35	0	1	0	76	36	507	4	659
Vegetation	0	38	0	24	0	0	111	53	648	874
Total	1036	426	392	2364	148	2198	1093	1213	706	9576

Accuracy: 0.622 **Kappa**: 0.552 **95% Confidence Interval**: (0.612,0.632)

Table B.4: SVM confusion matrix - July 22, 2013 as train, 12 Jul 15 as test

	Rice	Stover	River	Maize	Potato	Tomato	Vineyard	Other	Vegetation	Total
Rice	944	0	0	3	74	0	1	0	0	1022
Stover	49	126	0	220	0	53	26	382	4	860
River	0	0	183	0	0	0	0	204	0	387
Maize	26	48	0	1428	66	639	185	226	5	2623
Potato	0	27	0	373	52	61	78	15	0	606
Tomato	1	0	0	349	0	1607	15	27	3	2002
Vineyard	1	3	0	8	0	36	475	8	12	543
Other	0	32	0	1	0	61	13	542	10	659
Vegetation	2	8	0	7	0	0	77	22	758	874
Total	1023	244	183	2389	192	2457	870	1426	792	9576

Accuracy: 0.639 **Kappa**: 0.569 **95% Confidence Interval**: (0.629,0.648)

APPENDIX C

R CODE

loader.R

```
# loader.R
#
# This script aims to load values from one image into a data frame
# containing every value extracted from a specific shapefile
# and a specific folder
# This is the basis of this work, it's here the different
# folders can be defined
# and even, which bands to use (in case of the NDVI study
# obsValues was simply changed from 7:13 to 7:12)
# This script loads one image, and, as such, another loader is used
# for the multi image phase

require(rgdal)
require(raster)
require(caret)

#printf <- function(...) invisible(print(sprintf(...)))


# definition of some folder/pattern names, so they may be changed
# if necessary
# each one is self-explanatory
shapefileFolder <- "inputs/shapefiles"
shapefileName <- "jul_2015_NOVO_9classes_en"


```

```
bandsFolder <- "inputs/layers_2013_07_22_MAIN_corte"
imageDate <- substr(bandsFolder,14,24)
bandsPattern <- ".tif$"
outputFolder <- "outputs"

# definition of the values to be used in classification
# - obsValues are the band columns, which in this case are from 7 to 13
# *in this case* the NDVI band is the last one,
# so for the NDVI importance
# study, this can be simply changed to obsValues <- 7:12
# - classValues is the column where the class value is located,
# 1 in our case
# these variables aren't used here but in the classifier files
# columns 2 to 6 are "junk" added by arcmap/its sample manager

obsValues <- 7:13
classValues <- 1

# read shapefile
sdata <- readOGR(dsn=shapefileFolder, layer=shapefileName)
# do not remove
#sdata@data[ order(sdata@data$Classname), ] -> sdata@data


# load bands (all bandsPattern-like files in bandsFolder)
# and join them in a "stack"
bandnames <- list.files(bandsFolder, pattern=bandsPattern)
rlist <- list.files(bandsFolder, pattern=bandsPattern, full.names=TRUE)
xvars <- stack(rlist)

# name filter for the loop
# this is not 100% necessary but is a nice way to
# filter unwanted classes in the case the shapefile has classes
# other than the ones we want to use
# for example if we wanted simply 2-class classification we could do
# names <- c(class1Name, class2Name)
# and the loop would filter out the others

names <- c("Rice","Stover", "River", "Maize","Potato",
       "Tomato","Vineyard", "Other",
       "Vegetation")

# Extract the values from a Raster object
# at the locations of the spatial data provided.
# In this method we extract the values that will constitute
```

```
# our training and test data, from the polygons
# existing in the shapefile.
# all the other points exist only to provide a visualization
# of the images at the end, they are unecessary
# to the classification process
tmp <- extract(xvars,sdata)


# extract creates multiple lists of the values obtained
# this code aims to join them in a data frame with their
#  respective class.names
# e.g. a table containing
#
#
#   class.name | band1 measurement | band2 measurement | ...
#   ____________________________________________________
#   rice       | 0.021             | 0.02              | ...
#   rice       | 0.41              | 0.02              | ...
#   water      | 0.521             | 0.4               | ...
#   ......
#
#
#
# this is the fundamental table for this work,
# containing the data in the format that will be used for classification

mat = c()
for (i in 1:length(tmp)) {
  tmp_b <- cbind(sdata@data[i, ],
            as.data.frame(tmp[i]),
            row.names = NULL)
  tmp_b <- droplevels(tmp_b)
  name <- levels(tmp_b$Classname)
  if(!(name %in% names)) next
  mat <- rbind(mat,tmp_b)
  rm(tmp_b,name)
}

variables.keep <- c('classValues', 'obsValues', 'mat',
'xvars', 'imageDate', 'outputFolder','start.time')
rm(list= ls()[!(ls() %in% variables.keep )])
```

multi_loader.R

```
# multi_loader.R
#
# This script is the same as the loader.R, simply loading other data
# into other variables
# to be used in the multi image phase of this work, since you
# need data from two different dates (folders)



# definition of some folder/pattern names, so they may be changed
# if necessary. each one is self-explanatory
shapefileFolder <- "inputs/shapefiles"
shapefileName <- "jul_2015_NOVO_9classes_en"


bandsFolder <- "inputs/layers_2013_06_20_corte"#_nocorrection"
imageDate.new <- substr(bandsFolder,14,24)
bandsPattern <- ".tif$"
outputFolder <- "outputs"

# definition of the values to be used in classification
# - obsValues are the band columns, which in this case are from 7 to 13
# *in this case* the NDVI band is the last one,
# so for the NDVI importance
# study, this can be simply changed to obsValues <- 7:12
# - classValues is the column where the class value is located,
# 1 in our case
# these variables aren't used here but in the classifier files
# columns 2 to 6 are "junk" added by arcmap/its sample manager

obsValues <- 7:13
classValues <- 1

# read shapefile
sdata <- readOGR(dsn=shapefileFolder, layer=shapefileName)
# TODO: ver se isto da para tirar
#sdata@data[ order(sdata@data$Classname), ] -> sdata@data


# load bands (all bandsPattern-like files in bandsFolder)
# and join them in a "stack"
bandnames <- list.files(bandsFolder, pattern=bandsPattern)
rlist <- list.files(bandsFolder, pattern=bandsPattern, full.names=TRUE)
xvars <- stack(rlist)
```

```
45
46 # name filter for the loop
47 # this is not 100% necessary but is a nice way to
48 # filter unwanted classes in the case the shapefile has classes
49 # other than the ones we want to use
50 # for example if we wanted simply 2-class classification we could do
51 # names <- c(class1Name, class2Name)
52 # and the loop would filter out the others
53
54 names <- c("Rice","Stover", "River", "Maize","Potato",
55        "Tomato","Vineyard", "Other",
56        "Vegetation")
57
58 # Extract the values from a Raster object
59 # at the locations of the spatial data provided.
60 # In this method we extract the values that will constitute
61 # our training and test data, from the polygons
62 # existing in the shapefile.
63 # all the other points exist only to provide a visualization
64 # of the images at the end, they are unecessary
65 # to the classification process
66 tmp <- extract(xvars,sdata)
67
68 # extract creates multiple lists of the values obtained
69 # this code aims to join them in a data frame with their respective
70 # class.names. e.g. a table containing
71 #
72 #
73 #   class.name | band1 measurement | band2 measurement | ...
74 #   ____________________________________________________________
75 #   rice       | 0.021             | 0.02              | ...
76 #   rice       | 0.41              | 0.02              | ...
77 #   water      | 0.521             | 0.4               | ...
78 #   ......
79 #
80 #
81 #
82 # this is the fundamental table for this work,
83 # containing the data in the format that will be used for classification
84
85 new.mat = c()
86 for (i in 1:length(tmp)) {
87   tmp_b <- cbind(sdata@data[i, ],
88             as.data.frame(tmp[i]),
89             row.names = NULL)
```

```
  tmp_b <- droplevels(tmp_b)
  name <- levels(tmp_b$Classname)
  if(!(name %in% names)) next
  new.mat <- rbind(new.mat,tmp_b)
}
```

multi_SVM.R

```
# multi_SVM.R
#
# This script aims to perform SVM classification
# by training with data from one date (image)
# and then testing on another
# and finally generating the image
# the dates are set up in the loader.R/multi_loader.R files

#############################################################
# setup
#############################################################

source("loader.R")
set.seed(42)
data.ratio <- 1
data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]

#############################################################
# every entry from the first loader is now for training
#############################################################
training_x <- data[, obsValues]
training_y <- data[, classValues]

ctrl <- trainControl(method="none")
tuneGrid <- expand.grid(sigma=0.4, C=32)
svm.fit <- train(x = training_x,
         y = training_y,
         method = "svmRadial",
         trControl = ctrl,
        # tuneLength = tuneLength
        tuneGrid = tuneGrid
)

print(svm.fit)

#############################################################
# load the new data, test the algorithm on it
#############################################################
```

```
39
40 source("multi_loader.R")
41 new.data <- new.mat[sample(nrow(new.mat),nrow(new.mat)*data.ratio),]
42 new.test_x <- new.data[ ,obsValues]
43 new.test_y <- new.data[ ,classValues]
44
45 svm.predict <- predict(svm.fit, newdata = new.test_x)
46 cf1 <- confusionMatrix(svm.predict , new.test_y)
47 print(cf1)
48
49 ############################################################
50 # predict the final image and output it to a file
51 ############################################################
52
53 method_name <- "SVM"
54 method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
55 filename1 <- paste(outputFolder,"/multi_",method_name_parsed,
56 imageDate,"TRAIN",imageDate.new,"TEST",sep="")
57
58 finalimg <- predict(xvars, svm.fit, type="raw",
59           index=1, na.rm=TRUE, progress="text")
60 levels(finalimg)
61
62 writeRaster(finalimg, filename=filename1, format="GTiff", overwrite=TRUE)
```

multi_randomForest.R

```
1  # multi_randomForest.R
2  #
3  # This script aims to perform random forest classification
4  # by training with data from one date (image)
5  # and then testing on another
6  # and finally generating the image
7  # the dates are set up in the loader.R/multi_loader.R files
8
9  ############################################################
10 # setup
11 ############################################################
12
13 source("loader.R")
14 set.seed(42)
15 data.ratio <- 1
16 data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]
17
18 ############################################################
19 # every entry from the first loader is now for training
```

```
############################################################
training_x <- data[, obsValues]
training_y <- data[, classValues]


ctrl <- trainControl(method="none")
tuneGrid <- expand.grid(mtry=2)
randomForest.fit <- train(x = training_x,
              y = training_y,
              method = "rf",
              trControl = ctrl,
              tuneGrid = tuneGrid
)

print(randomForest.fit)

############################################################
# load the new data, test the algorithm on it
############################################################

source("multi_loader.R")
new.data <- new.mat[sample(nrow(new.mat),nrow(new.mat)*data.ratio),]
new.test_x <- new.data[ ,obsValues]
new.test_y <- new.data[ ,classValues]

randomForest.predict <- predict(randomForest.fit, newdata = new.test_x)
cf1 <- confusionMatrix(randomForest.predict , new.test_y)
print(cf1)

############################################################
# predict the final image and output it to a file
############################################################

method_name <- "random␣forest"
method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
filename1 <- paste(outputFolder,"/multi_",method_name_parsed,
imageDate,"TRAIN",imageDate.new,"TEST",sep="")

finalimg <- predict(xvars, randomForest.fit, type="raw",
          index=1, na.rm=TRUE, progress="text")
levels(finalimg)

writeRaster(finalimg, filename=filename1, format="GTiff", overwrite=TRUE)
```

percentages.R

```
##############################################################
# percentages.R
#
# This script aims to analyze how using only
# a percentage of the initial data
# affects the classification algorithms
# we used 5, 10, 25, 50, 75 and 100%
# the output is somewhat ready to be exported into LaTeX
# but the results variable contains all the information
# in case plots need to be generated
##############################################################

source("loader.R")
source("mlc/MLC.R")
library(xtable)
set.seed(42)
results <- list(c(),c(),c(),c(),c(),c(),c(),c(),c(),c())
values <- c(0.05,0.1,0.25,0.5,0.75,1)
for(i in values) {
  data.ratio <- i

  data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]
  prop.table(table(data$Classname))*100
  # generate indexes for split 1/3 -> test, 2/3 -> train

  indxTrain <- createDataPartition(y=data$Classname,
                                   p=0.66,
                                   list=FALSE)

  training_x <- data[indxTrain, obsValues]
  test_x <- data[-indxTrain,obsValues]

  training_y <- data[indxTrain, classValues]
  test_y <- data[-indxTrain, classValues]
  prop.table(table(training_y))*100
  prop.table(table(test_y))*100

  #createFolds(training_y, 3) -> flds
  ctrl <- trainControl(method="none")

###################### RF

  tuneGrid <- expand.grid(mtry=2)
  randomForest.fit <- train(x = training_x,
```

```
                              y = training_y,
                              method = "rf",
                              trControl = ctrl,
                              tuneGrid = tuneGrid
  )


  randomForest.predict <- predict(randomForest.fit, newdata = test_x)
  confusionMatrix(randomForest.predict , test_y) -> cf1

  results[[1]] <- rbind(results[[1]] ,cf1$overall["Kappa"])
  results[[2]]  <- rbind(results[[2]] , cf1$overall["Accuracy"])

###################### KNN
tuneGrid <- expand.grid(k=3)
kNN.fit <- train(x =  training_x,
         y = training_y,
         method = "knn",
         trControl = ctrl,
         tuneGrid = data.frame(k=3)
)

  #kNN.fit
  kNN.predict <- predict(kNN.fit, newdata = test_x)
  confusionMatrix(kNN.predict , test_y) -> cf2

 results[[3]] <- rbind(results[[3]] ,cf2$overall["Kappa"])
 results[[4]]  <- rbind(results[[4]] , cf2$overall["Accuracy"])
 # print(cf1)

###################### DT
tuneGrid <- expand.grid(cp=0.0008)
rpart.fit <- train(x = training_x,
                 y = training_y,
                 method = "rpart",
                 trControl = ctrl,
                 tuneGrid = tuneGrid
)

#kNN.fit
rpart.predict <- predict(rpart.fit, newdata = test_x)
confusionMatrix(rpart.predict , test_y) -> cf3

results[[5]] <- rbind(results[[5]] ,cf3$overall["Kappa"])
results[[6]]  <- rbind(results[[6]] , cf3$overall["Accuracy"])
```

```
90
91
92  ###################### SVM
93  tuneGrid <- expand.grid(sigma=0.4, C=32)
94  svm.fit <- train(x = training_x,
95           y = training_y,
96           method = "svmRadial",
97           trControl = ctrl,
98           # tuneLength = tuneLength
99           tuneGrid = tuneGrid
100 )
101
102 #kNN.fit
103 svm.predict <- predict(svm.fit, newdata = test_x)
104 confusionMatrix(svm.predict , test_y) -> cf4
105
106 results[[7]] <- rbind(results[[7]] ,cf4$overall["Kappa"])
107 results[[8]]  <- rbind(results[[8]] , cf4$overall["Accuracy"])
108
109 ##################### MLC
110
111 mlcInit <- list(coefs=list(),classes="")
112 class(mlcInit) <- append(class(mlcInit),"MLC")
113 names <- levels(data$Classname)
114 trainvalues <- cbind(training_y,training_x)
115 mlcObject <- trainMLC(mlcInit,trainvalues,names)
116
117 MLC.predict <- predict.MLC(mlcObject, newdata = test_x)
118 levels(MLC.predict) <- levels(test_y)
119 MLC.confusionMatrix <- confusionMatrix(MLC.predict , test_y)
120
121 results[[9]] <- rbind(results[[9]] ,
122 MLC.confusionMatrix$overall["Kappa"])
123 results[[10]]  <- rbind(results[[10]] ,
124  MLC.confusionMatrix$overall["Accuracy"])
125
126 }
127 list.titles <- c("RandomForest", "kNN", "DecisionTree",
128 "SupportVectorMachines", "MaximumLikelihood")
129 for(j in 1:5) {
130   cbind(c("5%","10%","25%","50%","75%","100%"),
131   signif(results[[2*j]],3),signif(results[[2*j-1]],3)) -> a
132   print(list.titles[j])
133   print(xtable(a))
134 }
```

plotHistograms.R

```
1  ############################################################
2  # plothistograms.R
3  #
4  # This script aims to generate boxplot graphs for each of the classes.
5  # This is not a part of the actual classification methods procedure,
6  # but instead, a first look at how different the spectral signatures
7  # for the different classes are.
8  #
9  # The aim of this script is purely to generate the graphs and
10 # visually perform a preliminar prediction
11 # of how well the classifiers might perform.
12 #
13 # e.g. class1 is highly unlike all the others so it should be
14 # easy to classify,
15 # class2 and class3 are very similar so there's a high chance
16 # they get misclassified, etc.
17 ############################################################
18
19 library("ggplot2")
20 source("loader.R")
21
22 ############################################################
23 # plot function
24 # just a few settings to make the plots prettier
25 # comments inside when relevant
26 ############################################################
27
28
29 plotting <- function(data_f, plot_name){
30   update_geom_defaults("point", list(colour = NULL))
31   bottom.y.lim <- 0
32   p <- ggplot(data_f, aes(factor(data_f[,1]), data_f[,2])) +
33     # boxplot settings
34     stat_boxplot(geom ='errorbar') +
35     geom_boxplot(fill='#a4a4a4', color="black", outlier.colour = "black",
36            outlier.shape = 16, outlier.size = 2) +
37     # theme settings
38     theme(axis.title.x=element_text(size=22),
39         axis.title.y=element_text(size=22),
40         axis.text = element_text(size = 18),
41         axis.text.y = element_text(size = 18),
42         legend.text=element_text(size=18),
43         legend.title=element_blank()) +
44     # labels
```

```
45      labs(x = 'Band␣num/name', y = 'Response') +
46
47
48      # y limits 0 to 1 if not river
49      # workaround since no other classes have <0 values
50      # so it makes it easier to distinguish
51      if(plot_name != "River") ylim(0, 1)
52    print(p)
53    #ggsave(file=paste(plot_name, ".png", sep=""),plot=p)
54    print(plot_name)
55  }
56
57
58  ############################################################
59  # plot loop
60  # the cbind arguments are the names of the bands
61  # and a filter of the original table for the class and that
62  # specific band, so for each class we will have a table as
63  #
64  #   band name  | measurement
65  #   ___________________________
66  #   NDVI       | 0.021
67  #   NDVI       | 0.41
68  #              ...
69  #   SWIR1      | 0.521
70  #              ...
71  #
72  # this table is then passed as an argument to the plotting function
73  # generating the boxplot (and saving the .svg files)
74  ############################################################
75
76  axis.x <- c("02␣Blue","03␣Green","04␣Red","05␣NIR",
77        "06␣SWIR1","07␣SWIR2","08␣NDVI")
78  bla <- c()
79  for(class.name in unique(mat[, 1])) {
80    filter <- c()
81    for(bandNumber in 2:8) {
82      tmp_3 <- cbind(axis.x[bandNumber-1],
83                     (mat[mat["Classname"]==class.name,bandNumber+5]))
84      filter <- rbind(tmp_3,filter)
85    }
86    data_f <- as.data.frame(filter)
87    # convert data to a numeric value instead of string
88    data_f[, 2] <- as.numeric(as.character(data_f[, 2]))
89    bla <- data_f
```

```
90    plotting(data_f, class.name)
91
92  }
```

si_decisionTree.R

```
1   # si_decisionTree.R
2   #
3   # This script aims to perform single image
4   # decision tree classification
5   # and generating a full image based on it
6
7   ###############################################################
8   # setup
9   ###############################################################
10
11  source("loader.R")
12  set.seed(42)
13  data.ratio <- 1
14  data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]
15
16  ###############################################################
17  # generate indexes for split 1/3 -> test, 2/3 -> train
18  ###############################################################
19
20  indxTrain <- createDataPartition(y=data$Classname,
21                                   p=0.66,
22                                   list=FALSE)
23
24  training_x <- data[indxTrain, obsValues]
25  test_x <- data[-indxTrain,obsValues]
26
27  training_y <- data[indxTrain, classValues]
28  test_y <- data[-indxTrain, classValues]
29
30  ###############################################################
31  # can be uncommented
32  # just to confirm the splits are done
33  # proportionally to each available class,
34  # and not (100/number of classes)%
35  ###############################################################
36
37  #print(prop.table(table(data$Classname))*100)
38  #print(prop.table(table(training_y))*100)
39  #print(prop.table(table(test_y))*100)
40
```

```
41 ############################################################
42 # separate into three folds for the inner cross-validation
43 # set up tune grid for this particular model
44 # and perform the parameter tuning / training
45 # output a summary of the results
46 ############################################################
47
48 flds <- createFolds(training_y, 3)
49 ctrl <- trainControl(method="cv", index=flds, number=3)
50 tuneGrid <- expand.grid(cp=seq(0.0002,0.02,0.0002))
51
52 rpart.fit <- train(x = training_x,
53                    y = training_y,
54                    method = "rpart",
55                    trControl = ctrl,
56                    tuneGrid = tuneGrid
57 )
58 print(rpart.fit)
59
60 ############################################################
61 # predict the test data
62 # and generate the confusion matrix
63 ############################################################
64
65 rpart.predict <- predict(rpart.fit, newdata = test_x)
66 rpart.confusionMatrix <- confusionMatrix(rpart.predict , test_y)
67 print(rpart.confusionMatrix)
68
69 ############################################################
70 # predict the final image and output it to a file
71 ############################################################
72
73 cat("predicting␣the␣final␣image..\n")
74 method_name <- "decision␣tree"
75 method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
76 filename1 <- paste(outputFolder,"/w1_",method_name_parsed,
77 imageDate,sep="")
78
79 finalimg <- predict(xvars, rpart.fit, type="raw",
80                     index=1, na.rm=TRUE, progress="text")
81
82 writeRaster(finalimg,
83       filename=filename1,
84       format="GTiff", overwrite=TRUE)
```

si_kNN.R

```
# si_kNN.R
#
# This script aims to perform single image
# k-nn classification
# and generating a full image based on it

############################################################
# setup
############################################################

source("loader.R")
set.seed(42)
data.ratio <- 1
data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]

############################################################
# generate indexes for split 1/3 -> test, 2/3 -> train
############################################################

indxTrain <- createDataPartition(y=data$Classname,
                                 p=0.66,
                                 list=FALSE)

training_x <- data[indxTrain, obsValues]
test_x    <- data[-indxTrain, obsValues]

training_y <- data[indxTrain, classValues]
test_y <- data[-indxTrain, classValues]

############################################################
# can be uncommented
# just to confirm the splits are done
# proportionally to each available class,
# and not (100/number of classes)%
############################################################

#print(prop.table(table(data$Classname))*100)
#print(prop.table(table(training_y))*100)
#print(prop.table(table(test_y))*100)

############################################################
# separate into three folds for the inner cross-validation
# set up tune grid for this particular model
# and perform the parameter tuning / training
```

```
# output a summary of the results
############################################################


flds <- createFolds(training_y, 3)
ctrl <- trainControl(method="cv", index=flds, number=3)
tuneGrid <- expand.grid(k=seq(3,21,2))

kNN.fit <- train(x =  training_x,
                 y = training_y,
                 method = "knn",
                 trControl = ctrl,
                tuneGrid = tuneGrid
                )

print(kNN.fit)

############################################################
# predict the test data
# and generate the confusion matrix
############################################################

kNN.predict <- predict(kNN.fit, newdata = test_x)
kNN.confusionMatrix <- confusionMatrix(kNN.predict , test_y)
print(kNN.confusionMatrix)

############################################################
# predict the final image and output it to a file
############################################################

method_name <- "knn"
method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
filename1 <- paste(outputFolder,"/w1_",method_name_parsed,
imageDate,sep="")

finalimg <- predict(xvars, kNN.fit, type="raw",
          index=1, na.rm=TRUE, progress="text")

writeRaster(finalimg,
      filename=filename1,
      format="GTiff", overwrite=TRUE)
```

si_randomForest.R

```
# si_randomForest.R
#
# This script aims to perform single image
# random forest classification
# and generating a full image based on it

############################################################
# setup
############################################################

source("loader.R")
set.seed(42)
data.ratio <- 1
data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]

############################################################
# generate indexes for split 1/3 -> test, 2/3 -> train
############################################################

indxTrain <- createDataPartition(y=data$Classname,
                 p=0.66,
                 list=FALSE)

training_x <- data[indxTrain, obsValues]
test_x    <- data[-indxTrain, obsValues]

training_y <- data[indxTrain, classValues]
test_y <- data[-indxTrain, classValues]

############################################################
# can be uncommented
# just to confirm the splits are done
# proportionally to each available class,
# and not (100/number of classes)%
############################################################

#print(prop.table(table(data$Classname))*100)
#print(prop.table(table(training_y))*100)
#print(prop.table(table(test_y))*100)

############################################################
# separate into three folds for the inner cross-validation
# set up tune grid for this particular model
# and perform the parameter tuning / training
```

```
# output a summary of the results
############################################################


flds <- createFolds(training_y, 3)
ctrl <- trainControl(method="cv", index=flds, number=3)

tuneGrid <- expand.grid(mtry=2:5)
randomForest.fit <- train(x = training_x,
                y = training_y,
                method = "rf",
                trControl = ctrl,
                tuneGrid = tuneGrid
                )

print(randomForest.fit)

############################################################
# predict the test data
# and generate the confusion matrix
############################################################

randomForest.predict <- predict(randomForest.fit, newdata = test_x)
randomForest.confusionMatrix <-
confusionMatrix(randomForest.predict , test_y)
print(randomForest.confusionMatrix)

############################################################
# predict the final image and output it to a file
############################################################

method_name <- "random␣forest"
method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
filename1 <- paste(outputFolder,"/w1_",method_name_parsed,
imageDate,sep="")

finalimg <- predict(xvars, randomForest.fit, type="raw",
          index=1, na.rm=TRUE, progress="text")

writeRaster(finalimg,
      filename=filename1,
      format="GTiff", overwrite=TRUE)
```

si_MLC.R

```
# si_MLC.R
#
# This script aims to perform single image
# maximum likelihood classification
# and generating a full image based on it

############################################################
# setup
############################################################

source("loader.R")
source("mlc/MLC.R")
set.seed(42)
data.ratio <- 1
data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]

############################################################
# generate indexes for split 1/3 -> test, 2/3 -> train
############################################################

indxTrain <- createDataPartition(y=data$Classname,
                 p=0.66,
                 list=FALSE)

training_x <- data[indxTrain, obsValues]
test_x    <- data[-indxTrain, obsValues]

training_y <- data[indxTrain, classValues]
test_y <- data[-indxTrain, classValues]

############################################################
# can be uncommented
# just to confirm the splits are done
# proportionally to each available class,
# and not (100/number of classes)%
############################################################

#print(prop.table(table(data$Classname))*100)
#print(prop.table(table(training_y))*100)
#print(prop.table(table(test_y))*100)

############################################################
# create maximum likelihood object
# and train it using the training data
```

```
# there is no parameter tuning here, and as such
# no need for aditional splitting of the training data
# into three folds
############################################################

mlcInit <- list(coefs=list(),classes="")
class(mlcInit) <- append(class(mlcInit),"MLC")
names <- levels(data$Classname)
trainvalues <- cbind(training_y,training_x)
mlcObject <- trainMLC(mlcInit,trainvalues,names)

############################################################
# predict the test data
# and generate the confusion matrix
############################################################

MLC.predict <- predict.MLC(mlcObject, newdata = test_x)
levels(MLC.predict) <- levels(test_y)
MLC.confusionMatrix <- confusionMatrix(MLC.predict , test_y)
print(MLC.confusionMatrix)

############################################################
# predict the final image and output it to a file
############################################################

method_name <- "mlc"
method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
filename1 <- paste(outputFolder,"/w1_",method_name_parsed,
imageDate,sep="")

finalimg <- predict(xvars, mlcObject, type="response", fun=predict.MLC,
          index=1, na.rm=TRUE, progress="text")

writeRaster(finalimg,
      filename=filename1,
      format="GTiff", overwrite=TRUE)
```

si_SVM.R

```
# si_SVM.R
#
# This script aims to perform single image
# SVM classification
# and generating a full image based on it

############################################################
```

```
8  # setup
9  ############################################################
10
11 source("loader.R")
12 set.seed(42)
13 data.ratio <- 1
14 data <- mat[sample(nrow(mat),nrow(mat)*data.ratio),]
15
16 ############################################################
17 # generate indexes for split 1/3 -> test, 2/3 -> train
18 ############################################################
19
20 indxTrain <- createDataPartition(y=data$Classname,
21                  p=0.66,
22                  list=FALSE)
23
24 training_x <- data[indxTrain, obsValues]
25 test_x    <- data[-indxTrain, obsValues]
26
27 training_y <- data[indxTrain, classValues]
28 test_y <- data[-indxTrain, classValues]
29
30 ############################################################
31 # can be uncommented
32 # just to confirm the splits are done
33 # proportionally to each available class,
34 # and not (100/number of classes)%
35 ############################################################
36
37 #print(prop.table(table(data$Classname))*100)
38 #print(prop.table(table(training_y))*100)
39 #print(prop.table(table(test_y))*100)
40
41 ############################################################
42 # separate into three folds for the inner cross-validation
43 # set up tune grid for this particular model
44 # and perform the parameter tuning / training
45 # output a summary of the results
46 ############################################################
47
48
49 flds <- createFolds(training_y, 3)
50 ctrl <- trainControl(method="cv", index=flds, number=3)
51
52 tuneGrid <- expand.grid(sigma=seq(0.2,2,0.2),
```

```
            C=c(0.25,0.5,1,2,4,8,16,32))
svm.fit <- train(x = training_x,
         y = training_y,
         method = "svmRadial",
         trControl = ctrl,
         # tuneLength = tuneLength
         tuneGrid = tuneGrid
)
print(svm.fit)

#############################################################
# predict the test data
# and generate the confusion matrix
#############################################################

svm.predict <- predict(svm.fit, newdata = test_x)
svm.confusionMatrix <- confusionMatrix(svm.predict , test_y)
print(svm.confusionMatrix)

#############################################################
# predict the final image and output it to a file
#############################################################

method_name <- "svm"
method_name_parsed <- gsub('([[:punct:]])|\\s+','_',method_name)
filename1 <- paste(outputFolder,"/w1_",method_name_parsed,
imageDate,sep="")

finalimg <- predict(xvars, svm.fit, type="raw",
          index=1, na.rm=TRUE, progress="text")

writeRaster(finalimg,
      filename=filename1,
      format="GTiff", overwrite=TRUE)
```

MLC.R

```
# MLC.R
#
# functions adapted from the rasclass package
# this is the BARE minimum code modifications from the rasclass
# package functions
# and, as such, not very commented


trainMLC <- function(x,data,names) { UseMethod("trainMLC",x)}
trainMLC.MLC <- function(x,data,names) {
  byClass <- split(data[,2:ncol(data)], data[,1])
  samplesize <- length(na.omit(data[, 1]))
 x$classes <- 1:length(names(byClass))

  x$classnames <- names
  x$byClass <- byClass

  for(cat in names(byClass)){
    frame <- byClass[[cat]]

    # Calculate parameters of the multivariate normal distribution
    prior <- log(nrow(frame)/samplesize)
    meanVector  <- colMeans(frame)
    classCov <- cov(frame)
    if(sum(diag(classCov) == 0) != 0){
      failnames <- names(diag(classCov))[diag(classCov) == 0]
      warning('No variation of variable(s) "',
      paste(failnames, collapse = ', '), '" within class ',
       cat,'\n. Ignoring variable for prediction in this class.')
      diag(classCov)[diag(classCov) == 0] <- 1
    }
    determinant <- log(det(classCov))
    inverseCov  <- solve(classCov)
    x$coefs[[cat]] <- list(prior, determinant, meanVector, inverseCov)

  }


  return(x)
}

predict.MLC <- function (object, newdata,...){

  coefs <- object$coefs
```

```
  classes <- object$classes
  newdata -> varlist
  dataVars <- as.matrix(varlist)
  predicted <- rep(NA, nrow(dataVars))
  probs <- rep(NA, length(classes))
  for(i in 1:nrow(dataVars)){
    for(j in 1:length(probs)){
      delta <- dataVars[i,] - coefs[[j]][[3]]
      probs[j] <- coefs[[j]][[1]] - coefs[[j]][[2]]
      - t(delta)%*%coefs[[j]][[4]]%*%delta
    }


    classnames <- object$classnames
    pred_tmp <- classes[probs==max(probs)]

    predicted[i] <- pred_tmp



  }

  as.factor(predicted)
}
```