




## Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in : <http://oatao.univ-toulouse.fr/>  
Eprints ID : 18649

**To cite this version :**

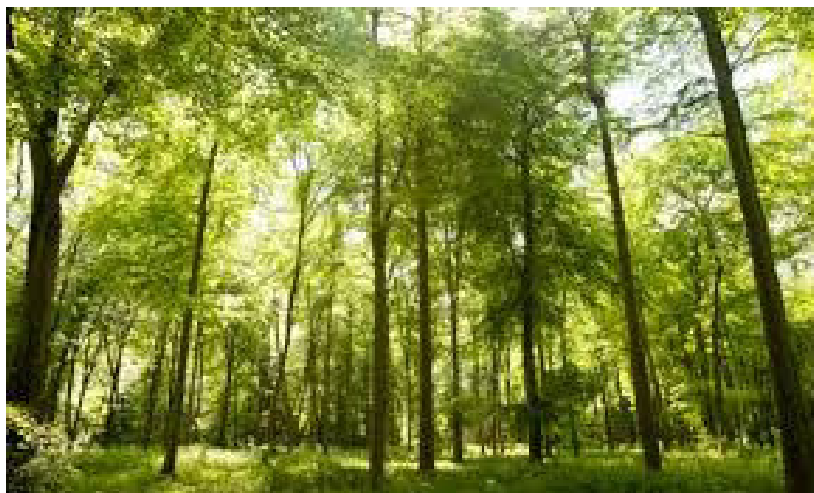
Josipović, Veliborka  *Tree species discrimination in temperate woodland using high spatial resolution Formosat-2 time series.*  
(2015) [Mémoire]

Any correspondence concerning this service should be sent to the repository administrator: [staff-oatao@listes-diff.inp-toulouse.fr](mailto:staff-oatao@listes-diff.inp-toulouse.fr)

**Université de Toulouse**  
**École nationale supérieure d'électronique, d'électrotechnique,**  
**d'informatique, d'hydraulique et des télécommunications (ENSEEIHT)**

**MASTER OF SCIENCE AND TECHNOLOGY**  
Electronic Systems for Embedded and Communicating Applications (ESECA)  
Signal and Image Processing

**Tree species discrimination in temperate woodland using  
high spatial resolution Formosat-2 time series**



**Veliborka Josipović**

UMR 1201 DYNAFOR  
Dynamiques et écologie des paysages agriforestiers



**Supervisors:**  
**Mathieu Fauvel**  
**David Sheeren**

**September 2015**

## **Acknowledgment**

I would like to thank my supervisors, Mathieu Fauvel and David Sheeren, for their guidance and inspiration during the work on this study. I would also like to thank my family and friends for their help and support.

## Abstract

Assessment and mapping of the tree species distribution is an important technical task for forest ecosystem services and habitat monitoring. Since traditional methods (e.g. field surveys) used for the mapping of the tree species tend to be time consuming, date lagged and too expensive, a technology of remote sensing might potentially offer a practical solution for the problem of tree species mapping, especially over large areas.

The main purpose of this study was to investigate the potential of Formosat-2 multi-spectral image time series for classification of the tree species in temperate woodlands. Since phenological variations might increase spectral separability of the trees species, additional aim of the study was to assess the possibility of using multispectral-image time series as an alternative to hyper-spectral data for forest type mapping. Noise from the Formosat-2 images was removed with the Whittaker smoother algorithm, which performed quite well although some additional work might be needed during the selection of the optimal regularization parameter. Several supervised classification methods, Support Vector Machines (SVM), Random Forest (RF) and Gaussian Mixture Model (GMM), were used to discriminate tree species from the image time series. All of the classifiers performed reasonably well, with classification accuracies from 88.5 % to 99.2 % (Kappa statistic), although SVM model was the most accurate, while GMM was the most efficient in terms of computing time. High classification accuracy also indicated that the multi-spectral image time series and remote sensing might be a useful method for the mapping of tree species.

## List of figures

Figure 1.1. Basic principle of NDVI. The healthy vegetation absorbs more Red light (left) than the unhealthy vegetation (right). Source: <a href="http://earthobservatory.nasa.gov">http://earthobservatory.nasa.gov</a> . ....	3
Figure 3.1 Map of France with highlighted study area and typical landscape for study area .	19
Figure 3.2. Reference pixels distributed over the study area. Aerial photograph (left) and zoomed part of the reference pixels (right).....	20
Figure 3.3. Multispectral image and cloud mask acquired on 12/01/2012.....	22
Figure 3.4. Forest/non-forest map for the Haute-Garonne department and for the study area (zoomed part). Area covered by forest is presented in green colour. ....	23
Figure 4.1 Simplified flow-chart of the project .....	24
Figure 4.2. Simplified flow-chart of the pre-processing part of the project .....	25
Figure 5.1. Generalized and Ordinary Cross-Validation errors for different values of $\lambda$ for year 2013.....	29
Figure 5.2. Temporal profiles of the pixel in blue band before and after smoothing in case without clouds or cloud shadows detected. The temporal profiles are superimposed.....	30
Figure 5.3. Temporal profiles of the pixel in blue band affected by cloud before and after smoothing (left) and temporal profiles of the pixel in infra-red band affected by cloud shadow before and after smoothing (right) .....	30
Figure 5.4. Temporal profiles of the pixel containing NDVI values before and after smoothing affected by clouds .....	31
Figure 5.5. Temporal profiles of the pixel in red band before and after smoothing affected by clouds present on two dates .....	31
Figure 5.6. The grey-scale images with the clouds present in top-right corner before (left) and after (right) smoothing .....	32
Figure 5.7. Test classification accuracy for the different size of feature subsets selected using SFS or SFSS algorithms. Classification was based on the spectral bands together with the NDVI indices for level 2 of classification .....	34
Figure 5.8. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).....	40
Figure 5.9. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).....	41

Figure 5.10. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).....	41
Figure 5.11. Classification maps for level 1 based on spectral bands with GMM (left) and SVM (right).....	43
Figure 5.12. Classification maps for level 2 based on spectral bands with GMM (left) and SVM (right).....	43
Figure 5.13. Classification maps for level 3 based on spectral bands with GMM (left) and SVM (right).....	43
Figure 5.14. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 1 based on spectral bands with GMM (left) and SVM (right).....	44
Figure 5.15. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 2 based on spectral bands with GMM (left) and SVM (right).....	44
Figure 5.16. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 3 based on spectral bands with GMM (left) and SVM (right).....	44
Figure A1.1. The test classification accuracy for Hekla data set (top left), KSC (top right) and Pavia (bottom left) .....	48
Figure A1.2. Learning time needed to extract 15 best features for each iteration for Hekla data set (top left), KSC (top right) and Pavia (bottom left) .....	49
Figure A3.1. Histograms for Generalized and Ordinary Cross-Validation errors for different values of $\lambda$ for year 2011.....	71
Figure A3.2. Histograms for Generalized and Ordinary Cross-Validation errors for different values of $\lambda$ for year 2012.....	71
Figure A3.3. Histograms for Generalized and Ordinary Cross-Validation errors for different values of $\lambda$ for year 2014.....	71
Figure A3.4. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).....	84
Figure A3.5. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).....	84
Figure A3.6. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).....	85

Figure A3.7. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).....	85
Figure A3.8. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).....	86
Figure A3.9. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).....	86
Figure A3.10. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).....	87
Figure A3.11. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).....	87
Figure A3.12. The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).....	88
Figure A4.1. The influence of the regularization parameter on smoothing of the simulated data with a second-order penalty (d=2) for several values of the parameter $\lambda$ .....	89
Figure A4.2. Generalized and Ordinary Cross-Validation errors for different values of $\lambda$ .....	90
Figure A4.3 Smoothing the simulated data with $\lambda_{GCV}$ (on the left) and $\lambda_{OCV}$ (on the right) ...	90

## List of tables

Table 3.1. The number of the reference pixels for 14 tree species analyzed in the study .....	20
Table 3.2. Characteristics of the Formosat-2 satellite sensor (Site Airbus Defence and Space) .....	21
Table 3.3. Meaning of the bits in the mask of clouds and cloud shadows.....	21
Table 3.4. Formosat-2 imagery, available dates and number of acquired images for each year .....	22
Table 5.1 Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for year 2013. NPFS_sfs denotes NPFS GMM based classifier computed with a set of features that was chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over 50 repetitions in percentages. The best results for each level are reported in bold face.....	33
Table 5.2 Overall accuracy and Kappa statistics for three defined levels of classification for each of the four years. Only spectral bands were used as the spectral features. NPFS_sfs denotes NPFS GMM based classifier computed with a set of features that was chosen using standard feature selection algorithm described in Section 2.3.1. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over 50 repetitions in percentages. The best results for each level were reported in bold face.....	35
Table 5.3. Confusion matrix for level 1, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	36
Table 5.4. Confusion matrix for level 1, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	37
Table 5.5. Confusion matrix for level 2, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages .....	37
Table 5.6. Confusion matrix for level 2, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	37
Table 5.7. Confusion matrix for level 3, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...	38



Table 5.8. Confusion matrix for level 3, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...39

Table A1.1. Mean values of the overall accuracies and Kappa statistics for each data set ....48

Table A1.2 The mean processing time for learning and prediction for each data set.....49

Table A3.1. Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2011. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.....72

Table A3.2. Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2012. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.....73

Table A3.3. Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2014. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.....74

Table A3.4. Confusion matrix for level 1, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....75

Table A3.5. Confusion matrix for level 1, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....75

Table A3.6. Confusion matrix for level 2, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages .....75

Table A3.7. Confusion matrix for level 2, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....75

Table A3.8. Confusion matrix for level 3, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...	76
Table A3.9. Confusion matrix for level 3, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...	77
Table A3.10. Confusion matrix for level 1, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	78
Table A3.11. Confusion matrix level 1, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.....	78
Table A3.12. Confusion matrix for level 2, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages .....	78
Table A3.13. Confusion matrix for level 2, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	78
Table A3.14. Confusion matrix for level 3, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...	79
Table A3.15. Confusion matrix for level 3, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...	80
Table A3.16. Confusion matrix for level 1, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	81
Table A3.17. Confusion matrix for level 1, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	81
Table A3.18. Confusion matrix for level 2, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages .....	81
Table A3.19. Confusion matrix for level 2, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. ....	81

Table A3.20. Confusion matrix for level 3, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...82

Table A3.21. Confusion matrix for level 3, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour. ...83

# Table of contents

Acknowledgment

Abstract

List of figures

List of tables

<b>1 Introduction.....</b>	<b>1</b>
1.1 General introduction .....	1
1.2 Remote sensing and forest .....	1
1.3 Working laboratory and study objectives .....	3
<b>2 Theoretical background .....</b>	<b>6</b>
2.1 Background of smoothing filter .....	6
2.1.1 Whittaker smoother for equally spaced data .....	6
2.1.2 Whittaker smoother for unequally spaced data .....	9
2.1.3 Choosing a value for smoothing parameter $\lambda$ .....	11
2.2 Background of feature selection .....	13
2.2.1 Non Linear Parsimonious Feature Selection (NPFS).....	13
2.2.2 The Sequential Forward Floating Selection algorithm (SFFS) .....	17
<b>3 Study area and data collection.....</b>	<b>19</b>
3.1 Study area.....	19
3.2 Field data collection.....	19
3.3 Remote sensing data .....	21
3.4 Ancillary data.....	23
<b>4 Methodology .....</b>	<b>24</b>
4.1 General procedure .....	24
4.2 Pre-processing.....	24

4.3 Classification and accuracy assessment.....	26
4.4 Used software and implementation.....	28
<b>5 Results and discussion .....</b>	<b>29</b>
5.1 Pre-processing.....	29
5.2 Classification.....	32
<b>6 Conclusion and future work .....</b>	<b>45</b>
<b>Appendix 1.....</b>	<b>47</b>
<b>Appendix 2.....</b>	<b>51</b>
<b>Appendix 3.....</b>	<b>71</b>
<b>Appendix 4.....</b>	<b>89</b>
<b>References.....</b>	<b>91</b>

# **1 Introduction**

## **1.1 General introduction**

Forest is one of dominant terrestrial ecosystem types, providing essential services to the human society, e.g. wood production, climate control, habitat for animal and plant species, carbon sink, water, human recreation [1]. Since the total area under forests is vast, around 4 billion hectares or 31% of the total land area [2], assessment of the forests current and future states is very important.

To correctly estimate the forest state in one area it is necessary to first produce distribution map of the tree species in the study area. Traditionally, distributions of the species were estimated by ground based surveys during forest inventories. However, it is usually better to have tree species distribution maps already during the planning phases of the forest inventory in order to allocate resources and train ground based crews in time. Ground survey is still by far the most accurate and detailed way of forest monitoring, although it is very elaborate, time consuming, and expensive. One of the possible alternatives for consistent and continuous monitoring is to use remote sensing and automated image analysis techniques [3]. Therefore, remote sensing approach for mapping tree species has been researched for a very long time [4]. Nevertheless, accurate estimation of the tree species distributions from remote sensing data is still a very difficult problem, since there are many factors influencing spectral response of species, e.g. tree age, vegetation phase, tree vitality, presence or absence of the understory.

## **1.2 Remote sensing and forest**

Remote sensing is a scientific discipline which analyses and interprets measurements of electromagnetic radiation (EMR) that is reflected from or omitted by a target and observed or recorded from a vantage point by an observer or instrument that is not in contact with the target. Remote sensing can be active or passive, depending on whether the acquired signal was transmitted from a natural source like the sun or it was emitted from an artificial source such as sensors. It is often used for earth observation, which is done by interpreting and understanding EMR measurements of objects on the Earth's land, ocean or ice surfaces and which are usually made by satellite, together with making relationships between these measurements and the nature of phenomena on the Earth's surface [5].

Several satellite sensors were launched to collect useful data from Earth's surface in the last decades. Satellite sensors have different spatial, spectral and temporal resolution depending on their function and orbit. The spatial resolution defines the minimum size of an object that can be detected in an image, which determines the pixel size of the images covering the Earth surface. The spectral resolution defines the ability of a satellite to distinguish between two neighbouring wavelengths. Therefore, it depends on the number of spectral bands of the image. Depending on the number of spectral bands, three types of images can be defined: panchromatic (one black-and-white band), multi-spectral images (approximately 3 to 7 bands), and hyper-spectral images (over 100 bands). The higher the spectral resolution is, the higher will be the precision of the spectral signature of an object and it is likely that it will be well discriminated. The temporal resolution is the ability of a satellite sensor to revisit the same area after certain period of time. The temporal resolution is one of the most important

characteristics of the satellites for remote sensing environmental applications. In fact, it is necessary to monitor Earth's surface changes or short time varying phenomena, caused mainly by human factors or natural evolution of vegetation. Furthermore, the number of acquisitions that could be exploited may be really small because of the bad weather conditions e.g. clouds or cloud shadows, which limit the view of the Earth's surface. Therefore, this can be the reason for not obtaining even one usable image over, for example, several months of acquisition time if the satellite has very low revisiting frequency.

In the past, many attempts to discriminate the tree species were based on the spatial resolution of the data. Tree species classification was performed using aerial photography [6] or high spatial resolution imagery [7], [8]. However, the results showed a limited success with potentially high confusion rates [9]. With only few spectral bands in the multispectral images accuracy of the species discrimination is diminished. Therefore, several studies explored the ability of hyperspectral imagery to identify the tree species. Using LiDAR data or a combination of multiple sources [10], [11], much higher accuracies were obtained, but due to the limited availability and high cost of the hyperspectral imagery, the operational use of these data remains difficult. Thus, several studies have addressed the problem of tree species classification using satellite image time series (SITS). These studies were based on the assumption that phenological variations from the start of the growing season to the senescence should increase the spectral separability of the deciduous tree species [12], due to the dependence of vegetation multispectral reflectance on its phenological phases. However, most of the studies looking into SITS potential were based on Landsat time series composed of a limited number of acquisition dates, sometimes from different years, and with only partial coverage of the key phenological periods. Only few studies demonstrated the potential of dense SITS acquired through entire growing season, but all of them were based on airborne images [13], [14].

Currently, it is possible to obtain large dataset with very high temporal resolution using MODIS. However, the spatial resolution of these images is relatively small: 250m (and even 500m or 1km), which is not enough to discriminate different tree species. Until this year the only satellite that could provide SITS with high spatial resolution is FORMOSAT-2. Main issue with this source is the high cost, which is the main reason for relatively few studies using FORMOSAT-2 images. In the future, there is a potential to use Sentinel-2 images to obtain SITS with high spatial resolution. This satellite will provide a global coverage of the Earth's land surface with high spatial resolution optical imagery and high temporal resolution (every 10 days with one satellite and 5 days with 2 satellites) and it will be free of charge.

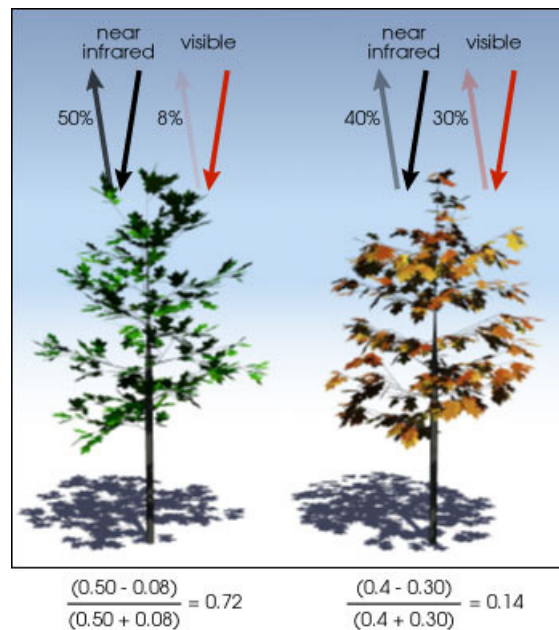
The red band (600 nm to 700 nm) and the near-infrared band (700 nm to 1100 nm) are the most commonly used to characterize the vegetation. From these bands, it is possible to create different indices of vegetation, or to estimate biophysical variables.

Although there are several vegetation indices, one of the most widely used is the Normalized Difference Vegetation Index (NDVI) [15]. Advantages of the NDVI for monitoring various phenology phases of the vegetation during and between the seasons is that NDVI is very well correlated with the photosynthetic activity and chlorophyll contents, can be easily computed, and it is mostly independent from soil type and current climate conditions. Calculation of the NDVI is based on two properties of the leaf cells of the green plants: 1) chlorophyll pigment found in these cells strongly absorbs in the red and blue part of the visible light, while

reflecting most of the green, and 2) leaf cell structures reflect around 50% of the near-infrared light (700-1100 nm) that hits them (Figure 1.1). Therefore, presence of the plants is indicated by the reflection of near-infrared light, while plant vitality, if there are any plants present, is indicated by visible light absorption.

NDVI is calculated from the visible (red) and near-infrared light reflected by vegetation using following equation:

$$NDVI = \frac{NIR - RED}{NIR + RED} \quad (1.1)$$



**Figure 1.1. Basic principle of NDVI. The healthy vegetation absorbs more Red light (left) than the unhealthy vegetation (right). Source: <http://earthobservatory.nasa.gov>.**

From the following NDVI values vary from -1 to 1. Since the reflection in near-infrared band for green vegetation is always higher than in the red band, NDVI values for the vegetation are always higher than 0. Bare ground and water have very low reflectance in the near-infrared band and thus their NDVI values are lower or equal to 0.1. NDVI values from the interval [0.2 – 0.5] indicate the presence of sparse vegetation of grassland and shrubs. NDVI values higher than 0.5 indicate the presence of green leaves (values close to 1 are related to the high density of green leaves) [16]. Finally, it should be noted that NDVI is mostly stable for the conifer species, while it can vary for the broadleaf species.

### 1.3 Working laboratory and study objectives

My internship was done within the Joint Research Unit DYNAFOR, that is attached to the French National Institute for Agricultural Research (INRA). This public institute was established in 1946. and it is under the joint authority of the *Ministries of Research and Agriculture*. INRA is composed of 13 departments in order to complete all tasks that are entrusted to it.



DYNAFOR is a JMU (Joint Research Unit) which was established in 2003. Its mission is mainly part of the 6<sup>th</sup> research area of the INRA center of Toulouse. This unit brings together researchers from two INRA departments: SAD (*Science pour l'Action et le Développement*) and EFPA (*Ecologie des Forêts, Prairies et milieux Aquatiques*) with teaching-researchers from ENSAT (*Ecole Nationale Supérieure Agronomique de Toulouse*) integrated into the National Polytechnic Institute of Toulouse, and teaching-researchers from the EI Purpan (*Ecole d'Ingénieurs de Purpan*). DYNAFOR activities are focused on the sustainable management of forest resources and rural areas as part of landscape ecology. The main objective of DYNAFOR is to understand and model the relationships between ecological processes, biotechnical processes and socio-economic processes in the management of renewable natural resources. It addresses the current issues in rural and forest areas, induced by global changes that are affected together by the climate, the land use, the biodiversity and the human activities. Finally, another purpose of this JMU is to develop ecological engineering of rural areas so that they could ensure their sustainability and capacity for providing the products and services that are expected by the company.

The organization of DYNAFOR is divided into three main areas:

- Area 1: ecosystem services in landscape,
- Area 2: data, space, geomatics and modelling,
- Area 3: biodiversity of rural forests and natural environments.

My work was mainly based on Area 2, with an objective to discriminate tree species from the multi-spectral satellite image time series (SITS) collected in the time period of four consecutive years (2011, 2012, 2013, 2014). In fact in this study the ability of mapping forest species using a dense high resolution multispectral Formosat-2 image time-series was explored. A couple of supervised classification methods were performed and compared in order to find the most convenient method for tree species discrimination. Feature selection algorithm was used to extract the most important features from SITS which represent the best dates for the tree species discrimination. Finally, the thematic maps were produced over the study area for each of the processing years. Thematic maps from different years were then compared to find the most accurate classified species and to assess the robustness of the classifications. This work was initiated by work [17]. It was assumed that phenological variations increase the spectral separability of the deciduous tree species. However, several parts from [17] needed to be improved. Thus, the main tasks of my work were to:

- I. Apply smoothing filter to SITS in order to reconstruct pixels contaminated with clouds and cloud shadows in the time series data. In the previous work [17], images that were affected by clouds or cloud shadows were not used for SITS creation. However, since the tree species discrimination was based on the assumption that phenological variations increase the spectral separability of the deciduous tree species it was very important to use all of the available images. Some of the images, affected by bad weather conditions and thus not used in the previous work [17], were acquired during key growing and senescing periods of the year, which may have limited discrimination potential of the study [17]. Therefore, it was expected that adding these images, after smoothing filter application, could significantly improve the identification of the most informative dates for the image acquisition.

- II. Perform three supervised classification methods on smoothed SITS created for four years. Instead of using only NDVI indices as the spectral features, and in order to see if it is possible to get a better solution, classification was performed on SITS created from all of the spectral bands and from both NDVI indices and spectral bands together.
- III. Implement a new algorithm for feature selection, the Sequential Floating Forward Selection (SFFS), in order to improve the standard feature selection algorithm which was used in the study [17], the Sequential Forward Selection (SFS) algorithm.
- IV. Produce thematic maps for each year and compare the results in order to draw conclusions about the robustness of the applied classification methods and to find the spatial distribution of the tree species, which were the most frequently assigned to the same class for the consecutive years.

During the internship period, I had interactions with a Master student Marc Lang and a PhD student Mailys Lopez, who worked in the same laboratory on the grassland classification using the same SITS.

## 2 Theoretical background

This chapter presents a detailed description of the algorithms used for the purposes of this work.

### 2.1 Background of smoothing filter

High reliability of the analysed image time series is required for all applications, i.e. the image time series has to be closely related to the observed land surfaces [18]. However, this is not often the case since optical image time series are usually affected by clouds and cloud shadows which add noise to the recorded signal [19]. Therefore, the SITS needs to be smoothed in order to fill data gaps which appeared due to this noise.

In this study, the smoothing of the image time series was performed by Whittaker smoother algorithm [20]. This algorithm is chosen because, compared to other smoothing filters, e.g. [21], it gives continuous control over the smoothness, adapts automatically to the boundaries, deals well with missing values by introducing the vector containing 0 and 1 weights and gives automatic choice for the smoothing parameter due to the fast cross-validation.

Basic Whittaker smoother algorithm, which assumes equally spaced data, is described in Section 2.1.1. Section 2.1.2 describes how basic algorithm can be modified to be applicable on unequally spaced data. The algorithm for selecting the optimal regularization parameter is presented in Section 2.1.3.

#### 2.1.1 Whittaker smoother for equally spaced data

The following notations are used in Section 2.1. Each recorded pixel time series is denoted as  $N$ -dimensional vector  $\mathbf{z} \in R^N$ , where  $N$  is the number of the time samples:

$$\mathbf{z} = \begin{bmatrix} z(t_1) \\ \vdots \\ z(t_N) \end{bmatrix} \quad (2.1.1)$$

Suppose this pixel is contaminated with noise. We can represent each noisy pixel sample as  $z(t_i) = x(t_i) + b(t_i)$ , where  $i \in [1, \dots, N]$ ,  $x(t_i)$  is a pixel sample we want to retrieve from the noisy sample  $z(t_i)$  and  $b$  is a white noise. Therefore our goal is to find the smooth pixel vector  $\mathbf{x}$ :

$$\mathbf{x} = \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_N) \end{bmatrix} \quad (2.1.2)$$

## The basic Whittaker smoother algorithm

To describe the basics of the Whittaker smoother we need to assume equally spaced data.

Whittaker smoother is based on penalized least squares method with basic principle that smoothing of the noisy/incomplete time series is a compromise between 1) fidelity to the data and 2) roughness of the reconstruction. Whittaker smoother finds the smooth series  $\mathbf{x}$  that minimizes a function combining these two conflicting goals.

The measure of the roughness  $R_d$  can be expressed as a squared sum of the differences  $\Delta^d x_i$ , where  $\Delta^d x_i$  represents a  $d$  order difference of  $x_i$ :

$$R_d = \sum_{i=1}^N (\Delta^d x_i)^2 \quad (2.1.3)$$

The 1<sup>st</sup> order difference is:

$$\Delta x_i = x_i - x_{i-1} \quad (2.1.4)$$

General expression for  $d^{\text{th}}$  order difference is:

$$\Delta^d x_i = \Delta(\Delta^{d-1} x_i). \quad (2.1.5)$$

For example the 2<sup>nd</sup> and 3<sup>rd</sup> order differences are:

$$\Delta^2 x_i = \Delta(\Delta x_i) = (x_i - x_{i-1}) - (x_{i-1} - x_{i-2}) = x_i - 2x_{i-1} + x_{i-2} \quad (2.1.6)$$

$$\Delta^3 x_i = \Delta(\Delta^2 x_i) = (x_i - 2x_{i-1} + x_{i-2}) - (x_{i-1} - 2x_{i-2} + x_{i-3}) = x_i - 3x_{i-1} + 3x_{i-2} - x_{i-3} \quad (2.1.7)$$

Deviation of the smooth pixel  $\mathbf{x}$  from the observed pixel  $\mathbf{z}$  can be expressed as the sum of the squared differences between observed samples  $z_i$  and smooth samples  $x_i$ :

$$S = \sum_{i=1}^N (z_i - x_i)^2 \quad (2.1.8)$$

The function which combines these two measures is:

$$Q = S + \lambda R_d \quad (2.1.9)$$

Parameter  $\lambda$  in equation (2.1.9) is a smoothing parameter that has to be defined by the user. With increasing parameter  $\lambda$ , influence of the roughness in  $\mathbf{x}$  will be stronger and the deviation of  $\mathbf{x}$  from the observation  $\mathbf{z}$  will also increase. Some examples showing this effect are given in *Appendix 4*. The aim of the penalized least squares is to find series  $\mathbf{x}$  that minimizes the final function  $Q$ .

To simplify the equations, the function  $Q$  will be expressed using matrices and vectors as:

$$Q = \|\mathbf{z} - \mathbf{x}\|^2 + \lambda \|\mathbf{D}_d \mathbf{x}\|^2 = (\mathbf{z} - \mathbf{x})^T (\mathbf{z} - \mathbf{x}) + \lambda \mathbf{x}^T \mathbf{D}_d^T \mathbf{D}_d \mathbf{x} \quad (2.1.10)$$

In equation (2.1.10)  $\|\mathbf{a}\|^2 = \sum_i a_i^2$  is a quadratic norm of any vector  $\mathbf{a}$  and  $\mathbf{D}_d$  is a matrix such that  $\mathbf{D}_d \mathbf{x} = \mathbf{\Delta}^d \mathbf{x}$ . E.g. for the first order difference and  $N=6$ , matrix  $\mathbf{D}_1$  is a  $(N-1) \times N$  given by:

$$\mathbf{D}_1 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (2.1.11)$$

Finding partial derivatives of the final function  $Q$  and equating them to 0, we get the solution for the pixel vector  $\mathbf{x}$ :

$$\begin{aligned} \frac{\partial Q}{\partial \mathbf{x}} &= -2(\mathbf{z} - \mathbf{x}) + 2\lambda \mathbf{D}_d^T \mathbf{D}_d \mathbf{x} = 0 \Rightarrow \\ \mathbf{x} &= (\mathbf{I} + \lambda \mathbf{D}_d^T \mathbf{D}_d)^{-1} \mathbf{z} \end{aligned} \quad (2.1.12)$$

In equation (2.1.12)  $\mathbf{I}$  is  $N \times N$  identity matrix.

### Dealing with missing data

The previous algorithm for smoothing can be easily modified to smooth the observations with missing values. In this case the vector of weights  $\mathbf{w}$  is introduced of the length equal to the length of  $\mathbf{z}$  (in our case  $N$ ). Vector  $\mathbf{w}$  can take values  $w_i=0$  for missing data elements and  $w_i=1$  for non-missing data elements. With missing values in the data, deviation of the smooth pixel  $\mathbf{x}$  from the observed pixel  $\mathbf{z}$  is changed to:

$$S = \sum_{i=1}^N w_i (z_i - x_i)^2 = (\mathbf{z} - \mathbf{x})^T \mathbf{W} (\mathbf{z} - \mathbf{x}) \quad (2.1.13)$$

Where  $\mathbf{W}$  is  $N \times N$  matrix with vector  $\mathbf{w}$  on its diagonal.

The measure of the roughness  $R_d$  is calculated in the same way as in equation (2.1.3).

The final function  $Q$  changes to:

$$Q = (\mathbf{z} - \mathbf{x})^T \mathbf{W} (\mathbf{z} - \mathbf{x}) + \lambda \mathbf{x}^T \mathbf{D}_d^T \mathbf{D}_d \mathbf{x} \quad (2.1.14)$$

In the same way as in equation (2.1.12) a solution for the smooth pixel  $\mathbf{x}$  is calculated as:

$$\begin{aligned}\frac{\partial Q}{\partial \mathbf{x}} &= -2\mathbf{W}(\mathbf{z} - \mathbf{x}) + 2\lambda\mathbf{D}_d^T \mathbf{D}_d \mathbf{x} = 0 \Rightarrow \\ \mathbf{x} &= (\mathbf{W} + \lambda\mathbf{D}_d^T \mathbf{D}_d)^{-1} \mathbf{W}\mathbf{z}\end{aligned}\quad (2.1.15)$$

## 2.1.2 Whittaker smoother for unequally spaced data

It is often necessary to adapt basic Whittaker smoother algorithm to the algorithm for smoothing unequally spaced data since the period between two consecutive image acquisitions is usually not constant.

Deviation of the smooth pixel from the observed pixel ( $S$ ) is estimated as for the equally spaced data with missing values (equation 2.1.13).

The measure of the roughness of  $\mathbf{x}$ ,  $R_d$ , is given as:

$$R_d = \sum_{i=1}^N (\Delta^d x(t_i))^2 = \|\mathbf{D}_d \mathbf{x}\|^2 = \mathbf{x}^T \mathbf{D}_d^T \mathbf{D}_d \mathbf{x} \quad (2.1.16)$$

The equation (2.1.16) looks the same as the equation (2.1.3) but for unequally spaced data the difference will change and therefore the measure of the roughness will also change e.g. for  $d=1$ , the difference and the measure of the roughness for unequally spaced data will be:

$$\Delta x(t_i) = \frac{x(t_i) - x(t_{i-1})}{t_i - t_{i-1}} \Rightarrow R_1 = \sum_{i=2}^N \left[ \frac{x(t_i) - x(t_{i-1})}{t_i - t_{i-1}} \right]^2 \quad (2.1.17)$$

Thus, matrix  $\mathbf{D}_1$  is given as:

$$\mathbf{D}_1 = \begin{pmatrix} \frac{1}{\delta_2} & \frac{1}{\delta_2} & 0 & \dots & 0 \\ 0 & \frac{1}{\delta_3} & \frac{1}{\delta_3} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \frac{1}{\delta_N} & \frac{1}{\delta_N} \end{pmatrix} \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{\delta_2} & \frac{1}{\delta_2} & 0 & \dots & 0 \\ 0 & -\frac{1}{\delta_3} & \frac{1}{\delta_3} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\frac{1}{\delta_N} & \frac{1}{\delta_N} \end{pmatrix} \quad (2.1.18)$$

where  $\delta_i = t_i - t_{i-1}$ . Size of the matrix  $\mathbf{D}_1$  is  $(N-1) \times N$ .

The second order difference is:

$$\Delta^2 x(t_i) = \frac{\Delta x(t_i) - \Delta x(t_{i-1})}{t_i - t_{i-2}} \quad (2.1.19)$$

Thus, the matrix  $\mathbf{D}_2$  can be written as:

$$\mathbf{D}_2 = \begin{pmatrix} \frac{1}{(\delta_2 t_3)(\delta_2)} & \frac{-1}{(\delta_2)(\delta_2)} & \frac{1}{(\delta_2 t_3)(\delta_2)} & 0 & \dots & 0 \\ 0 & \frac{1}{(\delta_2 t_4)(\delta_2)} & \frac{-1}{(\delta_2)(\delta_2)} & \frac{1}{(\delta_2 t_4)(\delta_2)} & \dots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \dots & 0 & \frac{1}{(\delta_2 t_N)(\delta_2)} & \frac{-1}{(\delta_2)(\delta_2)} & \frac{1}{(\delta_2 t_N)(\delta_2)} \end{pmatrix} = \mathbf{\Lambda}_2 \mathbf{D}_1 \quad (2.1.20)$$

where  $\delta_2 t_i = t_i - t_{i-2}$  and  $\mathbf{\Lambda}_2$  is:

$$\mathbf{\Lambda}_2 = \begin{pmatrix} \frac{-1}{\delta_2 t_3} & \frac{1}{\delta_2 t_3} & 0 & \dots & 0 \\ 0 & \frac{-1}{\delta_2 t_4} & \frac{1}{\delta_2 t_4} & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & \frac{-1}{\delta_2 t_N} & \frac{1}{\delta_2 t_N} \end{pmatrix} \quad (2.1.21)$$

And finally, a  $d^{\text{th}}$  order difference is:

$$\Delta^d x(t_i) = \frac{\Delta^{d-1} x(t_i) - \Delta^{d-1} x(t_{i-1})}{t_i - t_{i-d}} \quad (2.1.22)$$

To find matrix  $\mathbf{D}_d$  the following recursion is used:

$$\mathbf{D}_d = \mathbf{\Lambda}_d \mathbf{D}_{d-1} \quad (2.1.23)$$

with

$$\mathbf{\Lambda}_d = \begin{pmatrix} \frac{-1}{\delta_d t_{d+1}} & \frac{1}{\delta_d t_{d+1}} & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & \frac{-1}{\delta_d t_N} & \frac{1}{\delta_d t_N} \end{pmatrix} \quad (2.1.24)$$

where  $\delta_d t_i = t_i - t_{i-d}$ .

Thus, the derivative matrix of order  $d$  is obtained using the recursive formula:

$$\mathbf{D}_d = \mathbf{\Lambda}_d \mathbf{\Lambda}_{d-1} \dots \mathbf{\Lambda}_2 \mathbf{D}_1 \quad (2.1.25)$$

Replacing deviation of the smooth from the observed pixel and roughness in the function Q and finding partial derivatives (see equation 2.1.12) a solution for the smooth pixel  $\mathbf{x}$  is obtained as:

$$\mathbf{x} = (\mathbf{W} + \lambda \mathbf{D}_d^T \mathbf{D}_d)^{-1} \mathbf{W} \mathbf{z} \quad (2.1.26)$$

### 2.1.3 Choosing a value for smoothing parameter $\lambda$

Value for the smoothing parameter  $\lambda$  can be iteratively chosen until we obtain a visually satisfactory result. However, more objective and automatic choice for  $\lambda$  can be made using cross-validation [20].

The procedure is done by leaving out one of the non-missing elements of the pixel time series  $z_i$  for  $i \in \{1, \dots, N\}$ , then applying Whittaker smoother to the remaining elements and predicting the left out element. Cross-validation error is found by doing this procedure for all non-missing elements of the pixel  $\mathbf{z}$ . Regularization parameter is chosen so that prediction is as good as possible, i.e. error of the prediction is minimum.

#### Ordinary Cross-Validation

The Ordinary Cross-Validation mean square error is defined as:

$$OCV(\lambda) = \frac{1}{\sum_i w_i} \sum_{i=1}^N (z(t_i) - x(t_i)^{-i})^2 w_i \quad (2.1.27)$$

In equation (2.1.27),  $x(t_i)^{-i}$  is the estimation of the value  $z(t_i)$  after removing the  $i^{th}$  element of  $\mathbf{z}$ .

For the chosen interval for  $\lambda$ , the Ordinary Cross-Validation Estimate of  $\lambda$  is:

$$\lambda_{OCV} = \arg \min_{\lambda \in R^+} OCV(\lambda) \quad (2.1.28)$$

Using the equation (2.1.26) and introducing the hat matrix  $\mathbf{H}$  follows:

$$\begin{aligned} \mathbf{H} &= (\mathbf{W} + \lambda \mathbf{D}_d^T \mathbf{D}_d)^{-1} \mathbf{W} \Rightarrow \\ \mathbf{x} &= \mathbf{H} \mathbf{z} \end{aligned} \quad (2.1.29)$$

Each of the elements of the vector  $\mathbf{x}$  is calculated as:



$$x(t_i) = \sum_{j=1}^N h_{ij} z(t_j), \text{ for } i \in \{1, \dots, N\} \quad (2.1.30)$$

After removing the  $i^{\text{th}}$  element of the vector  $\mathbf{z}$ , estimation  $x(t_i)^{-i}$  is:

$$\begin{aligned} x(t_i)^{-i} &= \sum_{j=1}^N h_{ij} x(t_j) \text{ where } x(t_j) = \begin{cases} z(t_j), & \text{if } j \neq i \\ x(t_i)^{-i}, & \text{if } j = i \end{cases} \Rightarrow \\ x(t_i) - x(t_i)^{-i} &= \sum_{j=1}^N h_{ij} z(t_j) - \sum_{\substack{j=1 \\ j \neq i}}^N h_{ij} z(t_j) - h_{ii} x(t_i)^{-i} \Rightarrow \\ x(t_i) - x(t_i)^{-i} &= h_{ii} z(t_i) - h_{ii} x(t_i)^{-i} \Rightarrow \\ x(t_i)^{-i} &= \frac{x(t_i) - h_{ii} z(t_i)}{1 - h_{ii}} \end{aligned} \quad (2.1.31)$$

Replacing  $x(t_i)^{-i}$  in equation (2.1.27), OCV of the parameter  $\lambda$  is:

$$OCV(\lambda) = \frac{1}{\sum_i w_i} \sum_{i=1}^N \left( \frac{z(t_i) - x(t_i)}{1 - h_{ii}} \right)^2 w_i \quad (2.1.32)$$

### Generalized Cross-Validation

To compute Generalized Cross-Validation error,  $h_{ii}$  is replaced by the mean of the diagonal elements of the matrix  $H$ :

$$GCV(\lambda) = \frac{1}{\sum_i w_i} \frac{\sum_{i=1}^N [z(t_i) - x(t_i)]^2 w_i}{\left[ \frac{1}{\sum_i w_i} \text{trace}(I - H) \right]^2} = \frac{1}{\sum_i w_i} \frac{\sum_{i=1}^N [z(t_i) - x(t_i)]^2 w_i}{\left[ 1 - \frac{\text{trace}(H)}{\sum_i w_i} \right]^2} \quad (2.1.33)$$

As for Ordinary Cross-Validation, for chosen interval of the parameter  $\lambda$ , the Generalized Cross-Validation Estimate of  $\lambda$  is selected as:

$$\lambda_{GCV} = \arg \min_{\lambda \in R^+} GCV(\lambda) \quad (2.1.34)$$

## 2.2 Background of feature selection

Image classification is used for producing the thematic maps which provide an informative description of the study area, i.e. the thematic maps show spatial distribution of the tree species inside area of interest. Furthermore, classification was used in this project for selecting the best features, which in this case represent the most useful images to discriminate tree species and better define the image acquisition plan.

In this study, discrimination of the tree species from the multi-temporal images was performed with three supervised learning methods: Support Vector Machines (SVM) [22, Chapter 12], Random Forest (RF) [22, Chapter 15] and Gaussian Mixture Model (GMM) classifier. Nonlinear Parsimonious Feature Selection (NPFS) package was used to learn GMM model. NPFS was presented in [23] and represents feature selection algorithm which is used in this study for identifying the most important dates for discriminating the tree species.

The basics of the NPFS algorithm are described in Section 2.2.1. Section 2.2.2 presents the Sequential Floating Forward Selection (SFFS) algorithm as the replacement for the standard feature selection algorithm in the NPFS.

### 2.2.1 Non Linear Parsimonious Feature Selection (NPFS)

Nonlinear Parsimonious Feature Selection (NPFS), compiles a pool of selected features by iteratively selecting a spectral feature from the original set of features. This pool is used to learn a Gaussian Mixture Model (GMM). Successive features will be selected according to their classification rate, until the stopping criterion is reached. The estimation of the classification rate is done using k-fold Cross-Validation (k-CV).

Fast GMM parameterization when k-CV is computed is crucial for the efficient implementation of the NPFS. From the following, it is possible to quickly perform k-CV and forward selection by using parameter update rules and the marginalization properties of the Gaussian distribution.

#### Gaussian Mixture Model

Let  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$  be a set of training pixels, where  $\mathbf{x}_i$  is a D-dimensional pixel vector,  $\mathbf{x}_i \in R^D$  and  $y_i \in \{1, \dots, C\}$  its class. C is the number of classes, n is the number of training pixels and  $n_c$  is the number of training pixels belonging to the class c.

A Gaussian mixture model is a probabilistic model, which assumes that each pixel vector is generated from a mixture of a finite number of the Gaussian distributions as:

$$p(\mathbf{x}) = \sum_{c=1}^C \pi_c p(\mathbf{x}/c) \quad (2.2.1)$$

In equation (2.2.1)  $\pi_c$  is a prior probability for each class ( $0 \leq \pi_c \leq 1$  and  $\sum_{c=1}^C \pi_c = 1$ ) and  $p(\mathbf{x}/c)$  is a prior class conditional probability function for pixel vector  $\mathbf{x}$  given as a D-dimensional Gaussian distribution:

$$p(\mathbf{x}/c) = \frac{1}{(2\pi)^{D/2} |\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^T \Sigma_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right) \quad (2.2.2)$$

where  $\boldsymbol{\mu}_c$  is the mean vector of the class  $c$  and,  $\Sigma_c$  and  $|\Sigma_c|$  are the covariance matrix of the class  $c$  and its determinant.

According to Bayes rule, the posterior probability of the class  $c$  when given the pixel vector  $\mathbf{x}$  is:

$$p(c/\mathbf{x}) = \frac{\pi_c p(\mathbf{x}/c)}{p(\mathbf{x})} \quad (2.2.3)$$

Using the maximum *a posteriori* rule, pixel vector is classified to the class  $c$  if  $p(c/\mathbf{x}) \geq p(j/\mathbf{x}), \forall j \in [1, \dots, C]$ .

Since in equation (2.2.3)  $p(\mathbf{x})$  is a constant which does not affect the final result, the maximum a posteriori rule is given by:

$$\text{Assign } \mathbf{x} \text{ to class } c \text{ if } c = \arg \max_{j=1, \dots, C} \pi_j p(\mathbf{x}/j) \quad (2.2.4)$$

After replacing the eq. (2.2.2) in eq. (2.2.4) and by taking the log of eq. (2.2.4) the final decision rule is given as:

$$Q_c(\mathbf{x}) = -(\mathbf{x}-\boldsymbol{\mu}_c)^T \Sigma_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) - \ln(|\Sigma_c|) + 2 \ln(\pi_c) \quad (2.2.5)$$

The estimators of the model parameters are obtained using standard maximization of the log-likelihood as:

$$\begin{aligned} \hat{\pi}_c &= \frac{n_c}{n}, \\ \hat{\boldsymbol{\mu}}_c &= \frac{1}{n_c} \sum_{i=1}^{n_c} \mathbf{x}_i, \\ \hat{\Sigma}_c &= \frac{1}{n_c} \sum_{i=1}^{n_c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T \end{aligned} \quad (2.2.6)$$

## The Sequential Forward Feature Selection (SFS)

The main goal of feature selection is to select a subset of  $k$  features from the given set of  $D$  measurements,  $k < D$ , without significantly degrading the performance of the recognition system [24]. Feature selection techniques usually need a criterion for evaluating a performance of the model and an optimization procedure for finding the subset of features that maximizes/minimizes the criterion [25].

Let  $X_k = \{x_i : 1 \leq i \leq k, x_i \in Y\}$  be the set of  $k$  features selected from the complete set of measurements  $Y = \{y_i : 1 \leq i \leq D\}$ , where  $D$  is the number of available features.

The basic Sequential Forward Selection (SFS) method can be split in several steps [22, Chapter 3] [26]:

- Start with an empty set of selected features,  $X_k = \emptyset$
- Iteratively add the most significant feature with respect to  $X_k$  from the set of available features  $Y - X_k$
- Stop if the increase of the estimated classification rate is too low or if the maximum number of the features is reached

Classification rate was estimated with stratified  $k$ -fold cross-validation. The set of training pixels  $S$  was divided into  $k$  subsets of equal size. Each subset contains approximately the same percentage of the pixels which belong to the same class as the initial set  $S$ . One of the  $k$ -subsets was used for model testing, while remaining subsets were used as a training data. Since each of the  $k$  subsets needs to be used exactly once for validation, this procedure was repeated exactly  $k$  times. The  $k$  test errors were computed and then averaged to compute the mean test error.

## Fast estimation of the GMM sub-models parameters during the Cross-validation process

Suppose the number of the pixels used for validation during the Cross Validation procedure is  $v$ . Parameters of the model  $S^{n-v}$  can be estimated from the full learned GMM model using update rules. Classification rate can be estimated with subset of features using marginalization properties of the Gaussian distribution parameters. Therefore, GMM model needs to be learned only once during the entire training step.

The update rules after removing  $v$  samples for validation:

Rule 1 (Class proportion):

$$\hat{\pi}_c^{n-v} = \frac{n \hat{\pi}_c - v_c}{n - v} \quad (2.2.7)$$

In equation (2.2.7)  $\hat{\pi}_c^n$  and  $\hat{\pi}_c^{n-v}$  are the proportions of the class  $c$  computed over  $n$  and  $(n-v)$  samples, respectively and  $v_c$  is the number of the removed pixels which belong to the class

$c$ . Note that  $\sum_{c=1}^C v_c = v$ .

Rule 2 (Mean vector):

$$\hat{\boldsymbol{\mu}}_c^{n_c - v_c} = \frac{n_c \hat{\boldsymbol{\mu}}_c^{n_c} - v_c \hat{\boldsymbol{\mu}}_c^{v_c}}{n_c - v_c} \quad (2.2.8)$$

In equation (2.2.8)  $\hat{\boldsymbol{\mu}}_c^{n_c}$  and  $\hat{\boldsymbol{\mu}}_c^{n_c - v_c}$  are the mean vectors of the class  $c$  computed over  $n_c$  and  $n_c - v_c$  training samples and  $\hat{\boldsymbol{\mu}}_c^{v_c}$  is the mean vector of the class  $c$  computed over  $v_c$  removed samples.

Rule 3 (Covariance matrix):

$$\boldsymbol{\Sigma}_c^{n_c - v_c} = \frac{n_c}{(n_c - v_c)} \boldsymbol{\Sigma}_c^{n_c} - \frac{v_c}{(n_c - v_c)} \boldsymbol{\Sigma}_c^{v_c} - \frac{n_c v_c}{(n_c - v_c)^2} (\hat{\boldsymbol{\mu}}_c^{n_c} - \hat{\boldsymbol{\mu}}_c^{v_c})(\hat{\boldsymbol{\mu}}_c^{n_c} - \hat{\boldsymbol{\mu}}_c^{v_c})^T \quad (2.2.9)$$

In equation (2.2.9),  $\boldsymbol{\Sigma}_c^{n_c}$  and  $\boldsymbol{\Sigma}_c^{n_c - v_c}$  are the covariance matrices of the class  $c$  computed over  $n_c$  and  $n_c - v_c$  training samples, respectively.

### Marginalization of Gaussian distribution

To get the GMM model over a subset of the original feature set, it is only necessary to exclude non-selected features from the mean vector and the covariance matrix [27].

If initial set of features is represented as:  $\mathbf{x} = [\mathbf{x}_s, \mathbf{x}_{ns}]$ , where  $\mathbf{x}_s$  and  $\mathbf{x}_{ns}$  are selected and non-selected feature sets, respectively, then mean vector and covariance matrix computed over the full model can be written as:

$$\hat{\boldsymbol{\mu}} = [\boldsymbol{\mu}_s, \boldsymbol{\mu}_{ns}] \quad (2.2.10)$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{s,s} & \boldsymbol{\Sigma}_{s,ns} \\ \boldsymbol{\Sigma}_{ns,s} & \boldsymbol{\Sigma}_{ns,ns} \end{bmatrix} \quad (2.2.11)$$

From the marginalization of Gaussian distribution follows that  $\mathbf{x}_s$  is also a Gaussian distribution with mean vector  $\boldsymbol{\mu}_s$  and covariance matrix  $\boldsymbol{\Sigma}_{s,s}$ . Therefore, when full model is learned, all of the sub-models built with a subset of the original variables will be available at no additional computational cost.

## 2.2.2 The Sequential Forward Floating Selection algorithm (SFFS)

Sequential forward selection algorithm (SFS), described in Section 2.2.1, suffers from the so-called "nesting effect", which means that features once selected can not be excluded later from the pool of the selected features. This can lead to the sub-optimal subset of the chosen features. Since there is rather widely accepted belief [24, 29, 30, 31] that floating search methods (Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS)) [24] are superior to the simple sequential ones, SFS and SBF [28], SFFS algorithm [24][32, Chapter 9] is implemented in this study as a modification of the original SFS algorithm in order to avoid its "nesting effect".

If the  $k$  features have already been selected and form the subset  $X_k$  with its criterion function  $J(X_k)$ , the values of the criterion function  $J(X_i)$  have to be stored for all the previously subsets of size  $i = 1, 2, \dots, (k-1)$ . As in the SFS algorithm, in this project,  $J$  represents the estimated classification rate.

The SFFS algorithm also starts from the empty set of features ( $k=0$  and  $X_0=\emptyset$ ). For the selection of the first two features original SFS method is applied. Then the algorithm continues with the step 1.

- **Step 1: Inclusion.** Using the basic SFS method, select the most significant feature  $x_{k+1}$  with respect to  $X_k$  from the set of available measurements,  $Y - X_k$ , and add it to the subset  $X_k$ . The most significant feature is obtained as:

$$x_{k+1} = \arg \max_{x \in (Y - X_k)} [J(X_k + x)] \quad (2.2.12)$$

Thus the new formed feature set is  $X_{k+1} = X_k + x_{k+1}$ .

- **Step 2: Conditional exclusion.**  
Find the least significant feature in the set  $X_{k+1}$  as:

$$x_r = \arg \max_{x \in X_{k+1}} [J(X_{k+1} - x)] \quad (2.2.13)$$

1. If the least significant feature is the one just added in the first step,  $x_r = x_{k+1}$ , keep it, set the number of selected features to  $k=k+1$  and return to step 1.
  2. If  $x_r$  is the least significant feature and  $1 \leq r \leq k$ , exclude it from the set  $X_{k+1}$  to form the new subset  $X'_k = X_{k+1} - x_r$ . Note that now  $J(X'_k) > J(X_k)$ . If  $k=2$  set  $X_k = X'_k$ ,  $J(X_k) = J(X'_k)$  and return to step 1, otherwise continue with the step 3.
- **Step 3: Continuation of conditional exclusion.**  
In the same way as in the Step 2, find the least significant feature  $x_s$  in the set  $X'_k$ .
    1. If  $J(X'_k - x_s) \leq J(X_{k-1})$  set  $X_k = X'_k$ ,  $J(X_k) = J(X'_k)$  and return to Step 1.
    2. If  $J(X'_k - x_s) > J(X_{k-1})$  then exclude  $x_s$  from  $X'_k$  to form new set  $X'_{k-1} = X'_k - x_s$ . Set  $k=k-1$ . If  $k=2$  set  $X_k = X'_{k-1}$ ,  $J(X_k) = J(X'_{k-1})$  and return to step 1, otherwise repeat Step 3.

- The algorithm stops when the maximum number of features is selected.

## 3 Study area and data collection

### 3.1 Study area

The study area is located in southwest of France, in Midi-Pyrénées region, about 30 km west of Toulouse (Figure 3.1). Climate of the area is characterized by mild and rainy winters and dry and hot summers, i.e. Cfb climate type as classified by Köppen-Geiger climate classification system [33]. Annual mean air temperature is higher than 13°C while mean annual precipitation is 656 mm. Forests are found on 10 % of the area, while the most prevalent tree species is Oak (*Quercus spp.*). Non-forest part of the area consists of grasslands and crops (combination of wheat, sunflower and maize).

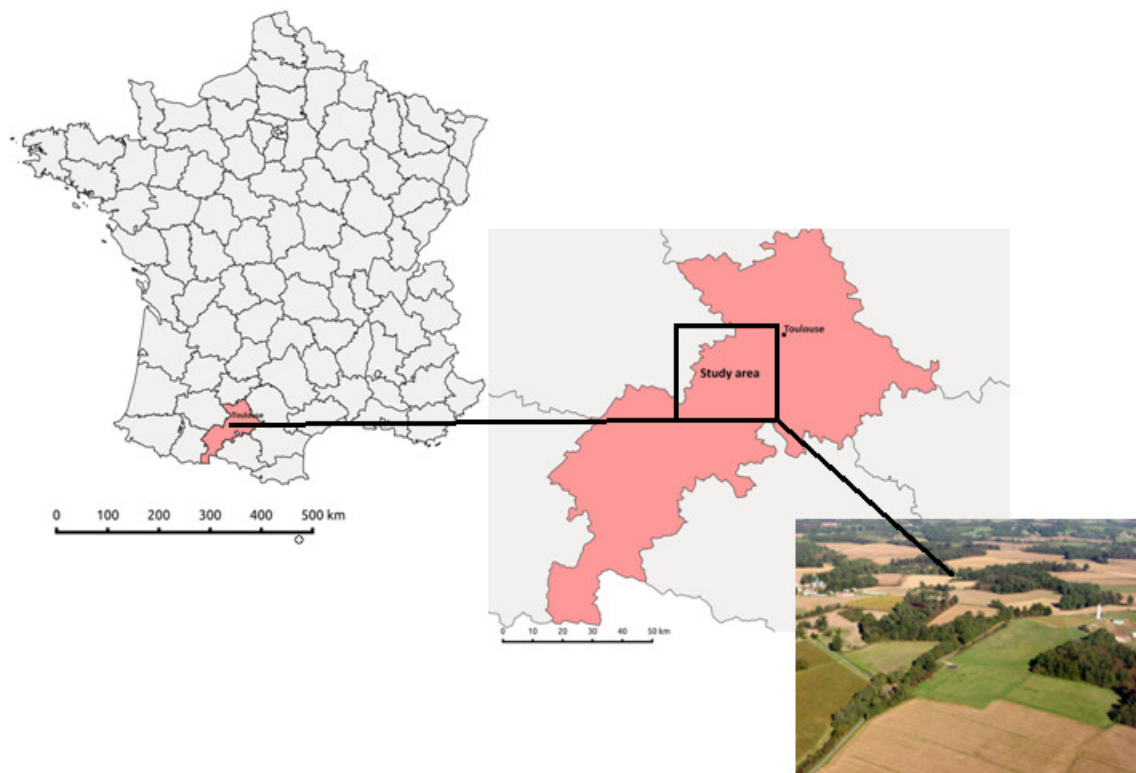


Figure 3.1 Map of France with highlighted study area and typical landscape for study area

### 3.2 Field data collection

Since supervised classification methods were used for tree species discrimination, it was necessary to use ground truth data, which was collected by the DYNAFOR lab. during three surveys in November 2013, January 2014 and May 2014. In total, 1038 sample points of the dominant broadleaf and conifer tree species were collected from the study area. Two observers were used to record each plot covering approximately an area of 576 m<sup>2</sup>, which is equivalent to nine contiguous FORMOSAT-2 pixels of 8m x 8m. Each plot was homogeneous in terms of tree species. GPS coordinates of plots were estimated with Garmin GPSMap 62st receiver and all the plots were distributed over the whole study area. Plots were later converted to polygons of one pixel size to use them in the classification (Figure 3.2).



Three thematic levels based on the Forest National Inventory [34] were defined from the collected ground truth data (Table 3.1). The first one classifies the forest into broadleaf and coniferous species (level 1). The second level splits level 1 forest groups into two sub-categories. Broadleaf forests are for the most part deciduous, except for the evergreen Eucalyptus, while conifer forests can be pine forest or other. Finally, third level includes all the fourteen tree species. Sample size per species varied from 35 pixels for Black locust and Douglas fir to 209 pixels for Aspen.

Level 1	Level 2	Level 3	Sample size
Broadleaf	Deciduous	Silver birch ( <i>Betula pendula</i> )	75
Broadleaf	Deciduous	Pedunculate/Pubescent/Sessile oak ( <i>Quercus robur/pubescens/petraea</i> )	125
Broadleaf	Deciduous	Red oak ( <i>Quercus rubra</i> )	100
Broadleaf	Deciduous	European ash ( <i>Fraxinus excelsior</i> )	40
Broadleaf	Deciduous	Aspen ( <i>Populus tremula</i> )	209
Broadleaf	Deciduous	Black locust ( <i>Robinia pseudoacacia</i> )	35
Broadleaf	Deciduous	Willow ( <i>Salix</i> )	39
Broadleaf	Evergreen	Eucalyptus ( <i>Eucalyptus</i> )	100
Conifer	Pine	Corsican pine ( <i>Pinus nigra subsp. laricio</i> )	40
Conifer	Pine	Maritime pine ( <i>Pinus pinaster</i> )	40
Conifer	Pine	Black pine ( <i>Pinus nigra subsp. salzmannii</i> )	40
Conifer	Pine	Austrian black pine ( <i>Pinus nigra var. austriaca</i> )	85
Conifer	Other conifer	Douglas fir ( <i>Pseudotsuga menziesii</i> )	35
Conifer	Other conifer	Silver fir ( <i>Abies alba</i> )	75

Table 3.1. The number of the reference pixels for 14 tree species analyzed in the study



Figure 3.2. Reference pixels distributed over the study area. Aerial photograph (left) and zoomed part of the reference pixels (right).

### 3.3 Remote sensing data

This study uses images acquired by FORMOSAT-2, Taiwanese high resolution satellite, as a source of the remote sensing data. The Remote Sensing Instrument (RSI) onboard FORMOSAT-2 has multispectral sensors (sensitive to blue, green, red and near-infrared wavelengths) and a panchromatic sensor which is sensitive to all wavelengths from visible to near-infrared, processed as a gray-level image. Image spatial resolution is 2 m for panchromatic FORMOSAT images and 8 m for multispectral images. Scene coverage is 24x24 km. Table 3.2 lists spectral bands and wavelengths for each spectral band.

Band	Colour	Wavelength ( $\mu\text{m}$ )	Resolution
Band 1	Blue	0.45-0.52	8x8m
Band 2	Green	0.52-0.60	8x8m
Band 3	Red	0.63-0.69	8x8m
Band 4	Near-infrared	0.76-0.90	8x8m
Band 5	Panchromatic	0.45-0.90	2x2m

**Table 3.2. Characteristics of the Formosat-2 satellite sensor (Site Airbus Defence and Space)**

Due to their technical limitations, optical sensors do not capture the signals from objects located under clouds and cloud shadows. Thus, it was necessary to use masks to exclude samples which belong to the cloudy area (the corresponding values in the masks are higher than zero) (*Figure 3.3*).

The mask is defined by the numbers in bits where each bit is specified in the following way:

Bit 0 (1)	All clouds (except thin ones) or shadows
Bit 1 (2)	All clouds (except thin ones)
Bit 2 (4)	Cloud detected through absolute threshold
Bit 3 (8)	Cloud detected through multi-t threshold
Bit 4 (16)	Very thin clouds
Bit 5 (32)	High clouds detected with 1.38 $\mu\text{m}$ band (LANDSAT 8 only)
Bit 6 (64)	Cloud shadows matched with a cloud
Bit 7 (128)	Cloud shadows in the zone where clouds could be outside the image

**Table 3.3. Meaning of the bits in the mask of clouds and cloud shadows**

All the multi-spectral images used in the study, together with cloud masks for each date, were provided by CESBIO (Centres d'Etudes Spatiales de la Biosphère). *Table 3.4* shows the number and dates of the available images and masks for each year processed in the project. All the images were acquired with a constant viewing angle to reduce the within-species spectral variation. Operational pre-processing chain was used for orthorectification, atmospheric correction and cloud detection including cloud shadows and to obtain surface reflectance time-series data [35].



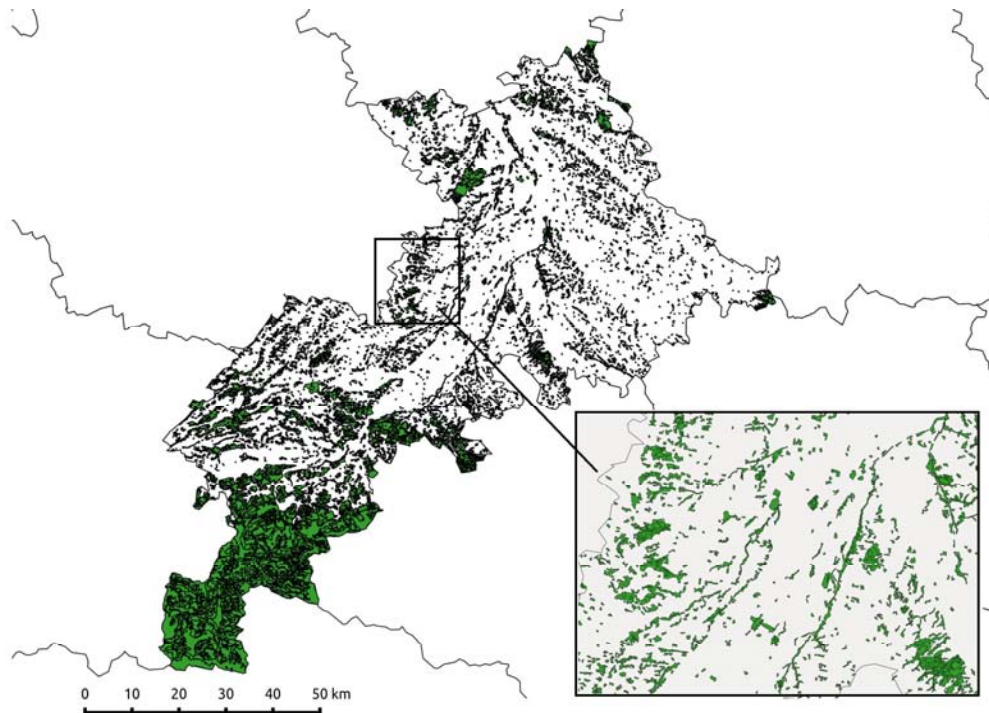
**Figure 3.3. Multispectral image and cloud mask acquired on 12/01/2012**

<b>Year 2011 (12 images)</b>	<b>Year 2012 (13 images)</b>	<b>Year 2013 (17 images)</b>	<b>Year 2014 (15 images)</b>
January 25	January 12	February 16	March 02
April 15	February 18	March 03	March 28
May 09	March 07	May 06	April 23
May 20	March 27	May 26	May 05
May 24	May 03	Jun 06	May 16
Jun 21	Jun 20	Jun 26	Jun 05
July 08	July 07	July 06	July 16
September 06	July 17	July 20	August 10
September 27	August 10	July 30	August 24
October 06	August 22	August 11	September 07
October 22	November 01	August 22	September 17
December 10	December 15	September 01	September 27
	December 31	September 21	October 23
		October 12	November 02
		October 27	November 19
		November 28	
		December 20	

**Table 3.4. Formosat-2 imagery, available dates and number of acquired images for each year**

### 3.4 Ancillary data

Ancillary map was used to mask non-forest areas of the images (*Figure 3.4*). This mask provides information about forest/non-forest areas within Haute-Garonne department with a minimum forest area size of 2.25 hectares. It was obtained from the French National Forest Inventory database (IGN BD Forêt®, v.1) produced in 1996.



**Figure 3.4.** Forest/non-forest map for the Haute-Garonne department and for the study area (zoomed part). Area covered by forest is presented in green colour.

## 4 Methodology

### 4.1 General procedure

Simplified method used in this study consists of two main parts: pre-processing and classification with accuracy assessment (*Figure 4.1*). Pre-processing part uses as inputs multi-spectral images from one year together with cloud masks for each image. Output of the pre-processing is Satellite Image Time Series, which is used together with the map containing reference pixels as an input to the second part of the project. The final result of the project consists of the classification accuracies and the thematic maps across the study area.

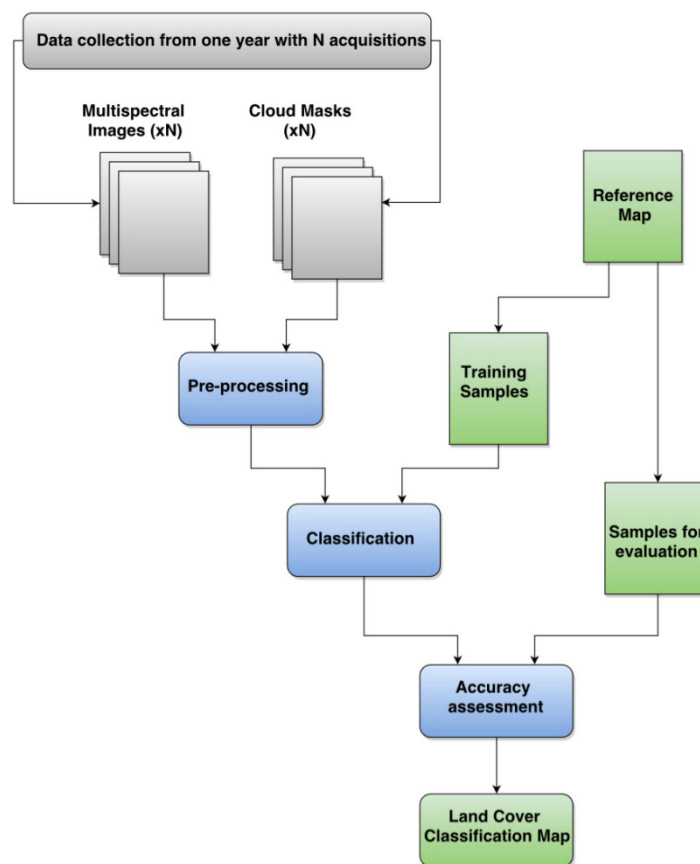


Figure 4.1 Simplified flow-chart of the project

### 4.2 Pre-processing

This part of the project had the aim to obtain Satellite Image Time Series (SITS) from the available images and prepare it for classification (*Figure 4.2*).

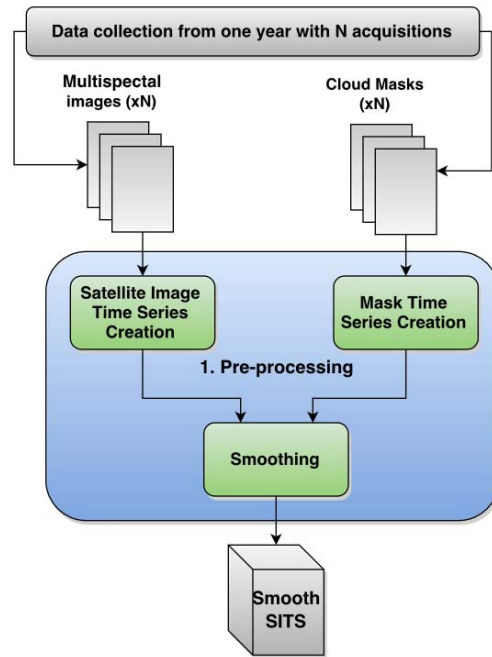


Figure 4.2. Simplified flow-chart of the pre-processing part of the project

Pre-processing part of the project was divided into two steps:

### Step 1. The Satellite Image Time Series creation

The SITS were created from the available multi-spectral images using OTB (*Orfeo Toolbox*) applications package and Shell Programming. Two steps were necessary for the SITS creation: the creation of the spectral features and concatenation. Spectral features were generated firstly from the NDVI indices, then from spectral bands and finally from both the NDVI indices and the spectral bands. Once the spectral features were generated for each multi-spectral image, all the spectral features were concatenated into one multi-temporal image, referred to as SITS. SITS were created for each of the four years separately, as well as for each of the four years from NDVI indices, from spectral bands and from spectral bands and NDVI indices together.

In this part of the project, masks containing clouds and cloud shadows were also concatenated into one multi-temporal mask, Mask Time Series. Same as the SITS, Mask Time Series were created for each of the four years.

An example of the *shell scripts* that were made for the purposes of this part of the project is given in *Appendix 2 (script\_sits.sh)*.

### Step 2. Smoothing

Smoothing function applies Whittaker smoother for unequally spaced data on each pixel in the SITS along temporal axis. Before executing smoothing function, it was necessary to build temporal axis which consisted of the days of the year for each acquisition (e.g. image acquired on 18/02/2012, corresponds to the 49<sup>th</sup> day of the year).



At the beginning of the smoothing function each pixel time sample from the SITS under clouds or cloud shadows was marked as the missing value. This was done by setting the values of the vector  $w$  to  $w_i = 0$  if the corresponding pixel time sample in the Mask Time Series is higher than zero and  $w_i = 1$  if it is equal to zero. The size of the vector  $w$  was different for each year and equal to the number of multi-spectral images (time acquisitions) for that year.

Order of the Whittaker smoother  $d$  was set to two, since this value showed to be a good balance between satisfactory smoothing results and computational complexity. Regularization parameter  $\lambda$  varied across the years. To find the optimal  $\lambda$  for each SITS pixel, belonging to the forest area, Generalized and Ordinary Cross-Validations were computed for  $\lambda = \{10^0, \dots, 10^{15}\}$ . Then, for each of these pixels, the optimal  $\lambda$  was chosen so that ordinary and generalized cross-validations had minimum values. Optimal  $\lambda$  for smoothing the whole SITS of one year was set to the value which appeared most frequently.

At the end of the pre-processing part of the project, the output image contains the filtered data where each pixel has the same number of samples.

Functions for generating the temporal axis (*create\_TimeStamp.py*), smoothing function (*smooth\_image.py*) and the program which finds the optimal regularization parameter (*find\_optimal\_lambda.py*), were implemented in Python and these codes are reported in *Appendix 2*.

### **4.3 Classification and accuracy assessment.**

This part of the general procedure was based on three steps:

#### **Step 1. Extracting the training/validation samples from the SITS generated in the pre-processing part of the project**

Reference Map was used in order to get reference pixels, one part of which was used for training the model and the rest for the model evaluation. Training samples were provided in vector format (.shp). In order to use it, the vector data had to be converted to raster data. After rasterization, raster image has the same dimension and size for the pixels as the images in SITS. In this image, each pixel which has non-zero value is a reference pixel, and its value corresponds to its class.

The OTB function was used for converting a vector data to a raster image. OTB function, alongside the function for extracting training/validation samples which was implemented in Python (*get\_samples\_from\_roi.py*), is presented in *Appendix 2*.

#### **Step 2. Learning the classifier and computing its performance (classification accuracy)**

In order to learn GMM classifier with NPFS algorithm, both the SFS and SFFS algorithms were used for feature selection. Stratified 5-fold CV was used to estimate the classification accuracy. Maximum number of features for SFFS algorithm was set to 15. The SFS algorithm stops either if the increase in classification accuracy is lower than 0.01% or the maximum number of 15 features is reached.

SVM and RF classifiers were learned and predicted using the *scikit-learn* Python library [36]. SVM was used with a Gaussian Radial Basis Function (RBF) kernel due to its good performance in comparison with other kernel functions [37] and low number of hyper-parameters that should be fitted: the regularization parameter  $C$  and the width of the RBF kernel function  $\gamma$ . They were fitted using grid search for the range of the regularization parameter  $C = \{1, 10, \dots, 10^5\}$  and  $\gamma = \{2^{-5}, \dots, 2^5\}$ . Optimal number of the classification trees in RF algorithm was found using the range from 10 to 500 trees with the step of 50 trees. In both cases, 5-fold CV was used to estimate the classification accuracy and fitted values were chosen to produce the best cross-validated classification accuracy.

Classification was based on three different levels. On the first level we tried to discriminate between Broadleaf and Conifer species. On the second level discrimination was based on 4 classes (deciduous broadleaves, evergreen broadleaves, pines or other conifers). On the third level of classification discrimination was based on thirteen possible tree species (*Table 3.1*).

All of the classifiers were trained with 2/3 randomly selected reference pixels per class, and the remaining pixels were used for the validation of the classifiers. To assess the classification accuracy, the confusion matrix, kappa statistics and overall accuracy were computed for 50 iterations and then averaged.

The number of the spectral features was different depending on the creation of SITS and depending on the year. If SITS were generated from the NDVI indices the number of the spectral features was equal to the number of acquisitions in the processing year (*Table 3.4*). If SITS were generated from the spectral bands the number of the spectral features was equal to the number of acquisitions multiplied by four (the number of spectral bands for each multi-spectral image).

The main program which applies all of the classifiers to SITS (*script\_classif\_formosat.py*) is presented in Appendix 2. This program implements RF and SVM classifiers using the *scikit-learn* Python library. The implementation of SFFS algorithm is also presented in Appendix 2 (*forward\_selection\_sffs.py*).

### **Step 3. Prediction on all the pixels of the SITS to produce thematic maps**

The last part of the project was to create thematic maps across the study area using three classifiers described before. In order to learn the classifiers for this case, the entire set of the available reference pixels was used. Prediction was done for each pixel from SITS which corresponds to the forest-part of the study area, which was presented in *Section 3.4*. The main function which was implemented for this part of the project is presented in Appendix 2 (*predict\_image.py*).

After thematic maps were produced for each year and for each of the classifiers, the most frequently discriminated species for each pixel was found, as well as the frequency of classifying the same pixel to the same class. This function is also presented in Appendix 2 (*thematic\_map\_consecutive\_years.py*).



#### **4.4 Used software and implementation**

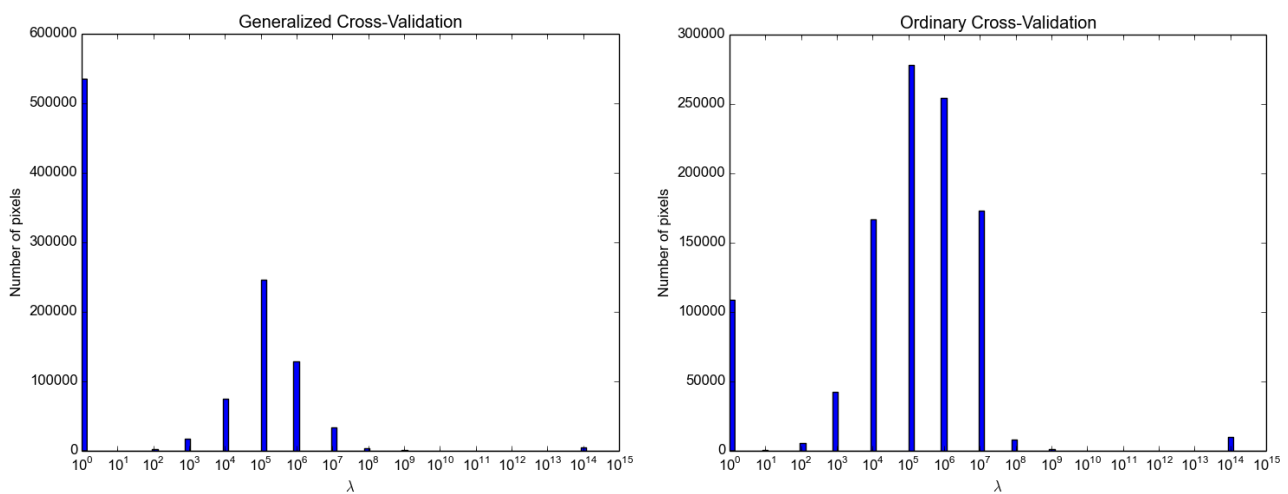
OTB (*Orfeo Toolbox*) applications and Shell Programming were used to create SITS. Smoothing function and all the classification methods were performed in Python with its GDAL library (*Geospatial Data Abstraction Library*). Moreover, QGIS software was used for the visualization of the images and rasterization of vector data.

## 5 Results and discussion

### 5.1 Pre-processing

#### Choosing optimal $\lambda$

The optimal value for regularization parameter for year 2013, for the vast number of pixels obtained using Generalized Cross Validation was  $\lambda=10^0$ . However, this value is very low and using it would make smoothing insensible. To remove the noise from SITS, only the values of  $\lambda$  that are higher than 1 were considered. Thus, the most frequent value for regularization parameter  $\lambda$  was  $10^5$  obtained from both Generalized and Ordinary Cross-Validation (Figure 5.1).

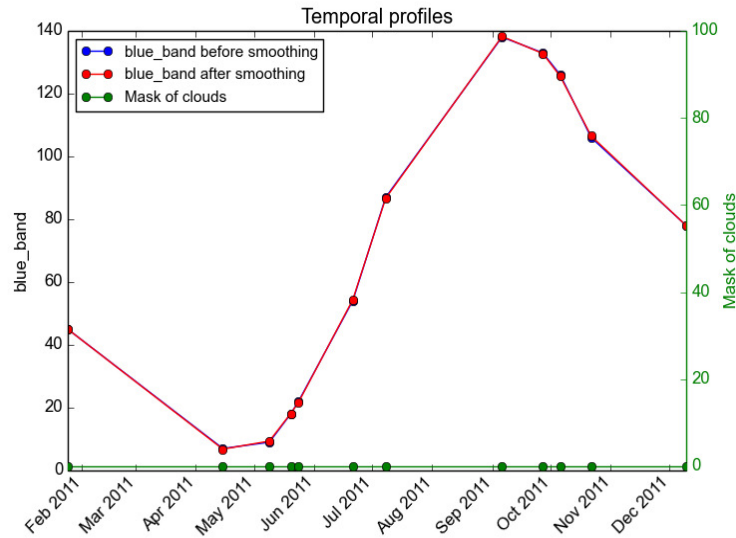


**Figure 5.1. Generalized and Ordinary Cross-Validation errors for different values of  $\lambda$  for year 2013**

The same regularization parameter value  $\lambda=10^5$  was obtained for year 2012. For year 2011 the optimal value obtained with Generalized Cross Validation was  $\lambda=10^5$  and with Ordinary Cross Validation was  $\lambda=10^6$ . However, the optimal value for that year was set to  $\lambda=10^5$ , since it was visually noticed that the final results were better than the results obtained with the second value. The optimal value for year 2014 was selected in the same way as for year 2011. The figures for years 2011, 2012 and 2014 are given in *Appendix 3 (Figures A3.1, A3.2 and A3.3)*.

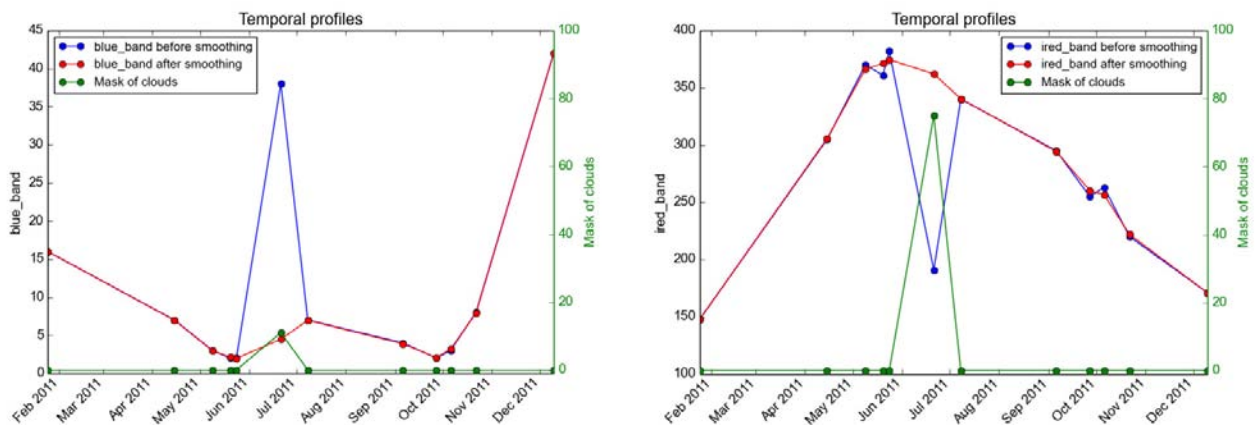
#### Temporal profiles before and after smoothing

Clouds and cloud shadows have a significant impact on the image time series. When present, white clouds will add positive noise to the values in blue, green, red and infrared bands, whilst cloud shadows will add negative noise to the same bands. Clouds and cloud shadows have opposite effect on NDVI, i.e. there are sudden drops of NDVI values when the clouds are present and increase of NDVI values in the presence of cloud shadows. To illustrate how smoothing struggles due to these effects, several pixels were chosen and their temporal profiles before and after smoothing are presented together with the corresponding temporal profiles of the pixels from the mask containing the information about the cloud and cloud shadow presence (*Figures 5.2, 5.3, 5.4 and 5.5*).

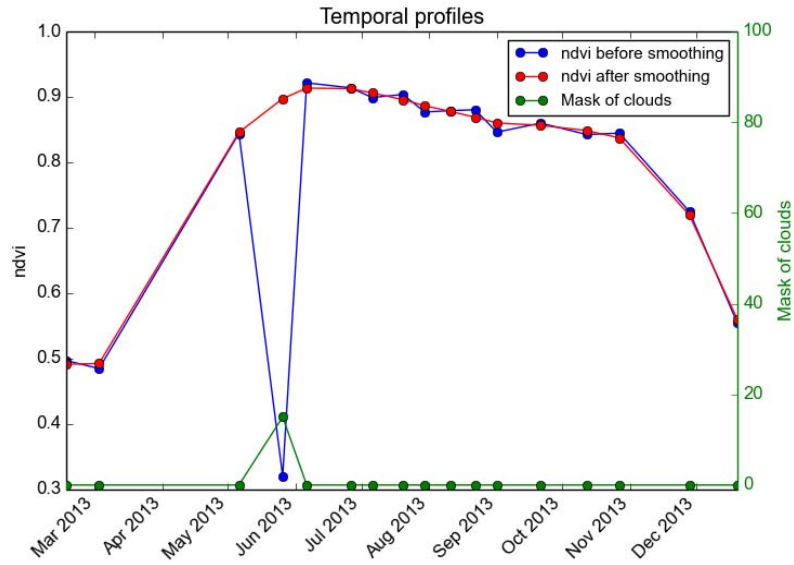


**Figure 5.2. Temporal profiles of the pixel in blue band before and after smoothing in case without clouds or cloud shadows detected. The temporal profiles are superimposed.**

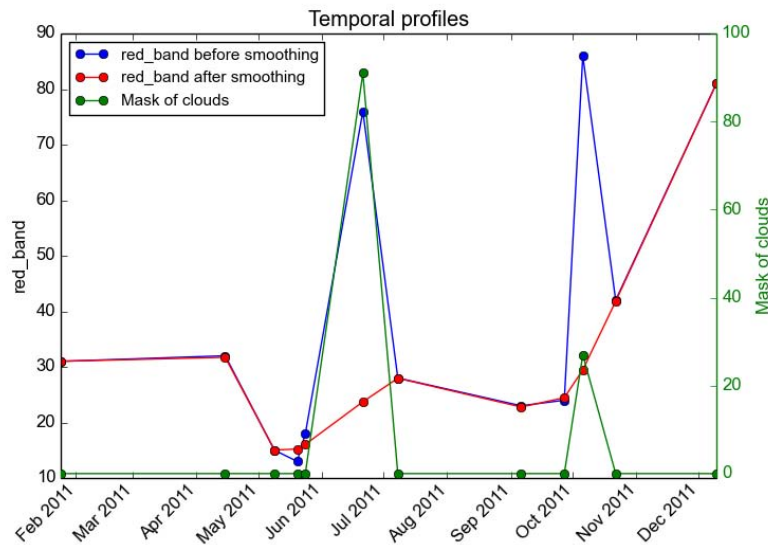
Figure 5.2 shows the situation when there were no clouds or cloud shadows detected (values of the mask were equal to 0 for all dates). In this case, values before and after smoothing are almost the same. Figure 5.3 (left), shows an example with white cloud which added noise to the value in the blue band. Figure 5.3 (right), illustrates an example when there was cloud shadow which caused sudden drop of the value in infrared band before smoothing was applied. In both examples it can be seen that smoothing algorithm managed to fill in the missing values. Figure 5.4 shows that NDVI reacts differently on white clouds and that smoothing managed to correct this value. And finally Figure 5.5 shows the result of smoothing of the pixel time series in the red band affected by clouds presented on two dates.



**Figure 5.3. Temporal profiles of the pixel in blue band affected by cloud before and after smoothing (left) and temporal profiles of the pixel in infra-red band affected by cloud shadow before and after smoothing (right)**



**Figure 5.4. Temporal profiles of the pixel containing NDVI values before and after smoothing affected by clouds**



**Figure 5.5. Temporal profiles of the pixel in red band before and after smoothing affected by clouds present on two dates**

Whittaker smoother showed high potential to eliminate the negative effects of clouds and cloud shadows from SITS (Figures 5.2, 5.3, 5.4 and 5.5). Moreover, Whittaker smoother removed some of the noise not caused by the presence of the clouds or cloud shadows, which could have appeared due to the aerosols in the atmosphere (Figures 5.3 - right, 5.4 and 5.5).

## Results of smoothing from the viewpoint of the images in the SITS

When comparing images with present cloud cover before and after smoothing (*Figure 5.6*), the image obtained after smoothing reveals high ability of Whittaker smoother to eliminate the cloud cover effects. However, applied smoothing technique is not perfect since the area under clouds remained somewhat darker than the area which was not under cloud cover.

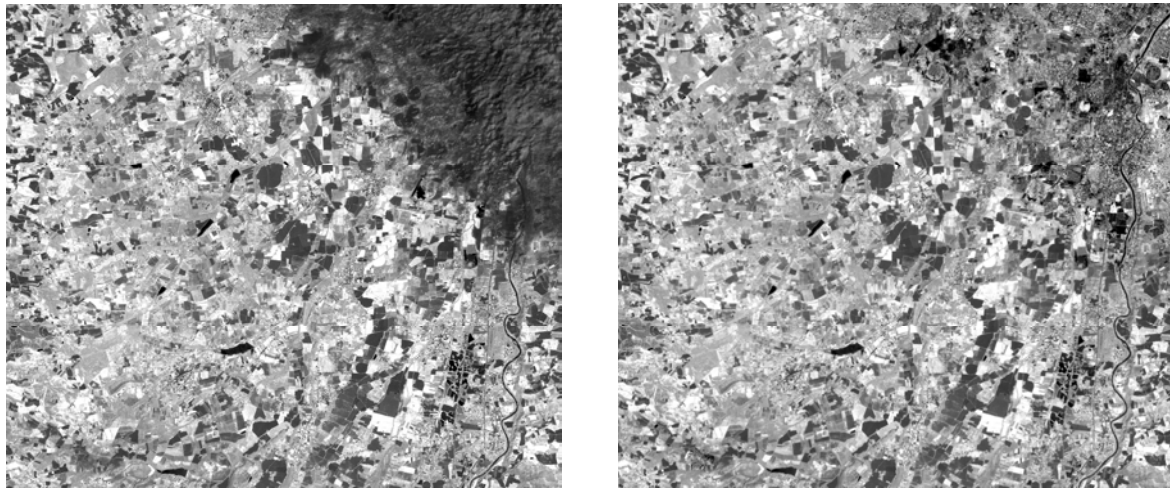


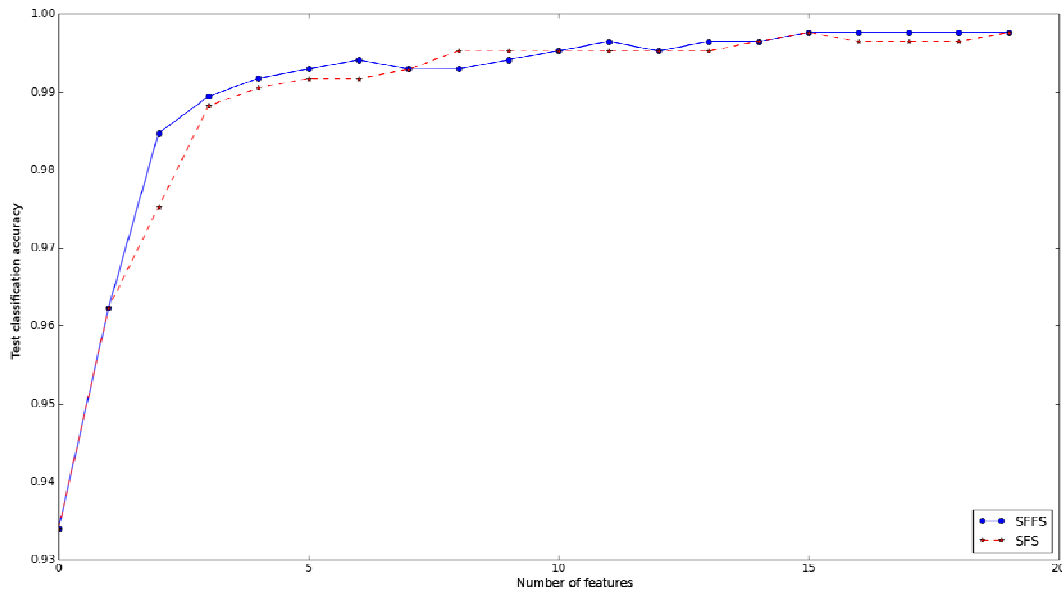
Figure 5.6. The grey-scale images with the clouds present in top-right corner before (left) and after (right) smoothing

## 5.2 Classification

Unexpectedly, SFFS algorithm did not perform any better than SFS algorithm (*Table 5.1*). The results obtained with SFFS algorithm were sometimes even worse than the results obtained with SFS algorithm. The test classification accuracies indicate that the original SFFS algorithm has a minor drawback. During backtracking after the number of variables was excluded, it may happen that including new variables could make the algorithm end up with the variable subset that is worse than a subset of the same size, which was found before backtracking. The problem is that the SFFS algorithm continues to follow this worse feature subset, although a better one has already been found (*Figure 5.7*). This drawback can be corrected by storing previous feature subsets and checking whether the SFFS algorithm follows wrong feature subset at each step. If SFFS algorithm follows wrong feature subsets than algorithm needs to change the current variable subset to the best one of the same size found earlier. This leads to the rather heavily increase of computational complexity both in terms of memory and time. Somol *et al.* [38] have observed this drawback of the original SFFS algorithm and proposed an updated version of the SFFS algorithm. Nevertheless, several experiments were performed where the original SFFS algorithm was applied on the hyper-spectral data sets with more reference samples and the results and discussion about these experiments are reported in *Appendix 1*. But for now the conclusion is that the original SFFS algorithm is not a good solution for a feature selection of Formost-2 image time series, because even when it gives better results than the SFS algorithm, these results are not significantly better when we take into considerations that slightly better results were achieved through the cost of significant increase in time complexity.

Level 1 (classification based on 2 classes)						
	NDVI (17 features)		SB (68 features)		SB+NDVI (85 features)	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)	OA (%)	Kappa (%)
NPFS_sfs	86.8 ± 0.03	70.0 ± 0.10	97.1 ± 0.01	93.0 ± 0.06	97.2 ± 0.01	93.3 ± 0.08
NPFS_sffs	87.2 ± 0.04	68.8 ± 0.35	97.5 ± 0.01	94.1 ± 0.06	98.0 ± 0.00	95.1 ± 0.03
RF	96.5 ± 0.01	91.8 ± 0.04	98.2 ± 0.00	95.7 ± 0.02	98.4 ± 0.00	96.2 ± 0.02
SVM	<b>97.0 ± 0.01</b>	<b>92.8 ± 0.06</b>	<b>99.6 ± 0.00</b>	<b>99.2 ± 0.01</b>	<b>99.5 ± 0.00</b>	<b>98.9 ± 0.01</b>
Level 2 (classification based on 4 classes)						
	NDVI (17 features)		SB (68 features)		SB+NDVI (85 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	91.6 ± 0.02	85.5 ± 0.05	96.2 ± 0.02	93.4 ± 0.05	96.1 ± 0.01	93.3 ± 0.05
NPFS_sffs	89.2 ± 0.08	81.4 ± 0.24	96.6 ± 0.01	94.1 ± 0.04	96.6 ± 0.01	94.0 ± 0.03
RF	95.5 ± 0.01	92.3 ± 0.04	96.9 ± 0.01	94.6 ± 0.03	98.0 ± 0.01	96.6 ± 0.03
SVM	<b>95.7 ± 0.01</b>	<b>92.6 ± 0.03</b>	<b>99.0 ± 0.00</b>	<b>98.2 ± 0.01</b>	<b>98.9 ± 0.00</b>	<b>98.0 ± 0.01</b>
Level 3 (classification based on 14 classes)						
	NDVI (17 features)		SB (68 features)		SB+NDVI (85 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	88.3 ± 0.03	87.0 ± 0.03	89.7 ± 0.14	88.5 ± 0.17	92.4 ± 0.08	91.6 ± 0.1
NPFS_sffs	87.9 ± 0.09	86.5 ± 0.12	88.8 ± 0.64	87.6 ± 0.77	91.9 ± 0.10	91.0 ± 0.13
RF	89.4 ± 0.02	88.2 ± 0.03	94.4 ± 0.01	93.8 ± 0.01	95.2 ± 0.01	94.7 ± 0.02
SVM	<b>91.8 ± 0.02</b>	<b>90.9 ± 0.02</b>	<b>98.2 ± 0.00</b>	<b>98.0 ± 0.00</b>	<b>98.1 ± 0.00</b>	<b>97.8 ± 0.01</b>

**Table 5.1 Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for year 2013. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that was chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over 50 repetitions in percentages. The best results for each level are reported in bold face.**



**Figure 5.7. Test classification accuracy for the different size of feature subsets selected using SFS or SFFS algorithms. Classification was based on the spectral bands together with the NDVI indices for level 2 of classification**

The results given in the *Table 5.1* revealed a high ability of the classifiers to discriminate tree species using Formost-2 image time series. For the classification level 1 (discrimination between broadleaf and coniferous tree classes) the lowest overall accuracy was obtained for NPFS\_sfs GMM based classifier when using only NDVI indices as the spectral features. The highest overall accuracy was obtained for SVM classifier using both NDVI indices and spectral bands together as the spectral features. Similar results have been obtained for the classification levels 2 and 3. In general, using only the NDVI indices as spectral features produced the lowest overall accuracy. Classifications based on spectral bands gave notably better results. Adding NDVI to spectral bands produced only slight improvements in classification accuracy, when compared to classifications based on spectral bands only. When compared to other classifiers, SVM classifier performed the best in terms of classification accuracy, while the NPFS\_sfs GMM based algorithm is more than two times faster than SVM algorithm (The mean processing time for NPFS\_sfs was 11 s while the mean processing time for SVM was 27 s). RF classifier performed the worst in terms of computational time with mean processing time of 59 s.

The classification results for other years and each of the created SITS are given in *Appendix 3* (*Tables A3.1, A3.2 and A3.3*). For all of the years the results were consistent with the previously explained results obtained for year 2013. *Table 5.2* reports the results in terms of overall accuracy and Kappa statistics for each of the four years based on only spectral bands, since for each of the years the spectral bands showed to be a satisfactory solution for the SITS creation. The classification accuracy for one year and for the same classifier decreases with increasing the number of classes that need to be discriminated i.e. going down from level 1 to level 3. For classification levels 1 and 2 the highest accuracy was obtained for year 2012, while for level 3 the highest accuracy was obtained for year 2014. This is probably due to the variation of the image acquisition dates available for the SITS creation (*Table 3.4*). For year 2012 a lot of images were acquired during winter, spring and autumn, allowing the relatively easy discrimination between coniferous and broadleaf classes. For year 2014 there was a lot of images acquired during key phenological periods (March, April, September,

October, November), which might have helped discriminating between the species at level 3 of classification.

Level 1 (classification based on 2 classes)						
	NPFS_sfs		RF		SVM	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)	OA (%)	Kappa (%)
2011 (48 features)	97.3 ± 0.01	93.7 ± 0.05	97.6 ± 0.01	94.3 ± 0.04	99.5 ± 0.00	98.7 ± 0.01
2012 (52 features)	<b>98.3 ± 0.01</b>	<b>95.9 ± 0.03</b>	<b>99.3 ± 0.00</b>	<b>96.7 ± 0.02</b>	<b>99.7 ± 0.00</b>	<b>99.2 ± 0.00</b>
2013 (68 features)	97.1 ± 0.01	93.0 ± 0.06	98.2 ± 0.00	95.7 ± 0.02	99.6 ± 0.00	99.2 ± 0.01
2014 (60 features)	97.7 ± 0.01	94.6 ± 0.05	98.6 ± 0.00	96.6 ± 0.03	99.5 ± 0.00	98.8 ± 0.01
Level 2 (classification based on 4 classes)						
	NPFS_sfs		RF		SVM	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)	OA (%)	Kappa (%)
2011 (48 features)	96.3 ± 0.01	93.7 ± 0.04	95.7 ± 0.01	92.6 ± 0.04	97.6 ± 0.01	95.9 ± 0.03
2012 (52 features)	<b>97.1 ± 0.01</b>	<b>94.9 ± 0.02</b>	<b>98.1 ± 0.00</b>	<b>96.8 ± 0.02</b>	<b>99.0 ± 0.00</b>	<b>98.2 ± 0.01</b>
2013 (68 features)	96.2 ± 0.02	93.4 ± 0.05	96.9 ± 0.01	94.6 ± 0.03	<b>99.0 ± 0.00</b>	<b>98.2 ± 0.01</b>
2014 (60 features)	96.6 ± 0.01	94.2 ± 0.04	97.5 ± 0.01	95.7 ± 0.02	98.8 ± 0.00	98.0 ± 0.01
Level 3 (classification based on 14 classes)						
	NPFS_sfs		RF		SVM	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)	OA (%)	Kappa (%)
2011 (48 features)	90.4 ± 0.04	89.4 ± 0.05	90.9 ± 0.03	89.9 ± 0.03	95.3 ± 0.01	94.8 ± 0.01
2012 (52 features)	91.4 ± 0.03	90.5 ± 0.03	94.4 ± 0.04	93.7 ± 0.05	97.9 ± 0.01	97.6 ± 0.01
2013 (68 features)	89.7 ± 0.14	88.5 ± 0.17	94.4 ± 0.01	93.8 ± 0.01	98.2 ± 0.00	98.0 ± 0.00
2014 (60 features)	<b>94.7 ± 0.01</b>	<b>94.1 ± 0.02</b>	<b>95.9 ± 0.01</b>	<b>95.5 ± 0.02</b>	<b>98.6 ± 0.00</b>	<b>98.4 ± 0.00</b>

**Table 5.2 Overall accuracy and Kappa statistics for three defined levels of classification for each of the four years. Only spectral bands were used as the spectral features. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that was chosen using standard feature selection algorithm described in Section 2.3.1. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over 50 repetitions in percentages. The best results for each level were reported in bold face.**



As expected, SVM showed less confusion than GMM classifier for each of the four years (Tables 5.3-5.8 and A3.4-A3.21). Confusion matrices for level 1 of the both classifiers in the year 2013 indicate very high accuracies (Tables 5.3 and 5.4). For level 2 and the same year, confusion matrices show lower accuracies in comparison to the accuracies obtained for level 1. For GMM classifier, *Deciduous* and *Evergreen* species showed high accuracies, while the confusions appeared with *Pine*, of which 5.48 % were assigned to *Deciduous* and 5.62 % to *other conifer*, and *other conifer*, of which 4.16 % were assigned to *Deciduous* and 6.22 % to *Pine* (Table 5.5). The highest confusions for SVM classifier appeared among two *Conifer* species, where 4.60 % of the *Other conifers* were wrongly assigned to *Pine* (Table 5.6). Confusion matrices for the third level in year 2013 had the lowest accuracies of all three levels (Tables 5.7 and 5.8). For GMM classifier, among conifer tree species the lowest accuracy is obtained for *Douglas fir* (59.83 %), where the highest confusion appeared with *Silver fir* (18.50 %) and *Black pine* (67.29 %) where the main confusion appeared with *Oak* (14.71 %). Higher accuracies were obtained for broadleaf species, with the lowest accuracy for *Black locust* (67.83 %), where 24.50 % confusions were with *Oak*, while *Willow*, *Red oak* and *Aspen* showed the highest accuracies (Table 5.7). For SVM classifier *Douglas fir* was again the most difficult to discriminate among coniferous tree species with the accuracy 82.50 %, where the highest confusion was again with *Silver fir* (12.17 %), while other coniferous tree species as well as all of the broadleaf species showed high accuracies, of which *Silver birch*, *Eucalyptus*, *Willow* and *Red oak* had the highest (Table 5.8), while the lowest accuracy was obtained for *Black locust*. As expected, a decrease in classification accuracy was observed with increase of the number of classes, i.e. the classification accuracy decreased going from level 1 to level 3. Results for other years indicate similar accuracies as for year 2013 (Tables A3.4-A3.21).

Results obtained for level 2 showed that both GMM and SVM models had more confusions among conifer tree species. The reason for this might be that there were significantly less reference pixels available for coniferous tree species (315) than the number of reference pixels available for broadleaf tree species (723) but also the fact that phenology is less informative for coniferous than for deciduous tree species. Results obtained for level 3 showed that for both GMM and SVM models the lowest classification accuracies among all of the conifer tree species was obtained for *Douglas fir* which had the lowest number of reference pixels of all the conifer tree species (Table 3.1). The same conclusion can be made after looking at the results for Broadleaf tree species. Again, for both SVM and GMM classifiers, among all of the Broadleaf tree species, the lowest accuracy was obtained for *Black locust*. The reason for this may be the same, since *Black locust* also had the lowest number of reference samples among broadleaf tree species (35) which is very small compared to the available reference samples for *Aspen* (209).

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	92.86	7.14
<b>Broadleaf</b>	1.05	98.95

**Table 5.3. Confusion matrix for level 1, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.**

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>99.35</b>	0.65
<b>Broadleaf</b>	0.23	<b>99.77</b>

Table 5.4. Confusion matrix for level 1, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.59</b>	0.01	0.40	0.00
<b>Evergreen</b>	0.35	<b>99.00</b>	0.41	0.24
<b>Pine</b>	5.48	0.64	<b>88.26</b>	5.62
<b>Other conifer</b>	4.16	0.11	6.22	<b>89.51</b>

Table 5.5. Confusion matrix for level 2, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.93</b>	0.00	0.07	0.00
<b>Evergreen</b>	1.06	<b>98.65</b>	0.29	0.00
<b>Pine</b>	0.52	0.00	<b>98.81</b>	0.67
<b>Other conifer</b>	1.35	0.05	4.60	<b>94.00</b>

Table 5.6. Confusion matrix for level 2, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

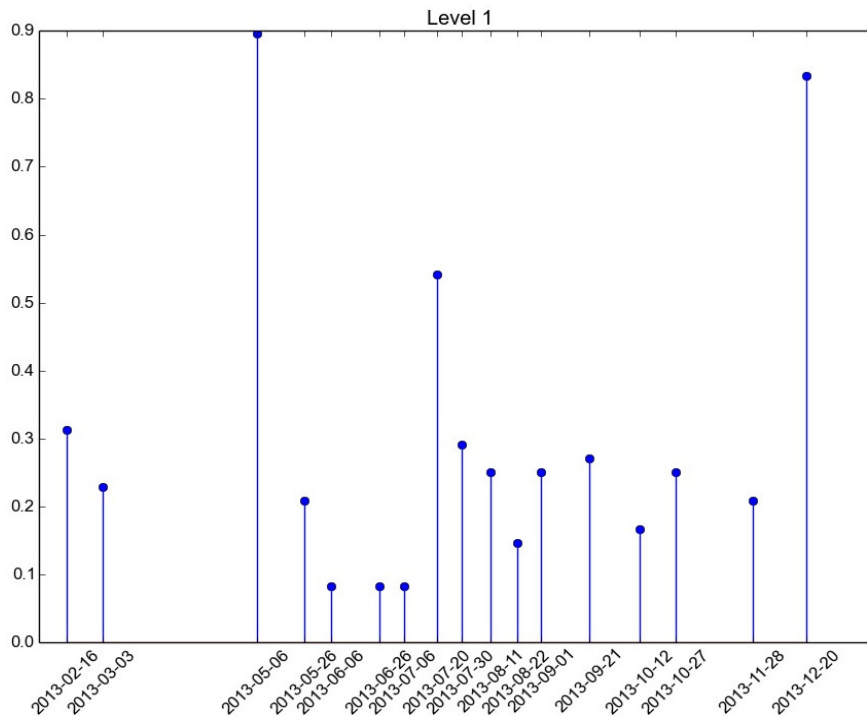
<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	<b>91.36</b>	0.64	2.72	<b>4.24</b>	0.00	0.00	0.08	0.00	0.00	0.00	0.24	0.40	0.00	0.32
Oak	0.00	<b>90.05</b>	1.05	0.29	1.86	1.81	0.05	1.28	<b>1.95</b>	0.09	0.24	0.62	0.09	0.62
Black pine	6.57	<b>14.71</b>	<b>67.29</b>	4.29	0.00	0.14	3.29	0.00	0.00	0.00	1.57	1.57	0.00	0.57
Douglas fir	<b>18.50</b>	6.33	3.50	<b>59.83</b>	0.00	0.33	3.00	0.00	0.17	0.00	3.00	4.00	0.00	1.33
Silver birch	0.00	<b>6.00</b>	0.00	0.00	<b>92.40</b>	0.00	0.00	0.00	0.16	0.24	0.00	0.00	0.00	1.20
European ash	0.00	<b>15.00</b>	0.00	0.00	0.00	<b>80.43</b>	0.00	1.57	2.86	0.00	0.00	0.00	0.00	0.14
Maritime pine	0.14	1.71	4.29	<b>6.14</b>	0.00	0.00	<b>84.43</b>	0.00	0.00	0.00	0.71	0.29	0.00	2.29
Black locust	0.00	<b>24.50</b>	0.00	0.00	0.33	5.17	0.00	<b>67.83</b>	1.83	0.00	0.00	0.00	0.00	0.33
Aspen	0.00	<b>2.74</b>	0.00	0.00	0.03	0.14	0.00	0.03	<b>96.17</b>	0.14	0.00	0.00	0.17	0.57
Red oak	0.00	<b>1.88</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.65	<b>97.18</b>	0.06	0.00	0.00	0.24
Eucalyptus	0.12	<b>1.47</b>	0.41	0.71	0.00	0.82	0.12	0.06	0.06	0.00	<b>95.41</b>	0.06	0.00	0.76
Corsican pine	1.00	<b>4.71</b>	1.86	2.71	0.00	0.14	0.14	0.00	0.00	0.00	0.14	<b>88.71</b>	0.00	0.57
Willow	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>98.77</b>	<b>1.08</b>
Austrian black pine	0.00	0.07	0.48	2.14	0.00	0.00	<b>4.21</b>	0.00	0.00	0.00	3.24	0.07	0.00	<b>89.79</b>

Table 5.7. Confusion matrix for level 3, year 2013. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

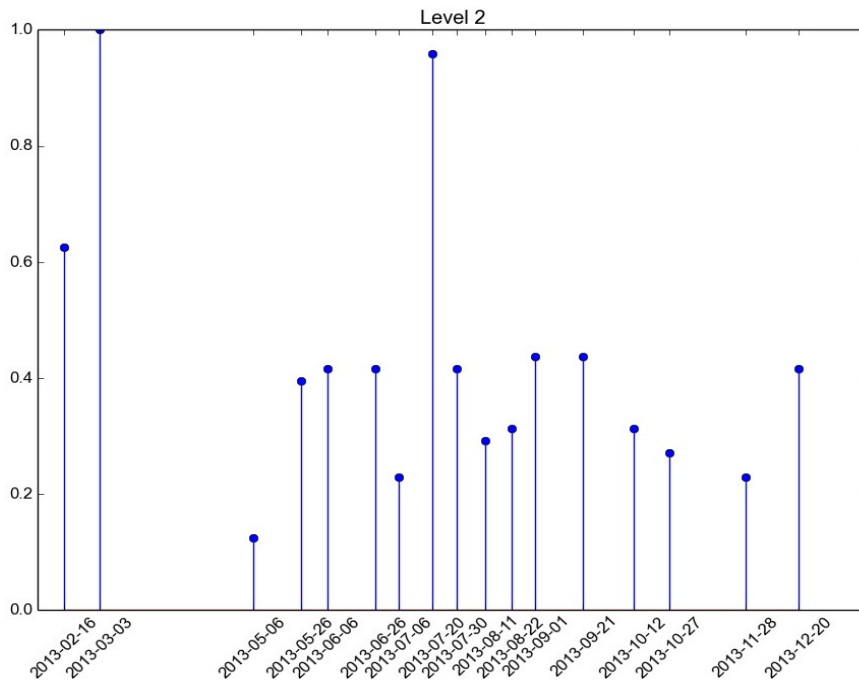
<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	96.72	0.00	1.60	1.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Oak	0.00	98.24	0.00	0.00	0.57	0.10	0.05	1.05	0.00	0.00	0.00	0.00	0.00	0.00
Black pine	0.57	2.57	96.00	0.00	0.00	0.00	0.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Douglas fir	12.17	0.83	1.50	82.50	0.00	0.00	0.67	0.00	0.00	0.00	0.50	1.83	0.00	0.00
Silver birch	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
European ash	0.00	1.29	0.00	0.00	0.00	98.71	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Maritime pine	0.00	0.00	3.57	0.00	0.00	0.00	94.86	0.00	0.00	0.00	0.00	0.14	0.00	1.43
Black locust	0.00	3.17	0.00	0.00	0.00	1.00	0.00	95.83	0.00	0.00	0.00	0.00	0.00	0.00
Aspen	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	99.91	0.00	0.06	0.00	0.00	0.00
Red oak	0.00	0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.18	99.59	0.00	0.00	0.00	0.00
Eucalyptus	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.94	0.00	0.00	0.06
Corsican pine	0.00	1.29	0.29	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	98.00	0.00	0.00
Willow	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.00	99.69	0.00
Austrian black pine	0.00	0.00	0.00	0.00	0.00	0.00	1.03	0.00	0.00	0.00	0.00	0.00	0.00	98.97

Table 5.8. Confusion matrix for level 3, year 2013. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

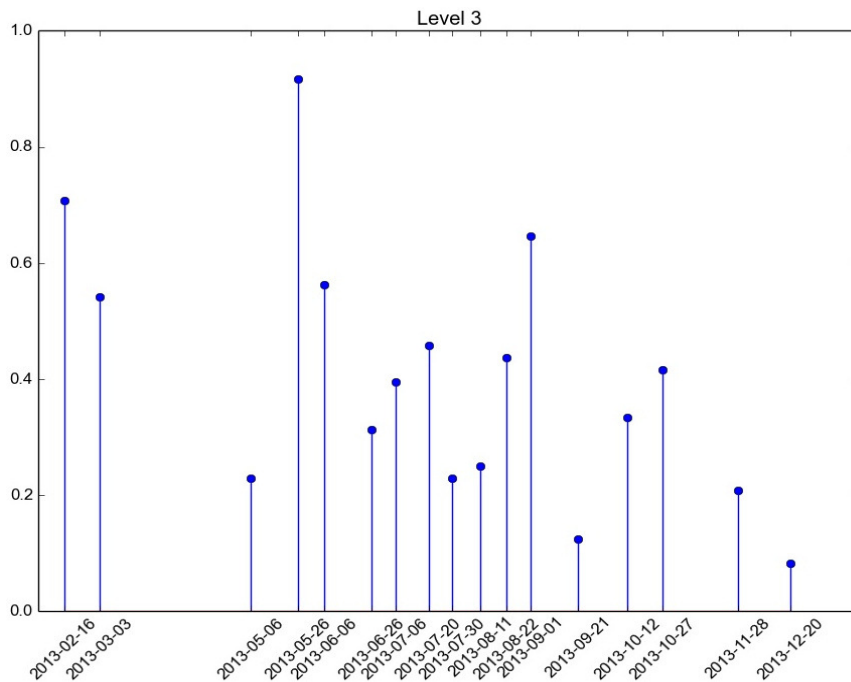
To identify the most important dates for discriminating among tree classes for all three levels, the nonlinear parsimonious feature selection approach (NPFS) was used. The selection rate of the used dates over 50 iterations for year 2013 is shown in *Figures 5.8, 5.9 and 5.10*. Images that were acquired during May had the highest selection rate (43/50) for discriminating at level 1. Furthermore, images from December were also frequently chosen (with the selection rate of 40/50). This was expected since the images that were acquired during spring and winter should be the one of the best for separating broadleaf and coniferous species. Winter images together with the images acquired in July were the most frequently selected for the classification at level 2. At level 3, among many different dates which were frequently selected, the highest selection rate is obtained for the images acquired in winter, May and mid-summer. Contrary to our expectations, images acquired during growing and senescing seasons were not very useful for discriminating among tree species, possibly because only few images were acquired during key phenological periods. The most frequently selected dates for the other years are given in *Appendix 3 (Figure A3.4-A3.12)*.



**Figure 5.8.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).



**Figure 5.9.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).



**Figure 5.10.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2013. The selection rate is 1 when the image is selected systematically (50/50).

Classification maps for each level obtained with GMM and SVM classifiers for year 2013 are presented in *Figures 5.11, 5.12 and 5.13*. Classification maps obtained here should be verified in the future with ground surveys, although confusion matrices indicate that the results obtained with GMM classifier were noisier (with higher confusions) than with the SVM classifier.

*Figures 5.14, 5.15 and 5.16* represents the frequency maps which show the number of time that each pixel was classified with the same label looking at the classification maps obtained for four year. Pixels having a frequency value of 4 indicated high consistency between four classifications. A lower frequency indicated more differences between the years and most likely more uncertainty on the classification accuracy, since it is unlikely that species composition changed through the four studied years. For levels 1 and 2, most of the pixels had a frequency value of 4 or 3, suggesting high stability of the classification between the years. Higher instability is indicated for level 3, since most of the pixels had the frequency value of 2. One of the reasons for the different positioned classification errors for consecutive years could be explained by the difference in the dates of the images acquisition between the years (*Table 3.4*). These maps should be analysed more deeply in the future.

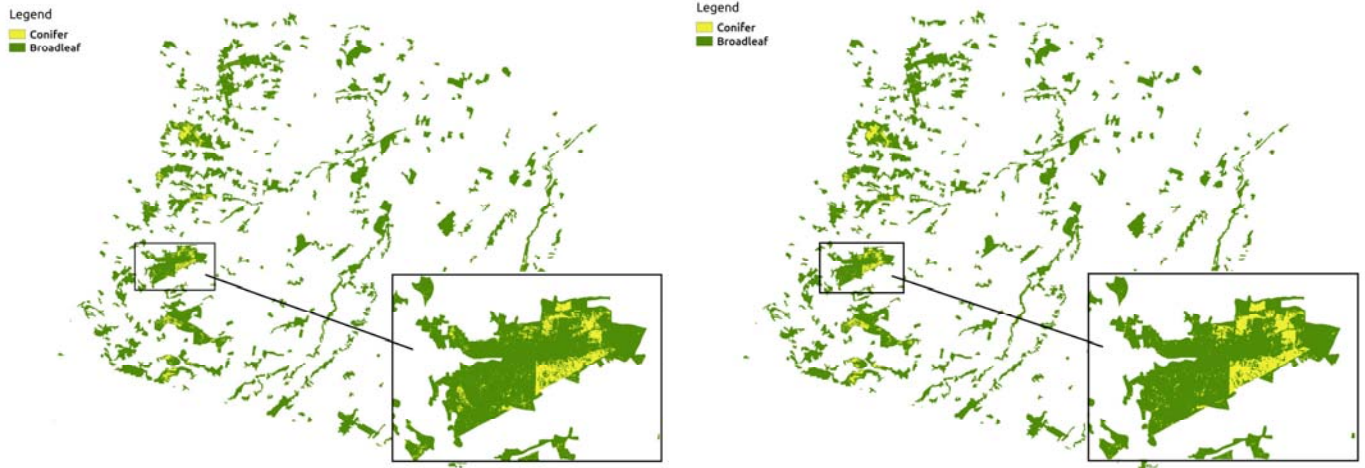


Figure 5.11. Classification maps for level 1 based on spectral bands with GMM (left) and SVM (right)

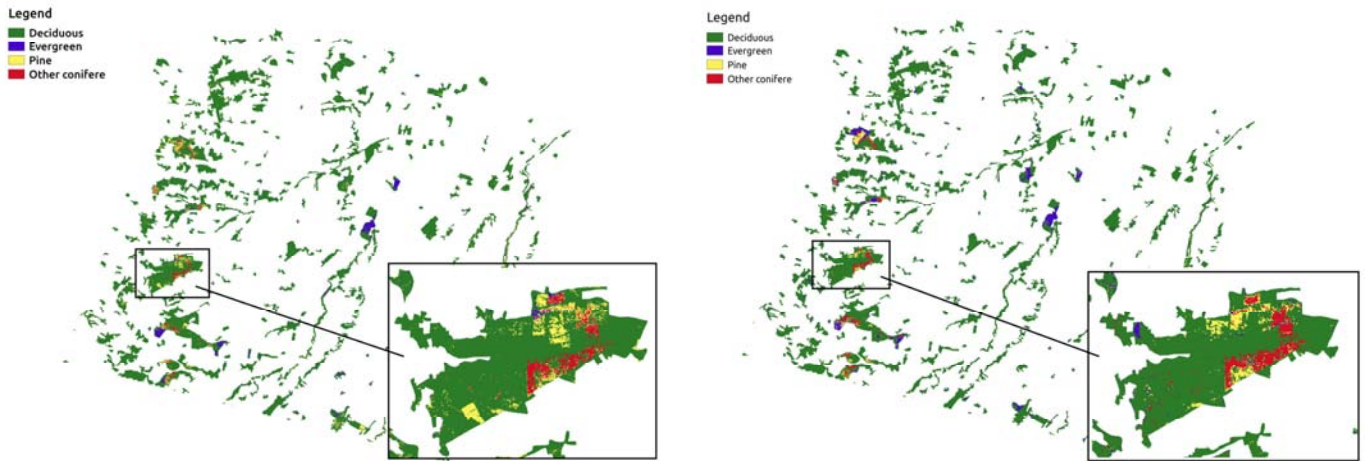


Figure 5.12. Classification maps for level 2 based on spectral bands with GMM (left) and SVM (right)

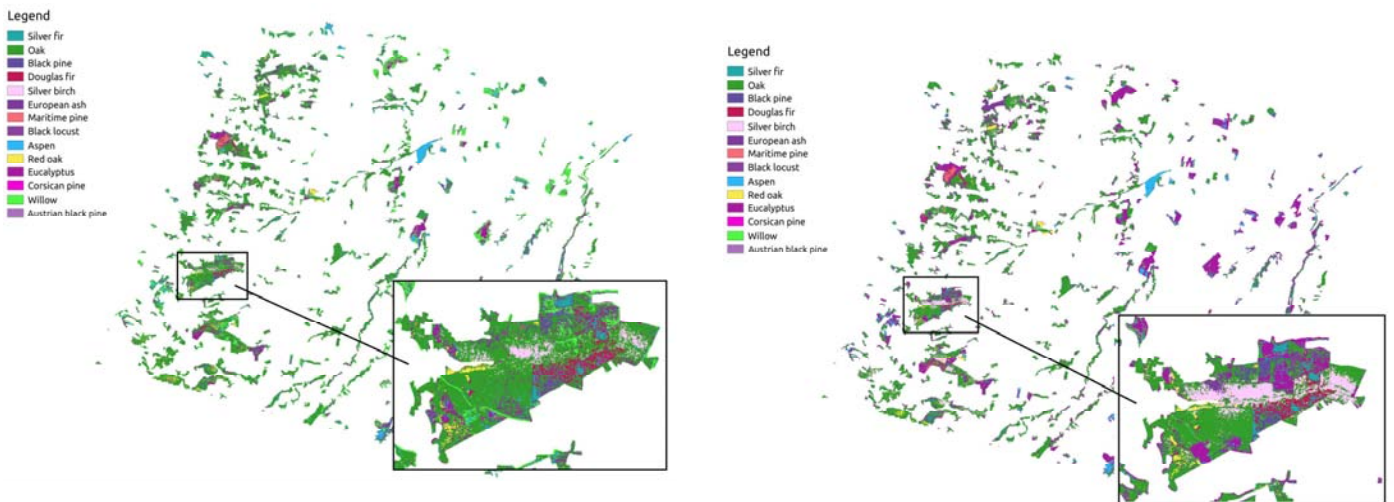
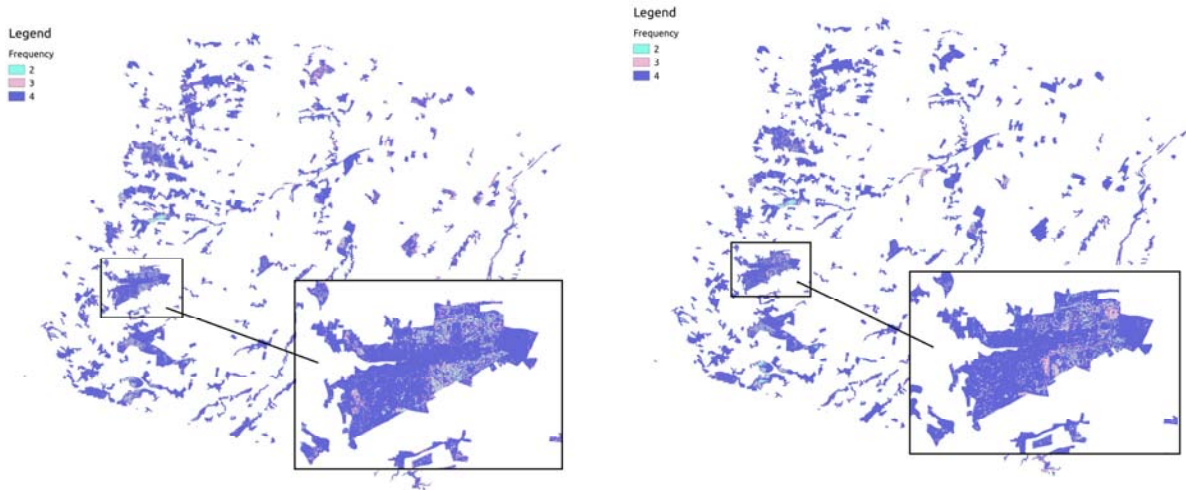
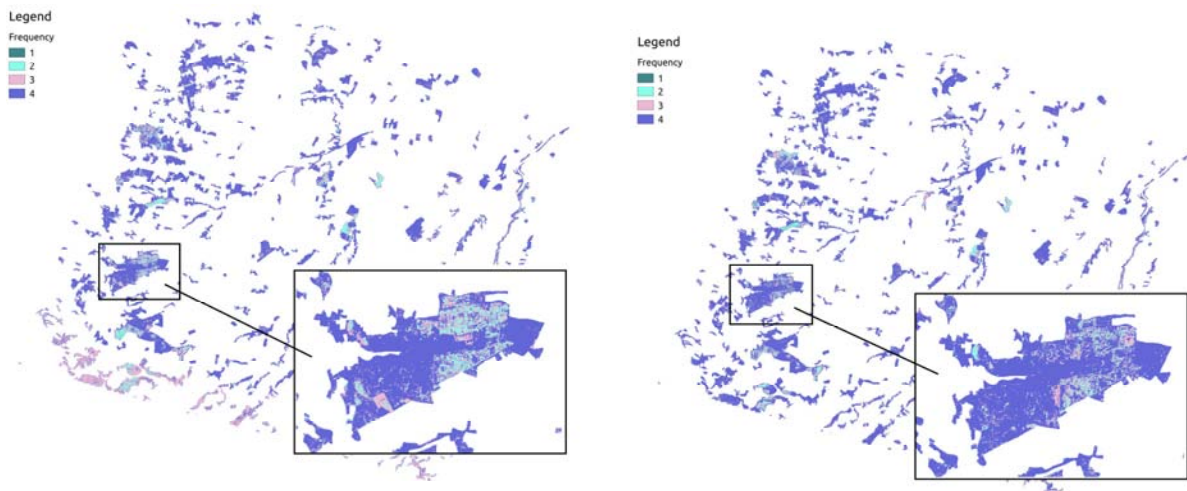


Figure 5.13. Classification maps for level 3 based on spectral bands with GMM (left) and SVM (right)

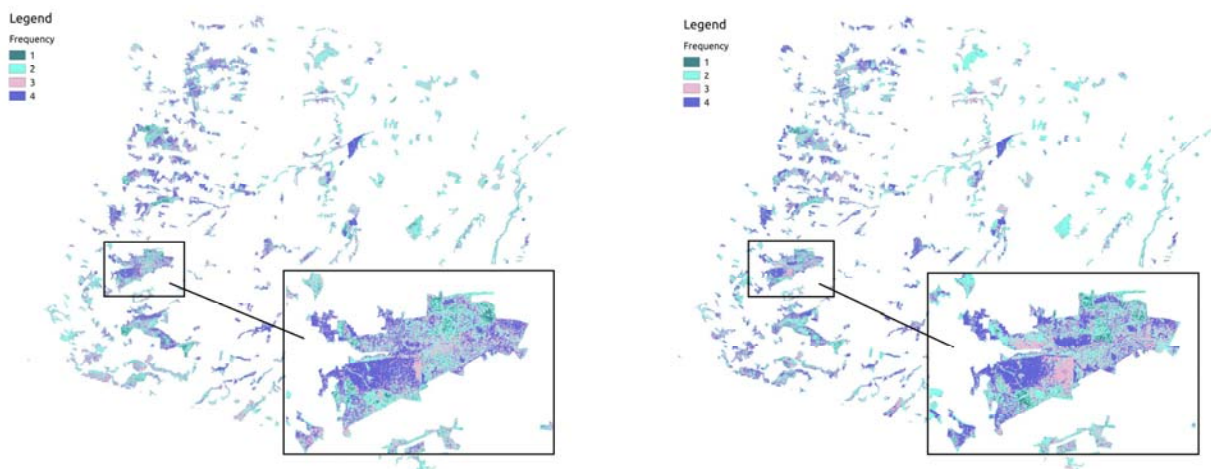




**Figure 5.14. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 1 based on spectral bands with GMM (left) and SVM (right)**



**Figure 5.15. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 2 based on spectral bands with GMM (left) and SVM (right)**



**Figure 5.16. Frequency maps obtained for years 2011, 2012, 2013 and 2014 for level 3 based on spectral bands with GMM (left) and SVM (right)**

## 6 Conclusion and future work

The main aim of this study was to investigate the ability of Formosat-2 image time series to discriminate different tree species. It was assumed that phenological variations of the tree species during the year could potentially increase the spectral separability between different species. The main conclusions of the study were:

- The Whittaker smoothing filter showed to be a good solution for smoothing the image time series. However, the optimal regularization parameter obtained with generalized cross-validation sometimes gave the results which were under-smoothed whilst ordinary cross validation gave the results which were over-smoothed. In these cases the optimal regularization parameter value was chosen to give visually good results.
- All of the classifiers produced high classification accuracies for each of the considered years which indicate the importance of phenological information for discriminating various tree species. However, SVM classifier performed the best in terms of classification accuracy whilst NPFS performed the best in terms of computing time. In fact, NPFS algorithm usually extracted only a few features (lower than 5% of the total number) and was therefore much faster and more efficient than the other two classifiers, while giving comparable classification accuracy. RF classifier gave lower classification accuracy than SVM algorithm and better accuracy than GMM classifier but used more processing time. Therefore, RF algorithm might not be a good solution for tree species discrimination from SITS. The accuracies obtained in this study were higher than the accuracies obtained in the comparable studies. For example, *Table 11* from the study [43] gives an overview of different studies that were based on tree species discrimination using data sensors with different spatial and spectral resolution. Wide range of the classification methods was applied and the reported accuracies varied between 45 % and 96 %. The highest values were obtained for discriminating between only a few tree species (96 % was the highest obtained accuracy when classification was based on only 3 tree species). In comparison, results of this study on level 3 of classification (discrimination between 14 tree species), for year 2014 with SVM classifier, indicate overall accuracy of 98.6 % which is significantly better result with significantly more species.
- The experimental results of this study indicate that SFFS algorithm might not be superior to the SFS algorithm. The same conclusion was made in the study [42]. However, this conclusion contradicts widely-held belief that floating search methods are superior to the simple sequential ones. For example, Jain et al. [29] claim that “in almost any large feature selection problem, floating search methods [SFFS and SBFS] perform better than the straightforward sequential searches, SFS and SBS.” This claim together with many others [24, 30, 31] persuade many people to use floating variable selection methods instead of simple algorithms [39, 40, 41], although according to the results of this study these simple algorithms may give equally good results in significantly less time.

When it comes to the future work, further development is needed to improve the way of choosing optimal regularization parameter for smoothing. Additional reference data could be also collected to improve the classification accuracy for the species which showed the lowest classification results, at least for the species with the lowest number of reference pixels (Douglas fir and Black locust). As the post-processing and the next step of this study spatial smoothing could be applied to the classification maps in order to eliminate isolated classified pixels and to produce more compact classes (homogeneous regions).

### **Personal contributions of the internship**

This internship was of remarkable importance to me because I have immensely improved my previous knowledge during these 5 months.

During the academic year I took a course in Remote Sensing, where we studied about basics of remote sensing with the implementation of different classification approaches and the analysis of the temporal profiles of the NDVI vegetation index. Also, I took a course in Classification and Pattern Recognition where we studied about different classification methods. This internship helped me a lot to improve my previous knowledge and to apply the theoretical knowledge obtained through these two courses in practice.

Through this internship I have also improved my programming skills with the Python programming language and Shell Programming, previously completely unknown to me, and I also worked for the first time with the GIS software QGIS.

I have learned a lot about forests and their importance for our environment. This subject was very interesting to me since it is related to a very important issue.

I have also discovered the world of the research though many interactions with other people from the lab, which is also very significant experience for me.

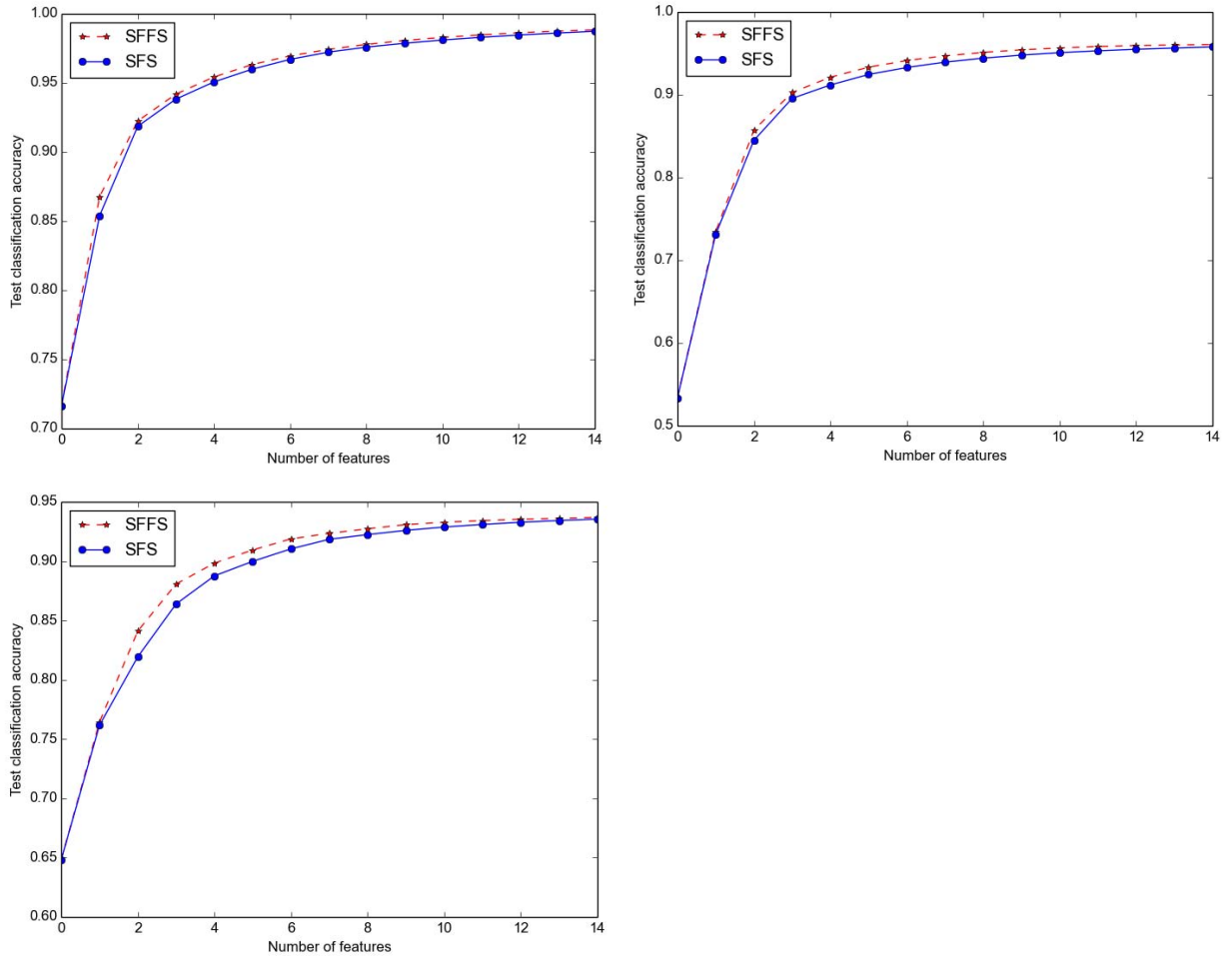
## Appendix 1

Several additional experiments were performed on real hyper-spectral data sets to explore the ability of implemented SFFS algorithm to find a superior subset of features in comparison to SFS algorithm using data sets with more spectral bands and more reference pixels. Four data sets were used for this purpose. The first data set has been acquired in the region surrounding the volcano Hekla in Iceland by the AVIRIS sensor. It contains 157 spectral channels from 400 to 1840 nm. The total number of reference pixels was 10227 for which 12 classes have been defined. The second data set has been acquired during a flight campaign over Pavia, northern Italy, with 103 spectral channels recorded from 430 to 860 nm. The number of reference pixels was 42776 for 9 defined classes. The third data set has been acquired by the NASA AVIRIS instrument over the Kennedy Space Centre (KSC), with 176 spectral bands from 400 to 2500 nm used for the analysis, and 13 classes for a total of 361753 pixels.

For each data set, two thirds of the pixels were randomly selected per class as training pixels and the remaining reference pixels were used for validation. 50 repetitions were performed and new training set generated for each repetition. In order to compare SFFS and SFS algorithms the maximum number of extracted features was set to 15. Each variable has been standardized before the processing in order to get zero mean and unit variance.

Stratified 5-fold cross validation was used to estimate the mean test classification accuracy. As described in Section 2.1, parameters of the GMM sub-models have been estimated using update rules. The mean test classification accuracy with a subset of selected features has been estimated using the marginalization properties of the Gaussian distribution parameters.

*Figure A1.1* presents the estimated mean test classification accuracies during the training procedure for each size of selected feature subsets until the size of 15 best features is reached. For each data set SFFS performed better than SFS for the smaller size of feature set. Moreover, by increasing the number of features in the feature set SFFS performance was becoming similar to the performance of the SFS algorithm.

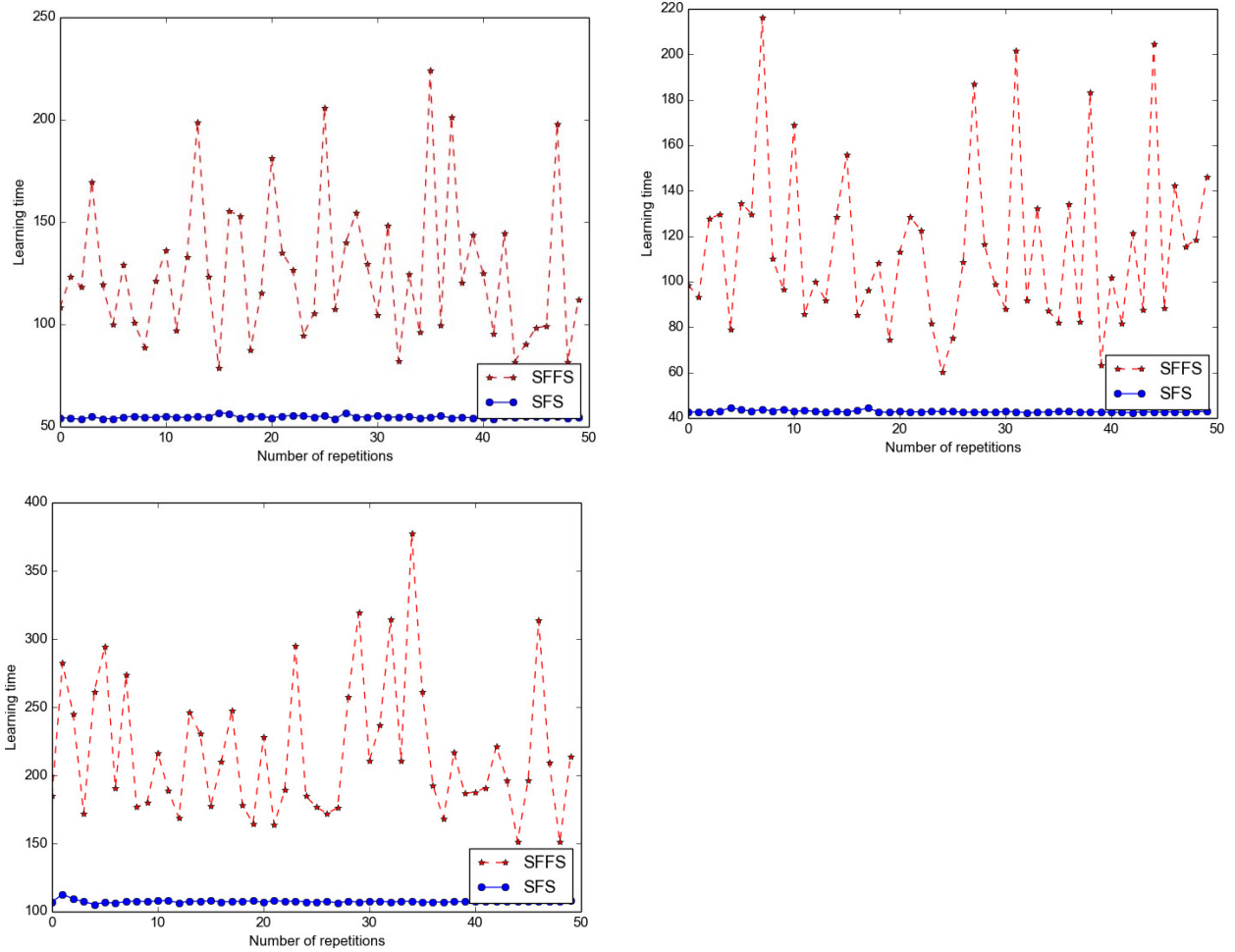


**Figure A1.1.** The test classification accuracy for Hekla data set (top left), KSC (top right) and Pavia (bottom left)

Table A1.1 shows the mean value of the overall accuracy (OA) and Kappa statistics averaged over 50 repetitions for each data set. From there, it can be seen that SFFS performed the same as SFS for Hekla data set, whilst for the other sets it gave slight improvements. However, each data set needed a lot of backtracking to be done, i.e. much more time was spent to obtain these slightly better results (Figure A1.2 and Table A1.2). This is the general problem of the SFFS algorithm independently of the data on which it is applied, since it is not possible to tell in advance for how long the algorithm is going to run. The complexity will be similar to the complexity of the SFS algorithm if data set is such that no backtracking needs to be done, but then the result will not be improved. On the other hand, if a lot of backtracking is needed, the results will be slightly better, but more time will be needed to obtain it.

<i>Data sets</i>	<i>SFS algorithm</i>		<i>SFFS algorithm</i>	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)
Hekla	98.3	98.1	98.3	98.1
Pavia	93.4	91.3	93.5	91.4
Ksc	94.9	94.4	95.1	94.6

**Table A1.1.** Mean values of the overall accuracies and Kappa statistics for each data set



**Figure A1.2.** Learning time needed to extract 15 best features for each iteration for Hekla data set (top left), KSC (top right) and Pavia (bottom left)

<i>Data sets</i>	<i>Learning time [s]</i>		<i>Prediction time [ms]</i>	
	SFS	SFFS	SFS	SFFS
Hekla	54.75	126.2	51.8	50.5
Pavia	107.41	217.18	168.73	177.7
Ksc	42.87	115.14	28.21	29.95

**Table A1.2** The mean processing time for learning and prediction for each data set

Two more things related to SFFS algorithm were noticed:

- Even though the original SFFS algorithm sometimes gave the results which were superior during the training phase in terms of estimated classification accuracy, this does not always translate to better results for unseen data.
- The SFS algorithm will stop when the maximum number of variables is reached or if the increase of estimated classification accuracy is lower than a predetermined threshold. In this way, SFS sometimes finds a subset of features with the size of only 5% of the original feature set size and which gives very high classification accuracy and thus performs very fast in terms of computational time. On the contrary, stopping criterion for the SFFS algorithm should be defined only with the maximum number of features since introducing the threshold for the test classification rate can stop the algorithm before backtracking. For example, it can stop the algorithm when only a few features were selected, although the algorithm may find better feature subset of the same size, after it finishes with the higher feature subset backtracking.

## Appendix 2

This chapter presents one part of the code that I have implemented for the purposes of this work. Most of them I implemented in Python whilst the SITS creation I did using highly specialized tool, OTB. SITS creation could also be done in Python, but this way was faster and easier.

### Pre-processing

#### SITS creation

Follows an example for using OTB applications and Shell Programming for creating SITS from NDVI indices for year 2012.

The file with the images was in the following format:

```
SudouestKalideos_20120112_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120218_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120307_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120327_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120503_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120620_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120707_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120717_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120810_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20120822_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20121101_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20121215_MS_fmsat_ortho_surf_pente_8m.tif
SudouestKalideos_20121231_MS_fmsat_ortho_surf_pente_8m.tif
```

The script that was used to create SITS from NDVI indices is [script\\_sits.sh](#) and is given as:

```
# Creating SITS from NDVI indices, processing year 2012
# Input parameter: $i - the year
# Usage: ./sits_2012.sh

for i in *8m.tif
do
    otbcli_BandMath -il $i -out ndvi_$i -exp "(im1b4-im1b3)/(im1b4+im1b3)"
done
otbcli_ConcatenateImages -il ndvi_*.tif -out sits_NDVI_$i.tif
rm -v ndvi_*.tif
```

The notation `im1b4` refers to the forth band in the images which is for Formosat images near-infrared band whilst `im1b3` refers to red band.

For each image NDVI index was computed using OTB function `otbcli_BandMath`. The command `$i` will return the name of one file from the list. Thus, for loop will process every file and the name of the output will be the name of the original file where `ndvi_` is added at the beginning. Then all the NDVI indices were concatenated into one image time series, in this case, `sits_NDVI_2012.sh`. The OTB function `otbcli_ConcatenateImages` was used for this purpose. And finally, to free the space, all the NDVI indices were removed, since they are in the file `sits_NDVI_2012.sh`. This was done using `rm -v ndvi_*.tif`.



## Defining the temporal axis

After SITS creation, the smoothing was performed. As the first step, temporal axis needed to be built.

Temporal axis was defined using the function `create_timestamp.py`, which is presented in the following box:

```
'''
    The program builds temporal axis for one year.
    TimeStamp will contain for each image its number in the year.
'''
from datetime import date # Import to manage the dates
import glob # Import to manage the files
import scipy as sp

NAME = glob.glob('*8m.tif')
NAME.sort()
TimeStamp=[]

# Build the temporal sampling
for name in NAME:
    # There are 17 characters before the date, which is written as year-
    month-day
    date_tp = date(int(name[17:21]),int(name[21:23]),int(name[23:25]))
    TimeStamp.append(date_tp.timetuple().tm_yday)
TimeStamp = sp.asarray(TimeStamp)
```

If we consider year 2012 (the format of the images is given in previous example), the temporal axis will be:

TimeStamp = [12, 49, 67, 87, 124, 172, 189, 199, 223, 235, 306, 350, 366].

## Smoothing SITS

The following function, `smooth_image.py`, applies Whittaker smoother filter to SITS.

```
import scipy as sp
from osgeo import gdal

import scipy as sp
from osgeo import gdal

def smooth_image(raster_name,mask_name,output_name,l,t):
    """
    The function applies a smoothing filter on all the pixels of the
    input image.
    Input:
    raster_name: the name of the original SITS
    mask_name: the name of the Mask Time Series. Every pixel with value
    greater than 0 refers to the clouds or cloud shadows presence and the
    corresponding pixel from SITS is masked.
    output_name: the name of the smoothed image
    l: the regularization parameter
    t: the temporal sampling (scipy array)
    """
    # Get
    import smoother as sm

    # Open Raster and get additionnal information
    raster = gdal.Open(raster_name,gdal.GA_ReadOnly)
    if raster is None:
        print 'Impossible to open '+raster_name
        exit()

    # Open Mask and get additionnal information
    mask = gdal.Open(mask_name,gdal.GA_ReadOnly)
    if mask is None:
        print 'Impossible to open '+mask_name
        exit()

    # Check size
    if (raster.RasterXSize != mask.RasterXSize) or (raster.RasterYSize !=
mask.RasterYSize) or (raster.RasterCount != mask.RasterCount):
        print 'Image and mask should be of the same size'
        exit()

    # Get the size of the image
    d = raster.RasterCount
    nc = raster.RasterXSize
    nl = raster.RasterYSize

    # Get the geoinformation
    GeoTransform = raster.GetGeoTransform()
    Projection = raster.GetProjection()

    # Get block size
    band = raster.GetRasterBand(1)
    block_sizes = band.GetBlockSize()
    x_block_size = block_sizes[0]
    y_block_size = block_sizes[1]
    del band
```

```

# Initialize the output
driver = gdal.GetDriverByName('GTiff')
dst_ds = driver.Create(output_name, nc,nl, d, gdal.GDT_Float64)
dst_ds.SetGeoTransform(GeoTransform)
dst_ds.SetProjection(Projection)

for i in xrange(0,nl,y_block_size):
    if i + y_block_size < nl: # Check for size consistency in Y
        lines = y_block_size
    else:
        lines = nl - i
    for j in xrange(0,nc,x_block_size): # Check for size consistency in
X
        if j + x_block_size < nc:
            cols = x_block_size
        else:
            cols = nc - j

        # Get the data
        X = sp.empty((cols*lines,d))
        M = sp.empty((cols*lines,d),dtype='int')
        for ind in xrange(d):
            X[:,ind] = raster.GetRasterBand(int(ind+1)).ReadAsArray(j,
i, cols, lines).reshape(cols*lines)
            M[:,ind] = mask.GetRasterBand(int(ind+1)).ReadAsArray(j, i,
cols, lines).reshape(cols*lines)
            # Put all masked value to 1
            M[sp.isnan(M)]=0
            M[M>0]=1
            X[M>0]=0

            # Do the smoothing
            Xf = sp.empty((cols*lines,d))
            for ind in xrange(cols*lines):
                smoother = sm.Whittaker(x=X[ind,:],t=t,w=1-
M[ind,:],order=2)
                Xf[ind,:] = smoother.smooth(1)

            # Write the data
            for ind in xrange(d):
                out = dst_ds.GetRasterBand(int(ind+1))
                out.WriteArray(Xf[:,ind].reshape(lines,cols),j,i)
                out.FlushCache()

            # Free memory
            del X,Xf,M,out

# Clean/Close variables
raster = None
mask = None
dst_ds = None

```

Before the function for smoothing was applied, it was necessary to find the optimal value for regularization parameter. Follows the function for finding the optimal regularization parameter, `find_optimal_lamda.py`:

```
from osgeo import gdal
import matplotlib.pyplot as plt
import sys
import smoother as sm
import scipy as sp
import random
from scipy.stats import mode
import time

'''
    The program finds an optimal regularization parameter for smoothing.
    Generalized and ordinary cross-validation are calculated for each of
    the pixels which correspond to the forest area for a range values of
    regularization parameter.
    Then for each of the pixels finds lamda, for which the ordinary and
    genealized cross validations have min values.
    For the optimal lamda takes the value which appears most frequently.
    Inputs:
        sits_name: the name of the original SITS.
        cloud_mask_name: the name of Mask Time Series.
        mask_forest_name: the mask containing the information about
        forest/nonforest parts of the images.
    Outputs:
        lamda_gcv: the optimal lamda obtained using Generalized Cross-
        Validation
        lamda_ocv: the optimal lamda obtained using Ordinary Cross-Validation
'''

## Open Raster
sits_name = 'sits_2012.tif'
sits = gdal.Open(sits_name, gdal.GA_ReadOnly)

if sits is None:
    print 'Impossible to open '+sits_name
    exit()

## Open mask with clouds
cloud_mask_name = 'mask_clouds_2012.tif'
cloud_mask = gdal.Open(cloud_mask_name, gdal.GA_ReadOnly)

if cloud_mask is None:
    print 'Impossible to open '+cloud_mask_name
    exit()

## Some tests
if (sits.RasterXSize != cloud_mask.RasterXSize) or (sits.RasterYSize !=
cloud_mask.RasterYSize) or (sits.RasterCount != cloud_mask.RasterCount):
    print 'SITS and mask with clouds should be of the same size'
    exit()

mask_forest_name = 'mask_final_2012.tif'
mask_forest = gdal.Open(mask_forest_name, gdal.GA_ReadOnly)

if mask_forest is None:
    print 'Impossible to open '+mask_forest_name
    exit()
```

```

## Get the number of variables and the size of the images
d = sits.RasterCount
nc = sits.RasterXSize
nl = sits.RasterYSize

rangeX = (0, nc) # The actual range in x values of the raster
rangeY = (0, nl) # The actual range in y values of the raster

l=10.0**sp.arange(0,15,1)

lamda_gcv = []
lamda_ocv = []

mask_forest_array = mask_forest.GetRasterBand(1).ReadAsArray()
t = sp.nonzero(mask_forest_array)
n = t[0].size

gcv = sp.empty((n,len(l)),dtype=float64)
ocv = sp.empty((n,len(l)),dtype=float64)

start_time = time.time()

for i in range(n):
    mask_forest_final =
mask_forest.GetRasterBand(1).ReadAsArray(t[1][i],t[0][i],1, 1)
    if mask_forest_final>0:
        X = sp.empty(d)
        M = sp.empty(d)
        for ind in xrange(d):
            X[ind] = sits.GetRasterBand(int(ind+1)).ReadAsArray(t[1][i],
t[0][i], 1, 1)
            M[ind] =
cloud_mask.GetRasterBand(int(ind+1)).ReadAsArray(t[1][i], t[0][i], 1, 1)

            M[M>0] = 1
            M[sp.isnan(M)]=0

            smoother = sm.Whittaker(x=X,t=TimeStamp,w=1-M,order=2)
            gcv[i,:]= smoother.GeneralizedCrossValidation(l) # Value for each l
            ocv[i,:] = smoother.OrdinaryCrossValidation(l)
            ind1 = gcv[i,:].argmin()
            ind2 = ocv[i,:].argmin()
            lamda_gcv.append(l[ind1])
            lamda_ocv.append(l[ind2])

time_duration = time.time() - start_time

print mode(lamda_gcv)[0][0]
print mode(lamda_ocv)[0][0]

plt.figure()
plt.hist(sp.log10(lamda_gcv),100)
plt.figure()
plt.hist(sp.log10(lamda_ocv),100)
plt.show()

```

## Classification and accuracy assessment

### Extracting training/validation samples

The training/validating samples were extracted from SITS using the following function, `get_samples_from_roi.py`:

```
from osgeo import gdal
import scipy as sp

def get_samples_from_roi(raster_name,roi_name):
    """
    The function get the set of pixels given in the thematic map.
    Data is read per block.
    Input:
        raster_name: the name of the raster file, could be any file that
            GDAL can open
        roi_name: the name of the thematic image: each pixel whose values
            is greater than 0 is returned
    Output:
        X: the sample matrix. A nXd matrix, where n is the number of
        referenced pixels and d is the number of variables. Each
        line of the matrix is a pixel.
        Y: the label of the pixel
    """

    ## Open Raster
    raster = gdal.Open(raster_name,gdal.GA_ReadOnly)
    if raster is None:
        print 'Impossible to open '+raster_name
        exit()

    ## Open ROI
    roi = gdal.Open(roi_name,gdal.GA_ReadOnly)
    if roi is None:
        print 'Impossible to open '+roi_name
        exit()

    ## Some tests
    if (raster.RasterXSize != roi.RasterXSize) or (raster.RasterYSize !=
    roi.RasterYSize):
        print 'Images should be of the same size'
        exit()

    ## Get block size
    band = raster.GetRasterBand(1)
    block_sizes = band.GetBlockSize()
    x_block_size = block_sizes[0]
    y_block_size = block_sizes[1]
    del band

    ## Get the number of variables and the size of the images
    d = raster.RasterCount
    nc = raster.RasterXSize
    nl = raster.RasterYSize

    ## Read block data
```

```

X = sp.array([]).reshape(0,d)
Y = sp.array([]).reshape(0,1)
for i in range(0,nl,y_block_size):
    if i + y_block_size < nl: #Check for size consistency in Y
        lines = y_block_size
    else:
        lines = nl - i
    for j in range(0,nc,x_block_size): #Check for size consistency in X
        if j + x_block_size < nc:
            cols = x_block_size
        else:
            cols = nc - j

        # Load the reference data
        ROI = roi.GetRasterBand(1).ReadAsArray(j, i, cols, lines)
        t = sp.nonzero(ROI)
        if t[0].size > 0:
            Y =
sp.concatenate((Y,ROI[t].reshape((t[0].shape[0],1)).astype('uint8')))
        # Load the Variables
        Xtp = sp.empty((t[0].shape[0],d))
        for k in xrange(d):
            band = raster.GetRasterBand(k+1).ReadAsArray(j, i,
cols, lines)
            Xtp[:,k] = band[t]
        try:
            X = sp.concatenate((X,Xtp))
        except MemoryError:
            print 'Impossible to allocate memory: ROI too big'
            exit()

        # Clean/Close variables
        del Xtp,band
        roi = None # Close the roi file
        raster = None # Close the raster file

return X,Y

```

## Learning the classifier and computing its performance (classification accuracy)

The main program, `script_classif_formosat.py`, for performing classification with described classifiers is given in the following box.

```
from npfs import *
import sys
from osgeo import gdal
import pdb
import get_samples_from_roi as fun
from accuracy_index import *
from sklearn.svm import SVC
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import StratifiedKFold
from script_classif_whole_sits import *

## ADDITIONAL FUNCTIONS
def standardize(x,M=None,S=None,REVERSE=None):
    ''' Function that standardize the data
    Input:
        x: the data
        M: the mean vector
        V: the standard deviation vector
    Output:
        x: the standardize data
        M: the mean vector
        V: the standard deviation vector
    '''
    ...
    if not sp.issubdtype(x.dtype,float):
        do_convert = 1
    else:
        do_convert = 0
    if REVERSE is None:
        if M is None:
            M = sp.mean(x,axis=0)
            S = sp.std(x,axis=0)
            if do_convert:
                xs = (x.astype('float')-M)/S
            else:
                xs = (x-M)/S
            return xs,M,S
        else:
            if do_convert:
                xs = (x.astype('float')-M)/S
            else:
                xs = (x-M)/S
            xs = (x-M)/S
            return xs
    else:
        return S*x+M

## LOAD DATA
rt = ['sits_NDVI', 'sits_bands', 'sits_bands_NDVI']

for raster_type in rt:
    for roilev in range(1,4):
        raster_name = raster_type + '_2012.tif'
```



```

roi_name = 'raster_ref_level' + str(roilev) + '.tif'

## PARAMETERS
REP,r = 50,0
SPLIT = 2.0/3
AC_gmm = list()
KAPPA_gmm = list()
AC_gmm_sffs = list()
KAPPA_gmm_sffs = list()
AC_svm = list()
KAPPA_svm = list()
AC_rf = list()
KAPPA_rf = list()
IDS = list()
IDS_sffs = list()

sig = 2.0**sp.arange(-5,5)
penalty = 10.0**sp.arange(0,5)
param_grid_svm = dict(gamma=sig, C=penalty)

n_estimators=sp.arange(10,500,50)
param_grid_rf = dict(n_estimators=n_estimators)

X,Y = fun.get_samples_from_roi(raster_name,roi_name)
X = standardize(X)[0]
n,d=X.shape
C = int(Y.max())

CONFU_gmm = sp.zeros((REP,C,C))
CONFU_gmm_sffs = sp.zeros((REP,C,C))
CONFU_svm = sp.zeros((REP,C,C))
CONFU_rf = sp.zeros((REP,C,C))

## START
while r < REP:
    print r
    # Random selection of the sample
    x = sp.array([]).reshape(0,d)
    y = sp.array([]).reshape(0,1)
    xt = sp.array([]).reshape(0,d)
    yt = sp.array([]).reshape(0,1)

    sp.random.seed(r)
    for i in range(C):
        t = sp.where((i+1)==Y)[0]
        nc = t.size
        ns = int(nc*SPLIT)
        rp = sp.random.permutation(nc)
        x = sp.concatenate((X[t[rp[0:ns]],:],x))
        xt = sp.concatenate((X[t[rp[ns:]],:],xt))
        y = sp.concatenate((Y[t[rp[0:ns]]],y))
        yt = sp.concatenate((Y[t[rp[ns:]]],yt))

    y.shape=(y.size,)
    cv = KFold(y.size, n_folds=5)

    ## GMM
    # Learn the model
    model = GMM()
    model.learn_gmm(x,y)

```

```

        model.ids,loo =
model.forward_selection(x,y,maxvar=15,delta=0.001,v=5)
        IDS.append(model.ids)

        # Predict
        yp = model.predict_gmm(xt,ids=model.ids)[0]

        # Confusion matrix
        confu = CONFUSION_MATRIX()
        confu.compute_confusion_matrix(yp,yt)
        AC_gmm.append(confu.OA)
        KAPPA_gmm.append(confu.Kappa)
        CONFU_gmm[r,:,:]=confu.confusion_matrix

        # GMM_sffs
        # Learn the model
        model_sffs = GMM()
        model_sffs.learn_gmm(x,y)
        model_sffs.ids,loo_sffs =
model_sffs.forward_selection_sffs(x,y,maxvar=15,v=5)

        t = sp.argmax(loo_sffs)

        model_sffs.ids = model_sffs.ids[0:(t+1)]
        IDS_sffs.append(model_sffs.ids)

        # Predict
        yp = model_sffs.predict_gmm(xt,ids=model_sffs.ids)[0]

        # Confusion matrix
        confu = CONFUSION_MATRIX()
        confu.compute_confusion_matrix(yp,yt)
        AC_gmm_sffs.append(confu.OA)
        KAPPA_gmm_sffs.append(confu.Kappa)
        CONFU_gmm_sffs[r,:,:]=confu.confusion_matrix

        # SVM
        grid = GridSearchCV(SVC(), param_grid=param_grid_svm, cv=cv)
        grid.fit(x,y)
        clf = grid.best_estimator_
        clf.fit(x,y)
        yp = clf.predict(xt).reshape(yt.shape)
        confu = CONFUSION_MATRIX()
        confu.compute_confusion_matrix(yp,yt)
        AC_svm.append(confu.OA)
        KAPPA_svm.append(confu.Kappa)
        CONFU_svm[r,:,:]=confu.confusion_matrix

        ## RF
        grid = GridSearchCV(RandomForestClassifier(),
param_grid=param_grid_rf, cv=cv)
        grid.fit(x, y)
        clf = grid.best_estimator_
        clf.fit(x,y)
        yp = clf.predict(xt).reshape(yt.shape)
        confu = CONFUSION_MATRIX()
        confu.compute_confusion_matrix(yp,yt)
        AC_rf.append(confu.OA)

```

```

KAPPA_rf.append(confu.Kappa)
CONFU_rf[r,:,:]=confu.confusion_matrix

    del x,xt,y,yt,model,model_sffs,confu,nc,ns,t
    r+=1

# SAVE RESULTS
sp.savez('res_gmm_2012_lev' + str(roilev) + raster_type
+'_2012',OA=AC_gmm,Kappa=KAPPA_gmm,confu=CONFU_gmm,ids=IDS)
sp.savez('res_gmm_sffs_2012_lev' + str(roilev) + raster_type
+'_2012',OA=AC_gmm_sffs,Kappa=KAPPA_gmm_sffs,confu=CONFU_gmm_sffs,ids=IDS_s
ffs)
sp.savez('res_svm_2012_lev' + str(roilev) + raster_type
+'_2012',OA=AC_svm,Kappa=KAPPA_svm,confu=CONFU_svm)
sp.savez('res_rf_2012_lev' + str(roilev) + raster_type
+'_2012',OA=AC_rf,Kappa=KAPPA_rf,confu=CONFU_rf)

```

## Prediction on all the pixels of the SITS to produce thematic maps

To produce the classification maps, the following function ([predict\\_image.py](#)) has been implemented:

```

from osgeo import gdal
import scipy as sp
import get_samples_from_roi as fun
from accuracy_index import *
from sklearn.svm import SVC
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from npfs import *

def
predict_image(raster_name,roi_name,classif_name,classifier_name,mask_name=N
one):
    """
        The function classifies the whole raster images, using block per
        block image analysis.
        Inputs:
        raster_name: the name of the raster image that should be
        classified, could be any file that GDAL can open
        roi_name: the name of the image which has referenced pixels
        classif_name: the name of the output image (which will contain
        zeros where there is no data for classification and number of the class for
        each pixel)
        classifier_name: the name of the classifier that is used for
        classification of the raster images. It can be 'NPFS','SVM' and 'RF'
        mask_name: the name of the mask which is equal to one for the
        pixels that should be classified. This mask is intersection between mask
        with the non-forest areas and
        mask which is equal to one if there are information for all the
        dates in the raster image .
    """

```

The pixels which for some dates have an information and for some dates do not have any information will not be taken into account (the value of the mask for these pixels is zero)

Each classifier is learned with all the referenced pixels and prediction is done with raster pixels (selected with respect to the mask)

```
'''
# Parameters
block_sizes = 512
sig = 2.0**sp.arange(-5,5)
penalty = 10.0**sp.arange(0,5)
param_grid_svm = dict(gamma=sig, C=penalty)

n_estimators=sp.arange(10,500,50)
param_grid_rf = dict(n_estimators=n_estimators)

# Open Raster and get additionnal information
raster = gdal.Open(raster_name,gdal.GA_ReadOnly)
if raster is None:
    print 'Impossible to open '+raster_name
    exit()

roi = gdal.Open(roi_name,gdal.GA_ReadOnly)
if roi is None:
    print 'Impossible to open '+roi_name
    exit()

# If provided, open mask
if mask_name is None:
    mask=None
else:
    mask = gdal.Open(mask_name,gdal.GA_ReadOnly)
    if mask is None:
        print 'Impossible to open '+mask_name
        exit()

# Check size
if (raster.RasterXSize != mask.RasterXSize) or (raster.RasterYSize !=
mask.RasterYSize):
    print 'Image and mask should be of the same size'
    exit()

X,Y = fun.get_samples_from_roi(raster_name,roi_name)
# X = standardize(X)[0]
X,M,S = standardize(X) ## Changed by MF

# Get the size of the image
d = raster.RasterCount
ncols= raster.RasterXSize
nlines = raster.RasterYSize
C = int(Y.max())

# TO DO: update capital and small X,Y

Y.shape=(Y.size,)
cv = KFold(Y.size, n_folds=5)

# Get the geoinformation
GeoTransform = raster.GetGeoTransform()
Projection = raster.GetProjection()

# Set the block size
```

```

x_block_size = block_sizes
y_block_size = block_sizes

## Initialize the output
driver = gdal.GetDriverByName('GTiff')
dst_ds = driver.Create(classif_name, ncols,nlines, 1, gdal.GDT_UInt16)
dst_ds.SetGeoTransform(GeoTransform)
dst_ds.SetProjection(Projection)
out = dst_ds.GetRasterBand(1)

if classifier_name is 'NPFS':
    # Learn the model
    model = GMM()
    model.learn_gmm(X,Y)
    model.ids,loo =
model.forward_selection(X,Y,maxvar=10,delta=0.05,v=5)
    nvar = len(model.ids)

elif classifier_name is 'SVM':
    # Learn the model
    grid = GridSearchCV(SVC(), param_grid=param_grid_svm, cv=cv)
    grid.fit(X,Y)
    clf = grid.best_estimator_
    clf.fit(X,Y)

else:
    # Learn the model
    grid = GridSearchCV(RandomForestClassifier(),
param_grid=param_grid_rf, cv=cv)
    grid.fit(X,Y)
    clf = grid.best_estimator_
    clf.fit(X,Y)

## Perform the classification
for i in range(0,nlines,y_block_size):
    if i + y_block_size < nlines: # Check for size consistency in Y
        lines = y_block_size
    else:
        lines = nlines - i
    for j in range(0,ncols,x_block_size): # Check for size consistency
in X
        if j + x_block_size < ncols:
            cols = x_block_size
        else:
            cols = ncols - j

    # Load the data
    Xb = sp.empty((cols*lines,d))
    for ind in range(d):
        Xb[:,ind] = raster.GetRasterBand(ind+1).ReadAsArray(j, i,
cols, lines).reshape(cols*lines)
        Xb = standardize(Xb,M,S)    ## Update Xb in the following

    # Do the prediction
    if classifier_name is 'NPFS':
        if mask is None:
            yp = model.predict_gmm(Xb)[0].astype('uint16')
        else:
            mask_temp=mask.GetRasterBand(1).ReadAsArray(j, i, cols,
lines).reshape(cols*lines)
            t= sp.where(mask_temp>0)[0]

```

```

        yp=sp.zeros((cols*lines,))
        if t.size>0:
            yp[t]=
model.predict_gmm(Xb[t,:],ids=model.ids)[0].astype('uint16')

    elif classifier_name is 'SVM':

        # Do the prediction
        if mask is None:
            yp =clf.predict(Xb)
        else:
            mask_temp=mask.GetRasterBand(1).ReadAsArray(j, i, cols,
lines).reshape(cols*lines)
            t= sp.where(mask_temp>0)[0]
            yp=sp.zeros((cols*lines,))
            if t.size>0:
                yp[t] = clf.predict(Xb[t,:]).astype('uint16')

        else:

            # Do the prediction
            if mask is None:
                yp =clf.predict(Xb)
            else:
                mask_temp=mask.GetRasterBand(1).ReadAsArray(j, i,
cols, lines).reshape(cols*lines)
                t= sp.where(mask_temp>0)[0]
                yp=sp.zeros((cols*lines,))
                if t.size>0:
                    yp[t] = clf.predict(Xb[t,:]).astype('uint16')

        out.WriteArray(yp.reshape(lines,cols),j,i)
        out.FlushCache()
        del Xb,yp

# Clean/Close variables
raster = None
dst_ds = None

```

The following program, ([thematic\\_map\\_consecutive\\_years.py](#)), was used to find the most frequent species and its frequency for each pixel looking at the classification maps obtained for four years.

```

from osgeo import gdal
import scipy as sp
from scipy.stats import mode

'''
    The function finds the most frequent species and its frequency within
the raster image.
    Inputs:
        raster_name: the name of the raster image which consists of
classification maps from several consecutive years from the same
classifier, the same level of classification and the same way of SITS
creation. Each band in the raster image corresponds to the classificatin
map from one year.
'''

```

mask\_name: the name of the mask which is equal to one for the pixels that are classified in each of the classification maps.

Output:

classif\_name: the name of the output image. It consists of two bands. First band presents the most frequent class for each pixel and its frequency can be found in the second band.

```
'''
raster_name =
'/home/veliborka/Documents/Bands_concatenation/classification_maps_all_years_levell_SB_SVM.tif'
mask_name =
'/home/veliborka/Documents/Bands_concatenation/mask_final_all_years.tif'
classif_name='/home/veliborka/Documents/Bands_concatenation/final_classification_map_levell_SVM.tif'

# Open Raster
raster = gdal.Open(raster_name,gdal.GA_ReadOnly)
if raster is None:
    print 'Impossible to open '+raster_name
    exit()

# Open forest/nonforest mask
mask = gdal.Open(mask_name,gdal.GA_ReadOnly)
if mask is None:
    print 'Impossible to open '+mask_name
    exit()

# Check size
if (raster.RasterXSize != mask.RasterXSize) or (raster.RasterYSize != mask.RasterYSize):
    print 'Image and mask should be of the same size'
    exit()

# Get the size of the image
d = raster.RasterCount
ncols= raster.RasterXSize
nlines = raster.RasterYSize

block_sizes = 512

# Get the geoinformation
GeoTransform = raster.GetGeoTransform()
Projection = raster.GetProjection()

# Set the block size
x_block_size = block_sizes
y_block_size = block_sizes

## Initialize the output
driver = gdal.GetDriverByName('GTiff')
dst_ds = driver.Create(classif_name, ncols,nlines, 2, gdal.GDT_UInt16)
dst_ds.SetGeoTransform(GeoTransform)
dst_ds.SetProjection(Projection)

## Find the most frequent appearing species and its frequency for each pixel
for i in range(0,nlines,y_block_size):
    if i + y_block_size < nlines: # Check for size consistency in Y
        lines = y_block_size
```

```

else:
    lines = nlines - i
for j in range(0,ncols,x_block_size): # Check for size consistency in X
    if j + x_block_size < ncols:
        cols = x_block_size
    else:
        cols = ncols - j

    # Load the data
    Xb = sp.empty((cols*lines,d))
    for ind in range(d):
        Xb[:,ind] = raster.GetRasterBand(ind+1).ReadAsArray(j, i, cols,
lines).reshape(cols*lines)

    # Do the prediction
    if mask is None:
        yp = mode(Xb,axis=1)
    else:
        mask_temp=mask.GetRasterBand(1).ReadAsArray(j, i, cols,
lines).reshape(cols*lines)
        t= sp.where(mask_temp>0)[0]
        yp=sp.zeros((cols*lines,))
        fr=sp.zeros((cols*lines,))
        if t.size>0:
            yp[t] = mode(Xb[t,:],axis=1)[0] # the most frequent
appearing species
            fr[t] = mode(Xb[t,:],axis=1)[1] # its frequency

        out = dst_ds.GetRasterBand(1)
        out.WriteArray(yp.reshape(lines,cols),j,i)
        out.FlushCache()
        out = dst_ds.GetRasterBand(2)
        out.WriteArray(fr.reshape(lines,cols),j,i)
        out.FlushCache()
    del Xb,yp,fr

# Clean/Close variables
raster = None
dst_ds = None

```

The following box presents the main part of the implemented SFFS algorithm ([forward\\_selection\\_sffs.py](#)):

```

def forward_selection_sffs(self,x,y,delta=0.1,maxvar=None,v=5,ncpus=None):
    """ Function that selects the most discriminative variables according
to a sequential floating forward search method (SFFS)
Inputs:
    x,y: the training samples and their labels
    delta: the minimal improvement in percentage when a variable
is added to the pool, the algorithm stops if the improvement is
lower than delta. Default value 0.1%
    maxvar: maximum number of extracted variables. Default value:
20% of the original number of features
    v: number of folds for the cross-validation.
Outputs:
    ids: the selected subset of features for SFFS method
    OA: the accuracy estimated for each subset ids by v-fold cv for
SFFS method

```



```

"""
## Get some information from the variable
C = int(y.max(0)); # Number of classes
n = x.shape[0]    # Number of samples
d = x.shape[1]    # Number of variables
if ncpus is None:
    ncpus=mp.cpu_count()# Get the number of core

## Initialization
r=0                # Initialization of the counter
r1=0
variable = sp.arange(d) # At step zero: d variables available
variable1 = sp.arange(d)
ids=[]            # and no selected variable
ids1=[]
OA=[]            # list of the evolution the OA estimation
OA1=[]
if maxvar is None:
    maxvar = sp.floor(d/5) # Select at max 20 % of the original number
of variables

    cv=CV()                # Initialize the CV sets
    cv.split_data_class(y,v=5) # Generate split indices
for the data

## Pre-update the models
model_pre_cv = []
for i in range(v):
    model_pre_cv.append(GMM(size=C,d=d))# List of updated GMM models
    X,Y=x[cv.iT[i],:], y[cv.iT[i]]
    nu = float(Y.size)
    for j in range(C): #Update the model for each
class
        k = sp.where(Y==(j+1))[0]
        nu_c = float(k.size)
        mean_t = sp.mean(X[k,:],axis=0)
        cov_t = sp.cov(X[k:],bias=1,rowvar=0)

        model_pre_cv[i].ni[j] = self.ni[j]-nu_c
        model_pre_cv[i].prop[j]= model_pre_cv[i].ni[j]/(n-nu)
        model_pre_cv[i].mean[j,:] = (self.ni[j]*self.mean[j,]-
nu_c*mean_t)/(self.ni[j]-nu_c)
        model_pre_cv[i].cov[j,:] = (self.ni[j]*self.cov[j,:,:] -
nu_c*cov_t - nu_c*self.ni[j]/model_pre_cv[i].ni[j]*sp.outer(self.mean[j,]-
mean_t,self.mean[j,]-mean_t))/model_pre_cv[i].ni[j]
        del k,nu_c,mean_t,cov_t
    del X,Y,nu

## Start the sequential forward floating search
while (r<maxvar and len(variable)>0):
    # Step 1, SFFS inclusion the most significant feature,
initialization with two the most significant features
    err = sp.zeros(variable.size)
    pool = mp.Pool(processes=ncpus)
    processes = [pool.apply_async(compute_v_cv_gmm,
args=(variable,model_pre_cv[i],x[cv.iT[i],:],y[cv.iT[i]],ids)) for i in
xrange(v)]
    pool.close()
    pool.join()
    for p in processes:
        err += p.get()

```

```

err /= v
del processes,pool
## Select the variable that provides the highest loocv
t = sp.argmax(err)
OA.append(err[t])
ids.append(variable[t])
variable = sp.delete(variable,t)
r += 1

# Step 2: Conditional Exclusion
if r>2:
    # The least significant feature in the subset ids
    worst_feat, worst_feat_val, crit_func_max =
find_worst_feature(model_pre_cv,x,y,cv,ids,OA,v)
    if worst_feat_val != None: # and worst_feat != (r-1):
        variable = sp.append(variable,ids[worst_feat])
        del ids[worst_feat]
        r -= 1
        condition = True # Go to step 3 if r is higher than 2
        if r == 2:
            OA[r-1] = crit_func_max
            del OA[r]
        else:
            condition = False # If worst feature value doesn't exist
got to step 1

# Step 3
while (r>2 and condition):
    # The least significant feature in the subset ids
    worst_feat1, worst_feat_val1, crit_func_max1 =
find_worst_feature(model_pre_cv,x,y,cv,ids,OA,v,r)
    if worst_feat_val1 != None:
        variable = sp.append(variable,ids[worst_feat1])
        del ids[worst_feat1]
        r -= 1
        crit_func_max = crit_func_max1
        if r == 2:
            OA[r-1] = crit_func_max
            del OA[r:]
        else:
            OA[r-1] = crit_func_max
            del OA[r:]
        condition = False # If worst feature value doesn't exist
got to step 1

## Return the final values
return ids,OA

```

The following function, `find_worst_feature.py`, is used within the previous SFFS algorithm to find the worst feature from the set of original features.

```

def find_worst_feature(model_cv,x,y,cv,ids,oa,v,ncpus=None):
    """ Function that finds the least significant feature in the subset of
    features ids.
    Inputs:
        model_cv: contains the models built with all the features
        x,y: the training samples and their labels (out of all
        referenced pixels)

```

```

        ids: the subset of the features
        cv: contains training samples indices for training and testing
            each model, cv.iT contains indices for testing each model
        oa: the accuracy estimated for each subset ids by v-fold cv
        v: number of folds for the cross-validation
    Output:
        worst_feat: the indice of the worst feature in the subset ids
        worst_feat_val: the value of the worst feature
        crit_func_max: the new overall accuracy estimated for ids
            without worst feature, if worst feature exists, otherwise it
            will not be changed

    """
    Used in GMM.forward_selection_sffs()
    """
    r = len(ids)

    if ncpus is None:
        ncpus=mp.cpu_count()    # Get the number of core

    worst_feat_val = None
    worst_feat = None
    crit_func_eval = []

    for i in range(0,len(ids)):
        ids_i = sp.delete(ids,i)
        pool = mp.Pool(processes=ncpus)
        processes = [pool.apply_async(compute_v_cv_gmm_sffs,
args=(model_cv[j],x[cv.iT[j],:],y[cv.iT[j]],ids_i)) for j in
 xrange(v)]
        pool.close()
        pool.join()

        crit_func_eval_i = 0
        for p in processes:
            crit_func_eval_i += p.get()
        crit_func_eval_i /= v
        crit_func_eval.append(crit_func_eval_i)
        del processes,pool

    crit_func_eval = sp.asarray(crit_func_eval)
    t1 = sp.argmax(crit_func_eval)    # get the indice of the
maximum of loocv
    crit_func_eval_max = crit_func_eval[t1]    # add the value
to loo

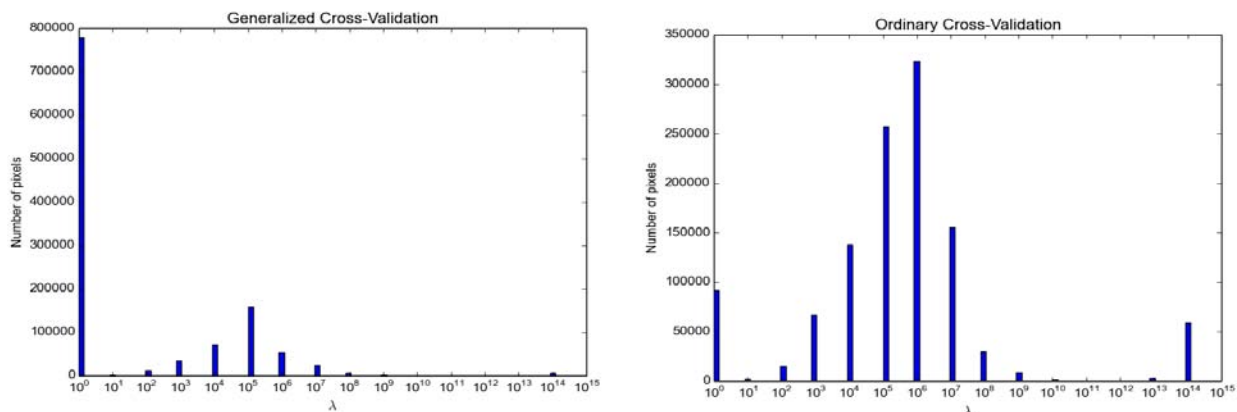
    crit_func_max = oa[r-2]
    if crit_func_eval_max > crit_func_max:
        worst_feat, crit_func_max = t1, crit_func_eval_max
        worst_feat_val = ids[worst_feat]

    return worst_feat, worst_feat_val, crit_func_max

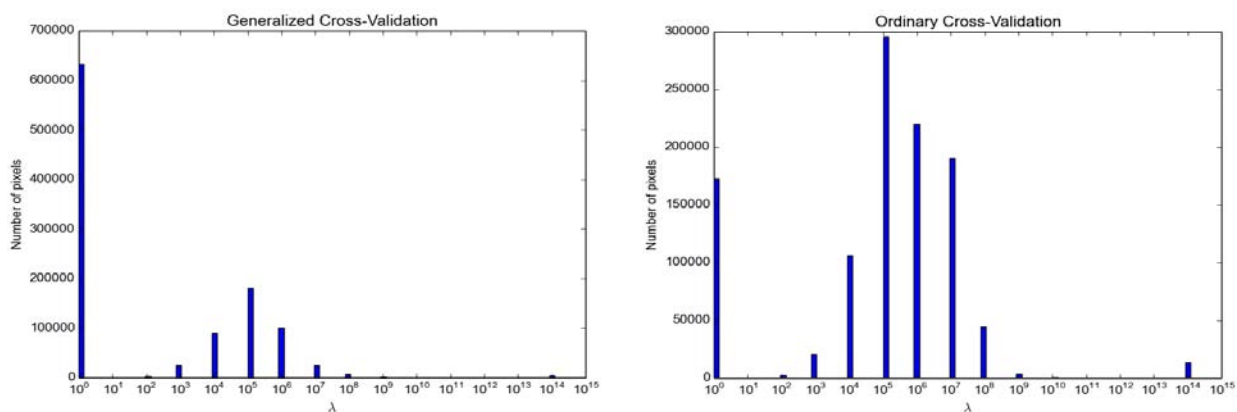
```

## Appendix 3

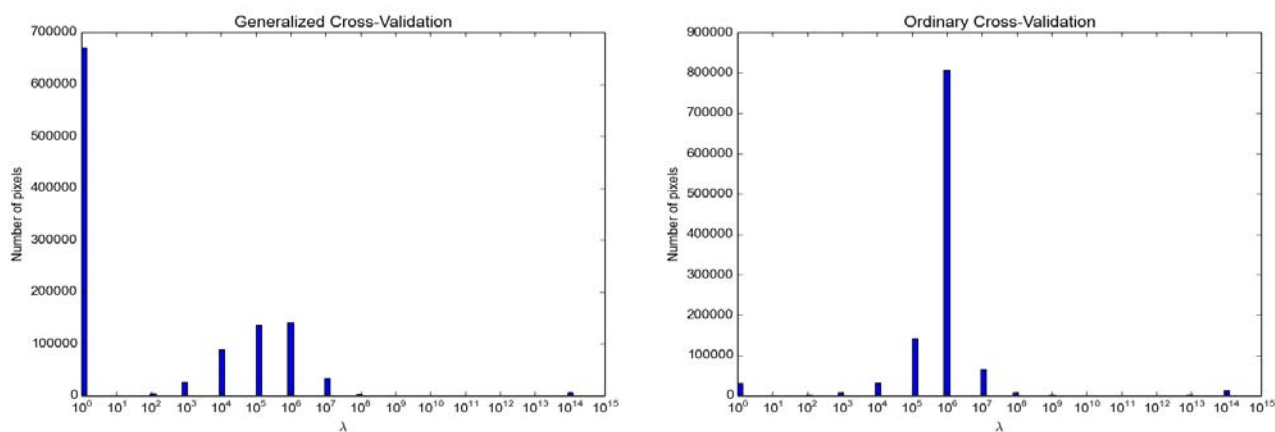
The results for years 2011, 2012 and 2014 after applying function `find_optimal_lambda.py` are presented in *Figures A3.1, A3.2 and A3.3*:



**Figure A3.1. Histograms for Generalized and Ordinary Cross-Validation errors for different values of  $\lambda$  for year 2011**



**Figure A3.2. Histograms for Generalized and Ordinary Cross-Validation errors for different values of  $\lambda$  for year 2012**



**Figure A3.3. Histograms for Generalized and Ordinary Cross-Validation errors for different values of  $\lambda$  for year 2014**

The classification results obtained for years 2011, 2012 and 2014 are presented in *Tables A3.1, A3.2 and A3.3.*

Level 1 (classification based on 2 classes)						
	NDVI (12 features)		SB (48 features)		SB+NDVI (60 features)	
	OA (%)	Kappa (%)	OA (%)	Kappa (%)	OA (%)	Kappa (%)
NPFS_sfs	97.7 ± 0.01	94.5 ± 0.05	97.3 ± 0.01	93.7 ± 0.05	98.0 ± 0.01	95.2 ± 0.05
NPFS_sffs	97.7 ± 0.01	94.6 ± 0.03	97.8 ± 0.01	94.8 ± 0.04	98.4 ± 0.01	96.2 ± 0.03
RF	98.6 ± 0.00	96.6 ± 0.02	97.6 ± 0.01	94.3 ± 0.04	97.9 ± 0.01	95.0 ± 0.05
SVM	<b>98.8 ± 0.00</b>	<b>97.1 ± 0.02</b>	<b>99.5 ± 0.00</b>	<b>98.7 ± 0.01</b>	<b>99.6 ± 0.00</b>	<b>99.0 ± 0.01</b>
Level 2 (classification based on 4 classes)						
	NDVI (12 features)		SB (48 features)		SB+NDVI (60 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	95.1 ± 0.01	91.5 ± 0.04	96.3 ± 0.01	93.7 ± 0.04	96.3 ± 0.02	93.7 ± 0.05
NPFS_sffs	95.4 ± 0.01	92.1 ± 0.04	96.5 ± 0.01	94.0 ± 0.03	96.5 ± 0.01	94.0 ± 0.02
RF	96.5 ± 0.01	94.0 ± 0.03	95.7 ± 0.01	92.6 ± 0.04	96.8 ± 0.01	94.5 ± 0.04
SVM	<b>97.0 ± 0.01</b>	<b>94.8 ± 0.03</b>	<b>97.6 ± 0.01</b>	<b>95.9 ± 0.03</b>	<b>97.9 ± 0.01</b>	<b>96.4 ± 0.03</b>
Level 3 (classification based on 14 classes)						
	NDVI (12 features)		SB (48 features)		SB+NDVI (60 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	84.8 ± 0.05	83.1 ± 0.06	90.4 ± 0.04	89.4 ± 0.05	90.3 ± 0.03	89.2 ± 0.04
NPFS_sffs	85.6 ± 0.05	83.9 ± 0.06	90.3 ± 0.04	89.2 ± 0.06	90.4 ± 0.03	89.3 ± 0.04
RF	86.4 ± 0.03	84.9 ± 0.04	90.9 ± 0.03	89.9 ± 0.03	91.6 ± 0.02	90.7 ± 0.02
SVM	<b>89.7 ± 0.03</b>	<b>88.6 ± 0.03</b>	<b>95.3 ± 0.01</b>	<b>94.8 ± 0.01</b>	<b>96.2 ± 0.01</b>	<b>95.8 ± 0.01</b>

**Table A3.1.** Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2011. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.

Level 1 (classification based on 2 classes)						
	NDVI (13 features)		SB (52 features)		SB+NDVI (65 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	97.0 ± 0.01	92.9 ± 0.04	98.3 ± 0.01	95.9 ± 0.03	98.4 ± 0.01	96.1 ± 0.03
NPFS_sffs	96.8 ± 0.01	92.2 ± 0.06	98.6 ± 0.00	96.7 ± 0.02	98.8 ± 0.00	97.1 ± 0.02
RF	<b>99.4 ± 0.00</b>	<b>98.5 ± 0.01</b>	99.3 ± 0.00	98.4 ± 0.01	99.4 ± 0.00	98.6 ± 0.01
SVM	99.3 ± 0.00	98.3 ± 0.01	<b>99.7 ± 0.00</b>	<b>99.2 ± 0.00</b>	<b>99.6 ± 0.00</b>	<b>99.1 ± 0.00</b>
Level 2 (classification based on 4 classes)						
	NDVI (13 features)		SB (52 features)		SB+NDVI (65 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	95.8 ± 0.01	92.8 ± 0.03	97.1 ± 0.01	94.9 ± 0.02	97.4 ± 0.01	95.5 ± 0.03
NPFS_sffs	95.7 ± 0.01	92.5 ± 0.03	97.2 ± 0.01	95.2 ± 0.02	97.8 ± 0.01	96.2 ± 0.02
RF	<b>97.8 ± 0.01</b>	<b>96.2 ± 0.02</b>	98.1 ± 0.00	96.8 ± 0.02	98.5 ± 0.01	97.4 ± 0.02
SVM	97.6 ± 0.01	95.9 ± 0.02	<b>99.0 ± 0.00</b>	<b>98.2 ± 0.01</b>	<b>98.9 ± 0.00</b>	<b>98.1 ± 0.01</b>
Level 3 (classification based on 14 classes)						
	NDVI (13 features)		SB (52 features)		SB+NDVI (65 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	87.4 ± 0.03	86.0 ± 0.03	91.4 ± 0.03	90.5 ± 0.03	91.4 ± 0.03	90.4 ± 0.03
NPFS_sffs	86.6 ± 0.03	85.1 ± 0.03	91.8 ± 0.03	90.8 ± 0.04	92.1 ± 0.04	91.3 ± 0.04
RF	89.3 ± 0.02	88.1 ± 0.03	94.4 ± 0.04	93.7 ± 0.05	94.8 ± 0.02	94.2 ± 0.03
SVM	<b>90.2 ± 0.02</b>	<b>89.1 ± 0.02</b>	<b>97.9 ± 0.01</b>	<b>97.6 ± 0.01</b>	<b>98.0 ± 0.01</b>	<b>97.7 ± 0.01</b>

**Table A3.2. Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2012. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.**

Level 1 (classification based on 2 classes)						
	NDVI (15 features)		SB (60 features)		SB+NDVI (75 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	94.5 ± 0.02	86.8 ± 0.12	97.7 ± 0.01	94.6 ± 0.05	97.8 ± 0.01	94.9 ± 0.05
NPFS_sffs	94.0 ± 0.03	85.3 ± 0.19	98.2 ± 0.01	95.6 ± 0.03	98.2 ± 0.01	95.6 ± 0.03
RF	97.8 ± 0.01	94.8 ± 0.03	98.6 ± 0.00	96.6 ± 0.03	98.6 ± 0.00	96.8 ± 0.02
SVM	<b>98.5 ± 0.00</b>	<b>96.4 ± 0.02</b>	<b>99.5 ± 0.00</b>	<b>98.8 ± 0.01</b>	<b>99.6 ± 0.00</b>	<b>99.1 ± 0.01</b>
Level 2 (classification based on 4 classes)						
	NDVI (15 features)		SB (60 features)		SB+NDVI (75 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	91.7 ± 0.02	85.6 ± 0.06	96.6 ± 0.01	94.2 ± 0.04	97.0 ± 0.01	94.9 ± 0.04
NPFS_sffs	91.2 ± 0.02	84.8 ± 0.07	97.4 ± 0.01	95.5 ± 0.02	97.2 ± 0.01	95.1 ± 0.02
RF	95.4 ± 0.01	92.1 ± 0.04	97.5 ± 0.01	95.7 ± 0.02	97.7 ± 0.01	96.0 ± 0.02
SVM	<b>97.2 ± 0.01</b>	<b>95.2 ± 0.02</b>	<b>98.8 ± 0.00</b>	<b>98.0 ± 0.01</b>	<b>98.7 ± 0.00</b>	<b>97.8 ± 0.01</b>
Level 3 (classification based on 14 classes)						
	NDVI (15 features)		SB (60 features)		SB+NDVI (75 features)	
	OA (%)	Kappa	OA (%)	Kappa	OA (%)	Kappa
NPFS_sfs	89.8 ± 0.02	88.6 ± 0.03	94.7 ± 0.01	94.1 ± 0.02	94.8 ± 0.01	94.2 ± 0.02
NPFS_sffs	89.1 ± 0.02	87.9 ± 0.03	94.6 ± 0.02	94.0 ± 0.03	94.7 ± 0.01	94.1 ± 0.01
RF	91.9 ± 0.01	91.0 ± 0.01	95.9 ± 0.01	95.5 ± 0.02	95.8 ± 0.01	95.3 ± 0.01
SVM	<b>94.7 ± 0.01</b>	<b>94.1 ± 0.01</b>	<b>98.6 ± 0.00</b>	<b>98.4 ± 0.00</b>	<b>98.8 ± 0.00</b>	<b>98.6 ± 0.00</b>

**Table A3.3. Overall accuracy and Kappa statistics for three defined levels of classification using NDVI indices, spectral bands (SB) and NDVI indices and spectral bands together for the year 2014. NPFS\_sfs denotes NPFS GMM based classifier computed with a set of features that were chosen using standard feature selection algorithm described in Section 2.3.1. Similarly, NPFS\_sffs denotes NPFS GMM based classifier computed with a set of features chosen using SFFS algorithm described in Section 2.3.2. The results correspond to the mean value and variance of the overall accuracy (OA) and Kappa statistics over the 50 repetitions in percentages. The best results for each level are reported in bold face.**

The confusion matrices obtained for year 2011 are presented in *Tables A3.4-A3.9*.

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>94.02</b>	5.98
<b>Broadleaf</b>	1.20	<b>98.80</b>

**Table A3.4.** Confusion matrix for level 1, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>98.84</b>	1.16
<b>Broadleaf</b>	0.26	<b>99.74</b>

**Table A3.5.** Confusion matrix for level 1, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.55</b>	0.02	0.38	0.05
<b>Evergreen</b>	0.24	<b>98.24</b>	1.47	0.06
<b>Pine</b>	1.59	0.14	<b>92.41</b>	5.86
<b>Other conifer</b>	1.62	0.00	14.65	<b>83.73</b>

**Table A3.6.** Confusion matrix for level 2, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.95</b>	0.00	0.05	0.00
<b>Evergreen</b>	0.65	<b>98.71</b>	0.65	0.00
<b>Pine</b>	1.28	0.00	<b>95.54</b>	3.19
<b>Other conifer</b>	2.81	0.00	9.68	<b>87.51</b>

**Table A3.7.** Confusion matrix for level 2, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.



<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	<b>93.84</b>	0.80	1.44	<b>3.68</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.08
Oak	0.00	<b>94.52</b>	0.48	0.14	<b>2.71</b>	0.43	0.00	0.95	0.43	0.05	0.00	0.14	0.14	0.00
Black pine	4.14	5.00	<b>76.71</b>	<b>6.00</b>	0.00	0.43	4.86	0.14	1.00	0.00	0.14	1.43	0.00	0.14
Douglas fir	<b>23.33</b>	3.17	3.33	<b>45.5</b>	0.00	0.00	5.67	0.00	0.00	0.00	0.17	17.17	0.00	1.67
Silver birch	0.00	<b>2.64</b>	0.00	0.00	<b>96.08</b>	0.88	0.00	0.08	0.32	0.00	0.00	0.00	0.00	0.00
European ash	0.00	<b>14.86</b>	0.00	0.00	0.86	<b>79.29</b>	0.00	0.43	4.29	0.29	0.00	0.00	0.00	0.00
Maritime pine	0.00	0.00	4.86	6.14	0.00	0.00	<b>77.14</b>	0.14	0.00	0.00	0.00	<b>8.14</b>	0.00	3.57
Black locust	0.00	<b>30.67</b>	0.17	0.00	0.00	0.50	0.00	<b>65.33</b>	3.33	0.00	0.00	0.00	0.00	0.00
Aspen	0.00	<b>2.26</b>	0.00	0.00	0.37	0.00	0.00	0.11	<b>97.26</b>	0.00	0.00	0.00	0.00	0.00
Red oak	0.00	<b>3.00</b>	0.00	0.00	0.18	0.00	0.00	0.00	0.00	<b>96.82</b>	0.00	0.00	0.00	0.00
Eucalyptus	0.00	0.06	0.00	0.24	0.00	0.00	0.00	0.00	0.06	0.00	<b>99.00</b>	0.06	0.06	<b>0.53</b>
Corsican pine	0.86	1.71	1.71	<b>18.43</b>	0.00	0.00	2.71	0.00	0.29	0.00	0.71	<b>70.14</b>	0.00	3.43
Willow	0.00	1.38	0.00	0.00	0.00	0.00	0.00	0.00	<b>2.46</b>	0.00	0.00	0.00	<b>95.85</b>	0.31
Austrian black pine	0.00	0.00	0.00	0.21	0.00	0.00	<b>1.24</b>	0.00	0.00	0.00	0.48	0.62	0.00	<b>97.45</b>

Table A3.8. Confusion matrix for level 3, year 2011. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
<b>Silver fir</b>	<b>96.48</b>	0.40	0.80	<b>2.32</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Oak</b>	0.00	<b>96.33</b>	0.00	0.00	<b>2.48</b>	0.38	0.00	0.76	0.00	0.00	0.00	0.05	0.00	0.00
<b>Black pine</b>	0.00	<b>4.71</b>	<b>89.57</b>	1.43	0.00	0.29	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Douglas fir</b>	<b>18.00</b>	1.83	0.33	<b>55.17</b>	0.00	0.00	9.00	0.00	0.00	0.00	0.00	15.67	0.00	0.00
<b>Silver birch</b>	0.00	<b>1.60</b>	0.00	0.00	<b>98.40</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>European ash</b>	0.00	<b>4.57</b>	0.00	0.00	0.86	<b>94.57</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Maritime pine</b>	0.00	0.00	<b>5.29</b>	1.43	0.00	0.00	<b>89.86</b>	0.00	0.00	0.00	0.00	3.43	0.00	0.00
<b>Black locust</b>	0.00	<b>5.17</b>	0.00	0.00	0.00	0.00	0.00	<b>94.33</b>	0.50	0.00	0.00	0.00	0.00	0.00
<b>Aspen</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.60</b>	<b>99.40</b>	0.00	<b>0.00</b>	0.00	0.00	0.00
<b>Red oak</b>	0.00	<b>0.41</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>99.59</b>	0.00	0.00	0.00	0.00
<b>Eucalyptus</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.41	0.00	0.00	0.00	<b>98.88</b>	0.00	<b>0.65</b>	0.06
<b>Corsican pine</b>	2.57	1.00	0.43	<b>13.00</b>	0.00	0.00	5.00	0.00	0.00	0.00	0.00	<b>78.00</b>	0.00	0.00
<b>Willow</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.46</b>	0.00	0.00	0.00	<b>99.54</b>	0.00
<b>Austrian black pine</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.07</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>99.93</b>

**Table A3.9. Confusion matrix for level 3, year 2011. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.**

The confusion matrices obtained for year 2012 are presented in *Tables A3.10-A3.15*.

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>94.93</b>	5.07
<b>Broadleaf</b>	0.22	<b>99.78</b>

**Table A3.10.** Confusion matrix for level 1, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>99.03</b>	0.97
<b>Broadleaf</b>	0.06	<b>99.94</b>

**Table A3.11.** Confusion matrix level 1, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.90</b>	0.02	0.04	0.04
<b>Evergreen</b>	0.59	<b>99.06</b>	0.35	0.00
<b>Pine</b>	4.06	0.14	<b>91.59</b>	4.20
<b>Other conifer</b>	3.46	0.00	6.92	<b>89.62</b>

**Table A3.12.** Confusion matrix for level 2, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.97</b>	0.00	0.03	0.00
<b>Evergreen</b>	0.00	<b>100.00</b>	0.00	0.00
<b>Pine</b>	1.01	0.00	<b>98.17</b>	0.81
<b>Other conifer</b>	1.68	0.00	4.59	<b>93.73</b>

**Table A3.13.** Confusion matrix for level 2, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	<b>93.20</b>	1.04	0.48	<b>5.04</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.00	0.00
Oak	0.00	<b>95.19</b>	0.38	0.14	1.14	0.52	0.00	0.86	<b>1.67</b>	0.05	0.00	0.00	0.05	0.00
Black pine	3.14	<b>8.71</b>	<b>77.57</b>	4.00	0.00	0.00	1.43	0.00	0.00	0.14	0.00	5.00	0.00	0.00
Douglas fir	<b>19.33</b>	3.67	3.83	<b>61.83</b>	0.00	0.00	1.33	0.00	0.33	0.00	0.00	9.67	0.00	0.00
Silver birch	0.00	<b>3.92</b>	0.00	0.00	<b>95.60</b>	0.08	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.00
European ash	0.00	<b>14.57</b>	0.00	0.00	2.00	<b>78.14</b>	0.00	0.14	5.14	0.00	0.00	0.00	0.00	0.00
Maritime pine	0.57	0.29	4.86	4.14	0.00	0.00	<b>85.00</b>	0.00	0.00	0.00	0.00	<b>5.00</b>	0.00	0.14
Black locust	0.00	<b>18.33</b>	0.00	0.00	0.00	1.00	0.00	<b>78.17</b>	1.50	0.17	0.00	0.00	0.83	0.00
Aspen	0.00	<b>4.63</b>	0.00	0.00	0.06	0.06	0.00	0.10	<b>95.14</b>	0.00	0.00	0.00	0.00	0.00
Red oak	0.00	<b>1.94</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.18	<b>97.88</b>	0.00	0.00	0.00	0.00
Eucalyptus	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.12	0.00	0.00	<b>99.65</b>	0.00	<b>0.18</b>	0.00
Corsican pine	1.43	1.71	3.00	<b>8.00</b>	0.00	0.00	1.57	0.00	0.00	0.00	0.00	<b>84.29</b>	0.00	0.00
Willow	0.00	0.62	0.00	0.00	0.00	0.00	0.00	0.00	<b>4.15</b>	0.00	0.31	0.00	<b>94.92</b>	0.00
Austrian black pine	0.21	0.41	2.00	2.00	0.00	0.00	0.97	0.00	0.55	0.00	<b>2.62</b>	0.34	0.21	<b>90.69</b>

Table A3.14. Confusion matrix for level 3, year 2012. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
<b>Silver fir</b>	<b>96.32</b>	0.40	0.80	<b>2.48</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Oak</b>	0.00	<b>98.38</b>	0.10	0.00	0.38	<b>0.90</b>	0.00	0.19	0.05	0.00	0.00	0.00	0.00	0.00
<b>Black pine</b>	0.00	<b>4.71</b>	<b>93.57</b>	0.00	0.00	0.00	1.71	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Douglas fir</b>	<b>10.67</b>	0.67	2.17	<b>84.67</b>	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00
<b>Silver birch</b>	0.00	<b>2.16</b>	0.00	0.00	<b>97.84</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>European ash</b>	0.00	<b>1.71</b>	0.00	0.00	1.14	<b>97.14</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
<b>Maritime pine</b>	0.00	0.00	<b>4.00</b>	0.86	0.00	0.00	<b>92.00</b>	0.00	0.00	0.00	0.00	3.14	0.00	0.00
<b>Black locust</b>	0.00	<b>0.67</b>	0.00	0.00	0.00	0.00	0.00	<b>99.33</b>	0.00	0.00	0.00	0.00	0.00	0.00
<b>Aspen</b>	0.00	<b>0.57</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>99.43</b>	0.00	0.00	0.00	0.00	0.00
<b>Red oak</b>	0.00	<b>0.12</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>99.88</b>	0.00	0.00	0.00	0.00
<b>Eucalyptus</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>100.00</b>	0.00	0.00	0.00
<b>Corsican pine</b>	0.00	0.00	<b>2.29</b>	0.29	0.00	0.00	1.14	0.00	0.00	0.00	0.00	<b>96.29</b>	0.00	0.00
<b>Willow</b>	0.00	<b>0.15</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.15</b>	0.00	<b>0.00</b>	0.00	<b>99.70</b>	0.00
<b>Austrian black pine</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.00</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>100</b>

Table A3.15. Confusion matrix for level 3, year 2012. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

The confusion matrices obtained for year 2014 are presented in *Tables A3.16-A3.21*

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>95.43</b>	4.57
<b>Broadleaf</b>	1.25	<b>98.75</b>

**Table A3.16.** Confusion matrix for level 1, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>	
	<b>Conifer</b>	<b>Broadleaf</b>
<b>Conifer</b>	<b>99.31</b>	0.69
<b>Broadleaf</b>	0.41	<b>99.59</b>

**Table A3.17.** Confusion matrix for level 1, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.78</b>	0.00	0.20	0.02
<b>Evergreen</b>	1.65	<b>95.35</b>	2.41	0.59
<b>Pine</b>	4.20	0.78	<b>91.51</b>	3.51
<b>Other conifer</b>	3.41	0.05	6.86	<b>89.68</b>

**Table A3.18.** Confusion matrix for level 2, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages

<i>Actual class</i>	<i>Predicted class</i>			
	<b>Deciduous</b>	<b>Evergreen</b>	<b>Pine</b>	<b>Other conifer</b>
<b>Deciduous</b>	<b>99.88</b>	0.00	0.12	0.00
<b>Evergreen</b>	0.18	<b>97.06</b>	2.76	0.00
<b>Pine</b>	0.43	0.20	<b>98.93</b>	0.43
<b>Other conifer</b>	1.24	0.11	4.49	<b>94.16</b>

**Table A3.19.** Confusion matrix for level 2, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages.

<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	<b>95.68</b>	0.00	1.04	<b>2.72</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.08	0.00	0.24
Oak	0.00	<b>97.90</b>	0.00	0.05	0.52	<b>0.62</b>	0.00	0.38	0.1	0.14	0.05	0.24	0.00	0.00
Black pine	2.29	1.14	<b>85.29</b>	<b>10.71</b>	0.29	0.00	0.14	0.00	0.00	0.00	0.14	0.00	0.00	0.00
Douglas fir	<b>16.17</b>	3.00	1.17	<b>71.17</b>	0.00	0.00	2.50	0.00	1.17	0.00	0.50	4.33	0.00	0.00
Silver birch	0.00	<b>3.52</b>	0.00	0.00	<b>95.68</b>	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00
European ash	0.00	<b>5.29</b>	0.00	0.29	0.00	<b>91.57</b>	0.00	0.71	1.43	0.00	0.71	0.00	0.00	0.00
Maritime pine	0.00	2.71	3.00	<b>3.29</b>	0.00	0.00	<b>91.00</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Black locust	0.00	<b>15.33</b>	0.00	0.50	0.00	5.67	0.00	<b>77.33</b>	0.50	0.00	0.50	0.00	0.00	0.17
Aspen	0.00	<b>1.09</b>	0.00	0.00	0.00	0.00	0.00	0.00	<b>98.91</b>	0.00	0.00	0.00	0.00	0.00
Red oak	0.00	<b>0.94</b>	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>99.06</b>	0.00	0.00	0.00	0.00
Eucalyptus	0.12	0.12	0.06	0.06	0.00	0.06	0.06	0.06	<b>0.82</b>	0.00	<b>97.76</b>	0.00	0.24	0.65
Corsican pine	0.43	<b>5.00</b>	0.43	3.86	0.00	0.14	0.00	0.00	0.00	0.00	0.00	<b>90.14</b>	0.00	0.00
Willow	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	<b>0.15</b>	0.00	<b>99.85</b>	0.00
Austrian black pine	0.14	0.07	0.07	0.69	0.00	0.00	0.00	0.00	0.62	0.00	<b>4.21</b>	0.07	0.14	<b>94.00</b>

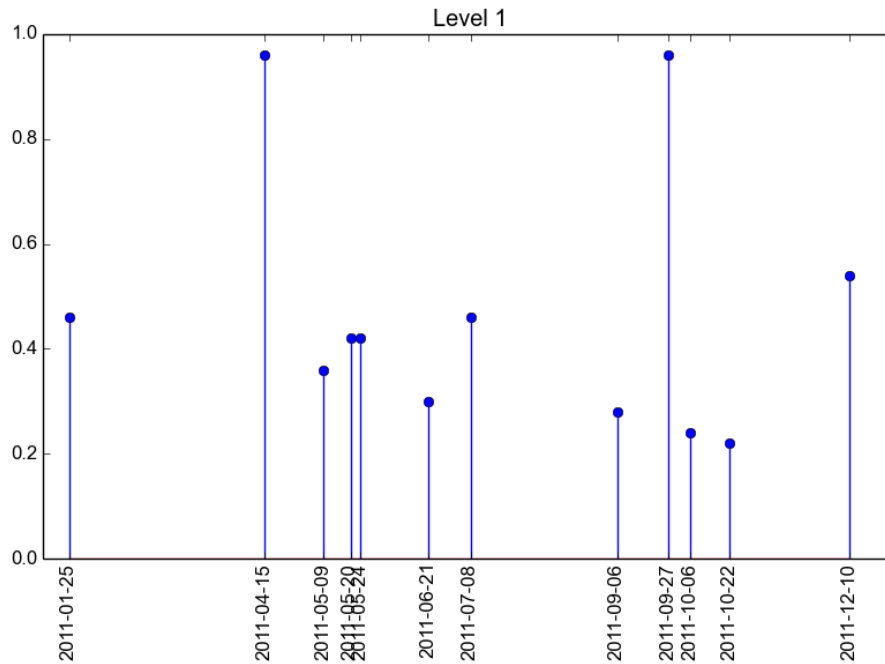
Table A3.20. Confusion matrix for level 3, year 2014. Classification was based on spectral bands and GMM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.

<i>Actual class</i>	<i>Predicted class</i>													
	Silver fir	Oak	Black pine	Douglas fir	Silver birch	European ash	Maritime pine	Black locust	Aspen	Red oak	Eucalyptus	Corsican pine	Willow	Austrian black pine
Silver fir	98.72	0.00	1.20	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Oak	0.00	98.71	0.00	0.00	0.00	0.81	0.00	0.10	0.00	0.00	0.00	0.38	0.00	0.00
Black pine	3.14	0.00	94.57	2.14	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Douglas fir	9.83	2.67	0.50	82.50	0.00	0.00	0.33	0.00	0.00	0.00	0.00	4.17	0.00	0.00
Silver birch	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
European ash	0.00	2.57	0.00	0.00	0.00	97.29	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00
Maritime pine	0.00	0.71	0.43	0.00	0.00	0.00	98.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Black locust	0.00	1.33	0.00	0.00	0.00	0.00	0.00	98.87	0.00	0.00	0.00	0.00	0.00	0.00
Aspen	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
Red oak	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
Eucalyptus	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.71	0.00	0.00	0.29
Corsican pine	0.00	0.43	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.43	0.00	0.00
Willow	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	99.85	0.00
Austrian black pine	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.10	0.00	0.00	98.90

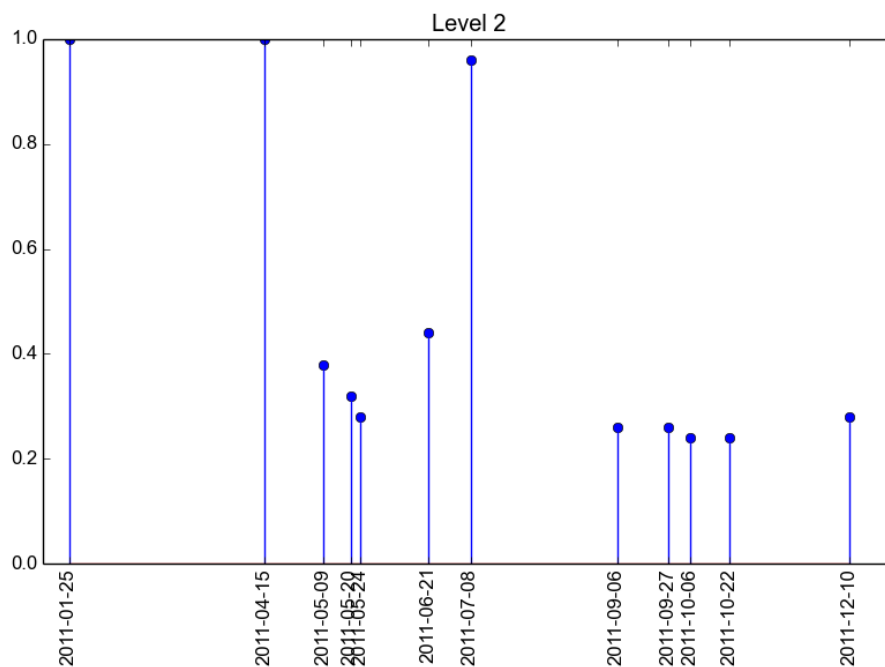
Table A3.21. Confusion matrix for level 3, year 2014. Classification was based on spectral bands and SVM classifier. The results are computed and averaged over 50 repetitions and presented in percentages. The main confusion for each species is reported in pink colour.



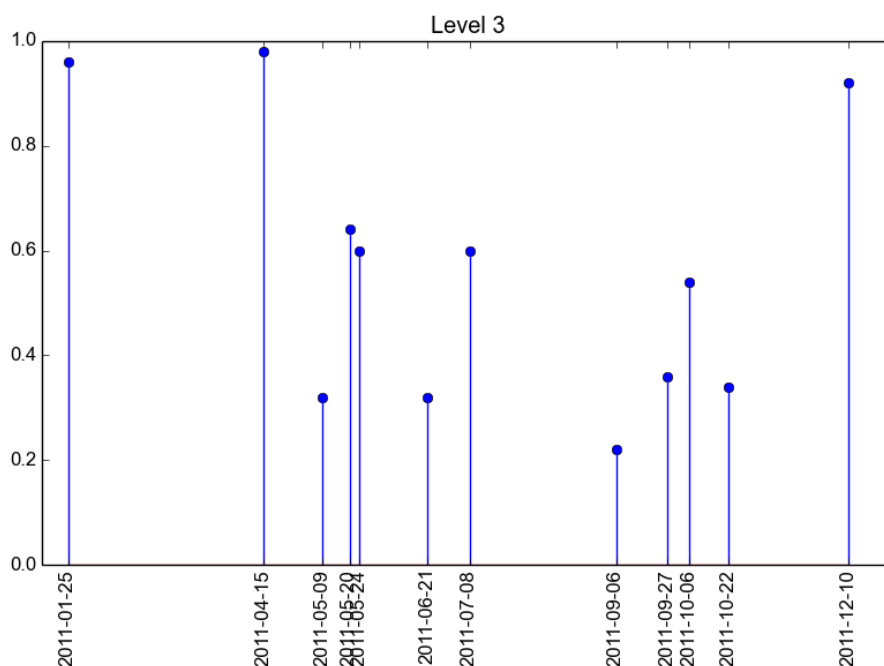
The most frequently selected dates for year 2011 for each level are presented in *Figures A3.4, A3.5 and A3.6*.



**Figure A3.4.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).

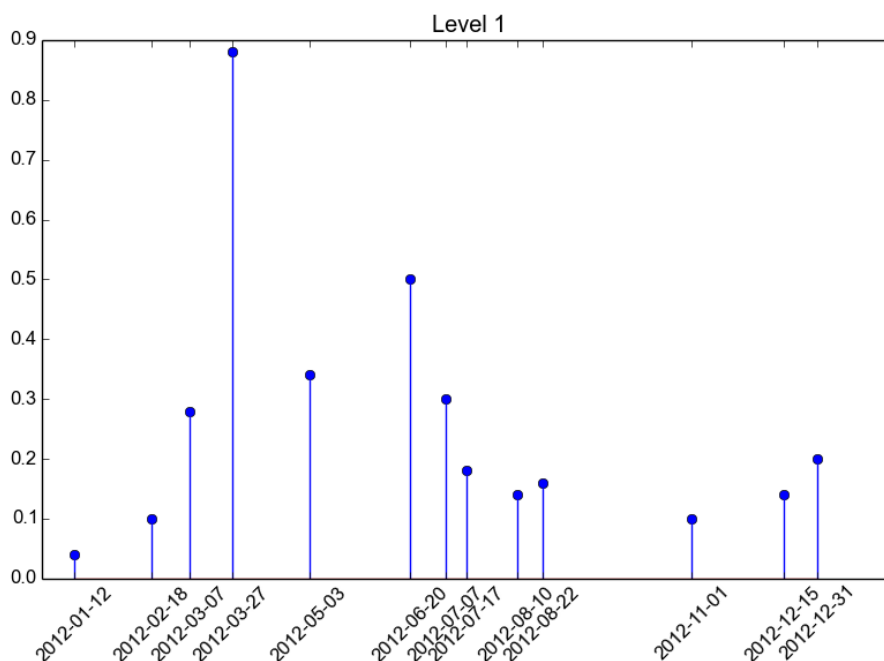


**Figure A3.5.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).

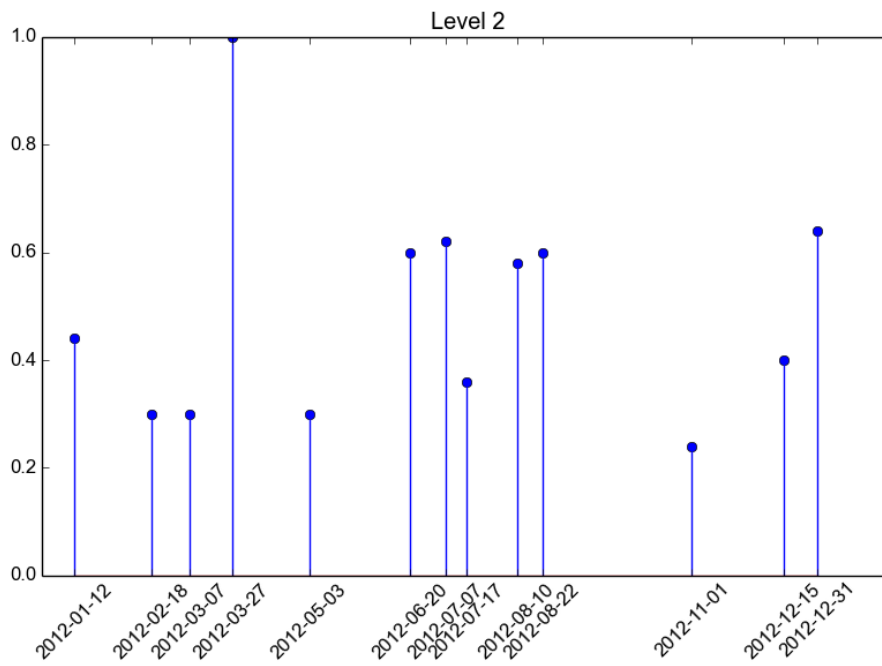


**Figure A3.6.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2011. The selection rate is 1 when the image is selected systematically (50/50).

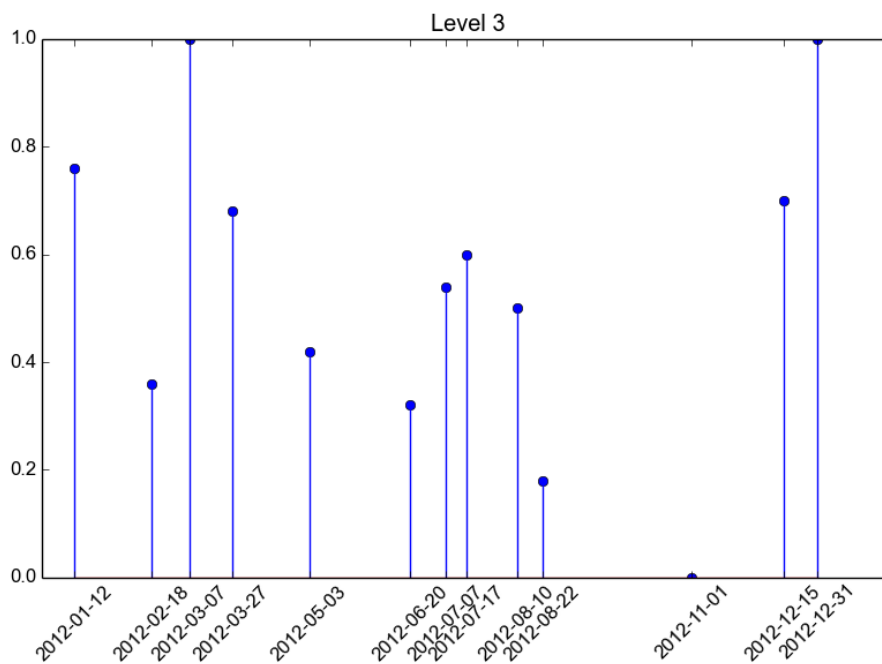
The most frequently selected dates for year 2012 for each level are presented in *Figures A3.7, A3.8 and A3.9.*



**Figure A3.7.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).

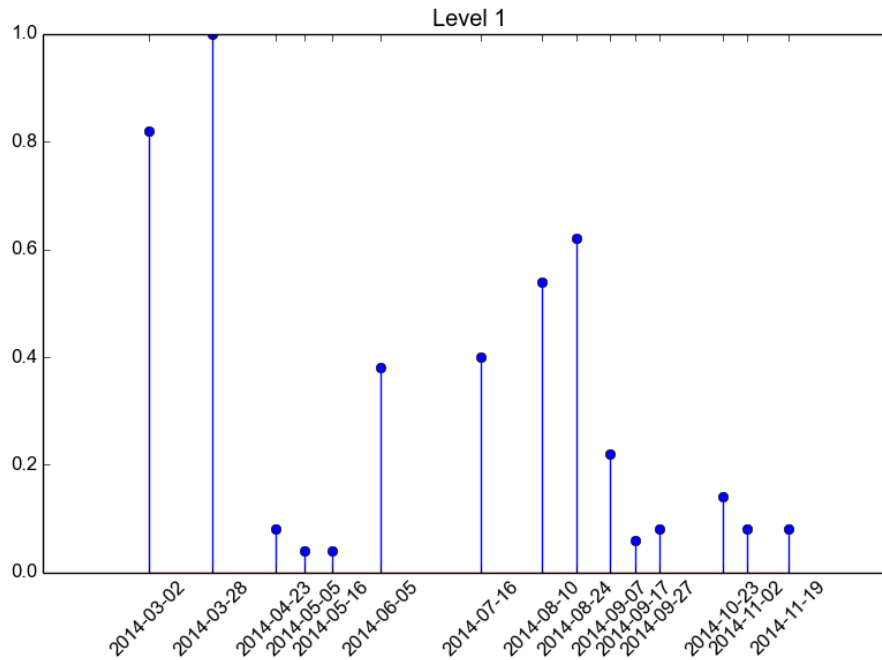


**Figure A3.8.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).

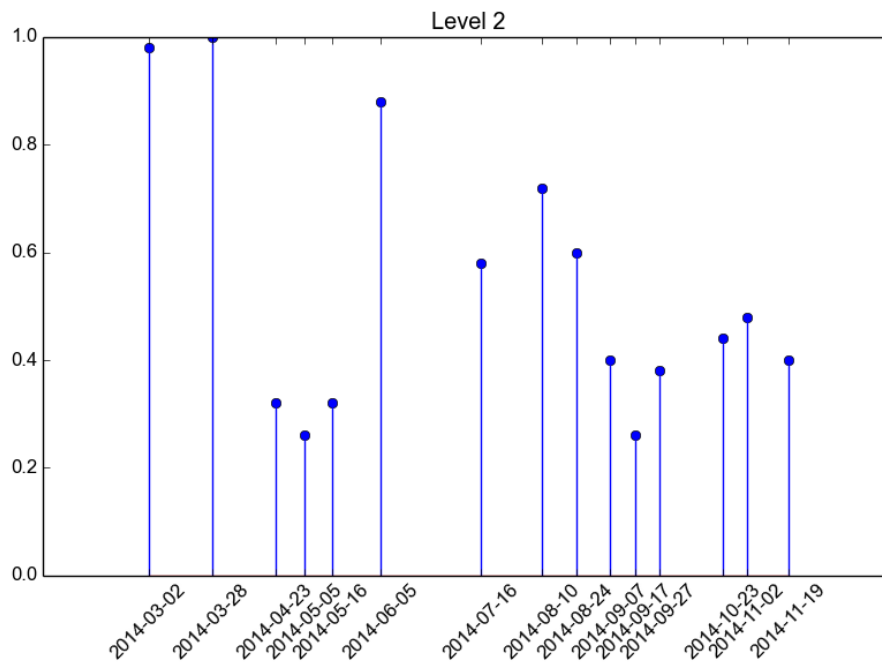


**Figure A3.9.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2012. The selection rate is 1 when the image is selected systematically (50/50).

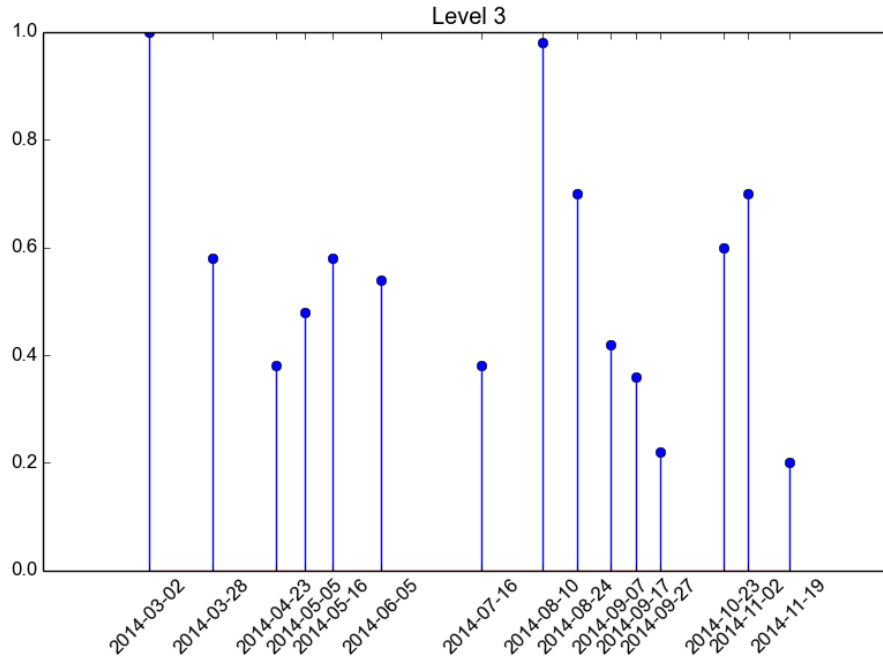
The most frequently selected dates for year 2014 for each level are presented in *Figures A3.10, A3.11 and A3.12*.



**Figure A3.10.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 1 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).



**Figure A3.11.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 2 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).

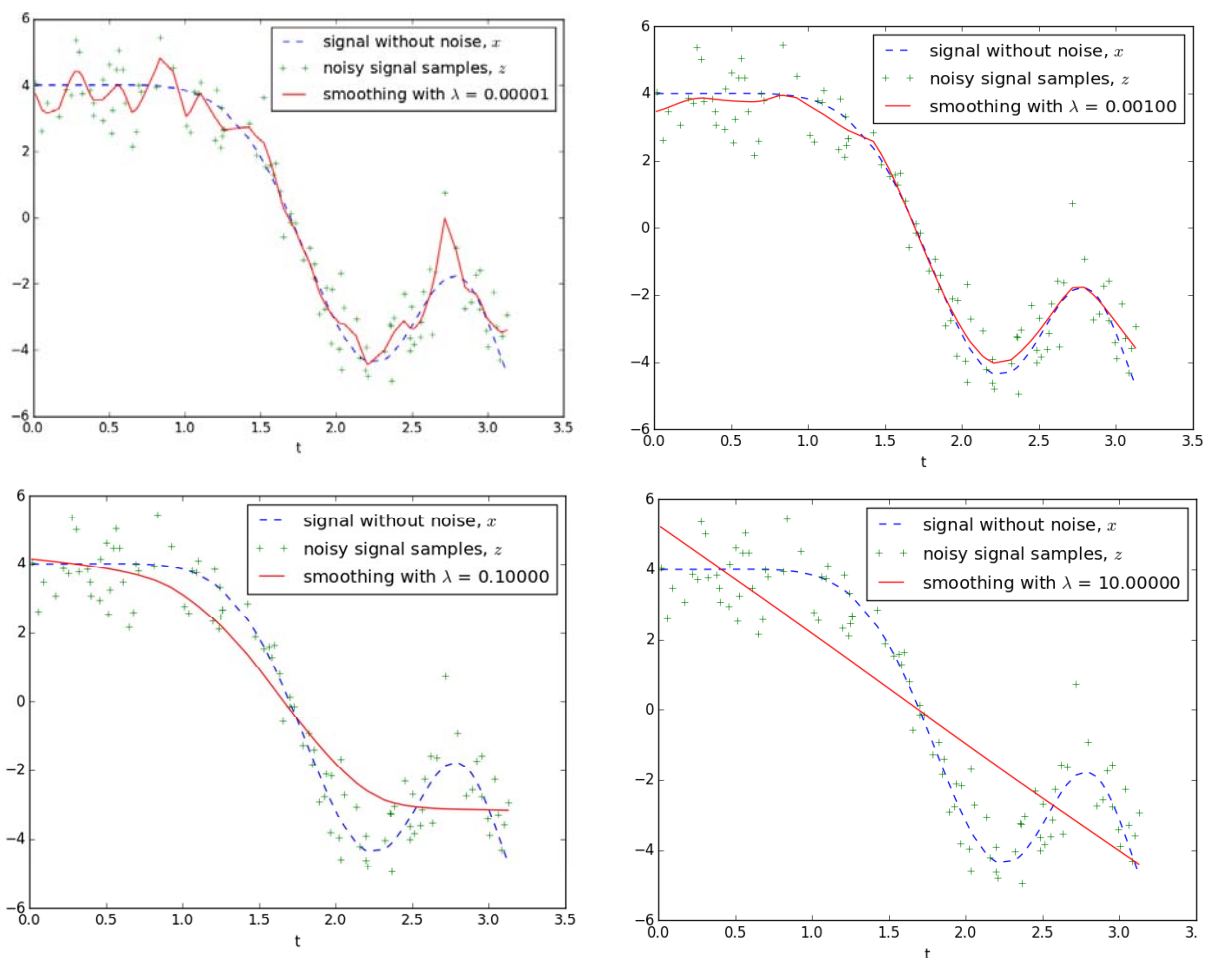


**Figure A3.12.** The most frequently selected dates over 50 repetitions obtained using NPFS classifier for level 3 and year 2014. The selection rate is 1 when the image is selected systematically (50/50).

## Appendix 4

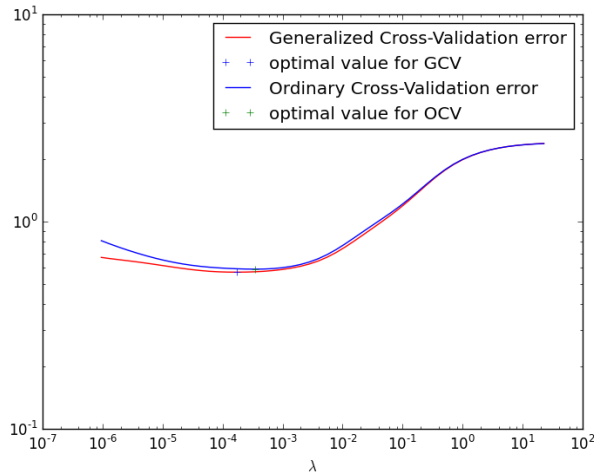
Some simulations were necessary to estimate the influence of the regularization parameter on SITS smoothing and to find the optimal value for regularization parameter  $\lambda$ . Signal without noise  $x$  is generated as a sum of sine and cosine functions. Afterwards, white noise with the variance of 0.8 was added to the original signal  $x$  to get noisy signal samples  $z$ . As a consequence, the noisy signal  $z$  will have the same mean as signal  $x$ , but also a different variance.

Basic result of the simulations was that with the increase of the smoothing parameter  $\lambda$ , the curve will get smoother (*Figure A4.1*).



**Figure A4.1.** The influence of the regularization parameter on smoothing of the simulated data with a second-order penalty ( $d=2$ ) for several values of the parameter  $\lambda$

*Figure A4.2* illustrates Generalized and Ordinary Cross-Validation errors for simulated data for different values of  $\lambda$ .



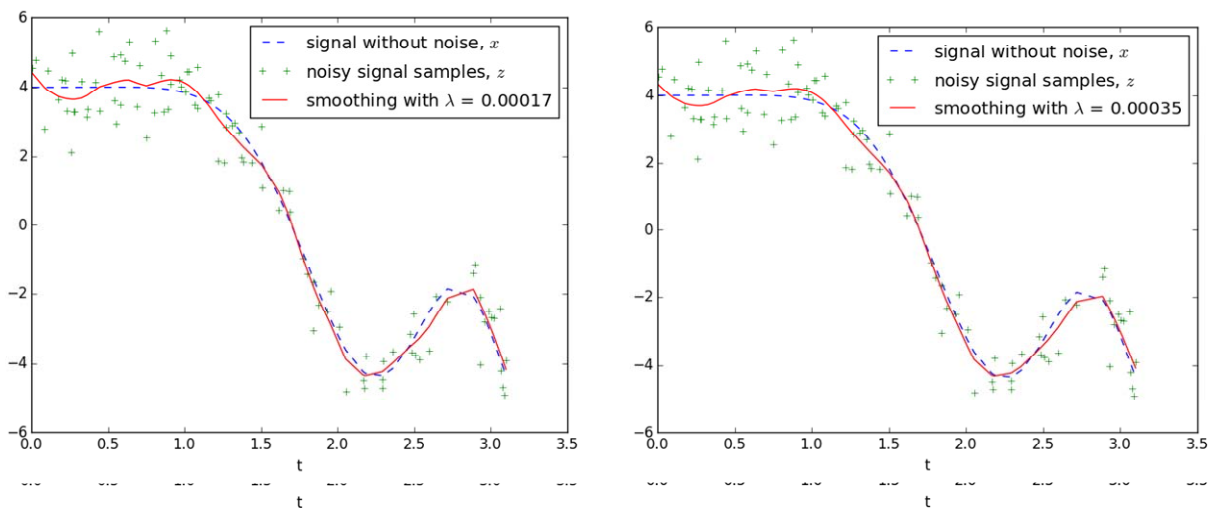
**Figure A4.2. Generalized and Ordinary Cross-Validation errors for different values of  $\lambda$**

The optimal values and errors for smoothing parameter  $\lambda$  using Generalized and Ordinary Cross-Validation (*Figure A4.1*) and simulated data (*Figure A4.2*) were:

$$\lambda_{GCV} = 0.00017, GCV(\lambda_{GCV}) = 0.57$$

$$\lambda_{OCV} = 0.00035, OCV(\lambda_{OCV}) = 0.59$$

When comparing smoothing results obtained using optimal  $\lambda_{GCV}$  and  $\lambda_{OCV}$  (*Figure A4.3*), Generalized Cross-Validation method has a lower error, but in general smoothing results obtained by both methods are almost the same.



**Figure A4.3 Smoothing the simulated data with  $\lambda_{GCV}$  (on the left) and  $\lambda_{OCV}$  (on the right)**

## References

- [1] C. B. Field, V. R. Barros, K. Mach and M. Mastrandrea, "Climate change 2014: impacts, adaptation, and vulnerability," *639 Contribution of Working Group II to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, vol. 640, 2014.
- [2] Food and Agriculture Organization of the United Nations, "Global forest resources assessment: Main report", 2010.
- [3] Y. Xie, Z. Sha and M. Yu, "Remote sensing imagery in vegetation mapping: a review," *Journal of Plant Ecology*, vol. 1, pp. 9-23, 2008.
- [4] D. S. Boyd and F. M. Danson, "Satellite remote sensing of forest resources: three decades of research development," *Progress in Physical Geography*, vol. 29, no. 1, pp. 1-26, 2005.
- [5] P. M. Mather and M. Koch, *Computer Processing of Remotely-Sensed Images: An Introduction, Fourth edition*, Wiley Blackwell, 2011
- [6] P. Meyer, K. Staenzb and K. I. Ittena, "Semi-automated procedures for tree species identification in high spatial resolution data from digitalized colour infrared-aerial photography," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 51, no. 1, pp. 5-16, 1996.
- [7] A. Carleer and E. Wolff, "Exploitation of very high resolution satellite data for tree species identification," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 1, pp. 135 – 140, 2004.
- [8] M. Immitzer, C. Atzberger, and T. Koukal, "Tree species classification with random forest using very high spatial resolution 8-band worldview satellite data," *Remote Sensing*, vol. 4, no. 9, pp. 2661 – 2693, 2012.
- [9] H. Nagendra, "Using remote sensing to assess biodiversity," *International Journal of Remote Sensing*, vol. 22, no. 12, pp. 2377–2400, 2001.
- [10] M. Dalponte, L. Bruzzone, and D. Gianelle, "Tree species classification in the southern alps based on the fusion of very high geometrical resolution multispectral/hyperspectral images and lidar data," *Remote Sensing of Environment*, vol. 123, no. 0, pp. 258 – 270, 2012.
- [11] A. Ghiyamat, H.Z.M. Shafri, G.A. Mahdiraji, A.R.M. Shariff, and S. Mansor, "Hyperspectral discrimination of tree species with different classifications using single- and multiple-endmember," *International Journal of Applied Earth Observation and Geoinformation*, vol. 23, no. 0, pp. 177 – 191, 2013.
- [12] J.G. Mickelson, D.L. Civco, and J.A. Silander, "Delineating forest canopy species in the northeastern united states using multi-temporal tm imagery," *Photogrammetric Engineering & Remote Sensing*, vol. 64, no. 9, pp. 891 – 904, 1998.



- [13] T. Key, T.A. Warner, J.B. McGraw, and M.A. Fajvan, "A comparison of multispectral and multitemporal information in high spatial resolution imagery for classification of individual tree species in a temperate hardwood forest," *Remote Sensing of Environment*, vol. 75, no. 1, pp. 100 – 112, 2001.
- [14] R. A. Hill, A.K. Wilson, M. George, and S.A. Hinsley, "Mapping tree species in temperate deciduous woodland using time-series multispectral data," *Applied Vegetation Science*, vol. 13, no. 1, pp. 86–99, 2010.
- [15] P. J. Sellers, "Canopy reflectance, photosynthesis and transpiration. II. The role of biophysics in the linearity of their interdependence," *International Journal Remote Sensing of Environment.*, vol. 6, pp. 1335–1372, 1987.
- [16] J. Weier and D. Herring, "Measuring Vegetation (NDVI & EVI)," *Earth Observatory, NASA, USA*, 2000.
- [17] D. Sheeren, M. Fauvel, C. Planque, J. Willm and J. F. Dejoux, "Tree species discrimination in temperate woodland using high spatial resolution Formosat-2 time series," *8th IEEE International Workshop on the Analysis of Multitemporal Remote Sensing Images*, Annecy, France, 2015.
- [18] C. Atzberger and P. H. C . Eilers, "Evaluating the effectiveness of smoothing algorithms in the absence of ground reference measurements," *International Journal of Remote Sensing*, vol. 32, no. 13, pp. 3689-3709, 2011.
- [19] S. N. Goward, B. Markham, D. G. Dye, W. Dulaney, J. Yang, "Normalized difference vegetation index measurements from the advanced very high resolution radiometer," *Remote Sensing of Environments*, vol. 35, pp. 257-277, 1991.
- [20] P. H. C. Eilers, "A Pefect Smoother," *Analytical Chemistry*, vol. 75, no. 14, pp. 3631-3636, 2003.
- [21] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, pp. 1627-1639, 1964.
- [22] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer series in statistics, Springer, 2001.
- [23] M. Fauvel, A. Zullo, and F. Ferraty, "Nonlinear parsimonius feature selection for the classification of hyperspectral images," in *6<sup>th</sup> Workshop on Hyperspectral image and signal processing: evolution in remote sensing*, Lausanne, Switzerland, 2014.
- [24] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119-1125, 1994.

- [25] S. B. Serpico and G. Moser, "Extraction of spectral channels from hyperspectral images for classification purposes," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 2, pp. 484-495, 2007.
- [26] A. W. Whitney, "A direct method of nonparametric measurement selection," *IEEE Trans. Computers*, vol. 20, pp. 1000-1003, 1971.
- [27] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive computation and machine learning)*, The MIT Press, 2005.
- [28] T. Marill and D. M. Green, "On the effectiveness of receptors in recognition systems," *IEEE Trans. Information Theory*, vol. 9, pp. 11-17, 1963.
- [29] A. K. Jain, R. P. W. Duin and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, 2000.
- [30] A. K. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 153-158, 1997.
- [31] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for patten classifiers," *Pattern Recognition*, vol. 33, pp. 25-41, 2000.
- [32] A. R. Webb, *Statistical Pattern Recognition, Second Edition*, John Wiley & Sons, 2003.
- [33] M. C. Peel, B. L. Finlayson, and T. A. McMahon, "Updated world map of the Köppen–Geiger climate classification," *Hydrol. Earth Syst. Sci.*, vol. 11, pp. 1633-1644, 2007.
- [34] J. G. Boureau, "Manuel d'interprétation des photographies aériennes: application aux milieux forestiers et naturels," *Inventaire Forestier National*, 267 p., 2008.
- [35] O. Hagolle, G. Dedieu, B. Mougenot, V. Debaecker, B. Duchemin, and A. Meygret, "Correction of aerosol effects on multi-temporal images acquired with constant viewing angles: Application to formosat-2 images," *Remote Sensing of Environment*, vol. 112, no. 4, pp. 1689-1701, 2008.
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

- [37] J. B. Féret and G. P. Asner, "Tree species discrimination in tropical forests using airborne imaging spectroscopy," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, 2013.
- [38] P. Somol, P. Pudil, Jana Novovicova and P. Paclik, "Adaptive floating search methods in feature selection," *Pattern Recognition Letter*, vol. 20, pp. 1157-1163, 1999.
- [39] E. Vyzas and R. Picard, "Offline and online recognition of emotion expression from phys-iological data," *In Emotion-Based Agent Architectures Workshop (EBAA'99) at 3rd Int. Conf. on Autonomous Agents*, pp. 135–142, Seattle, WA, USA, 1999.
- [40] J. Healey and R. Picard, "Smartcar: Detecting driver stress," *In Proc. of the 15th Int. Conf. on Pattern Recognition (ICPR'2000)*, pp. 218–221, Barcelona, Spain, 2000.
- [41] K. Wiltschi, A. Pinz and T. Lindeberg, "An automatic assessment scheme for steel quality inspection," *Machine Vision and Applications*, vol. 12, pp. 113–128, 2000.
- [42] J. Reunanen, "Overfitting in making comparisons between variable selection methods," *The Journal of Machine Learning Research*, vol. 3, pp. 1371-1382, 2003.
- [43] M. Immitzer, C. Atzberger and T. Kaukal, "Tree species classification with Random Forest using high spatial resolution 8-band WorldView-2 satellite data," *Remote Sensing*, vol. 4., pp. 2661-2693, 2012.