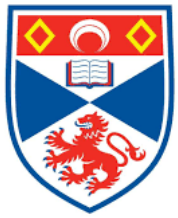
UNIVERSITY OF ST.ANDREWS

MASTER'S THESIS

A THESIS SUBMITTED IN FULFILLMENTS OF THE REQUIREMENTS FOR THE DEGREE OF
Msc Applied Statistics and Data mining



Application of Objected Oriented Methods and Semi-Supervised Machine learning algorithms for tree counts using Geo-spatial data

Author:
Sanil Ahan PURRYAG

Supervisor:
Dr. Carl R. DONOVAN
Dr. Andy LYNCH

August 17, 2018

Everything is related to everything else, but near things are more related than distant things.

Tobler's First Law of Geography

Acknowledgements

I want to dedicate this thesis to my parents and family, without whose support none of my work at the University of St Andrews would have been made possible. I would like to primarily thank Dr Carl R. Donovan and Dr Andy Lynch for their invaluable guidance and advice during the writing of this work. I would also like to extend my thanks to my fellow friends who have made this course a memorable and enjoyable experience. Finally, I want to give my special thanks to Pavlos Tsiantis, Jenovic Lumu, Niki Joannides and Tullio Mancini who have supported me by proofreading this work and inspired me to apply methods learned throughout the course to this project.

Declaration

I hereby certify that this dissertation, which is approximately 13,300 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree. This project was conducted by me at the University of St Andrews from June to August 2018 towards fulfilment of the requirements of the University of St Andrews for the degree of MSc Applied Statistics and Data Mining under the supervision of Dr. Carl R. Donovan and Dr. Andy Lynch.

Date: 17/08/2018

Abstract

The recent democratisation of aerial image capture devices has led to an abundance of Geospatial data and created the need for innovative ways of analysing such data. The current analysis methods are based around the use of specialised software to extract actionable intelligence from such datasets. While boasting a very high level of accuracy, the use of such software can often require much time and computational power to provide insights for the end user. This thesis aimed to extract and classify different regions of a map based on how similar the spatial points are between each other, in an automated fashion, through the use of the R Studio software suite. The study data encompassed the PortMoak Moss bog of Scotland, which covers an area of 0.459 km squared. The objective of classifying the PortMoak Moss map regions relates to the identification of tree-covered and non-tree covered areas, such that an estimate of the number of trees present is obtained.

To do so, Object-oriented methods and a Semi-supervised machine learning framework were considered for the task. The results from the Edge detection methods, involving the use of a laplace edge detector and a sobel edge detector, indicate that a range of tree counts from 154 to 321 trees is present in the area. Conversely, the results from the semi-supervised learning framework, including a K-means clustering algorithm and Random Forest classifier, suggest that around 3,684 trees are present. Nonetheless, given an initial estimated number of trees ranging from 436 to 9,687 based on certain assumptions and the area under study, the semi-supervised framework appears to provide a better estimation than the Object detection methods.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AUC Area under the curve

BCA Bias Corrected and Accelerated

CART Classification and Regression Tree

CHM Canopy Height Model

CI Confidence Intervals

CRS Coordinate Reference System

DSM Digital Surface Model

DTM Digital Terrain Model

EM Expectation Maximisation

EPSG European Petroleum Survey Group

EVI Enhanced Vegetation Index

FP False Positive

FN False Negative

GIS Geographic Information Systems

GPU Graphics Processing Unit

IDE Integrated Development Environment

KNN K-Nearest Neighbor

LIDAR Light Detection and Ranging

MSE Mean Squared Error

NIR Near Infrared

NDVI Normalised Differentiation Vegetation Index

OOB Out Of Bag

PSNR Peak Signal-to-Noise Ratio

RE Red Edge

RF Random Forests

ROC Receiver Operating Characteristic curve

RMSE Root Mean Squared Error

SE Standard Error

SPI Sierra Pacific Industries

SVM Support Vector Machines

TP True Positive

TN True Negative

TWSS Total Weighted Sum of Squares

UAV Unmanned Aerial Vehicle

WGS84 World Geodetic System of 1984

Chapter 1

Introduction

1.1 Background to the problem

O'Connor et al. (2017) indicate that the advent of affordable low-payload (less than 20kg) Unmanned Aerial Vehicle (UAV) has led to the increasing use of aerial imagery for studies and thus a corresponding abundance of geospatial data. This corresponding surge in such data has consequently led to numerous challenges, which includes the development of methods to aid users to match and express a spatiotemporal pattern of interest using maps as an interface for geospatial analysis (Robinson et al. 2017).

Presently, as indicated by the Environmental Systems Research Institute (ESRI) (2018), specialised software known as GIS are relied upon to extract actionable intelligence from such data and organise subsequent layers of information into visualisations for the end user. To differentiate between regions of a map and obtain the desired information, topographic analysis methods such as overlays are conventionally used and built on simplistic models. Nonetheless, the latter often involves several processing steps to arrive at interpretable and aggregated results (Tiede 2014). For instance, as Schweik (2011) outlines, the separation of areas of a map in a GIS involves manually joining the vertices of the area of interest into polygons which are in turn overlaid over the original map to allow for region segmentation.

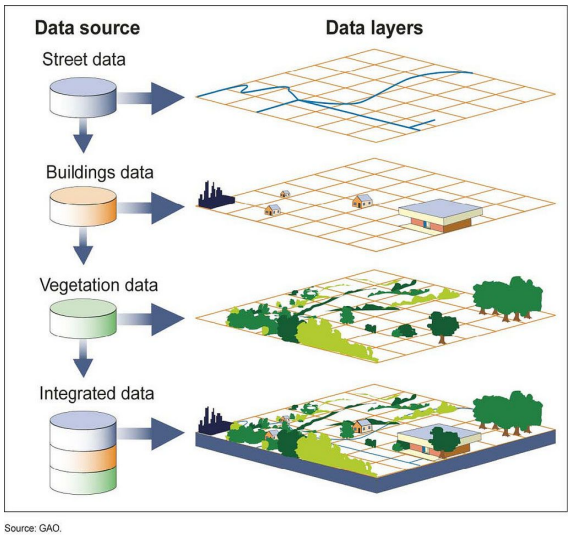


Figure 1.1: *GIS operation according to National Geographic Society (2012)*

1.2 Thesis Objective

This thesis aims to extract and classify different segments of a map based on how similar the spatial points are between each other, in an automated fashion, through the use of the R Studio software suite, running on the R back end which operates on version 3.4.1 (RStudio Team 2016). The considered segmentation procedure will aim to allow for polygon formation to distinguish between tree covered and non-tree covered areas, subsequently providing tree count estimates in the PortMoak Moss bog area.

1.3 Thesis Outline

The thesis will be organised as follows, where Section 2 will give a detailed outline of the data used and its structure. Section 3 will follow by providing an overview of the literature on tree crown identification. A theoretical background of the considered methods and a review of their efficiency in related works will be included. Section 4 will present the considered methodology, conceived from the literature review, followed by the results of its application on the dataset in Section 5. Section 6 will conclude and describe potential areas for further research as well as the limitations and challenges faced in this work.

Chapter 2

Overview of Data used

The provided data files for this study encompass a DSM and an orthophotograph (or orthophoto), encoded in a GeoTIFF format, which shall be described in further detail in the following subsections. Table 2.1 highlights the primary attributes of the DSM and stacked orthophoto (5 bands) studied while Table 2.2 illustrates the wavelength of the associated bands, with a ground sampling distance of 6.5m. With reference to Table 2.1 the spatial resolution refers to the size of the cells and their associated ratio of screen to image pixels at the present map scale. For instance, a X spatial resolution of 1:14 indicates that every screen pixel represents 14 cells of the X-axis data; the associated resolution would thus be approximately 0.07. As such, it can be noted that the data has a much higher spatial resolution, than the previous example, with more cells represented in each screen pixel.

Table 2.1: Characteristics of data

Attribute	**DSM Value**	**Stacked Orthophoto Value**
X spatial resolution	1.13433e-06	1.13435e-06
Y spatial resolution	6.32354e-07	6.32343e-07
Number of cells	526,267,160	287,328,000
Number of columns	23,710	17,520
Number of rows	22,196	16,400
Minimum X value	-3.338549	-3.334582
Maximum X value	-3.311654	-3.314708
Minimum Y value	56.1921	56.19463
Maximum Y value	56.20613	56.205

The study data was stored into raster files which organise information in a matrix-like structure where each pixel represents information about a specific characteristic such as spectral value or elevation units. With reference to the stacked orthophoto, each of its composing colour spectra has pixel intensity values ranging from 0 to 65,535 , in accordance with a digital 16 bit per channel colour scheme.

2.1 Study Area

The study area is located in Kinross, found in the south of Scotland, where aerial imagery was captured during the day from the Portmoak Moss forest area, located in the proximity of the Scottish gliding centre (Figure 2.1). The associated longitude and longitude of the forested area are 56°11'55.25"N and 3°19'31.93"W. The Woodland Trust Scotland (2004) underlines that the area is one of the few surviving raised bogs in Scotland and has a size of approximately 43.59 hectares or 0.4359 square km. The woodland is estimated to have a dominant conifer forest composition with some native broadleaved trees (The Woodland Trust 2018).

Figure 2.1: *Porkmoak Moss, Kinross, Scotland. 56° 11'55.25"N and 3° 19'31.93"W.* ***Google Earth****. Capture date: 01/01/2006. Accessed: 27/07/2018.*

The identification of the tree species making up PortMoak Moss will consequently allow for an initial tree estimation based on the tree crown and spacing between the trees. Gill et al. (2000) estimate that conifer species have an average crown radius ranging from 2.5m to 5.5m, or crown diameter between 5m to 11m, by analysing forest inventory data from Sierra Pacific Industries (SPI). Assuming an average tree crown diameter of 10m for all the trees in the area and a spacing of 10m between tree crowns, a high bound of approximately 48,433 trees can be obtained. However, since the area is not completely tree covered, a tree cover of 0.2 will be assumed, leading to a high bound of approximately 9,687 trees.

Alternatively, the estimates for tree establishment by the Forestry Commission England (2018) which estimate tree cover density to be approximately 1000 per acre will be used to obtain a lower bound for the area of Port Moak, such that approximately 436 trees will be estimated. Thus an interval of 436 trees to 9,687 trees is obtained as an initial estimation for the the Port Moak bog area, under the aforementioned assumptions.

2.2 Associated Projection system

As per Bivand et al. (2008), spatial data can vary based on how their position attributes are recorded and how their positioning has been determined. A standard way of describing locations is established through the Coordinate Reference System (CRS), which allows for distance calculations between observations and descriptions of the network topology based on the latter's relative positions. A CRS can be expressed either in terms of a geometric or projected model. A geometric model accounts for an ellipsoid that considers the curvature of the earth, and a datum, which anchors the geographic CRS to an origin point to allow for a three-dimensional representation.

Conversely, a projected CRS is a reprojection of a geometric model, keeping the latter's original attributes, to a desired planar system and specified measured of length (Bivand et al. 2008). A particular CRS system can be referenced in terms of its associated European Petroleum Survey Group (EPSG) code which illustrates how a CRS is defined and transformed (National Center for Ecological Analysis and Synthesis 2018). The provided data files are both encoded with the World Geodetic System of 1984 (WGS84) geometric datum with the related EPSG code 4326 (GISGeography 2018).

The rationale for reprojection relates to the ease of computation obtained, since the map is projected as a "flat" plane rather than an ellipsoid. For example, should area estimations for a geometric raster be considered, the latter will result in varying area estimates per latitude, conversely to a projected raster where a single constant and less computationally intensive estimate can be obtained as an approximation of the raster area.

2.3 DSM

As outlined by El Garouani et al. (2014), a DSM illustrates the elevation associated with any feature (natural or non-natural) located on the earth's surface. The Portmoak Moss DSM has been generated with its elevation units measured and fixed in inches with values ranging from 160 to 190 (approx. 4 - 5 meters), as shown in Figure 2.2.

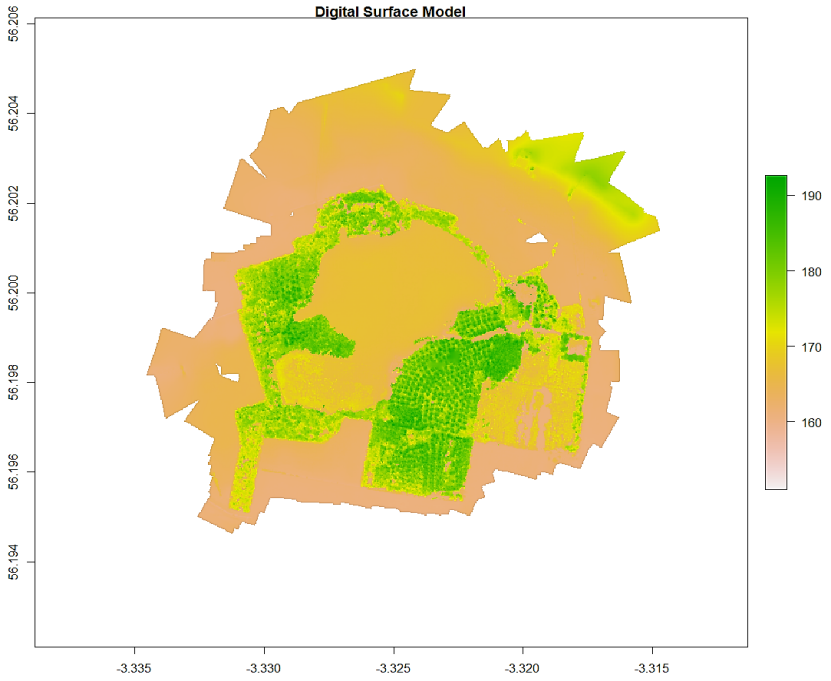


Figure 2.2: *DSM plot*

2.4 Orthophoto

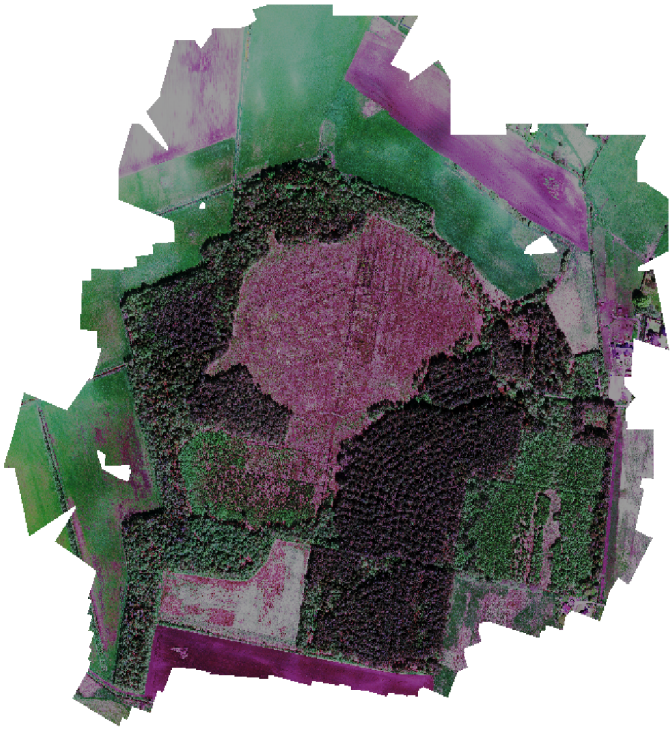
As defined by Rabiu & Waziri (2014), an orthophoto is a transformed vertical aerial photograph such that actual land features are preserved, rather than being represented as a combination of digital symbols and lines. The orthophoto for the study displayed the features of the Portmoak Moss forest using multi-spectral channels, which include: Blue, Green, Red, Near Infrared (NIR) and Red Edge (RE).

Table 2.2: Rapideye multi spectral band wavelength (Blackbridge 2014)

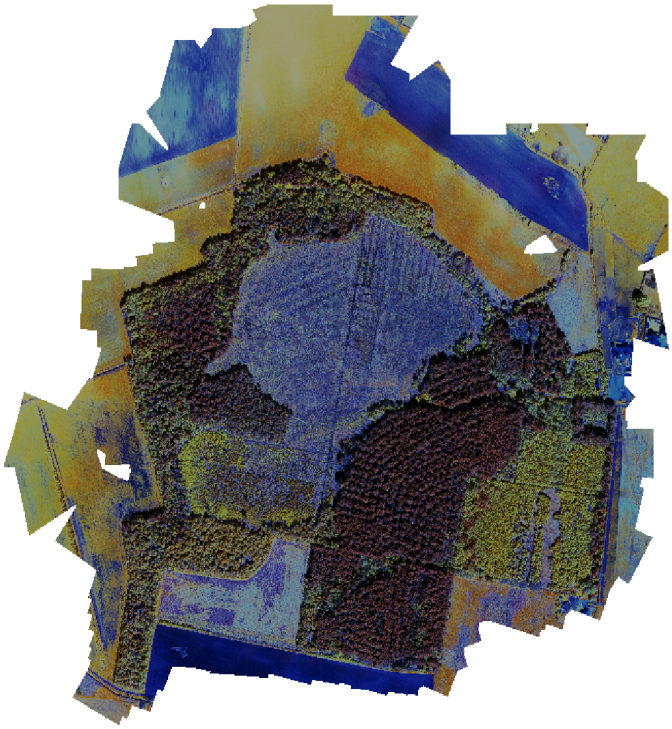
Band	**Wavelength (nanometers)**
Blue	440-510
Green	520-590
Red	630-685
NIR	690-730
RE	760-850

To view the different bands, two distinct types of stretches will be applied to the images; namely a linear stretch, keeping a one to one ratio to the present raster values (effectively projecting the true image), and a histogram stretch, which shall enhance the image by stretching the raster values relative to their frequency distribution. As it can be observed from Figure 7.3, the RE, NIR and Red color channels are the most contrasted images and thus have the most potential for tree identification. As previously mentioned, associated pixel intensity values for each band, which will later be used for feature extraction, range between 0 to 65,535 and are distributed as shown in Figure 7.1.

Alternate visualisations of the multi spectral bands include the creation of true and false colour composites, which are obtained by re-arranging the order of the bands to be rendered in the Red, Blue and Green channels of an image. A true color composite represents a near digital camera photo capture and is obtained by associated Red, Blue and Green to their default locations. Conversely a false colour composite helps to highlight vegetation areas of the study area and is obtained by selecting NIR as Red, Red as Green and Blue as Green. In this case, the true colour composite does not appear to be a direct comparison to the Google map capture shown in figure 2.1, analogous to a digital photograph, because of a tilted irradiation sensor from the remote sensing device at the time of capture and the 16 bit colour scheme used. The true and false colour composites are shown in Figure 2.3.



(a) True Color Composite



(b) False Color Composite

Figure 2.3: *Color Composites*

Chapter 3

Literature review

This chapter aims to provide an overview of the methods used in the literature for the tree crown delineation and will follow by outlining the theoretical underpinnings of selected methods from the literature.

3.1 Related Work

The existing literature is divided on how to achieve tree crown delineation, based on the availability of Light Detection and Ranging (LIDAR) data. Based on Fernández-Lozano et al. (2015), LIDAR is a remote sensing technology which measures variable distances to the earth and allows for the compilation of a Digital Terrain Model (DTM). As underlined by Matese et al. (2017), a DTM is then subtracted from the DSM to obtain a Canopy Height Model (CHM), which allows for the assessment of plant canopy height (Khosravipour et al. 2014). Liu et al. (2016) indicate that a CHM can then be used to delineate tree crowns and provide high-resolution tree crown maps. It should noted that since the CHM is essentially a subtraction of a raster from another, it provides more accurate results the higher the resolution of the involved rasters.

Given the unavailability of such data for this study and thus the impossibility for the use of the traditional method for tree crown delineation, the methods and related works investigated will be focused on tree crown identification from high-resolution satellite imagery. Since this problem is now understood to be analogous to an image segmentation or classification problem, two types of techniques will be considered, namely: Object-oriented methods and Machine learning methods. As argued by Ghassemian & Landgrebe (1988), object-oriented methods concentrate on identifying parts or pixels of an image which match certain features such as size and spectral response. Conversely, as advocated by Kanevski et al. (2008), machine learning algorithms are robust methods for modelling geospatial and associated extracting inherent patterns. The author elaborates that by adapting to the non-linear nature of geospatial data, machine learning algorithms are deemed to be easily implementable and robust data driven modelling tools that consider geospatial data not as two or three dimensional geographical data, but as high dimensionality data encompassing geographical features.

3.1.1 Object Oriented Methods

As discussed by Pal & Pal (1993), image segmentation can be used to either detect points, edges, regions or a combination of the former. For this study, edge and region detection, with methods combining both of the latter, will be investigated. As explained by Paine & Lodwick (1989), the edge extraction process is traditionally characterised by smoothing the image, detecting the edge, applying a threshold before thinning and linking the image. The work of Karimulla & Raja (2016) illustrates the application of the watershed segmentation algorithm for tree detection and count from high-resolution satellite imagery. Watershed segmentation, based on a greyscale image morphology, involves finding edges in the image by considering pixels having high elevation as light and pixels with a low elevation value as dark, relative to an established baseline for a region of the image (Singh 2016).

The authors expand on the use of watershed segmentation for tree counts after reducing the dimensionality of the satellite image using Principal Component Analysis (PCA), thresholding the image and applying an edge detection algorithm known as the Sobel Operator. They compare the performance of this methodology on images against a computed index known as the Normalised Differentiation Vegetation Index (NDVI), which aims to highlight vegetation covered areas from the bare ground. For this stream, they apply a decorrelation stretch, being a technique to enhance the colour separation of an image having highly correlated colour bands, on the NDVI and apply a different edge detection algorithm, known as the Canny operator, to identify tree crown boundaries.

It was found that the Canny operator performs better in this context than the watershed based segmentation. Certain other studies on edge detection and computer vision also support the notion that the Canny operator conventionally performs better than the Sobel operator (Sandhu & Mamta 2009, Kaur et al. 2012, Kaur & Singh 2016). Nonetheless, because of the iterative nature of the Canny operator and the computational strain associated, the Sobel operator shall be paired with another, more computationally efficient, edge detection method.

The Laplace Edge detection is identified as a more sensitive method to noise than the Sobel Edge detector and is deemed to have a better edge localisation, as well as isotropic properties (Kaur & Kaur 2014). Thus, to balance out the edge detection procedures used in the study, the Sobel Operator and the Laplace Operator shall be applied on the dataset before being compared against each other.

3.1.2 Machine learning algorithms

Machine learning algorithms fundamentally differ in their classification procedure based on whether they are supervised or unsupervised algorithms. Supervised algorithms assume the availability of a priori defined training sites on the image which encompass the predictor variable measured in the sampling unit and assign prior classes to the latter. On the other hand, unsupervised algorithms separate pixels based on their reflectance or intensity values into clusters or classes without the need for any prior information about the study area (Al-Doski et al. 2013). Since the data for this study does not have any predefined training sites or ground truth, both unsupervised and supervised classification methods will be examined with a view first to classify the regions of the map using unsupervised algorithms and then predict the tree cover using supervised algorithms, effectively creating a semi-supervised learning framework, similar to Rekik et al. (2006).

3.1.2.1 Unsupervised Classification

One of the most popular unsupervised methods employed in the literature is the K-means classification algorithm, which is deemed to be a conceptually simple and computationally scalable technique to create clusters (Xu & Wei 2012, Oyelade et al. 2010, Mills et al. 2011, Dhanachandra et al. 2015). Rekik et al. (2006) studied the application of K-means clustering for the creation of training datasets which have then been classified using Bayesian segmentation step. The results of their study indicate that the combination of the K-means clustering with a different segmentation algorithm performs better than simple unsupervised methods.

Given the popularity and effectiveness of the K-means algorithm, several variations have originated over time and have been applied to remote sensing data. Chandana et al. (2014) have investigated the effectiveness of K-means and its variants, namely Moving K-means, Fuzzy K-means and Fuzzy Moving K-means for remote sensing image classification. The results of their study indicate that while the standard K-means algorithm is the fastest and has the least number of iterations to achieve the cluster formation, the Fuzzy K-means moving algorithm is the most accurate. Nonetheless, given the size of the dataset considered and the need for computational speed, the simple K-means algorithm will be used for this study, based on its runtime, relative to its variants, as shown in Chandana et al. (2014).

3.1.2.2 Supervised Classification

Stephens & Diesing (2014) compared some of the most popular classification algorithm's performance when faced with satellite imagery. The results of their study indicate that the different implementations of the Naive Bayes Classifier and RF performed better overall, in terms of accuracy. Shataee et al. (2012) also compared the performance of RF with other classification algorithms, namely K-Nearest Neighbor (KNN) and Support Vector Machines (SVM), and concluded that SVM and RF had comparable performances to each other but superior to the KNN algorithm. Pal (2005) elaborate by comparing RF against SVM and find that RF are better performers having higher accuracy, simpler parameter specification and less training time.

Gislason et al. (2006) further indicate that RF have gathered much interest for image classification because of their non-parametric nature, which makes them operate without any assumption to the statistical distribution of the data studied, and because of their ability to identify which variables contribute to the classification in order of importance. Therefore, in line with Stephens & Diesing (2014) who advocate the use of simple lightweight models before resorting to more complicated variants, the RF classifier will be considered for the study.

3.1.2.3 Semi-Supervised Classification

To assess the efficiency and validity of the proposed method, a review of the literature on semi-supervised learning has been carried out. Miao et al. (2017) underline that some of the representative works of Semi-Supervised learning can be related to SVM, RF and Ferns (constrained decision tree ensemble model), among other models (Kursa 2012). For instance, De Freitas et al. (2009) studied the application of SVM as a semi-supervised learning method and compared them to traditional SVM as well as a Multi-layer perceptron neural network. The researchers have found that their semi-supervised framework outperformed the supervised learning framework, due to the extra information that was derived from the use of unlabelled training data combined with the already labelled training data. From a RF standpoint, Leistner et al. (2009) explored the application of a semi-supervised methodology combining a manifold learning algorithm and a RF on open object category datasets, including the Caltech-101 categorisation task (Fei-Fei et al. 2006). The results of their study indicate that their methodology provided high performance levels when using unlabelled data.

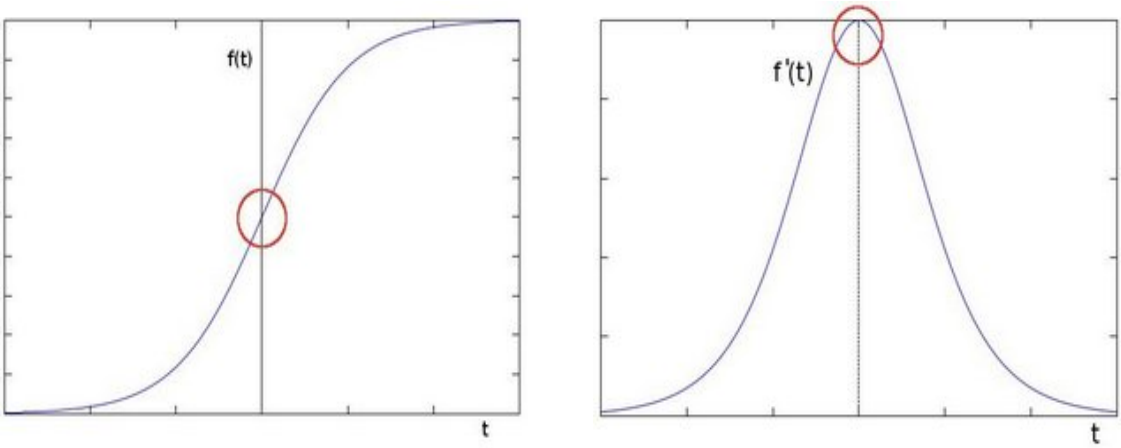
Additionally, Vatsavai R.R. (2005) investigated the use of a semi-supervised framework method based on an Expectation Maximisation (EM) algorithm combined with maximum likelihood and maximum a posteriori classifiers for thematic remote sensing classification. They noted that by using a small set of labelled and unlabeled training samples, classification accuracy increases overall but is not guaranteed to other datasets, given the intricacies of designing a semi-supervised classifier. Wang et al. (2015) indicates a similar finding with their semi-supervised algorithm based on a multinomial logit regression for remote sensing image classification which is faced with several constraints which stem from hardware limitations, such as long run time, despite having an overall good performance on their study data.

Finally, Maxwell et al. (2018) provides a review of semi-supervised applications to the field of remote sensing. He finds that there is no one best method to use and recommends trying different combinations of supervised and unsupervised learning algorithms, given the different idiosyncrasies present in different remote images. He elaborates by outlining that while tuning procedures are desirable, certain algorithms such as RF can provide generally robust results under a default parameter specification that are close to what they produce after having tuned. The latter are also observed to often outperform parametric based classifiers using solely their default parameters. The author argues that, in the case of remote sensing, class imbalance should also be an important consideration, should a theme based classification be the focus of the analysis.

3.2 Object detection methods

Kaur et al. (2012) define an edge as being a separation between 2 regions having specific grey level properties. These discontinuities can be estimated using the rate of change gradient of an image's pixel intensity values. An edge is identified when the gradient vector of an image, $f(x, y)$, points to the maximum rate of change, shown by an inflexion point. The calculation of the first and second derivatives at location (x, y) are considered as more computationally intensive but more indicative ways of estimating the rate of change in pixel intensity and the magnitude of the latter. Figure 3.1 shows how edge edges are identified through inflexion points, the first and second derivatives, considering an underlying function $f(t)$ which represents the values of a certain image.

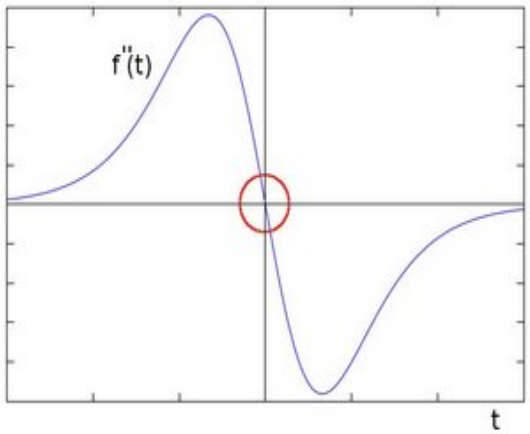
Taking into account the computational burden involved in detecting edges manually by calculating the gradient, certain edge detection methods, such as the Sobel Edge detector and the Laplace Edge detector, have been formulated and rely on approximations of the first and second derivatives. Furthermore, it should be outlined that those methods use certain predefined kernels that detect edges through convolution operations.



(a) Indication of Edge by inflexion point in image function values $f(t)$

(b) Identification of Edge through first derivative of image function $f'(t)$

Figure 3.1: *Edge detection as per Bradski (2018) and Bradski (2000)*



(a) Identification of Edge through second derivative of image function $f''(t)$

3.2.1 Sobel Edge Detection

As indicated by Shrivakshan & Chandrasekar (2012), the Sobel operator uses two 3 x 3 convolution kernels which are applied horizontally and vertically, respectively, over the pixel grid of the image under study. These convolution kernels are conceived such that they each can respond maximally to vertically and horizontally running edges relative to the pixel grid. The X coordinate is defined such that it gradually increases towards the right while the Y coordinate moves towards the bottom of the image, as the convolutions occur.

Consider G_x and G_y which are two images that contain the horizontal and vertical derivative approximations (Sobel Masks), obtained after multiplying the convolution kernels by the original image (Equation 3.1):

$$G_x = \begin{vmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{vmatrix} * A \quad G_y = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{vmatrix} * A \quad \textit{where A is the original source image} \tag{3.1}$$

The absolute magnitude $|G|$ can then computed using Equation 3.2 to obtain the output edges (Gupta & Ghosh Mazumdar 2012).

$$|G| = G_x + G_y \tag{3.2}$$

Moreover, the gradient vector (Equation 3.3), the orientation of the edge (Equation 3.4) and the rate of change of the gradient (Equation 3.5) can be calculated, as per Kaur et al. (2012), as follows:

$$\nabla f = \frac{G_x}{G_y} = \frac{\delta f / \delta x}{\delta f / \delta y} \tag{3.3}$$

$$\theta = \arctan(\frac{G_x}{G_y}) - \frac{3\pi}{4} \tag{3.4}$$

$$|\nabla f| = \sqrt{G_x^2 + G_y^2} \tag{3.5}$$

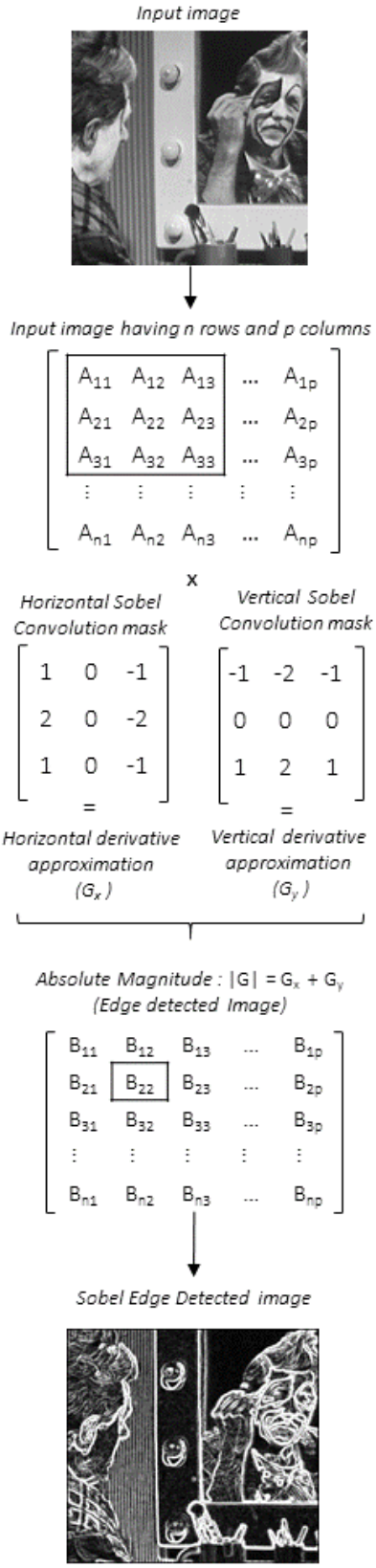


Figure 3.2: *Sobel detection algorithm functioning adapted from Fisher et al. (2003)*

3.2.2 Laplace Edge Detection

The Laplace Edge Detection Method detects zero crossings in the second derivative of an image, under the assumption that at a point where no further change occurs, a maximum is present. Therefore, the existence of a zero crossing indicates the presence of an edge for the Laplace edge detector and thus allows for the omission of unimportant edges that correlate with the first order derivative (Haralick 1984). This effectively makes the Laplace Edge detector more sensitive to noise than the Sobel Edge detector, because of its reliance on the second order derivative (Equation 3.6). It should also be noted that, unlike the Sobel Edge Detector, the laplacian edge detector uses only one convolution kernel, which for the purposes of this study excludes diagonals.

$$Laplace(f) = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2} \tag{3.6}$$

$$Laplace\ kernel = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix} \tag{3.7}$$

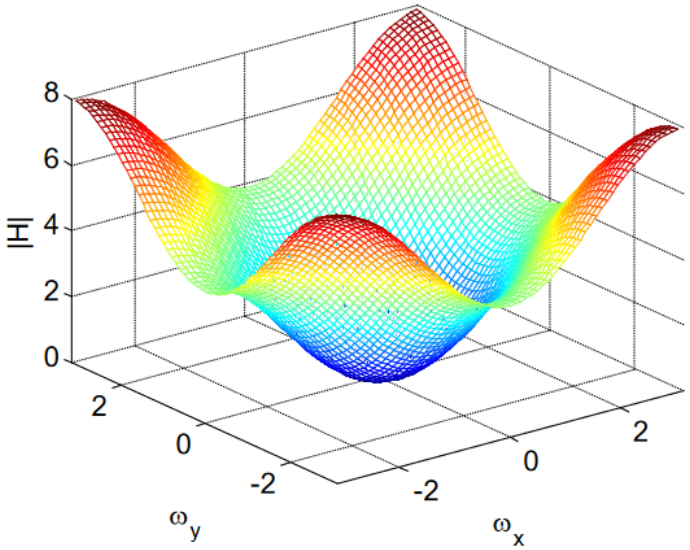


Figure 3.3: *Approximation of Laplacian operator by 3x3 filter sourced from Girod (2013)*

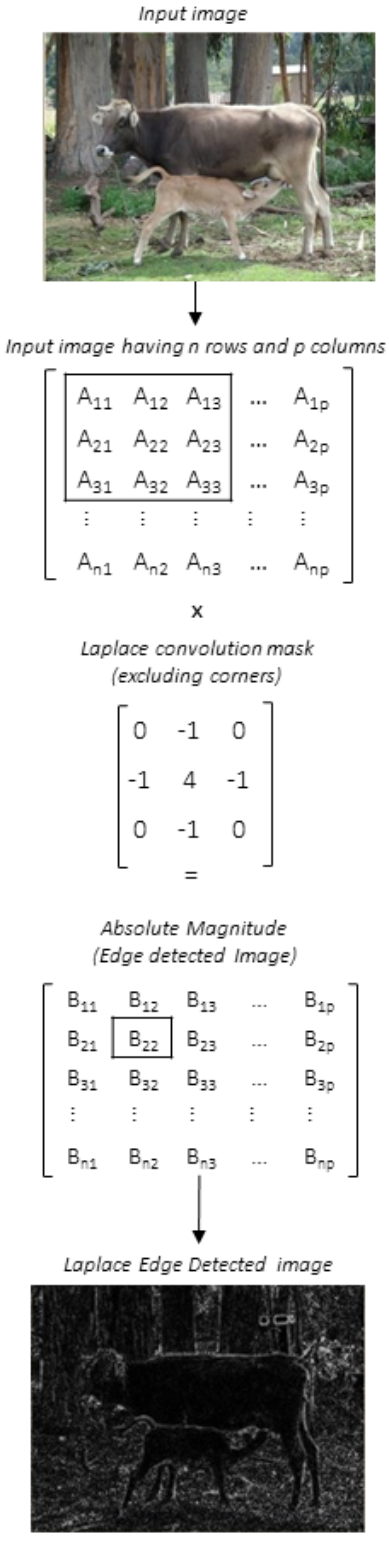


Figure 3.4: *Laplace detection algorithm functioning adapted from Bradski (2018) and Bradski (2000)*

3.3 Machine learning algorithms

3.3.1 K-means clustering

Sirait et al. (2017) report that K-means clustering is an iterative algorithm that aims to cluster observations based on their proximity to each other, such that the nearer observations belong to the same cluster, as compared to the further ones which are excluded from this specific cluster. Introduced by Macqueen (1967) and improved by Lloyd (1982), the K-means algorithm gained in popularity because of the simplicity associated with its implementation (McCool et al. 2012). Sirait et al. (2017) also indicate that the conventional measure of distance used for the proximity of the points is the Euclidean distance (Equation 3.8), which is computed as follows for two points x and y:

$$d_{euclidean}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.8}$$

The k-means algorithm operates as shown below, as per Sirait et al. (2017), and an illustration of how its operating mechanism is also provided in figure 7.5 of the Appendices.

Algorithm 1 K-means Clustering Algorithm (Sirait et al. 2017)

INPUT: Dataset of interest, Number of desired clusters (k)
OUTPUT: Clustered Dataset

1: **procedure** K-MEANS CLUSTERING ALGORITHM
2: Start
3: Set initial cluster centres (centroids) count $= k$
4: Determine intial centroid locations randomly from datapoints
5: **while** Reallocation of centroid possible **do**
6: Compute the Euclidean distance between the centroid and each data point
7: Group datapoints having the mimium distance to a specific centroid
8: Define new centroid by taking the average distance of the grouped data points
9: Stop iterations and output clusters
10: End

Note here that the K-means algorithm requires as input parameter the desired number of clusters. As such, particular numbers of clusters will influence the reallocation of centroids and consequently lead to differing results. Syakur et al. (2018) also propose that a wrong starting point for the K-means algorithm will lead to poor clustering results and high associated errors.

These authors advocate the use of the elbow method to determine the number of clusters, which involves computing different cluster numbers and comparing the resulting Total Weighted Sum of Squares (TWSS). The latter is done such that the optimal number of clusters is defined as the point when the TWSS is stable and when additional clusters do not warrant the increasing complexity in the algorithm, based on diminishing returns from the TWSS loss. Determining the elbow is, however, subjective and can thus lead to ambiguous solutions.

An alternative method of determining the number of clusters can be done using the GAP statistic. This technique involves the comparison of the TWSS obtained from computing between different cluster numbers using data with no apparent clustering, generated from the underlying input dataset by using Monte Carlo simulations (Tibshirani et al. 2000). Nonetheless, while the GAP statistic is considered for the analysis, its application might be inhibited by the hardware limitations, given its nature as a computer intensive method.

3.3.2 CART

An extensive body of research has been dedicated to machine learning methods, which would reconcile easily interpretable results and high predictive accuracy (Yang et al. 2017). One such highlighted method throughout the literature are regression trees, which rely on an iterative algorithm to define a tree-structured solution for either classification or regression problems. Breiman et al. (1984) engineered one of the most popular decision learning algorithm for classification trees, CART, that defines how a tree learns patterns over iterations. It is observed that the fundamental idea of tree methods rely on selecting a split such that the resulting child nodes are the purest, i.e have the most distinct separation between them (IBM Corporation 2013).

Yang et al. (2017) explains that the CART algorithm operates by taking one input and one splitting point, before forming 2 child nodes which separate the samples. Considering a data set already split in a training and validation set, they elaborate by showing that, by starting from the set of all training instances, i.e the root node, the best split is determined by an exhaustive search procedure, such that all the potential splits at that node are computed and the split returning the minimum deviations is chosen. It should be outlined that those minimum deviations are obtained by predicting two child nodes whose output is averaged Yang et al. (2017). Another possibility outlined in the literature is to find the best split that maximizes a splitting criterion that is indicative of a node's purity level (IBM Corporation 2013).

One such splitting criterion is the Gini index which is computed as, according to Jaworski et al. (2018), as shown in Equation 3.9.

$$Gini\ Index = 1 - \sum_{k=1}^{K} \frac{n^k(S)}{n(S)} \tag{3.9}$$

Where, considering that the components of the dataset are defined as C attributes that belong to k classes such that there is a total number of K classes; $n(S)$ represents the number of elements in the set S, such that $n^k(S)$ denotes the number of data elements in the S set, which also belong to the k class.

Yang et al. (2017) expands by outlining that the procedure iterates until an over fitted tree is obtained and a pruning procedure is considered. A pruning procedure involves removing the parts of the tree that do not markedly contribute to improving the classification of the different training instances. It is indicated that the pruning procedure is sequentially applied from the child nodes to the root node, to engender a sequence of candidate trees. Each candidate tree is finally tested against the aforementioned validation set, or using a predetermined cross validation procedure, and the tree returning the lowest generalisation error is selected as the final tree. A cross validation procedure involves splitting the data into to fit and predict the performance of competing models in an iterative fashion.

3.3.3 RF model

Breiman (1999) proposes that RF are an ensemble model, composed of multiple classification or regression tree predictors. Svetnik et al. (2003) build upon the latter by specifying that the tree predictors are unpruned and formed from a combination of random feature selection and bootstrapped samples of the data. Hastie et al. (2009) explain that the random sample of m is made from the full set of p predictors and each split of the tree is allowed to use solely one of those m predictors. They emphasize that a new sample of m predictors is selected at each split and chosen such that $m \approx \sqrt{p}$. This thus allows for the control over correlated predictions by allowing each split to consider only a subset of the predictors, such that an average of $(p - m)/p$ of the splits will not consider a dominant predictor and allow other variables to contribute to the predictions.

With reference to the 'randomForest' package used for the analysis, the m predictors are known as the *mtry* parameter. Therefore it can be noted that a too low *mtry* would lead to the possibility where variables with no predictive ability are selected for a split, as a result of the exhaustive search algorithm present in the classification trees. Conversely, an excessively high value of *mtry* would allow for predictors with the highest predictive ability to selected first consistently, leading to similar trees being ensembled across the RF. Thus, the value of *mtry* should be adjusted, or tuned, to allow for an adequate number of predictors to be considered at each split (Janitza & Hornung 2018).

Janitza & Hornung (2018) also indicate that random samples from the underlying dataset are used to construct the trees on the RF, conventionally obtained by a bootstrap. There are thus a number of datapoints, omitted from the bootstrap sample, which are not used to constuct trees. These observations, considered as Out Of Bag (OOB) samples, are used to compute an OOB error rate, such that predicted observations from trees not using OOB samples are compared against the associated OOB sample observations. Svetnik et al. (2003) finally propose that predictions are either made from a majority class voting (for classifications) or averaged results (for regressions) of the many predictor trees used in the RF model.

As described by Liaw & Wiener (2002*b*), the parameters of interest associated with the RF model, from the 'randomForest' package in R, are as follows in Table 3.1 accompanied by a description of the its overall functioning in Algorithm 2.

Table 3.1: RF Parameters of Interest

Parameter name	**Description**
mtry	Number of variables randomly sampled at each tree split
nodesize	Minimum size of terminal nodes
maxnodes	Maximum size of terminal nodes
ntree	Number of trees to grow
replace	boolean value to confirm if sampling of cases should be done with replacement
localImp	boolean value to confirm if variable importance should be computed

Algorithm 2 RF Classification Algorithm adapted from Liaw & Wiener (2002*a*)

INPUT: Dataset of Interest, *ntree*, *mtry*
OUTPUT: Predicted Classes, OOB Error

procedure RF CLASSIFICATION ALGORITHM
    Start
    Compute number of Bootstrap samples from Dataset of interest $=$ *ntree*
    for i in $sample_1$ to $sample_{ntree}$ **do**
        Grow unpruned classification tree
        Identify OOB observations by comparing sample observations against original data
        Randomly sample number of considered variables for split $=$ *mtry*
        Choose best split from *mtry* predictor
        Save prediction for $sample_i$
        Compare against OOB observations and obtain OOB error
        Save OOB error
    Return majority vote by aggregating predictions from $sample_1$ to $sample_{ntree}$
    Return OOB error associated with *mtry* value
    End

3.4 Other Related Concepts

In addition to the previously mentioned techniques, other concepts that shall be considered for the analysis have been identified throughout the extensive body of research that do not relate to either objected oriented methods or machine learning algorithms. This section will thus aim to describe these techniques and explain their functioning.

3.4.1 Vegetation Indexes

As established by Xue & Su (2017), vegetation indexes are simple and effective arithmetic combinations of two or more spectral bands which have allowed for improved identification of vegetation and tree cover from remote sensed data. Barati et al. (2011) explains that the effectiveness of such measures is derived by how biomass and leaves, among others, reflect differently the various wavelengths or image bands typically present in light and captured by satellite imagery (refer to Figure 3.5).

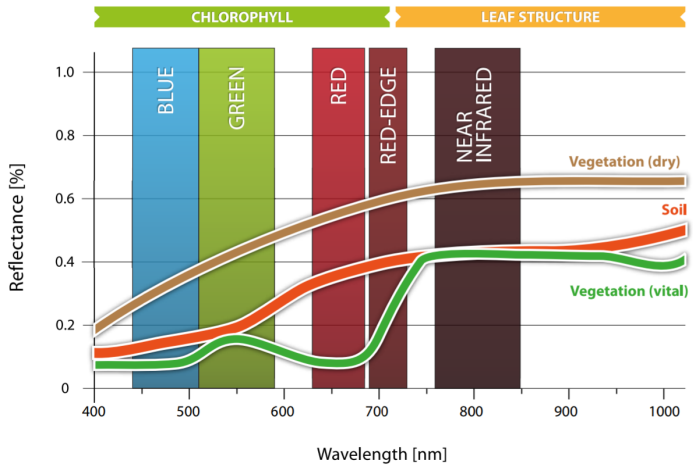


Figure 3.5: *Identification of vegetation based on wavelength from Blackbridge (2014)*

Therefore, two vegetation indexes, namely the NDVI and the Enhanced Vegetation Index (EVI) shall be used for the analysis.

3.4.1.1 NDVI

Roettger (2007) states that the Red and NIR color channels can be used to compute the NDVI, since living vegetation absorbs light in the range of the Red band and not in the NIR range. This differential between the two bands thus allows for a measure of vegetation activity, ranging from -1 to 1. Roettger (2007) emphasises that typically values for vegetation range from 0.1 to 0.7, with higher index value associated with healthier vegetation cover. The formula for the NDVI is as follows:

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{3.10}$$

3.4.1.2 EVI

Matsushita et al. (2007) assert that the EVI improves over the NDVI by adjusting for atmospheric conditions and canopy background characteristics to which the NDVI is sensitive, as presented in Liu & Huete (1995). The EVI allows for the detection of high biomass by considering also considering the Blue spectral band and a soil adjustment factor (L), in addition to the Red and NIR channels. As per Liu & Huete (1995), the EVI can be computed as follows:

$$EVI = G * \frac{NIR - Red}{NIR + (C_1 * Red - C_2 * Blue) + L} \tag{3.11}$$

where C_1 and C_2 are coefficients used to adjust for aerosol scattering in the red band by using the blue band. The default values associated with the above equation are G = 2.5, C_1 = 6.0, C_2 = 7.5 and L = 1, based on (Matsushita et al. 2007).

3.4.2 Hillshading

Pioneered by Horn (1981), hillshading allows the computation of the surface topography of a map based on the slope and aspect of the terrain. It should be noted that the computation of the slope values is dependent on the values of the neighboring cells for the hillshading. Thus, depending on the assumed surface of the terrain, either the method of Ritter (1987) using 4 neighboring cells (assuming a smooth surface) or Horn (1981) using 8 neighboring cells (assuming a rough surface) should be used. This technique shall thus be used in the analysis for the following reasons:

- To provide for a better visualisation of the terrain;
- To allow for the separation of the trees from the ground, based on the slight resulting elevation from the estimation of the slope and aspect of the terrain;
- To obtain a greyscale image that shall facilitate the application of the considered object oriented methods.

However, it should be outlined that the efficiency of the hillshading might be impaired by the fact that the default R command from the 'raster' package is designed for rasters with elevation units in meters. Thus, since the current DSM has been generated with elevation units in inches and cannot be accurately reprojected, the hill shading might not be as accurate as anticipated. Nonetheless, the technique shall still be used for the analysis for the generation of a monochromatic raster.

3.4.3 Aggregation

Aggregation involves reducing the resolution of a raster image by re-sampling values from the upper left of the raster and aggregating them based on a desired strategy. In this case, the aggregation function will operate by reducing the number of cells in a raster image by a specified factor and taking the average of the aggregated values. For instance, a raster aggregated by a factor of 25 will have 625 (25 x 25) times less cells .

3.4.4 Thresholding

Thresholding is a fundamental concept in image analysis that allows for background-foreground separation. For the purposes of the analysis, a simple local thresholding method shall be used in order to allow for region segmentation in the rasters of interest. Consider a threshold value T where (x, y) are the associated coordinates (Singh & James 2012).

Therefore, for a maximum threshold value $T(x,y)$, which is considered as the inverse of the minimum thresholding value, and pixel intensity values $I(x,y)$:

$$b(x,y) = \begin{cases} 1 & \text{if } I(x,y) \leq T(x,y) \\ 0 & \text{Otherwise} \end{cases}$$

3.4.5 Masking

Masking involves superimposing a raster onto another, with the same extent, to set certain pixels to NA values such that they are not included in the analysis. For instance, a thematic raster having different regions stored as classes could be superimposed onto a spectral raster band to find the matching regions between both rasters, such that the excluded regions are set to NA.

3.4.6 Bootstrapping

Boostrapping can be defined as a non-parametric computer intensive statistical technique that allows for probability based inferences about a population parameter of interest (θ^*), such as the mean (Woodroof 2000). The latter operates through the use of a re-sampling procedure where observations from an underlying dataset, with an unknown distribution, are re-sampled with replacement a defined number of times to simulate new samples. For each sample, a population parameter of interest, such as the mean, is computed for each such simulation. The standard non-parametric bootstrap algorithm can be described as follows in Algorithm 3.

Algorithm 3 Non-Parametric Bootstrap Procedure (Lei & Smith 2003)

INPUT: Dataset of interest, θ^*, Number of bootstrap re-samples
OUTPUT: Distribution of desired parameter values across re-samples

1: **procedure** NON-PARAMETRIC BOOSTRAP
2: Start
3: Define the original data such that $x = (x_1, x_2, ..., x_n)$
4: **for** i in 1:Number of bootstrap re-samples **do**
5: Re-sample randomly with replacement from the original data set
6: Obtain bootstrap re-sample such that $x^* = (x_1^*, x_,^* ..., x_n^*)$
7: Compute θ^* for the bootstrap re-sample such that $\theta^*_{sample\ i}$ is obtained
8: Save $\theta^*_{sample\ i}$
9: Return distribution of all computed $\theta^*_{sample\ i}$ as θ^*_{sample}
10: End

These repeated simulations can consequently allow for the estimation of the variation about point estimates, or Confidence Intervals (CI), since they provide an approximation of the true distribution of θ^*. To this effect, different CI estimation methods will be considered.

3.4.6.1 Normal CI

In the words of Lei & Smith (2003), the Normal CI is a standard way of computing a CI and is considered to be more accurate than its counterpart, the student's t CI for samples with more than 30 observations. The author thus expands that for a sample bootstrapped statistic θ^*_{sample} and its associated Standard Error (SE), the $(1-2\alpha)$ equal tailed CI can be computed as follows, where α is the estimated error level under the Z-distribution approximation:

$$Normal\ CI = \begin{cases} Upper\ limit & \theta^*_{sample} - z(1-\alpha) * SE(\theta^*_{sample}) \\ Lower\ limit & \theta^*_{sample} - z(\alpha) * SE(\theta^*_{sample}) \end{cases}$$

3.4.6.2 Percentile CI

The percentile CI is based on the intuitive notion to use the existing $100(1-\alpha)^{th}$ and $100(\alpha)^{th}$ percentiles of the distribution of θ^*_{sample} to define confidence bounds for the latter (Elias 2015). Therefore, the percentile CI is a simpler way to compute CI, not depending on the SE unlike the normal CI, and can be defined as:

$$Percentile\ CI = \begin{cases} Upper\ limit & 100(1-\alpha)^{th} \text{ percentile of the distribution of } \theta^*_{sample} \\ Lower\ limit & 100(\alpha)^{th} \text{ percentile of the distribution of } \theta^*_{sample} \end{cases}$$

It has been outlined however, that while simple and intuitive, the percentile method performed poorly when the value of θ^*_{sample} is distant from the actual value of θ^* or when bootstrapped distribution has a degree of skewness (Wagstaff et al. 2009).

3.4.6.3 Bias Corrected and Accelerated (BCA) CI

Wagstaff et al. (2009) elaborate that the BCA method has been proposed to overcome the shortcomings of the percentile method, with regards to skewness, by allowing for a bias correction factor. The BCA thus utlises the end points of the θ^*_{sample} distribution, obtained by using the percentile method, but corrects for any estimation errors that might arise from a non symmetric distribution.

Based on Lei & Smith (2003), the revised lower and upper CI, α_1 and α_2, can be obtained as follows:

$$BCA\ CI = \begin{cases} Upper\ limit & \alpha_2 = \phi(Z_0 + \frac{Z_0 + Z(\alpha)}{1 - a(Z_0 + Z(\alpha))}) \\ Lower\ limit & \alpha_1 = \phi(Z_0 + \frac{Z_0 + Z(1-\alpha)}{1 - a(Z_0 + Z(1-\alpha))}) \end{cases}$$

From the previous equation, ϕ is defined as the standard normal cumulative distribution function, with $Z(\alpha)$ and $Z(1-\alpha)$ as the percentile points of a normal distribution curve and α as the defined error level. Z_0 is the parameter used to measure the size of the median bias of θ^*_{sample} which is defined as the difference between the median value of θ^*_{sample} and θ^*. The acceleration parameter, a, captures the the speed at which the SE of θ^*_{sample} changes with regards to θ^*. It is thus seen that when $Z_0 = 0$ and $a = 0$, the BCA gives the same results as the Percentile method.

Chapter 4

Methodology

The objective of this section is to present the hardware used for the analysis and the different steps that will be involved in determining a tree count from the available DSM and orthophoto. In this regard, the hardware specifications will first be listed before providing a flowchart to illustrate how the considered methods will be applied in tandem, accompanied by a description of each stage of the analysis.

4.1 Hardware Specifications

The code was written using 64 bit "single candle" R-Studio Integrated Development Environment (IDE) version 1.0.154, with the R backend operating on version 3.4.1. The code was run on a personal machine with Quad-Core Intel i7-7500U 2.7 GHz with 8GB RAM and a dedicated Graphics Processing Unit (GPU) (GeForce 940MX) having 6GB of memory.

4.2 Proposed Work flow

This study will be divided into two segments, namely, one dedicated to estimating tree counts using object-oriented methods from the DSM while the other shall be focused on tree count estimation from the multispectral orthophoto using machine learning approaches. Figure 5.1 illustrates the steps involved in the below workflows.

4.2.1 DSM work stream

The DSM work stream will proceed by estimating slope and aspect characteristics from the DSM before applying the hillshading procedure to the raster. This will result in a monochromatic raster with 2 classes having the vegetation outlined in black and the ground outlined in grey. The hill shaded raster will be aggregated by a factor of 25, given the limitations of the current hardware specification, with regards to the file size. The factor level has been determined after several attempts to reduce the resolution of the file, leading to a factor of 25 being the lowest possible aggregation for the hill-shaded raster.

An initial local thresholding will be applied to separate the black and white regions, with a maximal threshold level of 0.5 to extract the vegetated areas from the raster. Edge detection methods, which include the Sobel Edge detector and the Laplace Edge detector, will consequently be applied to compute edges which are assumed to outline tree crowns, as shown in the work of Karimulla & Raja (2016). Associated error measures will then be computed before re-projecting the raster to a new grid. The chosen grid will refer to the EPSG:27700, which is associated with the United Kingdom, where pixels will be projected to a km based space instead of a latitude and longitude grid. Adjustments will also be made such that a pixel represents an area of 10m by 10m, which has been assumed to be the typical tree crown diameter for conifers in Scotland.

A frequency count of pixels will consequently be provided for both edge detection methods. Finally, the aforementioned procedure will be repeated to construct a distribution of the different tree counts available based on different threshold levels to build a CI, using bootstrapping techniques, for the tree count estimates using the different edge detection methods.

4.2.2 Orthophoto work stream

The orthophoto workstream will start by aggregating the multi spectral bands, specifically Red, Green, Blue, RE and NIR, by a factor of 45. This step was required for the same reasons mentioned for the DSM workstream, and the factor level was determined in the same manner as for the DSM work stream, due to hardware limitations.

The NDVI and the EVI will be computed from the aggregated multi spectral bands based on the previously presented formulas and stacked to as the basis for clustering the vegetation areas. The choice of using these vegetation indexes for the clustering of regions was based on their ability to identify vegetation from the ground, as shown in the literature, and thus potentially form well defined clusters as well as training labels. The K-means clustering algorithm will then be applied to the the aggregated vegetation indexes to identify tree covered and non-tree covered regions, with a view to form training labels to be used in the RF algorithm. Additionally, a separate tree count estimate using solely the K-means clustering method shall be provided by assigning the identified classes from the clustering results to the RE spectral band and projecting the latter to the EPSG:27700 grid. A count of the pixels belonging to the tree covered thematic raster will be calculated and then provided as the count from the unsupervised approach.

For the semi-supervised learning work stream, the two best clusters representing tree covered and non-tree covered regions will consequently be chosen and used to mask the aggregated multispectral bands to obtain the training data. The RF model will subsequently be trained using the masked multispectral bands, as the predictor variable to estimate the classification of the labels identified through kmeans, using the NDVI and EVI index. The landcover classification will be predicted based on the aggregated raster bands, and the same reprojection procedure as for the DSM will be applied. The new datum will refer to EPSG:27700, the distance metric will be specified to be in km, and the resolution will be modified to represent 10m by 10m per pixel. Tree count estimates will then be provided based on the model's predictions.

4.3 RF Tuning

Thornton et al. (2013) assert that tuning involves automatically specifying the parameters of a model such that it optimises an objective function. For instance, a RF can be tuned on a training dataset to find the lowest Root Mean Squared Error (RMSE) among the possible model configurations. The object of tuning in machine learning approaches is to allow for the model to be applied to unseen datasets while still maintaining an acceptable degree of accuracy. Therefore, there is a consideration for the model not to memorize the underlying signal that generated the training data perfectly.

For the purposes of this study, given that the underlying model configuration will not be applied to other rasters, it is desirable for the model to perfectly model the underlying signal in the training data and thus over fit. The model specification employed will thus fitted with the default parameter values, with the exception of the *mtry* parameter that shall be obtained through an OOB tuning procedure. It is assumed that the latter parameters will allow for a model to be intricate enough to perfectly model the training data.

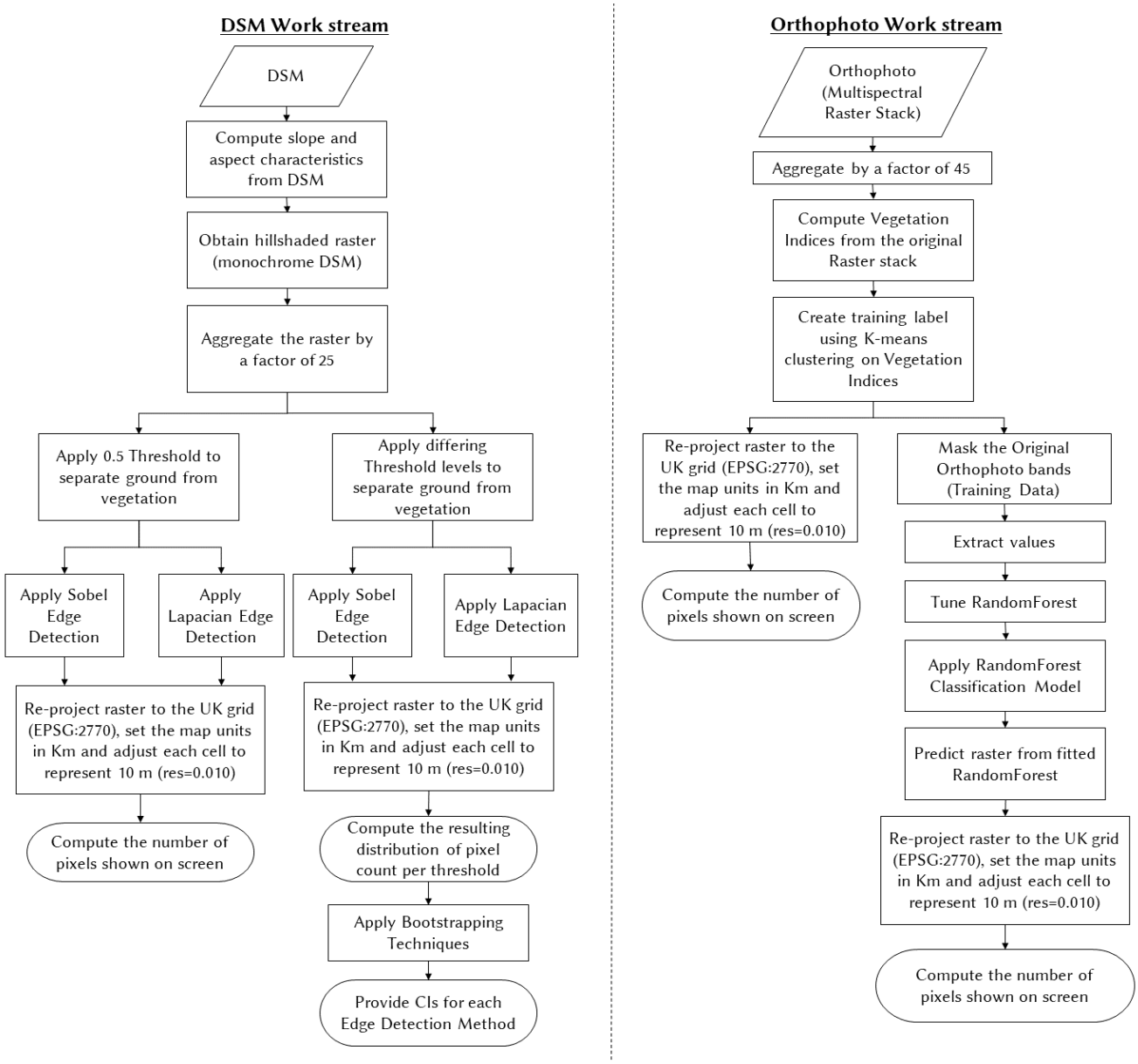


Figure 4.1: *Proposed methodology*

Chapter 5

Empirical Results

The aim of this section is to present an overview of the results obtained from the use of both work flows defined in the methodology and a discussion with regards to the latter. To do so, the definition of the error measures used for the estimation of the considered methods' accuracy will be provided before expanding upon the results from the considered object oriented methods and machine learning algorithms.

5.1 Definition of Errors used

Given that quantitative use of images or rather the measurements which stem from such use in this context, a clear presentation of how uncertainty is spread across those measures is needed as well as an indication of how hardware and software might influence its propagation. As shown by Tappan et al. (1987), sources of error that might arise and which are relevant to this study might include: the pixel aspect ratios, the variation with screen location and the grey scale range definition.

5.1.1 Edge Detection Error Measures

Sadykova & James (2017) outlines that the efficiency of filters is typically defined by the overall image quality produced after their application. Therefore, measures such as the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) will be considered for an assessment of the performance of the edge detection methods used.

5.1.1.1 MSE

The MSE can be defined as the averaged squared difference in the pixel intensity between the original and transformed image. Consider that the image size, in terms of length and width, is represented by M and N, while $I_1(s,t)$ and $I_2(s,t)$ represent pixel locations on the original and transformed images. The MSE equation can consequently be presented as follows:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(I_1(i,j) - I_2(i,j))^2 \tag{5.1}$$

5.1.1.2 PSNR

A different measure that relies on the MSE but allows for identification of the level of loss with regards to signal integrity is the PSNR (Wang et al. 2004). The PSNR is a measure of signal loss from the original image I by using the MSE as a denominator. The latter measure can be calculated as follows:

$$PNSR = 10 \log \left(\frac{max(I)^2}{MSE} \right) \tag{5.2}$$

5.1.2 Machine learning Error Measures

Given the nature of the study as a classification problem, from a machine learning standpoint, error measures for discrete classes will be considered. Consider a classifier algorithm that will allocate the input data to one of two classes, either Positive or Negative. With regards to the algorithm's predictions, there might be positive classes wrongly allocated as negatives and vice versa. This leads to the issue of True Positive (TP) and False Positive (FP), in addition to True Negative (TN) and False Negative (FN), such that each class is independent from each other, but their components are dependent on the ground truth (training data) and the associated predictions (predicted classifier values).

With reference to a thematic raster, the ground truth would be defined as the original class a pixel belongs to, such as "Vegetation", and predictions would relate to whether the pixel is actually defined as "vegetation" by the classification algorithm . The aforementioned classification measures can thus be presented as a confusion matrix and are defined as follows, when considering their application to rasters:

- TP: a pixel is classified as belonging to a class while the ground truth is positive;
- FP: a pixel is classified as not belonging to a class while the ground truth is positive;
- TN : a pixel is classified as not belonging to a class while the ground truth is negative;
- FN : a pixel is classified as belonging to a class while the ground truth is negative.

Consequently, the True Positive Rate , or Sensitivity, coupled with the True Negative Rate, or Specificity, of the classifier can be computed as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.3}$$

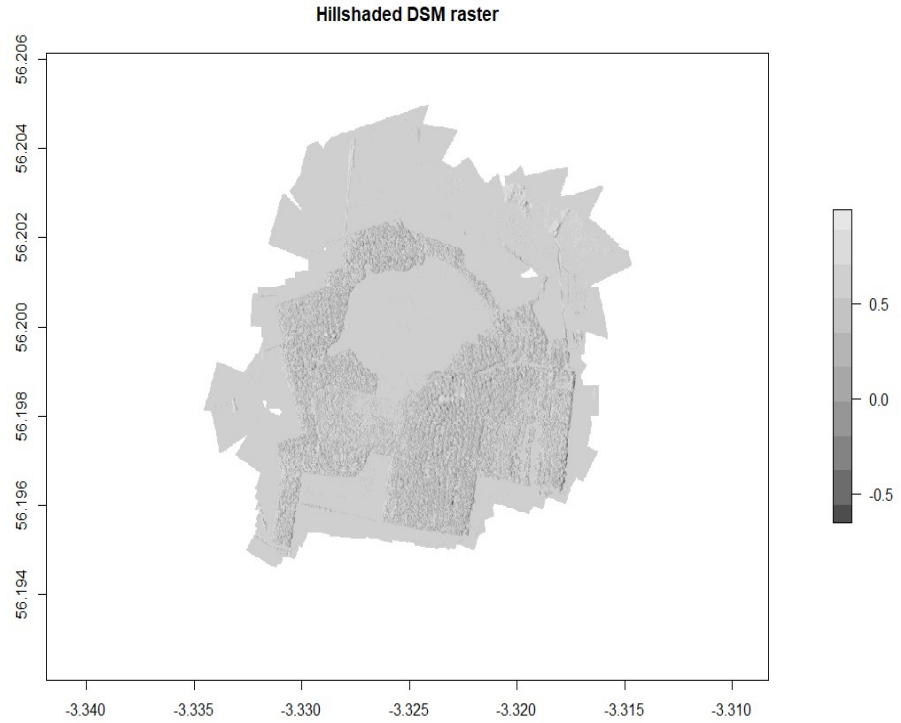
$$Specificity = \frac{TN}{FP + TN} \tag{5.4}$$

The Sensitivity and Specificity are thus indicative of a classifier's ability to correctly predict positive and negative classes respectively. These measures can be plotted against each other to obtain the Receiver Operating Characteristic curve (ROC), whose Area under the curve (AUC) can also be used as a measure of overall model accuracy. It should however be noted that the ROC is typically defined for a range of Thresholds, T, which influences the predictions of the model. Specifically, as per Wang (2016), the value of the threshold will define the sensitivity and the specificity such that:

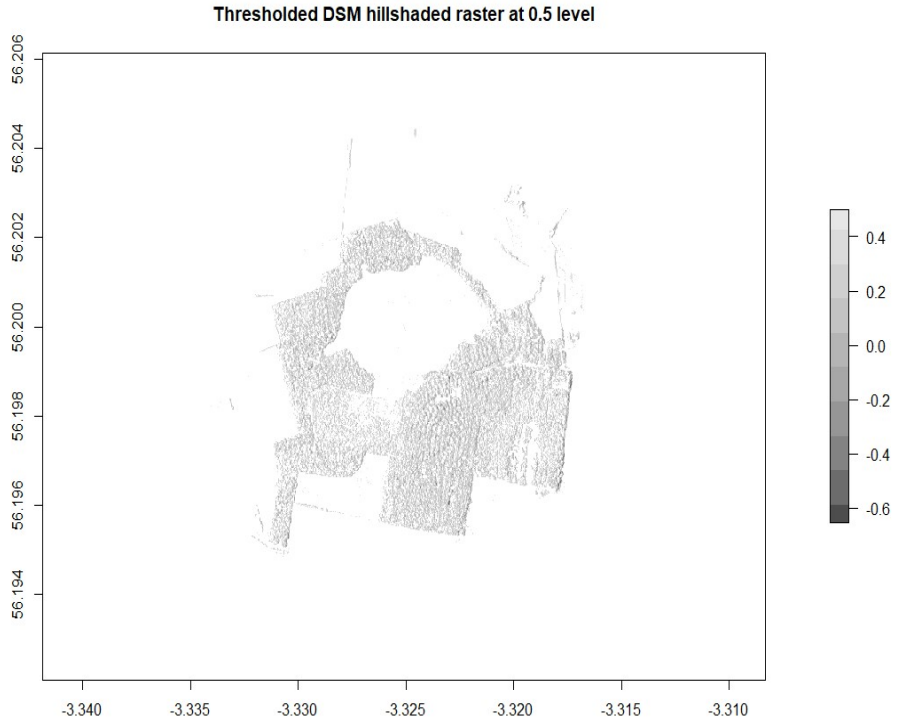
$$\begin{cases} T \to \infty & Sensitivity = Specificity = 0\% \\ T \to -\infty & Sensitivity = Specificity = 100\% \\ Otherwise & 0\% < Sensitivity < 100\% \\ & 0\% < Specificity < 100\% \end{cases}$$

5.2 Edge detection results

As per the DSM workstream, the hillshaded DSM has been first estimated, using a slope of 45 degrees for the elevation of the sun's light and an angle of 270 radians for the light's associated direction, or aspect. According to Horn (1981), 8 neighboring cells were used to compute the slope for the terrain, under the assumption that the PortMoak Moss' surface is relatively rough. An arbitrary threshold was applied to exclude the values greater than 0.5, which are assumed to represent the ground, as indicated by the figure 5.1. Subsequently, the considered edge detectors were applied to the thresholded hillshaded raster and used to compute the sum of bounded pixels through zonal statistics.



(a) Hillshaded DSM



(b) Thresholded Hillshaded DSM at 0.5 level

Figure 5.1: *Hillshading results*

5.2.1 Sobel Edge Detector Estimate

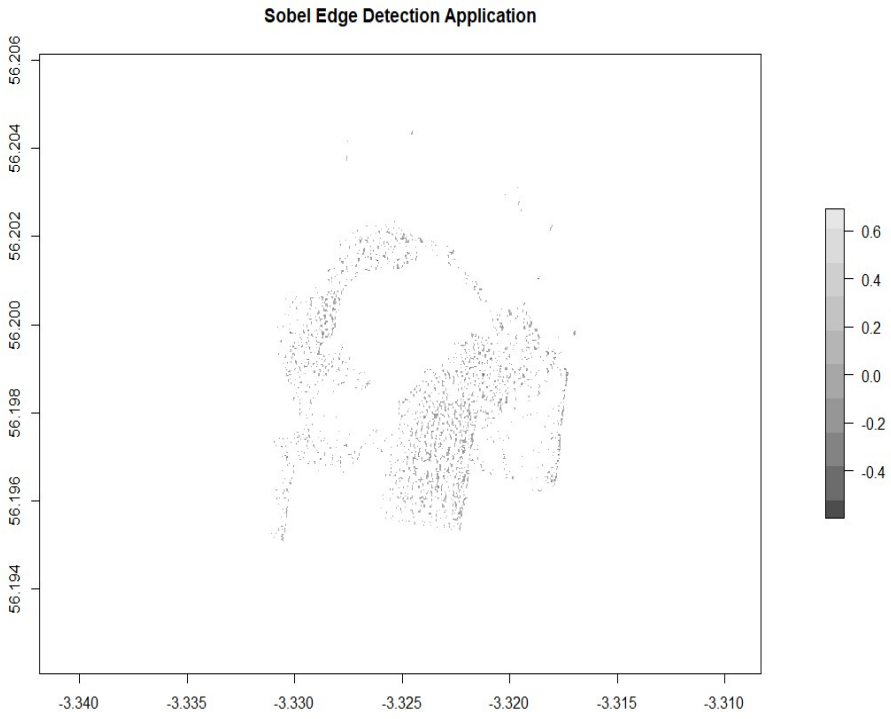
The use of the Sobel Edge detector on the aggregated hillshaded raster has led to a MSE of approximately 0.089 as well as a PSNR of 4.70, and substantial changes between the original and transformed image in their inherent pixel distributions, as shown in Table 5.1 and Figure 5.2.

Table 5.1: Distribution changes arising from Sobel Edge Detection

Attribute	**Original Values**	**Sobel Edge Detection Results**
Minimum raster values	-0.6543819	-0.6702429
Maximum raster values	0.9939324	0.6910431
Rounded value	**Original Pixel count**	**Sobel Edge Detection Pixel count**
-1	44	14
0	73,605	8,809
1	225,593	10
NA	543,470	83,3879

The above disparity can be explained by the application of the zonal statistic function, which considers only the sum of the values in the bounded regions created by the Sobel Edge Detector, while excluding all the other values. The summed values thus define the centres of their bounded regions, which are also assumed to have highest values within their bounds. The Sobel raster was then reprojected to the UK grid with each cell representing a region of 10m by 10m. Assuming that the trees in PortMoak have an average tree crown diameter of 10m, the pixel count of the resulting reprojected raster will provide the estimated tree number according to the rounded values of the raster. A frequency count of the values close to zero in the reprojected raster reveals an estimated tree count of 579 trees, as shown in Table 7.1 of the Appendices.

(a) Thresholded Hillshaded DSM at 0.5 level



(b) Sobel Edge Detection results

Figure 5.2: *Application of Sobel Edge detector on the Thresholded Hillshaded DSM at 0.5 level*

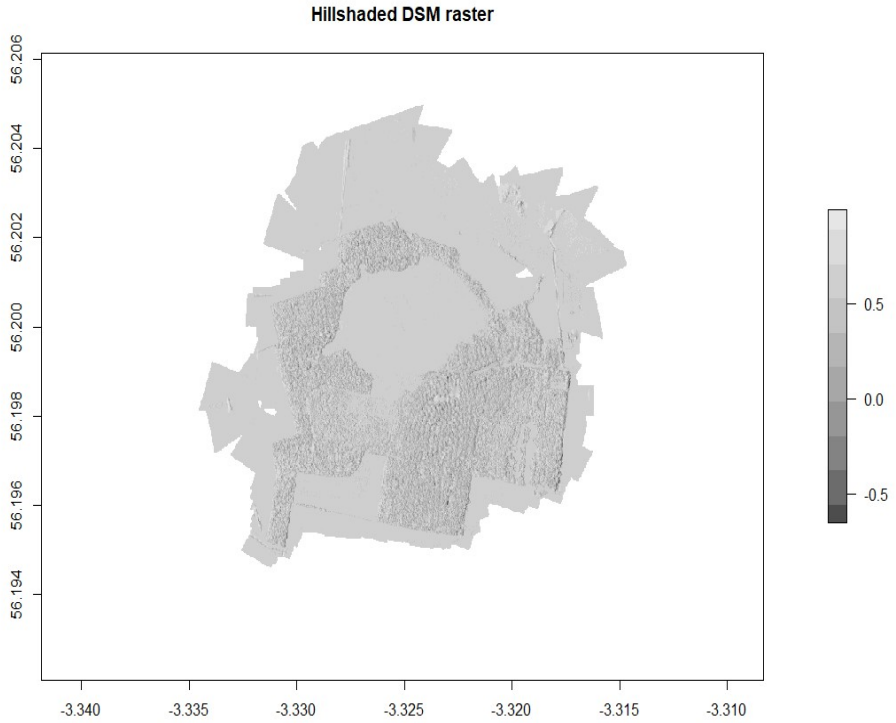
5.2.2 Laplace Edge Detector Estimate

The application of the Laplace Edge Detector Estimate on the aggregated hill shaded raster has resulted in a MSE of 0.628 coupled with a PNSR of 22, indicating a more pronounced difference between the original and the transformed image than for the Sobel Edge Detector, as shown in the figure 5.3.

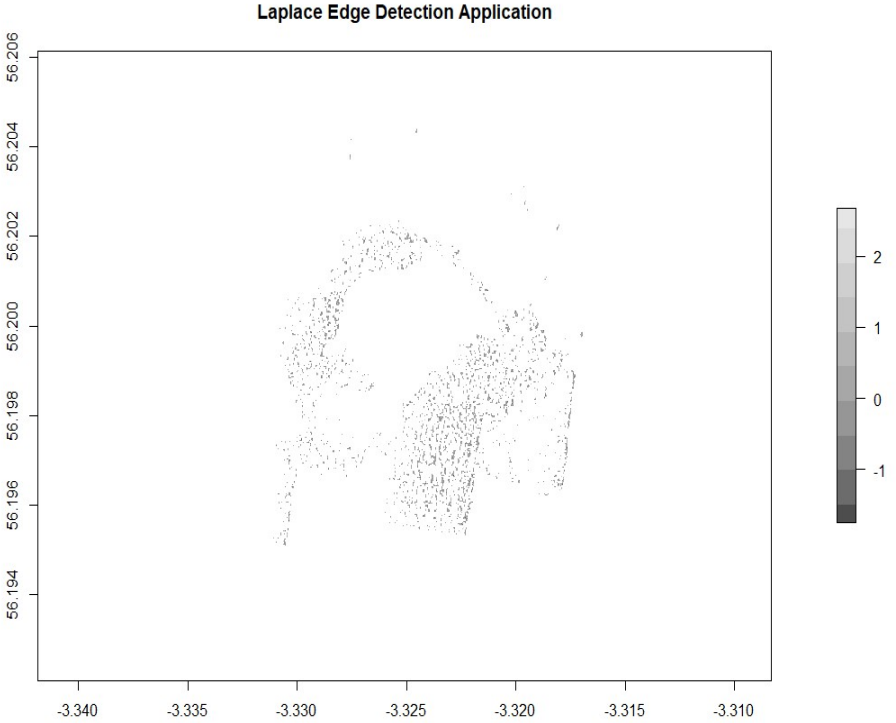
Table 5.2: Distribution changes arising from Laplace Edge Detection

Attribute	**Original Values**	**Laplace Edge Detection Results**
Minimum raster values	-0.6543819	-2.187541
Maximum raster values	0.9939324	2.675293
Rounded value	**Original Pixel count**	**Laplace Edge Detection Pixel count**
-2	0	19
-1	44	1,458
0	73,605	5,026
1	225,593	2,169
2	0	157
3	0	4
NA	543,470	833,879

The difference between the original and Laplace values can be justified by the Laplace's kernel sensitivity to noise. As previously mentioned, the use of the second derivative in the Laplace kernel makes it more sensitive to detect discontinuities in an image. This has consequently led to the identification of a wider range of pixel intensity values than the Sobel operator. After reprojecting the raster to the UK grid with each pixel characterising a 10m by 10m area and under the same assumption as before, a frequency count of values close to zero provides an estimated count of 449 trees, as shown in Table 7.2 of the Appendices.



(a) Thresholded Hillshaded DSM at 0.5 level



(b) Laplace Edge Detection results

Figure 5.3: *Application of Laplace Edge detector on the Thresholded Hillshaded DSM at 0.5 level*

5.2.3 Comparison of Edge Detectors and associated Confidence Intervals

From the previous results and methodology, it can be seen that the number of tree counts is dependent on the maximum threshold value that has been used to delimit the aggregated hillshaded raster. Thus, to ascertain the margin of error associated with the determination of the threshold, a CI shall be constructed for the Sobel and Laplace estimates. This procedure shall be implemented using the different methods outlined in Chapter 4, after calculating the range of possible tree counts from each edge detector after applying different threshold values between 0.1 and 1, with a step size of 0.01. The different estimated tree counts obtained for the Sobel and Laplace operator after the determination of the various threshold values are illustrated in the Figure 5.4.

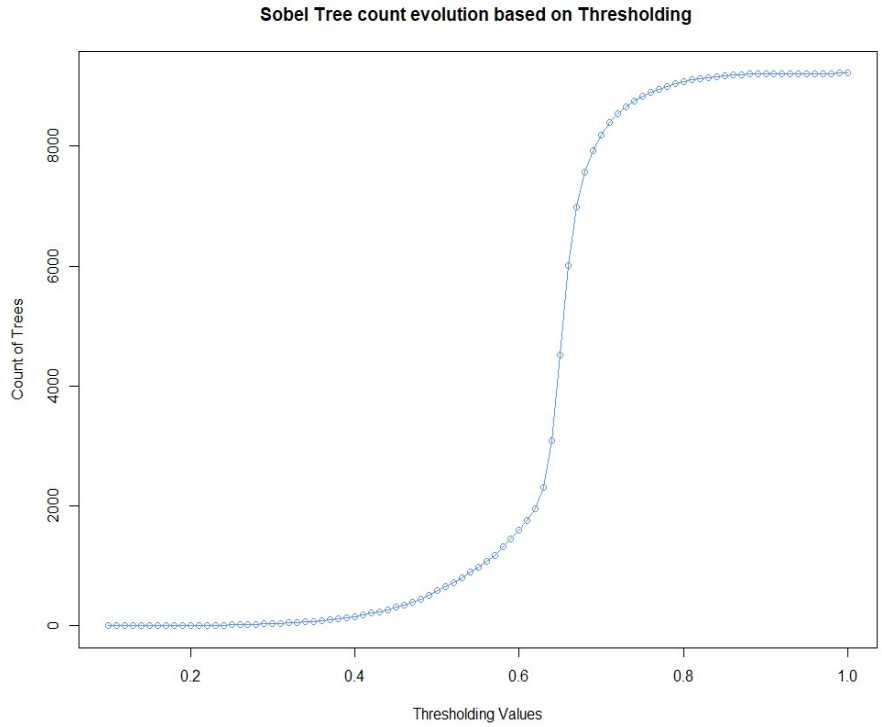
However, as it can be seen from the distribution's shape in figure 5.4, the number of trees grows drastically beyond the threshold value of 0.6, which includes ground pixel counts. Thus for a more accurate confidence interval, an estimation of the tree counts based on maximum thresholding values lying between 0.1 and 0.6 will be computed with a step size of 0.001 (refer to Figure 5.5).

5.2.3.1 Estimated CI

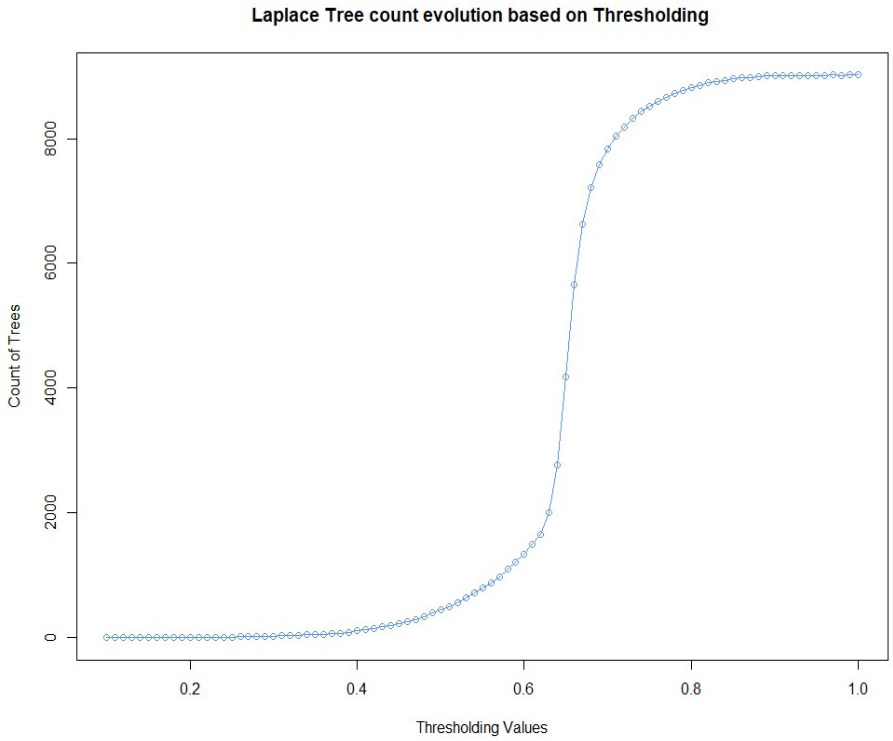
The below Table illustrates the results from the considered CI methods, where a 5% trimmed mean was computed from 2000 resamples from the values obtained from the previously calculated tree count estimates, obtained by applying maximum thresholding values lying between 0.1 and 0.6 with a step size of 0.001. The results indicate that the Laplace Edge Detector provides smaller CI than the Sobel detection with the Normal and BCA methods, with the exception of the Percentile method where the Sobel has a larger CI. It is also noted that when comparing the different methods, namely the Normal, Percentile and BCA methods, the percentile and BCA methods provide close estimates, indicating the re-sampled distribution has a fairly symmetric shape that did not require large acceleration and bias correction parameters for the CI estimation.

Table 5.3: 95% CI for Edge Detector estimates

95% CI (3 sf)	**Sobel Edge Detection**			**Laplace Edge Detection**		
	Normal	*Percentile*	*BCA*	*Normal*	*Percentile*	*BCA*
Lower Limit	250	201	202	196	154	155
Upper Limit	321	279	280	255	216	218

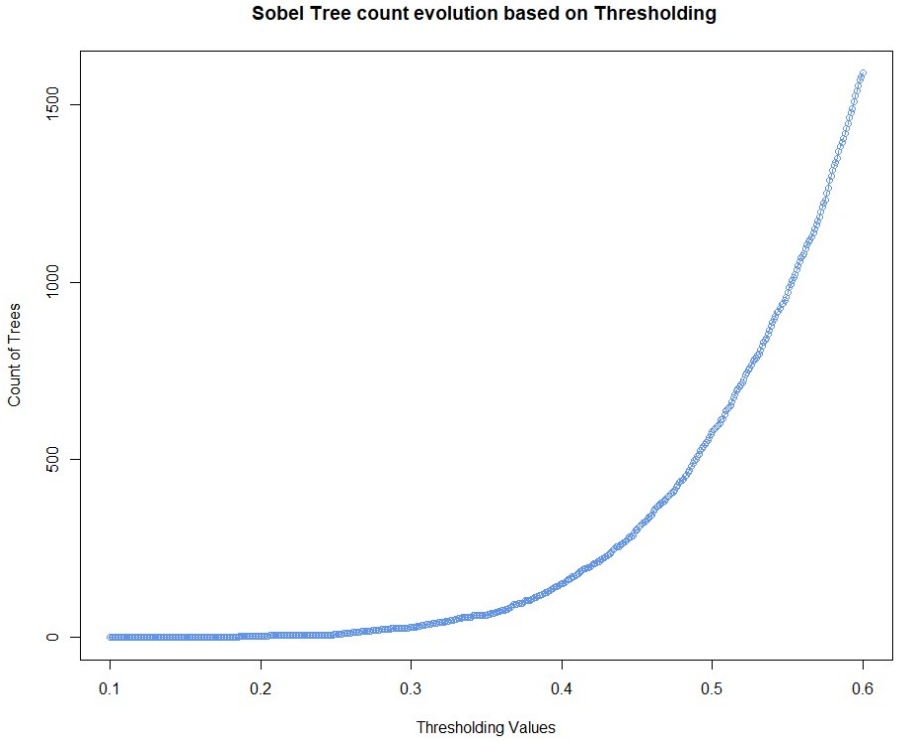


(a) Sobel Estimated counts

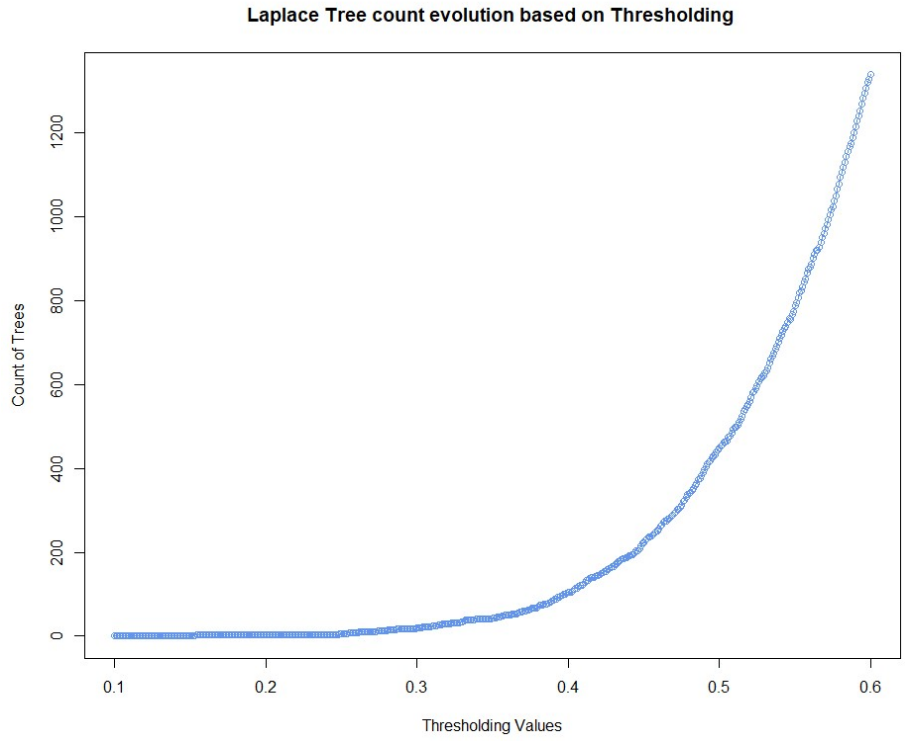


(b) Laplace Estimated counts

Figure 5.4: *Estimated tree counts based on Threshold levels between 0.1 and 1 (0.01 step size)*



(a) Sobel Estimated counts



(b) Laplace Estimated counts

Figure 5.5: *Estimated tree counts based on Threshold levels between 0.1 and 0.6 (0.001 step size)*

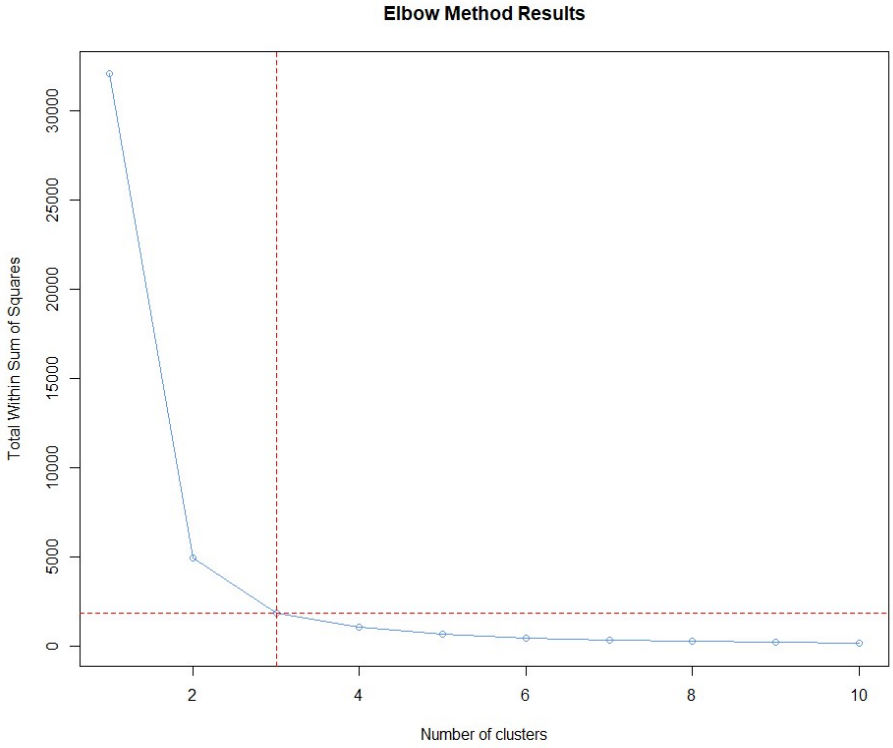
5.3 Unsupervised learning results

This section will depict the results obtained from the first part of the Orthophoto workstream, precisely, the clustering results from the application of the K-means algorithm on the NDVI and EVI . The determination of the number of clusters to be used for the analysis has been carried out solely through the use of the Elbow method, since the GAP statistic could not be computed as a result of hardware limitations. In this regard, the TWSS was computed for 2 to 10 clusters and plotted to find the elbow, indicating the point where diminishing returns occur as the number of clusters increases. The Elbow method plot indicates that 3 clusters (with a TWSS of 1840.68) are the inflexion point at which the increasing complexity of the clustering is not anymore justified, thus indicating that 3 clusters should be optimal to create a thematic raster. The resulting thematic raster, containing the training labels for the RF, the frequency count of pixels across the latter and the TWSS plot are shown below.

From the above, it can be seen that class 1 is indicative of the vegetated or tree covered areas. A reprojection of the above thematic raster to the UK grid with the same resolution of 10m, as before, has been carried out to allow for a tree count determined solely using K-means clustering. The resulting pixel count for class 1 indicates that the K-means method identifies **3,697** potential trees in the region, as seen in the the pre and post projection pixel count Table (ref to Table 5.4).

Table 5.4: Frequency count of pixels across thematic raster classes (pre and post projection)

Pixel Class	Pre-projection pixel count	Post-projection pixel count
1	37,011	3,697
2	39,762	4,048
3	65,577	6,551
NA	0	3,112



(a) TWSS plot for Elbow Method

(b) Classified RE raster, based on NDVI and EVI clustering results

Figure 5.6: *3 cluster K-means results*

5.4 Semi-Supervised learning results

This section will highlight the results obtained from the second part of the Orthophoto workstream, namely the classification results from the application of the RF algorithm. The result of the tuning procedure using the OOB method reveals that 4 variables should be randomly sampled at each tree split. The RF has thus been trained using the 5 multispectral original bands and the training labels obtained from the K-means method with the below parameters and after excluding class 3, such that the RF considers a binary classification problem.

Table 5.5: RF Parameter Specification

Parameter name	Description
mtry	4
nodesize	5
maxnodes	None
ntree	500

The predicted reprojected raster, shown in the Appendices, indicates through the frequency of the associated classes, that class 1 (assumed to represent tree cover), contains **3,684** pixels. The latter is a close approximation to the K-means number, indicating that the RF provides a count close to that using solely an unsupervised method. The following confusion matrix, obtained by setting a threshold of 0.5, and the associated AUC plot confirm that the RF almost perfectly models the underlying signal in the training labels, originally defined by the clustering method.

Table 5.6: RF Confusion Matrix with 0.5 Threshold

	Trees	Non-Trees	Classification error
Trees	36,701	310	0.008
Non-Trees	218	39,544	0.005

Table 5.7: Frequency count of pixels across predicted RF raster at 0.5 Threshold

Pixel Class	Predicted pixel count frequency per class
1	3,684
2	10,667
NA	3,858

Given the high associated count, the threshold level has been modified to give a lesser chance of a pixel being classified as class one and thus obtain a lower tree count which might be closer to the clustering approximation. A threshold of 0.1 results in an estimated tree count of **3,598** and in the following confusion matrix.

Table 5.8: RF Confusion Matrix with 0.1 Threshold

	Trees	**Non-Trees**	**Classification error**
Trees	35,418	1,593	4.304126e-02
Non-Trees	1	39,761	2.514964e-05

Table 5.9: Frequency count of pixels across predicted RF raster at 0.1 Threshold

Pixel Class	**Predicted pixel count frequency per class**
1	3,598
2	10,75
NA	3858

Additionally, the RF indicates that contrary to the literature, the Red image band has been the most important contributor to the predictions as per the variable importance plot, shown in Figure 8.13 of the Appendices.

Chapter 6

Conclusion

6.1 Concluding Remarks

This thesis has examined two ways of estimating tree counts, in the absence of a DTM and training polygons. The methods considered related to edge detection methods applied on the DSM and a semi-supervised learning framework applied on the orthophoto. With reference to the first defined interval of 436 to 9,687 trees, one could observe that only the RF estimates fall in the latter interval.

Consequently, it can be seen that the semi-supervised framework provides a tree count that falls into the assumed interval, with the K-means clustering method succeeding in segmenting the raster into classes that the RF predicts with near perfect accuracy. Conversely, the Edge detection methods reveal a much lower estimated tree count. This more conservative estimation might be the result of an inadequate use of the hillshading technique, potentially caused by the differential in the unit of measurement since the latter was applied on a raster with units in inches rather metres. Furthermore, the use of a function other than 'sum' for edge detector based focal statistics might have led to more accurate results. Furthermore, it should be underlined that the obtained CI for the DSM workstream was highly dependent on a range of threshold values chosen for the application of bootstrapping techniques. Thus, should a different range have been chosen, say 0.7 to 1.0, a more realistic CI could have been obtained.

In conclusion, the RF provides a near perfect prediction accuracy based on the training samples, defined by what appears to be an efficient use of the K-means algorithm. The edge detector methods' lead to lower estimates associated bootstrapped CI as a result of the inhibited efficiency of the methods in the face of the provided dataset.

6.2 Limitations

As with any study, several challenges and limitations that inhibited the effectiveness of the considered methods were experienced. The main limitations faced related mostly to the hardware used for the study. Given the previously defined hardware specifications, as well as the size of the files studied, several other considered methods were disregarded. With respect to the DSM work stream, the Canny edge detector was discarded while for the Orthophoto worksream, methods such as the GAP statistic could not applied to the analysis, as well as more complex classifiers. Additionally, the aforementioned hardware required the very high resolution of the orthophoto to be reduced, with differing levels of aggregation, for processing in the proposed methodology. This reduction has thus impacted the efficiency of the considered methods since less detailed and defined features are present in the reduced data. This has also prevented the use of the zoom function that would have allowed to see the edge detected trees in a clear manner, due to the resolution loss.

Furthermore, in line with the objective of tree count estimation, conventional tree detection methods such as CHM have not been used as a result of DTM and LIDAR cloud point data being unavailable. The latter would have allowed for a more precise tree count, which would not have relied on the central assumption of this thesis where trees were estimated to have a tree crown diameter of 10m. Additionally, the unit of measurement of the provided DSM raster inhibited the efficiency of the hill shading techniques, which would have led to more precise estimates for the edge detection methods. Finally, it should be outlined that the use of semi-supervised methods was warranted since no pre-defined training label was available, due to the area having no related field measurement for the canopy composition. Therefore, the best estimates would have been obtained in the event of an existing forest inventory for the PortMoak Bog area such that the latter could have been used as the training data.

6.3 Further Research

Further research in the area of tree count estimation, using the R software suite, could be geared towards the application of more intricate classifiers such as deep learning models to delineate tree crowns from orthophotos and which could potentially lead to more accurate estimates. Additionally, the estimation of a DTM from DSM could be further investigated, as a need to circumvent the DTM which can be a resource intensive endeavour. Finally, this problem could have been investigated from a different perspective, such as a species distribution problem, where tree locations as a planar x and y coordinate grid could have had their numbers and locations predicted, across the same time stamp, using similar methods.

Chapter 7

Appendices

7.1 Graphs and Tables

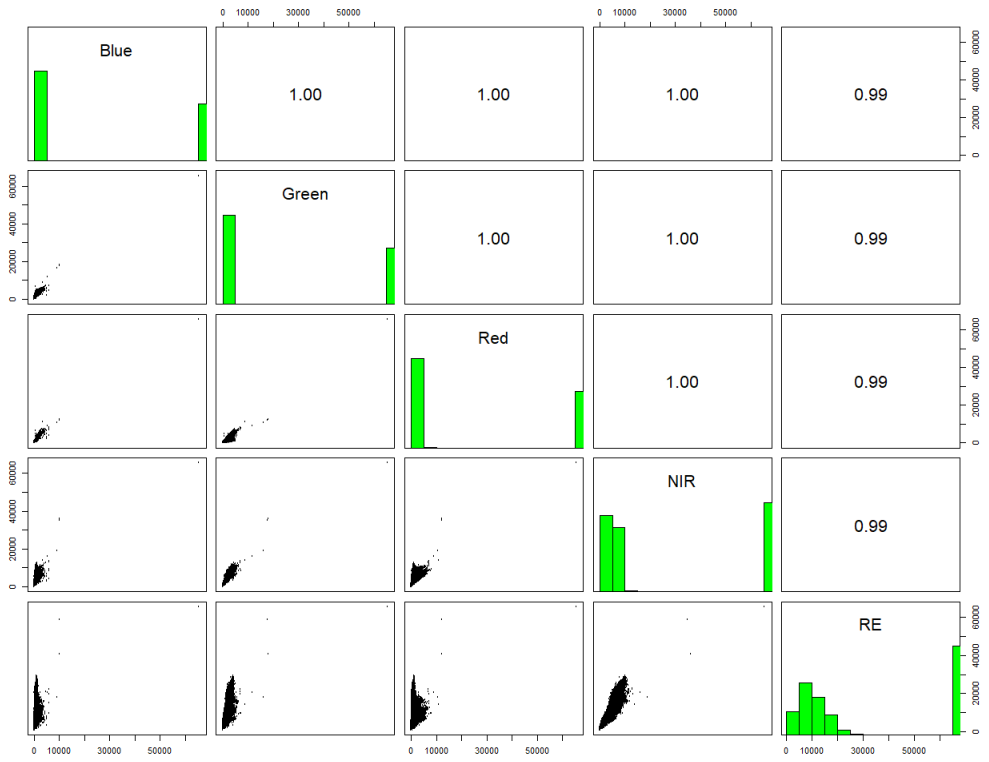


Figure 7.1: *Pairs plot of Multispectral bands*

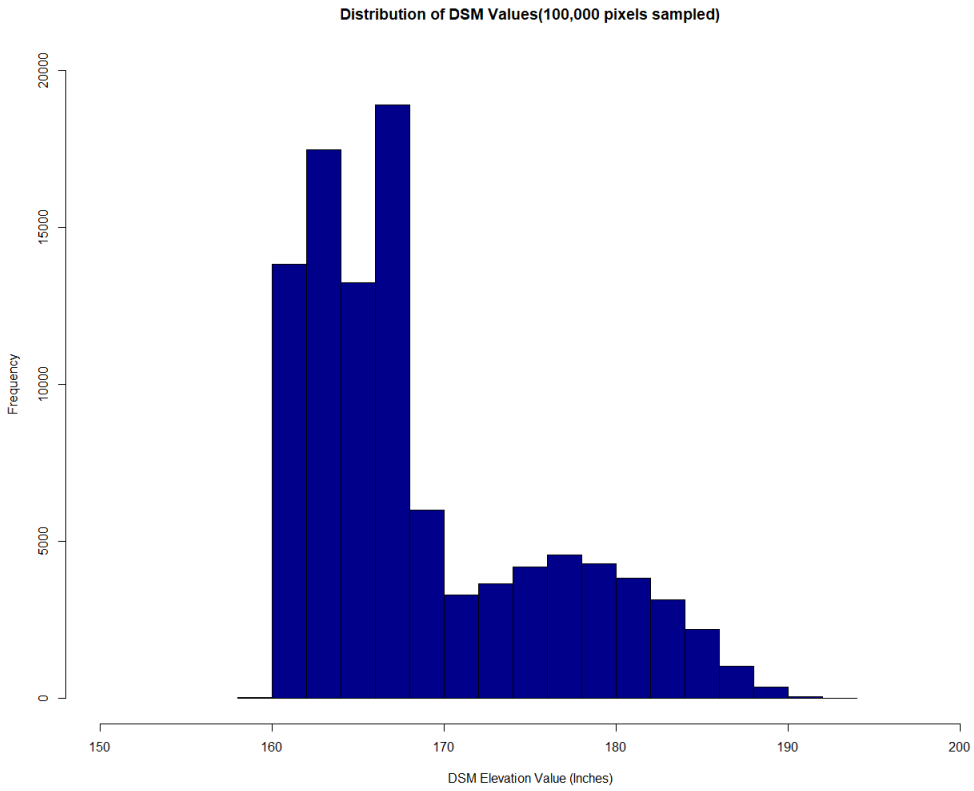


Figure 7.2: *DSM distribution of pixel intensity values*

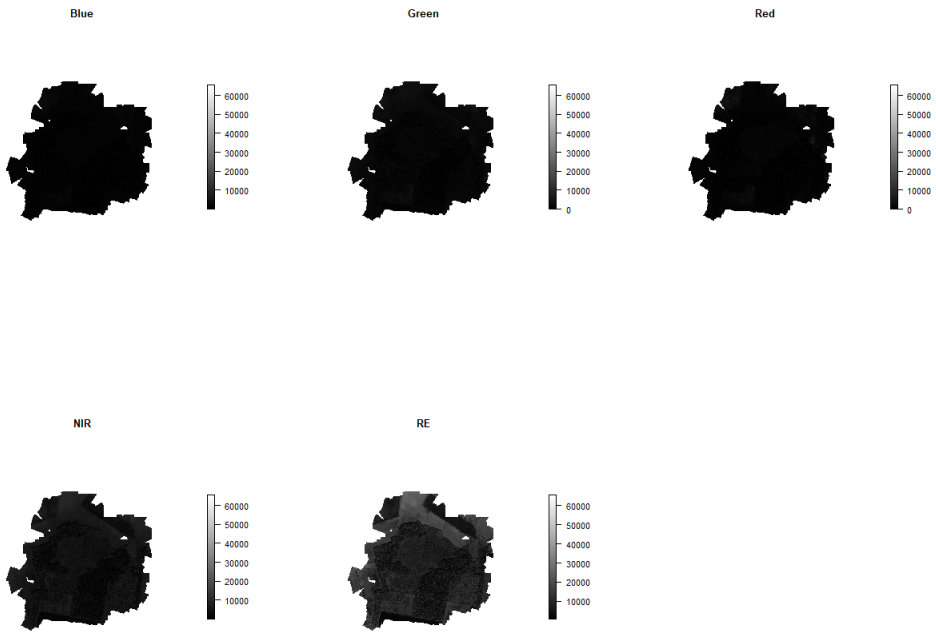


Figure 7.3: *Linear Stretch of multi-spectral bands*

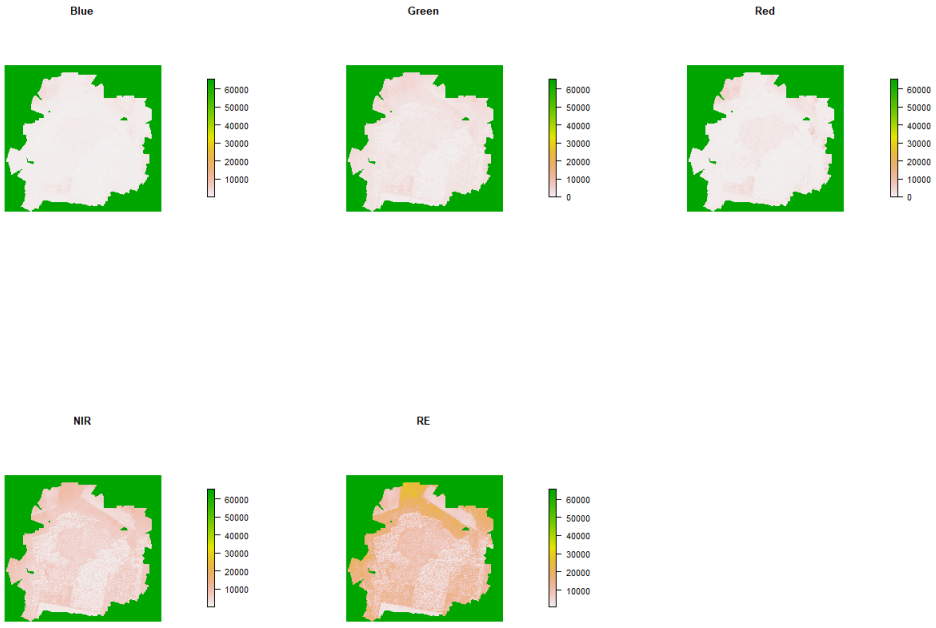


Figure 7.4: *Histogram Stretch of multi-spectral bands*

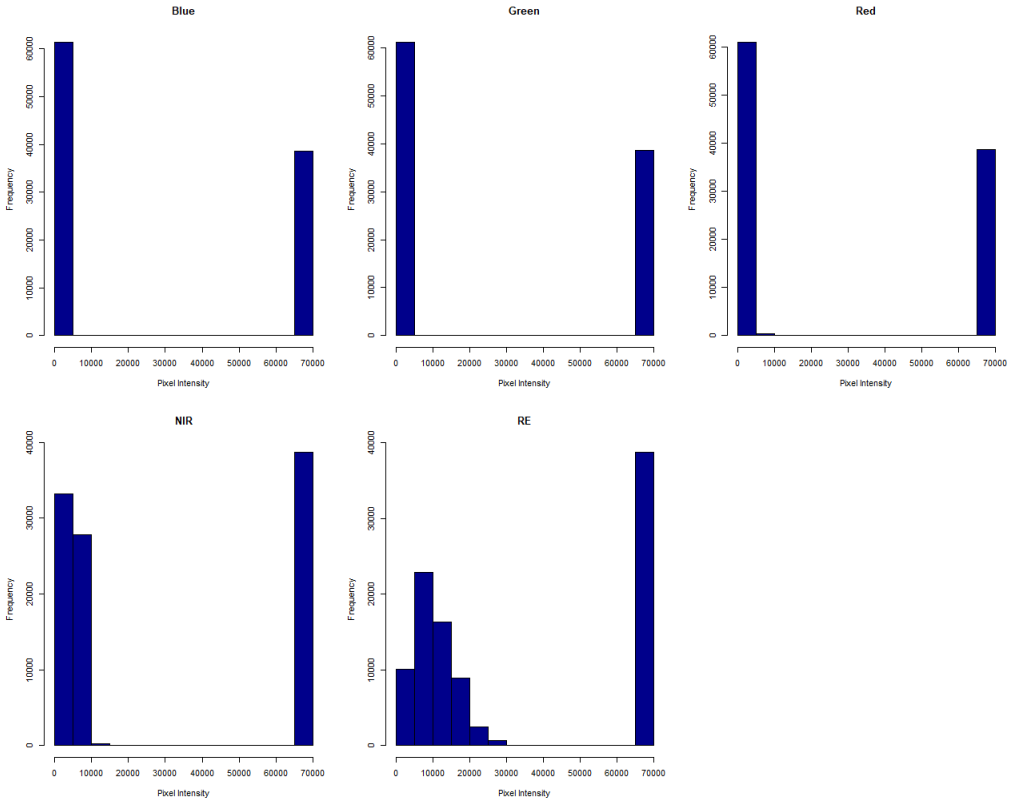


Figure 7.5: *Pixel intensity distribution across multi-spectral bands*

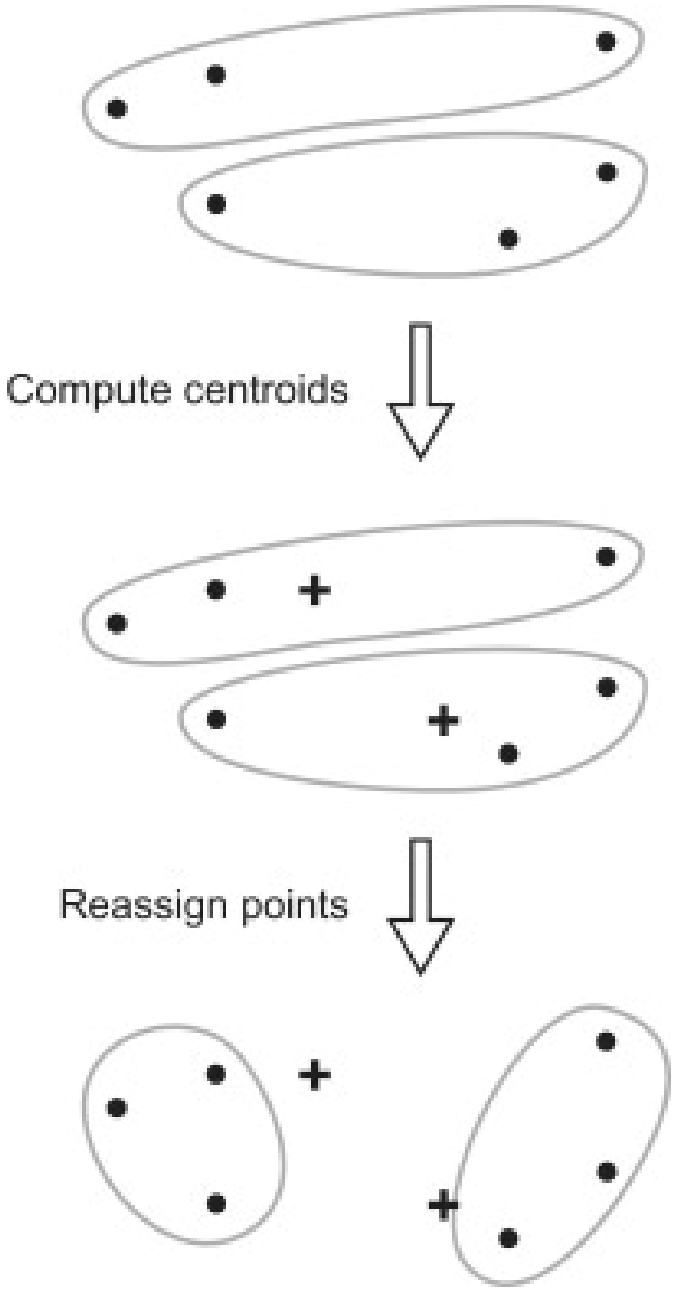
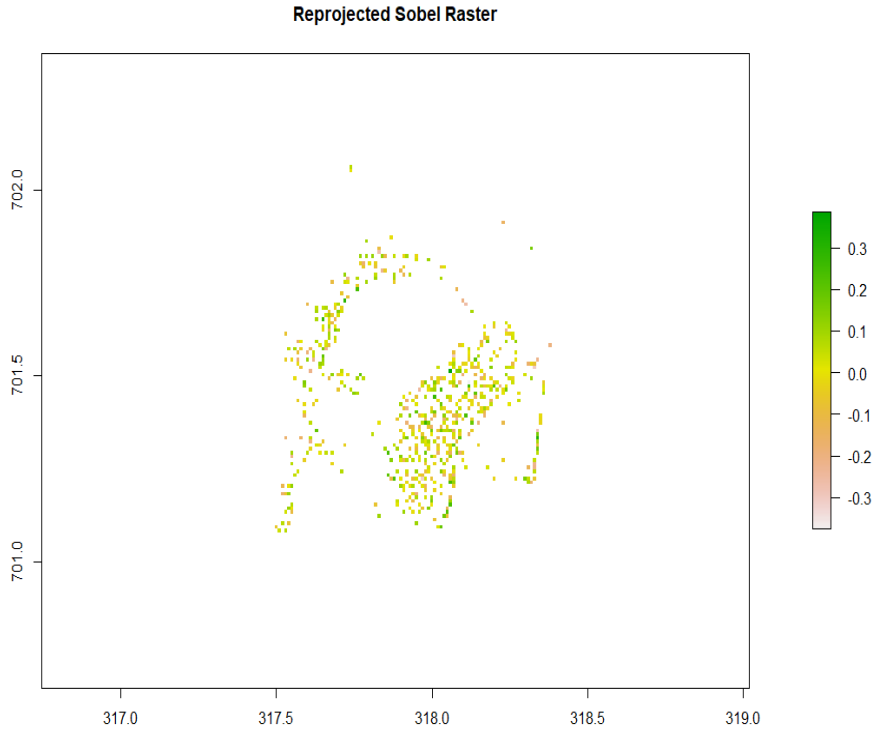
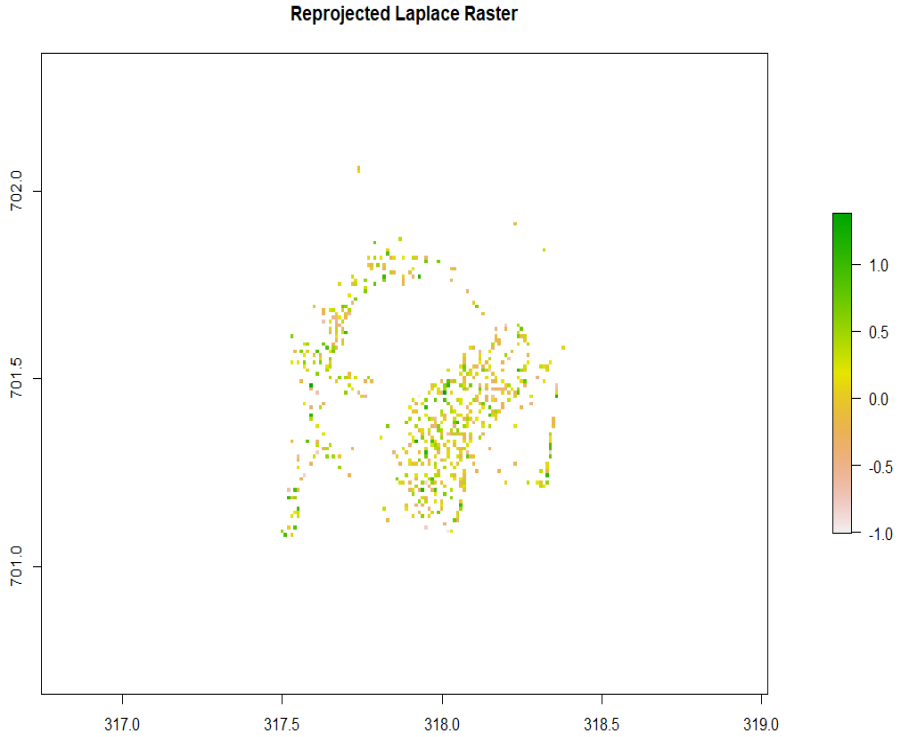


Figure 7.6: *Kmeans algorithm illustration sourced from McCool et al. (2012)*

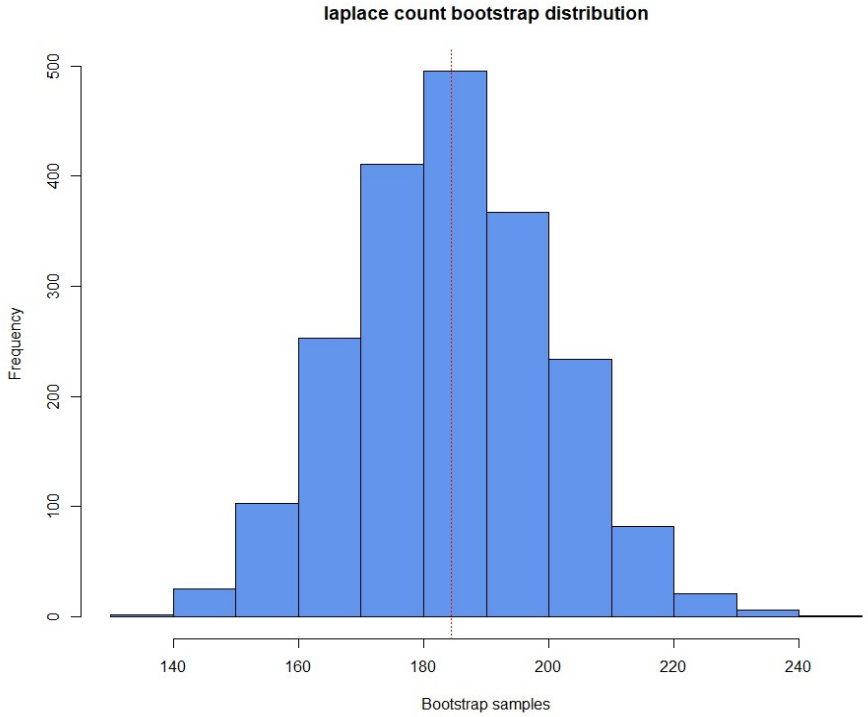


(a) Reprojected Sobel Raster

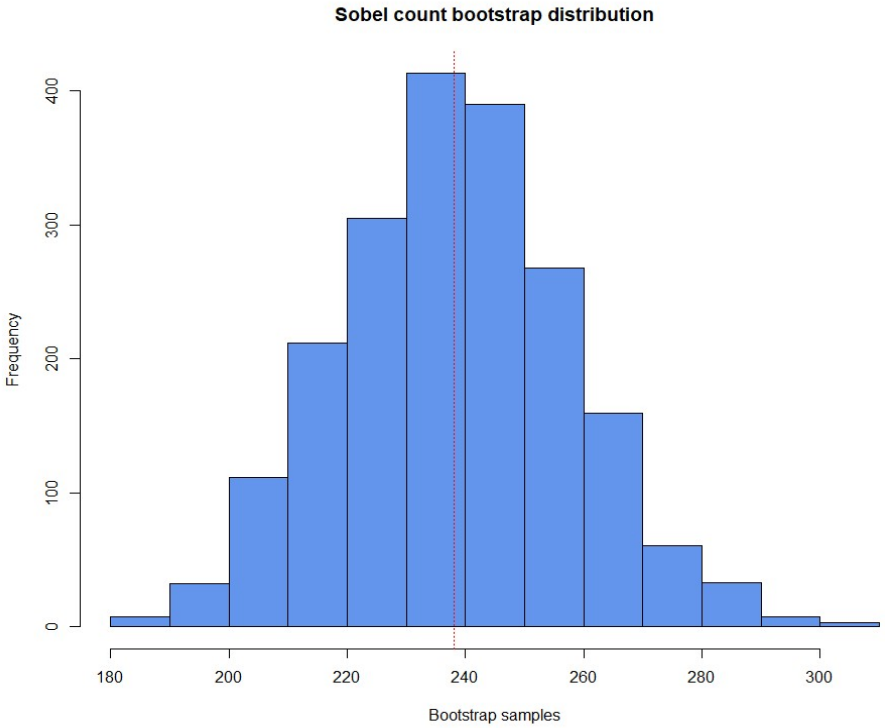


(b) Reprojected Laplace Raster

Figure 7.7: *Reprojected edge detected raster*

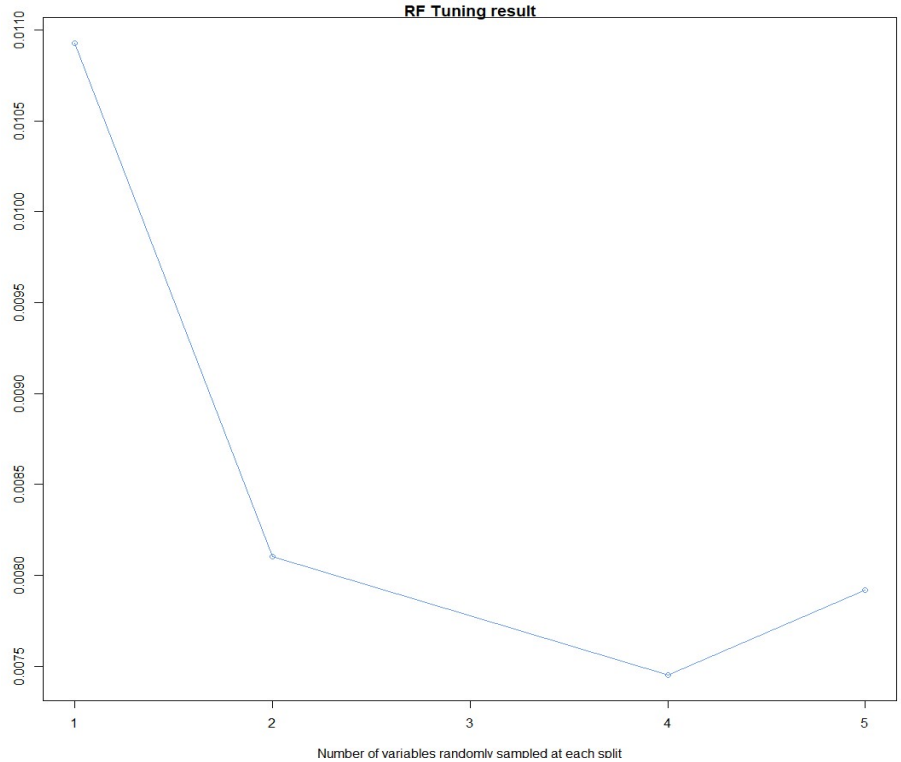


(a) Sobel Resample distribution

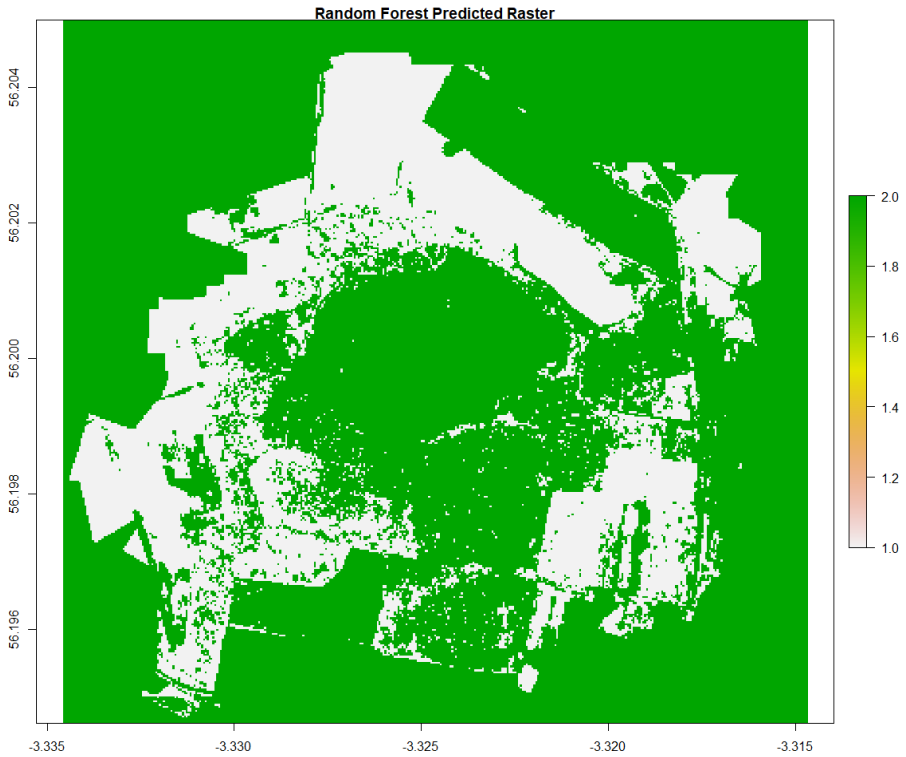


(b) Laplace Resample distribution

Figure 7.8: *Booststrap Resamples*

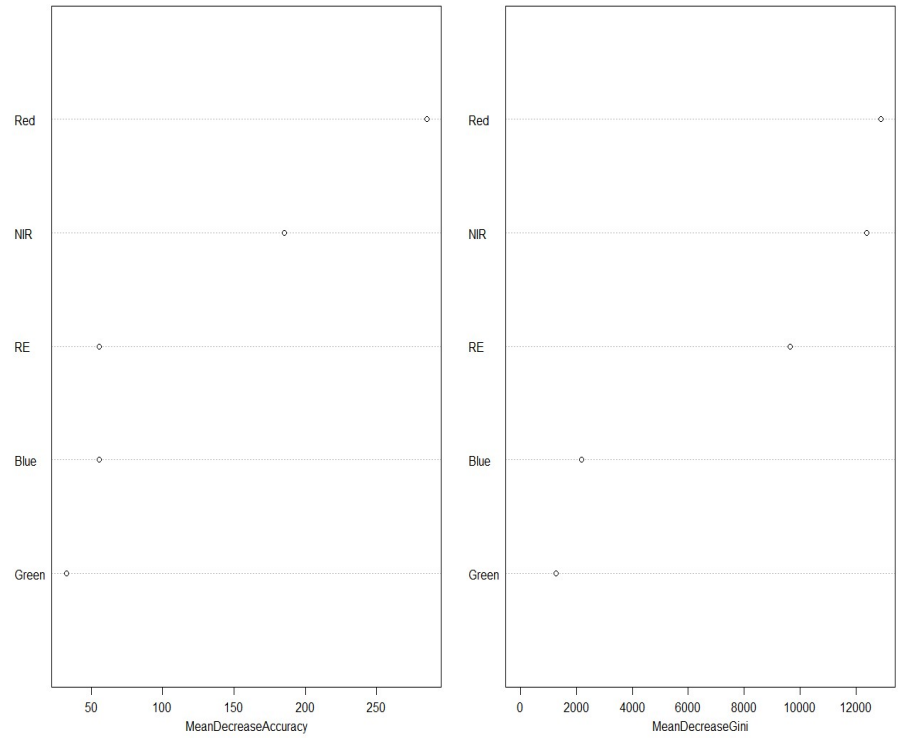


(a) RF mtry Tuning graph

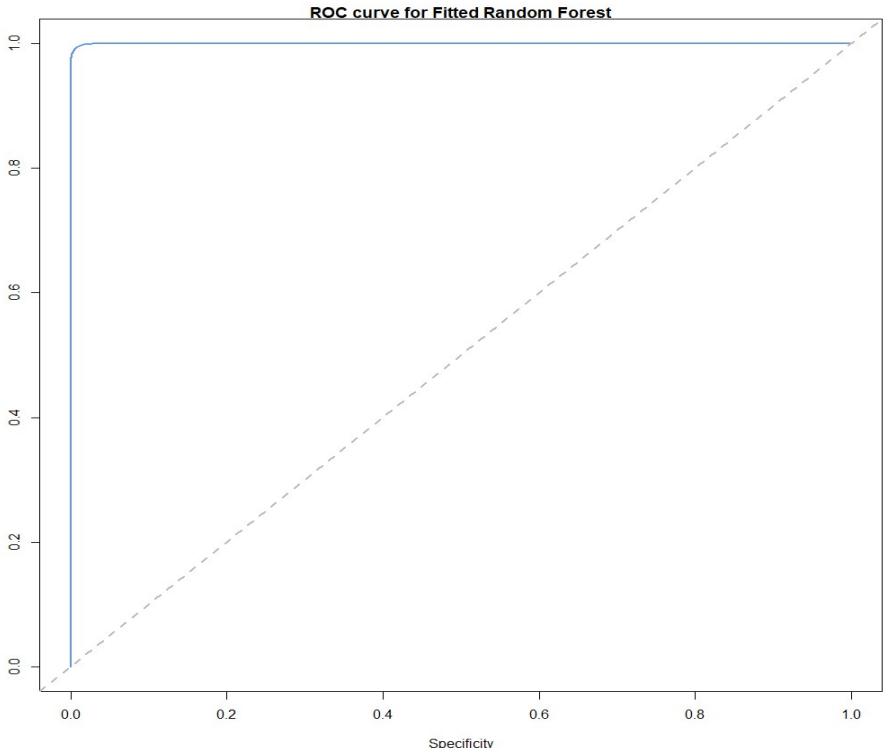


(b) RF predicted tree cover

Figure 7.9: *RF Tuning and Prediction based on 0.5 threshold*



(a) RF Variable Importance Plot



(b) Thresholded ROC at 0.5 level

Figure 7.10: *RF diagnostics*

7.2 Tables

Table 7.1: Sobel Raster Pixel Frequency

Pixel value	Count
0 values	579
NA values	30,543

Table 7.2: Laplace Raster Pixel Frequency

Pixel value	Count
-1 values	29
0 values	449
1 values	101
NA Values	30,543

7.3 R code

```
1
2
3  #Import libraries----------
4
5  library(raster)
6
7  library(rgdal)
8
9  library(rgeos)
10
11 library(landsat)
12
13 library(RStoolbox)
14
15 library(cluster)
16
17 library(NbClust)
18
19 library(factoextra)
20
21 library(tmap)
22
23 library(doParallel)
24
25 library(randomForest)
26
27 library(caret)
28
29 library(ROCR)
30
31 library(stargazer)
32
33 library(simpleboot)
34
35
36
37 #Set seed and multicore--------
38
39 set.seed(12345)
40
41
42
43 cl <- makeCluster(detectCores())
44
45 registerDoParallel(cl)
46
47
48
```

```
#Plot the different light bands from the Orthonomic photos ----------

#below code sourced from :

#https://www.earthdatascience.org/courses/earth-analytics/multispectral-remote-sensing-data/landsat-data-in-r-geotiff/


#Color band combinations for landsat 5:

#https://www.esri.com/arcgis-blog/products/product/imagery/band-combinations-for-landsat-8/?rmedium=redirect&rsource=/esri/arcgis/2013/07/24/band-combinations-for-landsat-8


spec_rgb <-

  stack(

    "C:/Users/Sanil Purryag/Google Drive/St Andrews docs/Modules/Semester 2/Thesis data/PortMoak_RTK_Ortho.tif"

  )


spec_br <- brick(spec_rgb)


#rename the different bands

names(spec_rgb) <- c('Blue', 'Green', 'Red', 'NIR', 'RE')


names(spec_br) <- c('Blue', 'Green', 'Red', 'NIR', 'RE')


#check if files have been properly renamed


names(spec_br)


#Import the DSM data
```

```
96
97 spec_lidar <-
98
99   raster(
100
101     "C:/Users/Sanil Purryag/Google Drive/St Andrews docs/Modules/Semester 2/Thesis
      data/PortMoak_RTK_DSM.tif"
102
103   )
104
105
106
107 #Plot each individual band
108
109
110
111 plot(
112
113   spec_br,
114
115   stretch = "hist",
116
117   maxv = 65535,
118
119   #col = gray(0:100 / 100),
120
121   box = FALSE,
122
123   axes = FALSE
124
125 )
126
127
128
129 #True color composite plot
130
131
132
133 par(col.axis = "white",
134
135     col.lab = "white",
136
137     tck = 0)
138
139 plotRGB(
140
141   spec_br,
142
143   r = 3,
144
145   g = 2,
```

```
146
147   #should be 2.5 times more
148
149   b = 1,
150
151   stretch = "hist",
152
153   axes = FALSE,
154
155   scale = 65535,
156
157   main = "RGB composite image/n Landsat Bands 3,2,1"
158
159 )
160
161
162
163
164
165 #False color composite plot
166
167 par(col.axis = "white",
168
169     col.lab = "white",
170
171     tck = 0)
172
173 plotRGB(
174
175   spec_br,
176
177   r = 5,
178
179   g = 4,
180
181   b = 3,
182
183   stretch = "hist",
184
185   axes = FALSE,
186
187   main = "RGB False composite image\n Landsat Bands 5,4,3"
188
189 )
190
191 box(col = "white")
192
193
194
195
196
```

```
#Exploratory analysis————————————


#Plot histogram of values from DSM


#assign and replace NAs with base 160
histlidar <- spec_lidar
NAvalue(histlidar) <- 160
#Plot distribution of values in hist
hist(
  histlidar,
  main = "Distribution of DSM Values(100,000 pixels sampled)",
  xlab = "DSM Elevation Value (Inches)",
  ylab = "Frequency",
  col = "cornflowerblue",
  ylim = c(0, 20000),
  xlim = c(150, 200)
)


hist(spec_br,
     xlab = "Pixel Intensity",
     ylab = "Frequency",
     col = "cornflowerblue")
# ylim = c(0, 20000),
# xlim = c(150, 200))


#Pairs plot of different bands
```

```
248
249 pairsplot <- pairs(spec_rgb)
250
251
252
253 #Hillshading (DSM Workstream)----------
254
255 #Below code adapted from:
256
257 #https://github.com/mgimond/Spatial/blob/master/A4-Raster-Operations.Rmd
258
259 #https://www.rdocumentation.org/packages/raster/versions/2.6-7/topics/focal
260
261
262
263 #Estimate the slope and aspect of the terrain
264
265 slope <- terrain(spec_lidar, opt = 'slope')
266
267 aspect <- terrain(spec_lidar, opt = 'aspect')
268
269
270
271 #Create hillshade (DTM) using the queen case (neighbours = 8)
272
273 #Current specification used for rough terrain estimation
274
275 hill <- hillShade(slope, aspect, 40, 270)
276
277
278
279 #plot the tmap package to see greyscale raster
280
281 hill <-
282
283   aggregate(hill, 25) # aggregate the raster by a factor of 25
284
285 tm_shape(hill) + tm_raster(palette = "Greys") +
286
287   tm_legend(outside = TRUE, text.size = .6) + tm_layout(title = "Hillshaded DSM")
288
289
290
291 #Histogram of hillside values - 100,000 sampled values
292
293 hist(
294
295   hill,
296
297   main = "Distribution of Aggregated hill values",
298
```

```
  xlab = "Hillshaded values",
  ylab = "Frequency",
  col = "cornflowerblue"
)
abline(v = 0.5, col = "Red", lty = 4)


#Zonal statistics - Sobel----


#============================================================================
    Sobel Kernel Application


#Assign hill to new object
hillside <- hill
#Threshold hillside
hillside[hillside > 0.5] <- NA
na.omit(hillside)
#plot code adapted from:
#https://stackoverflow.com/questions/28247812/r-raster-package-plot-producing-
    monochrome-images


plot(hill,
     main = "Hillshaded DSM raster",
     col = gray.colors(
       10,
       start = 0.3,
       end = 0.9,
       gamma = 2.2,
```

```
348
349         alpha = NULL
350
351       ))
352
353 plot(
354
355   hillside,
356
357   main = "Thresholded DSM hillshaded raster at 0.5 level",
358
359   col = gray.colors(
360
361     10,
362
363     start = 0.3,
364
365     end = 0.9,
366
367     gamma = 2.2,
368
369     alpha = NULL
370
371   )
372
373 )
374
375
376
377 #Sobel weight matrix definition
378
379 Sobel <- matrix(c(-1, 0, 1,-2, 0, 2,-1, 0, 1) / 4, nrow = 3)
380
381 f5    <- focal(hillside, w = Sobel, fun = sum)
382
383 # tm_shape(f5) + tm_raster(palette = "black") +
384
385 #   tm_legend(legend.show = TRUE, "Distribution of values")
386
387 freq(f5)
388
389 plot(f5,
390
391      col = gray.colors(
392
393        10,
394
395        start = 0.3,
396
397        end = 0.9,
398
```

```
399         gamma = 2.2,
400
401         alpha = NULL
402
403       ),
404
405       main = "Sobel Edge Detection Application")
406
407 #Flatten the map by reprojecting the raster to a Km by km dimension and to the
      British grid
408
409 Projected.sobel <-
410
411   projectRaster(f5, crs = "+init=epsg:27700 +units=km")
412
413 #project the raster to a resolution of 10m per cell
414
415 Reprojection.sobel <-
416
417   projectRaster(Projected.sobel,
418
419                 crs = crs(Projected.sobel),
420
421                 res = 0.010)
422
423 plot(Reprojection.sobel, main = "Reprojected Sobel Raster")
424
425 count.sobel <- freq(Reprojection.sobel)
426
427 print(count.sobel)
428
429
430
431 #================================================================================
      Sobel over different levels
432
433 # #Different levels of thresholding based on 0.01 increase between 0.1 and 1
434
435 # #100 numbers between 0 and 1
436
437 # seq.length <- seq(0.1, 1, 0.01)
438
439 # sobel.count <- list()
440
441 #
442
443 # for (i in seq.length) {
444
445 #   #Reset before thresholding
446
447 #   hillside <- hill
```

```
448
449 #   #threshold hillside by i
450
451 #   hillside[hillside > i] <- NA
452
453 #   na.omit(hillside)
454
455 #   #Compute sobel kernel and associated focal stats
456
457 #   Sobel <- matrix(c(-1, 0, 1,-2, 0, 2,-1, 0, 1) / 4, nrow = 3)
458
459 #   f6    <- focal(hillside, w = Sobel, fun = sum)
460
461 #   #Reproject the computed focal raster
462
463 #   Projected.sobel <-
464
465 #     projectRaster(f6, crs = "+init=epsg:27700 +units=km")
466
467 #   #project the raster to a resolution of 10m per cell
468
469 #   Reprojection.sobel <-
470
471 #     projectRaster(Projected.sobel,
472
473 #                   crs = crs(Projected.sobel),
474
475 #                   res = 0.010)
476
477 #   #Obtain the frequency of observed values
478
479 #   count.sobel <- as.matrix(freq(Reprojection.sobel))
480
481 #   sobel.count[[paste0("Threshold level ", i)]] <- count.sobel
482
483 #   #print(sobel.count)
484
485 # }
486
487 #
488
489 # tree.sobel <- c()
490
491 # for (m in 1:length(sobel.count)) {
492
493 #   tree.sobel <-
494
495 #     c(tree.sobel, (sobel.count[[m]][(as.numeric(which(sobel.count[[m]] == 0))),
    2]))
496
497 # }
```

```
498
499 #
500
501 # plot(
502
503 #   y = tree.sobel,
504
505 #   x = seq.length,
506
507 #   type = "o",
508
509 #   ylab = "Count of Trees",
510
511 #   xlab = "Thresholding Values",
512
513 #   main = "Sobel Tree count evolution based on Thresholding ",
514
515 #   col = "cornflowerblue"
516
517 # )
518
519 #=============================================================================
    Sobel Adjusted thresholds
520
521 #Different levels of thresholding based on 0.01 increase between 0 and 0.6
522
523 #100 numbers between 0 and 1
524
525 seq.length1 <- seq(0.1, 0.6, 0.001)
526
527 sobel.count5 <- list()
528
529
530
531 for (h in seq.length1) {
532
533   #Reset before thresholding
534
535   hillside <- hill
536
537   #threshold hillside by i
538
539   hillside[hillside > h] <- NA
540
541   na.omit(hillside)
542
543   #Compute sobel kernel and associated focal stats
544
545   Sobel <- matrix(c(-1, 0, 1,-2, 0, 2,-1, 0, 1) / 4, nrow = 3)
546
547   f7    <- focal(hillside, w = Sobel, fun = sum)
```

```
548
549   #Reproject the computed focal raster
550
551   Projected.sobel5 <-
552
553     projectRaster(f7, crs = "+init=epsg:27700 +units=km")
554
555   #project the raster to a resolution of 10m per cell
556
557   Reprojection.sobel5 <-
558
559     projectRaster(Projected.sobel5,
560
561                   crs = crs(Projected.sobel5),
562
563                   res = 0.010)
564
565   #Obtain the frequency of observed values
566
567   count.sobel5 <- as.matrix(freq(Reprojection.sobel5))
568
569   sobel.count5[[paste0("Threshold level ", h)]] <- count.sobel5
570
571   #print(sobel.count5)
572
573 }
574
575
576
577 tree.sobel1 <- c()
578
579 for (o in 1:length(sobel.count5)) {
580
581   tree.sobel1 <-
582
583     c(tree.sobel1, (sobel.count5[[o]][(as.numeric(which(sobel.count5[[o]] == 0))),
      2]))
584
585 }
586
587
588
589 plot(
590
591   y = tree.sobel1,
592
593   x = seq.length1,
594
595   type = "o",
596
597   ylab = "Count of Trees",
```

```
598
599   xlab = "Thresholding Values",
600
601   main = "Sobel Tree count evolution based on Thresholding ",
602
603   col = "cornflowerblue"
604
605 )
606
607
608
609
610
611
612
613 #=============================================================================
      Sobel Non parametric CI
614
615
616
617 #compute the SE of the simulated distribution
618
619 standard.error.sobel <- sd(tree.sobel1) / sqrt(length(tree.sobel1))
620
621 mean.error.sobel  <- mean(tree.sobel1)
622
623 #upper and lower normal limits
624
625 ul.sobel <- mean.error.sobel + 1.96 * (standard.error.sobel)
626
627 ll.sobel <- mean.error.sobel -  1.96 * (standard.error.sobel)
628
629 #add lines for bands to the plot y axis
630
631 abline(h = mean.error.sobel, col = "black", lty = 2)
632
633 abline(h = ul.sobel, col = "red", lty = 2)
634
635 abline(h = ll.sobel, col = "red", lty = 2)
636
637
638
639 #compute a nonparametric bootstrap from the different thresholded counts
640
641 #code adapted from:
642
643 #https://stats.stackexchange.com/questions/16516/how-can-i-calculate-the-confidence
      -interval-of-a-mean-in-a-non-normally-distribu
644
645
646
```

```r
#Take tree counts from different thresholds and laplace application

#2000 resamples, 5% trim



sobel.boot <- one.boot(tree.sobel1, mean, R = 2000, tr = 0.05)

hist(sobel.boot, main = "Sobel count bootstrap distribution", col = "cornflowerblue")

sobel.bootstrap <- boot.ci(sobel.boot, type = c("perc", "bca"))

print(sobel.bootstrap)




#Zonal statistics - Laplace----



#================================================================================
    Laplacian filter Application


#Assign hill to new object

hillside <- hill

#Threshold hillside

hillside[hillside > 0.5] <- NA

na.omit(hillside)


laplace <- matrix(c(0, 1, 0, 1,-4, 1, 0, 1, 0), nrow = 3)

f10    <- focal(hillside, w = laplace, fun = sum)

#tm_shape(f10) + tm_raster(palette = "black") +

#  tm_legend(legend.show = TRUE)

freq(f10)

plot(f10,
```

```
696
697     col = gray.colors(
698
699       10,
700
701       start = 0.3,
702
703       end = 0.9,
704
705       gamma = 2.2,
706
707       alpha = NULL
708
709     ),
710
711     main = "Laplace Edge Detection Application")
712
713
714
715 #Flatten the map by reprojecting the raster to a Km by km dimension and to the
      British grid
716
717 Projected.laplace <-
718
719   projectRaster(f10, crs = "+init=epsg:27700 +units=km")
720
721 #project the raster to a resolution of 10m per cell
722
723 Reprojection.laplace <-
724
725   projectRaster(Projected.laplace,
726
727                 crs = crs(Projected.laplace),
728
729                 res = 0.010)
730
731 count.laplace <- freq(Reprojection.laplace)
732
733 print(count.laplace)
734
735 plot(Reprojection.laplace, main = "Reprojected Laplace")
736
737 #============================================================================
      Laplacian over different levels
738
739
740
741 # #Different levels of thresholding based on 0.01 increase between 0.1 and 1
742
743 # #100 numbers between 0 and 1
744
```

```
# seq.length <- seq(0.1, 1, 0.01)

# laplace.count <- list()

#

# for (i in seq.length) {

#   #Reset before thresholding

#   hillside <- hill

#   #threshold hillside by i

#   hillside[hillside > i] <- NA

#   na.omit(hillside)

#   #Compute sobel kernel and associated focal stats

#   laplace <- matrix(c(0, 1, 0, 1,-4, 1, 0, 1, 0), nrow = 3)

#   f11    <- focal(hillside, w = laplace, fun = sum)

#   #Reproject the computed focal raster

#   Projected.laplace <-

#     projectRaster(f11, crs = "+init=epsg:27700 +units=km")

#   #project the raster to a resolution of 10m per cell

#   Reprojection.laplace <-

#     projectRaster(Projected.laplace,

#                   crs = crs(Projected.laplace),

#                   res = 0.010)

#   #Obtain the frequency of observed values

#   count.laplace <- as.matrix(freq(Reprojection.laplace))

#   laplace.count[[paste0("Threshold level ", i)]] <- count.laplace

#   #print(laplace.count)

# }

#
```

```
796
797 # tree.laplace <- c()
798
799 # for (m in 1:length(laplace.count)) {
800
801 #   tree.laplace <-
802
803 #     c(tree.laplace, (laplace.count[[m]][(as.numeric(which(laplace.count[[m]] ==
804
805 #                                                          0))), 2]))
806
807 # }
808
809 #
810
811 # plot(
812
813 #   y = tree.laplace,
814
815 #   x = seq.length,
816
817 #   type = "o",
818
819 #   ylab = "Count of Trees",
820
821 #   xlab = "Thresholding Values",
822
823 #   main = "Laplace Tree count evolution based on Thresholding ",
824
825 #   col = "cornflowerblue"
826
827 # )
828
829 #=============================================================================
    Laplacian over bounded threshold
830
831 #Different levels of thresholding based on 0.01 increase between 0.1 and 0.6
832
833 #100 numbers between 0 and 1
834
835 seq.length2 <- seq(0.1, 0.6, 0.001)
836
837 laplace.count5 <- list()
838
839
840
841 for (y in seq.length2) {
842
843   #Reset before thresholding
844
845   hillside <- hill
```

```
  #threshold hillside by i
  hillside[hillside > y] <- NA
  na.omit(hillside)
  #Compute sobel kernel and associated focal stats
  laplace <- matrix(c(0, 1, 0, 1,-4, 1, 0, 1, 0), nrow = 3)
  f16    <- focal(hillside, w = laplace, fun = sum)
  #Reproject the computed focal raster
  Projected.laplace5 <-
    projectRaster(f16, crs = "+init=epsg:27700 +units=km")
  #project the raster to a resolution of 10m per cell
  Reprojection.laplace5 <-
    projectRaster(Projected.laplace5,
                  crs = crs(Projected.laplace5),
                  res = 0.010)
  #Obtain the frequency of observed values
  count.laplace5 <- as.matrix(freq(Reprojection.laplace5))
  laplace.count5[[paste0("Threshold level ", y)]] <- count.laplace5
  #print(laplace.count)
}


tree.laplace1 <- c()
for (d in 1:length(laplace.count5)) {
  tree.laplace1 <-
    c(tree.laplace1, (laplace.count5[[d]][(as.numeric(which(laplace.count5[[d]] ==
                                                              0))), 2]))
```

```
}


plot(
  y = tree.laplace1,
  x = seq.length2,
  type = "o",
  ylab = "Count of Trees",
  xlab = "Thresholding Values",
  main = "Laplace Tree count evolution based on Thresholding ",
  col = "cornflowerblue"
)


#compute the SE of the simulated distribution
standard.error.laplace <-
  sd(tree.laplace1) / sqrt(length(tree.laplace1))
mean.error.laplace <- mean(tree.laplace1)
#upper and lower normal limits
ul.laplace <- mean.error.laplace + 1.96 * (standard.error.laplace)
ll.laplace <- mean.error.laplace - 1.96 * (standard.error.laplace)
#add lines for bands to the plot y axis
abline(h = mean.error.laplace, col = "black", lty = 1)
abline(h = ul.laplace, col = "red", lty = 2)
abline(h = ll.laplace, col = "red", lty = 2)



#compute a nonparametric bootstrap from the different thresholded counts
```

```
948
949 #code adapted from:
950
951 #https://stats.stackexchange.com/questions/16516/how-can-i-calculate-the-confidence
    -interval-of-a-mean-in-a-non-normally-distribu
952
953
954
955 #Take tree counts from different thresholds and laplace application
956
957 #2000 resamples, 5% trim
958
959
960
961 laplace.boot <- one.boot(tree.laplace1, mean, R = 2000, tr = 0.05)
962
963 hist(laplace.boot, main = "laplace count bootstrap distribution", col = "
    cornflowerblue")
964
965 laplace.bootstrap <- boot.ci(laplace.boot, type = c("perc", "bca"))
966
967 print(laplace.bootstrap)
968
969
970
971 #Measures of accuracy for zonal stats----------
972
973 #Code adapted from:
974
975 #https://gis.stackexchange.com/questions/119305/how-to-calculate-and-compare-rmse-
    between-two-dems
976
977 #https://gis.stackexchange.com/questions/4802/rmse-between-two-rasters-step-by-step
978
979
980
981 #Extract the values from the thresholded rasters at 0.5
982
983 hill.error <- getValues(hillside)
984
985 f5.error <- getValues(f5)
986
987 f10.error <- getValues(f10)
988
989
990
991 #Sobel at 0.5 threshold (RMSE)
992
993 #sqrt(mean(Actual - Predicted)^2)
994
995 diff.sobel <- (hill.error - f5.error) ^ 2
```

```
996
997 avg.sobel <- mean(na.omit(diff.sobel))
998
999 #RMSE.sobel <- sqrt(avg.sobel)
1000
1001 print(avg.sobel)
1002
1003
1004
1005 #Sobel at 0.5 threshold (PNSR)
1006
1007 PNSR.sobel <- 10 * log(((0.6910431 ^ 2) / avg.sobel))
1008
1009 print(PNSR.sobel)
1010
1011
1012
1013 #Laplacian at 0.5 threshold (RMSE)
1014
1015 diff.laplace <- (hill.error - f10.error) ^ 2
1016
1017 avg.laplace <- mean(na.omit(diff.laplace))
1018
1019 #RMSE.laplace <- sqrt(avg.laplace)
1020
1021 print(avg.laplace)
1022
1023
1024
1025 #Laplacian at 0.5 Threshold (PNSR)
1026
1027 PNSR.laplace <- 10 * log(((2.675293 ^ 2) / avg.laplace))
1028
1029 print(PNSR.laplace)
1030
1031
1032
1033 #Compute the different index and color composites----------
1034
1035
1036
1037 #Aggregate Orthophoto by a factor of 45
1038
1039 spec_br1 <- aggregate(spec_rgb, 45, progress = 'text')
1040
1041
1042
1043 names(spec_br1) <- c('Blue', 'Green', 'Red', 'NIR', 'RE')
1044
1045
1046
```

```
#Normalized Differentiation Vegetation Index (NDVI)

#Formula sourced from:

##https://www.earthdatascience.org/courses/earth-analytics/multispectral-remote-sensing-data/vegetation-indices-NDVI-in-R/



NIR <- spec_br1[[4]]

Red <- spec_br1[[3]]

Blue <- spec_br1[[1]]



#Compute the NDVI raster

Spec_ndvi <-

  overlay(

    NIR,

    Red,

    fun = function(x, y) {

      (x - y) / (x + y)

    }

  )



#Plot the NDVI

par(mar = c(1, 1, 1, 0.5))

plot(Spec_ndvi, main = "Normalized differentiation vegetation index", box = FALSE)



#Enhanced Vegetation index (EVI)

#Formula sourced from:

#https://cran.r-project.org/web/packages/LSRS/LSRS.pdf

```

```
1097
1098
1099 Spec_evi <- overlay(
1100
1101   NIR,
1102
1103   Red,
1104
1105   Blue,
1106
1107   fun = function(x, y, z) {
1108
1109     2.5 * ((x - y) / (x + (6 * y) + (7.5 * z) + 1))
1110
1111   }
1112
1113 )
1114
1115
1116
1117 #Unsupervised machine learning on stacked orthophoto (Kmeans)----------------
1118
1119 #Formula:
1120
1121 #https://medium.com/regen-network/remote-sensing-indices-389153e3d947
1122
1123
1124
1125 #Below code adapted from:
1126
1127 #http://rspatial.org/analysis/rst/9-remotesensing.html
1128
1129 #https://geoscripting-wur.github.io/AdvancedRasterAnalysis/
1130
1131
1132
1133
1134
1135 #Create determinant rasters of the cluster
1136
1137 #benchmark <- stack(Spec_evi, Spec_ndvi, spec_br1[[5]])
1138
1139 benchmark <- stack(Spec_evi, Spec_ndvi)
1140
1141
1142
1143 #============================================================================Plot
       all clustering possibilities
1144
1145 kposs <- c()
1146
```

```
kpossibilities <- function(num = 10) {

  for (k in 1:num) {

    set.seed(12345)

    kmncluster <-

      kmeans(

        na.omit(benchmark[]),

        centers = k,

        iter.max = 500,

        nstart = 5,

        algorithm = "Lloyd"

      )

    kposs <- c(kposs, kmncluster$tot.withinss)

    kclass <- spec_br1[[1]]

    kclass[] <- kmncluster$cluster

    plot(kclass, main = paste(k, "cluster(s)"))

  }

  return(kposs)

}


totalwss <- kpossibilities()

plot(

  y = totalwss,

  x = seq(1, 10, 1),

  main = "Elbow Method Results",

  type = "o",

  ylab = "Total Within Sum of Squares",
```

```
  xlab  = "Number of clusters",
  col = "cornflowerblue"
)
abline(v = 3,
       lty = 2,
       col = "red",
       lwd = 3)
abline(h = 1840.6808,
       lty = 2,
       col = "red",
       lwd = 3)



# nbclustering.avg <-
#   NbClust(
#     data = benchmark[],
#     diss = NULL,
#     distance = "euclidean",
#     min.nc = 2,
#     max.nc = 15,
#     method = "average",
#     index = "cindex",
#     alphaBeale = 0.1
#   )
```

```
#============================================================================
    Plot desired and create the training data


#Attempt to perform 3 cluster classification based on reduced NDVI image


#Kmeans clustering result
set.seed(12345)
kmncluster <-
  kmeans(
    na.omit(benchmark[]),
    centers = 3,
    iter.max = 500,
    nstart = 5,
    algorithm = "Lloyd"
  )


#Create training polygons and assign to arbitrary raster for visualisation
kclass <- spec_br1[[5]]
kclass[] <- kmncluster$cluster
#plot clustering result
plot(kclass)
freq(kclass)


#Edit the clustered raster details
classes <- kclass
names(classes) <- "Classes"
```

```
#plot Thematic raster

par(mar = c(1, 1, 1, 0.5))

plot(classes, main = "Thematic Raster")



#Create training data by isolating unwanted regions/classes through thresholding

#1 and 2 appears to be redundant here

classification <- classes

#plot(classification)

#Clusters less than 6 are omitted

#classification[classification < 6] <- NA

classification[classification > 2] <- NA



plot(classification)



#isolate the predictive bands; bands 1-4

#preds <- stack(spec_br1[[1]],spec_br1[[2]],spec_br1[[3]],spec_br1[[4]])

preds <- spec_br1

#mask predictors with defined classification to obtain training data

covmasked <- mask(preds, classification)

plot(covmasked)

trainlayer <- addLayer(covmasked, classification)

plot(trainlayer)



#Extract the values from training stack

values <- getValues(trainlayer)

valuetable <- na.omit(values)
```

```
1350
1351 valuetable <- as.data.frame(valuetable)
1352
1353 #colnames(valuetable)[5] <- "Classes"
1354
1355 valuetable$Classes <- as.factor(valuetable$Classes)
1356
1357 print(unique(valuetable$Classes))
1358
1359
1360
1361 #All band possibilities#####
1362
1363
1364
1365 allband <- stack(spec_br1, Spec_ndvi, Spec_evi)
1366
1367 names(allband) <-
1368
1369   c('Blue', 'Green', 'Red', 'NIR', 'RE', 'NDVI', 'EVI')
1370
1371
1372
1373
1374
1375 kpossibilities1 <- function(num = 10) {
1376
1377   for (k in 1:num) {
1378
1379     set.seed(12345)
1380
1381     kmncluster <-
1382
1383       kmeans(
1384
1385         na.omit(allband[]),
1386
1387         centers = k,
1388
1389         iter.max = 500,
1390
1391         nstart = 5,
1392
1393         algorithm = "Lloyd"
1394
1395       )
1396
1397
1398
1399     kclass <- spec_br1[[1]]
1400
```

```
    kclass[] <- kmncluster$cluster

    plot(kclass, main = paste(k, "cluster(s)"))

  }

}


kpossibilities1()




#Graph the valuetable based on the associated classes####

#Plot the different bands across the classes####

band.clust <- function(cluster) {

  par(mfrow = c(2, 3))

  val_class <- subset(valuetable, Classes == cluster)

  for (p in 1:5) {

    hist(

      val_class[[p]],

      main = c(paste(

        "Cluster",

        paste(cluster),

        colnames(val_class)[p],

        " Band distrbution"

      )),

      col = "cornflowerblue",

      xlab = c(paste(colnames(val_class)[p]), " values")

    )

  }
```

```
}



for (v in 6:7) {
  band.clust(v)
}


dev.off()


#Compute a tree count using kmeans----------


classification <- classes


Projected.kmeans <-
  projectRaster(classification, crs = "+init=epsg:27700 +units=km", res = 0.010)


plot(Projected.kmeans, main = "kmeans projected raster")


freq(Projected.kmeans)


#Supervised machine learning on stacked orthophoto (RandomForest)----------
#below code adapted from:
##https://geoscripting-wur.github.io/AdvancedRasterAnalysis/


tune <- tuneRF(x = valuetable[, c(1:5)],
```

```
                y = valuetable$Classes)

#plot OOB error

plot(

  tune,

  type = "o",

  main = "RF Tuning result",

  ylab = "OOB Error percentage",

  xlab = "Number of variables randomly sampled at each split",

  col = "cornflowerblue"

)



set.seed(12345)

#Fit the Random forest model

modelRF.final <-

  randomForest(

    x = valuetable[, c(1:5)],

    y = valuetable$Classes,

    mtry = 4,

    importance = TRUE

  )



#variable importance plot

varimp <- varImpPlot(modelRF.final)



#Predict tree cover on predictors (insample prediction)

predTree <- predict(preds, model = modelRF.final)
```

```
1554
1555
1556
1557 #plot predicted raster
1558
1559 par(mar = c(1, 1, 1, 0.5))
1560
1561 plot(predTree, main = "Random Forest Predicted Raster")
1562
1563
1564
1565 #Plot classes is one for tree and zero for notree
1566
1567 #Reproject ther raster
1568
1569 Projected.pred <-
1570
1571   projectRaster(predTree, crs = "+init=epsg:27700 +units=km")
1572
1573
1574
1575 #project the raster to a resolution of 10m per cell
1576
1577 Reprojection.pred <-
1578
1579   projectRaster(Projected.pred,
1580
1581                 crs = crs(Projected.pred),
1582
1583                 res = 0.010)
1584
1585 plot(Reprojection.pred)
1586
1587 #Threshold the reprojection for the count of trees (Based on color from legend)
1588
1589 Thresholded <- Reprojection.pred
1590
1591 RFcount <- freq(Thresholded)
1592
1593 print(RFcount)
1594
1595
1596
1597 #Edit confusion matrix for better interpretation - from computed RF model
1598
1599 colnames(modelRF.final$confusion) <-
1600
1601   c("Trees", "Non-Trees", "class Error")
1602
1603 rownames(modelRF.final$confusion) <- c("Trees", "Non-Trees")
1604
```

```
modelRF.final$confusion


#Plot sample tree from random forest

#(getTree(modelRF.final,4, labelVar = TRUE))



#below code adapted from:

#https://stackoverflow.com/questions/30366143/how-to-compute-roc-and-auc-under-roc-after-training-using-caret-in-r


#Compute the accuracy metrics

predictions <-

  as.vector(modelRF.final$votes[, 2]) # Random forest output

predictions.auc <- prediction(predictions, valuetable$Classes)

#AUC

auc.performance <- performance(predictions.auc, "auc")

#ROC

roc.performance <- performance(predictions.auc, "tpr", "fpr")

#plot the ROC

#below code adapted from:

#https://chandramanitiwary.wordpress.com/2014/03/17/r-tips-part2-rocr-example-with-randomforest/


plot(roc.performance,

     main = "ROC curve for Fitted Random Forest",

     col = "cornflowerblue",

     lwd = 2,
```

```
1654
1655      xlab = "Specificity",
1656
1657      ylab = "Sensitivity")
1658
1659 abline(
1660
1661   a = 0,
1662
1663   b = 1,
1664
1665   lwd = 2,
1666
1667   lty = 2,
1668
1669   col = "gray"
1670
1671 )
1672
1673
1674
1675 #AUC detailed stats
1676
1677 auc <- unlist(slot(auc.performance, "y.values"))
1678
1679 #Compute min and max values
1680
1681 minauc <- min(round(auc, digits = 2))
1682
1683 maxauc <- max(round(auc, digits = 2))
1684
1685 minauct <- paste(c("min(AUC) = "), minauc, sep = "")
1686
1687 maxauct <- paste(c("max(AUC) = "), maxauc, sep = "")
1688
1689
1690
1691
1692
1693 #Threshold definition adapted from:
1694
1695 #https://stackoverflow.com/questions/16347507/obtaining-threshold-values-from-a-roc
      -curve
1696
1697 cutoffs <- data.frame(cut=roc.performance @alpha.values[[1]], fpr=roc.performance
      @x.values[[1]],
1698
1699                       tpr=roc.performance @y.values[[1]])
1700
1701 head(cutoffs)
1702
```

```
#order cutoffs

cutoffs <- cutoffs[order(cutoffs$tpr, decreasing=TRUE),]


#########################################################################10% threshold


set.seed(12345)
#Fit the Random forest model
modelRF.final <-
  randomForest(
    x = valuetable[, c(1:5)],
    y = valuetable$Classes,
    mtry = 4,
    importance = TRUE,
    cutoff = c(0.9,0.1)
  )


#variable importance plot
varimp <- varImpPlot(modelRF.final)


#Predict tree cover on predictors (insample prediction)
predTree <- predict(preds, model = modelRF.final)


#plot predicted raster
par(mar = c(1, 1, 1, 0.5))
plot(predTree, main = "Random Forest Predicted Raster")

```

```
#Plot classes is one for tree and zero for notree

#Reproject ther raster

Projected.pred <-

  projectRaster(predTree, crs = "+init=epsg:27700 +units=km")


#project the raster to a resolution of 10m per cell

Reprojection.pred <-

  projectRaster(Projected.pred,

                crs = crs(Projected.pred),

                res = 0.010)

plot(Reprojection.pred)

#Threshold the reprojection for the count of trees (Based on color from legend)

Thresholded <- Reprojection.pred

RFcount <- freq(Thresholded)

print(RFcount)


#Edit confusion matrix for better interpretation - from computed RF model

colnames(modelRF.final$confusion) <-

  c("Trees", "Non-Trees", "class Error")

rownames(modelRF.final$confusion) <- c("Trees", "Non-Trees")

modelRF.final$confusion


#Plot sample tree from random forest

#(getTree(modelRF.final,4, labelVar = TRUE))
```

```
1805
1806
1807 #below code adapted from:
1808
1809 #https://stackoverflow.com/questions/30366143/how-to-compute-roc-and-auc-under-roc-
     after-training-using-caret-in-r
1810
1811
1812
1813 #Compute the accuracy metrics
1814
1815 predictions <-
1816
1817   as.vector(modelRF.final$votes[, 2]) # Random forest output
1818
1819 predictions.auc <- prediction(predictions, valuetable$Classes)
1820
1821 #AUC
1822
1823 auc.performance <- performance(predictions.auc, "auc")
1824
1825 #ROC
1826
1827 roc.performance <- performance(predictions.auc, "tpr", "fpr")
1828
1829 #plot the ROC
1830
1831 #below code adapted from:
1832
1833 #https://chandramanitiwary.wordpress.com/2014/03/17/r-tips-part2-rocr-example-with-
     randomforest/
1834
1835
1836
1837 plot(roc.performance,
1838
1839      main = "ROC curve for Fitted Random Forest",
1840
1841      col = "cornflowerblue",
1842
1843      lwd = 2,
1844
1845      xlab = "Specificity",
1846
1847      ylab = "Sensitivity")
1848
1849 abline(
1850
1851   a = 0,
1852
1853   b = 1,
```

```
  lwd = 2,
  lty = 2,
  col = "gray"
)


#AUC detailed stats
auc <- unlist(slot(auc.performance, "y.values"))
#Compute min and max values
minauc <- min(round(auc, digits = 2))
maxauc <- max(round(auc, digits = 2))
minauct <- paste(c("min(AUC) = "), minauc, sep = "")
maxauct <- paste(c("max(AUC) = "), maxauc, sep = "")



#Threshold definition adapted from:
#https://stackoverflow.com/questions/16347507/obtaining-threshold-values-from-a-roc-curve
cutoffs <- data.frame(cut=roc.performance @alpha.values[[1]], fpr=roc.performance @x.values[[1]],
                      tpr=roc.performance @y.values[[1]])
head(cutoffs)
#order cutoffs
cutoffs <- cutoffs[order(cutoffs$tpr, decreasing=TRUE),]
```

Bibliography

Al-Doski, J., Mansor1, S. B., Zulhaidi, H. & Shafri, M. (2013), 'Image Classification in Remote Sensing', **3**(10).
URL: *www.iiste.org*

Barati, S., Rayegani, B., Saati, M., Sharifi, A. & Nasri, M. (2011), 'Comparison the accuracies of different spectral indices for estimation of vegetation cover fraction in sparse vegetated areas', *Egyptian Journal of Remote Sensing and Space Science* **14**(1), 49–56.

Bivand, R., Keitt, T. & Rowlingson, B. (2018), *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*. R package version 1.3-3.
URL: *https://CRAN.R-project.org/package=rgdal*

Bivand, R. & Rundel, C. (2018), *rgeos: Interface to Geometry Engine - Open Source ('GEOS')*. R package version 0.3-28.
URL: *https://CRAN.R-project.org/package=rgeos*

Bivand, R. S., Pebesma, E. J. & Gmez-Rubio, V. (2008), *Applied spatial data analysis with R*, Springer, New York; London.

Blackbridge (2014), The RapidEye Red Edge Band.
URL: *https://resa.blackbridge.com/files/2014-06/Red_Edge_White_Paper.pdf*

Bradski, G. (2000), 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools* .

Bradski, G. (2018), 'Sobel Derivatives OpenCV 2.4.13.7 documentation'.
URL: *https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html*

Breiman, L. (1999), 'Random forest', *Machine Learning* **45**(5), 1–35.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. (1984), *Classification and regression trees Regression trees*, number June.

Chandana, B. S., Srinivas, K., Kiran Kumar, R., Saichandana, B. & Professor, A. (2014), 'Clustering Algorithm Combined with Hill Climbing for Classification of Remote Sensing Image', *International Journal of Electrical and Computer Engineering (IJECE)* **4**(6), 923–930.
URL: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.901.6899&rep=rep1&type=pdf*

Charrad, M., Ghazzali, N., Boiteau, V. & Niknafs, A. (2014), 'NbClust: An R package for determining the relevant number of clusters in a data set', *Journal of Statistical Software* **61**(6), 1–36.
URL: *http://www.jstatsoft.org/v61/i06/*

Corporation, M. & Weston, S. (2017), *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.11.
URL: *https://CRAN.R-project.org/package=doParallel*

De Freitas, G. M., De Ávila, A. M. H. & Papa, J. P. (2009), Satellite-based rainfall estimation through semi-supervised learning, *in* '2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009', Vol. 6, pp. 1–5.

Dhanachandra, N., Manglem, K. & Chanu, Y. J. (2015), 'Image Segmentation Using K - means Clustering Algorithm and Subtractive Clustering Algorithm', *Procedia Computer Science* **54**, 764–771.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S1877050915014143*

El Garouani, A., Alobeid, A. & El Garouani, S. (2014), 'Digital surface model based on aerial image stereo pairs for 3D building', *International Journal of Sustainable Built Environment* **3**(1), 119–126.

Elias, C. J. (2015), 'Practitioner's Corner Percentile and Percentile-t Bootstrap Confidence Intervals: A Practical Comparison', *J. Econom. Meth* **4**(1), 153–161.

Environmental Systems Research Institute (ESRI) (2018), 'What is GIS?, Geographic Information System Mapping Technology'.
URL: *https://www.esri.com/en-us/what-is-gis/overview*

Fei-Fei, L., Fergus, R. & Perona, P. (2006), 'One-shot learning of object categories', *IEEE transactions on pattern analysis and machine intelligence* **28**(4), 594–611.

Fernández-Lozano, J., Gutiérrez-Alonso, G. & Fernández-Morán, M. Á. (2015), 'Using airborne LiDAR sensing technology and aerial orthoimages to unravel roman water supply systems and gold works in NW Spain (Eria valley, León)', *Journal of Archaeological Science* **53**, 356–373.

Fisher, R., Perkins, S., A, W. & Wolfart, E. (2003), 'Feature Detectors - Sobel Edge Detector'.
URL: *http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm*

Forestry Commission England (2018), 'Tree Establishment'.
URL: *https://www.forestry.gov.uk/forestry/infd-6arg5y*

from Jed Wing, M. K. C., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C. & Hunt., T. (2018), *caret: Classification and Regression Training*. R package version 6.0-79.
URL: *https://CRAN.R-project.org/package=caret*

Ghassemian, H. & Landgrebe, D. (1988), 'Object-oriented feature extraction method for image data compaction', *IEEE Control Systems Magazine* **8**(3), 42–48.
URL: *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=476*

Gill, S. J., Biging, G. S. & Murphy, E. C. (2000), 'Modeling conifer tree crown radius and estimating canopy cover', *Forest Ecology and Management* **126**(3), 405–416.

Girod, B. (2013), Digital Image Processing : Edge Detection, Technical report.
URL: *https://web.stanford.edu/class/ee368/Handouts/Lectures/2014_Spring/Combined_Slides/11-Edge-Detection-Combined.pdf*

GISGeography (2018), 'World Geodetic System (WGS84) - GIS Geography'.
URL: *https://gisgeography.com/wgs84-world-geodetic-system/*

Gislason, P. O., Benediktsson, J. A. & Sveinsson, J. R. (2006), Random forests for land cover classification, *in* 'Pattern Recognition Letters', Vol. 27, pp. 294–300.

Goslee, S. C. (2011), 'Analyzing remote sensing data in R: The landsat package', *Journal of Statistical Software* **43**(4), 1–25.
URL: *http://www.jstatsoft.org/v43/i04/*

Gupta, S. & Ghosh Mazumdar, S. (2012), 'Sobel Edge Detection Algorithm', *International Journal of Computer Science and Management Research* **2**(2), 1578–1583.
URL: *www.ijcsmr.org*

Haralick, R. M. (1984), 'Digital step edges from zero crossing of second directional derivatives', *IEEE Trans. Pattern Anal. Mach. Intell.* **6**(1), 58–68.
URL: *http://www.ncbi.nlm.nih.gov/pubmed/21869165*

Hastie, T., Tibshirani, R. & Friedman, J. (2009), 'The Elements of Statistical Learning', *Elements* **1**, 337–387.
URL: *http://www.springerlink.com/index/10.1007/b94608*

Hijmans, R. J. (2017), *raster: Geographic Data Analysis and Modeling*. R package version 2.6-7.
URL: *https://CRAN.R-project.org/package=raster*

Hlavac, M. (2018), *stargazer: Well-Formatted Regression and Summary Statistics Tables*, Central European Labour Studies Institute (CELSI), Bratislava, Slovakia. R package version 5.2.1.
URL: *https://CRAN.R-project.org/package=stargazer*

Horn, B. K. (1981), 'Hill Shading and the Reflectance Map', *Proceedings of the IEEE* **69**(1), 14–47.

IBM Corporation (2013), 'IBM Knowledge Center - Tree Growing Process (CART algorithms)'.
URL: *https://www.ibm.com/support/knowledgecenter/en/SSLVMB_22.0.0/com.ibm.spss.statistics.algorithms/alg_tre*
cart_growth.htm

Janitza, S. & Hornung, R. (2018), 'On the overestimation of random forest's out-of-bag error', *PLOS ONE* **13**(8), e0201904.
URL: *http://dx.plos.org/10.1371/journal.pone.0201904*

Jaworski, M., Duda, P. & Rutkowski, L. (2018), 'New Splitting Criteria for Decision Trees in Stationary Data Streams', *IEEE Transactions on Neural Networks and Learning Systems* **29**(6), 2516–2529.

Kanevski, M., Pozdnoukhov, A. & Timonin, V. (2008), 'Machine Learning Algorithms for GeoSpatial Data. Applications and Software Tools', *iEMSs* pp. 320–327.
URL: *http://www.iemss.org/iemss2008/uploads/Main/S04-09-Kanevski_et_al-IEMSS2008.pdf*

Karimulla, S. & Raja, A. R. (2016), 'Tree Crown Delineation from High Resolution Satellite Images', *Indian Journal of Science and Technology ISSN* (9S1), 974–6846.
URL: *http://www.indjst.org/index.php/indjst/article/viewFile/107913/77633*

Kassambara, A. & Mundt, F. (2017), *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.5.
URL: *https://CRAN.R-project.org/package=factoextra*

Kaur, H. & Kaur, L. (2014), 'Performance Comparison of Different Feature Detection Methods with Gabor Filter', *International Journal of Science and Research* **3**(5), 1879–1886.
URL: *www.ijsr.net*

Kaur, J., Agrawal, S. & Vig, R. (2012), A Comparative Analysis of Thresholding and Edge Detection Segmentation Techniques, Technical Report 15.
URL: *https://pdfs.semanticscholar.org/d470/1e5e3dad68bfa231a0ba1e565a065993ced9.pdf*

Kaur, S. & Singh, I. (2016), Comparison between Edge Detection Techniques, Technical Report 15.
URL: *https://pdfs.semanticscholar.org/d5c1/d380263318c1b7a7d298500b3617e55ef2fa.pdf*

Khosravipour, A., Skidmore, A. K., Isenburg, M., Wang, T. & Hussin, Y. A. (2014), 'Generating pit-free canopy height models from airborne lidar', *Photogrammetric Engineering & Remote Sensing* **80**(9), 863–872.

Kursa, M. B. (2012), 'rferns: An implementation of the random ferns method for general-purpose machine learning', *arXiv preprint arXiv:1202.1121* .

Lei, S. & Smith, M. R. (2003), 'Evaluation of Several Nonparametric Bootstrap Methods to Estimate Confidence Intervals for Software Metrics', *IEEE Transactions on Software Engineering* **29**(11), 996–1004.

Leistner, C., Saffari, A., Santner, J. & Bischof, H. (2009), Semi-supervised random forests, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 506–513.

Leutner, B., Horning, N. & Schwalb-Willmann, J. (2018), *RStoolbox: Tools for Remote Sensing Data Analysis*. R package version 0.2.3.
URL: *https://CRAN.R-project.org/package=RStoolbox*

Liaw, A. & Wiener, M. (2002*a*), 'Classification and Regression by Random Forest', *R news* **2**, 18–22.

Liaw, A. & Wiener, M. (2002*b*), 'Classification and regression by randomforest', *R News* **2**(3), 18–22.
URL: *https://CRAN.R-project.org/doc/Rnews/*

Liaw, A. & Wiener, M. (2002*c*), 'Classification and regression by randomforest', *R News* **2**(3), 18–22.
URL: *https://CRAN.R-project.org/doc/Rnews/*

Liu, H. Q. & Huete, A. (1995), 'Feedback based modification of the NDVI to minimize canopy background and atmospheric noise', *IEEE Transactions on Geoscience and Remote Sensing* **33**(2), 457–465.

Liu, Q., Jing, L., Li, Y., Tang, Y., Li, H. & Lin, Q. (2016), A tree canopy height delineation method based on Morphological Reconstruction - Open Crown Decomposition, *in* 'IOP Conference Series: Earth and Environmental Science', Vol. 34.

Lloyd, S. P. (1982), 'Least Squares Quantization in PCM', *IEEE Transactions on Information Theory* **28**(2), 129–137.

Macqueen, J. (1967), 'Some methods for classification and analysis of multivariate observations', *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* **1**(233), 281–297.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M. & Hornik, K. (2018), *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.7-1 — For new features, see the 'Changelog' file (in the package source).

Matese, A., Di Gennaro, S. F. & Berton, A. (2017), 'Assessment of a canopy height model (CHM) in a vineyard using UAV-based multispectral imaging', *International Journal of Remote Sensing* **38**(8-10), 2150–2160.

Matsushita, B., Yang, W., Chen, J., Onda, Y. & Qiu, G. (2007), 'Sensitivity of the Enhanced Vegetation Index (EVI) and Normalized Difference Vegetation Index (NDVI) to topographic effects: A case study in high-density cypress forest', *Sensors* **7**(11), 2636–2651.

Maxwell, A. E., Warner, T. A. & Fang, F. (2018), 'Implementation of machine-learning classification in remote sensing: An applied review'.

McCool, M., Robison, A. D., Reinders, J., McCool, M., Robison, A. D. & Reinders, J. (2012), 'K-Means Clustering', *Structured Parallel Programming* pp. 279–289.
URL: *https://www.sciencedirect.com/science/article/pii/B9780124159938000116?via%3Dihub*

Miao, Q., Liu, R., Zhao, P., Li, Y. & Sun, E. (2017), 'A Semi-Supervised Image Classification Model Based on Improved Ensemble Projection Algorithm', *IEEE Access* **6**(Cccv).

Mills, R. T., Hoffman, F. M., Kumar, J. & Hargrove, W. W. (2011), Cluster analysis-based approaches for geospatiotemporal data mining of massive data sets for identification of forest threats, *in* 'Procedia Computer Science', Vol. 4, pp. 1612–1621.

National Center for Ecological Analysis and Synthesis (2018), Overview of Coordinate Reference Systems (CRS) in R, Technical report.
URL: *http://spatialreference.org/*

National Geographic Society (2012), 'Gis (geographic information system)'.
URL: *https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/*

O'Connor, J., Smith, M. J. & James, M. R. (2017), 'Cameras and settings for aerial surveys in the geosciences: Optimising image data', *Progress in Physical Geography* **41**(3), 325–344.

Oyelade, O. J., Oladipupo, O. O. & Obagbuwa, I. C. (2010), 'Application of k Means Clustering algorithm for prediction of Students Academic Performance', *International Journal of Computer Science and Information Security* **7**(1), 292–295.
URL: *http://arxiv.org/abs/1002.2425*

Paine, S. H. & Lodwick, G. D. (1989), 'Edge detection and processing of remotely sensed digital images', *Photogrammetria* **43**(6), 323–336.

Pal, M. (2005), 'Random forest classifier for remote sensing classification', *International Journal of Remote Sensing* **26**(1), 217–222.

Pal, N. R. & Pal, S. K. (1993), 'A review on image segmentation techniques', *Pattern Recognition* **26**(9), 1277–1294.

Peng, R. D. (2008), *simpleboot: Simple Bootstrap Routines*. R package version 1.1-3.

Rabiu, L. & Waziri, D. A. (2014), 'Digital Orthophoto Generation with Aerial Photographs', *Academic Journal of Interdisciplinary Studies* **3**(7), 133–141.
URL: *http://mcser.org/journal/index.php/ajis/article/view/5336*

Rekik, A., Zribi, M., Benjelloun, M. & Hamida, A. B. (2006), A k-means clustering algorithm initialization for unsupervised statistical satellite image segmentation, *in* '2006 1st IEEE International Conference on E-Learning in Industrial Electronics, ICELIE', pp. 11–16.

Ritter, P. (1987), 'A Vector-Based Slope and Aspect Generation Algorithm', *Photogrammetric Engineering & Remote Sensing* **53**(8), 1109–1111.

Robinson, A. C., Demšar, U., Moore, A. B., Buckley, A., Jiang, B., Field, K., Kraak, M.-J., Camboim, S. P. & Sluter, C. R. (2017), 'Geospatial big data and cartography: research challenges and opportunities for making maps that matter', *International Journal of Cartography* pp. 1–29.
URL: *https://www.tandfonline.com/doi/full/10.1080/23729333.2016.1278151*

Roettger, S. (2007), NDVI-based vegetation rendering, *in* '9th IASTED International Conference on Computer Graphics and Imaging, CGIM 2007', pp. 41–45.
URL: *http://www.scopus.com/inward/record.url?eid=2-s2.0-54949126385&partnerID=40&md5=bd829b13d933587*

RStudio Team (2016), *RStudio: Integrated Development Environment for R*, RStudio, Inc., Boston, MA.
URL: *http://www.rstudio.com/*

Sadykova, D. & James, A. P. (2017), Quality assessment metrics for edge detection and edge-aware filtering: A tutorial review, *in* '2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017', Vol. 2017-January, pp. 2366–2369.

Sandhu, P. S. & Mamta, J. (2009), 'Performance Evaluation of Edge Detection Techniques for Images in Spatial Domain', *International Journal of Computer Theory and Engineering* **1**(5), 1793–8201.
URL: *https://pdfs.semanticscholar.org/1350/cb01ea6f0c358178775b65144178b2109ba7.pdf*

Schweik, C. M. (2011), 'Introduction to Geographic Information Systems for Natural Resources Management'.
URL: *https://scholarworks.umass.edu/eco_ed_materials/1*

Shataee, S., Kalbi, S., Fallah, A. & Pelz, D. (2012), 'Forest attribute imputation using machine-learning methods and ASTER data: Comparison of k-NN, SVR and random forest regression algorithms', *International Journal of Remote Sensing* **33**(19), 6254–6280.

Shrivakshan, G. T. & Chandrasekar, C. (2012), 'A Comparison of various Edge Detection Techniques used in Image Processing', *International Journal of Computer Science Issues* **9**(5).
URL: *www.IJCSI.org*

Sing, T., Sander, O., Beerenwinkel, N. & Lengauer, T. (2005), 'Rocr: visualizing classifier performance in r', *Bioinformatics* **21**(20), 7881.
URL: *http://rocr.bioinf.mpi-sb.mpg.de*

Singh, I. (2016), Performance Evaluation of Watershed Segmentation Algorithm for Noisy and Noise Free Images Using Adaptive Thresholding and Masking, Technical Report 3.
URL: *http://www.iraj.in/journal/journal_file/journal_pdf/12-217-145407551085-89.pdf*

Singh, O. I. & James, O. (2012), 'Local Contrast and Mean based Thresholding Technique in Image Binarization', *International Journal of Computer Applications* **51**(6), 5–10.

Sirait, K., Tulus & Nababan, E. B. (2017), 'K-Means Algorithm Performance Analysis With Determining The Value Of Starting Centroid With Random And KD-Tree Method', *Journal of Physics: Conference Series* **930**, 012016.
URL: *http://stacks.iop.org/1742-6596/930/i=1/a=012016?key=crossref.8bafb33dff3caaa04a16efadf7f516d0*

Stephens, D. & Diesing, M. (2014), 'A comparison of supervised classification methods for the prediction of substrate type using multibeam acoustic and legacy grain-size data', *PLoS ONE* **9**(4).

Svetnik, V., Liaw, A., Tong, C., Christopher Culberson, J., Sheridan, R. P. & Feuston, B. P. (2003), 'Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling', *Journal of Chemical Information and Computer Sciences* **43**(6), 1947–1958.

Syakur, M. A., Khotimah, B. K., Rochman, E. M. S. & Satoto, B. D. (2018), 'Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile

Cluster', *IOP Conference Series: Materials Science and Engineering* **336**(1), 012017.
URL: *http://stacks.iop.org/1757-899X/336/i=1/a=012017?key=crossref.621d8c4eb09908543a2c62f980404c8f*

Tappan, J. H., Wright, M. E. & Sistler, F. E. (1987), 'Error sources in a digital image analysis system', *Computers and Electronics in Agriculture* **2**(2), 109–118.

Tennekes, M. (2018), 'tmap: Thematic maps in R', *Journal of Statistical Software* **84**(6), 1–39.

The Woodland Trust (2018), 'Portmoak Moss — Explore woods — The Woodland Trust'.
URL: *http://www.woodlandtrust.org.uk/visiting-woods/wood/4876/portmoak-moss/*

The Woodland Trust Scotland (2004), A guide to a community woodland, Technical report.
URL: *www.woodland-trust.org.uk*

Thornton, C., Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2013), 'Auto-WEKA', *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13* p. 847.
URL: *http://dl.acm.org/citation.cfm?id=2487629%5Cnhttp://dl.acm.org/citation.cfm?doid=2487575.2487629*

Tibshirani, R., Walther, G. & Hastie, T. (2000), 'Estimating the Number of Clusters in a Dataset via the Gap Statistic'.

Tiede, D. (2014), 'A new geospatial overlay method for the analysis and visualization of spatial change patterns using object-oriented data modeling concepts', *Cartography and Geographic Information Science* **41**(3), 227–234.

Vatsavai R.R., S. S. B. T. E. (2005), A semi-supervised learning method for remote sensing data mining, *in* 'Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI', Vol. 2005, pp. 207–211.
URL: *http://www.scopus.com/inward/record.url?eid=2-s2.0-33845884299&partnerID=40&md5=f8f3409e11e03791*

Wagstaff, D. A., Elek, E., Kulis, S. & Marsiglia, F. (2009), 'Using a nonparametric bootstrap to obtain a confidence interval for pearson's r with cluster randomized data: A case study', *Journal of Primary Prevention* **30**(5), 497–512.

Wang, C., Guo, Z., Wang, S., Wang, L. & Ma, C. (2015), 'Improving Hyperspectral Image Classification Method for Fine Land Use Assessment Application Using Semisupervised Machine Learning', *Journal of Spectroscopy* **2015**, 1–8.
URL: *http://www.hindawi.com/journals/jspec/2015/969185/*

Wang, R. (2016), 'Performance Measurements'.
URL: *http://fourier.eng.hmc.edu/e161/lectures/classification/node5.html*

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004), 'Image quality assessment: From error visibility to structural similarity', *IEEE Transactions on Image Processing* **13**(4), 600–612.

Woodroof, J. (2000), 'Bootstrapping: As easy as 1-2-3', *Journal of Applied Statistics* **27**(4), 509–517.

Xu, M. & Wei, C. (2012), 'Remotely sensed image classification by complex network eigenvalue and connected degree.', *Computational and mathematical methods in medicine* **2012**, 632703. **URL:** *http://www.ncbi.nlm.nih.gov/pubmed/22242041 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=P*

Xue, J. & Su, B. (2017), 'Significant remote sensing vegetation indices: A review of developments and applications'.

Yang, L., Liu, S., Tsoka, S. & Papageorgiou, L. G. (2017), 'A regression tree approach using mathematical programming', *Expert Systems with Applications* **78**, 347–357.