ANKARA YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

USAGE OF MACHINE LEARNING METHODS ON PRECISION AGRICULTURE APPLICATIONS

M. Sc. Thesis by

Yekta Can YILDIRIM

Department of Computer Engineering

January, 2021

ANKARA

USAGE OF MACHINE LEARNING METHODS ON PRECISION AGRICULTURE APPLICATIONS

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering, Department of Computer Engineering

by

Yekta Can YILDIRIM

January, 2021

ANKARA

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **“USAGE OF MACHINE LEARNING METHODS ON PRECISION AGRICULTURE APPLICATIONS”** completed by **YEKTA CAN YILDIRIM** under the supervision of **ASST. PROF. DR. MUSTAFA YENİAD** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Mustafa YENİAD
Supervisor

Assoc. Prof. Dr. Baha ŞEN
Jury Member

Assoc. Prof. Dr. Yalçın İŞLER
Jury Member

Prof. Dr. Ergün ERASLAN
Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: 18/01/2021 **Signature: ………………………………………...**

Name & Surname: Yekta Can YILDIRIM

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Asst. Prof. Dr. Mustafa YENİAD for his support and guidance. I also would like to thank to Asst. Prof. Dr. Özkan KILIÇ for sharing his knowledge and experiences with me. Lastly, I would like to Osman Selami BUDAK and Dr. Ayşe Ayşen KARAKAŞ in AVEO Bilişim Mühendislik Proje ve Danışmanlık Denetim LTD.ŞTİ for their support while preparing my thesis.

2021, January 18 **Yekta Can YILDIRIM**

USAGE OF MACHINE LEARNING METHODS ON PRECISION AGRICULTURE APPLICATIONS

ABSTRACT

Precision Agriculture technologies aim to help the farmers with the decision making process by providing them information and control over their land, crop status and environment using remote sensing systems. Vegetation indices derived from multispectral bands of the remote sensing systems carry useful information about the crops such as nitrogen content, chlorophyll content and water stress which supports the farmers to plan irrigation and pesticide spraying processes without the need of manual examination.

For this study, the aim was to explore the usage of machine learning on Precision Agriculture applications and the focus was on olive trees in Manisa, Turkey. Using the spectral band information gathered from an Orange-Cyan-NIR (OCN) camera embedded unmanned aerial vehicle (UAV) system, Normalized Difference Vegetation Index (NDVI) was calculated and the data was preprocessed by segmentating the tree pixels from background based on those values using MiniBatchKMeans algorithm. NDVI, Normalized Nitrogen Index (NNI), Normalized Difference Water Index (NDWI) and tree pixel sizes were selected as optimal features based on accuracy comparison for yield and disease predictions. A Decision Tree Regressor (DTR) model was trained for yield prediction while a Random Forest Classifier (RFC) model was trained for disease prediction.

The results showed that crop segmentation had an accuracy rate of 0.85-0.95, while DTR and RFC models had an R2 score of 0.95 and accuracy rate of 0.98 respectively, which displayed the importance and usefulness of vegetation indices.

Keywords: Machine learning, artificial intelligence, random forest classifier, decision tree regressor, precision agriculture, yield prediction, disease prediction, crop segmentation

HASSAS TARIM UYGULAMALARI ÜZERİNDE MAKİNE ÖĞRENMESİ TEKNİKLERİ KULLANIMI

ÖZ

Hassas Tarım teknolojileri çiftçilere arazileri, ürünlerinin durumu ve çevreleri hakkında uzaktan algılama sistemlerinin yardımıyla bilgi ve kontrol imkanı sağlayarak karar verme sürecinde onlara destek olmayı planlamaktadır. Uzaktan algılama sistemi multispektral bantlarından elde edilmiş bitki örtüsü indeksleri bitkilerin nitrojen, klorofil ve su miktarı hakkında önemli bilgi taşır, bu da çiftçilerin manüel bir incelemeye gerek kalmadan sulama ve ilaçlama planlamalarına destek sağlar.

Bu çalışmada hassas tarım uygulamalarında makine öğrenmesi tekniklerinin kullanımının keşfi hedeflenmiş olup, Türkiye'nin Manisa ilçesindeki zeytin ağaçlarına yoğunlaşılmıştır. Bir insansız hava aracı (İHA) sistemine entegre edilmiş OCN kamerasının spektral verilerini kullanarak, NDVI verisi hesaplanmış ve ağaç pikselleri bu veriyi kullanarak MiniBatchKMeans algoritmasıyla diğer piksellerden ayrıştırılmıştır. Rekolte ve hastalık tahmini için azami öznitelikler modellerin doğruluk oranlarının karşılaştırılması sonucu belirlenmiş ve NDVI, NNI, NDWI ve ağaç piksel büyüklükleri seçilmiştir. Rekolte verisi için Karar Ağacı Regresyon (KAR) modeli eğitilmiş, hastalık tahmini için Rastgele Orman Sınıflandırma (ROS) modeli eğitilmiştir.

Ağaç piksel ayrıştırma sonucunun doğruluk oranı 0.85 ve 0.95 arasında belirlenmiş, KAR algoritmasının R2 puanı 0.95 ve ROS algoritmasının doğruluk oranı 0.98 olarak hesaplanmıştır, bu da bitki örtüsü indekslerinin önemi ve kullanışlılığını vurgulamaktadır.

Anahtar Kelimeler: Makine öğrenmesi, yapay zeka, rastgele orman sınıflandırma, karar ağacı regresyon, hassas tarım, rekolte tahmini, hastalık tahmini, bitki ayrıştırma

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Since the beginning of mankind, gathering quality food has been the priority for survival. Hunting and gathering were the primary sources of food before the invention of tools and humanity's transition to sedentism. Sedentism brought population growth which caused a need to find constant source of food. While hunting and gathering were still viable options, they were often unreliable, time consuming, costly and most importantly insufficient. Using their inventions, humans learned to produce their own food by planting the seeds they have found in the wilderness. This was the starting steps of agriculture, one of the most important aspects of a developing civilization. Not only agriculture managed to provide food, it provided more than people could actually eat in a sitting, which was a good solution for the insufficiency of the previous methods.

Agriculture management brought its own set of problems such as soil nutrition, crop health monitoring, harmful insects, weed/wild seed growth, irrigation and yield estimation. In the past, it was possible for farmers to manually handle the fields in terms of problems mentioned above due to the small sized, naturally distinguishable fields. Following the growth in population, as mechanization replaced manual labor and intensive production required bigger fields, it became increasingly difficult for the farmers to manually tend to the soil and crops as they did before due to increased costs and environmental impact, which brought insufficient care, inaccurate spraying and irrigation, and unhealthy soil and crops over the land.

The development of technology, especially sensing systems and cameras, brought opportunities to handle such problems. Satellite imagery, with features such as providing geographic information with centimeter accuracy and information heavy representation of a given area with the help of multiple spectral bands contained within in different resolutions, was proven to be helpful to give fast, accurate results over an area.

The more recent technology, UAV systems with embedded specialized cameras, provides a more local, detailed crop report over a specific plot of land. While it lacks the reach, band count and quickness of satellite imagery, crop specific information can be gathered using this approach. Drone technology, while useful for monitoring health of the crops and soil, can also be used for automated spraying and irrigation based on the information gathered.

Agricultural monitoring and analysis of data to be used in management decisions to increase the productivity, quality, profitability, sufficiency and continuity of agricultural production is called precision agriculture, which uses the sensing systems to gather and interpret data.

Machine learning, the ability to learn from past data to predict future behaviour, can be used in order to make sense of the data gathered to make predictions over the land to determine irrigation and spraying areas, crop illnesses and amount of yield, which satisfies some of the aims of precision agriculture. Machine learning application on precision agriculture is a relatively new subject with growing interest and while there is room for improvement, gathering relevant and enough information leads to results with high accuracy.

This study explores the usage of machine learning methods over vegetation areas using a UAV system to segmentate crop areas, determine crop illnesses and predict yearly yield. Chapter 2 is separated into three sections and displays recent work on the usage of machine learning for each Precision Agriculture application stated above. Chapter 3 is focused on the materials and the methodology of this study. Chapter 4 presents the results of the proposed method. Chapter 5 discusses future work and ideas about how to improve the method further. Chapter 6 summarizes this study and gives final thoughts about the subject.

CHAPTER 2

LITERATURE REVIEW

This chapter covers the recent developments considering the applications of Precision Agriculture. The literature will be discussed in 3 sections.

2.1. Crop Segmentation

As real data is chaotic and images taken from a field contains both soil, vegetation and other elements that can affect the further processes, the first step is to separate the vegetation from the others. This section covers the recent works in this subject.

Bai et al suggested color segmentation in L*a*b* color space on images of rice plants. They proposed their own morphology modeling method on color distributions for different L slices. They based their model on the assumption of simply connected domains, a set of connected data in space, and of each L slice to have different color distributions. They compared their method with others and claimed that their method performed better, stating that their experiments were successful for different states of weather and illumination in the area. Their accuracy rate was 87.2% using ATRWG evaluation method and they claimed to get 96% accuracy rate using another evaluation method [1].

Torres-Sanchez, Lopez-Granados & Pena proposed an object based method for finding an optimal threshold to determine crop coverage using Object Based Image Analysis tool. They focused on wheat, sunflower and maize fields and their dataset was consisted of images taken from both visible Red-Green-Blue (RGB) and multispectral cameras embedded to a UAV. Their method was based on Otsu's thresholding algorithm and the optimal threshold was determined using Excess Green Index (ExG) and NDVI calculated from the band information of the cameras. They studied the effects of scale, shape and compactness on how their algorithm performed and claimed that while scale was a huge factor on their accuracy, the other parameters were not as influential. They found the optimal object scale for each field in order to get a lower classification error and reported their method as a success with error rates between 0% and 10% [2].

Huang, Li & Chen proposed a model for tree crown detection from UAV images using watershed segmentation algorithm. Their research focused on Osmanthus and Podocarpus trees. They applied bias field estimation, which they described as a method originally used to correct and smooth over magnetic resonance images (MRI), on their data and created a correction model using the estimation. They used the model to smooth over their images for segmentation, however claimed that watershed algorithm did not perform well due to high spectral variance of the UAV imagery. Instead, they used the bias field itself which they described to have smoother reflectance changes. They reported that compared to the original images, watershed algorithm performed significantly better with the bias field images. They used F-score to evaluate their detection rate and got results of 98.2% and 93.1% for Osmanthus and Podocarpus trees respectively [3].

Lottes et al worked on sugar beets and proposed a segmentation algorithm on UAV images for crop and weed classification. They first applied thresholding on the field, taking ExG index as input to single out the green areas from the image. Then they extracted the features of the green areas by combining object and keypoint based feature extraction, which were based on contour detection and nearest neighbour detection respectively. With the assumption that the crops were planted in parallel lines, they detected each row and calculated the distance between each to add as features for their model. They also found the geometrical shape of each object and extracted additional features. They trained a RFC model with the features they gathered from preprocessing and evaluated their success rate by recall and precision evaluation metrics. Their results showed a recall of 78% and precision of 90% [4].

Mancini et al used UAV data to perform segmentation on vineyard areas. They first synthetically calculated the Digital Surface Model (DSM) of the area to prepare their model. Claiming that global thresholding wouldn’t work on DSM files due to the possibilities of sharp changes in terrain slopes, they preprocessed the DSM file by image slicing and using 2D convolution kernels to act as adaptive thresholding filters. They processed each sliced image differently to handle the data as a time series, derived tree/terrain information and reported that the model works better with sliced images compared to global thresholding on the original image. They also worked with multispectral bands, NDVI and DSM information and applied terrain/tree classification using AdaBoost algorithm. Their evaluation metric was the accuracy score and their results showed for 96% for their height map convolution/slicing method and 96% for AdaBoost classification model [5].

Onyango and Marchant proposed a color based segmentation algorithm on cauliflower plants with the knowledge of plantation grids and growth stages of the plant in different states of time using RGB images. They used morphological filtering and line fitting to determine the central points of the grids and bivariate Gaussian distribution model to find the likely crop pixel distributions. For post-processing, they applied texture analysis using a morphological model in order to classify the weed and crop pixels close to each other. They evaluated their model by classification rate and their results showed 82%-96% for cauliflower classification and 68%-92% weed classification [6].

Dyson et al proposed a deep learning model for crop segmentation on vineyards using the synthesized height map of the area and vegetation indices derived from multispectral cameras. They compared two known algorithms for DSM segmentation named FANSCAN and CARSCAN and stated that the quality of the segmentation is maximal using the former. They created another index by the combination of FANSCAN-segmentated DSM and NDVI indices. They tried both NDVI, DSM and DSM/NDVI index on a small, non-specialized object detection Convolutional Neural Network model and stated that the combined index performed better than the others. Their results showed 65% improvement rate over pixel recognition using the combined index compared to standard indices [7].

2.2. Yield Prediction

Estimation of yield before harvest helps the farmers to plan import/export decisions and tend to areas with low yield values by irrigation, crop rotation or spraying. This section explores the recent works based on the subject.

Pantazi et al worked on estimating yield of a wheat field and compared three neural network algorithms named Counter Propagation Artificial Neural Network (CP-ANN), XY-Fused Network (XY-Fs) and Supervised Kohonen Network (SKN) using satellite imagery. They used soil sensors to collect soil based information such as acidity, magnesium and calcium on-site and created a dataset that also contained NDVI index derived from satellite band information and yield data for previous two years to train their models. They handled the estimation as a classification problem and used [Low Yield, Average Yield, High Yield] labels for prediction. They used cross validation evaluation metric and their results showed that SKN had the best performance compared to the other algorithms with 91% for low yield class, 70% for average yield class and 83% for high yield class with an overall accuracy of 81.65% while CP-ANN and XY-Fs had the overall performance of 78.3% and 80.92% respectively [8].

Ashapure et al proposed an artificial neural network model for yield estimation of tomato fields using UAV imagery. They used canopy object height map, crop pixel volume based on plantation grid, crop growth stage, ExG index and weather information such as temperature and humidity as an input to estimate yearly yield. They used a 3-layered artificial neural network (ANN) for prediction and tested their model on 3 yearly acquired yield datasets. They first evaluated their model based on a single year and reported that they got R2 scores in between 0.78 and 0.89. Their next trial was a multi-year yield estimation which they trained the model with one of the datasets and tested their model on the other datasets. They stated that the accuracy rate decreased due to lack of data and they got an R2 score of 0.70 [9].

Panda et al proposed Back Propagation Neural Network (BPNN) algorithm on corn fields using 4 vegetation indices; NDVI, Green Vegetation Index (GVI), Soil Adjusted Vegetation Index (SAVI) and Perpendicular Vegetation Index (PVI). They gathered data from sites with diverse conditions such as different amount of nitrogen and irrigation in order to increase the variance in their data. They divided the vegetation index images into grids and found means and standard deviations for each grid as additional inputs for their model in addition to index data. They worked on 3 separate yearly yield datasets and created 4 different models for vegetation indices to determine which index performed better with a total of 12 BPNN models. They evaluated their model individually for each year and with mixed samples from each year using R2 score and accuracy score evaluation metrics. They determined that PVI index was more consistent and performed better than the other indices with an average accuracy score of 94% and an R2 score of 0.45 for mixed samples and 83.50%-96.04% individually. In order to increase the accuracy of their model, they applied $\log_{10}$ transformation function on their PVI data and reported the performance increase by an accuracy score of 93% and an R2 score of 0.72 for mixed samples and 90%, 97% and 98% for each year respectively [10].

You et al proposed a Convolutional Neural Network (CNN) and Long-short Term Memory Network (LSTM) model on soybean fields using a RGB multispectral camera and stated that they applied dimensionality reduction on the data to increase the learning capabilities of their models by transforming each multispectral band into histogram bins and handling them separately with the assumption of permutation invariance, which means no spatial dependence within the features, between the location of pixels and the estimated yield. They used the temporal model LSTM, which they stated would work better due to the sequential nature of the histogram data. They also trained a CNN model by stacking the histograms to create a 3D representation of data. They reported that while their models worked accurately, their assumption of permutation invariance would not hold in some cases and supported their idea with an example of locationwise importance considering soil data. They claimed that since locationwise soil data did not have a notable change over time, a pattern can be derived from the data. They used a Gaussian model on top of their trained models to achieve that and noticed improved accuracy rates over most cases. They evaluated their models using the Root Mean Squared Error (RMSE) metric and reported that CNN, combined by the Gaussian Process gave the best results with an average of 5.55 [11].

Gonzalez-Sanchez, Frausto-Solis & Ojeda-Bustamante analyzed the performance of some of the machine learning techniques on crop yield estimation. They worked on different crop species and selected crop location, irrigation status, weather, temperature and season-duration cultivar information as their parameters to train their models. They selected Multiple Linear Regression (MLR), ANN, K-Nearest Neighbour (KNN), Support Vector Regression (SVR) and M5-Prime regression trees for comparison and reported that M5-Prime and KNN performed the best with an RMSE of 5.14 and 4.91, Root Relative Squared Error (RRSE) of 79.46% and 79.78%, Mean Absolute Error (MAE) of 18.12% and 19.42% and correlation coefficients (R) of 0.41 and 0.42. They concluded that M5-Prime performs generally better on all crop datasets they have worked on and thus outperforms the other methods [12].

Nari and Yang-Won focused on corn yields on Iowa state in United States. They gathered satellite and climate information and chose temperature information, NDVI, Enhanced Vegetation Index (EVI), Leaf Area Index (LAI), Fracture of Photosynthetically Active Radiation (FPAR), soil moisture and the vegetation water consumption and evaporation rate as parameters to train their models. They implemented Random Forest (RF), Deep Learning, Support Vector Machine (SVM) and Extremely Randomized Trees (ERT) models and made a comparative analysis using three datasets with different time periods. They reported that Deep Learning model performed better in terms of R score for all three datasets, 0.776, 0.796 and 0.800 respectively [13].

Schwalbert et al focused on soybean fields in Brazil and proposed a model that combines weather data with satellite imagery. Taking NDVI, EVI, surface temperature and rain information as input, they trained 3 different machine learning models for comparison; RF, Multivariate Regression (MVR) and LSTM respectively. They stated that their aim was to accurately estimate yield information as early as possible, handling their data as a Time Series. They divided their dataset into four different time portions in increasing order to estimate the models' performances on early forecasting using MAE, RMSE and Mean Squared Error (MSE) evaluation metrics. They reported that LSTM performed better for the last three time portions with a MAE of 0.42, 0.25 and 0.24 compared to RF's MAE of 0.44, 0.37 and 0.32 and MVR's MAE of 0.46, 0.40 and 0.32 respectively, while the MVR performed better for first time period with a MAE of 0.42 compared to LSTM's MAE of 0.52, noting the decrease of performance in the first period for most cases [14].

Maimaitijiant et al proposed a deep learning model on soybean fields in Columbia, Missouri, USA combining RGB, multispectral, thermal, texture, Digital Elevation Model (DEM), DSM and Canopy Height Map (CHM), the subtraction of DEM and DSM, data gathered from sensors embedded to a UAV. They implemented Partial Least Squares Regression (PLSR), RF, SVR, input level feature fusion based DNN (DNN-F1) and intermediate level fusion based DNN (DNN-F2) and compared their performance for individual and combined parameters. They reported that combined parameters increase yield prediction accuracy and DNN models perform better than other methods for the validation datasets they provided with DNN-F2 coefficient of determination (R2) score of 0.72 and an RMSE of 0.15. They stated that DNN models were less affected by saturation and had a stable performance for lack of spatial variation and dependency [15].

2.3. Disease Prediction

Early detection of crop diseases is another important subject for the farmers to control the damage, plan pesticide spraying operations and separate diseased crops from the healthy to prevent spreading. This section covers the recent work based on the subject.

Ferentinos implemented multiple CNN models to detect disease over several crop species. He trained AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat and VGG models with healthy and diseased plant leaf images and evaluated and compared each model's performance. He stated that one third of his training dataset consisted of images of plants in real life conditions while the rest was taken in laboratory conditions, and training his dataset with the combined dataset gave the best results. His results showed that the VGG model outperformed the other models with 99.53% accuracy rate. He experimented on selecting one dataset to train the model and making predictions for the other one and reported a significant decrease of accuracy in both cases. While the model trained with the pictures of plants in real conditions could predict the plants in laboratory conditions with 68% accuracy, model trained with laboratory conditions could only predict the pictures of plants in real conditions with 33% accuracy. Based on the results, he emphasized on the importance of adding more real data to the dataset, stating that identification of plants in real conditions is more complex and difficult [16].

Khamparia et al focused on several crops and diseases and proposed a Deep Convolutional Encoder Network (DCEN) for classification. Their dataset was consisted of healthy and diseased crop leaf images. They combined a CNN model with autoencoders, which they described as a type of ANN that is used for learning useful encodings in the data. Their model is consisted of 3 layers; convolutional encoder layer, max pooling layer and fully connected layer respectively. They batch normalized each layer in order to increase the performance of their model. They used rectified linear unit (ReLU) activation function to introduce non-linearity. Their results showed 97.5% accuracy rate using 2x2 convolution filter while 100% with 3x3 convolution filter and they stated that their model outperformed the traditional models [17].

Pantazi et al focused on the detection of a Microbotryum silybum, which is a type of fungus affecting milk thistle weed. They used a handheld, RGB+NIR spectrometer to determine the leaf-based reflectance values of healthy and diseased crops. They preprocessed the data by applying normalization, second derivative and principle component analysis (PCA) dimensionality reduction to find the features most relevant to the prediction of diseases and increase accuracy. They implemented SKN, CP-ANN, XY-F models and evaluated their performance on the preprocessed dataset and reported that CP-ANN and SKN models acquired up to 90% overall accuracy rate while XY-F model performed generally better with 95.16% [18].

Chung et al focused on foolish seedling disease on rice seedlings using morphological and color analysis. They used flatbed scanners that provided RGB imagery to acquire data which they converted to CIE color space for ease of color comparison. Morphological analysis was done considering the anatomy and the appearance of rice seedlings, such as seedling shape, width and aspect ratio. They implemented a SVM model and selected the optimal features using a genetic algorithm. They determined a fitness value equal to weighted summation of accuracy, sensitivity and specificity evaluation metrics. By randomly selecting features to train the SVM model, they determined the combinations that gave the best fitness value results and applied genetic algorithm's mutation and crossover operations on those pair of combinations to create the next generation until there was no increase in fitness values. They selected the combination that gave the best fitness value as the optimal feature group to train the SVM model. They evaluated their model and reported that the classification was performed in 87.9% accuracy rate [19].

Duarte-Carvajalino et al focused on late blight disease on potato crops. They gathered their data using Blue-Green-NIR camera and implemented ANN, RF, SVR and CNN for a comparative analysis. They divided the images into smaller batches using sliding window technique and created 4 datasets using the batches, spectral differences of the bands, NDVI derived from NIR/BLUE bands and principle component analysis and sliding window technique applied original image information to train their models. They evaluated their models for each dataset and the results showed that CNN models performed better than the other models using the batch NIR+G+B information with a MAE of 11.72, RMSE of 15.09 and R2 score of 0.74, while RF performed better for a non-image model using band difference dataset with a MAE of 12.96, RMSE of 16.15 and R2 score of 0.75 [20].

Moshou et al proposed a neural network algorithm on classifying diseased and healthy wheat plants focusing on yellow rust disease. They used multispectral imagery and stated that their aim was to determine the spectral differences between healthy and diseased wheat crops. They implemented multilayer perceptron (MLP) and Quadratic Discriminant (QDA) models for comparison and Self Organizing Map (SOM) model for visualization of the class relationships. They normalized their data to increase accuracy and applied stepwise discriminant analysis on their data for feature selection and trained their models with the selected features in addition of NDVI bands (NIR/RED). They reported that MLP outperforms the QDA with accuracy rates in between 95% and 99% [21].

Ebrahimi et al focused on thrips insects on strawberry fields. They used RGB imagery to acquire their data. They used Otsu's thresholding algorithm to segmentate the foreground images from the background and continued the iteration until they singled out the thrips. They took hue, saturation and intensity values in addition to a regional index that considered the flower sizes and trained an SVM model. They evaluated their model using Mean Percentage Error (MPE), MSE, RMSE and MAE evaluation metrics and stated that SVM worked well prediction the intensity values with an MPE of 2.25% [22].

CHAPTER 3

METHODOLOGY

As discussed in Chapter 2, remote sensing systems are used in order to gather data. There are two ways to acquire information, either by UAV or satellite, using specialized cameras that can capture multiple wavelengths of electromagnetic spectrum, including the wavelengths invisible to human eye, of an image using filters, which separate the aforementioned wavelengths into bands. Each band can be thought like a separate image that carry reflectance information of a specific wavelength with well defined bandwith. Those specialized cameras are called multispectral cameras and the image produced from those cameras are called multispectral imagery. The information provided by the band data of the imagery is used to segmentate the crops, detect illnesses and predict monthly yield by training a machine learning model that takes the information as input.

Satellite imagery covers a much larger area and contains more band information than UAV multispectral cameras yet it lacks the resolution that UAV imagery provides. Table 3.1 displays the comparison between two methods.

Table 3.1: UAV and Satellite Imagery Comparison

Methods Topics	UAV Imagery	Satellite Imagery
Cost	Travelling + drone repairs	High resolution imagery
Area of interest	Local	Wide
Bands	Limited	High amount
Spatial Resolution	< 6 cm, Very high resolution	Sentinel 2A resolution = 10 meters, Landsat 8 resolution = 30 meters

Table 3.1 (cont.): UAV and Satellite Imagery Comparison [23]

Data acquirement	User defined	Acquisition occurs every 10-15 days
Weather/Cloud coverage	Doesn't get affected by cloud coverage, can't fly in rain.	Affected by cloud coverage, weather conditions decrease the quality of image.

As there is a significant loss of resolution and delay of data acquisition for satellite imagery, it is not useful for tree based operations that require more sensitivity. Satellite imagery can describe general information about the land's health, nitrogen and water status, which, while useful in other contexts, does not provide meaningful information for the purposes of this study.

3.1. Unmanned Aerial Vehicle

3.1.1. Data Gathering

The data gathering process consists of 4 steps;

1. Acquiring the data.

2. Calibrating the data for clearer image.

3. Creating DSM and Digital Terrain Model (DTM) images

4. Creating orthomosaic image

This study focuses on the crop segmentation, illness and yield prediction using machine learning techniques on olive tree fields in several districts of Manisa, Turkey and the tree data was gathered using a multispectral camera over several plots of land in May, July, August and September of 2019 to increase the data count and give variability to the dataset by introducing trees in variable states. The training data was created using the band information gathered from the cameras to explore the importance of light's reflectance and vegetation indices derived from that information. Vegetation indices will be discussed in Section 3.1.2. The plot numbers of study fields are given in Table 3.2.

Table 3.2: Study field plot numbers within the districts of Manisa, Turkey

District Name	Plot Number
Akhisar	240, 617, 618
Gölmarmara	376, 388, 398, 432
Turgutlu	533, 583, 997, 997
Saruhanlı	323, 325, 327, 332
Alaşehir/Toygar	266_19, 266_20, 267_2

Figure 3.1 depicts the flow of processes to acquire data.

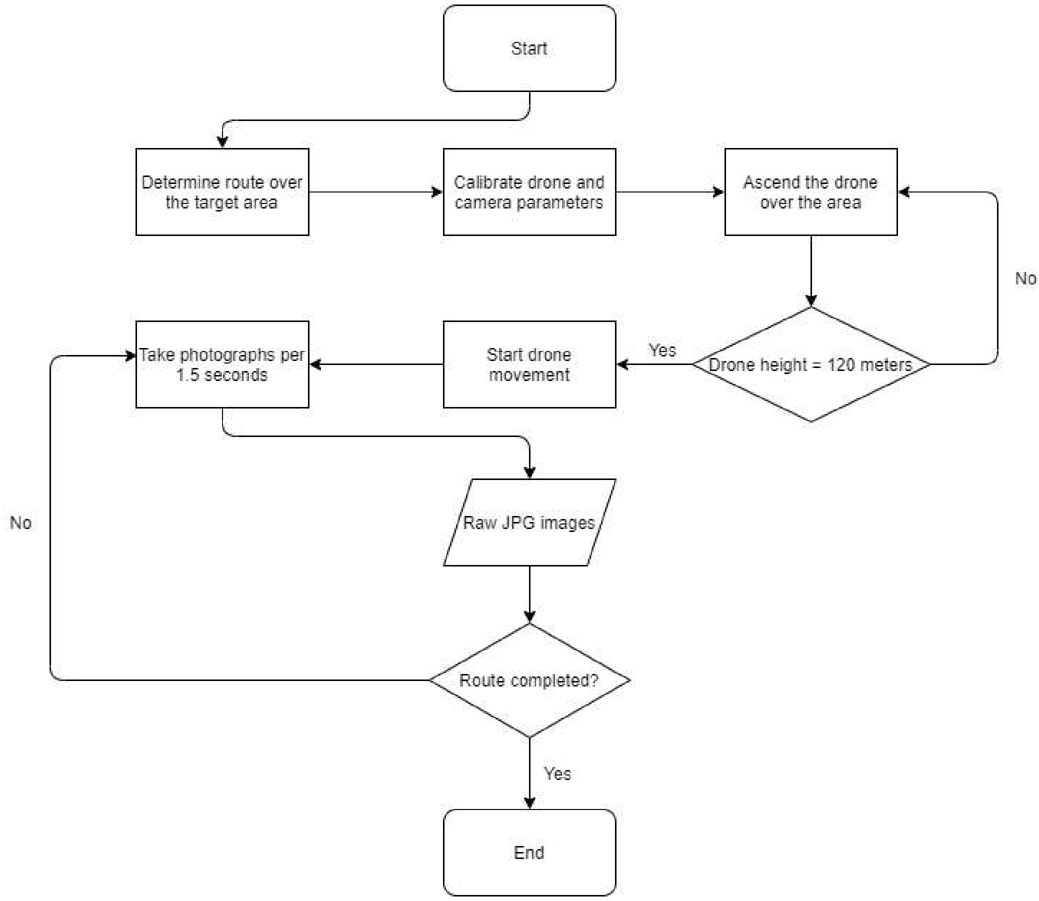


Figure 3.1: Data gathering flowchart

A flight plan was constructed which determines the boundaries of an area of interest and a multispectral camera embedded UAV was flown over that area, taking photos in 1.5 second intervals from a height of 120 meters above. Figure 3.2 shows an example flight plan.

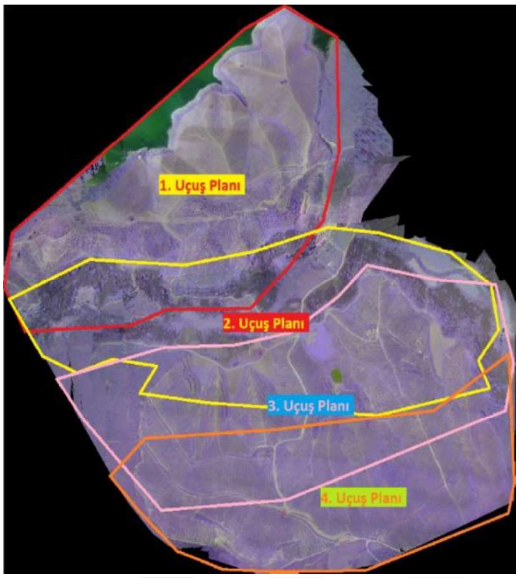


Figure 3.2: UAV flight plan for gathering data

The drone keeps a flight log as the flight commences which shows the route the drone takes and photographs taken along the flight. Information stored in the log file can be reviewed by using an external software. Figure 3.3 shows the log file reviewed in MissionPlanner.

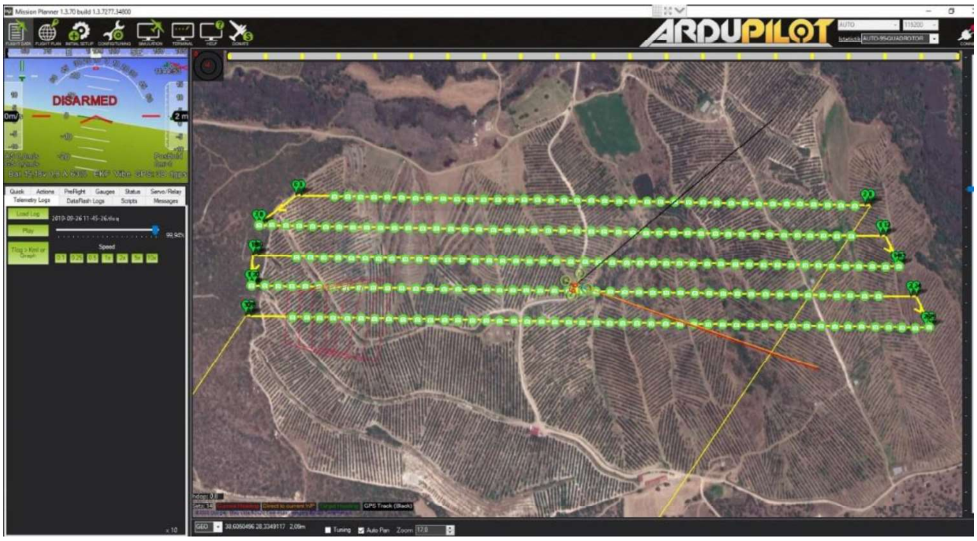


Figure 3.3: Log file information

The multispectral camera used for this study is Mapir Survey 3, which is a 12 megapixel camera that filters 3 spectral bands. Table 3.3 shows camera specifications.

Table 3.3: Mapir Survey 3 specifications [24]

Sensor Resolution (megapixel)	12
Image Type	JPG
Sensor Size (mm)	6.20 x 4.65
Sensor Pixel Size (um)	1.55
Lens Focal Length (mm)	8.25(3N)
Image Dimensions (pixels)	4000 x 3000
Capture Speed (sec)	1.5
Shutter Speed (sec)	Auto

In order to capture clear and non-blurry data, Survey 3 comes with a shutter whose timer can be set. For the purposes of this study, shutter speed was set to Auto, which means the shutter closes automatically after gathering enough light. As mentioned before, Survey 3 can capture 3 different spectral bands, RGB, RGN(RED-GREEN-NIR) or OCN respectively.

According to Mapir website, OCN camera is an improvement over RGN camera, using different wavelengths in order to reduce soil noise and blurring, and give a more defined color transition while calculating vegetation indices. Soil noise occurs since it reflects a lot of red light, causing the soil to be surrounded by yellow and the spectral image to become blurry. Figure 3.4 shows the comparison of OCN and RGN cameras in terms of soil noise and data clarity [25].

Figure 3.4: OCN and RGN camera data comparison [25]

Compared to RGN, OCN camera bandwidths are wider to provide more contrast. Figure 3.5 shows the light transmission percentage for both cameras, which means the range of wavelengths that pass through the filters [25].

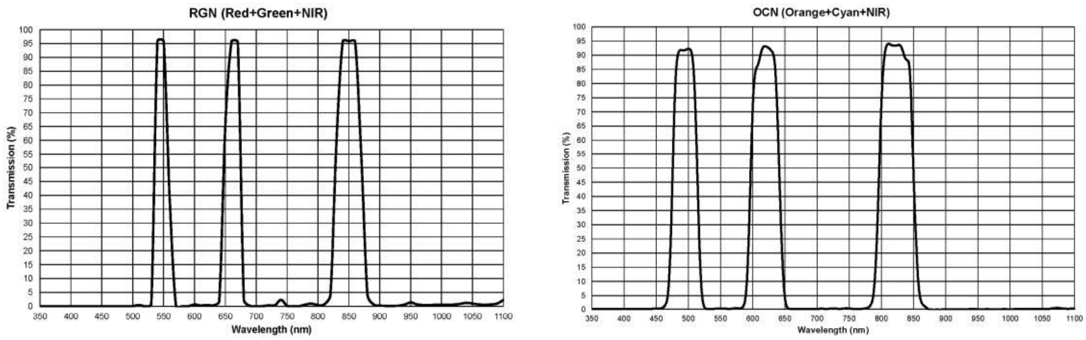


Figure 3.5: Light transmission percentage for OCN and RGN cameras [25]

As seen from Figure 3.5, for OCN, NIR wavelength transmission rate centers around 808nm, Orange wavelength centers around 615nm and Cyan wavelength centers around 490nm light. For RGN, NIR wavelength transmission rate centers around 850m, while Red and Green transmission rates center around 660nm and 550nm light respectively.

OCN camera was used for data gathering as it provides more accurate information, After the images are acquired, the next step is to create a map in order to start processing the data.

3.1.1.1. Creation of Orthomosaic, DSM and DTM Files

Since the images are geo-codded, it is possible to combine the images to create a spatial model for the area. This process is done by using a photogrammetry software specialized in drone mapping and the result is named orthomosaic image. Pix4DMapper was selected for the mapping process, which is a very popular and effective software that produces fast results.

Figure 3.6 shows the general workflow of the creation of required files for further processing using Pix4DMapper and MAPIR Camera software.

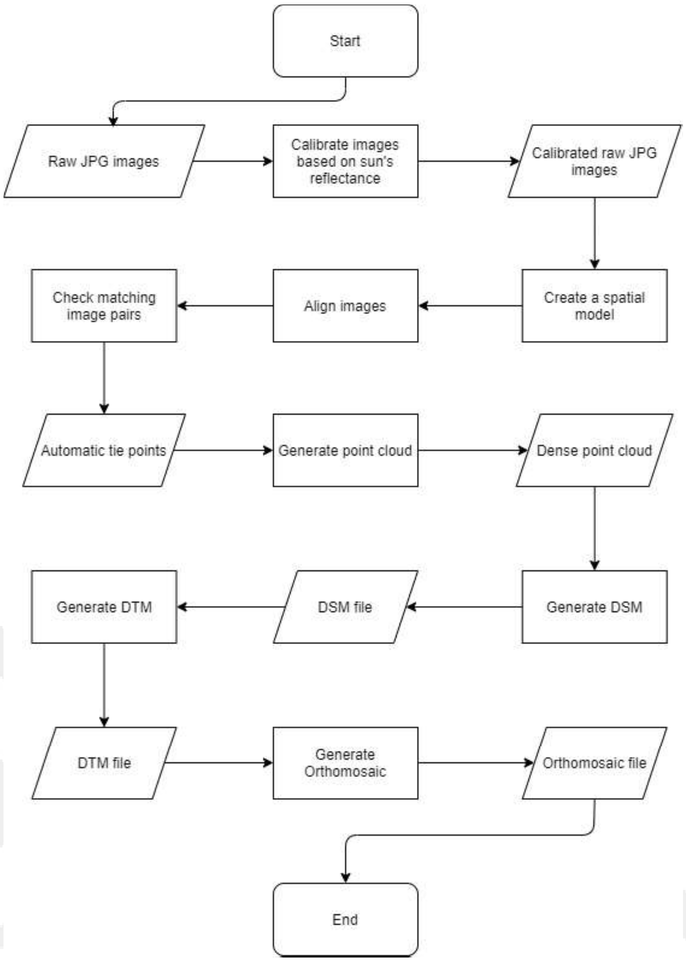


Figure 3.6: Pix4DMapper and Mapir Camera software workflow

Sun's reflectance over the area is an important factor that can affect the accuracy of the data. Mapir Survey 3 provides a Calibration Ground Target to solve that problem, which contains 4 panels that have distinct reflectance curves. Figure 3.7 shows the device.

Figure 3.7: Mapir Camera Reflectance Ground Target

The device is put onto a shadow and debris free area faced straight up, then the drone is held over the target from a height of approximately 3 meters while keeping the focus of the multispectral camera centered on the panels and a photograph of the device is taken. Using MAPIR Camera Control software, which takes camera model, lens, filter, calibration target and uncalibrated photograph folder as input, the photographs are calibrated based on calculated reflectance values from the target. Figure 3.8 displays the software tool.

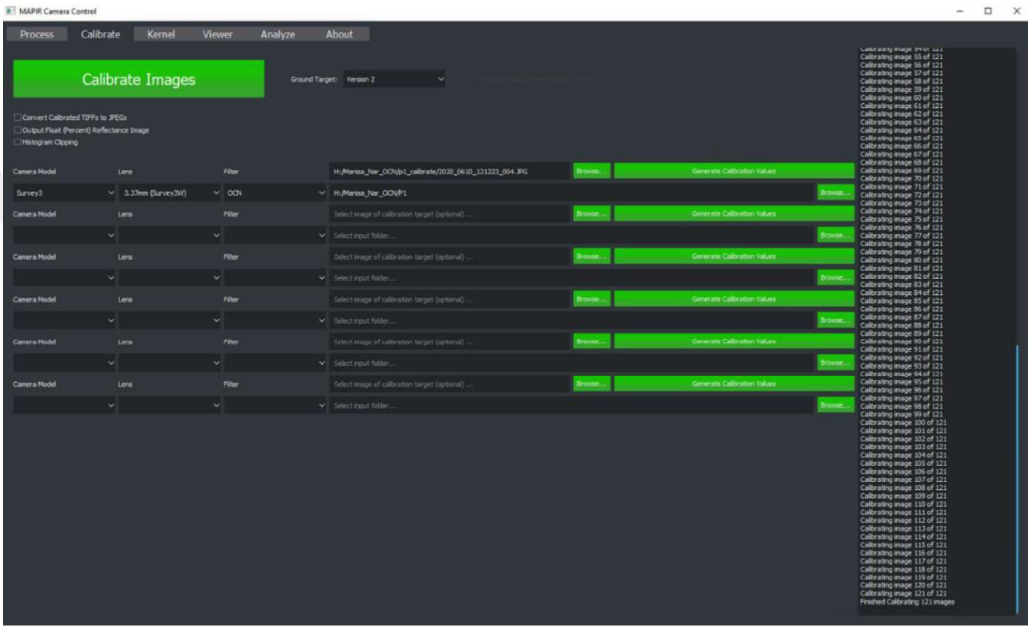


Figure 3.8: MAPIR Camera image calibration

Once the calibration process is completed and the directory of the images is selected, Pix4DMapper automatically detects the camera type and the coordinate system of the geolocated images and calibrates the process according to the camera specifications. Geolocated images also contain altitude, horizontal and vertical accuracy information in addition to coordinate information, which are used to make the mapping process smoother. Figure 3.9 provides an example for the mentioned parameters.

Enabled	Image	Group	Latitude [degree]	Longitude [degree]	Altitude [m]	Accuracy Horz [m]	Accuracy Vert [m]
☑	2020_0610_1212...	group1	38.73332925	28.48422036	404.300	5.000	10.000
☑	2020_0610_1212...	group1	38.73332925	28.48422036	404.200	5.000	10.000
☑	2020_0610_1212...	group1	38.73333331	28.48421631	404.300	5.000	10.000
☑	2020_0610_1212...	group1	38.73333331	28.48421631	404.500	5.000	10.000
☑	2020_0610_1212...	group1	38.73333331	28.48421631	404.800	5.000	10.000
☑	2020_0610_1212...	group1	38.73333331	28.48421631	404.800	5.000	10.000
☑	2020_0610_1219...	group1	38.73744708	28.48318683	520.400	5.000	10.000
☑	2020_0610_1219...	group1	38.73747964	28.48346353	520.500	5.000	10.000
☑	2020_0610_1219...	group1	38.73752033	28.48375242	520.900	5.000	10.000
☑	2020_0610_1219...	group1	38.73755694	28.48403725	521.300	5.000	10.000

Figure 3.9: Pix4DMapper Image Properties

Accuracy of the generation of orthomosaic, DSM and DTM files is determined by Ground Control Points. which are points on the ground with known coordinates in the spatial coordinate system, Knowing these points makes the orientation and alignment of the photographs possible, so the mapping process can work properly. With enough knowledge about the terrain, it's possible to select the GCPs manually but as the drone used for this study has an embedded GPS tool, photographs already contain geographic coordinate information, which are used by Pix4DMapper to create GCPs automatically. Figure 3.10 shows the generation of GCPs and the alignment of the photographs.

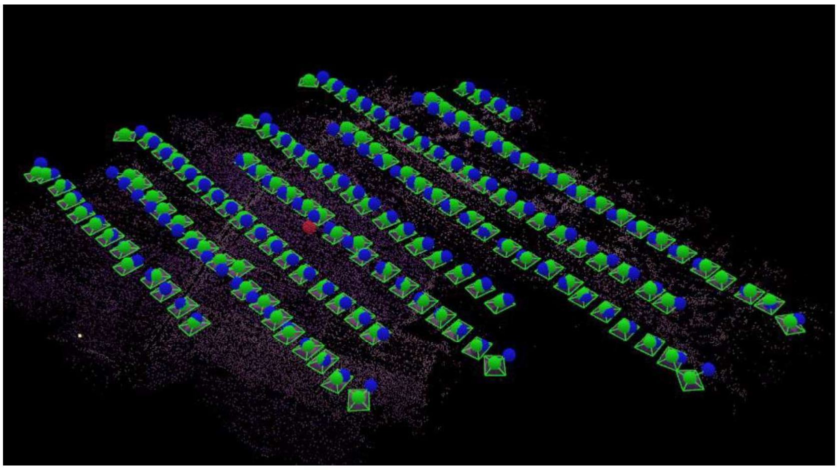


Figure 3.10: Alignment of images (green) according to automatically generated GCPs (purple)

Then the program checks for matching image pairs, which means a pair of overlapping images that show the same part of the area from different positions. Its principles are the same as the human visual system; just as the human brain creates a model of the area from the information it gets by combining the images from both left and right eye, the program creates a more detailed map by using the pair of overlapping images of the area. Thus, if an area is covered by multiple overlapping images, it will appear more detailed in the orthomosaic image. Figure 3.11 provides an example coverage map of an area.

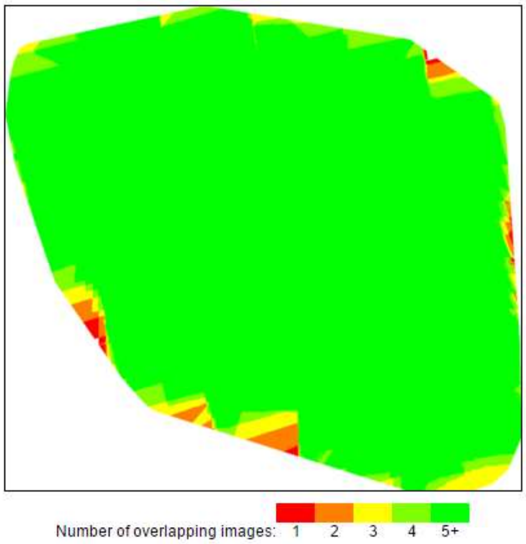


Figure 3.11: Area coverage map

Comparison of an area with multiple and insufficient overlapping images can be seen in Figure 3.12. While the area with multiple overlapping images is a lot clearer and more detailed, the other area is distorted and blurry which proves the importance of overlapping images in photogrammetry and mapping.

Figure 3.12: Comparison of an area with multiple (left) and insufficient overlapping images (right)

The mapping process takes between 2 to 12 hours, depending on the number of photographs and the quality settings. First, the program creates the DSM and DTM files. DSM is the approximation and the generation of a height map of the surface of the area, considering both the terrain and the objects on the terrain. DTM considers only the height of the terrain of the area. Difference of the two gives the Digital Canopy Model (DCM) or Canopy Height Model (CHM), which is the height of the objects only, This information is particularly useful to segmentate and determine the age of the trees. Figure 3.13 displays the comparison of DSM and DTM.

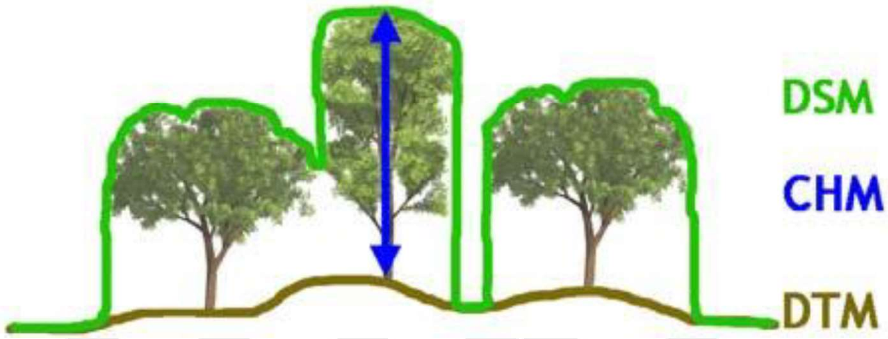


Figure 3.13: DSM and DTM comparison [26]

When the DSM and DTM creation is completed, the program then starts generating the orthomosaic file. The resulting images are in GeoTIFF (tiff or tif for short) format which contain combined band and coordinate information.

3.1.2. Dataset Creation

Using the information gathered from DSM, DTM and orthomosaic files, dataset creation process is started. Figure 3.14 shows a flowchart that summarizes the next two subsections.

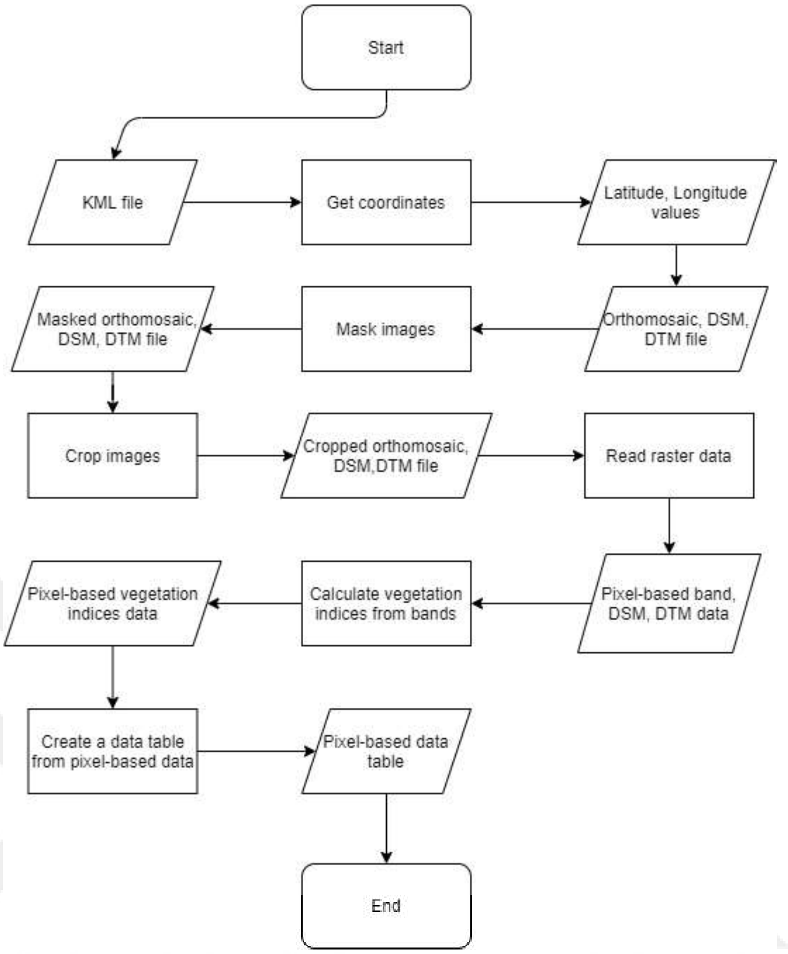


Figure 3.14: Dataset creation flowchart

3.1.2.1. Cropping the Data

The created files are big in size as they cover bigger areas while the target area is usually quite smaller. Processing the files without editing consumes too much memory due to the amount of pixels need to be handled. Thus, Keyhole Markup Language (KML) files were used in order to crop the target area. KML is a specialized Extensible Markup Language (XML) file that contains important geographic information about an area which can be used to display in an Earth browser such as Google Earth. The structure of a KML file is given in Figure B.1.

While some aspects of the KML file may change according to the geographical data shared within, coordinate information is kept inside a similar tag structure which makes the automation of extracting coordinates possible. Once the file is opened, the coordinate information is used to pinpoint the location. Figure 3.15 provides the visual representation of a KML file on Google Earth.

Figure 3.15: Visual representation of a KML file

A KML file can be acquired in several ways.

1. Google Earth Pro: From the application, clicking the “Draw Polygon” option then naming and saving the result as a KML file. Figure 3.16 shows the creation process of a KML file.

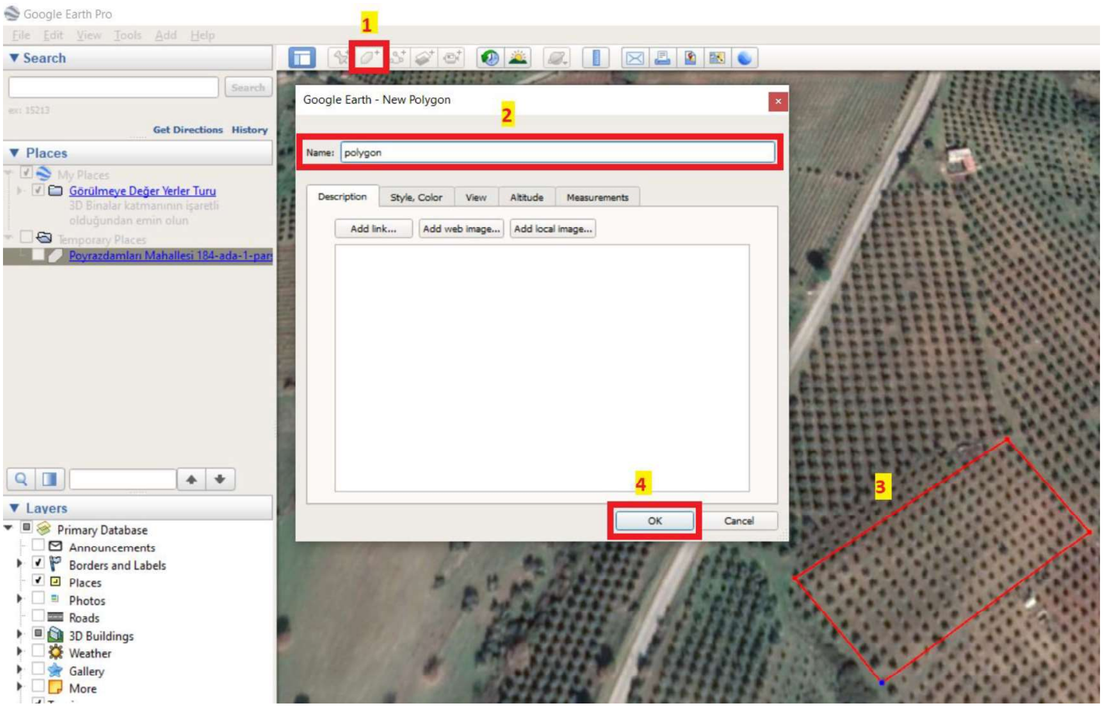


Figure 3.16: KML creation on Google Earth Pro

2. QGIS: Quantum Geographic Information System is an application which supports viewing, editing and analyzing geospatial data. This tool can be used for preanalyzing the data and discovering patterns which helps with the fine-tuning of parameters. The tool also handles the process of cropping big data using shapefiles. Shapefiles can be exported as a KML file after the process is done. Figure 3.17 shows the steps to create a shapefile.

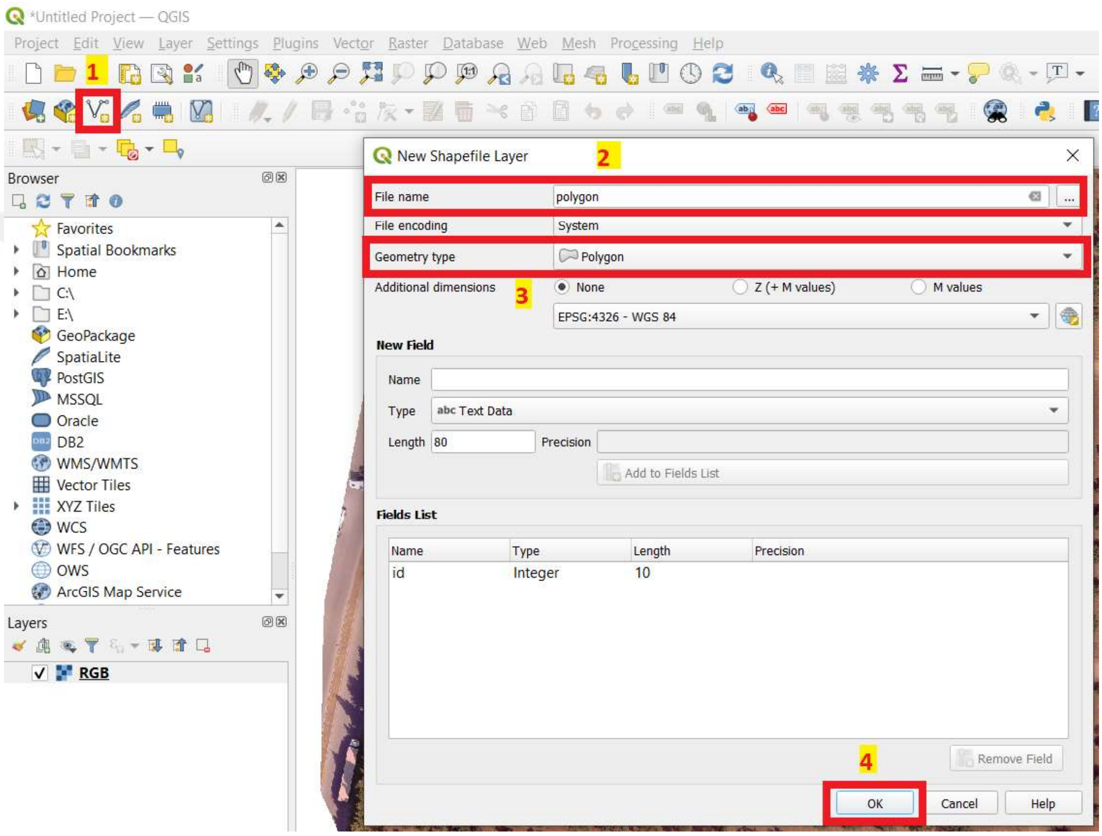


Figure 3.17: QGIS Shapefile creation

After generating the shapefile, it is edited and a polygon is drawn for the target area. The drawn polygon is exported as a KML file once the drawing process is done. Figure 3.18 shows the steps to draw the polygon.

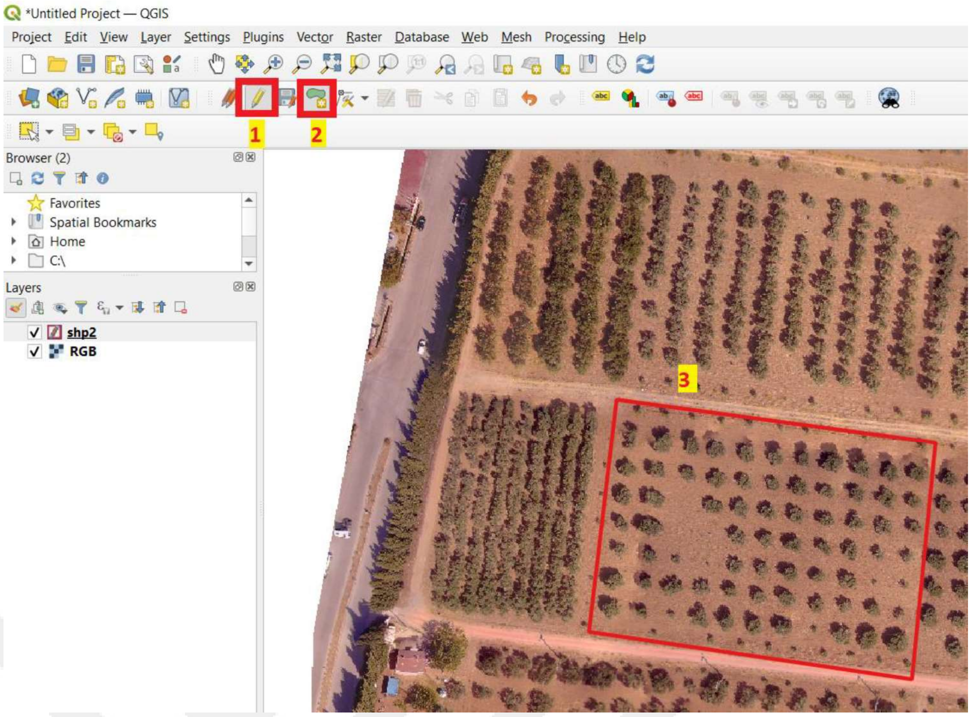


Figure 3.18: Creation of a shapefile polygon on QGIS

3. Land Registry Database: If the block and plot number of the land is known, it is possible to search for the land in the database and save the result as a KML file.

Method A.1 explains the cropping process after the KML file is acquired, The process is applied for all GeoTIFF files.

3.1.2.2. Data Characterization and Feature Selection

Geospatial Data Abstraction Library (GDAL) is a software tool that is used for reading and writing raster geospatial data format. A raster image is a bitmap, which is a matrix of individual pixels that creates an image. Using this function of the library, band information of the images was extracted to create a dataset.

Each pixel in the image has its own 3 band information ranged between 0 and 255. However, multispectral bands are not enough for the segmentation process as they only relay the base reflectance information. Vegetation indices were also used in order to differentiate the pixels further. Vegetation indices are equations which highlight a particular property of vegetation using different reflectance values and they often represent and can be grouped by 3 properties; health, nitrogen content and water stress. Table 3.4 provides examples for some vegetation indices with different purposes.

Table 3.4: Vegetation Indices

HEALTH		
Vegetation Index	**Formula**	**Usage**
Normalized Difference Vegetation Index	$NDVI = \frac{NIR - RED}{NIR + RED}$	Highlights vegetation's photosynthetic activity.
Green NDVI	$GNDVI = \frac{NIR - GREEN}{NIR + GREEN}$	More sensitive to chlorophyll than NDVI
Soil Adjusted Vegetation Index	$SAVI = \left(\frac{NIR - RED}{NIR + RED + L}\right) * (1 + L)$	Used on areas where vegetation cover is low and soil brightness is an influence.
Enhanced Vegetation Index	$EVI = L * \left(\frac{NIR - RED}{NIR + L * RED + L * BLUE + L}\right)$	Decreases soil noise and corrects atmospheric conditions. Used in areas with high concentration of vegetation.
Atmospherically Resistant Vegetation Index	$ARVI = \frac{NIR - (2 * RED - BLUE)}{NIR + (2 * RED - BLUE)}$	Corrects atmospheric effects on the land such as fog, smoke and rain.
NITROGEN		
Vegetation Index	**Formula**	**Usage**
Normalized Green Red Vegetation Index	$NGRDI = \frac{GREEN - RED}{GREEN + RED}$	Estimates nitrogen status of the plants by determining the difference between area chlorophyll content and carotenoid.
Nitrogen Nutrition Index	$NNI = L * \left(\frac{RED - GREEN}{RED + GREEN}\right) + L$	Estimates nitrogen content of vegetation with optimization parameters based on land properties.
Normalized Difference Red Edge	$NDRE = \frac{NIR - REDEDGE}{NIR + REDEDGE}$	Maps chlorophyll content which is a strong indicator of Nitrogen content. Used for vegetation with higher chlorophyll concentration.
Normalized Pigment Chlorophyll Ratio Index	$NPCRI = \frac{RED - BLUE}{RED + BLUE}$	Determines nitrogen content by calculating the difference between carotenoid and chlorophyll content of vegetation.

Table 3.4 (cont.): Vegetation Indices [27-33]

WATER		
Vegetation Index	**Formula**	**Usage**
Normalized Difference Water Index	$NDWI = \frac{GREEN - NIR}{GREEN + NIR}$	Highlights water amount in water bodies such as lakes or seas
Normalized Difference Moisture Index	$NDMI = \frac{NIR - SWIR1}{NIR + SWIR1}$	Highlights variance of moisture in vegetated areas.
Normalized Difference Snow Index	$NDSI = \frac{GREEN - SWIR1}{GREEN + SWIR1}$	Highlights snow coverage over the area.

NDVI for health estimation, negative NDWI for water stress to find plant's water stress as it is normally used for water bodies and NNI for nitrogen content were selected as the most accurate results were acquired using these indices. RGN and OCN bands are interchangeable, which means orange band can be used in place of red band and cyan band can be used for the green band.

DCM pixel information was also added to the dataset as it provides useful information about the age and species of trees. Method A.2 shows the preparation of band and index information for the dataset.

Coordinate information is also important to determine where exactly each tree is placed for precision agriculture monitoring and functionality purposes. Some parameters are important in order to calculate coordinate values for each pixel; namely x_offset, pixel_width, x_rotation, y_offset, pixel_height, y_rotation which are contained within the GeoTIFF file and can be extracted by using GDAL. For north-up images the rotation1 and rotation2 coefficients are zero, while (x_offset,y_offset) pair indicates the top left corner of the top left pixel of image. The calculated points are transformed into the desired spatial reference, which determines where each pixel is located in the real world. Coordinate transformation is handled by the library provided by Open Source Geospatial Foundation. Method A.3 explains the process.

European Petroleum Survey Group (EPSG) was an organization specialized in surveying, geodesy and cartography over oil rich areas. They published a dataset of spatial reference and coordinate system information coded from EPSG:1024 to EPSG:32767 with machine readable well known text representation in 1985. The dataset is made public in 1993 and is widely used for geospatial applications. EPSG:4326, also known as WGS (World Geodetic System) 84, is a coordinate system which is used by Global Positioning System and is based on Earth's center of mass.

For dataset creation, DataFrame from Pandas library, a software tool for data analysis and manipulation, was used which is a two dimensional labeled data structure that resembles a spreadsheet wih named columns. With each data reshaped to vectors and stacked using Numpy library, a package that handles arrays and matrices for scientific computing and high level mathematical functions, to represent columns, each row represents the full pixel band and index information. Method A.4 details the creation of the dataset. Figure 3.19 shows the resulting dataset format.

lat	lng	cyan	nir	orange	ndvi	nni	ndwi
38.4624553501	27.3678255239	162.00000000…	219.00000000…	253.00000000…	-0.0720338983	0.3483132530	-0.1496062992
38.4624558196	27.3678255263	149.00000000…	213.00000000…	255.00000000…	-0.0897435897	0.3892574257	-0.1767955801
38.4645622947	27.3674988739	147.00000000…	211.00000000…	253.00000000…	-0.0905172414	0.3917500000	-0.1787709497
38.4645618253	27.3674988715	146.00000000…	213.00000000…	255.00000000…	-0.0897435897	0.3982294264	-0.1866295265
38.4624553520	27.3678249269	141.00000000…	194.00000000…	236.00000000…	-0.0976744186	0.3793899204	-0.1582089552
38.4643773702	27.3676370552	137.00000000…	206.00000000…	254.00000000…	-0.1043478261	0.4242710997	-0.2011661808
38.4643769008	27.3676370528	135.00000000…	202.00000000…	254.00000000…	-0.1140350877	0.4306169666	-0.1988130564
38.4645437888	27.3674122032	134.00000000…	197.00000000…	247.00000000…	-0.1126126126	0.4217585302	-0.1903323263
38.4643769027	27.3676364557	133.00000000…	201.00000000…	255.00000000…	-0.1184210526	0.4387113402	-0.2035928144
38.4624548807	27.3678255215	133.00000000…	189.00000000…	231.00000000…	-0.1000000000	0.3957692308	-0.1739130435
38.4643773721	27.3676364581	131.00000000…	200.00000000…	252.00000000…	-0.1150442478	0.4401305483	-0.2084592145
38.4625931736	27.3674381313	130.00000000…	196.00000000…	253.00000000…	-0.1269487751	0.4450913838	-0.2024539877
38.4643773684	27.3676376523	130.00000000…	199.00000000…	253.00000000…	-0.1194690265	0.4450913838	-0.2097264438
38.4625927041	27.3674381290	129.00000000…	191.00000000…	245.00000000…	-0.1238532110	0.4346524064	-0.1937500000
38.4643768989	27.3676376499	128.00000000…	196.00000000…	254.00000000…	-0.1288888889	0.4533507853	-0.2098765432

Figure 3.19: Dataset structure

3.1.3. Data Preprocessing

3.1.3.1. Data Segmentation

As the data in its raw form contains both vegetation and soil noise, pixels that belong to vegetation areas need to be isolated from the others. In order to do that, the pixels need to be classified and labeled based on their properties. Machine learning can be used to find patterns between the properties of pixels and classify each pixel according to the discovered relationship. There are two ways to achieve this.

1. Supervised approach: A model is pretrained by an already labeled dataset and predictions are made based on the model.

2. Unsupervised approach: Determining relationships from the input data based on distance and clustering neighbour points without a labeled dataset.

Second option was more viable due to the lack of a training dataset. As mentioned above, each pixel was considered as a point at a specific position in space and closer points were clustered together. Table 3.5 displays some of the widely used clustering algorithms.

Table 3.5: Clustering algorithms

Algorithm	Parameters	Scalability	Optimal Usage	Metric
K-Means	Number of clusters	Very large: n_samples Large: n_clusters	Even sized, fewer clusters with flat shape	Distance between points
Affinity Propagation	Preference, damping factor	Not scalable	Uneven sized, many clusters with non-flat shape	Graph distance (nearest neighbour)
Mean-shift	Bandwidth	Not scalable	Uneven sized, many clusters with non-flat shape	Distance between points
Spectral Clustering	Number of clusters	Medium: n_samples Small: n_clusters	Even sized, fewer clusters with flat shape	Graph distance

Table 3.5 (cont.): Clustering algorithms [34]

Ward hierarchial clustering	Number of clusters or distance threshold	Large: n_samples, n_clusters	Many clusters, connectivity constraints	Distance between points
Agglomerative Clustering	Number of clusters or distance threshold, linkage type, distance	Large: n_samples, n_clusters	Many clusters, connectivity constraints and non Euclidean distances	Any pairwise distance
DBSCAN	Neighborhood size	Very large: n_samples Medium: n_clusters	Uneven sized clusters with flat shape	Distance between points
OPTICS	Minimum cluster membership	Very large: n_samples Large: n_clusters	Uneven sized clusters with flat shape and variable density	Distance between points
Gaussian Mixture Model	Many	Not scalable	Density estimation, clusters with non-flat shape	Mahalanobis distances to centers
Birch	Branching factor, threshold, optional global clusterer	Large: n_samples, n_clusters	Big data, outlier removal, data reduction	Euclidean distance between points

Figure 3.20 shows the comparison of accuracy and computation time of clustering algorithms for different data types.

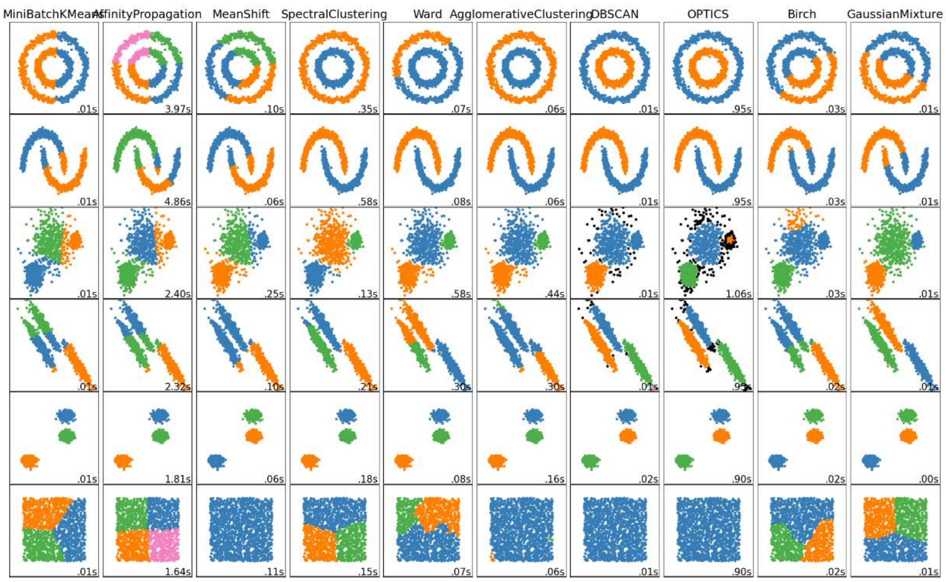


Figure 3.20: Clustering algorithms comparison [34]

K-Means algorithm was selected for its scalability and accuracy for the dataset provided. The method clusters the data points by determining specified amount of randomly selected centroids. With each iteration, as the points are added to the cluster, centroid positions change into the mean of point coordinates within the cluster and the process continues until convergence. The aim of the method is to minimize inertia, which is the distance between the points in the same cluster. Figure 3.21 shows the general process of K-Means algorithm.

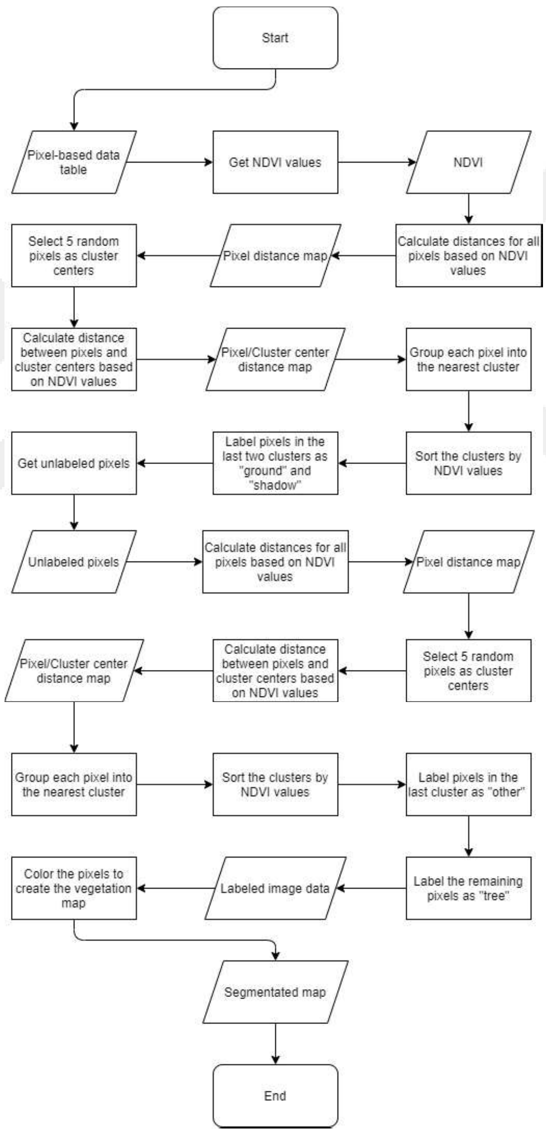


Figure 3.21: K-Means process

MiniBatchKMeans algorithm, provided by Scikit-learn which is a machine learning library, was used in order to save computation time and memory. While the principle is the same, instead of handling the whole dataset, the method selects a subset of the dataset randomly sampled in each iteration and centroids are updated per sample. Once each point in the sample is put into a cluster, new centroids are selected by taking average of the sample and the previous samples assigned to the centroid. The process continues until convergence. While the method is faster than the traditional K-Means, there is a small decrease in quality. MiniBatchKMeans algorithm uses kmeans++ for initialization, which is an algorithm that selects the centroids as far away from each other as possible in order to prevent poor clustering.

NDVI information was used in order to segmentate the data as chlorophyll content is higher in vegetation areas than soil. Method A.5 explains the process in detail.

The pixels were divided into 5 clusters which take NDVI values as input. The last two clusters with the smallest centroids were determined as ground and shadow while the remaining values were labeled as tree. The parameters have been decided as such after analyzing the data in QGIS. It can be achieved by selecting Raster => Raster Calculator from the menu and calculating NDVI manually using the software by applying the formula. Figure 3.22 shows the result after the pixels are clustered into 5 groups and the clusters with the least NDVI values are excluded from the final result in QGIS.

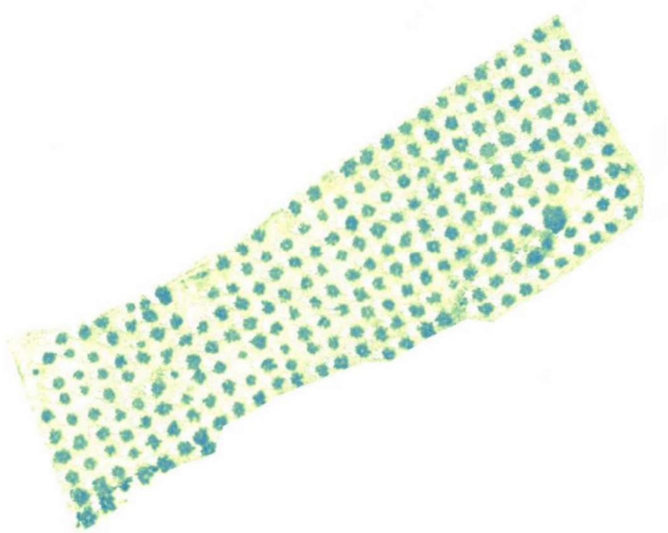


Figure 3.22: Pixel clustering in QGIS

As seen from Figure 3.22, the process is not enough to single out the trees. The same process was applied for the pixels that have been labeled as trees to further eliminate the pixels that do not satisfy the criteria. Instead of eliminating 2 clusters, only 1 cluster was eliminated and the rest of the label values remain the same. Method A.6 shows the process.

After the second clustering process, if the area is still not segmentated optimally, additional clustering processes with parameters based on the area can be added using NDVI and DCM values as other vegetation compared to the tree pixels have smaller DCM values.

Another problem to consider is no-data values. The cropping process creates pixels with minimum/negative values which do not exist in the real data. Those pixels can be clustered with the real data, which creates data inaccuracy. In order to prevent that, an additional process was added after the clustering process is completed. The no-data pixels have band values of all zeroes and DTM and DSM values smaller than -9000, so the labels of the pixels that have those values were replaced by an additional class called "outer_value". Lastly, the pixels were colored according to their label values. Figure 3.23 provides examples after the clustering process is done.

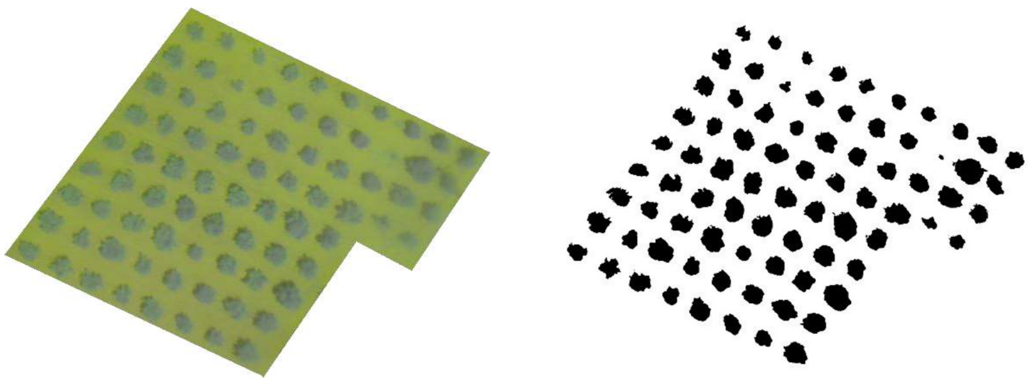


Figure 3.23: Clustered and segmentated pixels.

3.1.3.2. Tree Individualization

While each pixel has been classified into their respective groups and pixels that belong to vegetation were selected, each pixel is treated as they belong to the same tree. Precision agriculture principles require each tree to be individualized as the information comes tree-based and necessary operations are applied to trees individually. In order to achieve this, each pixel is turned into objects. The trees have a circular shape that are usually distant from each other and closer trees require processing to weaken the link in between in order to be objectified correctly.

The process to individualize the trees is given in Figure 3.24.

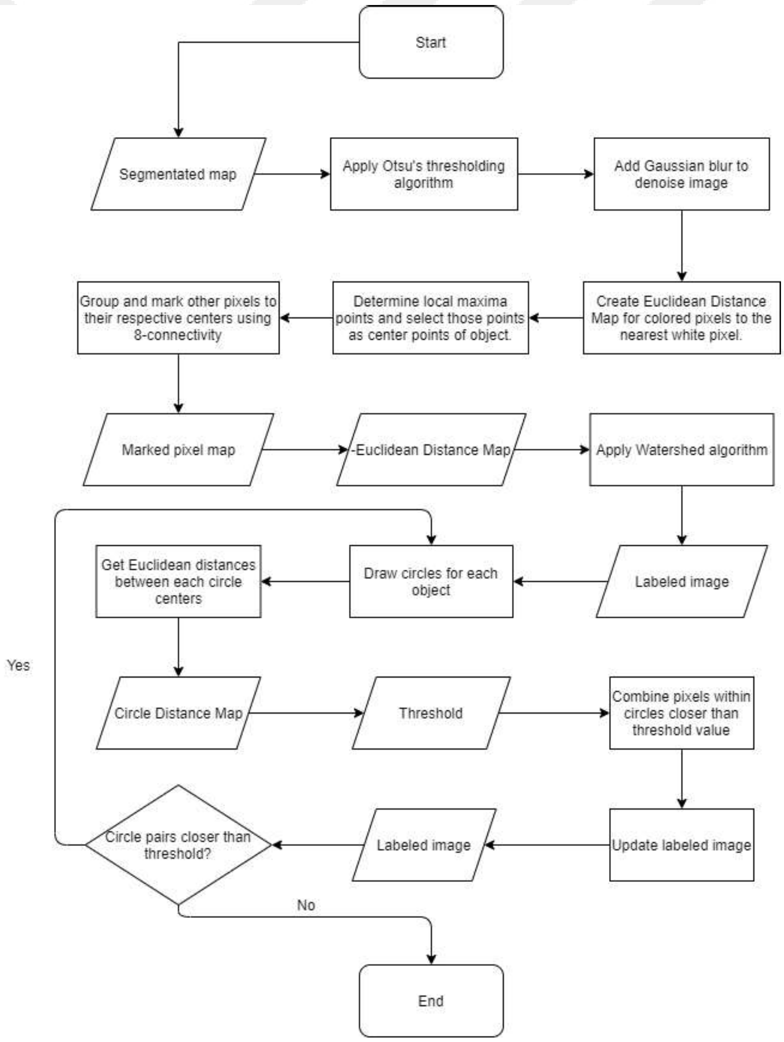


Figure 3.24: Tree individualization process

Traditional image processing techniques and contour detection can be used to turn the pixels into objects based on their boundary points. Contour detection, combined with thresholding to segmentate the image's background from the foreground, detects the group of pixels where a sharp color change occurs within the image. The pixels that the mentioned color change occurs is called an edge point. The collection of edge points which are connected and eventually return to the original point is called a contour. And the other pixels that is inside the boundary points is called an object. Thresholding can either be manual, optimal for the scenario that the image histogram, graphical representation of frequency of values, has only one peak or can be found programmatically. For manual thresholding, the peak value can be given as a parameter to turn the image into black and white, which is called binarization. However, for images with multiple peaks, which are called bimodal images, this algorithm does not provide a good solution.

Otsu's thresholding algorithm classifies the bimodal image into foreground and background and finds the optimal threshold by minimizing variance within the same class by calculating the cumulative sum of weighted variances of both classes. Method A.7 explains the process in more detail.

The algorithm is provided by OpenCV, a library for image processing and complex computer vision problems. Before starting the process, the image was turned into grayscale to make the calculations easier and Gaussian blur was applied on the grayscale image for the image can contain additional pixel noise and the outlier points can be objectified inaccurately. Gaussian filter algorithm is described in Method A.8.

An example result image after traditional contour detection algorithm was applied on a tree area is given in Figure 3.25. The image was prepared by applying Gaussian Blur algorithm and Otsu's thresholding algorithm beforehand.

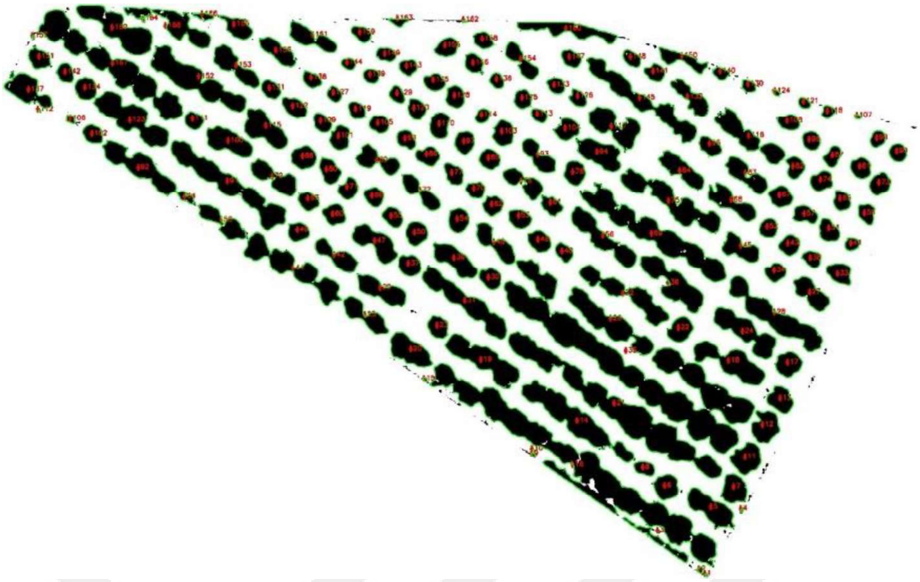
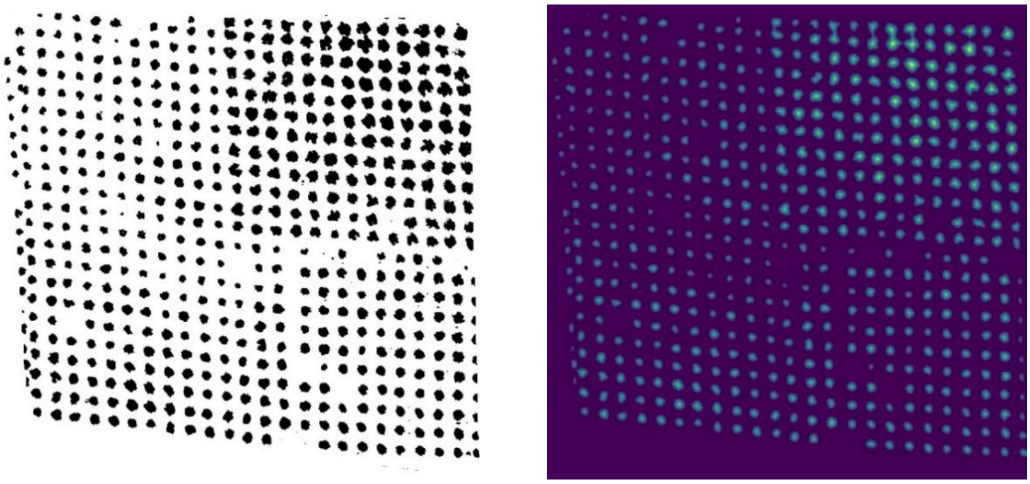


Figure 3.25: Result image after traditional contour detection algorithm is applied

As seen in Figure 3.25, while the single trees can be objectified correctly by traditional methods, overlapping trees are determined as the same object since the edge pixels are connected. Relying on pixel position is not enough for a correct objectification as trees are not two dimensional objects. Watershed algorithm considers the image as valleys and peaks and can be used as an alternative to the traditional methods in order to correctly objectify the overlapping data.

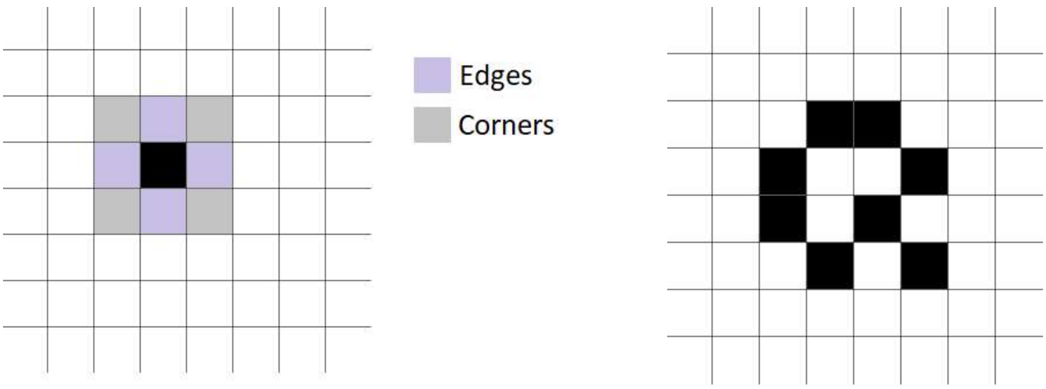
The algorithm uses Euclidean Distance Transform to find the distance of each foreground pixel to the nearest background pixel. Figure 3.26 shows the distance map of a segmentated image.



$$Euclidean\ Distance = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Figure 3.26: Distance map of the segmentated image

Then local maxima, the pixels that are the most distant from the background pixels, are selected as the central point of the object. The nearby pixels are marked using 8-connectivity, which means pixels that are connected by edges or corners. Figure 3.27 shows an 8-connected pixel distribution.



$$connected\_pxls = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * image\_matrix$$

Figure 3.27: 8-connectivity rule and 8-connected pixel distribution example

The marked pixels and the local minima points are fed into Watershed algorithm, which label the pixels into their respective clusters by "flooding" the valleys, marked pixels, until different markers are encountered. As Watershed algorithm considers the markers as local minima points, negative value of the Euclidean Distance Map is fed into the method. Once the labeling process is done, the result array is fed into the contouring process and each tree is objectified and marked with a circle attached with an individual ID of the tree. The labels are ranged from 0, which indicates background, to the count of objects found. Method A.9 summarizes the process.

Figure 3.28 shows the result image once the process is done.

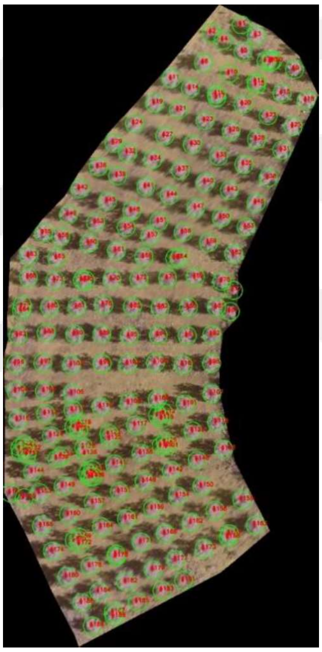


Figure 3.28: Tree individualization result

As seen in Figure 3.28, some trees were marked multiple times for having a non-circular shape. This situation causes data inaccuracy since the information for the same tree gets divided into multiple parts as if they belong to different trees. In order to prevent that, this study proposes a Circle Reduction algorithm that handles incorrect marking of individual trees. The circles that are closer than the threshold value are merged with a new central point of mean (x,y) pairs, while the new diameter is the mean of two circles' diameters. Method A.10 shows the extraction and summation of (x,y,r) points from the circles.

The algorithm requires the calculation of Euclidean distances between the circles in order to create a distance map that contains the distances of all circle pairs. A threshold value was determined based on the tree alignment and sizes of the area. The pairs of circles that have smaller distance than the threshold value were selected for the merging process. Method A.11 explains the process.

The selected pairs' (x,y,r) values were added together using the function in Method A.10. The important matter to consider is that if the circles that are close are in sequence. Suppose that there are circle pairs of (0,1,20.22) and (1,2,10.10). Both of these pairs are circles that need to be merged if the threshold value is considered to be bigger than 20. However, they are in sequence, meaning that (0,1) and (1,2) pairs in fact belong to the same tree. So, the new circle pair that needs to be considered is (0,1,2). In order to achieve that, indices of the circle pairs that contain one of the circles inside the current circle pair are found and circles that are members of those circle pairs are considered sequential. The iteration continues until all sequential circles are gathered and the merging process is applied to the circles afterwards. All close circles are merged until none remains and the label values found in Watershed algorithm are changed according to the new array of circles. Method A.12 explains the method in more detail.

Once every circle pair in the first iteration is merged and the new circles are calculated, the Euclidean distance map of the new circle array is created, merging process starts for the new batch of circle pairs and labels are reorganized after transformation until the circle array does not contain pairs of close circles. The result labels were added to the dataset as a new column and each pixel that belong to the "tree" class were individualized. The pixels were grouped by their individual tree numbers and mean values of vegetation indices, band information, location, height and object sizes for each group were calculated to provide an overall status for the trees. Method A.13 shows the process.

As Method A.13 shows, another way to eliminate the unwanted trees is by their sizes. Since the circles have their radius values, a threshold value was determined to eliminate very young trees that have just been planted or trees that are not fully in frame due to the cropping process. In both scenarios, the overall status of the trees can be inaccurate, creating noise in the data. The result image can be seen in Figure 3.29.

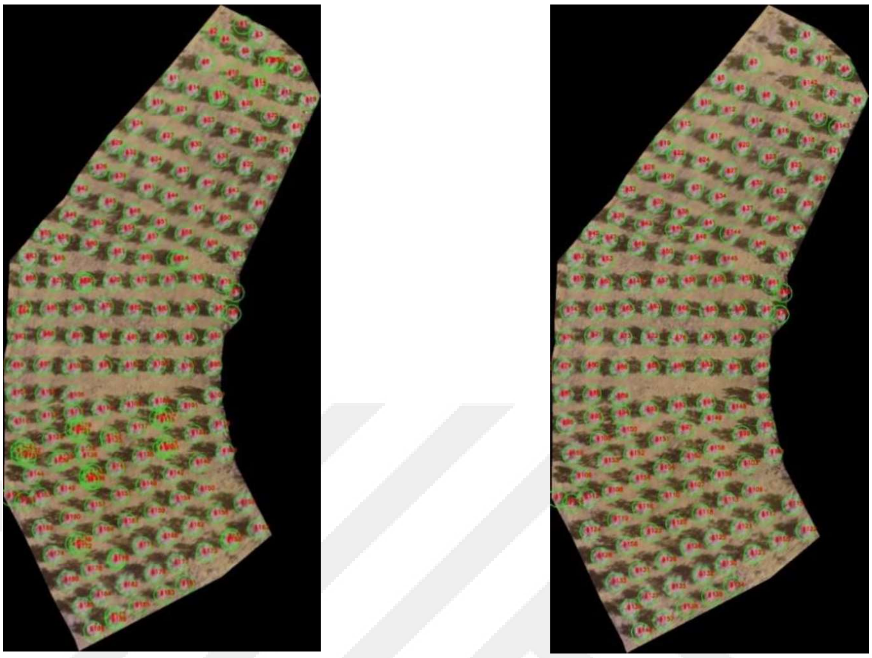


Figure 3.29: Circle reduction method results

The final tree object dataset after the preprocessing can be seen in Figure 3.30.

orange	cyan	nir	lat	lng	ndvi	nni	ndwi	dcm	treesize	idasize	treenum
179.04589761…	55.0917306823	150.90362145…	38.4647355314	27.3670027982	-0.1426128254	0.6491823316	-0.4719028874	0.2217094781	15491	0.0123474700	1
184.39917012…	58.8522821577	152.70290456…	38.4648218427	27.3670131772	-0.1329017865	0.6395461817	-0.4546679473	0.3347432877	2410	0.0019209478	2
185.09830633…	59.5452871870	156.11561119…	38.4646418577	27.3670248913	-0.1285248146	0.6320324076	-0.4527819075	0.2896886722	2716	0.0021648524	3
181.43997205…	57.7011263424	151.01423207…	38.4645214224	27.3670399700	-0.1419901953	0.6379672837	-0.4545074527	0.2640035395	11453	0.0091288860	4
166.07870370…	48.2962962963	134.18981481…	38.4642313605	27.3670409903	-0.1356052361	0.6725038010	-0.4832524335	-0.0001051161	216	0.0001721679	5
180.17603125…	57.2378490176	151.10272319…	38.4644954235	27.3670422986	-0.1452813273	0.6388876656	-0.4584523598	0.2006322963	8703	0.0069369331	6
179.60076757…	57.1913320228	149.75128709…	38.4643847944	27.3670538874	-0.1338882597	0.6364919489	-0.4531648917	0.1503794269	10683	0.0085151392	7
182.01030927…	56.5549571729	155.29255782…	38.4645151654	27.3670828222	-0.1246814311	0.6481492284	-0.4757172930	0.2606724981	19497	0.0155405475	8
196.66850828…	70.2762430939	158.28176795…	38.4647283658	27.3670935154	-0.1157143807	0.5921422923	-0.3878548958	0.0025875871	181	0.0001442704	9
180.93494240…	56.8832693363	150.47043335…	38.4641674794	27.3670892648	-0.1664413749	0.6446509285	-0.4620672437	0.3524115683	9115	0.0072653275	10
177.35811100…	52.2526751320	149.77785704…	38.4647500427	27.3671013145	-0.1321327157	0.6666106285	-0.4931501221	0.3250971242	7009	0.0055866901	11
179.71692984…	55.6743463629	149.98317369…	38.4640847323	27.3671023675	-0.1434029862	0.6462612109	-0.4647550976	0.1833473579	7726	0.0061581920	12
180.65596627…	55.9637916126	152.23638132…	38.4640046744	27.3671069356	-0.1350000723	0.6464275755	-0.4688203633	0.2440553097	9252	0.0073745266	13
179.45374811…	56.4266136769	151.36151843…	38.4646630683	27.3671198763	-0.1355596194	0.6445835821	-0.4672994966	0.2890770959	7297	0.0058162474	14
179.34368822…	52.9350465970	151.47783112…	38.4639184477	27.3671238047	-0.1113062240	0.6637743661	-0.4897992531	0.1456917496	7082	0.0056448765	15
182.01312364…	55.4914316703	153.53915401…	38.4645798773	27.3671349705	-0.1302367740	0.6528530962	-0.4766814423	0.2311940688	9220	0.0073490203	16

Figure 3.30: Finalized tree object dataset

3.1.4. Yield Prediction

Yield prediction is one of the most important applications of Precision Agriculture which helps the farmers to determine the low-yield areas to tend to and to make timely import and export decisions. Yield is affected by several parameters, such as topography of the area, soil properties and nutrition, plant properties and climate information. This study has access to plant properties, band information and vegetation indices, and topography of the area, DTM and DCM.

3.1.4.1. Data Gathering

Evaluating all trees in an area is not optimal as it is time and resource consuming. For that purpose, several groups of trees which have low and high NDVI, chlorophyll content, were selected and those trees' monthly yields were evaluated on site. The reason why those specific groups were selected was to get an accurate overall yield for the area as selecting high or low values only can lead to a wrong calculation. Figure 3.31 shows an example chlorophyll content map and tree group selection.

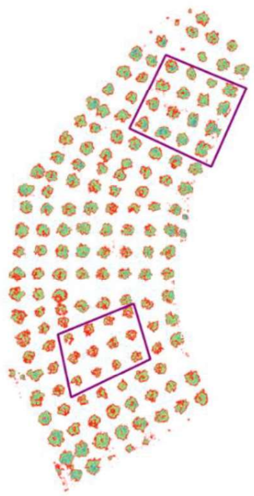


Figure 3.31: Tree selection for yield prediction

Sum of calculated yield values was divided by the number of trees inside the groups to find the average yield value for the area. Total yield was determined by multiplying average yield with the total number of trees. Total yield was apportioned to the trees based on their NDVI values to organically increase the data count.

3.1.4.2. Model Selection

Since the predicted variable is continuous, a regression model was used in order to create a formula that best represents the dataset. There are several machine learning models to consider which are given in Table 3.6.

Table 3.6: Regression techniques

Model Name	**Usage**
Linear Regression	No outliers, linear relationship between features and target variable.
Polynomial Regression	Non-linear, Nth degree polynomial relationship between features and target variable.
Stepwise Regression	Used for high dimensional data, determines the significance of features and combinations of said variables that maximizes prediction accuracy using statistics.
Lasso Regression	It is used for multicolinearity problem where there is a high correlation between features. It aims to regularize the data by adding a penalty function to the error function that is the absolute value of the slope of the best fitting line multiplied by a parameter. Provides feature selection. L1 regularization.
Ridge Regression	Works similarly as Lasso Regression, but instead of absolute values, sum of squares is used for the penalty function. Optimal slope does not become zero. More stable. L2 regularization.
ElasticNet Regression	Combines Lasso and Ridge Regression's stability and feature selection capabilities. As an advantage over Lasso Regression, while Lasso Regression selects only one feature, ElasticNet can select groups of features.
Support Vector Regression	Works for both linear and non-linear data, finds the line/hyperplane that best describes the data using kernel functions.
Decision Tree Regression	Divides dataset into smaller subsets and transforms the subsets into a tree structure to make predictions based on basic rules derived from the characteristics of data.
Artificial Neural Network	A weighted, nodal network that takes features and target variable to find an optimal formula for the dataset. Optimization of the model is by "punishment", tuning the weight parameters based on error values until no more improvement, which is called back propagation.

Decision Tree Regression was used as the model fits the dataset well. Vegetation indices and rescaled pixel sizes of the trees based on total size were selected as features to predict average yield value. Decision Tree regression models act as binary trees, in which values smaller than the root value are stacked to the left and bigger values are stacked to the right. These models split the data by features until minimum sample count, minimum number of instances inside the split group, and/or maximum tree depth, longest path from root to leaf, are reached and aim to minimize the error by selecting the optimal feature for the split. The error function was selected as mean squared error which is given in Equation 3.1.

$$\mathrm{MSE}=\frac{1}{N}\sum_{i=1}^{n}(\mathrm{ypred}_i-\mathrm{ytrue}_i)^2 \tag{3.1}$$

Dataset was consisted of 2912 rows and was split 80%/20% into training and validation datasets in order to train the model and test for accuracy assessment. Minimum sample size was determined as 1% of the length of the training dataset while the maximum tree depth was selected as 10. Chlorophyll content, nitrogen nutrition and water stress, as well as the rescaled pixel sizes of the plants based on total size were selected for the learning process. An example tree structure for prediction is given in Figure 3.32.

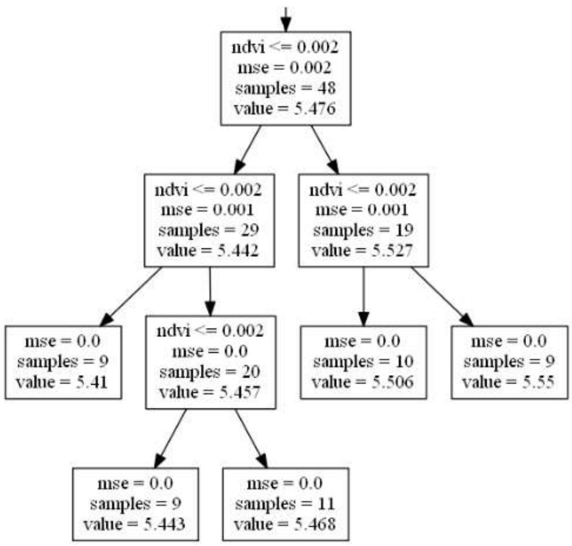


Figure 3.32: Decision Tree Regressor prediction tree structure

3.1.5. Disease Prediction

Another important application of Precision Agriculture is disease prediction as most of the diseases spread quickly and decrease the quality of the harvest considerably. Early detection of diseased areas also help with the amount of spraying and decision of medicine as each disease requires different procedures.

3.1.5.1. Data Gathering

The data gathering process works similarly with yield prediction. Areas that have low chlorophyll content were selected and those trees were examined on site for diseases. This study focuses on Palpita unionalis, which is a type of larva that feeds on fresh olive tree leaves and products, causing the yield to decrease considerably for the following years if it is not treated [37].

Trees that contain the Palpita unionalis were selected to create the training dataset, with the addition of healthy trees which have high chlorophyll content.

3.1.5.2. Model Selection

Since the target variable is categorical and a labeled training dataset exists, a classification model was used. Applicable model options are given in Table 3.7.

Table 3.7: Classification algorithms

Model Name	**Usage**
Logistic Regression	Classifies binary target variable. Uses logistic function to separate classes. Works for both independent and correlated features.
Naive Bayes	Assumes independency of features. Uses Bayes theorem to calculate the probability of each state considering the frequency of variables in selected features and declares the highest probability state as the prediction.
Support Vector Machine	Finds closest two points that have different class labels then finds a line that’s furthest from both points that optimally separates those two classes. Works for both linear and non-linear data.

Table 3.7 (cont.): Classification algorithms

K-Nearest Neighbors	KNN is a lazy learning algorithm, meaning that it stores the training data and does not create a model until testing process. Calculating distance between each instance and the test variable, K neighbors are selected closest to the test variable and it is labeled based on the class majority of neighbours belong to.
Artificial Neural Network	A weighted, nodal network that takes features and target variable to find an optimal formula for the dataset for predictions. It uses sigmoid function for binary target variable and softmax for multiclass target variable. Optimization of the model is by "punishment", tuning the weight parameters based on error values until no more improvement occurs.
Decision Tree/Random Forest	The algorithm transforms the dataset into a tree structure and makes predictions based on basic rules derived from dataset characteristics. Random Forest's principle is to create multiple Decision Trees from randomly chosen training data subsets and make predictions based on majority vote.

Random Forest algorithm was selected for the dataset. Number of Decision Trees to be generated was determined as 10 and split criterion was selected as entropy. Entropy is the measure of disorder used to determine if the subset is useful for prediction. High entropy means the subset contains similar amount of variables from both of the classes and leads to poor performance. Entropy formula is given in Equation 3.2.

$$\text{Entropy} = \sum_{i=1}^{n} -p_i \log_2 p_i \;\; \text{where } p_i = \frac{\#_{\text{class}_{\text{elements}}}}{\#_{\text{class}_{\text{total}}}} \text{ and } n = \#_{\text{classes}} \qquad \textbf{(3.2)}$$

As the tree model is constructed, entropy change from previous node to the current determines if the subset is suitable to become a split. The difference between two values is called Information Gain. Information Gain is ranged between 0 and 1, where 0 means there is no change and 1 means the subset is the optimal split. Iteration continues until all optimal splits are selected and the tree is constructed. The process was done 10 times to create the Decision Trees required for the majority vote.

Dataset was consisted of 255 rows and was split 80%/20% into training and test datasets. Chlorophyll content, nitrogen nutrition and water stress, as well as the rescaled pixel sizes of the plants based on total size were selected for the learning process.

CHAPTER 4

RESULTS

4.1. Crop Segmentation

The crop segmentation results was determined by comparing the predicted data and the ground truth image. The ground truth image was obtained by manual segmentation of pixels by NDVI values using the QGIS tool.

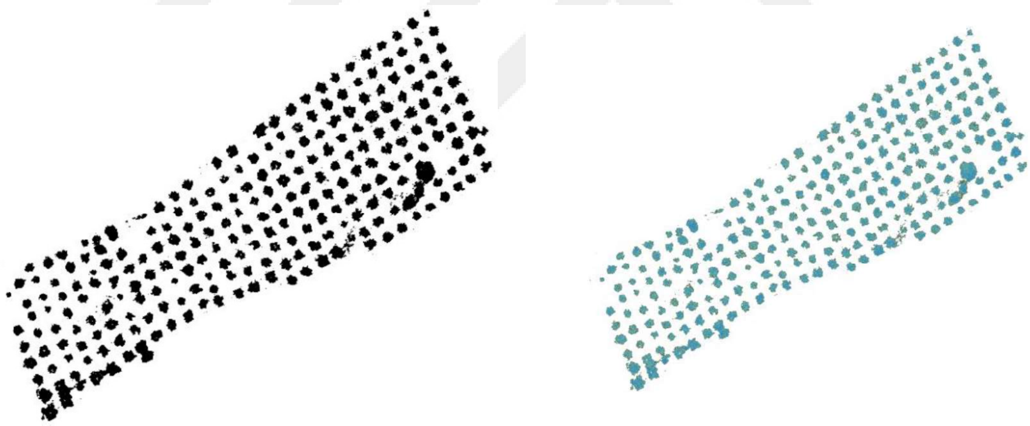


Figure 4.1: Predicted data and the ground truth image

The evaluation metric was selected as accuracy score and its formula is given in Equation 4.1.

$$\text{Accuracy Score} = \frac{\text{\# of matching black pixels}}{\text{\# of all black pixels}} \qquad \textbf{(4.1)}$$

In average, between 85%-95% accuracy score was obtained in the experiments, worsening for areas that have high weed occurance and mapping errors. For comparative analysis, summary of the methods in the literature review and the method used in this study is given in Table 4.1.

Table 4.1: Crop segmentation comparative analysis

Article Author	Crop Name	Method Name	Evaluation Metric	Results
Bai et al [1]	Rice plant	Color segmentation	ATRWG, other	87.2%, 96%
Torres-Sanchez, Lopez-Granados & Pena [2]	Wheat, sunflower and maize	Object Based Image Analysis tool, Otsu's thresholding algorithm	Classification error	Between 0% and 10%
Huang, Li & Chen [3]	Osmanthus and Podocarpus	Watershed algorithm	F-score	98.2% (Osmanthus), 93.1% (Podocarpus)
Lottes et al [4]	Sugar beet	RFC	Recall, precision	78%, 90%
Mancini et al [5]	Vineyard	DSM convolution/slicing, AdaBoost	Accuracy score	96% (DSM convolution), 96% (AdaBoost)
Onyango and Marchant [6]	Cauliflower plant	Texture analysis	Classification rate	Between 82% and 96%
Dyson et al [7]	Vineyard	CNN	Improvement rate	65%
This study	**Olive trees**	**MiniBatchKMeans and Watershed algorithm**	**Accuracy rate**	**Between 85% and 95%**

4.2. Yield Prediction

The predicted data and real data comparison was given in Figure 4.2.

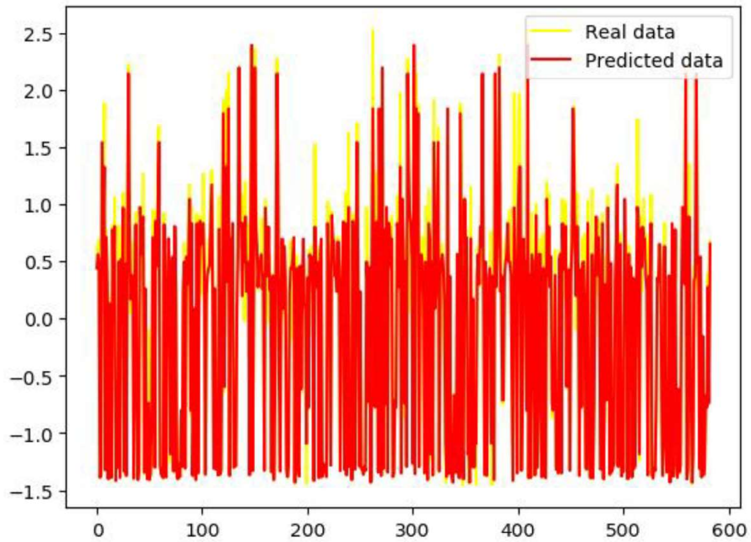


Figure 4.2: Yield prediction comparison of real and predicted data

The evaluation metrics were chosen as RMSE and R-squared (R2) score to determine the success of the model. While a smaller value is optimal for RMSE, a higher value is better for R2 score, which is between 0 and 1. Usage for each metric and the results are shown in Table 4.2.

Table 4.2: Yield prediction metrics and results

Name	**Formula**	**Usage**	**Result**
Mean Squared Error	$RMSE=\sqrt{\frac{1}{N}\sum_{i=1}^{n}(ypred_i\text{-}ytrue_i)^2}$	Increases the penalty for larger errors.	0.22
R-Squared Score	$R^2=1-\frac{\sum_{i=1}^{n}(ytrue_i\text{-}ypred_i)^2}{\sum_{i=1}^{n}(ytrue_i\text{-}ymean)^2}$	Shows the correlation between variables.	0.95

Table 4.2 proves that the Decision Tree Regressor model was a good choice for this dataset and the yield amount for trees can be successfully predicted using tree sizes and vegetation indices. For larger yield values, the method’s accuracy rate decreases due to lack of training data. There is also human bias to consider as the field team reports their own predictions instead of actual calculated data.

Methods in literature and this study is compared in Table 4.3.

Table 4.3: Yield prediction comparative analysis

Article Author	Crop Name	Method Name	Evaluation Metric	Results
Pantazi et al[8]	Wheat	SKN	Cross validation	91% for low yield class, 70% for average yield class and 83% for high yield class
Ashapure et al [9]	Tomato	ANN	R2 score	Between 0.78 and 0.89
Panda et al [10]	Corn	BPNN	Accuracy score, R2 score	93%, 0.72
You et al [11]	Soybean	CNN, LSTM, Gaussian	RMSE	(CNN and Gaussian average) 5.55
Gonzalez-Sanchez, Frausto-Solis& Ojeda-Bustamante [12]	Multiple crops	MLR, ANN, KNN, SVR and M5-Prime	RMSE, RRSE, MAE, R	(M5-Prime) 5.14, 79.46%, 18.12%, 0.41
Nari and Yang-Won [13]	Corn	RF, Deep Learning, SVM, ERT	R score	(Deep Learning) 0.776, 0.796 and 0.800 for 3 different datasets
Schwalbert et al [14]	Soybean	RF, MVR, LSTM	MAE	(LSTM) 0.42, 0.25 and 0.24 for 3 different datasets.
Maimaitijiant et al [15]	Soybean	PLSR, RF, SVR, DNN-F1, DNN-F2	R2, RMSE	(DNN-F2) 0.72, 0.15
This study	**Olive trees**	**DTR**	**R2, RMSE**	**0.95, 0.22**

4.3. Disease Prediction

Confusion matrix was used to describe the performance. Also known as error matrix, it compares actual class and predicted class counts and displays them in an NxN matrix where N is the total number of classes. The confusion matrix is important to understand the evaluation metrics. The structure of a confusion matrix is given in Table 4.4.

Table 4.4: Confusion matrix structure

Predicted / **Actual**	**0**	**1**
0	True Positive	False Negative
1	False Positive	True Negative

Confusion matrix after prediction process is given in Table 4.5.

Table 4.5: Confusion matrix after prediction process

Predicted / Actual	**Palpita unionalis**	**Healthy**
Palpita unionalis	32	1
Healthy	0	18

To evaluate the performance of the model, several evaluation metrics were used, namely Accuracy Score, Precision, Recall and F1 score. Descriptions and results for each metric are given in Table 4.6.

Table 4.6: Classification evaluation metrics

Metric Name	**Formula**	**Usage**	**Results**
Accuracy Score	$\frac{TP+TN}{TP+FP+FN+TN}$	Ratio of correctly predicted values to total values.	0.98
Precision	$\frac{TP}{TP+FP}$	Ratio of correctly predicted positive values to all predicted positive values.	1.0
Recall (Sensitivity)	$\frac{TP}{TP+FN}$	Ratio of correctly predicted positive values to actual count of positive values.	0.97
F1 Score	$2*\frac{Precision*Recall}{Precision+Recall}$	Takes both false positive and negatives into account. Carries more useful information than accuracy.	0.98

The results show that Random Forest classifier algorithm works well for the features provided and Palpita unionalis can be detected by vegetation indices and the size of the trees accurately. As Palpita unionalis is a leaf based disease, providing leaf based data to the dataset in addition to the tree based data would increase the accuracy rate.

Method comparison is given in Table 4.7.

Table 4.7: Disease prediction comparative analysis

Article Author	Crop Name	Disease Name	Method Name	Evaluation Metric	Results
Ferentinos [16]	Multiple crops	Multiple diseases	AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat and VGG	Accuracy rate	(VGG) 99.53%
Khamparia et al [17]	Multiple crops	Multiple diseases	DCEN	Accuracy rate	100%
Pantazi et al [18]	Milk thistle weed	Microbotryum silybum	SKN, CP-ANN, XY-F	Accuracy rate	(XY-F) 95.16%
Chung et al [19]	Rice seedlings	Foolish seedling disease	SVM	Accuracy rate	87.9%
Duarte-Carvajalino et al [20]	Potato	Late blight	ANN, RF, SVR and CNN	MAE,RMSE, R2	11.2,15.09,0.74
Moshou et al [21]	Wheat	Yellow rust disease	MLP, QDA, SOM	Accuracy score	(MLP) 95%-99%
Ebrahimi et al [22]	Strawberry	Thrips insect	SVM	MPE	2.25%
This study	**Olive trees**	**Palpita unionalis**	**RFC**	**Accuracy score**	**98%**

CHAPTER 5

FUTURE WORKS AND DISCUSSION

The results show that machine learning algorithms can be useful to predict future behaviour for Precision Agriculture problems and produce an estimation for yield and disease information of trees that would require manpower and resources to manually examine. Vegetation indices carry vital information and are useful for characterization of the trees.

While the discussed models work very well with the datasets provided, it is assumed that the weather conditions stay the same in all instances. Since weather conditions affect the light reflectance values, the vegetation indices also change. Thus, the assumption may lead to inaccurate predictions later on. It is helpful to gather data from the target area in several weather conditions to teach the model the behaviour of the light during different states for the predictions to be a lot more reliable by adding those different states as a categorical feature to the dataset.

Another way to improve the reliability of the model is more physical information such as soil type and nutrients. Even chemical information that might be affecting the soil or the crops at that moment since farmers tend to spray the area from time to time. Adding these information will improve the characteristics of the observed instances.

While gathering data, one of the problems that has been encountered was the mapping process. Since different groups were responsible for taking photographs of the area, their camera specifications often didn't match. Due to that, the created maps were not compatible for training dataset creation. While conducting an experiment, some parameters are kept fixed for the consistency of the experiment. For this study, those parameters were camera specifications and the route the drone takes. Changing one of those parameters will change the photograph's quality and thus, it's usefulness as training data.

The photogrammetry software that was used to create orthomosaic, DSM and DTM files automatically detects GCPs which are used to align photographs and determine the quality of the created files. The detected GCPs are not always optimal and can cause blurryness or distortions in the end result. An example can be seen in Figure 5.1.

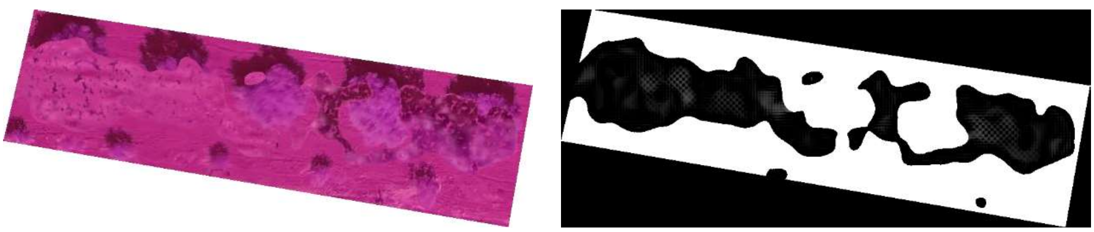


Figure 5.1: Orthomosaic and CHM distorted data due to incorrect merging

GCPs can be defined manually to solve the alignment problem. Another issue to consider is the program's compatibility with the camera itself. Pix4D can be used for all cameras that have geospatial capabilities, but performs better in some cases. Parrot Sequoia and MicaSense Altum cameras are observed to produce better results. Comparison of specifications for both cameras is given in Table 5.1.

Table 5.1: Comparison of two cameras [38][39]

Topics	**MicaSense Altum**	**Parrot Sequoia**
Sensor Resolution (megapixel)	3.2	1.2
Capture Rate (per second)	1	1
Image Dimensions (pixels)	2064x1544	1280x960
Lens Focal Length (mm)	8	3.98
Bands	Blue, green, red, red edge, NIR	Green, red, rededge, NIR

Having more bands leads to calculation of more vegetation indices that can further characterize the dataset. Mapir lacks Red Edge and SWIR bands, which contain wavelengths that capture nitrogen and water information more accurately.

Palpita unionalis is a leaf and product based disease. Thus, leaf and product-based data can be gathered to reinforce the tree-based dataset and provide specialization.

Weed grown in the area due to neglection affect the dataset if they were not handled beforehand. As their vegetation index values are similar to the actual data, it is difficult to separate them by clustering algorithms or manual thresholding. Thus, occurance of weed is considered noise and an effective noise removal algorithm is needed to handle the problem. Figure 5.2 shows the effects of occurance of weed on the dataset.

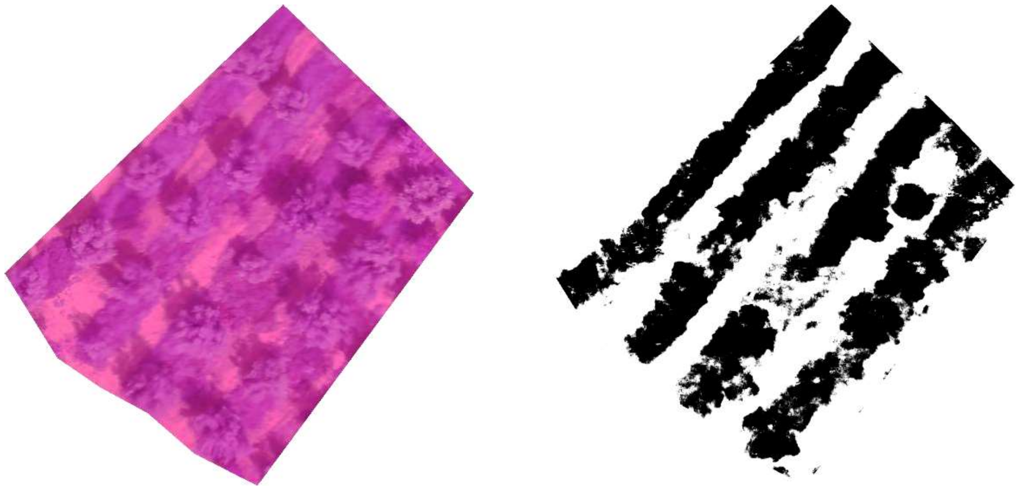


Figure 5.2: Segmentation of a neglected area

To sum up, while the base model acts sufficiently, there is still room to improve the model further by considering the issues stated in this section. Real data is chaotic and machine learning is rule based, thus it is important to prepare the model for each possible combination of events that can simulate real data.

CHAPTER 6

CONCLUSION

This study's aim was to explore the usage of machine learning on Precision Agriculture applications. The primary focus was on three subjects, namely crop segmentation, illness detection and yield prediction. Crop segmentation process was achieved using MiniBatchKMeans algorithm. For disease detection, this study focused on Palpita unionalis, a type of tree worm that feeds on lively olive tree leaves and fruits. Random Forest classifier algorithm was used to determine the trees that contain the tree worms. Lastly, for yield prediction, Decision Tree Regression algorithm was used.

Examining the comparative analysis tables in Chapter 4 for each application, this study performs well in all cases despite using basic vegetation index and band information, which shows that vegetation indices and size of the trees are highly correlated with the topics of research. Machine learning algorithms work well with Precision Agriculture applications and are helpful to contextualize the information gained by monitoring process.

With future work discussed in Chapter 5, it is highly possible to obtain a generalized model that has high predictive capabilities. This opportunity helps the farmers to reduce the cost and manpower needed to manually examine the areas for each operation and prevent wrong irrigation and spraying for areas that require care.

REFERENCES

[1] Bai, X. D., Cao, Z. G., Wang, Y., Yu, Z. H., Zhang, X. F., & Li, C. N. Crop segmentation from images by morphology modeling in the CIE L*a*b* color space. Computers and Electronics in Agriculture, *99,* 21–34. 2013

[2] Torres-Sánchez, J., López-Granados, F., & Peña, J. M. An automatic object-based method for optimal thresholding in UAV images: Application for vegetation detection in herbaceous crops. Computers and Electronics in Agriculture, *114,* 43–52. 2015

[3] Li, Y., Huang, Z., Cao, Z., Lu, H., Wang, H., & Zhang, S. Performance Evaluation of Crop Segmentation Algorithms. IEEE Access, *8,* 36210–36225. 2020

[4] Lottes, P., Khanna, R., Pfeifer, J., Siegwart, R., & Stachniss, C. UAV-based crop and weed classification for smart farming. 2017 IEEE International Conference on Robotics and Automation (ICRA), 3024-3031. 2017

[5] Mancini, A., Dyson, J., Frontoni, E., & Zingaretti, P. Soil / crop segmentation from remotely sensed data acquired by Unmanned Aerial System. 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 1410-1417. 2017

[6] Onyango, C. M., & Marchant, J. A. Segmentation of row crop plants from weeds using colour and morphology. Computers and Electronics in Agriculture, *39*(3), 141–155. 2013

[7] Dyson, J., Mancini, A., Frontoni, E., & Zingaretti, P. Deep Learning for Soil and Crop Segmentation from Remotely Sensed Data. Remote Sensing, *11*(16), 1859. 2019

[8] Pantazi, X. E., Moshou, D., Alexandridis, T., Whetton, R. L., & Mouazen, A. M. Wheat yield prediction using machine learning and advanced sensing techniques. Computers and Electronics in Agriculture, *121,* 57–65. 2016

[9] Ashapure, A., Oh, S., Marconi, T.G., Chang, A., Jung, J., Landivar, J., et al. Unmanned aerial system based tomato yield estimation using machine learning. In J. A. Thomasson, M. McKee, & R. J. Moorhead (Eds.). *SPIE DEFENSE + COMMERCIAL SENSING Proceedings Vol. 11008:Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV* (171-180). Washington: SPIE. 2019

[10] Panda, S. S., Ames, D. P., & Panigrahi, S. Application of Vegetation Indices for Agricultural Crop Yield Prediction Using Neural Network Techniques. Remote Sensing, *2*(3)*,* 673–696. 2010

[11] You, J., Li, X., Low, M., Lobell, D. & Ermon, S. Deep Gaussian Process for Crop Yield Prediction Based on Remote Sensing Data. AAAI, 4559-4566. 2017

[12] Gonzalez-Sanchez, A., Frausto-Solis, J., & Ojeda-Bustamante, W. Predictive ability of machine learning methods for massive crop yield prediction. Spanish Journal of Agricultural Research, *12*(2), 313-328. 2014

[13] Nari, K. & Yang-Won, L. Machine Learning Approaches to Corn Yield Estimation Using Satellite Images and Climate Data: A Case of Iowa State. Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography, *34*(4), 383–390. 2016

[14] Schwalbert, R. A., Amado, T., Corassa, G., Pott, L. P., Prasad, P. V. V., & Ciampitti, I. A. Satellite-based soybean yield forecast: Integrating machine learning and weather data for improving crop yield prediction in southern Brazil. Agricultural and Forest Meteorology, *284*, 107886. 2020

[15] Maimaitijiang, M., Sagan, V., Sidike, P., Hartling, S., Esposito, F., & Fritschi, F. B. Soybean yield prediction from UAV using multimodal data fusion and deep learning. Remote Sensing of Environment, *237,* 111599. 2020

[16] Ferentinos, K. P. Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, *145,* 311–318. 2018

[17] Khamparia, A., Saini, G., Gupta, D., Khanna, A., Tiwari, S., & de Albuquerque, V. H. C. Seasonal Crops Disease Prediction and Classification Using Deep Convolutional Encoder Network. Circuits, Systems, and Signal Processing. *39,* 818-836. 2019

[18] Pantazi, X. E., Tamouridou, A. A., Alexandridis, T. K., Lagopodi, A. L., Kontouris, G., & Moshou, D. Detection of Silybum marianum infection with Microbotryum silybum using VNIR field spectroscopy. Computers and Electronics in Agriculture, *137,* 130–137. 2017

[19] Chung, C. L., Huang, K.J., Chen, S. Y., Lai, M. H., Chen, Y. C., & Kuo, Y. F. Detecting Bakanae disease in rice seedlings by machine vision. Computers and Electronics in Agriculture, *121,* 404–411. 2016

[20] Duarte-Carvajalino, J., Alzate, D., Ramirez, A., Santa-Sepulveda, J., Fajardo-Rojas, A., & Soto-Suárez, M. Evaluating Late Blight Severity in Potato Crops Using Unmanned Aerial Vehicles and Machine Learning Algorithms. Remote Sensing, *10*(10), 1513. 2018

[21] Moshou, D., Bravo, C., West, J., Wahlen, S., McCartney, A., & Ramon, H. Automatic detection of "yellow rust" in wheat using reflectance measurements and neural networks. Computers and Electronics in Agriculture, *44*(3), 173–188. 2004

[22] Ebrahimi, M. A., Khoshtaghaza, M. H., Minaei, S., & Jamshidi, B. Vision-based pest detection based on SVM classification method. Computers and Electronics in Agriculture, *137,* 52–58. 2017

[23] UAV Imagery vs. Satellite. Retrieved February 2, 2021, from https://www.cropcopter.co/uav-imagery-vs-satellite/

[24] Survey3N OCN Camera Software Settings. Retrieved October 20, 2020, from https://www.mapir.camera/products/survey3n-camera-orange-cyan-nir-ocn-ndvi

[25] OCN Filter Improves Results Compared to RGN Filter. Retrieved November 3, 2020, from https://www.mapir.camera/pages/ocn-filter-improves-contrast-compared-to-rgn-filter

[26] Perko, R., Raggam, H., Gutjahr, K., & Schardt, M. The capabilities of TerraSAR-X imagery for retrieval of forest parameters. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, *38,* 452-456. 2010.

[27] 6 Spectral Indexes to Make Vegetation Analysis Complete. Retrieved November 5, 2020, from https://eos.com/blog/6-spectral-indexes-on-top-of-ndvi-to-make-your-vegetation-analysis-complete/. February 22, 2019.

[28] Broadband Greenness. Retrieved November 5, 2020, from https://www.harrisgeospatial.com/docs/broadbandgreenness.html

[29] Landsat Surface Reflectance-Derived Spectral Indices. Retrieved November 10, 2020, from https://www.usgs.gov/core-science-systems/nli/landsat/landsat-surface-reflectance-derived-spectral-indices

[30] Raster Calculator. Retrieved November 20, 2020, from https://www.bluemarblegeo.com/knowledgebase/global-mapper-19/Raster_Calculator.htm

[31] Zhao, B., Duan, A., Ata-Ul-Karim, S. T., Liu, Z., Chen, Z., Gong, Z. et al. Exploring new spectral bands and vegetation indices for estimating nitrogen nutrition index of summer maize. European Journal of Agronomy, *93,* 113–125. 2018.

[32] Normalized Difference Rededge. Retrieved December 1, 2020, from https://eos.com/agriculture/ndre/

[33] Jiang, J., Cai, W., Zheng, H., Cheng, T., Tian, Y., Zhu, Y. et al. Using Digital Cameras on an Unmanned Aerial Vehicle to Derive Optimum Color Vegetation Indices for Leaf Nitrogen Concentration Monitoring in Winter Wheat. Remote Sensing, *11*(22), 2667. 2019

[34] Overview of clustering methods. Retrieved December 3, 2020, from https://scikit-learn.org/stable/modules/clustering.html

[35] Image Thresholding. Retrieved December 5, 2020, from https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

[36] Kumar, A. Computer Vision: Gaussian Filter From Scratch. Retrieved December 5, 2020, from https://medium.com/@akumar5/computer-vision-gaussian-filter-from-scratch-b485837b6e09. March 17, 2019.

[37] Tarımsal Araştırmalar ve Politikalar Genel Müdürlüğü. ZEYTİN FİDANTIRTILI. In M. Aydemir (Ed.). *Zirai mücadele teknik talimatları* (5th ed.) (140-141). Ankara. 2008.

[38] Parrot Sequoia FAQs. Retrieved December 10, 2020, from https://www.pix4d.com/product/sequoia/faq

[39] MicaSense Altum. Retrieved December 10, 2020, from https://micasense.com/altum/

APPENDICES

Appendix A: Methods

```
coordinates = iterateKMLFile() -> findTag("coordinates") -> getText() -> split(" ")
x,y,boundary_box = [],[], []
for each point in coordinates:
        point = point -> split(",")
        x = append() -> point[0]
        y = append() -> point[1]
x_min,y_min, x_max, y_max = getMin() -> x , getMin() -> y , getMax() -> x , getMax() -> y
boundary_box = append() -> [(x_min,y_min,x_max,y_max)]
geom = createPolygon() -> [input = boundary_box]
masked_image = mask() -> (input = Input_TIFF_File , mask = geom, crop = True)
```

Method A.1: Pseudocode of the cropping process

```
dataset1 = gdal -> Open() -> (cropped_tif_file)
orange = dataset1 -> GetRasterBand(1) -> ReadAsArray() -> (type = float)
cyan = dataset1 -> GetRasterBand(2) -> ReadAsArray() -> (type = float)
nir = dataset1 -> GetRasterBand(3) -> ReadAsArray() -> (type = float)
dataset2 = gdal -> Open() -> cropped_dsm_file
dataset3 = gdal -> Open() -> cropped_dtm_file
dsm = dataset2 -> getRasterBand(1) -> ReadAsArray() -> (type = float)
dtm = dataset3 -> getRasterBand(1) -> ReadAsArray() -> (type = float)
dcm = dsm – dtm
ndvi = (nir – orange) / (nir + orange) IF exception 0 ELSE value
nni = 0.95 * ((orange – cyan) / (orange + cyan)) + 0.14 IF exception 0 ELSE value
ndwi = - (cyan – nir) / (cyan + nir) IF exception 0 ELSE value
```

Method A.2: Raster band and indices preparation for dataset

```
x_offset, px_width, x_rotation, y_offset, px_height, y_rotation = dataset1 ->
getGeoTransform()
image_column = dataset1 -> RasterXSize -> value
image_row = dataset1 -> RasterYSize -> value
crs = osgeo-> osr -> SpatialReference() -> ImportFromWkt() -> (dataset1 -> getProjection())
t_crs = osgeo-> osr -> SpatialReference() -> ImportFromEPSG() -> 4326
transformFunction = osgeo -> osr -> CoordinateTransformation() -> (input = crs, target =
t_crs)
lat_points, lng_points = [],[]
for i= 0 to image_row:
        for j=0 to image_column:
                x = x_offset + px_width * i + x_rotation * j
                y = y_offset + px_height * i + y_rotation * j
                (latitude,longitude,z) = transformFunction -> TransformPoint() -> (x,y)
                lat_points -> append() -> latitude
                lng_points -> append() -> longitude
```

Method A.3: Acquirement of pixel coordinates

```
all_data = [[lat_points,lng_points,orange,cyan,nir,dsm,dtm,dcm,ndvi,ndwi,nni]]
for i to all_data[0].size:
        all_data[0][i] = numpy-> reshape () -> (all_data[0][i] , -1)
lat_points,lng_points,orange,cyan,nir,dsm,dtm,dcm,ndvi,ndwi,nni  = all_data[0] ->
splitIntoArrays()
image_dataset = numpy -> stack() ->
((lat_points,lng_points,orange,cyan,nir,dsm,dtm,dcm,ndvi,ndwi,nni), axis = -1)
column_names = ['lat','lng','orange','cyan','nir','dsm','dtm','dcm','ndvi','ndwi','nni']
dataFrame = pandas -> DataFrame() -> (image_dataset, columns = column_names)
```

Method A.4: Dataset creation

```
labeled_data = []
kmeans = MiniBatchKmeans() -> (init=kmeans++, batch_size = 100, n_clusters = 5) -> fit() ->
(dataFrame -> ndvi)
c_centers = kmeans -> cluster_centers
labels = kmeans -> labels
sorted_list = numpy -> sort() -> (c_centers)
for i = 0 to labels.size
        if labels[i] = sorted_list[-1] -> index
                labeled_data -> append() -> 'ground'
        else if labels[i] = sorted_list[-2] -> index
                labeled_data -> append() -> 'shadow'
        else
                labeled_data -> append() -> 'tree'
dataFrame -> results = labeled_data
new_kmeans_data = dataFrame -> getNDVI() -> if results = 'tree'
```

Method A.5: K-means pixel classification

```
tree_result = []
kmeans = MiniBatchKMeans(init = kmeans++, batch_size = 100, n_clusters = 5) -> fit() ->
new_kmeans_data
labels = kmeans -> labels
c_centers = kmeans -> cluster_centers
sorted_list = numpy -> sort() -> (c_centers)
for i = 0 to labels.size
        if labels[i] = sorted_list[-1] -> index
                tree_result -> append() -> "other_vegetation"
        else
                tree_result -> append() -> "tree"
dataFrame -> results -> getRowsByValue() -> "tree" -> replaceColumn() -> results -> (input =
tree_result)
```

Method A.6: Second K-Means for further pixel clustering

```
image_hist = calculateHistogram() -> target_image
normalized_hist = image_hist / (max() -> image_hist)
histogram_bins = [for i=0 to 256]
predicted_threshold = -1
minimum_variance = infinity
for each item in normalized_hist:
        temp = temp + item
        cumulative_sum -> append (temp)
for i = 1 to 256:
        weight_bg,weight_fg = normalized_hist -> splitColumns() -> i
        cumulative_bg, cumulative_fg = cumulative_sum[i], cumulative_sum[255]  -
cumulative_sum[i]
        bin_bg, bin_fg = histogram_bins -> splitColumns() -> i
        mean_bg, mean_fg = sum(weight_bg * bin_bg) /cumulative_bg , sum(weight_fg *
bin_fg) /cumulative_fg
        variance_bg, variance_fg = sum((bin_bg− mean_bg)^2 * weight_bg) /
cumulative_bg , sum((bin_fg− mean_fg)^2 * weight_fg) / cumulative_fg
        within_class_variance = variance_bg * cumulative_bg + variance_fg *cumulative_fg
        if within_class_variance <= minimum_variance:
                minimum_variance= within_class_variance
                predicted_threshold = i
```

Method A.7: Otsu’s thresholding algorithm [35]

```
filter_size = input-> value()
kernel_size = pow (filter_size,2)
/* filter_size should be an odd number */
gaussian_kernel = zeroMatrix() -> (filter_size,filter_size)
sigma = 0.3*((kernel_size -1)*0.5 - 1) + 0.8
row = filter_size / 2
column = filter_size / 2
for x in range(-column, column+1):
        for y in range(-row, row+1):
                x1 = 2 * pi * pow(sigma,2)
                x2 = exponential(-(pow(x,2) + pow(y,2))/ (2 * pow(sigma,2))
                gaussian_kernel[x+row, y+column] = (1/x1)*x2
image_zero_matrix = zeroMatrix() -> (target_image_x, target_image_y)
image_zero_matrix = convolveForAllBands() -> (image_matrix,gaussian_kernel)
```

Method A.8: Gaussian Blur Algorithm [36]

```
gray_img = turnGrayscale() -> target_img
gray_img = addGaussianBlur() -> (filter_size =25, input = gray_img)
threshold = threshold_otsu() -> (input = gray_img)
euclidean_dist_map = ed_transform () -> (input= threshold)
local_maxima = find_peaks() -> (input=euclidean_dist_map, min_distance = 10, labels = threshold)
marker_array = findConnectedPixels() -> (input = local_maxima, structure = "8-connectivity")
result_labels = watershed() -> (input = - euclidean_dist_map, markers = marker_array, mask=
threshold)
circles = []
for labels = 1 to result_labels.max():
        if labels == 0
                skip
        mask = zeroMatrix() -> (gray_image_x, gray_image_y)
        mask[result_labels == labels] = 255
        contour = findContours() -> (input=mask)
        x,y,r = drawCircleFromContourArea() -> (input = contour)
        circles -> append((x,y,r))
        for each c in circles:
                drawCircle() -> (input = target_img , coordinates = ((c[0],c[1]),c[2]))
```

Method A.9: Tree individualization process

```
define function getBothCircleValues(circles, new_sum_X, new_sum_Y, new_sum_radius,
first_circle_id, next_circle_id):
        new_sum_X += circles -> getDataFromIndex() -> (index = first_circle_id, column = 0)
        + circles -> getDataFromIndex() -> (index = next_circle_id, column = 0)
        new_sum_Y += circles -> getDataFromIndex() -> (index = first_circle_id, column = 1)
        + circles -> getDataFromIndex() -> (index = next_circle_id, column = 1)
        new_sum_radius += circles -> getDataFromIndex() -> (index = first_circle_id, column
        = 2) + circles -> getDataFromIndex() -> (index = next_circle_id, column = 2)
        return new_sum_X,new_sum_Y,new_sum_radius
```

Method A.10: Coordinate and radius extraction and summation

```
define findPairs(circles,threshold)
	distance, threshold_distance = [],[]
	temp_circle = circles
	for i = 0 to temp_circle.length:
		for j = i+1 to temp_circle.length:
			dist = findEuclideanDistance() -> (x1 = temp_circle[i][0], x2 =
		temp_circle [j][0], y1 = temp_circle[i][1], y2 = temp_circle[j][1])
			distance -> append() -> (i,j,dist)
	for i=0 to distance.length:
		if distance[i][2] < threshold:
			threshold_distance -> append() -> (distance[i])
	return temp_circle, threshold_distance
```

Method A.11: Circle pair selection based on distance

```
define mergeCircles(circles,labels,threshold_distance)
	for i = 0 to threshold_distance.length:
		new_sum_X,new_sum_Y, new_sum_radius = 0,0,0
		if i in index_list
			pass
		indices = findIndices() where ((threshold_distance-> first_element =
	threshold_distance[i][0] or threshold_distance[i][1]) or (threshold_distance->
	second_element = threshold_distance[i][0] or threshold_distance[i][1]))
		all_circles = threshold_distance -> getIndices() -> indices -> getCircle()
		previous_circle = []
		until all_circles <= previous_circle
			previous_circle = all_circles
			for circle in all_circles:
				indices = findIndices() where ((threshold_distance->
		first_element = circle) or (threshold_distance-> second_element =circle))
				index_list -> append() -> all i in indices if i is not in index_list
				found_circles = threshold_distance -> getIndices() -> indices
		-> getCircle()
				all_circles -> append() -> found_circle in found_circles if
			found_circle not in all_circles
			for ft=1 to all_circles.length:
				temp_circle -> dropIndex() -> all_circles[ft]
				labels -> changeValue() -> (input = all_circles[ft], target =
all_circles[0])
				for st=ft+1 to all_circles.length:
					new_sum_X, new_sum_Y, new_sum_radius=
				getBothCircleValues(circles,new_sum_X,new_sum_Y,new_s
				um_radius, all_circles[ft-1], all_circles[st-1])
				count = all_circles.size * (all_circles.size -1)
				temp_circle -> changeValues() -> (index= all_circles[0],
input= (new_sum_X/count,new_sum_Y,new_sum_radius/count))
	return labels,temp_circle
```

Method A.12: Circle merging method

```
define treeStatusExtraction(circles,labels,dataFrame,threshold)
        threshold_indices = circles -> getIndices() -> (target = getRadius() > threshold)
        dataFrame -> treeID = labels
        dataFrame -> treeID -> changeValues() -> (values = threshold_indices, target_values
        = 0)
        tree_results = dataFrame -> groupBy() -> (target= treeID) -> getMean() ->
        (target_values = [orange,cyan,nir,lat,lng,ndvi,ndwi,nni,dcm] -> dropIndex() -> (index
        = 0)
        pixel_size = dataFrame -> groupBy() -> (target = treeID) -> getSize()
        tree_results -> pixelsize = pixel_size
        tree_results -> pixelsize_ratio = pixel_size / (sum() -> (pixel_size))
        return tree_results
```

Method A.13: Tree status identification

Appendix B: Figures


```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document id="root_doc">
<Schema name="orto_kml" id="orto_kml">
    <SimpleField name="id" type="float"></SimpleField>
</Schema>
<Folder>
    <name>orto_kml</name>
    <Placemark>
        <Style><LineStyle><color>ff0000ff</color></LineStyle><PolyStyle><fill>0</fill></PolyStyle></Style>
        <ExtendedData><SchemaData schemaUrl="#orto_kml">
        <SimpleData name="id">45</SimpleData>
    </SchemaData></ExtendedData>
      <MultiGeometry><Polygon><outerBoundaryIs><LinearRing><coordinates>
      28.7811178764154,41.0254032731177
      28.7816226138678,41.0256440191007
      28.7824949779353,41.0259193529233
      28.7831074634301,41.0262108910418
      28.7834999418554,41.026232904329
      28.7840075921859,41.0264773043514
      28.7846717700399,41.0268167896057
      28.7850918109079,41.0267191305314
      28.7848817029771,41.0265056336092
      28.7845780235417,41.0263513507248
      28.7846366348632,41.0260340594355
      28.7842950967669,41.0258252467585
      28.7837112187869,41.0255565901469
      28.7833572401279,41.0254052024485
      28.7832381710208,41.0254531514088
      28.7828413669608,41.0252706275386
      28.7823611954047,41.0250067187939
      28.7815940377389,41.0246878906908
      28.7811187875197,41.0248139481331
      28.7809425742952,41.0251304057488
      28.7809537094244,41.0253358565066
      28.7811178764154,41.0254032731177
      </coordinates></LinearRing></outerBoundaryIs></Polygon></MultiGeometry>
    </Placemark>
</Folder>
</Document>
</kml>
```


Figure B.1: KML file structure