Statistical Learning Methods for Tree Classification using Remote Sensing Imagery

---

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Sarah Maebius

May 2021

Approved for the Division
(Mathematics - Statistics)

_____

Tom Allen

Acknowledgements

Deeply grateful to...

Table of Contents

List of Tables

List of Figures

Abstract

Over the past decade, Western Redcedars have reportedly been in decline. This tree species is native to the Pacific Northwest and necessary to the ecosystem. The decline of Western Redcedars lacks published literature, so this research seeks to answer the question of where Western Redcedars are located to better understand the decline of this species. The approach to this question applies statistical learning methods to remote-sensing RGB imagery at a spatial resolution of 3 meters combined with field-level data to predict six tree species: Bigleaf Maple, Douglas-Fir, English Oak, Giant Sequoia, Norway Maple, Western Redcedar. The models trained in the research were random forest models and support vector machines. The best performing model was a random forest model with 7 predictors on 5 tree species (by grouping maple and oak trees into a "Broadleaf" category) with an overall testing accuracy of 0.65 and 0.22 for Western Redcedars. This model was applied to a masked image spanning Portland's city boundary to predict the locations of Western Redcedars. The model allows researchers to locate clusters of Western Redcedars for tracking the progress of the species overtime in locations where field crews do not have the resources to visit and record tree information. This research also improves methods of classification using freely accessed satellite imagery with low spatial and spectral resolution.

Introduction

0.1 Research Problem and Background

Western redcedar trees are evergreen trees that typically grow up to 75 feet tall and are located throughout the Pacific Northwest, making it an organism with a tolerance for shaded regions with moist environments. These trees are native to the land and have served many purposes to people and animals living in the vicinity of the trees, including medicinal, building, and habitat functions (Peterson, n.d.). Western redcedars were culturally important to indigenous peoples who valued the strong wood and utilized it for construction and everyday necessities ("Western red cedar," n.d.). Over the past decade, reports of dead western redcedars have been increasing, suggesting that something other than natural causes is killing off this species. In general, western redcedars experience several hardships in surviving in the Pacific Northwest, with common causes of death such as forest fires, clearcutting, small animals eating the saplings, and harsh weather including strong winds that easily uproot the trees (Peterson, n.d.). Dying redcedar trees can be identified by their branches which turn brownish yellow or fall off completely. Another sign is that the top of a tree will turn brown and lose leaves ("Western redcedar Dieback," n.d.). Losing this native tree would have a detrimental effect on animals in the area who rely heavily on the trees for their lifestyle. Scientists have speculated that western redcedar decline might be caused by recent dry summers, the spread of tree disease, insects, or other weather related events ("Western redcedar Dieback," n.d.). Since this is a recent issue, there is not a lot of resources explaining the decline. This research aims to provide more insight into the cause of the western redcedar decline by first predicting the location of the western redcedar trees in Portland and then predicting their condition in terms of health. Having more insight helps to prevent further tree deaths and save the western redcedar species, which also extends to similar tree species and provides more knowledge about environmental changes in the Pacific Northwest region.

Modelling in this research will be conducted by combining information gathered from remotely sensed images with ground level information. There are several sources publishing research done using satellite imagery for land classification (Castelluccio, Poggi, Sansone, & Verdoliva, 2015) and for predicting tree species (Fricker et al., 2019), which will be the groundwork for this project's application of remote sensing models to the specific topic of western redcedar mortality.

Remote sensing is a process for identifying the physical aspects of a large area of land by collecting and measuring the reflected light using airplanes or satellites. This method provides access to lot of information about the region of interest that would otherwise be limited from a human's ground-level persepective. Remote sensing is used to detect characteristics on land as well as the sea. Some remote sensing land applications include tracking forest fires, volcanic eruptions, city growth, and also forest changes.

Satellites orbit the earth, carrying sensors that record levels of electromagnetic radiation detected by the reflection of the sun on the earth. The data collected is the measured radiation from different regions of the spectrum (visible light, infrared, ultraviolet, etc.). Depending on the surface, sunlight gets absorbed or reflected back

at the orbiting satellite. The remotely sensed data is available at certain resolutions depending on the instruments used. Spatial resolution is the area imaged by a satellite image, where a detailed image has smaller spatial resolution hence smaller pixels. Spectral resolution is a measure of the size of the wavelength interval, where smaller bands and smaller wavelengths improve the spectral resolution of an image and enhance the level of detail. Satellite images need to be corrected to combine the bands into an image depending on the analysis.

Imagery come in various band widths and different number of bands. Hyperspectral imagery has narrow bands of sizes 10-20nm and hundreds of bands. So, hyperspectral imagery is successful in distinguishing fine details, but at a cost of more complexity. Multispectral imagery has wider bands and up to 10 total bands. This type of imagery is not complex to work with, but misses some details that hyperspectral imagery detects.

Previous literature has been published in utilizing satellite imagery for predictive models, however, many issues arise in classifying land type through remote sensing mainly due to image quality. A single image can be composed of image strips taken over the course of multiple flight paths. Consequently, these images are taken at different times of day, which compiles into a single image with a lot of variation due to changes in the weather as well as the different angles of the sun's position (Castelluccio et al., 2015). Ideally, a solid classifying model surpasses any error introduced by imperfection in the satellite images. Common approaches to classification on satellite images include support vector machines, random forests, and decision trees. One study on land type classification explores the performance of convolutional neural networks (CNN) as classification models (Castelluccio et al., 2015).

Another related study identifies 7 tree species by applying a hyperspectral CNN model. This study was completed using field data with measured information about tree count, tree species, and mortality status and hyperspectral imagery data. The hyperspectral data is converted into the form of a tree canopy height model with circular polygons centered at individual tree canopies. The study analyzes the performance of CNNs for both hyperspectral imagery data and a Red-Green-Blue (RGB) subset of the hyperspectral imagery data. The results of the experiment conclude that training a CNN on the hyperspectral data outperforms the CNN trained on RGB data in classifying land type on the UC Merced Land Use dataset. The RGB model does not perform as well as the hyperspectral model in terms of distinguishing tree classes, but does perform around the same level of accuracy in terms of genus classification. For this project, the data comes from Planet.com and only has RGB and IR data available for the Portland region.

This thesis follows along with the methods in (Fricker et al., 2019) for combining the satellite imagery data with ground data by creating spatial polygons and extracting pixel-level information to train classification models. The work in this thesis differs from the methods in the cited literature by using random forests and support vector machines as classification models instead of CNNs and uses the RGB model instead of hyperspectral model. Finally, this thesis applies established classification methods for satellite imagery data to answer the specific question about the cause of death of western redcedars in the Pacific Northwest. The models predict the loca-

tions for each tree species to facilitate the identification of any patterns in the species mortality over the past decade.

0.2 Overview

0.2.1 A Statistical Learning Approach to Identifying Location of Western Redcedars

This work combines RGB imagery data from Planet.com with ground level tree data from the RStudio `pdxTrees` library and applies random forest and support vector machine classification methods to predict the location of tree species (specifically western redcedars) in Portland, Oregon and model the condition of western redcedars to ultimately understand the cause of death in this species. Chapter 1 describes the two levels of data in this research: ground level and pixel level data, where the data comes from, and how the data is combined for modeling. Chapter 2 discusses the process behind training random forest and support vector machine models. Chapter 3 summarizes how the data is prepared for the final model and the outcome of the research followed by a discussion and conclusion.

Chapter 1

Data

There are two sources of data in this project. Data at a pixel level come from satellite images downloaded from Planet.com (`https://www.planet.com`), and data at the ground level come from library `pdxTrees` in RStudio (`https://github.com/mcconvil/pdxTrees`).

1.0.1 Imaging

Satellite images come from Planet.com's PS2.SD instrument found on Dove-R satellites that were launched in 2017. Satellites have an approximate frame size of 20 km x 12 km parallel to flight path. From a single pass, 4 to 5 overlapping framed scenes are combined to form a tile. To separate the light into red, blue, green, and near-infrared (NIR) channels, the telescope has a high-performance butcher-block filter made of 4 individual pass-band filters. Planet.com equates this pass-band filter with that of Sentinel-2.

As it orbits the Earth, the satellite captures continuous strips of single frame images that are split into a RGB frame and NIR frame. The butcher-block pass separately photographs the red, blue, green, and NIR bands and then combines all four to form an image. Images are are downloaded already fully processed and ready to be analyzed. The process includes corrections for radiometric calibration, terrain distortions corrections, elevation corrections, and atmospheric corrections.

The images used in this research were taken on September 2nd, 2020 at an altitude of 475 km. Images are taken from the summer months to reduce the chance of rain clouds and capture peak greenness of the trees. The compiled multi-spectral image is composed of 4 bands: blue, green, red, and NIR, with center wavelengths 490 nm, 565 nm, 665 nm, and 865 nm respectively, with an average bandwidth of 41 nm. The average spatial extent is around 25 km by 8 km. The images cover the entire Portland area, with specific measurements of 26 km from west to east and 30 km from north to south. The spatial resolution of the pixels is approximately 3 meters.

Planet.com provides a way to download free images that are already processed and corrected for analysis. Some drawbacks to using this data are the limited number of downloads available per free account, the low spatial resolution, which decreases the performance of statistical models trying to predict tree species location, and the low

spectral resolution, so the satellite sensor cannot detect wavelengths in detail.

1.0.2 Ground Data

Ground level data is available from the 'pdxTrees' package in RStudio. This data was collected as part of the Portland's Parks and Recreation's Urban Forestry Tree Inventory Project, which collected park tree information from 2017 to 2019. Originally, the inventory project started with the goal of improving the city's tree management plans. Under the guidance of US Forestry staff, volunteers around the city's neighborhoods are trained in identifying tree species and provided with tools necessary to record tree measurements. Measurements in feet were made using diameter tape. The Portland park trees inventory consists of over 25,000 trees, each with information about tree location, size, species, and health. For the purposes of this project, the following variables were selected:

- Spatial information: Longitude and latitude of tree. Location of tree is recorded on a tablet with Collector for ArcGIS and recorders manually select the tree from a satellite image on the screen. The tree's location is mapped over the satellite images in QGIS for analysis.

- Species: Tree genus, species, and common name. The tree species is used to train the model and predict the tree species of pixels in the test data.

- Crown size: Crown width from north to south, crown width from east to west, and the base height of the crown in feet. The crown of the tree is the tree's above ground leaves, so the crown width is the longest horizontal distance that can be measured between the leaves of the tree. A measuring wheel is used to measure this distance.

- Tree Condition: Categorical variable with four categories (from best to worst), good, fair, poor, and dead. A tree was considered good if the tree is strong and has no apparent issues, fair if the tree is average condition with possibly a few dead branches, poor if the tree has major wounds and dead major canopy loss, and dead if the tree has no live leaves.

To prepare the ground level data for training the model, the data was filtered by species and crown size. The following tree species were identified and included: maples, oak, Douglas-fir, Western Redcedar, and sequoia. These species are all native to the Pacific Northwest, so they benefit the ecosystem and having a model of these trees will help track the species' locations. Since the data is used to identify trees in the satellite images, the ground data was also filtered to only include the trees with a crown width from north to south of 20 feet or more, this ensures that there will be around 9 to 20 pixels per tree.

Portland's Tree Inventory includes a variable for identifying trees including Western Redcedars as well as variables for filtering to include larger trees canopies (at least 6m in diameter) in the dataset, which helps construct a training set of tree polygons

over a satellite image. This data would be even better suited for the purposes of this project if the street trees subdata also contained variables for tree canopy size.

1.0.3 Training Data

To datasets were used to train the models and their performance was compared. A pixels dataset was created by extracting pixel band values (red, green, blue, and infrared) from manually drawn polygons around 100 of each tree species, so that the pixels dataset has multiple pixel observations representing a single tree. The other dataset was a point pixel dataset created by only extracting band values from the pixel directly below the point indicated by a street or park tree. This dataset is smaller than the pixels dataset and differs from the pixels dataset in that it is more likely to have the pure tree pixel values instead of maybe a dirt or shadow pixel value. Having the two datasets allows a comparison between the performance of polygon pixel extraction methods versus single point extraction methods.

Characteristics of Pixels Training Data

The training data at the ground-level was filtered by tree size and tree type. Trees with at least 20 feet in north to south crown width were included in the training dataset to ensure that the polygons around each tree contain around 9 to 20 pixels. Figure 1.1 plots the north to south crown width values against the west to east crown width values to present the ranges of crown widths in the entire pdxTrees parks dataset and to justify only using the north to south crown width for filtering the dataset since the two variables have a strong positive correlation.

Table 1.1: Common tree names included in the tidy data and their total counts.

Tree Name	Tree Count
Douglas-Fir	6485
Norway Maple	1406
Western Redcedar	652
Bigleaf Maple	464
Giant Sequoia	315
English Oak	135

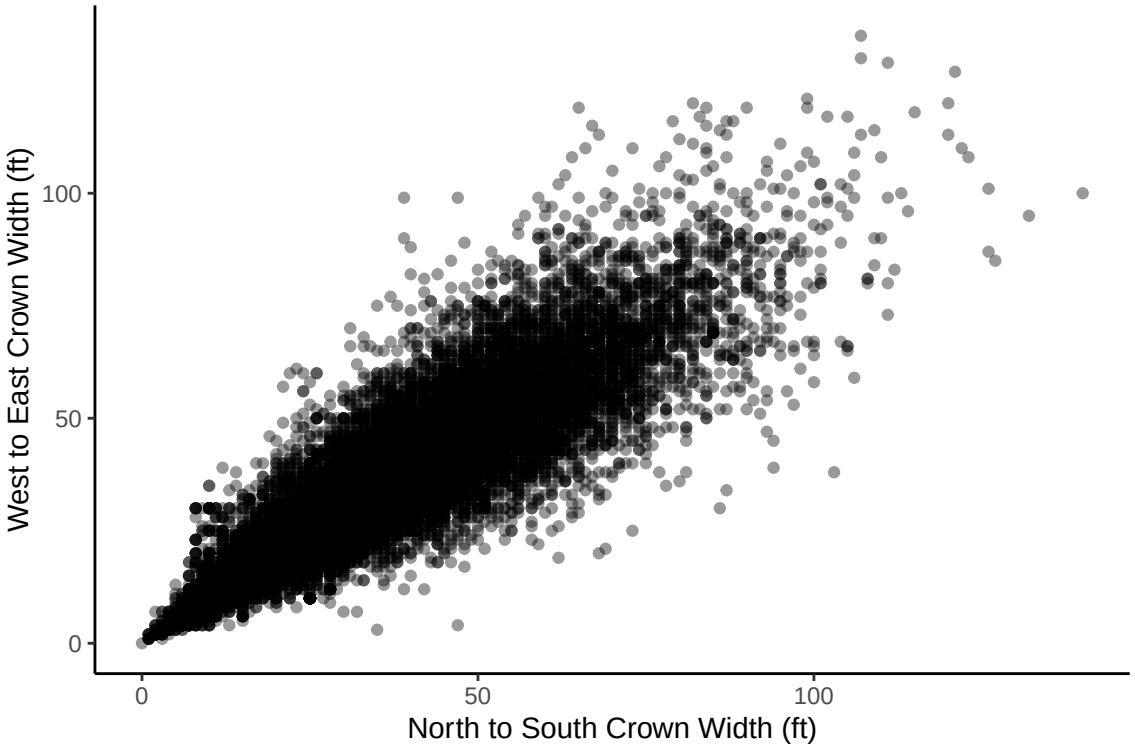

Figure 1.1: Plot comparing North to South Crown Width (ft) to West to East Crown Width (ft). A tree with a higher north-south crown width has a higher west-east crown width, so without loss of generality the north-south crown width variable is used to filter the dataset for larger trees only.

Six tree types were determined to have large enough canopies on average and appear frequently enough to construct a training dataset with plenty of observations for each species of tree: Douglas-Fir, English Oak, Giant Sequoia, Maple, Western Redcedar. Table 1.1 provides the number of trees under each common name category. The largest group contains 6485 observations of Douglas-Fir trees, and the smallest group contains 135 observations of English Oak trees.

Creating Spatial Polygons

The training dataset was converted into shapefiles for each type of tree and exported into QGIS where polygons were manually drawn around 100 of the trees for each species. The point shapefile layers were displayed over the raster images downloaded from Planet.com in QGIS, which provided a guide point for locating individual trees. Then using QGIS's drawing tool, a polygon is carefully drawn around the tree canopy (see Figure 1.2 and Figure 1.3). Most polygons turned out to be four to six-sided polygons to retain the general shape of the tree canopy. Trees polygons were only created if the outline of the tree canopy was clearly visible or surrounded by other trees of the same species in order to avoid including pixels from the wrong species in that polygon. Also, the shadows of the trees are visible in the raster images, so the polygons were drawn with the intention of not including the tree shadow. For the five different species of trees, at least 100 polygons were drawn around trees of that type, with each polygon containing at least 6 pixels and at most 20 pixels. The polygon shapefiles were then exported into RStudio where the rest of the analysis was conducted.

Table 1.2 displays the number of polygons per raster strip and the amount of pixels from those polygons. Table 1.3 displays these counts in terms of tree species.

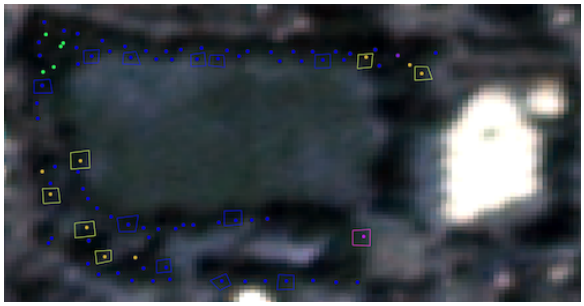

Figure 1.2: Polygons drawn around different tree species: maples (blue), sequoias (yellow), redcedars (pink). Note there are no polygons around trees with canopies that overlap with canopies of different tree species, like douglas-firs (green), to avoid misclassifying pixels from different tree species.

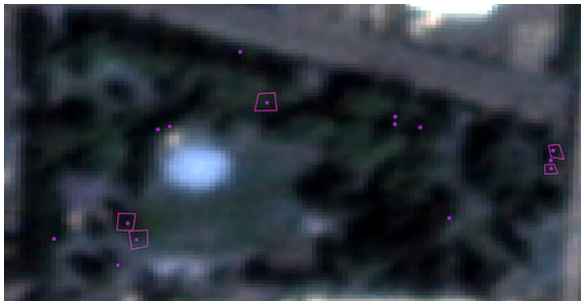

Figure 1.3: Polygons drawn around redcedar trees

Table 1.2: Number of polygons per raster strip

Raster Strip	Polygons	n
a	197	3614
b	257	4887
c	36	359
d	8	98
e	177	2395

Table 1.3: Number of polygons per tree type per raster strip

Raster Strip	Tree Name	Polygons	n
a	Bigleaf Maple	28	283
a	Douglas-Fir	35	432
a	English Oak	30	288
a	Giant Sequoia	26	336
a	grass	1	1158
a	Norway Maple	48	747
a	Western Redcedar	29	370
b	Bigleaf Maple	26	279
b	Douglas-Fir	38	604
b	English Oak	62	618
b	Giant Sequoia	32	399
b	grass	3	1579
b	Norway Maple	70	1026
b	Western Redcedar	26	382
c	Bigleaf Maple	9	55
c	Douglas-Fir	19	202
c	English Oak	2	39
c	Western Redcedar	6	63
d	Douglas-Fir	5	56
d	Western Redcedar	3	42
e	Bigleaf Maple	37	268
e	Douglas-Fir	6	86
e	English Oak	6	80
e	Giant Sequoia	54	701
e	grass	1	186
e	Norway Maple	37	567
e	Western Redcedar	36	507

Table 1.4: Variables and pixel values included in the pixels dataset

ID	red	green	blue	ir	rstrip	id_type	Genus	Species	Cmmn_Nm	Cr_W_NS	Cr_W_EW	Crw_B_H	Conditn	St_Wdth	ndvi	red_blue	ir_red	red_green	blue_green
43	4707	4125	3187	6627	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.1694018	1.476938	1.407903	1.141091	0.7726061
43	4384	3860	2864	7111	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.2372336	1.530726	1.622035	1.135751	0.7419689
43	4446	3803	2807	7443	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.2520818	1.583897	1.674089	1.169077	0.7381015
43	4619	3911	2859	7801	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.2561997	1.615600	1.688894	1.181028	0.7310151
43	4695	4060	3201	5996	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.1216911	1.466729	1.277103	1.156404	0.7884236
43	4344	3719	2728	6747	a	acpl	Acer	ACPL	Norway Maple	28	28	8	Fair	NA	0.2166622	1.592375	1.553177	1.168056	0.7335305

Polygon Limitations

Ideally, the raster strips contain training trees of each species, and as a result, the polygons are evenly distributed across the raster strips, but due to limitations of our ground-level data as well as the coverage of the strips across the city of Portland, this is not achievable. Each of the raster strips contains some number of manually drawn polygons, however, raster strips 'C' and 'D' are missing some of the tree species. The raster strips do not all cover Portland to the same extent. Strips 'A' and 'B' encompass most of the city while strips 'C' and 'D' only make up a small portion that covers the corners of the city not reached by strips 'A' and 'B'.

The polygons are drawn with the intention of outlining pixels for a known tree species so that the extracted pixels truly represent that species and so that there is a sufficient amount of pixels per species, however, limitations of the satellite images' resolution make some error inevitable. Many parks trees have canopies that overlap, which cause some polygons to contain pixels from trees of different species. Other pixel errors might come from including pixels that are cast in shadow by the polygon's training tree or by surrounding infrastructure. To avoid these errors, polygons with uncertain canopies are cross-checked with the satellite filter on Google Maps. For example, a point in QGIS that is indicated as a douglas-fir tree might appear as one tree on the low-resolution image from Planet.com, but a closer look at Google Maps shows that it is actually two trees in close proximity. Having low spatial resolution decreases the number of pixels available within a polygon, but that has to be balanced out with the need for a large number of pixels. With regards to the number of pixels per tree species, having around the same number of tree polygons per species results in differing amounts of pixels per species due to the different average sizes of tree canopies for different species. After drawing the images, the pixels totals per tree species is computed to ensure that the training pixels data contains at least 800 pixels per tree species.

Combining Ground-level Data with Pixel-level Data

The first spatial join in RStudio was conducted to match up each Spatial Polygon with a point in the training dataset. Then the raster images were loaded and joined with the polygons to extract the pixel values inside each polygon for all 4 bands (red, green, blue, infrared). Ultimately this turned into a pixel table with rows representing each pixel with its corresponding polygon, light reflection intensity values for all 4 bands, and the ground information about that tree. Table 1.4 displays the first few entries of the pixels dataset. Table 1.5 contains the summary of the total number of pixels for each tree type. Figure 1.4 displays the range of reflection intensity

Table 1.5: Number of pixels per tree type

Tree Name	Pixels Count
Bigleaf Maple	885
Douglas-Fir	1380
English Oak	1025
Giant Sequoia	1436
grass	2923
Norway Maple	2340
Western Redcedar	1364

pixel values per tree type. Each tree species has similar ranges of values over the red, green, blue, and infared bands. Of the species included in the training data, the Giant Sequoia trees appear to have higher density counts for the red, green, and blue bands.

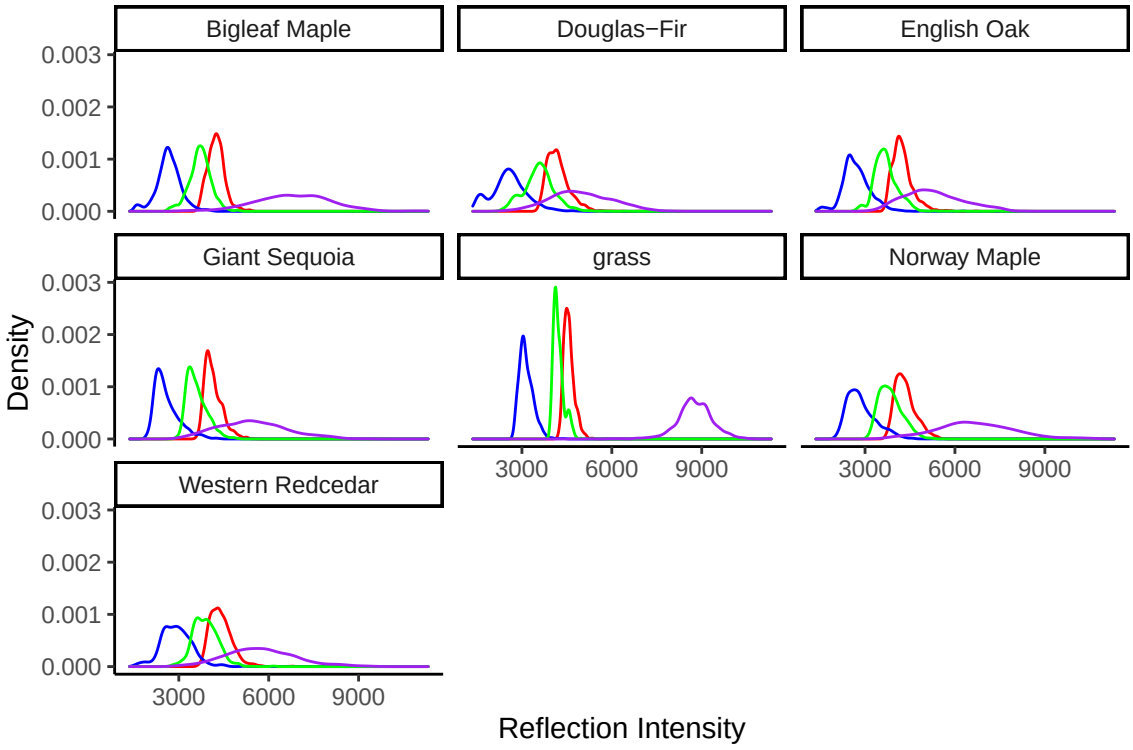

Figure 1.4

1.1 Preparing Raster Images

The training model has to be applied to the entire raster image to predict the location of Western redcedars, however, extracting all the pixels from raster images is a slow process. Filtering the rasters to keep vegetation and to recognize the difference between grass pixels and tree pixels alleviates the computational intensity of extract-

ing all the pixels and improves the performance of the models. To reduce the size of computations, all five raster strips are masked and cropped to only include pixels within the city boundary of Portland, since that is the region of interest, and that is the extent of the ground level data.

1.1.1 Masking Vegetation

The satellite images in RStudio need to be masked to reduce the number of pixels that the model has to classify and prevent the chance of a non-vegetation surface being predicted as a tree. A common mask applied to raster images is a Normalized Difference Vegetation Index (NDVI) mask. This index is a measure of the greenness of a pixel, with higher values indicating vegetation and lower values indicating infertile areas such as a rock. The formula for NDVI is $NDVI = \frac{NIR-Red}{NIR+Red}$. Figure 1.5 displays the density of NDVI values over the raster images. To remove the pixels that are not vegetation, an NDVI threshold is determined to be 0.00 to create a mask dividing the pixels into vegetation (1) and non-vegetation (0). When the trained model is applied to the raster image, it is applied to the masked raster image that only keeps pixels with NDVI mask values of 1.

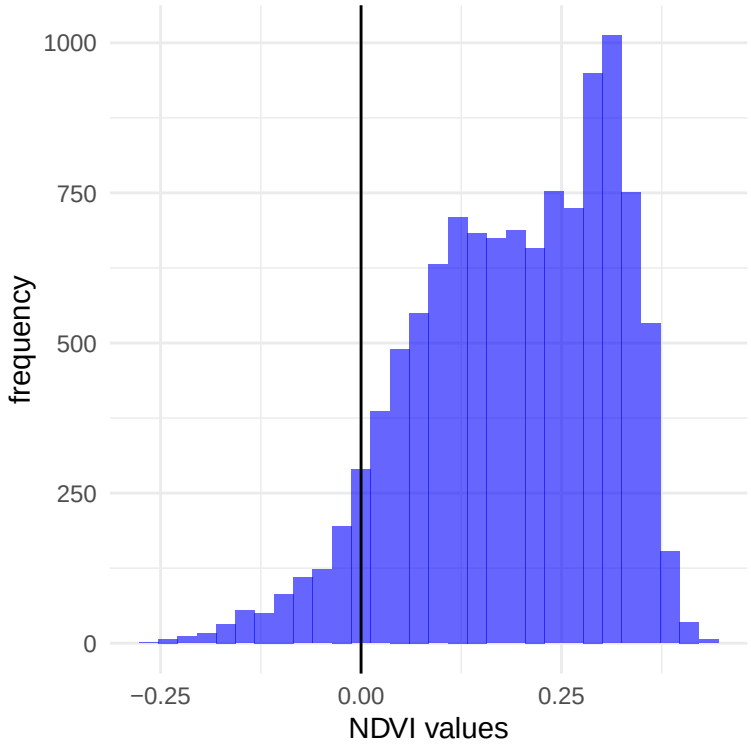

Figure 1.5: Normalized Difference Vegetation Index (NDVI) is a measure of the greenness of a pixel. The histogram displays the frequency of the tree pixels' NDVI values. A good threshold for masking out the nonvegetation pixels is an NDVI of 0.

1.1.2 Masking Grass

After masking the images to only contain vegetation pixels, the effectiveness of a second mask was investigated to try and determine whether it is feasible to distinguish between tree pixels and grass pixels in a satellite image. This would be useful for preventing the final model from predicting fields of grass as trees. Inspired by previous research, a grass index was created by averaging the values of the four bands, $GRASS = \frac{RED+BLUE+GREEN+NIR}{4}$ (Qian, Zhou, Nytch, Han, & Li, 2020). Grass polygons were created in QGIS to add a grass attribute in the pixels dataset. Five fields of grass were outlined as polygons in QGIS. In RStudio, the pixels were extracted from the grass polygons for a total of 2,923 grass pixels. Figure 1.6 displays how the average band values for the grass pixels differ from the other tree types average band values. Based on the figure, pixels in the raster image that have an average band value above 4900 are filtered out. Masking the grass pixels ensures that a field of grass will not be classified as a tree when applied to the entire raster image.

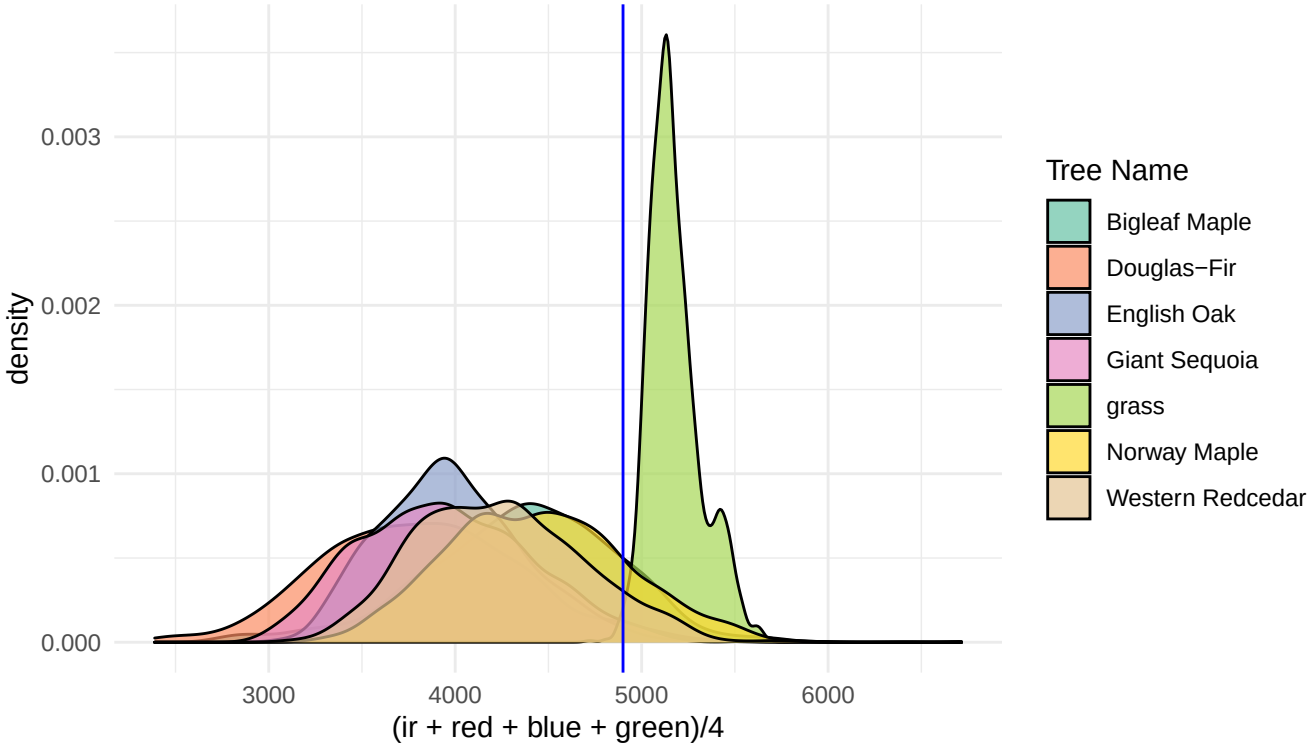

Figure 1.6: Average of band values distribution across different tree types in comparison with grass pixel average band values. Most grass pixels have an average value above 4900, so a mask is applied to the raster image to remove pixels with average values above this threshold.

Figure 1.7 displays a raster strip with every pixel included (top), only the pixels with NDVI values above 0.00 (middle), and only pixels with NDVI values above 0.00 and average band values below 4900 (bottom). Ideally, the bottom image is left with only the tree pixels in the image.

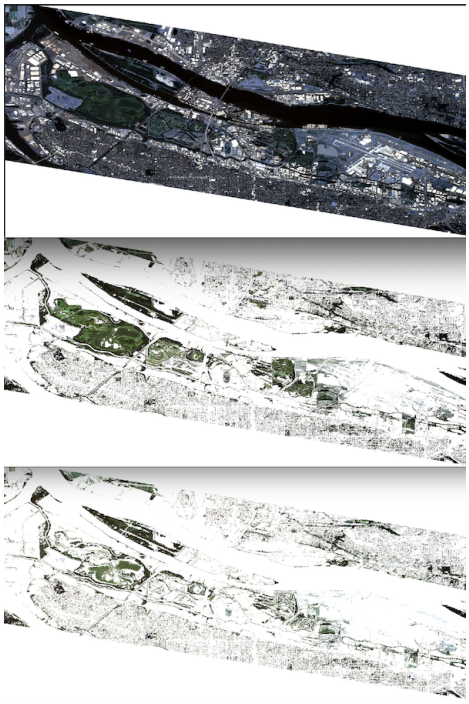

Figure 1.7: A raster strip with every pixel included (top), only the pixels with NDVI values above 0.00 (middle), and only pixels with NDVI values above 0.00 and average band values below 4900 (bottom)

1.1.3 Masking Limitations

The low resolution of the raster images cause shapes to appear blurry, and especially tree canopies that are close together appear as a single formation.

Masked raster images were imported in QGIS and compared to Google Maps. In general, the NDVI mask was successful in keeping the forest pixels while still removing building structures and roads. For the GRASS mask, if a field of grass is a brownish color, it gets removed, but some greener fields are not masked out. The masks consistently remove river pixels but not lake pixels if the lake is a greenish color. For example, Smith Lake in North Portland is considered an urban wetland, which signifies a body of water with a lot of vegetation. From the perspective of a satellite, this looks like a field of grass. However, the average band pixel values are less than 4900, so they are not masked out by the final GRASS mask stage, and instead are treated as tree pixels in the model (see Figure 1.8).

To address leftover grass pixels in the filtered raster image, access to lidar data would provide a method of determining the height of certain pixels. Setting a height threshold would ensure that grass pixels are removed while higher vegetation pixels are kept. This would also address the masks' limitations in removing lakes with high vegetation, since those pixels would not meet the threshold for height. The GRASS mask did not perform as hoped, so the grass pixels extracted from polygons were included to train the model to distinguish observations as trees or grass pixels instead of applying a second mask on the raster images.

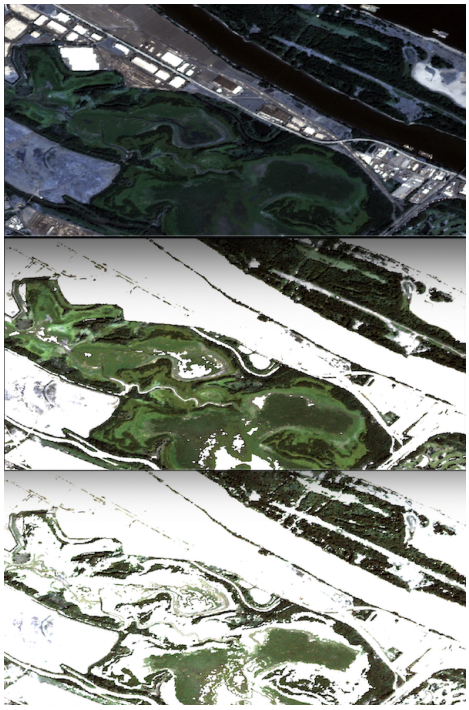

Figure 1.8: Satellite view of Smith Lake (top), after NDVI mask applied (middle), and NDVI mask and GRASS mask applied (bottom). The final masked image still contains parts of the lake because the aquatic vegetation fits the pixel values descriptions set by the masks.

Chapter 2

Methods

2.1 Existing Methods

Since the Western redcedar decline is recent, there are currently no published work for identifying tree species like Western redcedars for the Pacific Northwest using satellite imaging. However, a similar study identifies tree species for a strip of land in California by combining ground-level data with satellite images (Fricker et al., 2019). Their methods involved drawing polygons around rastered images that are layered with ground data as explained in Chapter 1, but the pixels from the polygons are used to train a convolutional neural network (CNN) model instead of the random forests and support vector machines in this study. The CNN model was then evaluated with k-fold cross validation. CNN models are appropriate in a classification setting and when working with satellite images because they account for spatial relations, which is likely to appear in classifying tree species. This study follows the methods of preparing the data for modelling as well as the method of cross validation to evaluate the performance of the random forests and support vector machines models. Random forests are used in a classification setting or regression setting and have the advantage of being simplier to train than CNN models and still performs at a similar level. Random forests use random feature selections to create decision trees and increase the number of correctly classified observations. Since this research strives to provide a computationally simple method of estimating tree locations to track changes in the ecosystem, a random forest is suitable for the study. If a researcher was interested in making more precise predictions, a CNN would require more time but would produce better estimates.

2.2 Current Methods

This research applies RF and SVM models on trained data and compares overall training performance, overall testing performance, overall polygon testing accuracy, and Western Redcedar training and testing performance. The following sections provide background about RF and SVM models.

2.2.1 Random Forest

In building a decision tree for classification, recursive binary splits are made by minimizing the Gini index (G), the total variance across K classes:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where $\hat{p}_{mk}$ is the proportion of observations in the m^{th} region from the k^{th} class (James, Witten, Hastie, & Tibshirani, 2013). A small Gini index value represents a region containing mostly observations from a single class, while a large Gini index value represents a lot of variation across classes. Then the decision tree is constructed by repeatedly considering different attributes and making splits where the Gini index is minimized. Note that this top-down, greedy approach tends to overfit the training data, so finding the decision tree that performs the best on test data involves a cost-complexity pruning algorithm to obtain smaller trees and apply k-fold cross validation to choose the best tree that minimizes the average error. K-fold cross validation is performed when training the models by splitting the data into $k = 10$ folds and training ten models each time withholding one of the ten folds until the process goes through every fold. This process aids in understanding how the models will perform against data that has not been included in training the model.

In the classification setting, a stronger predictive model than a decision tree is a random forest, which uses an element of randomness to decorrelate bootstrapped decision trees. Bootstrapped samples from the training dataset are used to construct decision trees, where splits are based on minimizing the Gini index when selecting from a random sample m of attributes from all p available. In common random forest applications, the number of attributes considered at a split is $m \approx \sqrt{p}$. Theoretically, by forcing only a subset of attributes to be featured in the tree, this expands the number of possible subtrees that might not have been achieved by the top-down, greedy approach. Hence, random forests provide a method of predicting classes with low variance while also keeping bias at a minimum.

2.2.2 Support Vector Machine

Another method investigated in this study imagines the multidimensional data in space and classifies the data using hyperplanes positioned to minimize misclassification error and maximize distance from observations to separate the data (James et al., 2013). Support vectors are vectors in space that represent the subset of observations which influence the hyperplane classifier. For example, in two dimensions, imagine a line separating A's one one side and B's on the other side. This line is the maximal margin hyperplane if it is the farthest possible from the observations and also separates the two classes perfectly. Observations that are closer to the hyperplane (support vectors) have more influence over the line if they move than observations that are further. Consequently, more support vectors indicates a classifier with lower variance and high bias. A tuning parameter can be altered to expand or shrink the

margin of error surrounding the hyperplane (how far away the observations have to be from the plane) and decide the level of tolerance for misclassifying observations.

Support vector machines can be applied in a linear or nonlinear setting with the use of kernels. A kernel is a function $K(x_i, x_{i'})$ of two observations specifying the similarities between two points. Given data with n observations, let x_i be an observation, where $i = 1, ..., n$. A linear kernel is

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

with a support vector classifier

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle.$$

The parameters α_i and β_0 are estimated using the inner products of the observations, but $\alpha_i \neq 0$ if and only if x_i is a support vector. Note that this function is similar to a linear regression function. Extending this model to a nonlinear context involves a similar approach as adding nonlinear terms to a regression model.

The polynomial kernel of degree d is defined to be

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

with support vector classifier

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i).$$

For some positive constant k, the radial kernel is defined to be

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right),$$

which makes the support vector machine more sensitive to nearby observations as opposed to further observations.

Support vector machines are less sensitive to outliers and successful predictors of categorical variables in high-dimensional data. A support vector machine does not perform well for data with observations that overlap and if the kernel is incorrectly selected.

2.3 Training Models

Data used to train the RF and SVM models are the pixels training data and the point pixels training data. To prepare the data for modelling, first the data is separated into a training set and then a test set by a ratio of 70% training data and 30% testing data. The `dplyr` function `slice_sample` was used to take a stratified sample from

the entire pixels dataset where 70% of each tree species in an individual raster strip was randomly selected. For example, the training set consisted of 70% of the bigleaf maple trees in raster strip 'A', 70% of bigleaf maple trees in raster strip 'B', and so on. From the 70% pixels training data, 500 observations for each tree species are extracted to obtain an even distribution of species. From the point pixels training data, 190 observations for each tree species are sampled.

The selected variables for training the models are the four bands (red, green, blue, infrared) and an NDVI variable, and the predictive variable is the tree species name. Some predictor variables were manually computed using ratios of bands. Typically, a long over short band ratio provides some more information to train the model. The following bands were created: red/blue, ir/red, red/green, and blue/green. Examining a correlation matrix with these predictors reveals possibly useful and not too highly correlated variables are red/green, red/blue, and blue/green (see Figure 2.1).

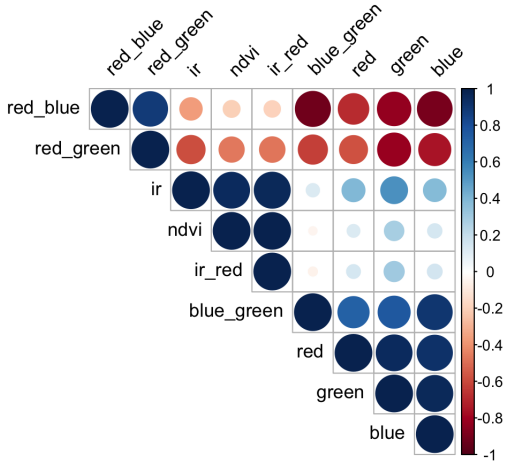

Figure 2.1: Correlation matrix of possible predictor variables to include in the model. Size and shade intensity represents stronger correlation (red/blue and blue are highly negatively correlated, while NDVI and green have almost no correlation). Positive correlation indicates higher values of one variable results in higher values of the other variable, while negative correlation indicates higher values of one variable results in lower values of the other variable.

To address the small number of predictor variables available to predict the large number of classes (7 tree species), some species are grouped together. The Bigleaf Maple, Norway Maple, and English Oak species are grouped under a new "Broadleaf" tree category. This reduces the number of classes to 5: Broadleaf, Douglas-Fir, Giant Sequoia, Grass, and Western Redcedar. Both the full 7 tree species training data and the grouped (5 class) species training data are used to train the models.

The `caret` package (Kuhn, 2020) in RStudio stands for classification and regression training, and it contains functions for facilitating the process of creating training models in R and determining the tuning parameters.

2.3.1 Random Forest Tuning

The `train` function from the `caret` package (Kuhn, 2020) with the specified `method = rf` option is used to train a random forest model on the training dataset to predict the tree species. The function takes parameters for data, the predictive variable, and the method of training the model. To perform k-fold cross validation, the `trainControl` function is used to create an object that tells the `train` function to perform cross validation 10 times with a different fold left out each time. The function automatically tries multiple number of tries at each split (mtry) and determines the optimal number through cross-validation. Training preliminary models with varying number of predictors provided a sense of the number of predictors to include in the training data that still add information to the model but avoid adding redundancy or multicollinearity to the model. For example, just having 5 predictors in the model results in a random forest that only selects one variable for each split, while 12 predictors guarantees an issue of multicollinearity. Models were considered with total predictors around 7 or 8 predicting classes of size 7 (individual species) or 5 (oaks and maples grouped together):

- 7 predictor model: red, green, blue, infrared, NDVI, $\frac{\text{red}}{\text{green}}$, $\frac{\text{blue}}{\text{green}}$
- 8 predictor model: red, green, blue, infrared, NDVI, $\frac{\text{red}}{\text{green}}$, $\frac{\text{red}}{\text{blue}}$, $\frac{\text{blue}}{\text{green}}$

7 predictor Random Forest

The 7 predictor random forest model uses the following predictors: red, green, blue, infrared, NDVI, $\frac{\text{red}}{\text{green}}$ and $\frac{\text{blue}}{\text{green}}$ band values predictors. Figure 2.2 displays the accuracy of the model predicting all 7 tree classes and the grouped 5 classes with the pixels training set with a range of mtry values. The 7 predictor model with the highest accuracy (determined by cross-validation) randomly selected 6 predictors at each split with an accuracy of 0.44. The variable with the highest predictive power was infrared band values with NDVI values with the second highest predictive power. For the grouped data, the optimal mtry value of 1 produced an accuracy of 0.57. The most important variable for this model was infrared followed by NDVI values.

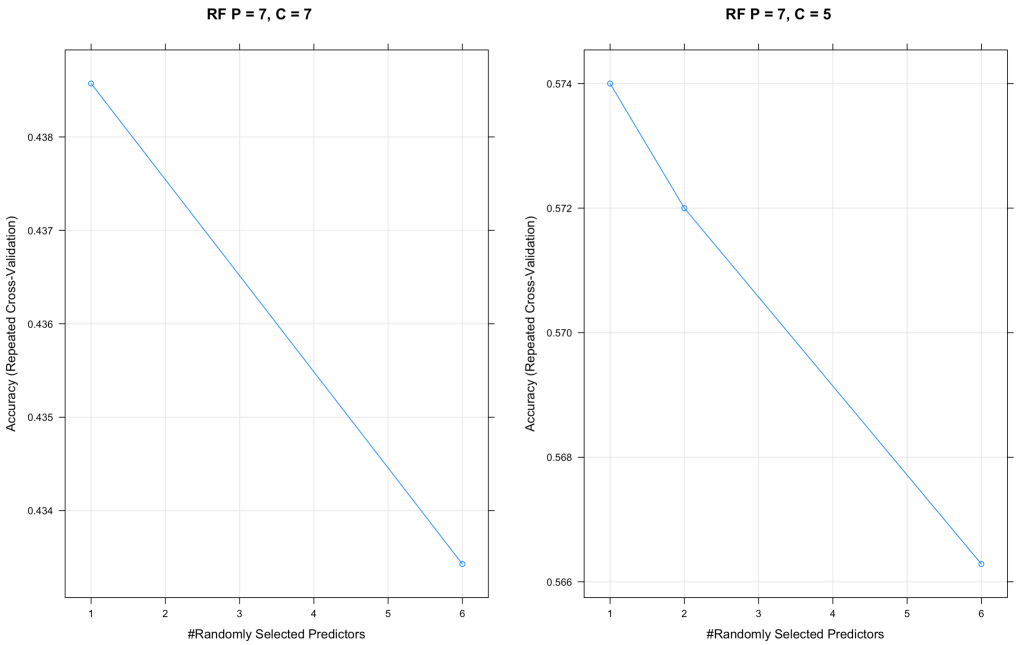

Figure 2.2: Accuracy of the 7 predictor trained random forest model for different mtry numbers over 10-fold cross-validation on 7 class and 5 class training data. (P = Predictors, C = Classes)

8 predictor Random Forest

The 8 predictor random forest model added a $\frac{\text{blue}}{\text{green}}$ predictor to the 7 predictor model. Figure 2.3 displays the results of a different number of randomly selected predictors to be considered at each split in the tree for 7 class data and 5 class. For modelling all 7 classes, the optimal mtry value was 4 with an accuracy of 0.45. The variables with the higher predictive power in decreasing order were infrared, NDVI, and $\frac{\text{red}}{\text{green}}$ band values predictors. The model using grouped data with 5 tree classes, had an optimal mtry value of 8 with an accuracy of 0.57. The most important variable was infrared.

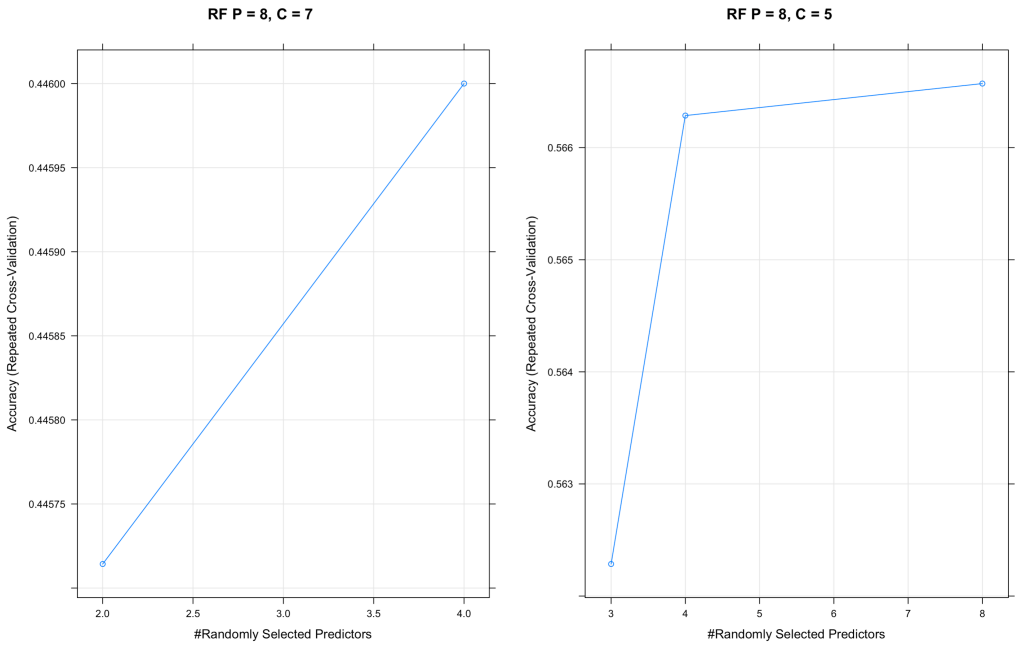

Figure 2.3: Accuracy of the 8 predictor trained random forest model for different mtry numbers over 10-fold cross-validation on 7 class and 5 class training data. (P = Predictors, C = Classes)

8 predictor Random Forest on Point Pixels

A RF model was trained using the 8 predictors from the previous model on the point pixels data. According to Figure 2.4, for modelling all 5 classes (Bigleaf Maple, Douglas-Fir, Giant Sequoia, Norway Maple, Western Redcedar), the optimal mtry value was 5 with an accuracy of 0.22. The variables with the higher predictive power were $\frac{\text{red}}{\text{green}}$ and red band values predictors. The model using grouped data with 4 tree classes (Broadleaf, Douglas-Fir, Giant Sequoia, Western Redcedar), had an optimal mtry value of 1 with an accuracy of 0.33. The most important variable was $\frac{\text{red}}{\text{green}}$.

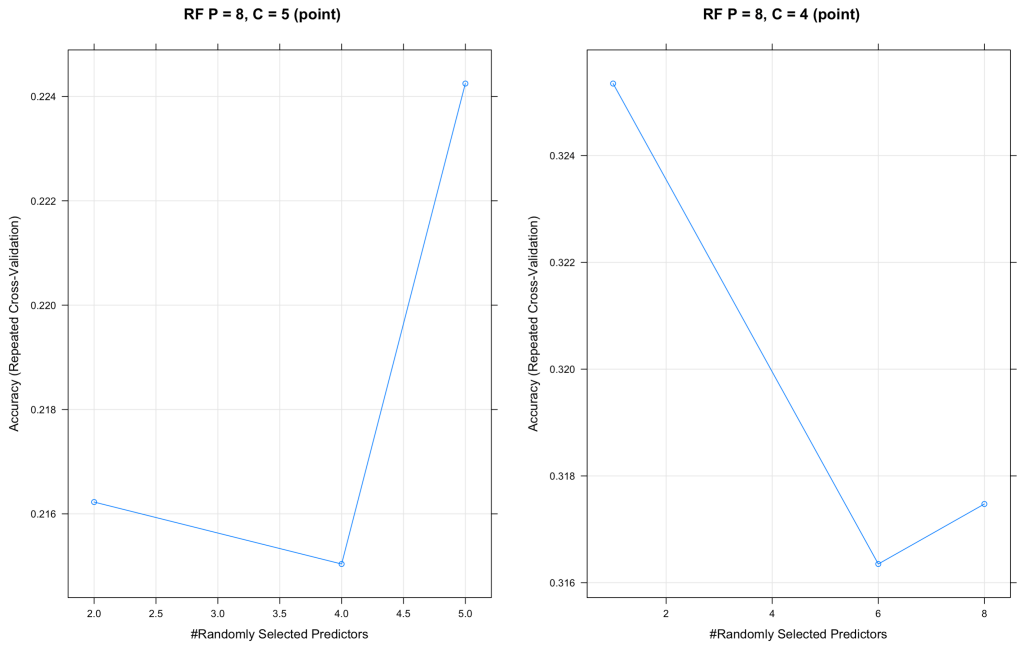

Figure 2.4: Accuracy of the 8 predictor RF Model on point pixels data for different mtry numbers over 10-fold cross-validation on 5 class and 4 class training data. (P = Predictors, C = Classes)

Best Random Forest

Figure 2.5 compares the results of the models based on overall model accuracy in predicting the test data.

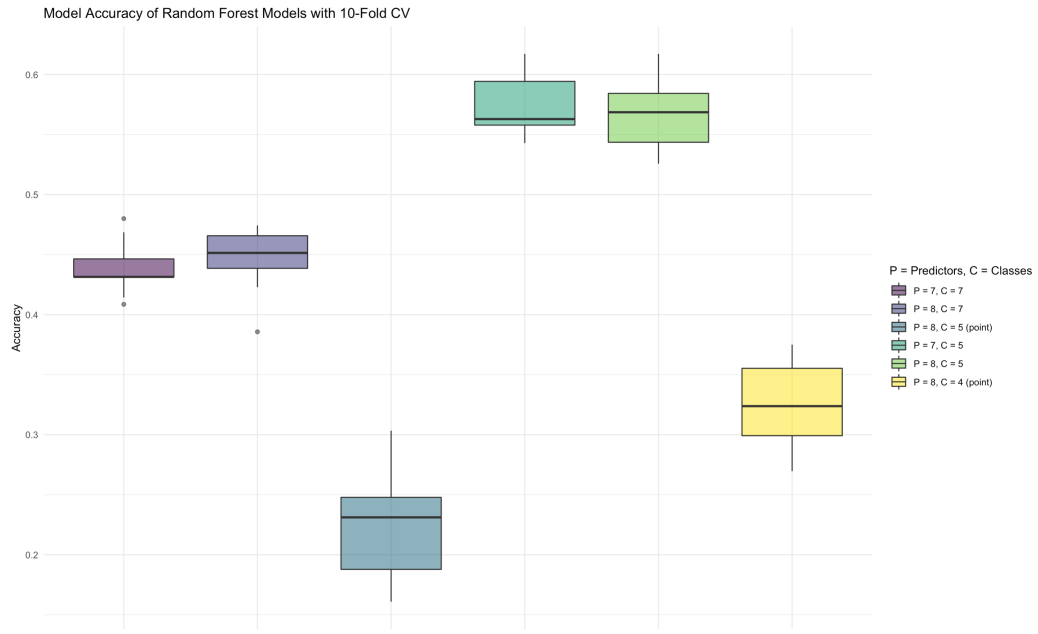

Figure 2.5: Comparison of results from random forest models with different predictors included

For modelling the 7 individual tree classes, the model with the 7 predictors included has a slightly higher maximum overall accuracy than the 8 predictor model. A confusion matrix located in (cite appendix) was constructed for each model averaging the entry counts over all ten cross validation resamples to investigate model performance just predicting Western Redcedar tree pixels. Both the 7 and the 8 predictor models performed similarly classifying Western Redceder pixels, the 7 predictor model had a prediction accuracy of 32%, the 8 predictor model had a prediction accuracy of 33%. Out of both models, Western Redcedars tend to be inaccurately classified under douglas-firs, english oak, and norway maple trees. The 8 predictor model trained on the point pixels data had a prediction accuracy of 29% for 5 class predictions. This model also misclassified Western Redcedars as norway maples, giant sequoias, and bigleaf maples.

Modelling only 5 tree classes by grouping some categories produced higher overall accuracies than the full classes models. The 7 predictor and 8 predictor model produced the highest overall average accuracies of 0.57 through cross-validation. The 7 predictor model and the 8 predictor model accurately predicted Western redcedars 36% and 35% of the time respectively. Most inaccurately predicted redcedars were classified under the broadleaf category. Models on both training sets predicted grass pixels with the highest accuracy. The 4 class model with the point pixel data had an overall accuracy of 0.33 with 0.25 Western Redcedar prediction accuracy.

Table 2.1 compares the hyperparameters accross all random forest models.

Table 2.1: Accuracy of random forest models with different number of predictors and mtry determined by ten-fold cross validation.

Random Forest Model	Accuracy	Kappa	mtry
P = 7, C = 7	0.4386	0.3450	1
P = 8, C = 7	0.4460	0.3537	4
P = 8, C = 5 (point)	0.2242	0.0300	5
P = 7, C = 5	0.5740	0.3819	1
P = 8, C = 5	0.5666	0.3772	8
P = 8, C = 4 (point)	0.3253	0.0024	1

2.3.2 Support Vector Machine Tuning

The `train` function has the `method = svmLinear/svmRadial/svmPoly` option to train a SVM model on the dataset to classify pixels into species of trees. For multiple class SVM training on k classes, this function classifies "one-against-one" by training $k(k-1)/2$ binary classifiers. This option also has parameters for cost, loss function, class weights, and normalized variables. For this project, three support vector machines were trained: linear, radial basis, and polynomial basis, and the preprocess setting for all three models were specified to center and scale (in R: `preProcess = c("center", "scale")`), which standardizes the variables. This is commonly performed with SVM models because the scale of the variables influences the optimal hyperplane decision. Since the light intensities extracted from the band data are only positive values, it is reasonable to consider normalizing the variables by setting the maximum value in the variable to 1 and the minimum value to 0, however, in comparing the SVM models under this preprocess setting (in R: `preProcess = "range"`) to the standardized setting, the accuracy of the models is not as high as the accuracy of the standardized variable models. The `caret` package selects the best cost tuning parameters based on accuracy through cross-validation.

Linear Support Vector Machine

The linear kernel SVM model had the highest training accuracy at 0.41 when cost parameter held constant at a value of 1. The final model contained 3017 support vectors. The linear kernel SVM model on the 5 classes data held the cost parameter constant at 1 predicted the grouped tree classes with an accuracy of 0.55 contained 2739 support vectors. The SVM model with point pixel data had an accuracy of 0.23 with C = 1 and 866 support vectors on 5 class data. On 4 class data, the linear SVM model with point pixel data had a prediction accuracy of 0.40 with C = 1 and 837 support vectors.

Radial Support Vector Machine

Figure 2.6 displays the results of different cost parameters on training accuracy for the radial basis kernel function on pixel data. The radial kernel SVM model on the

7 classes data set performed best when cost was 0.5 and $\sigma = 0.27$ for a training accuracy of 0.45. The final model had 2972 support vectors. The final values for the radial model on the 5 classes data set were $\sigma = 0.23$ and cost = 1 with an accuracy of 0.59 and 2630 support vectors. On the point pixels data, these values were C = 1, $\sigma = 0.32$, overall accuracy of 0.21, and 868 support vectors for the 5 class model. For the 4 class model on the point pixels data, the values were C = 0.1, accuracy of 0.40, and 849 vectors. See figure 2.7.

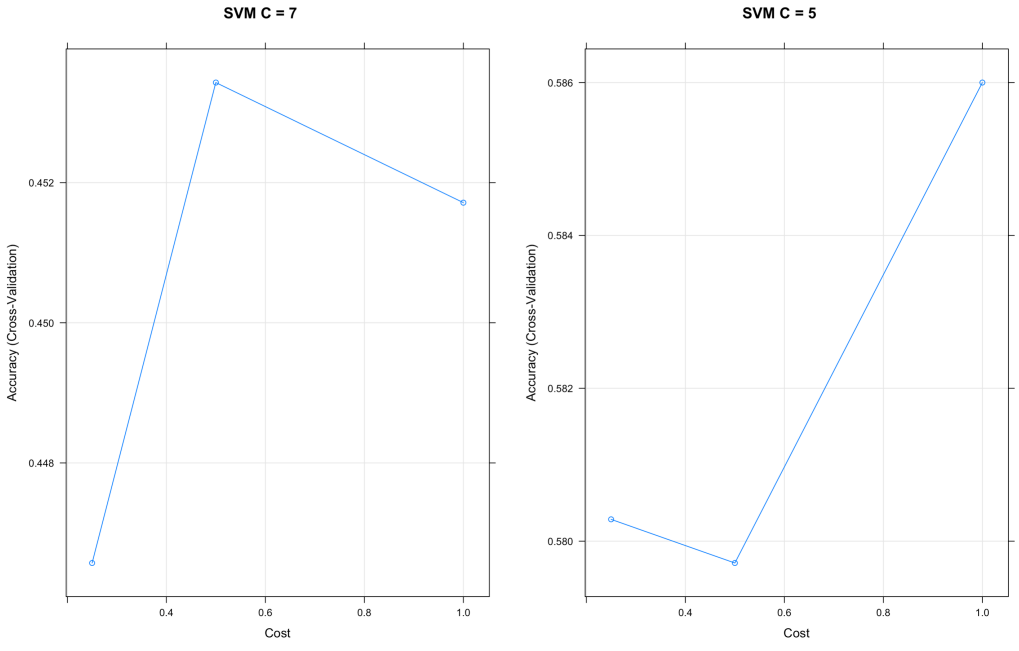

Figure 2.6: Comparison of accuracy results from different cost parameters for radial basis kernel function on pixel data.

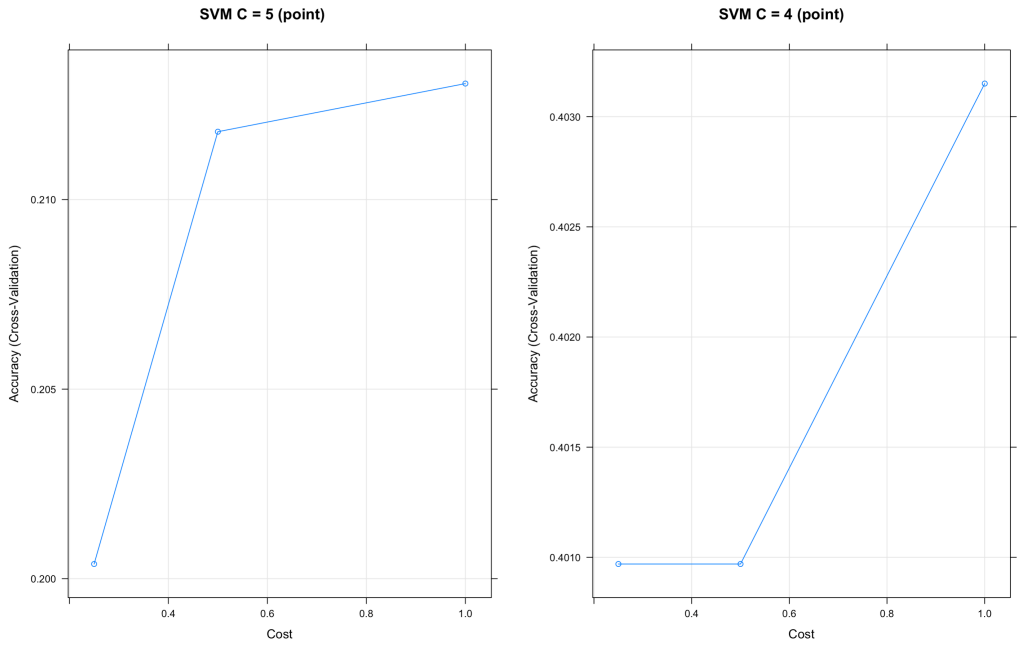

Figure 2.7: Comparison of accuracy results from different cost parameters for radial basis kernel function on point pixel data.

Polynomial Support Vector Machine

Figure 2.8 displays the results of different cost parameters on training accuracy for the polynomial basis function kernel. The final values used for the model predicting 7 classes were degree = 3, scale = 0.1, and cost = 1, which corresponded to a training accuracy of 0.44. The final model had a total of 2959 support vectors. The final model predicting only 5 classes produced an accuracy of 0.57 under degree = 3, scale = 0.1, and cost = 1. This final model had 2643 support vectors. Figure 2.9 shows that the results for the point pixels data with 5 class predictions had an accuracy of 0.23 for degree = 1, scale = 0.1, cost = 1, and 869 support vectors. For 4 class predictions, the overall accuracy was 0.41 with degree = 3, scale = 0.1, cost = 0.1, and 852 support vectors.

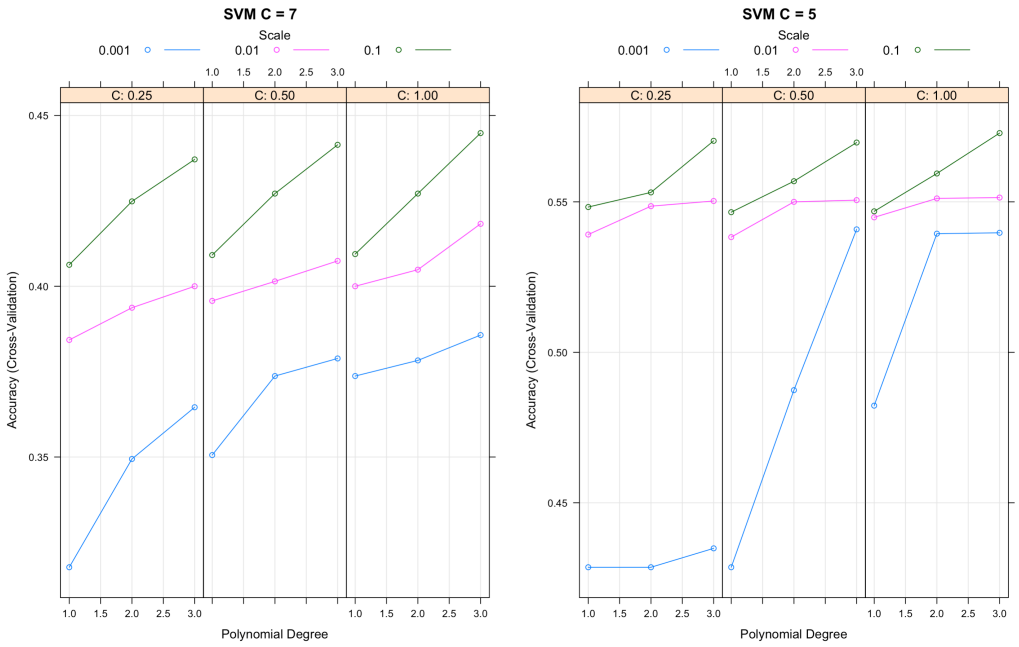

Figure 2.8: Comparison of accuracy results from different cost parameters with scale ranging from 0.001 to 0.1, C from 0.25 to 1, and degree from 1 to 3 for polynomial SVM on pixel data.

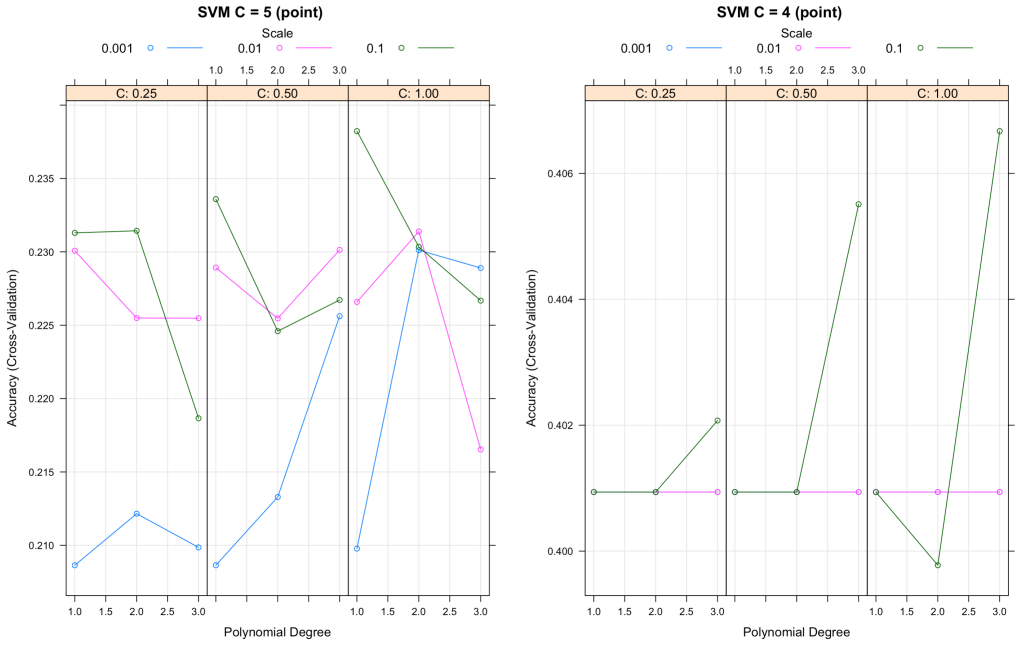

Figure 2.9: Comparison of accuracy results from different cost parameters with scale ranging from 0.001 to 0.1, C from 0.25 to 1, and degree from 1 to 3 for polynomial SVM on point pixel data.

Best Support Vector Machine

Figure 2.10 displays the results of the linear, radial basis, and polynomial basis support vector machine models. Based on the overall prediction accuracy, the radial basis is an appropriate kernel choice for the pixels data with the 5 class prediction.

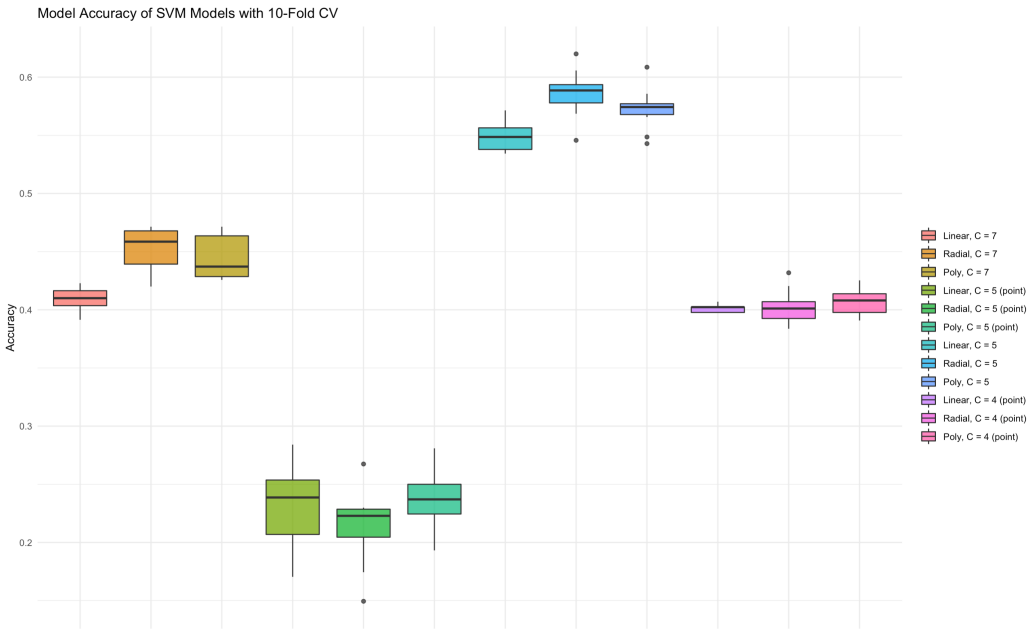

Figure 2.10: Comparison of results from support vector machine models with different kernel types: linear, radial, and polynomial.

As with the random forest model comparisons, a confusion matrix was constructed using the average counts over all ten cross-validation resamples to obtain the model results predicting Western redcedars. The support vector machine model that performed best overall was the radial basis kernel model, and it had a prediction accuracy of 35% for the Western Redcedar class. The polynomial basis kernel had a 34% prediction accuracy and the linear kernel also had an accuracy of 33% for Western Redcedars. For Western redcedar prediction on the grouped classes, the radial model had 44% accuracy. The highest Western Redcedar prediction accuracy for SVM models with point pixel training data was for the polynomial basis model at 24%.

Table 2.2 compares the hyperparameters accross all support vector machine models.

2.4 Testing Models

A subset of the data was withheld from training the models as test data to provide some measure of how the models perform in predicting results that the answer is unknown. Based on overall and redcedar accuracy, the appropriate models to consider with the testing data are the 5 class 7 predictor RF and the 5 class radial SVM. The

Table 2.2: Accuracy of support vector machine models with different kernels. Best parameter is determined by ten-fold cross validation.

SVM Model Kernel	Accuracy	Kappa
Linear, C = 7	0.4100	0.3117
Radial, C = 7	0.4534	0.3623
Poly, C = 7	0.4449	0.3523
Linear, C = 5 (point)	0.2314	0.0402
Radial, C = 5 (point)	0.2131	0.0169
Poly, C = 5 (point)	0.2382	0.0484
Linear, C = 5	0.5486	0.2778
Radial, C = 5	0.5860	0.3675
Poly, C = 5	0.5729	0.3310
Linear, C = 4 (point)	0.4009	0.0000
Radial, C = 4 (point)	0.4032	0.0120
Poly, C = 4 (point)	0.4067	0.0140

Table 2.3: Overall accuracy of 7 predictor RF (5 class) and Radial SVM (5 class) models on test data

RF Accuracy	SVM Accuracy
0.6533809	0.6583471

results for the overall test accuracy and the prediction test accuracy for Western redcedars are shown in Tables 2.3 and 2.4. According to the test data, the best performing models for further analysis are the radial SVM model for 5 class prediction and the 7 predictor RF model for 5 class prediction.

The polygon prediction accuracy on the test dataset was also computed to see how pixels of the same tree were classified compared to individual pixel classification. A "correct" prediction was indicated if more than half the pixels in a polygon are correctly predicted. Table 2.5 displays the results over polygons in the test dataset.

Table 2.4: Western redcedar accuracy of 7 predictor RF (5 class) and Radial SVM (5 class) models on test data

rf_accuracy	svm_accuracy
0.2164352	0.1134259

Table 2.5: Accuracy of radial SVM model with 5 classes and 7 predictor RF model with 5 classes for polygons in test data. A correct prediction is considered to be a polygon with more than half of the pixels correctly classified.

Type	Result	n
SVM C = 5	Correct	362
SVM C = 5	Incorrect	311
RF P = 7, C = 5	Correct	349
RF P = 7, C = 5	Incorrect	324

Chapter 3

Results

3.1 Training Results

Predicting on the grouped data produced the highest overall accuracies. The best model on the grouped data was the RF training model with an mtry value of 3 predictors at each split and the following predictors: red, green, blue, infrared, NDVI, $\frac{\text{red}}{\text{green}}$, $\frac{\text{red}}{\text{blue}}$. This model had a training predictive accuracy of 0.57 overall and 0.36 for Western redcedars specifically. On the test dataset, this model accurately predicted 65% of the pixels, 22% of Western redcedars, while the accuracy for predicting more than half the pixels correctly in the same polygon was 52%.

The model that performed best among SVM models was the radial SVM on 5 class data. This model had a training accuracy of 0.59 overall, 0.35 for just Western redcedars, and testing accuracy of 0.66 overall and 0.11 for Western redcedar pixels. For polygon predictions, the radial SVM C = 5 model had an accuracy of 54%. Both models perform similarly, however, for Western redcedar predictions, the RF model outperforms the SVM model. These models are used to proceed with the final model over Portland.

3.2 Modelling Tree Species in Portland

After masking the raster images of the entire region of Portland by NDVI values, the pre-trained models are used to predict the tree species of individual pixels in the masked raster images. The `predict.raster` function from RStudio's `raster` package takes a model and applies that model to each cell of the raster, leaving a raster with tree classification predictions. This function is used to predict tree species over all five raster strips, and the raster strips are merged to display predictions for the entire city. Figure 3.1 displays the tree classification predictions given by the optimal random forest model over Portland and figure 3.2 displays the predictions from the support vector machine model.

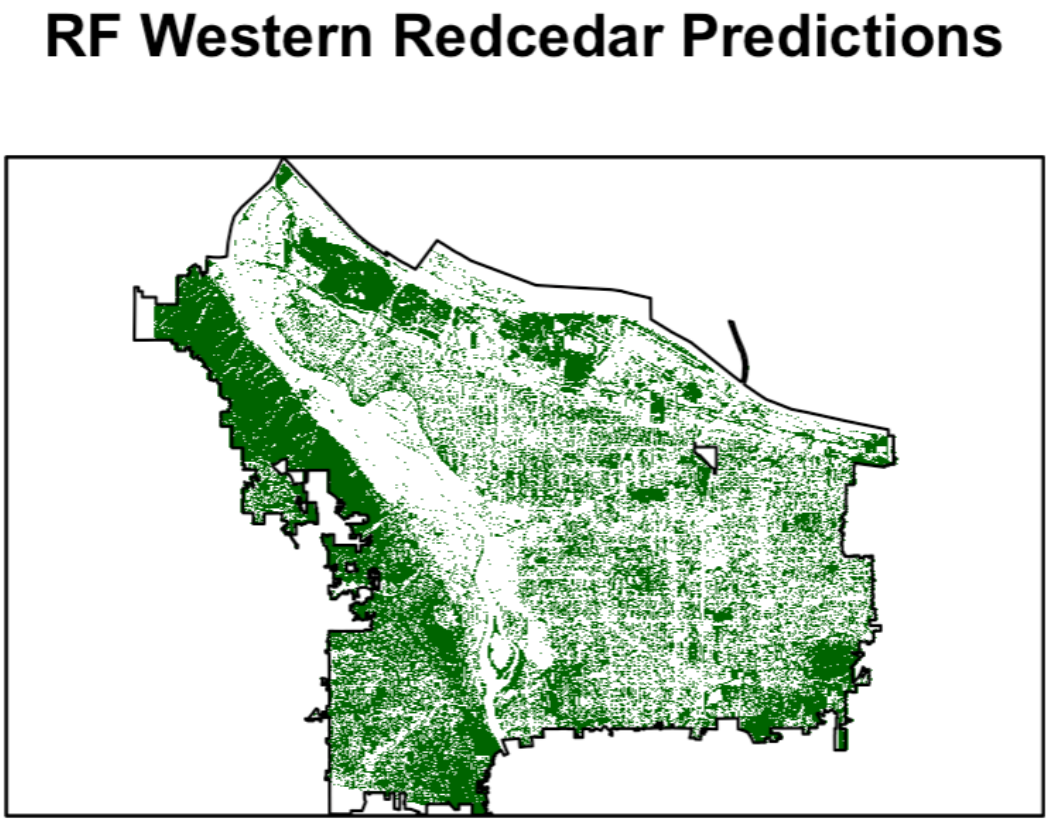

Figure 3.1: Model tree classification predictions of Western Redcedars over entire Portland region using the random forest model with 7 predictors on 5 classes.

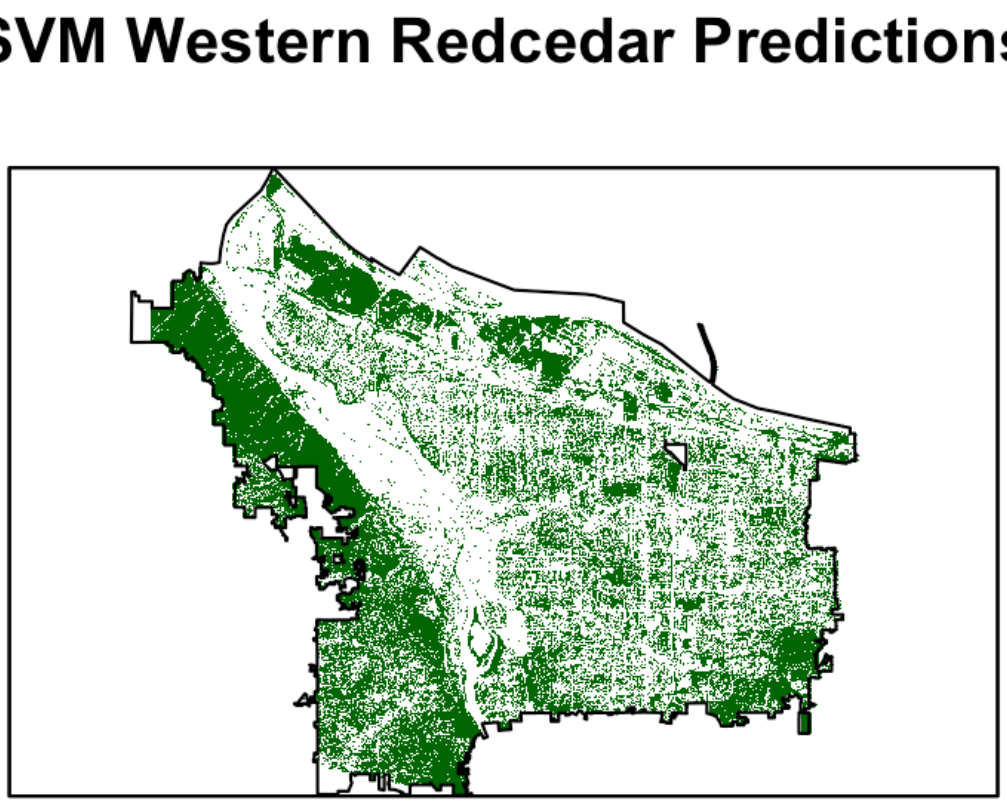

Figure 3.2: Model tree classification predictions of Western Redcedars over entire Portland region using radial SVM C = 5.

In comparing the RF P = 7, C = 5 model results to the radial SVM C = 5 results, 77% of the pixel predictions were the same predictions under both models.

To assess the performance of the model over the Portland region, the results are compared to the remaining Western redcedars from `pdxTrees` both street and park data. The shapefiles for `pdxTrees` Western redcedars ground data are mapped over the classified pixel predictions, and the number of correctly and incorrectly classified pixels is recorded to be 66% accurate for RF P = 7, C = 5 and radial SVM C = 5. Figure 3.3 visualizes this comparison for RF P = 7, C = 5.

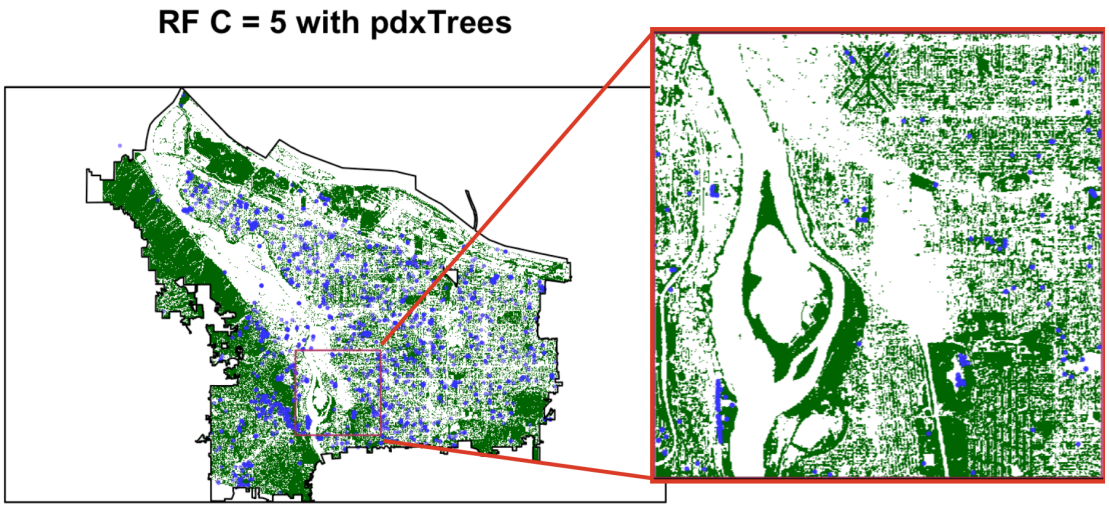

Figure 3.3: Comparison of Wester redcedar RF pixel predictions (in green) to Western redcedar trees from pdxTrees (in blue).

Chapter 4

Discussion

4.1 Polygon Pixel Extraction Versus Point Pixel Extraction

Following along with the Fricker Methods (Fricker et al., 2019), a training dataset is constructed with multiple pixels from a single tree (extracted from the polygon shape). Another dataset was included in the methods of this research constructed with single pixels representing individual trees to investigate how the methods compare. Especially with low-resolution imagery, this analysis questions whether several pixels representing one tree in training a model is more beneficial than having a single pixel representing one tree (where that pixel is the center location of the tree's canopy). In terms of overall accuracies, the point pixel models did not perform as well as the pixel models. The most comparable Western Redcedar prediction accuracy among the point pixel models was the RF P = 8, C = 5 model, which was 0.29, which is low compared to the radial SVM C = 5 model on pixel data with 0.44 Western Redcedar accuracy, but close to the RF C = 7 models which predicted around 0.33 for Western Redcedars. The other point pixel models performed worse in overall validation accuracy, suggesting that polygon pixel extraction provides more information to train the model with more accuracy. One explanation for this discrepancy is that the field data from `pdxTrees` does not perfectly line up with tree locations on the satellite images, and this is especially true for smaller trees. Larger trees tend to have canopies that extend over the point pixel location, making up for the error in the point representing the tree's marked location. On the other hand, smaller trees (especially with canopies less than 3 meters across) are not centered around the point pixel, instead they tend to be about a pixel away. Extracting these pixel values is likely extracting the values of the street, sidewalk, shadow, grass, or other structure instead of the desired tree pixel. This analysis reveals how the model performance is sensitive to the training data going into the model, which emphasizes the importance of accurate field work. A potential improvement could be to relocate the point pixels so that they are perfectly centered over the tree canopies, however, this approach would be time-consuming.

4.2 RF Performance Versus SVM Performance

The 7 predictor RF model predicting 5 classes outperforms the SVM model predicting 5 classes in Western redcedar training accuracy and testing overall accuracy as well as testing prediction of Western Redcedars. Surprisingly, the overall testing accuracies were higher than the validation accuracies for both the RF and SVM models. This suggests that the training set was not representative of the true data or the model was underfitting the data. The SVM model performs significantly worse than the RF model in Western Redcedar predictions on the test data. One explanation might be that the validation Western Redcedar accuracy aggregates the results of all 10 cross-validation sets, which might inflate the values, but the test results only consists of one set. Another explanation could be that the lower testing accuracy is an indication of model overfitting, however, the overall testing accuracy is higher than the overall training accuracy. In general, having a small number of predictors compared to the number of classes tends to perform poorly. It was also observed that the models predicting less classes (Broadleaf, Douglas-Fir, Giant Sequoia, Grass, and Western Redcedar) outperformed models predicting on a larger set of tree classes (Bigleaf Maple, English Oak, Norway Maple, Douglas-Fir, Giant Sequoia, Grass, and Western Redcedar). This is likely a result of only having 4-band imagery plus the variables that are linear combinations of these 4 bands; 8 predictors training a model to predict 7 classes will not predict as well as 8 predictors training a model to predict 5 classes. In comparing the entire RF results on Portland to the `pdxTrees`, the test accuracy was 0.66.

4.3 Model Results Compared to Fricker Methods (Fricker et al., 2019)

Fricker's methods are conducted under ideal circumstances. The single image strip used in the study had a spatial resolution of 1 meter, they had access to both hyperspectral imagery and RGB imagery, and the ground data was carefully collected to consist of seven tree species (Fricker et al., 2019). Fricker's results show that models using hyperspectral images performed better than RGB image models because they provide additional information by including spectral band combinations not available from RGB images. For this research, the images only consist of 4 bands (red, green, blue, and infrared), are accessible online, and have a spatial resolution of 3 meters. Upon closer inspection of the point data from `pdxTrees` on the satellite images, the pixels are not all properly aligned with the tree location. This is especially apparent with smaller tree canopies, where the pixel directly below the tree point often does not even overlap with the tree. This negatively impacts the point pixel results of the analysis, however, the polygon pixel results involved data that was carefully extracted to ensure the tree pixels are representative of a tree canopy, so it performed better. Unfortunately, the inconsistency of the point pixel locations impacts the interpretation of the testing results since the estimates for Western Redcedar locations are compared against the locations of the `pdxTrees` Western Redcedar locations. The

resulting test accuracy is likely an underestimate. Fricker's methods applying a convolution neural network using hyperspectral imagery produced cross-validation training accuracies of 87%. On just RGB imagery, this accuracy dropped to 64%, which is a slight improvement over the best performing model (RF P = 7, C = 5) from this research which had a cross-validation training accuracy of 57%. Across Portland, the (lower bound) testing accuracy of the RF P = 7, C = 5 model was 66% for Western Redcedars.

4.4 Methods of Improvement/Further Work

As Fricker's article has shown, increasing the spectral resolution by having access to hyperspectral data adds more information to the models and improves performance. Hyperspectral imagery collects information from more regions of the spectrum than RGB imagery, which allows it to capture finer details. Tree and vegetation pixels reflect a certain amount of light, especially regions of dense forests, so having more bands can allow the models to detect subtle differences in vegetation pixels with non-vegetation pixels. By increasing the number of variables in the model, the risk of multicollinearity that comes from artificially creating predictors with linear combinations of the RGB bands also gets removed. The benefits of high spatial and spectral resolution imagery also extends to improving the masks applied to the images, which were successful, for example, in removing building structures and dead fields of grass, but incorrectly recognized mossy lakes as trees. Carefully masking the images improves the ability of the model to locate tree pixels and classify them.

In terms of data, another improvement to this research would be to increase the spatial resolution to 1 meter or better if possible. This refines the process of creating polygons by enhancing the visibility of the tree canopy outlines and reduces noise introduced by having shadows, dirt, building structures, and other non-tree pixels within the polygons. Another method to improve the results is by improving the drawn polygons themselves to only include pixels that represent the tree canopy. Drawing the polygons for each tree species by hand is likely to introduce error to the analysis because the first few polygons are not likely to be drawn the same way that the last few polygons are drawn. This issue is addressed by taking care to only include pure tree pixels (no shadows, dirt, or other structures) and ensuring that each polygon consists of at least 10 pixels.

An ideal training data set for the models would consist of data from imagery with high spatial and spectral resolution and also polygons with pure tree canopy pixels. If multiple raster strips are neccessary for the area of interest, the polygons need to be evenly distributed across all raster strips and ideally the tree species per polygon as well. The trees themselves should have large canopies that hopefully do not overlap with other tree canopies of different species.

With these improvements, the research will allow researchers to have a better understanding of the locations of redcedars in Portland. From there, this information provides a way to track future changes in redcedar trees's health, and inform decisions about tree management in the city. In a broader context, applying this method to

Portland imagery data with multiple strips provides information about the applicability of modelling tree locations in different parts of the world and with data from different satellites. Ultimately, tracking the changes of the ecosystem allows people to take steps to preserve dying species.

4.5 Conclusion

Satellite imagery is widely applied to spatial data applications in environmental fields, and a lot of research with remote sensing images has been published for classifying land type. Often this research is conducted under ideal circumstances with fine spatial and spectral resolution data, however, images at such a resolution usually require payment. This research applies imaging methods to satellite data with fewer bands and poorer resolution and investigates improvements to make up for the loss of resolution.

Reports of Western redcedars in the Pacific Northwest over the past 10 years have sparked an interest in investigating the decline of the species. This research combined RGB imagery and ground-level data available for the city of Portland to try and predict the locations of Western redcedars and progress towards understanding the decline of the species and eventually expand the methods to larger regions like the Pacific Northwest. Ground-level data was used to locate individual tree canopies and label their species on the satellite images, then the light intensities were extracted for each tree canopy pixel into a training data set. Random forest and support vector machine models were trained to classify 7 or 5 tree species. The best performing model was the random forest model with 7 predictors on 5 tree species with an F-score of 0.27 for Western redcedars. These results were not as high as other results from data with better conditions (higher spatial and spectral resolutions), however, they provide a better way of classifying a tree pixel as Western redcedar than a random guess of one of the 7 species (probability of $1/7 \approx 0.14$).

The results of this research are replicable, since it involves data from open sources and the coding is completed in RStudio, and not too computationally intensive to require more than a day to run all of the code. Another key takeaway is that this research applies the methods of using remote sensing imagery to classify tree species in Portland specifically and provides a guideline for general areas to focus on locating Western redcedars. The research is another application of remote sensing for tree species classification, which has the potential to predict more tree species in larger regions with more research.

Appendix A

Tables

This appendix includes the cross-validation confusion matrices for all of the models on the training data and the confusion matrices on the testing data.

A.1 A.2 A.3 A.4 A.5 A.6 A.7 A.8 A.9 A.10 A.11 A.12 A.13 A.14 A.15 A.16 A.17

Table A.1: RF Cross-Validation Confusion Matrix for P = 7, C = 7

	Bigleaf Maple	Douglas-Fir	English Oak	Giant Sequoia	grass	Norway Maple	Western Redcedar
Bigleaf Maple	6.3714	0.8286	1.4857	1.7429	0.0571	3.3714	1.8286
Douglas-Fir	0.9143	4.4571	1.8000	1.8571	0.0286	0.8857	2.2000
English Oak	1.0286	2.8000	5.3143	2.2857	0.0000	1.1143	2.4571
Giant Sequoia	1.4857	2.2000	2.1714	4.7429	0.0286	1.6000	1.2286
grass	0.9429	0.0000	0.0286	0.1429	13.8286	1.0286	0.3429
Norway Maple	2.4857	1.3714	0.9429	1.8571	0.2857	4.5429	1.6286
Western Redcedar	1.0571	2.6286	2.5429	1.6571	0.0571	1.7429	4.6000

Table A.2: RF Cross-Validation Confusion Matrix for P = 8, C = 7

	Bigleaf Maple	Douglas-Fir	English Oak	Giant Sequoia	grass	Norway Maple	Western Redcedar
Bigleaf Maple	6.4000	0.7143	1.6857	1.8571	0.1714	3.2000	1.6857
Douglas-Fir	0.8857	4.8000	1.8857	1.5143	0.0286	1.2571	1.9143
English Oak	0.9429	2.8571	5.4571	2.4000	0.0000	0.8857	2.5143
Giant Sequoia	1.5429	2.2571	2.0000	4.8286	0.0000	1.6571	1.4000
grass	0.8571	0.0000	0.0286	0.1429	13.6571	0.9143	0.3714
Norway Maple	2.5429	1.1714	0.7143	1.9714	0.3429	4.7714	1.7143
Western Redcedar	1.1143	2.4857	2.5143	1.5714	0.0857	1.6000	4.6857

Table A.3: RF Cross-Validation Confusion Matrix for P = 8, C = 5 (point)

	Bigleaf Maple	Douglas-Fir	Giant Sequoia	Norway Maple	Western Redcedar
Bigleaf Maple	2.6346	4.4674	4.6964	3.7801	5.0401
Douglas-Fir	4.8110	5.9565	3.4364	4.0092	2.9782
Giant Sequoia	4.9255	2.7491	3.2073	4.5819	2.1764
Norway Maple	2.9782	4.0092	3.5510	4.2383	3.4364
Western Redcedar	4.6964	2.8637	4.9255	3.4364	6.4147

Table A.4: RF Cross-Validation Confusion Matrix for P = 7, C = 5

	Broadleaf	Douglas-Fir	Giant Sequoia	grass	Western Redcedar
Broadleaf	33.2000	7.2000	8.7429	0.6000	9.0286
Douglas-Fir	2.0571	4.1429	1.1143	0.0286	1.1714
Giant Sequoia	2.8000	1.4571	3.3143	0.0000	0.6857
grass	1.8286	0.0000	0.1143	13.6286	0.2857
Western Redcedar	2.9714	1.4857	1.0000	0.0286	3.1143

Table A.5: RF Cross-Validation Confusion Matrix for P = 8, C = 5

	Broadleaf	Douglas-Fir	Giant Sequoia	grass	Western Redcedar
Broadleaf	32.2000	6.6571	8.0857	0.8571	8.6000
Douglas-Fir	2.3429	4.2857	1.4571	0.0286	1.3714
Giant Sequoia	3.3714	1.6000	3.5429	0.0000	0.8000
grass	1.6571	0.0000	0.1143	13.3714	0.2571
Western Redcedar	3.2857	1.7429	1.0857	0.0286	3.2571

Table A.6: RF Cross-Validation Confusion Matrix for P = 8, C = 4 (point)

	Broadleaf	Douglas-Fir	Giant Sequoia	Western Redcedar
Broadleaf	22.7950	13.6312	12.8293	12.3711
Douglas-Fir	5.1546	3.2073	1.8328	2.2910
Giant Sequoia	5.7274	1.4891	3.0928	1.9473
Western Redcedar	6.4147	1.7182	2.0619	3.4364

Table A.7: Linear SVM Cross-Validation Confusion Matrix C = 7

	Broadleaf	Douglas-Fir	Giant Sequoia	Western Redcedar
Broadleaf	40.0916	20.0458	19.8167	20.0458
Douglas-Fir	0.0000	0.0000	0.0000	0.0000
Giant Sequoia	0.0000	0.0000	0.0000	0.0000
Western Redcedar	0.0000	0.0000	0.0000	0.0000

Table A.8: Radial SVM Cross-Validation Confusion Matrix C = 7

	Bigleaf Maple	Douglas-Fir	English Oak	Giant Sequoia	grass	Norway Maple	Western Redcedar
Bigleaf Maple	7.1429	0.8000	1.6857	2.4286	0.0000	4.3429	2.2286
Douglas-Fir	0.5714	3.6000	0.4857	0.2286	0.0000	0.1143	0.7143
English Oak	0.6286	2.7714	5.0857	2.1429	0.0000	0.9143	2.0286
Giant Sequoia	1.5143	3.1143	3.1429	5.9714	0.0000	1.8286	1.4000
grass	1.3429	0.0000	0.0571	0.2286	14.0000	1.2571	0.4571
Norway Maple	2.2000	1.0571	0.7714	1.3429	0.2857	3.8286	1.7429
Western Redcedar	0.8857	2.9429	3.0571	1.9429	0.0000	2.0000	5.7143

Table A.9: Polynomial SVM Cross-Validation Confusion Matrix C = 7

	Bigleaf Maple	Douglas-Fir	English Oak	Giant Sequoia	grass	Norway Maple	Western Redcedar
Bigleaf Maple	8.2000	1.4857	2.1714	3.1714	0.0286	5.4000	2.9429
Douglas-Fir	0.4571	3.1714	0.4286	0.1143	0.0000	0.0857	0.4286
English Oak	0.6000	2.9429	4.6571	1.9429	0.0000	0.8286	2.2286
Giant Sequoia	1.2857	3.1429	3.3143	5.8857	0.0000	1.7714	1.3143
grass	1.0857	0.0000	0.0286	0.1429	14.0000	1.2286	0.4286
Norway Maple	1.7143	0.8000	0.6286	0.9429	0.2286	3.0857	1.4571
Western Redcedar	0.9429	2.7429	3.0571	2.0857	0.0286	1.8857	5.4857

Table A.10: Radial SVM Cross-Validation Confusion Matrix C = 5

	Broadleaf	Douglas-Fir	Giant Sequoia	grass	Western Redcedar
Broadleaf	37.7143	9.5143	11.4857	0.3714	11.8000
Douglas-Fir	0.7429	3.0571	0.1143	0.0000	0.4286
Giant Sequoia	1.0857	1.2571	2.3143	0.0000	0.1143
grass	2.1429	0.0000	0.1143	13.9143	0.3429
Western Redcedar	1.1714	0.4571	0.2571	0.0000	1.6000

Table A.11: Polynomial SVM Cross-Validation Confusion Matrix C = 5

	Broadleaf	Douglas-Fir	Giant Sequoia	grass	Western Redcedar
Broadleaf	39.2286	10.9143	12.4000	0.4286	13.5714
Douglas-Fir	0.7143	2.4571	0.0857	0.0000	0.2000
Giant Sequoia	0.5714	0.8857	1.6857	0.0000	0.1429
grass	2.1714	0.0000	0.1143	13.8571	0.3143
Western Redcedar	0.1714	0.0286	0.0000	0.0000	0.0571

Table A.12: Linear SVM Cross-Validation Confusion Matrix C = 5 (point)

	Bigleaf Maple	Douglas-Fir	Giant Sequoia	Norway Maple	Western Redcedar
Bigleaf Maple	1.4891	1.3746	1.6037	1.2600	2.4055
Douglas-Fir	4.5819	7.3310	5.0401	5.3837	4.2383
Giant Sequoia	10.6529	8.0183	10.3093	8.2474	9.2784
Norway Maple	1.3746	1.3746	1.0309	1.8328	1.9473
Western Redcedar	1.9473	1.9473	1.8328	3.3219	2.1764

Table A.13: Radial SVM Cross-Validation Confusion Matrix C = 5 (point)

	Bigleaf Maple	Douglas-Fir	Giant Sequoia	Norway Maple	Western Redcedar
Bigleaf Maple	1.2600	2.2910	1.7182	1.7182	1.6037
Douglas-Fir	5.3837	6.5292	4.1237	5.7274	4.1237
Giant Sequoia	5.8419	3.8946	6.1856	4.9255	5.6128
Norway Maple	2.1764	2.7491	3.2073	2.0619	3.4364
Western Redcedar	5.3837	4.5819	4.5819	5.6128	5.2692

Table A.14: Polynomial SVM Cross-Validation Confusion Matrix C = 5 (point)

	Bigleaf Maple	Douglas-Fir	Giant Sequoia	Norway Maple	Western Redcedar
Bigleaf Maple	2.0619	2.1764	2.8637	2.2910	2.6346
Douglas-Fir	5.1546	7.5601	4.8110	5.9565	4.6964
Giant Sequoia	9.3929	6.7583	9.3929	6.8729	8.4765
Norway Maple	0.8018	1.0309	0.6873	1.3746	0.8018
Western Redcedar	2.6346	2.5200	2.0619	3.5510	3.4364

Table A.15: Radial SVM Cross-Validation Confusion Matrix C = 4 (point)

	Broadleaf	Douglas-Fir	Giant Sequoia	Western Redcedar
Broadleaf	39.0607	18.7858	19.7022	19.8167
Douglas-Fir	0.9164	1.1455	0.1145	0.1145
Giant Sequoia	0.0000	0.0000	0.0000	0.0000
Western Redcedar	0.1145	0.1145	0.0000	0.1145

Table A.16: Polynomial SVM Cross-Validation Confusion Matrix C = 4 (point)

	Broadleaf	Douglas-Fir	Giant Sequoia	Western Redcedar
Broadleaf	39.8625	19.2440	19.7022	19.8167
Douglas-Fir	0.1145	0.8018	0.1145	0.2291
Giant Sequoia	0.1145	0.0000	0.0000	0.0000
Western Redcedar	0.0000	0.0000	0.0000	0.0000

Table A.17: RF Test Data Confusion Matrix for P = 7, C = 5

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: Broadleaf	0.78	0.68	0.57	0.85	0.57	0.78	0.66	0.35	0.27	0.48	0.73
Class: Douglas-Fir	0.32	0.96	0.52	0.92	0.52	0.32	0.40	0.11	0.04	0.07	0.64
Class: Giant Sequoia	0.23	0.96	0.45	0.90	0.45	0.23	0.31	0.12	0.03	0.06	0.60
Class: grass	0.95	0.96	0.92	0.98	0.92	0.95	0.93	0.31	0.29	0.32	0.95
Class: Western Redcedar	0.22	0.95	0.35	0.91	0.35	0.22	0.27	0.11	0.02	0.07	0.58

Table A.18: SVM Test Data Confusion Matrix C = 5

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: Broadleaf	0.87	0.58	0.53	0.90	0.53	0.87	0.66	0.35	0.31	0.58	0.73
Class: Douglas-Fir	0.22	0.99	0.73	0.91	0.73	0.22	0.33	0.11	0.02	0.03	0.60
Class: Giant Sequoia	0.15	0.98	0.50	0.90	0.50	0.15	0.23	0.12	0.02	0.04	0.57
Class: grass	0.96	0.96	0.91	0.98	0.91	0.96	0.94	0.31	0.30	0.33	0.96
Class: Western Redcedar	0.11	0.98	0.47	0.90	0.47	0.11	0.18	0.11	0.01	0.03	0.55

Appendix B

Code

This appendix includes the workflow and all the R code for replicating this research. First the rasters need to be downloaded from Planet.com by creating an account and selecting a date with clear weather and Portland's location. The following libraries and functions are necessary to be loaded into your workspace:

```
# libraries
library(caret)
library(doParallel)
library(gridExtra)
library(pdxTrees)
library(raster)
library(sf)
library(sp)
library(tidyverse)
library(tigris)

# write shapefile function
species_shapefile <- function(species, id){
  train <- pdxTrees_combined %>%
  filter(Species %in% species, Crown_Width_NS >= 20)
  coordinates(train) = ~ Longitude + Latitude
  proj4string(train)<- CRS("+proj=longlat +datum=WGS84")
  raster::shapefile(train, paste0(id, "_shapefile.shp"))
}

# extract values and tidy function
names <- c("ID", "red", "green", "blue", "ir")
pixels_extraction <- function(stack, poly_df, id) {
  val <- raster::extract(stack, poly_df, df = TRUE)
  colnames(val) <- names
  val <- drop_na(val) %>%
    mutate(rstrip = id)
```

```
  return(val)
}

# function to sample data
sampling_data <- function(dat, ratio, n_pixels) {
  dat_sampled <- dat %>%
  group_by(rstrip, Cmmn_Nm) %>%
  slice_sample(prop = ratio) %>%
  ungroup() %>%
  group_by(Cmmn_Nm) %>%
  slice_sample(n = n_pixels) # selects n pixels from each class
  return(dat_sampled)
}

# function to select columns from data
selecting_data <- function(dat) {
  dat_selected <- dat %>%
  dplyr::select(red, green, blue, ir, ndvi, Cmmn_Nm, red_blue, red_green,
                blue_green)
  return(dat_selected)
}

# function to create grouped data
grouping_data <- function(dat) {
  dat_grouped <- dat %>%
    mutate(Cmmn_Nm = case_when(
    Cmmn_Nm %in% c("Bigleaf Maple", "English Oak", "Norway Maple") ~ "Broadleaf",
    TRUE ~ as.character(Cmmn_Nm)
  ))
  return(dat_grouped)
}

# random forest function
training_rf <- function(dat) {
  model <- train(Cmmn_Nm~., data = dat, method = "rf", trControl = control)
  return(model)
}

# support vector machine function
training_svm <- function(dat, method) {
  model <- train(Cmmn_Nm ~., data = dat, method = method,
                 trControl = train_control,
              preProcess = c("center","scale"))
  return(model)
```

```
}

# prep rasters function
predRaster <- function(stack) {
  stack <- stack(stack, ((stack[[4]] - stack[[1]])/(stack[[4]] + stack[[1]])),
                 (stack[[1]]/stack[[3]]), (stack[[4]]/stack[[1]]),
                 (stack[[1]]/stack[[2]]), (stack[[3]]/stack[[2]]))
  names(stack) <- c("red", "green", "blue", "ir", "ndvi", "red_blue",
                    "ir_red", "red_green", "blue_green")
  cl <- makeCluster(3, type = "FORK")
  registerDoParallel(cl)
  results_RF <- raster::predict(stack, rf_small_grouped, progress = "text")
  results_SVM <- raster::predict(stack, svm2_grouped, progress = "text")
  results <- stack(results_RF, results_SVM)
  stopCluster(cl)
  return(results)
}
```

This code creates shapefiles for six tree species from `pdxTrees`:

```
# load data
pdxTrees_parks <- get_pdxTrees_parks()
pdxTrees_streets <- get_pdxTrees_streets() # site width in ft., diameter in in.
pdxTrees_parks_filtered <- pdxTrees_parks %>%
  select(Longitude, Latitude, Genus, Species, Common_Name, Crown_Width_NS,
         Crown_Width_EW, Crown_Base_Height, Condition)
pdxTrees_streets_filtered <- pdxTrees_streets %>%
  select(Longitude, Latitude, Genus, Species, Common_Name, Site_Width, Condition)

pdxTrees_combined <- full_join(pdxTrees_parks_filtered, pdxTrees_streets_filtered)

# species shapefiles
species_shapefile("QURO", "quro")
species_shapefile("SEGI", "segi")
species_shapefile("ACPL", "acpl")
species_shapefile("PSME", "psme")
species_shapefile("ACMA", "acma")

# Western redcedar shapefile
train_redcedar <- pdxTrees_combined %>%
  filter(Common_Name %in% "Western Redcedar", Crown_Width_NS >= 20)
coordinates(train_redcedar) = ~ Longitude + Latitude
proj4string(train_redcedar)<- CRS("+proj=longlat +datum=WGS84")
raster::shapefile(train_redcedar, "redcedar_shapefile.shp")
```

Once the tree species shapefiles are created, they are imported into QGIS for manual polygons to be drawn around 100 of each tree species. Since the Planet.com imagery has a spatial resolution of 3 meters, the drawn polygons should try to contain around 10-20 pixels. The polygon layer is imported into RStudio for further analysis. To create the pixels data, the polygons are used to indicate the pixels that should be extracted. The point pixels data comes from filtering `pdxTrees`. The following code performs this geographic join of data:

```
# load polygon shapefiles
acpl <- st_read("~/tree_canopy/acpl_polygon.shp")
psme <- st_read("~/tree_canopy/psme_polygon.shp")
quru <- st_read("~/tree_canopy/quru_polygon.shp")
redcedar <- st_read("~/tree_canopy/redwood_polygon.shp")
segi <- st_read("~/tree_canopy/segi_polygon.shp")
acma <- st_read("~/tree_canopy/acma_polygon.shp")
grass <- st_read("~/tree_canopy/grass_polygon.shp")

# add polygon tree type column
acpl$id_type <- "acpl"
psme$id_type <- "psme"
quru$id_type <- "quru"
redcedar$id_type <- "redcedar"
segi$id_type <- "segi"
acma$id_type <- "acma"
grass$id_type <- "grass"

# combine polygon shapefiles
combined_poly <- rbind(acpl, psme, quru, redcedar, segi, acma, grass)

# create multipolygon object
multi_poly <- st_multipolygon(combined_poly$geometry)

# load point datasets
acpl_pts <- st_read("~/tree_imaging/shapefiles/acpl_shapefile.shp")
psme_pts <- st_read("~/tree_imaging/shapefiles/psme_shapefile.shp")
quru_pts <- st_read("~/tree_imaging/shapefiles/quru_shapefile.shp")
redcedar_pts <- st_read("~/tree_imaging/shapefiles/redcedar_shapefile.shp")
segi_pts <- st_read("~/tree_imaging/shapefiles/segi_shapefile.shp")
acma_pts <- st_read("~/tree_imaging/shapefiles/acma_shapefile.shp")

# combine point datasets
combined_pts <- rbind(acpl_pts, psme_pts, quru_pts, redcedar_pts, segi_pts,
                      acma_pts)

# join combined points to combined polygons
```

```
poly_join <- st_join(combined_poly, combined_pts, largest = T)

# tidy polygon-point dataset
poly_join <- poly_join %>%
  mutate(id = c(1:nrow(poly_join)))

# load raster layer object
stack_a <- stack("planet/20200902_184428_1005_3B_AnalyticMS_clip.tif")
stack_b <- stack("planet/20200902_184429_1005_3B_AnalyticMS_clip.tif")
stack_c <- stack("planet/20200902_183008_24_1065_3B_AnalyticMS_clip.tif")
stack_d <- stack("planet/20200902_183006_18_1065_3B_AnalyticMS_clip.tif")
stack_e <- stack("planet/20200902_184430_1005_3B_AnalyticMS_clip.tif")

# match raster project to polygons
raster_crs <- CRS(projection(stack_a))
poly_join_reprojected <- spTransform(as_Spatial(poly_join), raster_crs)
# saveRDS(poly_join_reprojected, "data/poly_join_reprojected.rds")

# extract pixel values from polygons into a dataframe
val_a <- pixels_extraction(stack_a, poly_join_reprojected, "a")
val_b <- pixels_extraction(stack_b, poly_join_reprojected, "b")
val_c <- pixels_extraction(stack_c, poly_join_reprojected, "c")
val_d <- pixels_extraction(stack_d, poly_join_reprojected, "d")
val_e <- pixels_extraction(stack_e, poly_join_reprojected, "e")

# semi_join returns rows of x where it can find a match in y
val_combined <- rbind(val_a, anti_join(val_b, val_a, by = "ID"))
val_combined <- rbind(val_combined, anti_join(val_c, val_combined, by = "ID"))
val_combined <- rbind(val_combined, anti_join(val_d, val_combined, by = "ID"))
val_combined <- rbind(val_combined, anti_join(val_e, val_combined, by = "ID"))

# tidy pixel table
pixels_data <-left_join(val_combined, poly_join_reprojected %>%
                          rename(ID = id)) %>%
  mutate(ndvi = (ir - red)/(ir + red),
         Cmmn_Nm = replace_na(as.character(Cmmn_Nm), "grass"),
         red_blue = red/blue,
         # ir_red = ir/red,
         red_green = red/green,
         blue_green = blue/green)

# load single pixel tree points
pdxTrees_parks <- get_pdxTrees_parks()
pdxTrees_streets <- get_pdxTrees_streets()
```

```
pdxTrees_combined <- full_join(pdxTrees_parks, pdxTrees_streets)
trees_dat <- pdxTrees_combined %>%
  dplyr::filter(Common_Name %in% "Western Redcedar" |
                  Species %in% unique(pixels_data$Species),
                DBH >= 15)


# convert to spatial points and reproject
coordinates(trees_dat) <- ~ Longitude + Latitude
proj4string(trees_dat)<- CRS("+proj=longlat +datum=WGS84")
trees_reprojected <- spTransform(trees_dat, raster_crs)

# extract pixels for all trees
pix_a <- pixels_extraction(stack_a, trees_reprojected, "a")
pix_b <- pixels_extraction(stack_b, trees_reprojected, "b")
pix_c <- pixels_extraction(stack_c, trees_reprojected, "c")
pix_d <- pixels_extraction(stack_d, trees_reprojected, "d")
pix_e <- pixels_extraction(stack_e, trees_reprojected, "e")

# semi_join returns rows of x where it can find a match in y
pix_combined <- rbind(pix_a, anti_join(pix_b, pix_a, by = "ID"))
pix_combined <- rbind(pix_combined, anti_join(pix_c, pix_combined, by = "ID"))
pix_combined <- rbind(pix_combined, anti_join(pix_d, pix_combined, by = "ID"))
pix_combined <- rbind(pix_combined, anti_join(pix_e, pix_combined, by = "ID"))

# tidy centers pixel table
pix_tree_center <- pix_combined %>%
  mutate(ID = as.character(ID)) %>%
  left_join(trees_reprojected@data, by = c("ID" = "UserID")) %>%
  mutate(ndvi = (ir - red)/(ir + red),
         red_blue = red/blue,
         # ir_red = ir/red,
         red_green = red/green,
         blue_green = blue/green,
         Cmmn_Nm = case_when(
           Common_Name %in% "Maple, Norway" ~ "Norway Maple",
           Common_Name %in% "Maple, Bigleaf" ~ "Bigleaf Maple",
           TRUE ~ as.character(Common_Name)
         )) %>%
  dplyr::filter(Cmmn_Nm %in% c("Western Redcedar", "Bigleaf Maple",
                               "Douglas-Fir", #"English Oak",
                               "Giant Sequoia", "Norway Maple"))

# save results
```

```
# write.csv(poly_join_reprojected,'data/poly_join_reprojected1.csv', row.names =
# write.csv(val_combined,'data/val_combined1.csv', row.names = F)
# write.csv(pixels_data,'data/pixels_data.csv', row.names = F)
# write.csv(pix_tree_center,'data/pix_tree_center.csv', row.names = F)
```

To prepare the data for training and testing the models, the pixels and point pixels data are split into training and testing data:

```
# load data
pixels_data <- read.csv('data/pixels_data.csv')
pix_tree_center <- read.csv('data/pix_tree_center.csv')

# take note of distribution of classes
pixels_data %>%
  count(Cmmn_Nm)
pix_tree_center %>%
  count(Cmmn_Nm)

# training data
set.seed(2)
train_full <- sampling_data(pixels_data, 0.7, 500)
train <- selecting_data(train_full)
train_grouped <- grouping_data(train)
train_center <- selecting_data(sampling_data(pix_tree_center, 0.7, 175))
train_center_grouped <- grouping_data(train_center)

# testing data
test_full <- anti_join(pixels_data, train_full)
test_full_grouped <- grouping_data(test_full)
test <- test_full %>%
  dplyr::select(red, green, blue, ir, ndvi, Cmmn_Nm, red_blue, red_green,
                blue_green)
test_grouped <- grouping_data(test)
test_center <- anti_join(pix_tree_center, train_center) %>%
  dplyr::select(red, green, blue, ir, ndvi, Cmmn_Nm, red_blue, red_green,
                blue_green)
test_center_grouped <- grouping_data(test_center)

# save results
# write.csv(train,'data/train.csv', row.names = F)
# write.csv(train_grouped,'data/train_grouped.csv', row.names = F)
# write.csv(train_center,'data/train_center.csv', row.names = F)
# write.csv(train_center_grouped,'data/train_center_grouped.csv', row.names = F)
# write.csv(test,'data/test.csv', row.names = F)
```

```
# write.csv(test_grouped,'data/test_grouped.csv', row.names = F)
# write.csv(test_center,'data/test_center.csv', row.names = F)
# write.csv(test_center_grouped,'data/test_center_grouped.csv', row.names = F)
# write.csv(test_full, 'data/test_full.csv', row.names = F)
# write.csv(test_full_grouped, 'data/test_full_grouped.csv', row.names = F)
```

The training data is used to train the models:

```
# load training data
train <- read.csv('data/train.csv')
train_grouped <- read.csv('data/train_grouped.csv')
train_center <- read.csv('data/train_center.csv')
train_center_grouped <- read.csv('data/train_center_grouped.csv')

# random search random forest using caret package
control <- trainControl(method = "repeatedcv", number = 10,
                        search = "random")

set.seed(2)
rf_small <- training_rf(train %>% dplyr::select(-c(red_blue)))
rf_small_grouped <- training_rf(train_grouped %>% dplyr::select(-c(red_blue)))
rf_full <- training_rf(train)
rf_full_grouped <- training_rf(train_grouped)
rf_center <- training_rf(train_center)
rf_center_grouped <- training_rf(train_center_grouped)

# 10 fold cross validation
train_control <- trainControl(method = "cv", number = 10)

set.seed(2)
# fit svm model with normalized variables
svm1 <- training_svm(train, "svmLinear")
svm2 <- training_svm(train, "svmRadial")
svm3 <- training_svm(train, "svmPoly")
svm1_grouped <- training_svm(train_grouped, "svmLinear")
svm2_grouped <- training_svm(train_grouped, "svmRadial")
svm3_grouped <- training_svm(train_grouped, "svmPoly")
svm1_center <- training_svm(train_center, "svmLinear")
svm2_center <- training_svm(train_center, "svmRadial")
svm3_center <- training_svm(train_center, "svmPoly")
svm1_center_grouped <- training_svm(train_center_grouped, "svmLinear")
svm2_center_grouped <- training_svm(train_center_grouped, "svmRadial")
svm3_center_grouped <- training_svm(train_center_grouped, "svmPoly")
```

```
# save models
# saveRDS(rf_small, "data/rf_small.rds")
# saveRDS(rf_small_grouped, "data/rf_small_grouped.rds")
# saveRDS(rf_full, "data/rf_full.rds")
# saveRDS(rf_full_grouped, "data/rf_full_grouped.rds")
# saveRDS(rf_center, "data/rf_center.rds")
# saveRDS(rf_center_grouped, "data/rf_center_grouped.rds")
# saveRDS(svm1, "data/svm1.rds")
# saveRDS(svm2, "data/svm2.rds")
# saveRDS(svm3, "data/svm3.rds")
# saveRDS(svm1_center, "data/svm1_center.rds")
# saveRDS(svm2_center, "data/svm2_center.rds")
# saveRDS(svm3_center, "data/svm3_center.rds")
# saveRDS(svm1_grouped, "data/svm1_grouped.rds")
# saveRDS(svm2_grouped, "data/svm2_grouped.rds")
# saveRDS(svm3_grouped, "data/svm3_grouped.rds")
```

Analysis on the trained models included in this research is conducted with the following code:

```
# load rf models
rf_small <- readRDS("data/rf_small.rds")
rf_small_grouped <- readRDS("data/rf_small_grouped.rds")
rf_full <- readRDS("data/rf_full.rds")
rf_full_grouped <- readRDS("data/rf_full_grouped.rds")
rf_center <- readRDS("data/rf_center.rds")
rf_center_grouped <- readRDS("data/rf_center_grouped.rds")

# rf mtry cross validation
grid.arrange(plot(rf_small, main = "RF P = 7, C = 7"),
             plot(rf_small_grouped, main = "RF P = 7, C = 5"), nrow = 1)
grid.arrange(plot(rf_full, main = "RF P = 8, C = 7"),
             plot(rf_full_grouped, main = "RF P = 8, C = 5"), nrow = 1)
grid.arrange(plot(rf_center, main = "RF P = 8, C = 5 (point)"),
             plot(rf_center_grouped, main = "RF P = 8, C = 4 (point)"),
             nrow = 1)

# data frame to compare rf models
results <- rbind(cbind(rf_small$resample, Model = rep("P = 7, C = 7", 10)),
                 cbind(rf_full$resample, Model = rep("P = 8, C = 7", 10)),
                 cbind(rf_center$resample,
                       Model = rep("P = 8, C = 5 (point)", 10)),
                 cbind(rf_small_grouped$resample,
                       Model = rep("P = 7, C = 5", 10)),
                 cbind(rf_full_grouped$resample,
```

```
                      Model = rep("P = 8, C = 5", 10)),
                cbind(rf_center_grouped$resample,
                      Model = rep("P = 8, C = 4 (point)", 10)))

# visualize rf model results
ggplot(results, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_boxplot(alpha = 0.6) +
  theme_minimal() +
  scale_fill_viridis_d() +
  labs(title = "Model Accuracy of Random Forest Models with 10-Fold CV",
       x = " ", fill = "P = Predictors, C = Classes") +
  theme(axis.text.x = element_blank())

# rf results by class
cm_small_rf <- confusionMatrix(rf_small,  mode = "prec_recall")
cm_full_rf <- confusionMatrix(rf_full,  mode = "prec_recall")
cm_center_rf <- confusionMatrix(rf_center,  mode = "prec_recall")
cm_small_grouped_rf <- confusionMatrix(rf_small_grouped,  mode = "prec_recall")
cm_full_grouped_rf <- confusionMatrix(rf_full_grouped,  mode = "prec_recall")
cm_center_grouped_rf <- confusionMatrix(rf_center_grouped,  mode = "prec_recall")

# save confusion matrices
# saveRDS(cm_small_rf, "data/cm_small_rf.rds")
# saveRDS(cm_full_rf, "data/cm_full_rf.rds")
# saveRDS(cm_center_rf, "data/cm_center_rf.rds")
# saveRDS(cm_small_grouped_rf, "data/cm_small_grouped_rf.rds")
# saveRDS(cm_full_grouped_rf, "data/cm_full_grouped_rf.rds")
# saveRDS(cm_center_grouped_rf, "data/cm_center_grouped_rf.rds")

# table comparing models
results_rf <- rbind(cbind(rf_small$resample, Model = rep("P = 7, C = 7", 10),
                          mtry = rep(rf_small$bestTune[1,1], 10)),
                    cbind(rf_full$resample, Model = rep("P = 8, C = 7", 10),
                          mtry = rep(rf_full$bestTune[1,1], 10)),
                    cbind(rf_center$resample,
                          Model = rep("P = 8, C = 5 (point)", 10),
                          mtry = rep(rf_center$bestTune[1,1], 10)),
                    cbind(rf_small_grouped$resample,
                          Model = rep("P = 7, C = 5", 10),
                          mtry = rep(rf_small_grouped$bestTune[1,1], 10)),
                    cbind(rf_full_grouped$resample,
                          Model = rep("P = 8, C = 5", 10),
                          mtry = rep(rf_full_grouped$bestTune[1,1], 10)),
                    cbind(rf_center_grouped$resample,
```

```
                          Model = rep("P = 8, C = 4 (point)", 10),
                          mtry = rep(rf_center_grouped$bestTune[1,1], 10)))

results_r <- results_rf %>%
  group_by(Model) %>%
  summarise("Accuracy" = round(mean(Accuracy), 4),
            "Kappa" = round(mean(Kappa), 4),
            "mtry" = mean(mtry)) %>%
  rename("Random Forest Model" = Model)

# save table
# write.csv(results_r,'data/results_r.csv', row.names = F)

# load svm models
svm1 <- readRDS("data/svm1.rds")
svm2 <- readRDS("data/svm2.rds")
svm3 <- readRDS("data/svm3.rds")
svm1_center <- readRDS("data/svm1_center.rds")
svm2_center <- readRDS("data/svm2_center.rds")
svm3_center <- readRDS("data/svm3_center.rds")
svm1_grouped <- readRDS("data/svm1_grouped.rds")
svm2_grouped <- readRDS("data/svm2_grouped.rds")
svm3_grouped <- readRDS("data/svm3_grouped.rds")

# collect resamples
results_svm <- rbind(cbind(svm1$resample, Model = rep("Linear, C = 7", 10, 10),
                           Parameter = paste("C =", svm1$bestTune[1,1])),
                     cbind(svm2$resample, Model = rep("Radial, C = 7", 10, 10),
                           Parameter = paste("C =", svm2$bestTune[1,2], "sigma =",
                                 round(svm2$bestTune[1,1], 4))),
                     cbind(svm3$resample, Model = rep("Poly, C = 7", 10, 10),
                           Parameter = paste("C =", svm3$bestTune[1,3], ", scale =
                                 svm3$bestTune[1,2], ", degree =", svm3$bestTune[1
                     cbind(svm1_center$resample,
                           Model = rep("Linear, C = 5 (point)", 10, 10),
                           Parameter = paste("C =", svm1_center$bestTune[1,1])),
                     cbind(svm2_center$resample,
                           Model = rep("Radial, C = 5 (point)", 10, 10),
                           Parameter = paste("C =", svm2_center$bestTune[1,2],
                                             "sigma =",
                                 round(svm2_center$bestTune[1,1], 4))),
                     cbind(svm3_center$resample,
                           Model = rep("Poly, C = 5 (point)", 10, 10),
                           Parameter = paste("C =", svm3_center$bestTune[1,3],
```

```
                                        ", scale =",
                                svm3_center$bestTune[1,2],
                                ", degree =", svm3_center$bestTune[1,1])),
                    cbind(svm1_grouped$resample,
                          Model = rep("Linear, C = 5", 10, 10),
                          Parameter = paste("C =", svm1_grouped$bestTune[1,1])),
                    cbind(svm2_grouped$resample,
                          Model = rep("Radial, C = 5", 10, 10),
                          Parameter = paste("C =", svm2_grouped$bestTune[1,2],
                                            "sigma =",
                                            round(svm2_grouped$bestTune[1,1], 4))),
                    cbind(svm3_grouped$resample,
                          Model = rep("Poly, C = 5", 10, 10),
                          Parameter = paste("C =", svm3_grouped$bestTune[1,3],
                                            ", scale =",
                                svm3_grouped$bestTune[1,2],
                                ", degree =", svm3_grouped$bestTune[1,1])),
                    cbind(svm1_center_grouped$resample,
                          Model = rep("Linear, C = 4 (point)", 10, 10),
                          Parameter = paste("C =", svm1_center_grouped$bestTune[1,1])),
                    cbind(svm2_center_grouped$resample,
                          Model = rep("Radial, C = 4 (point)", 10, 10),
                          Parameter = paste("C =", svm2_center_grouped$bestTune[1,2],
                                            "sigma =",
                                            round(svm2_center_grouped$bestTune[1,1], 4)
                    cbind(svm3_center_grouped$resample,
                          Model = rep("Poly, C = 4 (point)", 10, 10),
                          Parameter = paste("C =", svm3_center_grouped$bestTune[1,3],
                                            ", scale =",
                                svm3_center_grouped$bestTune[1,2],
                                ", degree =", svm3_center_grouped$bestTune[1,1])))

# visualize svm model results
ggplot(results_svm, aes(x = Model, y = Accuracy, fill = Model)) +
  geom_boxplot(alpha = 0.8) +
  theme_minimal() +
  scale_fill_discrete() +
  labs(title = "Model Accuracy of SVM Models with 10-Fold CV", x = " ", fill = " ") +
  theme(axis.text.x = element_blank())

grid.arrange(plot(svm2, main = "SVM C = 7"),
             plot(svm2_grouped, main = "SVM C = 5"), nrow = 1)
grid.arrange(plot(svm3, main = "SVM C = 7"),
             plot(svm3_grouped, main = "SVM C = 5"), nrow = 1)
```

```r
grid.arrange(plot(svm2_center, main = "SVM C = 5 (point)"),
             plot(svm2_center_grouped, main = "SVM C = 4 (point)"), nrow = 1)
grid.arrange(plot(svm3_center, main = "SVM C = 5 (point)"),
             plot(svm3_center_grouped, main = "SVM C = 4 (point)"), nrow = 1)

# sum results by class
scm1 <- confusionMatrix(svm1,  mode = "prec_recall")
scm2 <- confusionMatrix(svm2,  mode = "prec_recall")
scm3 <- confusionMatrix(svm3,  mode = "prec_recall")
scm1_center <- confusionMatrix(svm1_center,  mode = "prec_recall")
scm2_center <- confusionMatrix(svm2_center,  mode = "prec_recall")
scm3_center <- confusionMatrix(svm3_center,  mode = "prec_recall")
scm1_grouped <- confusionMatrix(svm1_grouped,  mode = "prec_recall")
scm2_grouped <- confusionMatrix(svm2_grouped,  mode = "prec_recall")
scm3_grouped <- confusionMatrix(svm3_grouped,  mode = "prec_recall")
scm1_center_grouped <- confusionMatrix(svm1_center_grouped,  mode = "prec_recall")
scm2_center_grouped <- confusionMatrix(svm2_center_grouped,  mode = "prec_recall")
scm3_center_grouped <- confusionMatrix(svm3_center_grouped,  mode = "prec_recall")

# save confusion matrices
# saveRDS(scm1, "data/scm1.rds")
# saveRDS(scm2, "data/scm2.rds")
# saveRDS(scm3, "data/scm3.rds")
# saveRDS(scm1_center, "data/scm1_center.rds")
# saveRDS(scm2_center, "data/scm2_center.rds")
# saveRDS(scm3_center, "data/scm3_center.rds")
# saveRDS(scm1_grouped, "data/scm1.rds")
# saveRDS(scm2_grouped, "data/scm2_grouped.rds")
# saveRDS(scm3_grouped, "data/scm3_grouped.rds")
# saveRDS(scm1_center_grouped, "data/scm1.rds")
# saveRDS(scm2_center_grouped, "data/scm2_center_grouped.rds")
# saveRDS(scm3_center_grouped, "data/scm3_center_grouped.rds")

# table comparing models
results_s <- results_svm %>%
  group_by(Model) %>%
  summarise("Accuracy" = round(mean(Accuracy), 4),
            "Kappa" = round(mean(Kappa), 4)) %>%
  rename("SVM Model Kernel" = Model)

# save table
# write.csv(results_s,'data/results_s.csv', row.names = F)

# load test data
```

```
test <- read.csv('data/test.csv')
test_grouped <- read.csv('data/test_grouped.csv')
test_center <- read.csv('data/test_center.csv')
test_center_grouped <- read.csv('data/test_center_grouped.csv')
test_full <- read.csv('data/test_full.csv')
test_full_grouped <- read.csv('data/test_full_grouped.csv')

# test set results for overall prediction
test_results <- data.frame(Class = test_grouped$Cmmn_Nm)
test_results$RF <- predict(rf_small_grouped, test_grouped)
test_results$SVM <- predict(svm2_grouped, test_grouped)

# test results accuracy table
test_results %>%
  summarise("RF Accuracy" = sum(Class == RF)/nrow(test_results),
            "SVM Accuracy" = sum(Class == SVM)/nrow(test_results))

# Western Redcedar test results
test_results_redceder <- test_results %>%
  dplyr::filter(Class %in% "Western Redcedar")

test_results_redceder %>%
  summarise(rf_accuracy = sum(Class == RF)/nrow(test_results_redceder),
            svm_accuracy = sum(Class == SVM)/nrow(test_results_redceder))

# confusion matrix with precision/recall
cm_test_rf <- confusionMatrix(test_results$RF, test_results$Class)
cm_test_svm <- confusionMatrix(test_results$SVM, test_results$Class)

# save matrices
# saveRDS(cm_test_rf, "data/cm_test_rf.rds")
# saveRDS(cm_test_svm, "data/cm_test_svm.rds")

# save results
# write.csv(test_results,'data/test_results.csv', row.names = F)

# join test results to full test data (with polygon info)
test_dat_rf <- cbind(test_full_grouped, RF = test_results$RF)
test_dat_svm <- cbind(test_full_grouped, SVM = test_results$SVM)

# rf polygon results for 5 class prediction
poly_test_rf <- test_dat_rf %>%
  group_by(ID) %>%
  count(ID, same_rf = (RF %in% Cmmn_Nm), total = n()) %>%
```

```
  pivot_wider(names_from = same_rf, values_from = n)

poly_test_rf[is.na(poly_test_rf)] <- 0 # replace na with 0

poly_test_rf <- poly_test_rf %>%
  mutate(Result = case_when(`TRUE` > `FALSE` ~ "Correct",
                            `TRUE` <= `FALSE` ~ "Incorrect"))

poly_test_rf %>%
  group_by(Result) %>%
  count(Result)

# sum polygon results for 7 class prediction
poly_test_svm <- test_dat_svm %>%
  group_by(ID) %>%
  count(ID, same_svm = (SVM == Cmmn_Nm), total = n()) %>%
  pivot_wider(names_from = same_svm, values_from = n)

poly_test_svm[is.na(poly_test_svm)] <- 0 # replace na with 0

poly_test_svm <- poly_test_svm %>%
  mutate(Result = case_when(`TRUE` > `FALSE` ~ "Correct",
                            `TRUE` <= `FALSE` ~ "Incorrect"))

poly_test_svm %>%
  group_by(Result) %>%
  count(Result)

# save poly test results
# write.csv(poly_test_rf,'data/poly_test_rf.csv', row.names = F)
# write.csv(poly_test_svm,'data/poly_test_svm.csv', row.names = F)
```

The rasters are prepped for the Portland model by masking NDVI values with the code:

```
# load city boundary shapefile
or_cities <- places("OR")
portland <- or_cities %>%
  filter(NAME == "Portland")

# load raster layer object
stack_a <- stack("planet/20200902_184428_1005_3B_AnalyticMS_clip.tif")
stack_b <- stack("planet/20200902_184429_1005_3B_AnalyticMS_clip.tif")
stack_c <- stack("planet/20200902_183008_24_1065_3B_AnalyticMS_clip.tif")
```

```
stack_d <- stack("planet/20200902_183006_18_1065_3B_AnalyticMS_clip.tif")
stack_e <- stack("planet/20200902_184430_1005_3B_AnalyticMS_clip.tif")

## crop and mask pixels outside portland
stack_a <- crop(mask(stack_a, spTransform(portland, crs(stack_a))),
                extent(portland))
stack_b <- crop(mask(stack_b, spTransform(portland, crs(stack_b))),
                extent(portland))
stack_c <- crop(mask(stack_c, spTransform(portland, crs(stack_c))),
                extent(portland))
stack_d <- crop(mask(stack_d, spTransform(portland, crs(stack_d))),
                extent(portland))
stack_e <- crop(mask(stack_e, spTransform(portland, crs(stack_e))),
                extent(portland))

# ndvi mask remove values below 0
ndvi_a <- (stack_a[[4]] - stack_a[[1]]) / (stack_a[[4]] + stack_a[[1]])
ndvi_b <- (stack_b[[4]] - stack_b[[1]]) / (stack_b[[4]] + stack_b[[1]])
ndvi_c <- (stack_c[[4]] - stack_c[[1]]) / (stack_c[[4]] + stack_c[[1]])
ndvi_d <- (stack_d[[4]] - stack_d[[1]]) / (stack_d[[4]] + stack_d[[1]])
ndvi_e <- (stack_e[[4]] - stack_e[[1]]) / (stack_e[[4]] + stack_e[[1]])

plot(ndvi_a, main = "NDVI", axes = FALSE, box = FALSE)

# masked ndvi values
ndvi_a[ndvi_a < 0] <- NA
ndvi_b[ndvi_b < 0] <- NA
ndvi_c[ndvi_c < 0] <- NA
ndvi_d[ndvi_d < 0] <- NA
ndvi_e[ndvi_e < 0] <- NA

ndvi_mask_a <- mask(stack_a, ndvi_a, filename = "data/ndvi_mask_a.tif",
                    overwrite = T)
ndvi_mask_b <- mask(stack_b, ndvi_b, filename = "data/ndvi_mask_b.tif",
                    overwrite = T)
ndvi_mask_c <- mask(stack_c, ndvi_c, filename = "data/ndvi_mask_c.tif",
                    overwrite = T)
ndvi_mask_d <- mask(stack_d, ndvi_d, filename = "data/ndvi_mask_d.tif",
                    overwrite = T)
ndvi_mask_e <- mask(stack_e, ndvi_e, filename = "data/ndvi_mask_e.tif",
                    overwrite = T)
```

Finally the model is applied to the mask images of Portland and visualized with the following code:

```r
# load masked raster images
stack_a <- stack("data/ndvi_mask_a.tif")
stack_b <- stack("data/ndvi_mask_b.tif")
stack_c <- stack("data/ndvi_mask_c.tif")
stack_d <- stack("data/ndvi_mask_d.tif")
stack_e <- stack("data/ndvi_mask_e.tif")

# load trained models
rf_small_grouped <- readRDS("data/rf_small_grouped.rds")
svm2_grouped <- readRDS("data/svm2_grouped.rds")

results_a <- predRaster(stack_a) # 41min
results_b <- predRaster(stack_b)
results_c <- predRaster(stack_c)
results_d <- predRaster(stack_d)
results_e <- predRaster(stack_e)

# save results
# writeRaster(results_a,"data/results_a.grd", format = "raster", overwrite = T)
# writeRaster(results_b,"data/results_b.grd", format = "raster", overwrite = T)
# writeRaster(results_c,"data/results_c.grd", format = "raster", overwrite = T)
# writeRaster(results_d,"data/results_d.grd", format = "raster", overwrite = T)
# writeRaster(results_e,"data/results_e.grd", format = "raster", overwrite = T)

# load results
results_a <- stack("data/results_a.grd")
results_b <- stack("data/results_b.grd")
results_c <- stack("data/results_c.grd")
results_d <- stack("data/results_d.grd")
results_e <- stack("data/results_e.grd")

# merge rasters into one for plotting
r_list_RF <- list(results_a[[1]] == 7, results_b[[1]] == 7, results_c[[1]] == 7,
                  results_d[[1]] == 7, results_e[[1]] == 7)
r_list_SVM <- list(results_a[[2]] == 7, results_b[[2]] == 7, results_c[[2]] == 7,
                  results_d[[2]] == 7, results_e[[2]] == 7)
m_RF <- do.call(merge, r_list_RF)
m_SVM <- do.call(merge, r_list_SVM)

# compare rasters
r_matches <- m_RF == m_SVM
freq(r_matches)
freq(r_matches)[2, ]/(freq(r_matches)[2, ] + freq(r_matches)[1, ])
# (r_matches == 1)/((r_matches == 0) + (r_matches == 1)) percentage in common: 8
```

```
plot(r_matches == 1, main = "RF C = 5 and SVM C = 5 Predictions Comparisons",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "lightgreen")
plot(r_matches == 0, col = "maroon", labels = F, xaxt = 'n', yaxt = 'n',
     legend = F, add = T)

# load city outline
or_cities <- places("OR")
portland <- or_cities %>%
  filter(NAME == "Portland")
portland_wgs84 <- spTransform(portland,
                              CRS("+proj=utm +zone=10 +ellps=WGS84 +datum=WGS84 +units=m

# plot predictions
plot(m_RF, main = "RF Western Redcedar Predictions",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "darkgreen")
plot(portland_wgs84, add = T)

plot(m_SVM, main = "SVM Western Redcedar Predictions",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "darkgreen")
plot(portland_wgs84, add = T)

# load street trees western redcedars
pdxTrees_parks <- get_pdxTrees_parks()
pdxTrees_streets <- get_pdxTrees_streets()

redcedar_dat_street <- pdxTrees_streets %>%
  filter(Common_Name %in% "Western Redcedar") %>%
  dplyr::select(Longitude, Latitude, Common_Name)
redcedar_dat_park <- pdxTrees_parks %>%
  filter(Common_Name %in% "Western Redcedar") %>%
  dplyr::select(Longitude, Latitude, Common_Name)
redcedar_dat <- rbind(redcedar_dat_street, redcedar_dat_park)

# match coordinate reference system
coordinates(redcedar_dat) <- c("Longitude", "Latitude")
proj4string(redcedar_dat) <- CRS("+init=epsg:4326")
dat_transformed <- spTransform(redcedar_dat, CRS("+proj=utm +zone=10 +ellps=WGS84 +datum

# plot pdxTrees with predictions
p1 <- plot(m_RF, main = "RF C = 5 with pdxTrees",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "darkgreen")
plot(dat_transformed, col = rgb(red = 0.2, green = 0.2, blue = 1.0, alpha = 0.2),
     cex = .2, add = T)
plot(portland_wgs84, add = T)
```

```
plot(m_SVM, main = "SVM C = 5 with pdxTrees",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "darkgreen")
plot(dat_transformed, col = rgb(red = 0.2, green = 0.2, blue = 1.0, alpha = 0.2),
     cex = .2, add = T)
plot(portland_wgs84, add = T)

# match pixels with pdxTrees to test for accuracy
street_mask <- mask(m_RF, dat_transformed)
street_mask_group <- mask(m_SVM, dat_transformed)
freq(street_mask)[1, 2]/nrow(dat_transformed) # estimates for Western redcedars
# 52%
freq(street_mask_group)[1, 2]/nrow(dat_transformed) # estimates for Western redce
# 66%

# smaller visualization around Reed
reed_stack <- crop(m_RF, extent(525000, 530000, 5035000, 5040000))
trees_reed <- crop(dat_transformed, extent(525000, 530000, 5035000, 5040000))

# Portland with extent box
plot(m_RF, main = "RF C = 7 with pdxTrees",
     labels = F, xaxt = 'n', yaxt = 'n', legend = F, col = "darkgreen")
plot(dat_transformed, col = rgb(red = 0.2, green = 0.2, blue = 1.0, alpha = 0.6),
     cex = .2, add = T, pch = 1)
plot(portland_wgs84, add = T)
plot(extent(reed_stack), add = T, col = "maroon")

# Reed plot
plot(reed_stack, main = "", labels = F, xaxt = 'n', yaxt = 'n', legend = F,
     col = "darkgreen")
plot(trees_reed, col = rgb(red = 0.2, green = 0.2, blue = 1.0, alpha = 1),
     cex = .2, add = T, pch = 1)
plot(extent(reed_stack), add = T, col = "maroon")
```

References

Castelluccio, M., Poggi, G., Sansone, C., & Verdoliva, L. (2015). Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. *arXiv:1508.00092 [Cs]*. Retrieved from http://arxiv.org/abs/1508.00092

Fricker, G. A., Ventura, J. D., Wolf, J. A., North, M. P., Davis, F. W., & Franklin, J. (2019). A Convolutional Neural Network Classifier Identifies Tree Species in Mixed-Conifer Forest from Hyperspectral Imagery. *Remote Sensing*, *11*(19), 2326. http://doi.org/10.3390/rs11192326

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 103). New York, NY: Springer New York. http://doi.org/10.1007/978-1-4614-7138-7

Kuhn, M. (2020). *Caret: Classification and regression training*. Retrieved from https://CRAN.R-project.org/package=caret

Peterson, J. S. (n.d.). WESTERN RED CEDAR, 3.

Qian, Y., Zhou, W., Nytch, C. J., Han, L., & Li, Z. (2020). A new index to differentiate tree and grass based on high resolution image and object-based methods. *Urban Forestry & Urban Greening*, *53*, 126661. http://doi.org/10.1016/j.ufug.2020.126661

Western red cedar. (n.d.). Retrieved from https://www.oregonencyclopedia.org/articles/western_red_cedar/#.YB7fUy1h1N0

Western redcedar Dieback. (n.d.). *PPO Home*. Retrieved from https://ppo.puyallup.wsu.edu/plant-health-concerns/redcedar/