Örebro University
School of Business and Economics
Statistics, advanced level thesis II, 15hp
Supervisor: Farrukh Javed
Examiner: Nicklas Pettersson
Spring 2017

Classifying Forest Cover type with cartographic variables via the Support Vector Machine, Naive Bayes and Random Forest classifiers.

Hugo Sjöqvist
1992-07-19

Abstract

In this thesis we predicted the forest type classes using various classification methods such as the Support Vector Machine, Decision Trees' Random Forest and the Naive Bayes classifiers. Our focus remained on both the overall accuracy and accuracy with respect to specific classes. To reduce the dimensionality we used variable selection methods such as the Least Absolute Shrinkage and Selection Operator (Lasso), Random Forest Selection and Principal Component Analysis (PCA). The results showed that the Random Forest classifier with Principal Component Analysis outperformed all the other models.

Acknowledgements

This thesis would not have been possible without the help of my supervisor Farrukh Javed, and also Martin Längkvist from the School of Science and Technology at Örebro University. Their different knowledge and expertise helped me greatly during the whole thesis and I am very grateful for their help.

I would also like to extend my thanks for my examiner, Nicklas Pettersson for his feedback and thoughts which helped to improve this thesis further. Finally, I would like to thank my brother Axel for helping me with the grammatical parts and structure of the thesis.

Contents

List of abbreviations

Abbreviation	Expansion	First page
1V1	One-Versus-One	10
1VR	One-Versus-Rest	10
AI	Artificial Intelligence	2
AIC	Akaike information criterion	21
ANN	Artificial Neural Network	3
C5	Paralympic cycling classification	3
CER	Classification error rate	14
CH	Canonical hyperplanes	6
CT	Classification Trees	1
CT-RF	Classification Trees-Random Forest	5
DA	Discriminant Analysis	3
GI	Gini index	14
H	Hyperplane	5
HDTFP	Independent variable number 10	19
HDTH	Independent variable number 4	19
HDTR	Independent variable number 6	19
ICA	Independent Component Analysis	29
Lasso	Least Absolute Shrinkage and Selection Operator	17
MDA	Mean Decrease Accuracy	16
NB	Naive Bayes	24
NBC	Naive Bayes' Classifier	1
PC	Principal components	21
PCA	Principal Component Analysis	17
RBF	Radial basis function	8
RF	Random Forest	1
RFC	Random Forest classifier	25
RT	Regression Trees	13
SVM	Support Vector Machine	1
VDTR	Independent variable number 5	19
VFDTcNB	Very Fast Decision Trees classification using Naive Bayes	3
VI	Variable Importance	16

1 Introduction

Classifying non-urban or nature's environment has always been in the interest of many, whether they are land owners, foresters, Environmental Scientists, governments or land management agencies. To classify a specific type of nature usually requires field studies or to be estimated from some sort of remotely sensed data, both of these methods are either costly, time consuming, inaccurate or all of them. However, with the progression of satellite imagery, a new implementation to classify different types of nature has appeared: with *Machine Learning*.

In theory, machine learning has existed for decades but its use has exploded in the recent years both due to the development of new methods and increasingly powerful computers. Accuracy in the models tend to come at the cost of heavily increased computational power and time, something which in many fields is not very desirable.

What follows in this section is a presentation of the purpose of the thesis and then a quick outline of how the subjects are approached and presented.

1.1 Purpose

The purpose of this thesis is to calibrate, analyze and compare the accuracy and speed of selected machine learning methods for classification of forest cover types and select the most optimal method. The methods chosen for this purpose are *Support Vector Machine* (SVM), *Naive Bayes' Classifier* (NBC) and the *Classification Trees'* (CT) extension *Random Forest* (RF) - all of them are known and widely used models for classification of data in the field of machine learning.
The aim is to compare and evaluate these methods and with that contribute to the existing research in this field while also giving an in-sight to the differences between the three models.

1.2 Outline

The first part of the thesis explains the background of the machine learning theory along with a review of previous literature, followed by the theory and discussion on methods such as Support Vector Machine, Naive Bayes Classifier, Classification Trees - Random Forest and afterwards feature selection methods. The next section describes the data along with its variables. Results and analysis section presents the result of the models with an in-depth analysis of their outcomes. The Discussion section contains a summary of the previous sections consisting of a discussion, where the most optimal among the considered models in this thesis is to be decided.

2 Background

Machine learning has a long history and a wide field of use. In this section the background theory of machine learning relevant to this thesis will be presented and explained, which gives the reader a basic understanding of the implementation and its uses.

2.1 The origins of machine learning

At the beginning of recorded databases, everything was done manually by humans. Whether it was the astronomers who recorded patterns of planets and stars, the biologists whom noted outcomes of crossbreeding experiments of animals or plants, or city officials that kept records of its tax payers, outbreaks of diseases or population growth. All of it required a human to observe and then record the observation. Nowadays, thanks to substantial technological progress, these things are for the most part automated by computerized databases. (Lantz 2015)

Due to the age of automation, humanity has acquired gigantic databases of all possible records - everything from data of the motions of planets to a city's hospital registers of its inhabitants. Naturally, human beings have always been surrounded by large amounts of data, but it was not until now that data has become so easily accessible and have it to be directly processed by machines. Many databases are too vast and complex for an individual to make an efficient and accurate decision manually and at the same time use all the information from the data to its fullest potential. This is where *machine learning* entered the picture.

Machine learning is a part of Artificial Intelligence (AI) and can be specified as "*The field of study interested in the development of computer algorithms for transforming data into intelligent action*" (Lantz 2015). This has developed in recent years due to the rapid growth and development of available data, statistical methods and computing power. Anderson et al. refers to machine learning as: "*The study and computer modeling of learning processes in their multiple manifestations constitutes the subject matter of machine learning.*" (Anderson et al. 1986)

2.2 A few types and uses of machine learning

There are generally two different methods for training machine learning algorithms: Supervised and unsupervised learning. In supervised learning, the user knows both the input and output data beforehand and with this train the models to know the correct outcome beforehand (Alpaydin 2010). i.e. mapping of a specific type of disease, knowing who has the disease and who does not have it. Then we know the attributes of each individual and if they have the disease or not, which makes the model know

the correct outcome. Supervised learning is also known as *discrimination* in statistical theory (Michie et al. 1994).

In unsupervised learning one does not know the outcome and only have the attributes (i.e. the independent variables). Instead of knowing what the correct output is, the model tries to find patterns that occurs more often than others and irregularities in the data. Herein the focus is to see what generally happens and what does not - which is usually referred to as the *density estimation*. (Alpaydin 2010)

2.2.1 Noticable areas of machine learning

Machine learning is a broad field with a wide range of different implementations, a few of which shall be described here. One rather popular implementation nowadays is for *recommendation systems*, e.g. how a new product gets recommended to a user based on other, different products the user has consumed.

Speech recognition is another popular field of machine learning, in which the models are trained to identify the wave length and other attributes in a person's voice. For instance, this is used for modern mobile phones' voice recognition lock.

Text recognition can aid people with reading disabilities to identify words or letters in texts. *Data classification*, which this thesis will be focused on, can be used in training models to classify large or complex data based on different attributes into categories.

2.3 Literature review

The forest cover type dataset (which this thesis uses) has previously been used by other studies. The first study was published in *Computers and Electronics in Agriculture* by Blackard and Dean in 1999. They used a known method called *Artificial Neural Network* (ANN) and a model built on Gaussian Discriminant Analysis (DA) . Their overall prediction accuracy for the ANN was 70.6 % while their DA model gave an accuracy of 58.4 % (Blackard & Dean 1999).

Another paper achieved 70.0 % accuracy for backpropagation, 58.0 % for linear discriminant analysis, 68.6 % for the decision tree algorithm C4.5 and 62.4 % for the Very Fast Decision Trees classification using Naive Bayes (VFDTcNB) (Gama et al. 2003). The dataset has also been used in other projects where the reported accuracy was 78.6 % (Crain & Davis 2014). Using the Paralympic cycling classification (C5) model, Bagnall and Cawley achieved their highest accuracy with 83.7 % (Bagnall & Cawley 2003).

Other articles have also used this study where their highest accuracy gave 80.0 %

(Fu et al. 2010), 69.9 % (Sug 2010), 70.5 % (Gupta et al. 2015), 74.0 % (Ridgeway 2002) and 91.6 % (Karampatziakis & Mineiro 2013).

3 Theory & models

This section will be devoted to give a more in-depth explanation of the models and their theories. We will first consider the SVM approach, followed by the Naive Bayes Classifier, afterwards Classification Trees-Random Forest (CT-RF) and then finish the section by discussing the feature selection methods.

3.1 Support Vector Machine

The Support Vector Machine (SVM) is a machine learning method which attempts to generalize and make predictions on the gathered data. For SVM, we first divide the data into training and prediction (or *test*) sets. For classification, the training data has a set of input vectors (x_i) with different features or attributes where every observation (or input vector) is followed by a label/category denoted as y_i ($i = 1, ..., m$). For simplicity, consider y_i to be a binary variable with the values $y_i = +1$ or $y_i = -1$.

3.1.1 Binary Support Vector Model

Following the discussion in(Campbell & Ying 2011), the purpose of SVM is to find a *directed hyperplane* which is oriented in such a way that the $y_i = +1$ and $y_i = -1$ are graphically divided by a line into two categories on the hyperplane. The hyperplane (H) that is maximally the most distant from the two input classes is then considered to be the directed hyperplane - the points closest to the separating hyperplane will have most influence and are called *support vectors*. By denoting $\mathbf{W} \cdot \mathbf{x} + b = 0$ as the separating hyperplane, where $\cdot$ is the inner or scalar product, b the offset or bias from the origin in the input space to the hyperplane, the points located within the hyperplane are denoted as $\mathbf{x}$. The weights ($\mathbf{W}$) which are normal to the hyperplane, will determine its orientation.

Binary SVM classification is popular in statistical theory due to its ability in handling the upper bound of the *generalization error*, i.e. the error that comes from the theoretical application when applying the model to new and unseen instances. The two important properties of such an error are the following:

Property 1. *By maximizing the margin (**ma**) we can minimize the bound - the margin is the minimal distance between the hyperplane which separates the two classes and the data-points which are closest to the hyperplane.*

Property 2. *The bound in property 1 is not depending on the dimensionality of the space.*

With the $y_i = \pm 1$, the previous decision function will now be written as:

$$f(x) = sign(\mathbf{W} \cdot \mathbf{x} + b) \tag{1}$$

where *sign* is a non-zero indication. Since $\cdot$ being the inner product or scalar, then we have: $\mathbf{W}\cdot\mathbf{x} = \mathbf{W}^T\mathbf{x}$.
This will make the data correctly classified if

$$y_i(\mathbf{W}\cdot\mathbf{x}_i + b) > 0, \quad \forall i \tag{2}$$

since $(\mathbf{W}\cdot\mathbf{x}_i + b)$ is positive when $y_i = +1$ and negative when $y_i = -1$.

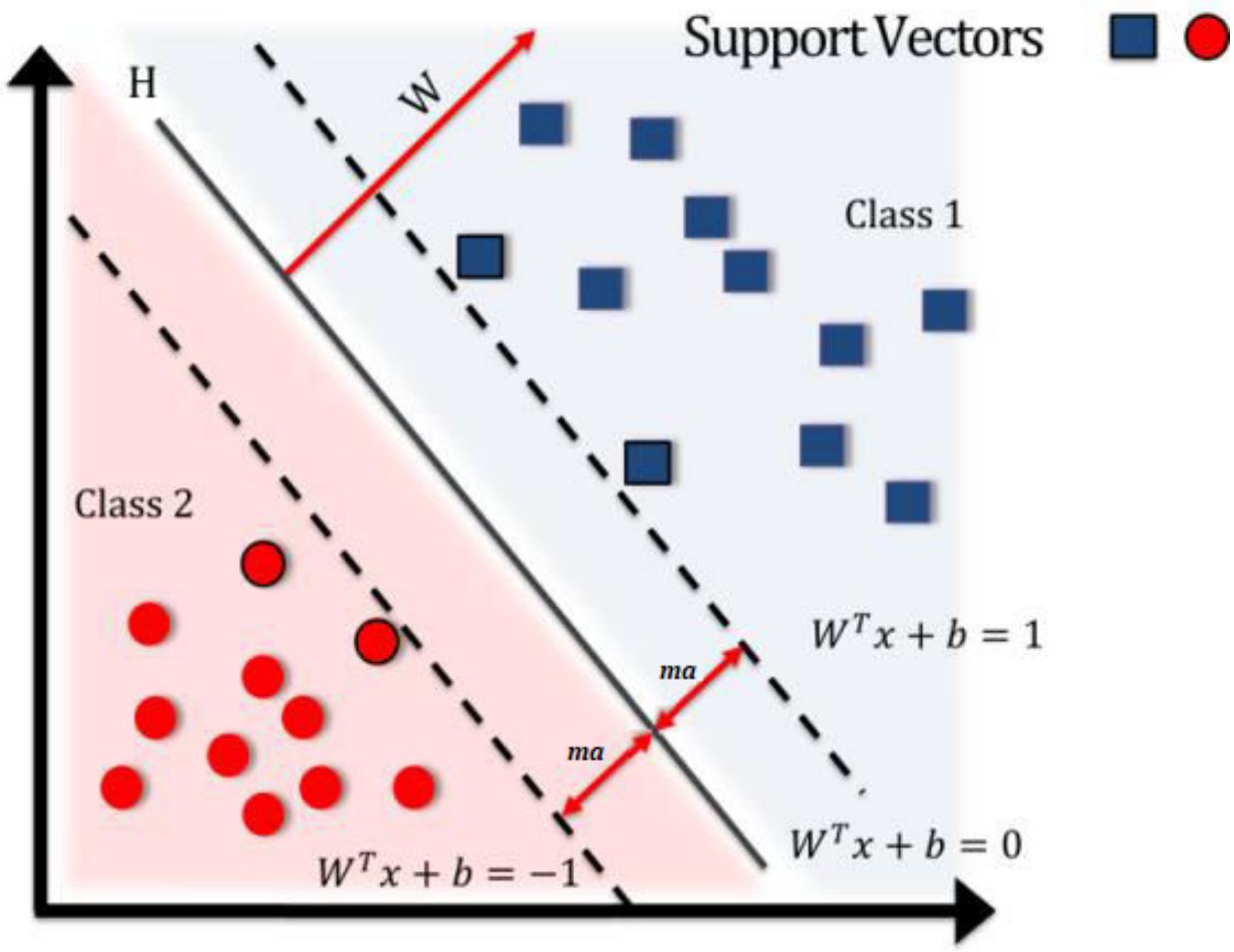

Figure 1: The basic layout and function of the binary SVM model for classification (Leal & Sanchez 2015)

Focusing on property 1, define a scale for $(\mathbf{W}, b)$ for the closest points on the two different sides by setting $\mathbf{W}\cdot\mathbf{x}+b = 1$ and $\mathbf{W}\cdot\mathbf{x}+b = -1$. The hyperplanes are called *canonical hyperplanes (CH)* when they pass through $\mathbf{W}\cdot\mathbf{x}+b = 1$ and $\mathbf{W}\cdot\mathbf{x}+b = -1$ - we define *margin band* as the region between those canonical hyperplanes. Now define

$\mathbf{x}_1$ and $\mathbf{x}_2$ as two points inside the CH. If we have that:

$$\mathbf{W}\cdot\mathbf{x}_1 + b = 1 \leftrightarrow b = (1 - \mathbf{W}\cdot\mathbf{x}_1) \tag{3}$$

and

$$\mathbf{W}\cdot\mathbf{x}_2 + b = -1 \leftrightarrow b = -(1 + \mathbf{W}\cdot\mathbf{x}_2) \tag{4}$$

it can now be deduced that since

$$b = b \leftrightarrow (1 - \mathbf{W}\cdot\mathbf{x}_1) = -(1 + \mathbf{W}\cdot\mathbf{x}_2) \leftrightarrow 2 = \mathbf{W}\cdot\mathbf{x}_1 - \mathbf{W}\cdot\mathbf{x}_2 \tag{5}$$

i.e. we have that: $\mathbf{W}\cdot(\mathbf{x}_1 - \mathbf{x}_2) = 2$.

Now define the normal vector for the hyperplane $\mathbf{W}\cdot\mathbf{x} + b = 0$ as $\mathbf{W}/||\mathbf{W}||_2$, where $||\mathbf{W}||_2$ is the square root of $\mathbf{W}^T\mathbf{W}$. The projection of $\mathbf{x}_1 - \mathbf{x}_2$ towards the normal vector $\mathbf{W}/||\mathbf{W}||_2$ will yield that $(\mathbf{x}_1 - \mathbf{x}_2)\cdot\mathbf{W}/||\mathbf{W}||_2 = 2/||\mathbf{W}||_2$. The margin is now $\mathbf{ma} = 1/||\mathbf{W}||_2$ which is half the distance between the two CH.
By maximizing the margin, it will be the same as minimizing:

$$\frac{1}{2}||\mathbf{W}||_2^2 \tag{6}$$

with the constraint:

$$y_i(\mathbf{W}\cdot\mathbf{x}_i + b) \geq 1, \quad \forall i \tag{7}$$

The equation above is a constrained optimization problem, which can be solved with the *Lagrange function* where the m constraints are multiplied by their respective *Lagrange multiplier*, which gives the primal function:

$$L(\mathbf{W}, b) = \frac{1}{2}(\mathbf{W}\cdot\mathbf{W}) - \sum_{i=1}^{m}\alpha_i(y_i(\mathbf{W}\cdot\mathbf{x}_i + b) - 1) \tag{8}$$

with α_i as the Lagrange multipliers where $\alpha_i \geq 0$. Taking the derivatives with respect to b and $\mathbf{W}$ while setting them to zero yields:

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{9}$$

and

$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{W} - \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i = 0 \tag{10}$$

Substituting $\mathbf{W}$ from equation 10 back into $L(\mathbf{W}, b)$ will give the *dual formulation* (a.k.a. the *Wolfe dual* (Wolfe 1961)) :

$$W_d(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{11}$$

This have to be maximized with the constraints:

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{12}$$

with respect to α_i.
Until now we have only covered property 1. Property 2 will give the following;

Going back to Wolfe's dual formulation, it is possible to see how $\mathbf{x}_i$ only appear inside an inner product. Using a so called *feature space* - a space with a different dimensionality used to get an alternative representation of the data by mapping the data-points into it; with a replacement give:

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{13}$$

Here $\Phi(\cdot)$ is defined as the mapping function. If the data is not linearly separable, i.e. as in Figure 2, in the input space, performing the mapping function will counter this problem by adding an extra dimension for the hyperplane to separate the classes (e.g. imagine going from a 2D plane to a 3D). As long as a margin can be defined, property 2 claims that there will be no loss of generalization performance when mapping to a feature space where the data is separable.

It is irrelevant if we know the functional form of the mapping $\Phi(\mathbf{x}_i)$ since it is implicitly defined by the choice of the *kernel*: $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Of course, there are restrictions such that the inner product in feature space must be consistently defined - which will restrict the feature space to being an inner product space (it is usually referred to as *Hilbert space*). With the mapping function one is now able to control for both linearly separable and non-linearly separable.
One can use the *linear kernel*: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$, which has no mapping to feature space. This will not yield a training error of zero if the data is not linearly separable while attempting to solve the optimization problem in equation 11 and 12.

The data might not necessarily be separable in its input space with the linear kernel, but it becomes separable when applying it to a higher-dimensional space by instead using a *Gaussian kernel* - also known as *Gaussian Radial basis function (RBF) kernel*, or just *RBF kernel*:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\mathbf{x}_i - \mathbf{x}_j)^2 / 2\sigma^2} \tag{14}$$

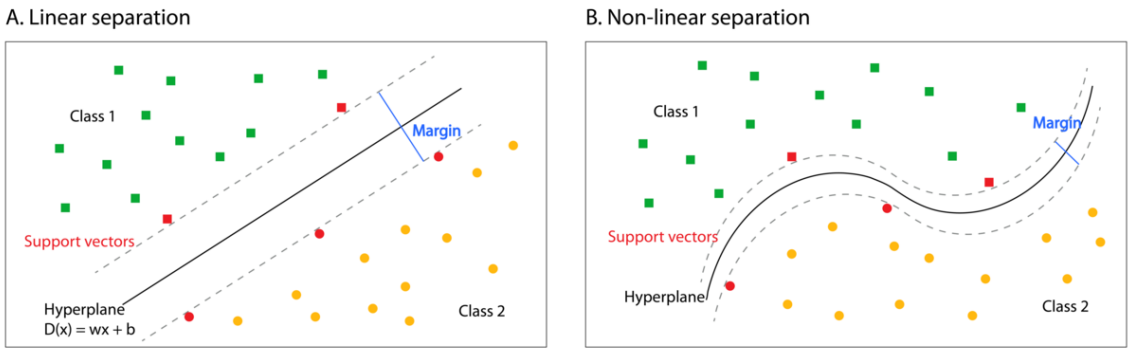

Figure 2: A comparison between linear and nonlinear SVM classification (Perseus documentation 2015)

where σ^2 is the Gaussian kernel parameter which must be specified, usually by using training data and specify it as the most optimal value. A Gaussian kernel is not the only option, there are several other possible kernel substitutions to be used like the *polynomial kernel* (Chang et al. 2010) or the *feedforward neural network classifier* (Franco & Cannas 2000) - but the focus in this thesis will lie on the Gaussian kernel.

Deciding the choice of kernel, the learning task for binary classification will therefore involve a maximization of:

$$W_d(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{15}$$

still subject to the constraint of $\alpha_i \geq 0$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$.

To identify the bias b, we have a data-point with $y_i = +1$ so it could be noted that:

$$\min_{[i|y_i=+1]} [\mathbf{w} \cdot \mathbf{x}_i + b] = \min_{[i|y_i=+1]} \left[\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right] + b = 1 \tag{16}$$

Recall the other class of $y_i = -1$, implementing the same logic and rewrite it will yield the bias as:

$$b = -\frac{1}{2} \left[\max_{[i|y_i=-1]} \left(\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) + \min_{[i|y_i=+1]} \left(\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \right] \tag{17}$$

Now place the $(\mathbf{x}_i, y_i)$ data into equation 15 and maximize $W(\alpha)$, taken its previous constraints into consideration. Now denote α_i^* as the most optimal value of α_i, the bias can be calculated with equation 17. The predicted class for an input vector $(\mathbf{z})$

can now be written as:

$$\phi(\mathbf{z}) = \sum_{i=1}^{m} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{z}) + b^* \tag{18}$$

where b^* is the bias at the most optimal value. Implementing it shall yield specific points which lie closest to the hyperplane which will have $\alpha_i^* > 0$, referred to as the *support vectors* - all other points will have $\alpha_i^* = 0$ and have the decision function to be independent from those samples.

3.1.2 Multi-class Support Vector Model

Until now the SVM has only had the theoretical explanation for a binary outcome. However, there are often cases in which we encounter situations with more than two outcomes that would require a slight deviation from the above mentioned theory. There are a few ways to extend the binary classification theory to a multi-class SVM.

One such method is the *One-Versus-Rest* (1VR, a.k.a. *One-Versus-All*) approach, in which one constructs k separate binary classifiers for a response variable of k categories. Now define the m-th classifier as a positive and the rest k-1 as negative outcomes. The class label will, during the training period, be determined by the binary classifier which gives the maximum output value (Ma & Guo 2014). A major problem with the 1VR approach is the imbalance of datasets. Since one only classifies one category at a time to be positive, it means that the ratio of positive to negative examples is $\frac{1}{k-1}$ if all categories are of a approximately equal size. The symmetry of the original problem will then be lost.

Another approach, refereed to as: the *One-Versus-One* approach (1V1) - a.k.a. *pair-wise decomposition* has been proposed (Bishop 2006). This method will pair all possible classifiers and evaluate them against each observation, yielding $k(k-1)/2$ individual binary classifiers. The pairs are then all compared against one observation, giving the class that were the most optimal in each pair one 'vote'. Finally it will summarize and apply the class to the example which received the most votes (Ma & Guo 2014).

It can be deduced that the 1V1 approach is considerably more symmetric than the 1VR approach, taking the 1VR's imbalance weakness into consideration. However, a negative aspect of the 1V1 approach might be that it is significantly more time consuming than the 1VR, since it requires more steps to make the classification. However, in this thesis, accuracy will be of a higher priority than speed, therefore, the *the One-Versus-One* approach will be used for Multi-class SVM prediction.

The cost hyperparameter (C) for the used SVM will be C=100 in this thesis, C is a measurement for how much one would like to avoid misclassification based on the

proximity of the hyperplane. The value of C give a trade-off between accuracy and computational time - the higher the C, the more accuracy one is bound to get but the time to train the model increases considerably.

3.2 Naive Bayes' Classifier

To understand the Naive Bayes' Classifier, one first need to understand Bayes' theorem. This subsection will have a brief overview of Bayes' theorem followed by the theory and implementation of the NBC algorithm.

3.2.1 Bayes' theorem

Bayes' theorem was proposed by Thomas Bayes (1701-1761), who is considered to be the father of Bayesian statistical theory. Bayes' theorem is given by this simple equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{19}$$

where A and B are events such that the $1 \geq P(A) \geq 0$ and $1 \geq P(B) \geq 0$.
For a model with the data y and the parameter θ, Bayes' theorem can be implemented for the posterior of θ given y. Since the data y can be seen as constant, Bayes' theorem can be written as

$$P(\theta \mid y) = \frac{f(y \mid \theta)f(\theta)}{\int f(y \mid \theta)f(\theta)d\theta} \propto f(y \mid \theta)f(\theta) \tag{20}$$

i.e. the posterior distribution for the parameter θ, given the data y can be seen as the proportional product of the likelihood function for y, given θ multiplied by the prior distribution for θ. The accuracy or estimation can then be improved by adding a prior distribution with a Bayesian approach, but it should be noted that it is not guaranteed (Sjöqvist 2017).

3.2.2 Implementation of the Naive Bayes' Classifier

The peculiar name of *Naive* Bayes' Classifier origins from the fact that the method assumes the attributes, given their category value, to be independent from each other - which might be a rather restrictive assumption but saves considerable computational time and power. Now recall Bayes' theorem, but instead of the parameter θ and data y it will instead be classified by n attribute values $\mathbf{x} = (x_1, x_2, ..., x_n)$ where x_i is the value of attribute X_i for an example (or observation). CL will be the classification (or category) variable with the value cl - for simplicity, the NBC will first be explained with CL as a binary variable with the values ± 1 instead of multi-categorical.

With this we have that the probability of an observation to be in the specific class:

$$p(cl|\mathbf{x}) = \frac{p(\mathbf{x}|cl)p(cl)}{p(\mathbf{x})} \tag{21}$$

where $\mathbf{x}$ is classified as $CL = +1$ if the Bayesian function

$$f_b(\mathbf{x}) = \frac{p(CL = +1|\mathbf{x})}{p(CL = -1|\mathbf{x})} \geq 1 \tag{22}$$

By assuming that all attributes are independent of each other given the class-value CL, it can be written as:

$$p(\mathbf{x}|cl) = p(x_1, x_2, ..., x_n|cl) = \prod_{i=1}^{n} p(x_i|cl) \tag{23}$$

This will then yield the NBC

$$f_{nb}(\mathbf{x}) = \frac{p(CL = +1)}{p(CL = -1)} \prod_{i=1}^{n} \frac{p(x_i|CL = +1)}{p(x_i|CL = -1)} \tag{24}$$

if CL is a category variable consisting of two classes.(Zhang 2004)

Leaving the simplified binary-class example, consider now the class (or output) CL to be CL_k with $k = 1, 2, ..., K$ number of categories - the vector $\mathbf{x}$ is still the same. This will give the *joint probability distribution* $p(\mathbf{x}, CL_k)$ with the prior probability $p(CL_k)$ and the class probability $p(CL_k|\mathbf{x})$.

Applying the probability $p(CL_k|\mathbf{x})$ with Bayes' theorem will now give:

$$p(CL_k|\mathbf{x}) = \frac{p(\mathbf{x}|CL_k)p(CL_k)}{p(\mathbf{x})} = \frac{p(x_1, x_2, ..., x_n|CL_k)p(CL_k)}{p(x_1, x_2, ..., x_n)} \tag{25}$$

Using the basic *Chain Rule* from probability theory $p(\mathbf{x}|CL_k)$ can be decomposed to: $p(x_1, x_2, ..., x_n|CL_k) = p(x_1|x_2, ..., x_n, CL_k) \cdot p(x_2|x_3, ..., x_n, CL_k) \cdot ... \cdot p(x_{n-1}|x_n, CL_k) \cdot p(x_n|CL_k)$. Assume now the 'Naive' feature of conditional independence in NBC, the decomposition can then be formulated as:

$$p(x_i|x_{i+1}, ..., x_n, CL_k) = p(x_i|CL_k) \rightarrow p(x_1, ..., x_n|CL_k) = \prod_{i=1}^{n} p(x_i|CL_k) \tag{26}$$

Due to independence between the conditional attributes. This can be implemented as:

$$\begin{aligned} p(CL_k|x_1, ..., x_n) &\propto p(CL_k, x_1, ..., x_n) \\ &\propto p(CL_k)p(x_1, ..., x_n|CL_k) \\ &\propto p(CL_k)p(x_1|CL_k)p(x_2|CL_k)...p(x_n|CL_k) \\ &\propto p(CL_k) \prod_{i=1}^{n} p(x_i|CL_k) \end{aligned} \tag{27}$$

The likelihood $p(x_i|CL_k)$ is usually modeled by using the same class of probability distribution, i.e. binomial or Gaussian. The chosen likelihood distribution in this thesis is the Gaussian distribution with the class proportions of the training sets as the prior probabilities/distributions.

The choice of output category itself is fairly simple. Assume two different categories CL_a and CL_b, CL_a will be the chosen category if:

$$p(CL_a)\prod_{i=1}^{n} p(x_i|CL_a) > p(CL_b)\prod_{i=1}^{n} p(x_i|CL_b) \rightarrow p(CL_a|\mathbf{x}) > p(CL_b|\mathbf{x}) \tag{28}$$

The complete mathematical notation for the chosen category CL_k for $(k = 1, 2, ..., K)$ number of categories can be written as:

$$\hat{CL} = \underset{k\in\{1,...,K\}}{\arg\max}\ \ p(CL_k)\prod_{i=1}^{n} p(x_i|CL_k) \tag{29}$$

Where $\hat{CL}$ is the estimated output category for the vectors/features $\mathbf{x}$.

3.3 Decision Trees

The theory followed in this section will be referenced from G. James et al. (2013). Compared to the other classification methods mentioned above, Decision Trees are significantly simpler. As the name explains, the method divides or stratifies the data into a "tree" and paths a way to the most expected class or value. An attractive attribute of tree based methods is their very simplistic interpretation. However, they have a tendency to perform worse than more advanced machine learning methods due to the danger of overfitting when it comes to prediction accuracy (James et al. 2014).

Decision Trees are used both for *regression* and *classification* problems with applied extensions such as *bagging* and *random forest*. The difference between Classification Trees (CT) and Regression Trees (RT) is that the CT are used to predict a qualitative response rather than a quantitative one while the RT focuses on outcomes with continuous quality. Since the response variables in the chosen data are of qualitative nature, the focused theory of Decision Trees will be towards Classification Trees.

3.3.1 Classification Trees

The basic principle of the CT is to "*predict that each observation belongs to the most commonly occurring class of training observation in the region to which it belongs*" (James et al. 2014) - i.e. if a class have a majority of a specific attribute, then those attributes will most likely give a representative picture for that specific class.

To make a Classification Tree grow, one uses (like in Regression Trees) *recursive binary splitting* by selecting the predictor X_j and the cutpoint s and split so that the *classification error rate* (CER) is given the most possible reduction in the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$. In short, consider all predictors $X_1, ..., X_p$ and all possible values for s such that the lower tree has the smallest possible CER - the CER is just the part of the training observations in that region that do not belong to the most common/predicted class. Assume that $R_1, ..., R_j$ are the distinct and non-overlapping regions that we divide the predictor space (i.e. $X_1, ..., X_p$) into. Assuming only R_1 and R_2, then we define the pair of half-planes for any j and s into

$$R_1(j, s) = \{X|X_j < s\} \qquad R_2(j, s) = \{X|X_j \geq s\} \tag{30}$$

We specify the CER as:

$$CER = 1 - \max_k(\hat{p}_{mk}) \tag{31}$$

with $\hat{p}_{mk}$ as the proportion of the training data in the mth region that are from the kth class. Due to the fact that the CER is not very sensitive for tree-growing there are two other possible measures to implement:

1. The *Gini index* (GI)
   It is defined by:

   $$GI = \sum_{k=1}^{k} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{32}$$

   As can easily be seen, it is a measurement of the total variance across the K classes. Of course, the GI will not take a small value if its proportions are close to zero or one - this causes the GI to be referred to as a measurement of *node purity*, which is a small value that indicates that a node holds most observations from a single class.

2. The *cross-entropy* is another alternative, given by

   $$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{33}$$

   Due to the proportion $\hat{p}_{mk}$ being $0 \leq \hat{p}_{mk} \leq 1$, it is easy to see that $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$. In short, this method will take on a value close to zero if most of the $\hat{p}_{mk}$'s are approximately zero or one.

GI estimation and cross-entropy usually gives a similar result. If the goal is to predict the accuracy of the final tree the CER estimation is preferable.

3.3.2 Random forest

As previously mentioned, Decision Trees usually has less accuracy than more complicated models such as SVM or Artificial Neural Network (ANN). However, by using extensions such as bagging and random forest one might be able to improve the accuracy.

Bagging is a procedure used to reduce the variance of a statistical method. Assume that we have n independent observations, $Z_1, ..., Z_n$ all of which has the variance σ^2 - the variance of the mean $\bar{Z}$ is of course then given by σ^2/n. This means that one can reduce the variance by averaging a set of observations. In this case, one can increase the accuracy of the method by taking many training sets from the population, build their own independent prediction models and the averaging the results of the predictions - i.e. calculating $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ by using B separate training sets. Then averaging all predictions will yield the bagging estimate:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \tag{34}$$

This method is popular to use in Decision Trees.

Random forest can be seen as an extension of bagging. It provides an improvement which decorrelates the trees by forcing each split to consider only a subset of the predictors, when faced with a considerable amount of notable predictors that might cause correlation if one tries to use them all. The random forest builds a number of decision trees on bootstrapping, such as bagging. But each time a split in a tree is considered the method takes a random sample of m predictors from the set of p predictors, that will be used as candidates - this might prevent decision trees from having the trees to look very similar to each other (which is a problem that might occur with bagging). The split will only use one those m predictors. After each split a new sample of m predictors are taken - usually, we have that $m \approx \sqrt{p}$.

The main difference between random forest and bagging is the choice of m's size, if $m = p$ then random forest and bagging will yield the same or very similar results.

3.4 Feature selection and data management

In statistics there is a thing called *parsimony*, which roughly refers to models that use the right amount of variables to explain the model well. The idea behind parsimony is that one should not use more variables in the model than necessary without losing too much information.

There exist cases where more variables might hurt rather than help the model and the model will be in danger of overfitting, also known as *the curse of dimensionality*.

Maybe two variables have an identical effect on the outcome, or maybe one variable has no association whatsoever with the outcome. In those cases we might either lose prediction accuracy and/or increase the computational time. Using unnecessarily many predictors might also lead to a high variance among estimates.

Therefore feature selection methods have been developed to examine whether variables are necessary to include in the model. A few of those developed methods are explained and implemented here due to their extensive use in the field, those are: *Random Forest selection*, *Lasso* and *Principal Component Analysis*.

3.4.1 Random Forest selection

The Random Forest selection method shows the importance of the included variables in the model, rather than recommending them to be included or not. This gives a sort of ranking to each independent variable and leave it up to the user to chose whether to include or exclude the variables.

The Variable Importance (VI) factor can be evaluated for a variable X_m when predicting Y with, for all nodes (t - nodes can be seen as the decision of splitting the tree into parts, with respect to data points), adding up the weighted impurity (measurements) decreases $p(t)\Delta i(s_t, t)$ where X_m is in practice by averaging over all N_T number of trees in the forest:

$$VI(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T: v(s_t) = X_m} p(t)\Delta i(s_t, t) \tag{35}$$

with $p(t)$ as the proportion N_t/N of the samples the internal nodes t and finally $v(s_t)$ as the variable used in the split of s_t. A measurement for the VI can be evaluated by the Gini measurement (see section *Theory & and models, Decision Trees*) or the *Mean Decrease Accuracy* (MDA) where the values of the variables in focus are randomly permuted in out-of-bag sample. Both the GI and MDA have shown to be biased in different studies due to depending on number of observations or categorical data, but also shown to perform well in others. (Louppe et al. 2013)

To calculate the overall VI and try to reduce a possible bias, a VI proportion will be calculated with the matrix $\mathbf{X}_{VI}$ consisting of the variables as rows and their 7 different VI values and also the GI and MDA for columns. Then the vector for the variable proportions ($\mathbf{X}_{prop}$) will simply be calculated with:

$$\mathbf{X}_{prop} = \sum_{i=1}^{k} (\mathbf{X}_{VI} / \sum_{i=1}^{k} \sum_{j=1}^{n} \mathbf{X}_{VI})^T \tag{36}$$

with k rows and n columns.

3.4.2 Lasso

The **L**east **A**bsolute **S**hrinkage and **S**election **O**perator (Lasso) is a variable selection and shrinkage method proposed by Tibshirani in 1996. The method simultaneously select variables of interest while estimating the model's parameters. Lasso is a continuation of the ridge estimator, which is almost identical except for the penalty function - the ridge estimator have the absolute value of β as squared, while the Lasso has β raised to the power of one. The method *shrinks* the regression parameters (β) with a generalized version of penalty estimators - for a given penalty function $\pi(\cdot)$ and regularization parameter λ. The general form of the ridge regression - a generalized version of penalty (which Lasso's foundation is based on) estimators can be written as:

$$S(\beta) = (y - \mathbf{X}\beta)'(y - \mathbf{X}\beta) + \lambda\pi(\beta) \tag{37}$$

with the penalty function:

$$\pi(\beta) = \sum_{j=1}^{m} |\beta_j| \tag{38}$$

Here $\sum_{j=1}^{m} |\beta_j|$ bounds the L norm of the parameters given the tuning parameter t as $\sum_{j=1}^{m} |\beta_j| \leq t$.

With this the complete Lasso function is given by:

$$\hat{\beta}^{Lasso} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{39}$$

Where p are the number of dimensions and n the sample size. The Lasso estimator are numerically feasible even if the dimensions are much higher than the sample size. (Ahmed 2013)

3.4.3 Principal Component Analysis

Principal **C**omponent **A**nalysis (PCA) is another popular approach to reduce the dimensional of variables. It seeks linear combinations for the independent variables X with maximal (or minimal) variance. Due to constraining the combinations to have unit length, the variance can be scaled be rescaling said combinations. Having S being the covariance matrix, it is be defined by (Venables & Ripley 2003) as:

$$nS = (X - n^{-1}\mathbf{1}\mathbf{1}^T X)^T (X - n^{-1}\mathbf{1}\mathbf{1}^T X) = (X^T X - n\bar{\mathbf{x}}\bar{\mathbf{x}}^T) \tag{40}$$

where $\bar{\mathbf{x}} = \mathbf{1}^T X/n$ is the means of the variables for the row vector - n is the number of rows, $\mathbf{1}$ represents the identity or the neutral multiplicative member of a group. The linear combination $\mathbf{xa}$ for the sample variance of a row vector $\mathbf{x}$ can be seen as

$\mathbf{a}^T\Sigma\mathbf{a}$. This will be max- or minimized subject to $||\mathbf{a}||^2 = \mathbf{a}^T\mathbf{a} = 1$ due to σ being a non-negative definite matrix with an eigendecomposition of

$$\Sigma = C_{ort}^T \Lambda C_{ort} \tag{41}$$

with Λ as a diagonal matrix of eigenvalues ≥ 0 in decreasing order. Have $\mathbf{b} = C_{ort}\mathbf{a}$ with C_{ort} as an orthogonal matrix, leading to $\mathbf{b}$ having the same length as $\mathbf{a}$. The aim is now to maximize $\mathbf{b}^T\Lambda\mathbf{b} = \Sigma\lambda_i b_i^2$ with the restriction that $\Sigma b_i^2 = 1$. By taking $\mathbf{b}$ to be the first unit vector, the variance can be maximized - similarly by taking $\mathbf{a}$ to be the column eigenvector corresponding to the largest eigenvalue of Σ. By taking subsequent eigenvectors it will yield vector combinations with the maximum, possible variance that are uncorrelated with the previously taken combinations. For Principal Component analysis, the ith principal component will be picked up by this procedure as the ith linear combination. (Venables & Ripley 2003)

3.5 Cross-validation

To be certain that the results did not appear due to randomness or pure luck, a cross-validation procedure will be performed. For this the data will be divided into five (k) parts, i.e. 20 % of the data will be in each section. For example, in the first run k_1 will be seen as test data and the rest as training, in the second k_2 will be test and the rest training, etc - The figure 3 below summarizes the whole procedure. Once

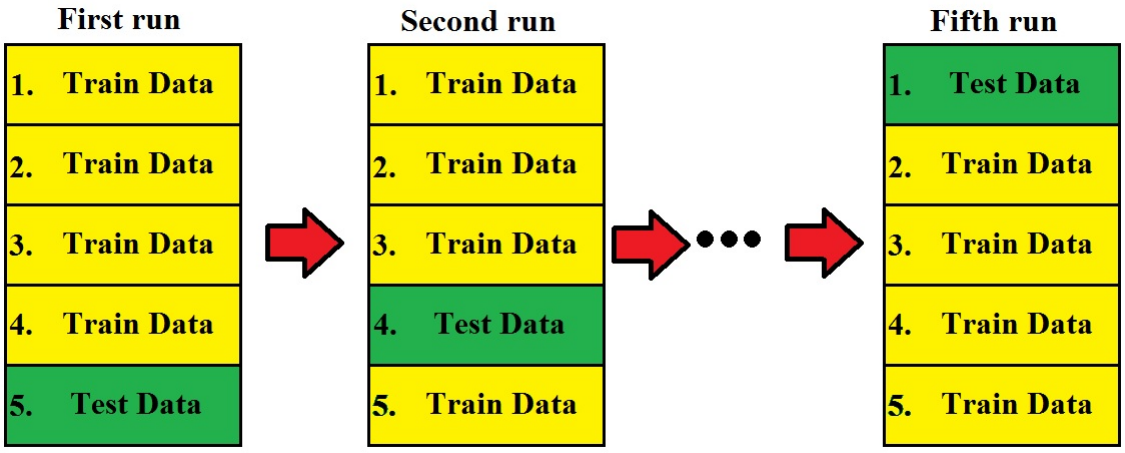

Figure 3: The partition and layout of the cross-validation

the cross-validations are completed, the calculated average of their accuracy will then be the approximated accuracy overall. Doing this will hopefully eliminate or at least greatly reduce the risk of untrustworthy results. This will be done for all methods with and without the implemented variable selection.

4 Data

This section presents a brief summary of the data along with the program used to run the algorithms. In the hope of avoiding extreme values, increasing the computational speed and to get a better accuracy, the input data will be normalized.

The data used was the open *Cover type dataset* from the University of California, Irvine, School of Information and Computer Sciences database (Lichman 2013). The data contains 581,012 observations, with no missing values or observations, of cover types from four wilderness areas located in the Roosevelt National Forest of northern Colorado. Apart from the prediction variable, *cover type*, which had a 1-7 category of different types, the data is followed by 12 attributes (Blackard & Dean 1999):

1. The elevation (in meters),
2. The aspect (in degrees azimuth),
3. The slope (in degrees),
4. The horizontal distance to the nearest surface of a water feature in meters (HDTH),
5. The vertical distance to nearest surface water feature in meters (VDTR),
6. The horizontal distance to nearest roadway in meters (HDTR),
7. A relative measure of incident sunlight at 09:00 h on the summer solstice (from a 0-255 index),
8. A relative measure of incident sunlight at noon on the summer solstice (from a 0-255 index),
9. A relative measure of incident sunlight at 15:00 h on the summer solstice (from a 0-255 index),
10. The horizontal distance to nearest historic wildfire ignition point in meters (HDTFP),
11. The soil type designation (40 binary values, one for each soil type)
and
12. The wilderness area designation (four binary values, one for each wilderness area).

The number of observations in each forest cover type is the following:

Table 1: Number of observations in each cover type category.

Cover type	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Observations	581,012	211,840	283,301	35,754	2,747	9,493	17,367	20,510
Percentage	100.0	36.5	48.9	6.15	0.47	1.63	2.99	3.53

In the same order as Table 1, the different forest cover types were **1** lodgepole pine (*Pinus contorta*), **2** spruce/fir (*Picea engelmannii* and *Abies lasiocarpa*), **3** ponderosa pine (*Pinus ponderosa*), **4** Douglas-fir (*Pseudotsuga menziesii*), **5** aspen (*Populus tremuloides*), **6** cottonwood/willow (*Populus angustifolia*, *Populus deltoides*, *Salix bebbiana* and *Salix amygdaloides*), and **7** krummholz. The krummholz forest cover type class is composed primarily of Engelmann spruce (*Picea engelmannii*), subalpine fir (*Abies lasiocarpa*) and Rocky Mountain bristlecone pine (*Pinus aristata*). (Blackard & Dean 1999)

As can be seen in Table 1, there exists quite a bit of imbalance between the categorical outcomes. This will be kept in consideration when analyzing the results, since different methods perform differently depending on how balanced the dataset is. The correlation matrix between the continuous variables are reported in the Appendix subsection 7.2.

To handle the data, performing algorithms and estimating the models, the R (version 3.1.3) statistical programming software was used.(R Core Team 2017)[1]

[1]The R-codes can be provided upon request.

5 Results and analysis

What follows here is the results of the variable selection methods and classifier models mentioned in Section 3. Short analyses will also be included in this section. To see the full, detailed results of the models, see Appendix subsection 7.1.

5.1 Variable selection

Each variable selection method proposed different choices of variables for the final model. It should be mentioned that a basic Stepwise forward-backward variable selection implementation of multinomial logistic regression (Menard 2002) was also performed with the Akaike information criterion (AIC) (Menard 2002). However, the most optimal AIC value was obtained for the full model choice (which might depend on the large numbers of observations and relatively few variables in the data), which is already included here - hence the Stepwise variable selection model will not be in focus, but rather indirectly referred to when referring to the full model.

The Lasso variable selection recommendation proposed a different variable selection for each of the seven categorical outcomes. This lead to that the excluded variables was decided by examining the pattern of the variables the Lasso selection deemed to have the least impact on the model overall in each outcome category. The result yielded the exclusion of the variables *horizontal distance to nearest roadway in meters (HDTR)* and *the horizontal distance to nearest historic wildfire ignition point in meters (HDTFP)*.

For the Random Forest Variable Importance factor, the plot can be seen in Figure 4, it was suggested to exclude the variable which was deemed to have the lowest VI - *The slope variable*. There was some consideration in removing all variables which had an importance proportion under 5 %, but it was dismissed due to the fact that the exclusion of 5 out of 12 variables would be considered too much.

For the Principal Component Analysis, there was no clear answer to where the cut-off of the variance proportion should be made. However, taking into consideration the number of principal components (PC) being 54, whereas quite a bit of them contributing to the proportional variance with an insignificant amount - as can be seen in Figure 5, it was decided to put the threshold at 80 % of the proportional variance. With this the selected number of PCs will be 33, yet most of the explained variance will be retained.

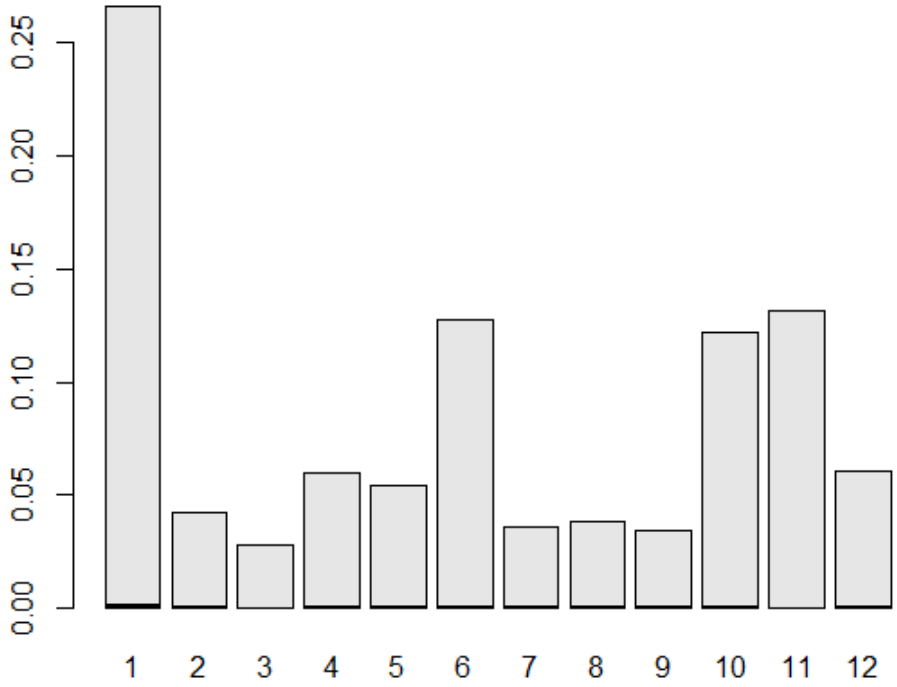

Figure 4: A bar plot over the proportional variable importances for Random Forest Variable selection in the same order as section 4.

5.2 Classification

The evaluation of the different models is going to be based on the overall accuracy and the accuracy in each of the cover types' classes. The overall accuracy was calculated by the average of each correctly predicted type divided by the total number of observations of all 5 cross-validation sets. The accuracy for type i was calculated with:
$1 - \frac{correct\ \#\ predictions_i - total\ \#\ predictions_i}{total\ \#\ predictions_i}, \qquad i = 1, 2, ..., 7.$

This section also presents the time it took to train the models. It should be noted that the time is presented approximately and also depends on the computer used to run it. Here three different computers were used to train different models simultaneously and all of them handled the large data differently time wise.

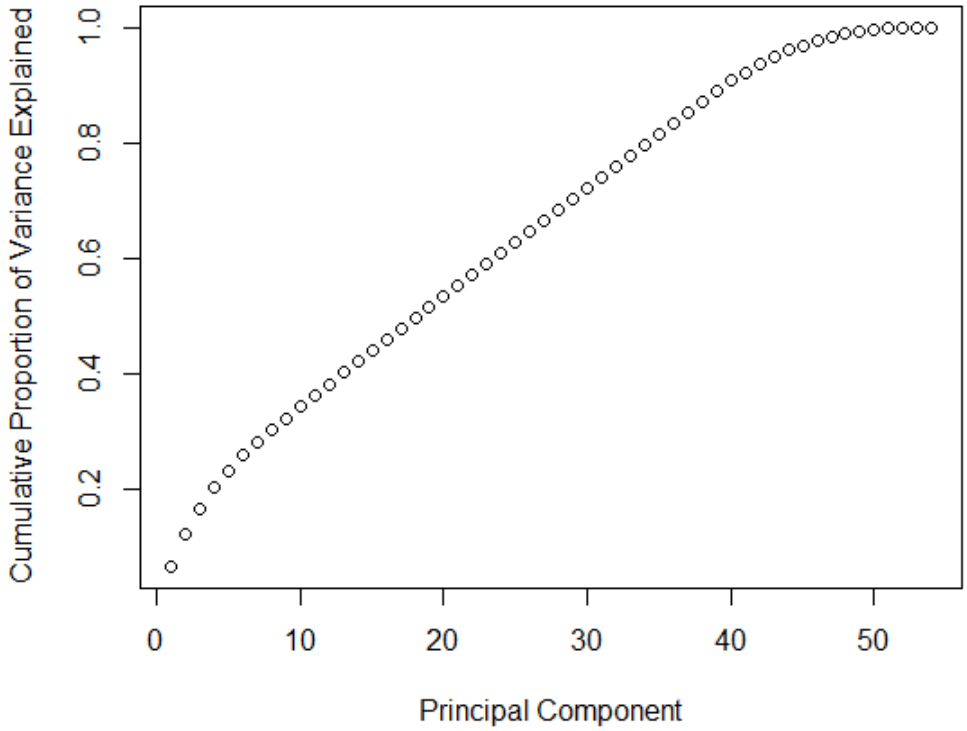

Figure 5: A cumulative plot of the proportion of the variance for the variables mentioned section 4 each PCA component explain.

5.2.1 Support Vector Machine

The SVM took more computational time than the other classifier. Table 2 shows the average results for the SVM predictions based of the 5-cross-validation. The full model gave an overall prediction percentage of 89 %. The Lasso variable selection method with SVM performed overall worse with a decrease from 89 to about 82 % from the full model.

The exclusion of the slope variable for the Random Forest variable importance selection yielded an overall accuracy of 89.1 %. Although slightly higher than the full model's accuracy being 89.0 %, the possibility of randomness can manipulate the predictions slightly, making one unable to assume that the RF selection performed with higher accuracy than the full model.

The full model performed decently on the data, given the fact of imbalance in the

Table 2: Overall and forest type accuracy (in percentage) for SVM.

Accuracy	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	89.0	85.4	92.4	91.0	84.9	67.3	79.9	82.5
Lasso	81.8	87.1	88.6	76.5	74.9	10.0	62.9	60.4
RF selection	89.1	85.5	92.5	91.3	85.0	67.4	80.2	92.7
PCA	82.1	77.2	88.4	87.3	70.4	30.2	58.4	83.4

outcome categories for the dataset. The lowest accuracy type was in Type 5 which was 67.3 % accurate. The Lasso selection had a higher Type 1 accuracy than the full model, but the rest of the Type predictions was predicted notable worse than the full model. The Type 5 prediction decreased from an accuracy of 67.3 % to 10.0 %.

The Random Forest selection did perform slightly better than the full model in all of the forest type categories. While the 0.1 % increase in the overall between the full model and RF selection did not necessarily have to be evidence of a significantly higher increase, the increased Type accuracies might show evidence of the RF selection slightly outperforming the full model.

The PCA with SVM could not provide any noteworthy results. Most of the Type predictions performed notably worse than the full model, and those who did perform with higher accuracy only did it marginally. Although it needs to be noted that the PCA's decrease in accuracy was not nearly as big as the Lasso selection's accuracy decrease from the full model.

5.2.2 Naive Bayes

The Naive Bayes Classifier was without a doubt the fastest of these three methods - the training of the models finished in a matter of seconds, making it irrelevant to evaluate the difference in speed between the four different Naive Bayes (NB) classification models. As can be seen in Table 3, the overall percentage for the full model was 66 %. Lasso selection had a 0.7 % increase in accuracy compared to the full model, but that could be explained in the randomness of the dividing of the 5-cross-fold for the test and training parts of the datasets.

Random Forest variable selection had a slightly higher accuracy from 66.1 to 66.6 %, meaning it performed better overall for the full model but not Lasso selection. The Principal Component Analysis had the lower result overall for the Naive Bayes Classifier. It almost had a 20 % decrease in accuracy from the full model to the PCA, giving it 48 % accuracy.

The classifier showed evidence of performing worse in Type 4, 5, 6 and 7 compared to

Table 3: Overall and forest type accuracy (in percentage) for NB.

Accuracy	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.1	67.5	69.0	78.4	39.0	21.3	30.8	45.1
Lasso	66.8	66.6	72.1	71.5	35.4	15.7	31.0	43.9
RF selection	66.6	67.7	69.8	78.9	39.1	19.8	29.2	45.7
PCA	48.0	78.9	25.5	45.9	69.5	9.9	23.3	79.2

the other 3 forest types, indicating that it performs worse on imbalanced data. Even though the full model and Lasso selection method had a similar overall accuracy, Lasso had a lower prediction accuracy on the categories with fewer observations.

It needs to be mentioned that even though the Random Forest selection had a worse percentage overall than the Lasso method, it did perform better in predicting the accuracy in Type 4 than both the full model and the Lasso selection with its 39.1 %. However, it did perform worse on the other types with low observations, rather showing evidence of that the *slope* variable was not significantly important in explaining type 4 covers.

As previously mentioned, the PCA had the lowest accuracy overall. A noteworthy observations is that it did achieve the highest accuracy on the Type 1, 4 and 7 - but at the same time it also achieved the lowest accuracy on Type 2, 3, 5 and 6. This mean that, for the NB Classifier, PCA both had either the highest or lowest accuracy when it came to predicting a specific cover type which do not seem to depend on class imbalance.

5.2.3 Random Forest classifier

Each Random Forest classifier (RFC) training model took approximately 3-4 hours each to finish, making the whole set to finish in about 15-20 hours. The variable selection methods did not seem to have any notable impact on the time.

As can be seen in Table 4, the full model had an overall accuracy of 84.6 %. Comparing the full model to the Lasso selection model shows clearly how Lasso performed worse with a notably lower accuracy of 75.2 instead of 84.6 %.

The Random Forest selection did not seem to affect the overall accuracy with any notable effect with only 0.4 % difference, which might not be surprising due to the similarities of the RFC and RF variable selection - the variable which the RF selection deemed to be least important was most likely the same variable the RFC put the least weight on.

The most notable result for the RFC was the PCA implementation. It did not only give the highest accuracy of the variable selection methods, but even performed better than the full model with its 94.7 %, compared to the full model's 84.6. The full Ran-

Table 4: Overall and forest type accuracy (in percentage) for RFC.

Accuracy	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.6	87.1	89.1	80.0	76.9	11.8	63.4	58.9
Lasso	75.2	78.9	80.6	71.2	45.2	0.46	32.7	45.4
RF selection	84.2	87.1	88.6	76.5	74.9	10.0	62.9	60.4
PCA	94.7	93.7	96.7	95.0	77.9	77.0	86.0	94.5

dom Forest classifier model follows the same pattern as most of the other classifiers and their selection methods and perform worse on Type 5, which might indicate that the attributes used to predict the cover types do not work very well for the fifth forest cover type. Otherwise the full RFC model performed decently even in the categories with a low percentage of the total observations.

The Lasso variable selection performed worse in every way, compared to the full RFC model. It did not achieve as high accuracy overall, but neither did it do any better for any of the categories - it even had the lowest accuracy prediction of every method in this thesis with 0.46 % accuracy for Type 5.

While RFC with Lasso performed worse in every way, compared to the full model, the RFC with PCA implementation performed better in every way. The overall accuracy was higher with almost 10 %, it handled the imbalanced outcome categories with higher accuracy. For Type 5, where the full model predicted 11.8 %, the PCA implementation managed to increase the prediction percentage to 77.0 %.

A last minute sensitivity analysis for the PCA accuracy can be seen in Figure 6 (Appendix) - note that cross-validation has not been performed on the accuracies. It shows evidence of supporting the theory of the NB classifier depends too much on independence to perform well with the PCA for this data, presented in the next section: *Discussion and conclusions*. The SVM seem to get approximately a 5 % accuracy increase going from 80 % of the variance to 100 %, but it still does not come close to its full model's accuracy. A notable observation is to see that one might be able to exclude approximately 44 of 54 components for the RF classifier and still retain most of the accuracy for the full model.

Due to that the Random Forest classifier with PCA implementation had the highest accuracy, a confusion matrix of the accuracy for the predicted and true values can be seen in Table 6 in *Discussion and conclusions*, section 6.

6 Discussion and conclusions

Table 5 summarizes the results of the three different classification methods together with various variable selection approaches to ease the comparison between them.

There was no clear answer to whether the accuracy was improved by using the full model or by any feature selection methods. The Lasso variable selection method did perform worse in both the Random Forest classifier and Support Vector Machine models, the Naive Bayes model performed slightly better with the Lasso selection model. An explanation might be the correlation or dependence between the variables - as can be seen in Appendix, Table 10 show the correlation between the continuous variables in the dataset. The Lasso selection forces itself to select between a group of independent variables that has a high correlation between each other, even if both of those variables might be needed to explain different variations in the data. This might also explain why the NB method did not get as heavily affected by the Lasso method as the RFC and SVM, due to that the NB has the assumption of independence between the predictors.

Table 5: Overall and forest type accuracy for all three methods.

SVM	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	89.0	85.4	92.4	91.0	84.9	67.3	79.9	82.5
Lasso	81.8	87.1	88.6	76.5	74.9	10.0	62.9	60.4
RF selection	89.1	85.5	92.5	91.3	85.0	67.4	80.2	92.7
PCA	82.1	77.2	88.4	87.3	70.4	30.2	58.4	83.4

NBC	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.1	67.5	69.0	78.4	39.0	21.3	30.8	45.1
Lasso	66.8	66.6	72.1	71.5	35.4	15.7	31.0	43.9
RF selection	66.6	67.7	69.8	78.9	39.1	19.8	29.2	45.7
PCA	48.0	78.9	25.5	45.9	69.5	9.9	23.3	79.2

RFC	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.6	87.1	89.1	80.0	76.9	11.8	63.4	58.9
Lasso	75.2	78.9	80.6	71.2	45.2	0.46	32.7	45.4
RF selection	84.2	87.1	88.6	76.5	74.9	10.0	62.9	60.4
PCA	94.7	93.7	96.7	95.0	77.9	77.0	86.0	94.5

The Random Forest variable selection performed well with the purpose of a variable (feature) selection. For all three of the methods it had a rather unchanged accuracy while managing to exclude a predictor, making the time it took to train the model slightly decline. The attractive attribute of Random Forest selection that make it less sensitive to extreme values was not particularly important here, since this data lacked

of such values - although, the other attribute of the Random Forest being hard to overfit might have played a big part.

The PCA differed a bit from the Lasso and RF selection. Lasso and RF selection gave a recommendation of which variables to exclude based on their criteria, but the PCA reworks the variables to principal components by putting a sort of weight on each of them, depending on their importance. It then allows one to exclude the components based of their proportion of the explained variance (as could be seen in Figure 5). While one might think that the NB classifier would perform better due to the PCA making the used attributes uncorrelated, it does not necessarily mean that they are independent, rather the PCA method in this case distorted the accuracy for the NB method.

The SVM classifier did not seem to go well with the PCA, either. One reason might be that the SVM kernel computation is not feature wise. PCA reduces the dimensional space, but the SVM does not necessarily need its dimensional space reduced, it is also a risk of the PCA missing out important attributes by transferring the variables into components.

Table 6: Confusion matrix for the true (T) and predicted (P) observations from the RFC model with PCA, row percentage in the parenthesis and the diagonally correct predictions are marked gray.

T \ P	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Sum**
1	198,501	12,554	14	0	89	43	639	**211,840**
	(93.7)	(5.9)	(0.0)	(0.0)	(0.0)	(0.0)	(0.3)	**(100.0)**
2	7,735	273,905	678	1	437	456	89	**283,301**
	(2.7)	(96.7)	(0.2)	(0.0)	(0.2)	(0.2)	(0.0)	**(100.0)**
3	6	596	33,969	166	34	983	0	**35,754**
	(0.0)	(1.7)	(95.0)	(0.5)	(0.1)	(2.7)	(0.0)	**(100.0)**
4	0	1	454	2,140	0	152	0	**2,747**
	(0.0)	(0.0)	(16.5)	(77.9)	(0.0)	(5.5)	(0.0)	**(100.0)**
5	159	1,858	117	0	7,309	50	0	**9,493**
	(1.7)	(19.6)	(1.2)	(0.0)	(77.0)	(0.5)	(0.0)	**(100.0)**
6	18	676	1,621	87	21	14,944	0	**17,367**
	(0.1)	(3.9)	(9.3)	(0.5)	(0.1)	(86.0)	(0.0)	**(100.0)**
7	970	147	0	0	2	0	19,391	**20,510**
	(4.7)	(0.7)	(0.0)	(0.0)	(0.0)	(0.0)	(94.5)	**(100.0)**
Sum	**207,389**	**289,737**	**36,853**	**2,394**	**7,892**	**16,628**	**20,119**	**581,012**
	(35.7)	**(49.9)**	**(6.3)**	**(0.4)**	**(1.4)**	**(2.9)**	**(3.5)**	**(100.0)**

Using the PCA and RF Classifier together with this data seems to work very well.

A reason for this might be that, while the Random Forest classifier compared to Decision Trees is much less likely to risk overfitting, the risk still exists. Overfitting could be a problem, especially since the data consists of two factor predictors, one of which has a high dimension. This makes it $40 \times 4 = 160$ unique combinations for 7 different outcomes, which gives the risk of overfitting. However, there are no factor types in a PC format, making all components as continuous instead and with this greatly reduce the risk of overfitting. In Table 6 it can be seen how specific categories have a tendency to mismatch with another specific category, e.g. the RFC with PCA model had for tree type 5 mismatched about 19.6 % into Type 2 instead

The Naive Bayes Classifier performed worse than the Support Vector Machine and Random Forest Classifier. With this data it might be hesitant to use it, due to the NB's naive assumption of the predictors being independent, which is doubtful when e.g. three of the variables are the shade index of different times of the day on the same spot. Even though the NB model was trained significantly faster than both the RFC and SVM, it did not make up for its low prediction accuracy.

For this data, the recommended model is the *Random Forest Classifier with Principal Component Analysis*, since it had a higher accuracy in every way compared to all other methods and feature selections and also trained its models notably faster than the SVMs. On the second place is the SVM Classifier - of course, this is only if the user is in no hurry due to the significant increase in time it take to train a SVM model compared to a RF.

Since one focus was to achieve as high accuracy as possible, a Random Forest classifier with the full PCA selection from a 5-fold cross-validation was applied and yielded an accuracy of 95.4 %, making this the highest achieved accuracy.

Albeit taken slightly out of context, the general saying that "*better data often beats better algorithms*" seems to be rather suiting here.

6.0.1 Further studies

It was interesting to observe how well the PCA performed with the RF model. Further studies could be to look at their performance with other datasets, especially those with attributes of high dimensions and many factors.

One might try the Independent Component Analysis (ICA) on the dataset used here to get the components independent and not just uncorrelated, especially to see if the NB performs better when making the attributes to independent components instead of uncorrelated.

7 Appendix

7.1 Complete tables of accuracy for the different methods

Table 7: Overall and forest type accuracy (in percentage) for SVM.

Average	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	89.0	85.4	92.4	91.0	84.9	67.3	79.9	82.5
Lasso	81.8	87.1	88.6	76.5	74.9	10.0	62.9	60.4
RF selection	89.1	85.5	92.5	91.3	85.0	67.4	80.2	92.7
PCA	82.1	77.2	88.4	87.3	70.4	30.2	58.4	83.4

Set 1	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	88.9	85.4	92.5	90.9	84.6	66.3	79.3	92.2
Lasso	81.7	87.4	88.3	77.1	75.0	11.8	64.5	60.9
RF selection	89.0	85.5	92.5	91.4	84.8	66.6	79.6	92.3
PCA	82.0	77.2	88.2	87.6	69.1	30.9	57.7	82.4

Set 2	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	89.1	85.5	92.5	91.5	85.0	68.3	80.0	92.8
Lasso	81.9	87.8	89.0	78.5	73.3	5.8	64.9	61.1
RF selection	89.2	85.6	92.8	91.2	85.7	67.7	79.6	92.6
PCA	82.4	77.4	88.7	87.4	70.8	30.3	59.5	82.8

Set 3	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	89.0	85.5	92.5	91.5	85.0	68.3	79.9	92.8
Lasso	81.8	86.8	88.3	77.6	73.5	10.5	62.2	64.5
RF selection	89.1	85.5	92.6	91.4	84.8	68.0	80.3	93.1
PCA	81.9	76.6	88.2	87.8	72.5	29.8	58.2	83.6

Set 4	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	88.8	85.2	92.3	91.1	82.7	67.4	80.4	92.2
Lasso	81.8	86.5	88.8	76.3	72.0	12.0	61.7	58.9
RF selection	89.0	85.3	92.4	91.5	82.9	67.8	81.2	92.5
PCA	82.1	77.2	88.3	86.9	68.6	30.4	58.6	84.0

Set 5	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	88.9	85.4	92.3	90.6	86.6	67.4	80.4	93.2
Lasso	81.8	86.9	88.7	73.2	80.6	10.0	61.4	56.6
RF selection	89.0	85.5	92.3	90.9	86.8	67.0	80.2	93.2
PCA	82.2	77.4	88.3	87.1	71.1	29.4	58.2	84.4

Table 8: Overall and forest type accuracy (in percentage) for NB.

Average	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.1	67.5	69.0	78.4	39.0	21.3	30.8	45.1
Lasso	66.8	66.6	72.1	71.5	35.4	15.7	31.0	43.9
RF selection	66.6	67.7	69.8	78.9	39.1	19.8	29.2	45.7
PCA	48.0	78.9	25.5	45.9	69.5	9.9	23.3	79.2

Set 1	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.0	67.5	68.8	78.7	40.4	20.1	31.3	44.9
Lasso	66.7	66.6	72.0	71.9	35.1	14.4	30.9	43.8
RF selection	66.5	67.7	69.7	79.0	39.7	18.8	29.7	45.5
PCA	48.1	78.7	25.6	46.6	69.8	11.9	25.3	79.0

Set 2	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.2	67.8	69.1	78.1	37.4	21.4	31.1	44.0
Lasso	66.8	66.8	72.2	71.3	32.9	15.9	31.4	42.6
RF selection	66.6	67.9	69.9	78.4	38.3	19.3	29.6	44.5
PCA	48.0	78.9	25.4	44.9	71.7	10.2	23.6	78.9

Set 3	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.1	67.5	69.0	78.0	38.0	22.1	30.7	45.1
Lasso	66.7	66.5	72.1	71.3	36.8	16.1	30.7	43.9
RF selection	66.6	67.7	69.8	78.9	39.2	20.3	28.9	45.5
PCA	48.1	78.8	25.8	46.8	68.2	7.7	21.2	79.6

Set 4	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.1	67.3	68.9	78.9	39.5	22.0	30.6	45.9
Lasso	66.7	66.5	72.1	71.3	35.2	16.7	31.1	44.6
RF selection	66.5	67.5	69.8	78.9	37.7	20.6	29.0	46.5
PCA	47.9	79.0	25.3	46.2	65.3	9.3	23.8	79.0

Set 5	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	66.2	67.7	69.0	78.4	39.8	20.7	30.1	45.7
Lasso	66.9	66.8	72.2	71.6	37.1	15.8	31.2	44.4
RF selection	66.7	67.9	69.9	79.1	40.5	19.6	28.8	46.7
PCA	48.0	79.0	25.5	44.9	72.0	10.7	22.5	79.3

Table 9: Overall and forest type accuracy (in percentage) for RF.

Average	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.6	87.1	89.1	80.0	76.9	11.8	63.4	58.9
Lasso	75.2	78.9	80.6	71.2	45.2	0.46	32.7	45.4
RF selection	84.2	87.1	88.6	76.5	74.9	10.0	62.9	60.4
PCA	94.7	93.7	96.7	95.0	77.9	77.0	86.0	94.5

Set 1	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.4	87.6	88.7	79.3	75.0	11.4	64.3	55.0
Lasso	75.1	79.2	80.4	68.4	43.3	1.27	35.0	46.2
RF selection	84.3	87.4	88.3	77.1	75.0	11.8	64.5	60.9
PCA	94.7	93.8	96.6	95.1	78.1	79.0	85.7	94.1

Set 2	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.8	86.9	89.5	81.0	75.7	6.8	63.0	59.1
Lasso	75.2	79.0	80.5	70.9	43.4	0.27	34.0	43.7
RF selection	84.8	87.8	89.0	78.5	73.3	5.8	64.9	61.1
PCA	94.9	93.9	96.9	95.3	80.1	76.4	86.6	94.7

Set 3	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.5	86.7	89.1	78.8	79.3	12.8	62.1	61.4
Lasso	75.3	78.7	81.0	71.2	47.8	0.15	31.6	45.2
RF selection	84.1	86.8	88.3	77.6	73.5	10.5	62.2	64.5
PCA	94.7	93.6	96.7	95.2	77.4	76.8	85.4	95.0

Set 4	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	84.5	87.1	89.0	79.8	71.5	14.9	62.4	57.9
Lasso	75.2	78.4	80.8	74.3	46.8	0.21	31.3	44.7
RF selection	83.9	86.5	88.8	76.3	72.0	12.0	61.7	58.9
PCA	94.6	93.5	96.7	95.1	76.0	76.4	86.1	94.4

Set 5	Overall	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7
Full model	85.0	87.2	89.1	81.0	83.2	13.1	65.3	61.2
Lasso	75.3	79.2	80.3	71.4	44.9	0.43	31.4	47.5
RF selection	83.9	86.9	88.7	73.2	80.6	10.0	61.4	56.6
PCA	94.6	93.7	96.5	94.3	77.9	76.5	86.5	94.6

7.2 Table of correlation

Table 10: Table of correlation for the continuous variables. Notable correlations are marked bold.

Variables	1	2	3	4	5	6	7	8	9	10
(1) elevation	1.000									
(2) aspect	0.016	1.000								
(3) slope	-0.243	0.079	1.000							
(4) HDTH	0.306	0.017	-0.011	1.000						
(5) VDTR	0.093	0.070	0.275	**0.606**	1.000					
(6) HDTR	0.366	0.025	-0.216	0.072	-0.046	1.000				
(7) hillshade9am	0.112	**-0.579**	-0.327	-0.027	-0.166	0.034	1.000			
(8) hillshadenoon	0.206	0.336	**-0.527**	0.047	-0.111	0.189	0.010	1.000		
(9) hillshade3pm	0.059	**0.647**	-0.176	0.052	0.035	0.106	**-0.780**	**0.594**	1.000	
(10) HDTFP	0.148	-0.109	-0.186	0.052	-0.070	0.332	0.133	0.057	-0.048	1.000

7.3 Accuracy runs with cumulative principal components

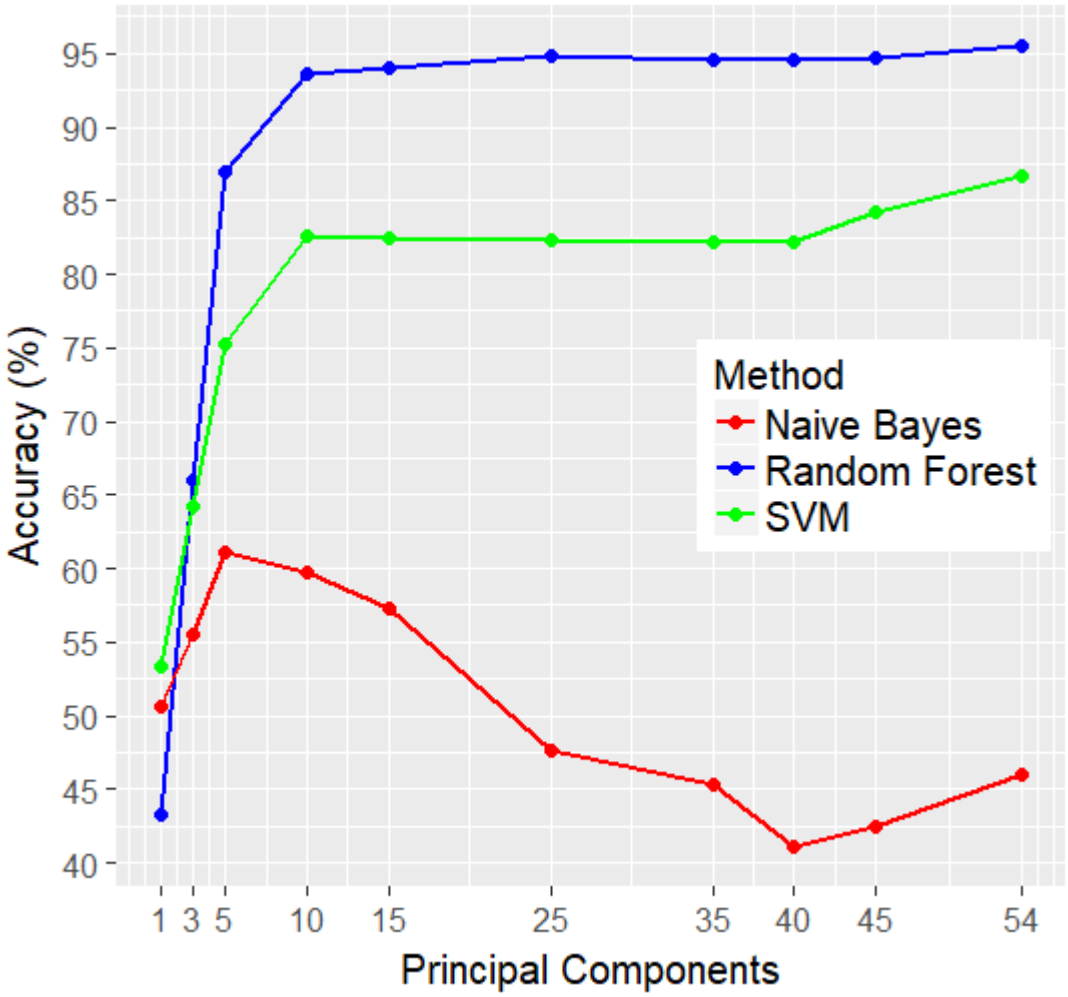

Figure 6: A plot of the overall accuracy based on the number of prinicipal components included for the Naive Bayes, Random Forest and SVM classifiers.

8 References

Ahmed, S. E. (2013), *Penalty, Shrinkage and Pretest Strategies: Variable Selection and Estimation*, Springer Publishing Company, Incorporated.

Alpaydin, E. (2010), *Introduction to Machine Learning*, 2nd edn, The MIT Press.

Anderson, J., Michalski, R., Carbonell, J. & Mitchell, T. (1986), *Machine Learning: An Artificial Intelligence Approach*, number v. 2 *in* 'Machine Learning: A Multistrategy Approach', Morgan Kaufmann.

Bagnall, A. J. & Cawley, G. C. (2003), Learning classifier systems for data mining: A comparison of xcs with other classifiers for the forest cover data set, *in* 'Proceedings of the International Joint Conference on Neural Networks, 2003.'.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blackard, J. A. & Dean, D. J. (1999), 'Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables', *Computers and Electronics in Agriculture* **24**.

Campbell, C. & Ying, Y. (2011), *Learning with Support Vector Machines*, Synthesis lectures on artificial intelligence and machine learning, Morgan & Claypool.

Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M. & Lin, C.-J. (2010), 'Training and testing low-degree polynomial data mappings via linear svm', *J. Mach. Learn. Res.* **11**, 1471–1490.

Crain, K. & Davis, G. (2014), 'Classifying forest cover type using cartographic features', Published report.

Franco, L. & Cannas, S. A. (2000), 'Generalization and selection of examples in feed-forward neural networks', *Neural Computation* **12**(10), 2405 – 2426.

Fu, Z., Robles-Kelly, A. & Zhou, J. (2010), 'Mixing linear svms for nonlinear classification', *IEEE Transactions on Neural Networks* **21**(12), 1963–1975.

Gama, J. a., Rocha, R. & Medas, P. (2003), Accurate decision trees for mining high-speed data streams, *in* 'Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '03, ACM, New York, NY, USA, pp. 523–528.

Gupta, A., Zalani, A. & Sawhney, H. (2015), Classifying forest categories using cartographic variables, Technical report, Indian Institute of Technology.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2014), *An Introduction to Statistical Learning: With Applications in R*, Springer Publishing Company, Incorporated.

Karampatziakis, N. & Mineiro, P. (2013), 'Discriminative features via generalized eigenvectors', *arXiv preprint arXiv:1310.1934* .

Lantz, B. (2015), *Machine Learning with R*, 2nd edn, Packt Publishing.

Leal, Nallig, L. E. & Sanchez, G. (2015), 'Marine vessel recognition by acoustic signature', *ARPN Journal of Engineering and Applied Sciences* **10**(20).

Lichman, M. (2013), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Louppe, G., Wehenkel, L., Sutera, A. & Geurts, P. (2013), Understanding variable importances in forests of randomized trees, *in* 'Proceedings of the 26th International Conference on Neural Information Processing Systems', NIPS'13, Curran Associates Inc., USA, pp. 431–439.
URL: *http://dl.acm.org/citation.cfm?id=2999611.2999660*

Ma, Y. & Guo, G. (2014), *Support Vector Machines Applications*, SpringerLink : Bücher, Springer International Publishing.

Menard, S. (2002), *Applied Logistic Regression Analysis*, number nr. 106 *in* 'Applied Logistic Regression Analysis', SAGE Publications.

Michie, D., Spiegelhalter, D. J., Taylor, C. C. & Campbell, J., eds (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, Upper Saddle River, NJ, USA.

Perseus documentation (2015), 'Classification parameter optimization'. [Online; accessed May 10, 2017].
URL: *http://www.coxdocs.org/doku.php?id=perseus:user:activities:matrixprocessing:learning:classificationparameteroptimization*

R Core Team (2017), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Ridgeway, G. (2002), 'Looking for lumps: Boosting and bagging for density estimation', *Computational Statistics & Data Analysis* **38**(4), 379–392.

Sjöqvist, H. (2017), Calibration of european call options with time varying volatility : A bayesian and frequentist analysis, Master's thesis, Örebro University.

Sug, H. (2010), 'The effect of training set size for the performance of neural networks of classification', *WSEAS Trans Comput* **9**, 1297–306.

Venables, W. & Ripley, B. (2003), *Modern Applied Statistics with S*, Statistics and Computing, Springer New York.

Wolfe, P. (1961), 'A duality theorem for non-linear programming', *Quarterly of Applied Mathematics* **19**(3), 239–244.
URL: *http://www.jstor.org/stable/43635235*

Zhang, H. (2004), The optimality of naive bayes, *in* V. Barr & Z. Markov, eds, 'Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)', AAAI Press.