Uppsala University
Department of Statistics
Bachelor's Thesis
Autumn 2015

UPPSALA
UNIVERSITET

Supervised Learning Techniques:

A comparison of the Random Forest and the Support Vector Machine

Jonni Fidler Dennis and Lukas Arnroth

Abstract

This thesis examines the performance of the support vector machine and the random forest models in the context of binary classification. The two techniques are compared and the outstanding one is used to construct a final parsimonious model. The data set consists of 33 observations and 89 biomarkers as features with no known dependent variable. The dependent variable is generated through k-means clustering, with a predefined final solution of two clusters. The training of the algorithms is performed using five-fold cross-validation repeated twenty times. The outcome of the training process reveals that the best performing versions of the models are a linear support vector machine and a random forest with six randomly selected features at each split. The final results of the comparison on the test set of these optimally tuned algorithms show that the random forest outperforms the linear kernel support vector machine. The former classifies all observations in the test set correctly whilst the latter classifies all but one correctly. Hence, a parsimonious random forest model using the top five features is constructed, which, to conclude, performs equally well on the test set compared to the original random forest model using all features.

Keywords: machine learning, biomarkers, cross-validation, receiver operating characteristic, k-means clustering, feature selection, binary classification

Supervisor: Inger Persson

Table of contents

1. Introduction

Enormous amounts of data are continuously stored and accumulated. A lot of individuals' everyday choices and acts are somehow registered and stockpiled. To take a few examples, data storing activities range from concerning financial transactions, shopping patterns and Internet behaviour to more general records about the population, demographics, socioeconomics and health. There are also huge quantities amassed concerning geological data, weather and satellite data. The sheer amount of data is in many cases unmanageable, hampering the possibility of making relevant interpretations. As a result, potential areas of data utilization are never realized (Bramer, 2007, Chapter Introduction).

Access to large amounts of data in itself is not very useful unless it is possible to analyse the data, extract information from it and utilize this information in a meaningful way. In many cases, depending on what type of data is available and what the aim of the analysis of the data is, it may be difficult to identify and thoroughly comprehend the process that explains the data at hand. Instead, if the possibility exists to detect and uncover certain patterns in the data, one might be able to construct a sufficient approximation of some part of this process. Thereby, using this approximation, it is possible to attain increased understanding of the underlying process that generates the data and predictions can be made. This uncovering of patterns is in essence what the field of machine learning is about (Alpaydin, 2010, Chapter 1).

Machine learning is a scientific field with origins in computer science, artificial intelligence and statistics. The main idea within the field is to uncover the mechanisms by which an explicit task is performed. In other words, a computer is to learn how to correctly perform a specified assignment by identifying certain patterns present in a set of data (Von Luxburg & Schölkopf, 2008). Machine learning is applied in a wide array of areas such as pattern and speech recognizing tasks, analysis of consumer behaviour, predicting credit losses and in bioinformatics (Alpaydin, 2010, Chapter 1).

Statistical learning theory is a subfield in statistics that has emerged from the field of machine learning (James, Witten, Hastie & Tibshirani, 2013, Chapter 1). As in the case of machine learning, the term learning here, in short, concerns the ability to identify and make sense of patterns and trends in large amounts of data. The majority of the statistical learning problems can be categorized into two groups; either supervised or unsupervised learning (Hastie, Tibshirani & Friedman, 2009, Chapter 1).

In supervised learning the aim is to use a set of variables in order to predict an outcome. The term supervised comes from fact that the outcome variable acts as a supervisor that oversees the learning process of how future outcomes are predicted (Camm, Cochran, Fry, Ohlmann & Anderson, 2014, Chapter 6). To be more specific, if an outcome is presented for each variable, then the supervisor states whether or not the outcome is correct (Hastie et al., 2009, Chapter 14). Moving over to the case of the unsupervised learning methods, these techniques are not employed to predict an outcome. Instead, they are only used to uncover and identify existing patterns and relationships in the data (Camm et al., 2014, Chapter 6). As no outcome is predicted, there exists no supervisor that determines whether an outcome is correct or not, hence the name unsupervised learning.

Supervised learning techniques are increasingly being employed within the fields of medicine and bioinformatics. The ability to extract useful information and knowledge from large amounts of data makes the techniques useful when dealing with, for example, the classifying of genes or cancer detection using DNA and gene expression microarrays (see for example Furey et al., 2000; Brown et al., 2000; Statnikov, Wang & Aliferis, 2008). A common characteristic of microarray data sets is that they usually consist of many variables and relatively few observations (Bennett & Campbell, 2000). There are supervised learning techniques especially suitable for analysing data sets with the above-mentioned traits, such as the methods known as the random forest and the support vector machine (Díaz-Uriarte & Andrés, 2006; Bennett & Campbell, 2000).

In this bachelor's thesis, the application of supervised learning techniques is further explored. More specifically, two of these methods are employed and compared, namely the just mentioned random forest and the support vector machine. The aim is to investigate how these methods perform when applied to a specific data set consisting of few observations and many variables. The data set in question has not been previously analysed using these particular techniques and is provided by the company Pharma Consulting Group Clinical Services.

The rest of this thesis is organized in the following way. In the remainder of section 1, the research problem and the aim of the thesis are presented as well as a short presentation of the company Pharma Consulting Group Clinical Services. This is followed by a discussion regarding the limitations of the thesis and motivation behind the choice of methods. Section 2 includes a review of relevant literature and in section 3, the nature of the study as well as the

data set are discussed. Section 4 covers supervised and unsupervised learning with emphasis put on the random forest and the support vector machine. In section 5, the results of this study are displayed, while in section 6 the results are discussed. Lastly, in section 7, the conclusions of this thesis are presented and future research options are considered.

1.1 Research problem and aim of the thesis

On behalf of Pharma Consulting Group Clinical Services, the aim of this bachelor's thesis is to analyse a specific data set they have provided. The main objective is to use the two supervised learning techniques, the random forest and the support vector machine, in order to classify the objects of the data set into either of two different classes. Thereafter, the classification performances of these two techniques are evaluated and compared. However, as the data provided only includes a set of objects with no information regarding any potential class label of these objects, supervised learning methods cannot be applied. A solution is to first use an unsupervised learning method to identify any existent structures and patterns in the data. In this case, the k-means clustering technique is chosen in order to detect any patterns and divide the data into two classes. Once this is achieved, the random forest and support vector machine methods are applicable and are used to classify the objects into the two classes determined by the k-means clustering. In this thesis it is determined which model performs best, within the context of the provided data, and through estimations of feature importance (also referred to as variable importance) a final parsimonious model is chosen. The goal of this bachelor's thesis can thereby be summarized by the following research questions:

- How successful are the two supervised learning techniques, the random forest and the support vector machine, in the case of classifying the specific data set provided and how do the two methods compare to each other?
- In regards to which method performs the best, based on feature importance measures, how does a final parsimonious model perform?

1.2 Pharma Consulting Group Clinical Services

Pharma Consulting Group Clinical Services laid the foundations for this thesis by providing the data set and expressing an interest in having the data analysed using supervised learning methods. The company is a contract research organization (CRO) founded and headquartered in Uppsala, Sweden. Their services range from consultancy and assistance within the areas of clinical trials to facilitating the complete trial process. More concretely, the company provides

services within project management, clinical operations, biometrics, medical writing, error detection and correction (EDC) and auditing and validation (Pharma Consulting Group, 2015).

1.3 Motivation of method and limitations

The two supervised learning techniques, the random forest and the support vector machine, are the main focus of this thesis. There are several supervised learning techniques, besides the random forest and the support vector machine, which can be used as classification tools such as neural networks and nearest neighbour classifiers (Cunningham, Cord & Delany, 2008). However, in this case, only the random forest and the support vector machine are applied. There are several reasons for this. One is that these techniques have shown to be both powerful and accurate tools for machine learning and data mining (Von Luxburg & Schölkopf, 2008). Also, they have been applied within the fields of medicine and bioinformatics with successful results (see for example Bennet & Campbell, 2000; Díaz-Uriarte & Andrés, 2006; Cutler & Stevens, 2006; Jia, Hu & Sun, 2013). Furthermore they both perform well in situations when working with data where there are more variables than observations (Díaz-Uriarte & Andrés, 2006; Bennett & Campbell, 2000). Therefore these two methods appear suited to the task at hand as the forthcoming results supplied in this bachelor's thesis are aimed at being as accurate as possible. Furthermore, the source providing the data set used was interested in how these two techniques, amongst others, would perform in comparison to each other.

A section regarding the unsupervised method k-means clustering is included. The reason is that the application of an unsupervised learning method is, in this case, used in order to enable the use of the supervised methods. However, this section is considerably shorter and less detailed than the sections about the supervised learning as the main objective of the thesis is to apply and compare the supervised learning methods. The unsupervised learning method is only used in order to form two classes based on the multivariate structure present in the data, rather than just randomly simulating a binary dependent variable. The choice of the k-means clustering technique is due to its simplicity and suitability for defining clusters in an unlabelled data set (Hastie et al., 2009, Chapter 13). Moreover, as the aim is to classify a data set into either of two groups, only the random forest and the support vector machine for binary classification is addressed. Therefore, neither regression nor novelty/outlier detection is covered in this thesis. Additionally, multiclass classifications are also overlooked.

2. Literature review

As previously brought to light, the random forest and the support vector machine methods employed in this thesis are in increasing manner being applied in cases within several different scientific fields, bioinformatics being one. One reason is their suitability regarding data sets consisting of relatively few observations and many variables (Díaz-Uriarte & Andrés, 2006; Bennett & Campbell, 2000), where examples of such sets are microarrays (Eckardt, 2004). Microarray technology enables the analysis of thousands of parameters simultaneously in a single experiment (Templin et al, 2002). Microarrays are often used to study expression levels of genes in an organism. These arrays are usually a glass slide where, at specific positions called spots, DNA molecules are located in an ordered fashion. Each spot can contain millions of molecules and each microarray can have thousands of spots (Babu, 2004).

Another reason for the increasing popularity of these methods is that they entail comparatively low computational load and are associated with relatively moderate computational complexity (Breiman, 2001; Hastie et al., 2009, Chapter 12; Karatzoglou, Meyer & Hornik, 2006). Below follows a brief presentation of a few examples of research papers where the two are compared and applied.

For instance, Meyer, Leish and Hornik (2003) compare the support vector machine to 16 other classification methods and nine regression methods, including the random forest method. The performance measures used were the classification error for classification and the mean squared error for the regression. In the case of the classification, 21 data sets were used while nine were used in the regression case. Looking at the simulation procedure, from each data set 100 training sets and 100 test sets were generated. The authors' results indicate that, even though they did not rank in the top in all data sets, the support vector machines performed strongly overall. In the case of the classification, the results were generally good while in the case of the regression, neural networks, projection pursuit regression and the random forest were found to perform better.

In the paper by Statnikov, Wang and Aliferis (2008) the classification prowess of the support vector machine and the random forest is compared. The aim is to find which of these methods

perform best when it comes to microarray-based cancer classification. In the experiment, 22 diagnostic and prognostic data sets are used. The results indicate that the support vector machine performs far better than the random forest, both on average and in most of the microarray data sets.

In addition, Carauna and Niculescu-Mizil (2006) perform empirical comparisons between ten supervised learning methods including the support vector machine and the random forest. The authors use eight performance metrics divided into the three groups: threshold metrics, rank metrics and probability metrics. The algorithms are compared on 11 binary classifications problems consisting of 9366 to 40222 observations where for each test, 5000 training observations are randomly selected and the rest are used as a final test set. The results indicate that before calibration, bagged trees, the random forest and neural nets perform best on average when evaluated by all performance metrics and classification problems. After calibration, the support vector machine is on the same level as neural nets, just behind boosted trees, the random forest and bagged trees.

One example where the support vector machine is applied is in the paper by Furey et al. (2000). The authors develop a method using support vector machines in order to analyse thousands of gene expression measurements generated by DNA microarray experiments. Tissue samples are classified and mislabelled or dubious tissue results are investigated. The microarray expression experiments are conducted using a previously unpublished data set consisting of 97802 DNA clones for 31 tissue samples where the samples are cancerous ovarian tissue, normal ovarian tissue or normal non-ovarian tissue. In order to show generality of the method, the experiments are also performed using previously published data sets. The results the authors present demonstrate that the support vector machines can classify tissue and cell types, however other techniques such as the perceptron algorithm perform comparably. Additionally, the support vector machine can be used to identify mislabelled data.

In a paper by El-Naqa, Yang, Wernick, Galatsanos and Nishikawa (2002) it is investigated how the support vector machine performs as a tool to detect microcalcification clusters in digital mammograms. The aim is to use the support vector machine as a classifier to test if a microcalcification is present in a mammogram or not. The classifier is developed using 76 clinical mammograms containing 1120 microcalcifications, where half the data set is used as

training data and the rest as test data. A ten-fold cross-validation was used for finding a suitable support vector machine classifier. Thereafter, its performance was compared to other methods commonly used for microcalcification detection. In this case, the other methods were IDT, the DoG method, a WD-based method and a TMNN method. The authors' results show that the support vector machine performed better than the other methods and indicate that it is a useful tool for object detection in medical imaging.

In a research paper by Díaz-Uriarte and De Andres (2006) the random forest technique is evaluated when used for classification of microarray data and variable selection (in this case gene selection). Generally, in gene expression studies, researchers try to detect the smallest possible set of genes while upholding good predictive performances. Nine microarray data sets are used as well as simulated data and the results of the random forest are compared to other methods used for classification and gene selection, including the linear kernel support vector machine. The results show, both when using the microarrays and the simulated data, that the random forest performs similarly to the other methods when it comes to classification. In the case of gene selection, the random forest often picks a smaller set of genes compared to other variable selection methods whilst keeping desired predictive performance. The authors conclude that due to the good performance of the random forest, the method is suitable when it comes the classification and variable selection of microarray data.

The above mentioned are but a few of the numerous examples of research papers where results show that the two methods perform well, both in the cases of classification and regression. In addition, to summarize, when compared, both with each other and with other methods, the support vector machine and the random forest generally perform comparably well.

In the light of the several promising prior results of the applications of the support vector machine and the random forest methods, expectations are high that they both will perform well as classifiers of the particular data set used in this thesis. However, which of them that is the superior classifier in this regard is impossible to form an opinion about, due to their, in general, similar capabilities and comparable classification prowess.

3. Nature of the study and the data set

This is an empirical study in the sense that it is based on real-world data. However, as many of the details regarding the data set are unknown, no results regarding the actual meaning of neither the variables nor the classifications are presented. As a consequence, any conclusions drawn solely concern the performance of the statistical methods employed. As mentioned earlier, the emphasis is therefore on comparing the classification prowess of the two supervised learning methods albeit the application is on real-world data.

3.1 Data set

The original data set provided consists of 33 observations and 92 different variables. The variables are protein biomarkers, whose quantities or levels have been determined in the 33 observations. The values of the levels have all been transformed to the $\log_2$-scale, hence, they are continuous numerical variables measured on the same scale. The reason for using the $\log_2$ – scale has not been disclosed. However, usually, in the case of microarrays, the data are transformed into the $\log_2$ – scale as the magnitude of the range of the data is decreased and the data generally becomes more normally distributed. Furthermore, interpreting the $\log_2$ – scale is straightforward; a one-unit change in the $\log_2$ – scale corresponds to a doubling in the original scale (Ballman, 2008). The data set used in this thesis is not exactly a microarray, however, the reasoning behind the use of the $\log_2$ – scale should still be applicable.

Further details regarding the data, such as for example selection criteria and selection method of the observations as well as the choice of protein biomarkers, are unknown.

The data set used in this thesis only includes 89 variables, instead of the original 92. The reason for this is that, in the cases of the three omitted ones, the protein levels present in the observations were so low that they were immeasurable. Therefore these three variables are excluded from the analysis.

3.2 Data processing

In the case of the support vector machine, the features of each input object have to be represented as a vector of real numbers, meaning that any categorical ones have to somehow be changed into numerical data. It is also of essence to scale the variables appropriately. Otherwise, there is a risk that if some of an object's attributes are measured in large numerical ranges and some are not, the former might dominate the latter. Examples of recommended

ranges to linearly scale to are [0, 1] or [−1,+1] (Hsu, Chang & Lin, 2003). Therefore, the decision was made to scale the $\log_2$ – scaled numerical quantities of the variables to [0, 1].

When implementing the random forest algorithms it is important that the numerical variables are measured on the same scale or are transformed accordingly. Also, if using categorical predictors the variable importance measures are biased for variables with more categories (Breiman, 2001).

As mentioned, the variables in this thesis are appropriately measured on the same scale and are numerical. Hence the data set is, after being scaled to [0, 1] and the omission of the three immeasurable variables, well suited for the application of both supervised learning methods and there is no need for any further adjustments and transformations. Using the same scale also enables reliable comparisons.

4. Methodology

In this section an overview of the statistical methods used in this thesis is presented. First of all, a brief description of the concepts of supervised learning, unsupervised learning, the k-means clustering and classification is given. This is followed by a more thorough presentation of the random forest and the support vector machine, the two supervised learning methods predominantly used and compared in this thesis. Lastly, parts regarding the reasoning behind the choice of method for classifier optimization, the statistical software used, as well as the setup of the algorithms, are included.

4.1 Supervised learning

When dealing with supervised learning problems one has a set of measured inputs, which have some sort of effect on one or more outputs. In other words, the input variables are used to predict the output. The inputs are sometimes referred to as predictors or independent variables while outputs are also called responses or dependent variables. Input variables can either be qualitative, quantitative or both. Depending on the distinction between input variables one uses different methods for prediction. The output variables can either be qualitative or quantitative. Qualitative variables are also called categorical, discrete or factors. When predicting quantitative output variables, the term used is regression while prediction of qualitative outputs is called classification (Hastie et al., 2009, Chapter 2).

4.1.1 Classification

One of the most prominent and commonly studied tasks within supervised learning is the ability to perform correct classifications (Von Luxburg & Schölkopf, 2008). In classification, within supervised learning, there is a division made between what is known as the training data and test data. The training data consist of objects, also called instances, where each object contains a class label and several features. The class label is also known as the target value or category and the features are known as attributes or observed variables. These aspects of the training data are known and, based on them, a model is constructed that acts as a classifier. When a satisfactory classifier has been created it used to classify the test data. The features of the objects of the test data are known and based on these features, the classifier is used to predict the class labels of these objects (Hsu et al., 2003).

In order to illustrate more specifically, consider the case of the basic binary classification. The training data consist of two spaces, the input space X and the output space Y. The input space consists of objects or instances while in the output space one has labels or categories. The

purpose is to find a functional relationship between the spaces X and Y i.e one wants to classify objects/instances into fixed categories/labels. This is achieved by the use of an algorithm that is trained in the sense that, in the training data, the objects are paired with a corresponding category. By this pairing, the algorithm "learns" which objects belong to which category with the aim to discover a way to correctly map the input space into the output space, with as high accuracy as possible. This mapping is then applied to the test data (Von Luxburg & Schölkopf, 2008).

4.2 Unsupervised learning

Unsupervised learning techniques are applied when solely the input objects are known and there is no information regarding any output. The aim is to identify if there is some sort of a meaningful structure present among the input objects and thereafter group them (Von Luxburg & Schölkopf, 2008).

4.2.1 K-means clustering

The k-means clustering method is used to determine clusters in an unlabelled data set (Hastie et al., 2009, Chapter 13). It is a partitioning method that minimizes within-cluster variation to create homogenous clusters. The first step is to decide the wanted number of clusters. Subsequently, the algorithm decides a centre of each cluster and the Euclidian distances between each object and the cluster centres is calculated. An object is allocated to the cluster centre that it is closest to. Then, by calculating the mean values of the objects of each cluster, the clusters' centroids are computed and new centres are formed. Then the objects are reallocated to the new cluster centre they are closest to. This process is repeated until the objects do not change clusters anymore (or some predetermined number of iterations is reached). The objects can change cluster belonging during the clustering process, which is contrary to hierarchical methods. Therefore k-means clustering is known as a non-hierarchical method (Sarstedt & Mooi, 2014, Chapter 9).

Compared to hierarchical methods, k-means clustering is less affected by outliers and irrelevant clustering variables. It is suitable to be applied to large data sets, as the method is less computationally challenging than hierarchical methods. The k-means clustering method is recommended to use on interval or ratio scaled data, although it can be used on ordinal data with the caveat that there might be some distortions (Sarstedt & Mooi, 2014, Chapter 9), thus the k-means clustering is suitable for the purpose of this thesis.

4.3 Terminology

As made clear in previous sections, there are several different terms used referring to the same thing within the learning literature. In order to simplify matters, from here on, the data consist of objects or observations. The input variables or independent variables are referred to as features and the output variables or dependent variables, as being qualitative, are denoted as class or class label. Feature importance and feature selection, used in the context of constructing a parsimonious model, are the terms used in this thesis. Important to note is that these concepts are also frequently referred to as variable importance and variable selection.

4.4 The random forest method

The random forest learning method is credited to Breiman in his influential article Random forests (2001). Its theoretical background rests in the concept of bagging and decision trees. The random forest is an ensemble method, which is a learning algorithm that constructs a set of individual classifiers, also referred to as base learners. The random forest classifies the observations based on how the majority of these base learners classify. This is commonly referred to as voting as the observations are classified based on which decision or vote the majority of the base learners make when classifying (Biau, Devroye & Lugosi, 2008). The classifiers used in the random forest method are classification and regression trees (CART) credited to Breiman amongst others (Breiman, Friedman, Olshen & Stone, 1984). Before the theoretical concepts of the random forest are treated, the CART method needs to be understood in the context of the random forest. As the main interest in this thesis is binary classification, the focus is on classification trees.

4.4.1 CART

CART is a hierarchical method which recursively splits the sample space into mutually exclusive subspaces that are more homogenous with respect to the class label (the dependent variable) than the initial sample space (Breiman et al. 1984, Chapter 2). Lets consider a simple example to illustrate this process. Assume there is a training set with 15 observations in a two-dimensional input space. Denote $\boldsymbol{x}_i = (x_{i1}, x_{i2})$ as the feature vector, with a binary class variable. In figure 1, a scatter plot of the training set is illustrated, where the two possible classes are displayed as squares and circles.

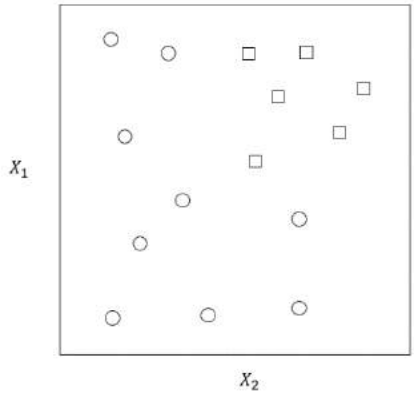

Figure 1. Scatterplot of the training set where squares and circles denote class belonging.

Through recursive binary splitting, a classification tree is constructed, which in full explains the partitioning of figure 1. Figure 2 shows the classification tree along with the illustration of how the space in figure 1 is divided.

The CART algorithm is a so-called greedy algorithm where all the training data are included in the first step, which is the root node (Friedman et al. 2001). The upper oval in figure 2 is the root node containing all observations. The next step is to consider a good split. Looking at equation 1,

$$R_1(j,s) = \{X|X_j \leq b\} \; and \; R_2(j,s) = \{X|X_j > b\} \qquad (1)$$

consider a splitting feature j and split point b. In figure 2, to the left, in step 1 the splitting feature is $X_j = X_2$ and the split point is b on the x-axis. The first split divides the data by a vertical line and two sub-regions, R_1 and R_2, are formed. After another split, this time horizontal, two new regions denoted R_1 and R_2, are created as displayed in figure 2. Each splitting of the data forms two new R_1 and R_2regions, which become more homogenous with respect to the class belonging compared to the regions formed in previous splits.

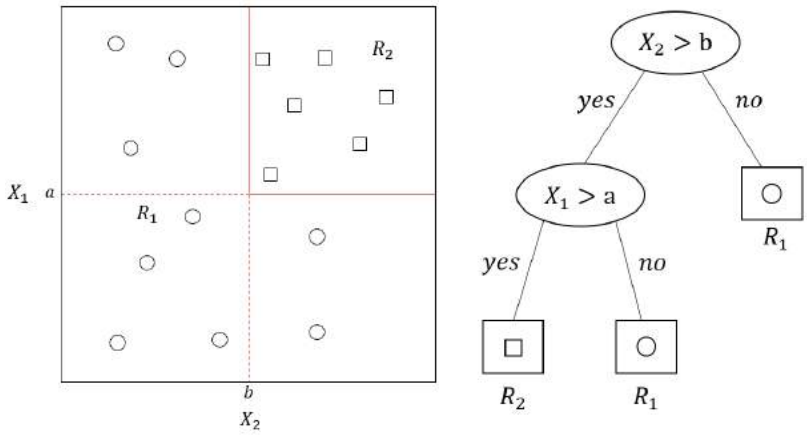

Figure 2. Classification tree and partitioned input space (adopted from Hastie et al., 2009;Chapter 9).

The first split, or partitioning, of the subspace is illustrated as two branches going from the root node. In the decision tree, to the left, the cases where $X_2 > b$ are seen, which is an internal node and to the right, $X_2 \leq b$, which is a leaf node. The terminology of CART is that when a split is made from a node to two others, the resulting nodes from the split are referred to as child nodes whilst the node they were created from is referred to as the parent node. An internal node is one where not all observations respond to a single value of the classification value and a leaf node is one where all the observations correspond to a single classification value. Then the same is done for the subspace to the right of b where the splitting feature is X_1 on point a. The resulting tree has three leaf nodes and fully describes the partitioning of the sample space. Note that often one does not continue partitioning until the tree only contains pure leaf nodes but rather a stopping rule is applied where one stops splitting the nodes when they contain a certain proportion of the sample (Caetano, Aires-de-Sousa, Daszykowski & Vander Heyden, 2005). In this straightforward example no algorithm was really needed to identify a good way to split the data in order to classify the observations. A cluster of squares can be seen in the upper right corner in figure 1, which is fully captured by the classification tree in figure 2. In reality some theory is needed concerning optimal splitting at each node of the tree. When constructing a classification tree this is quantified by an impurity measurement, which measures homogeneity in a node with respect to the class. The best split is found when the impurity function between the parent and two child nodes is minimized (Caetano et. al, 2005). The goodness of the split can be evaluated using equation 2

$$\Delta i(s,t) = i(t) - (P_L i(t_L) + P_R i(t_R)) \qquad (2)$$

where s is the candidate split of a feature x, t the parent node, $i(t)$is the impurity of the node t, p_L and p_R the proportions of objects going to the left or right child nodes, t_L and t_R, and $i(t_L)$, $i(t_R)$ their respective impurities (Breiman et al., 1984, Chapter 4).

Consider node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

be the proportion of class k observations in node m. Observations in node m are then classified to the majority class in that node. In the context of classification and the random forest, the standard impurity measurement is the Gini index (Ishwaran & Kogalur, 2015; Breiman & Cutler, 2004). In the below formula, lower values denote lesser impurity in the node while higher denote more impurity.

$$\textit{Gini index:} \quad \sum\nolimits_{k \neq k'} \hat{p}_{mk}\, \hat{p}_{mk'} = \sum\nolimits_{k=1}^{K} \hat{p}_{mk}\,(1 - \hat{p}_{mk})$$

4.4.2 The random forest algorithms

A formal definition of the random forest is that it is a classifier consisting of a collection of tree structured classifiers $\{h(\boldsymbol{x}, \Theta_k), k = 1, \ldots, K\}$ where Θ_k are independent and identically distributed random vectors, generated for the kth tree by the random feature selection for the splits. Finally each of the K trees cast a vote (a vote should in this context simply be understood as how a particular predictor in the ensemble classifies an observation) for the most popular class at input $\boldsymbol{x}$, where $\boldsymbol{x}$ is a feature vector (Breiman, 2001). The random forest, on one hand uses bootstrap aggregating (bagging) and on the other hand uses random feature selection when building the tree, meaning that a number of features (lower than the total amount of features) is randomly chosen at each split in the construction process. The latter is done to reduce the correlation between the decision trees (Díaz-Uriarte et al. 2006). Bagging is proven to be very successful when aggregating unstable learners such as CART. The term unstable refers to the hierarchical nature of CART, where changes early in the split decision lead to very different trees (Breiman, 1996). The reasoning for lowering the correlation between the trees is the reduction of variance when summing uncorrelated unbiased predictors (Breiman, 2001).

Next in this section a description of the bagging concept is presented. Assume there is a training set $L = \{(y_n, \boldsymbol{x}_n\}, n = 1, \dots, N\}$, where $\boldsymbol{x}_n$ is a feature vector. Furthermore a predictor $\varphi(\boldsymbol{x}, L)$ is needed. Now assume a situation where instead of only having L, there are $\{L^{(B)}\}$ bootstrap samples following the same underlying distribution as L. An ensemble method, when working with sorting observations to a class $j \in \{1, \dots, J\}$, would then consist of forming B predictors $\varphi(\boldsymbol{x}, L^{(B)})$ and aggregate by letting them vote to form φ_B (Breiman, 1996). The $\{L^{(B)}\}$ sets are data sets drawn from the original training set, with replacement, with N cases in each. This means that each $(y_n, \boldsymbol{x}_n)$ may appear several times or not at all in any particular $L^{(B)}$. It is also satisfying that the $\{L^{(B)}\}$ sets follow the same underlying distribution as L no matter what that is (Efron & Tibshirani, 1994; Chapter 3). In other words the random forest is not associated with any assumptions concerning distribution, an attractive feature of the method when working with large data sets containing many features. An important notion of this method is the "out-of-bag" data, which are observations that do not make it into a particular $L^{(B)}$. This "out-of-bag" data forms a natural test set for the tree that is fitted to that bootstrap sample (Cutler & Stevens, 2006). In each bootstrap training set about one-third of the observations are not included, and there is empirical evidence that "out-of-bag" estimates is as accurate as using a test sample of the same size as the training set. With the random forest it is not necessary to set aside a test data sample (Breiman, 2001).

What differentiates the random forest from simply bagging several trees is the randomization at each split in every tree. At each node a small group of input features, $m < p$, where p is the total number of features, are selected at random and split where m is held constant throughout the whole random forest procedure (Cutler et. al, 2007). The most common value is $m \approx \sqrt{p}$, which is the standard value in the randomForest package in R (Breiman & Cutler, n.d). The rationale behind this process is that if all features are considered possible candidates at each split, the similarity of the trees increases as only a few features are used for splitting, due to differences in feature importance for the algorithm. When instead using a smaller amount of randomly selected features, the correlation between the trees is reduced and therefore, so is the variance of the estimates (James et al., 2013, Chapter 8). If simply choosing $m = p$, there is bagging of the decision trees.

Another attractive feature of the random forest is that estimates of feature importance are extracted from the algorithm (Breiman and Cutler, n.d). These estimations can be used to create a more parsimonious model.

This section is concluded by outlining the procedure more specifically:

1. Create bootstrap samples by randomly drawing observations from the original set with replacement.
2. Grow classification trees as outlined in section 4.5.2 for each bootstrap with randomly selected features tried at each split.
3. Classify observations by votes from each tree.

In this section, the foundations and basic workings of the random forest have been described. This is one of the two supervised learning methods that are being applied and compared in this thesis. The following section describes the fundamentals of the other supervised learning method, the support vector machine.

4.5 The support vector machine

To start with, in this section an overview and the general idea of the support vector machine is presented. Then, to illustrate the basic attributes of the support vector machine, a theoretical review is conducted using figures and equations with accompanying text (figures are adopted from Hastie et al., 2009, Chapter 12; James et al., 2013, Chapter 9; Bennett & Campbell, 2000 unless specified differently). The aim is to demonstrate the fundamental properties of the support vector machine, its applications and usefulness.

4.5.1 Background and overview

The support vector machine was developed in the 1990s and was originally designed to handle binary classification (Cortes & Vapnik 1995). It is a supervised statistical learning technique that creates input-output mapping functions from a labelled training data set. Since the technique was introduced it has been developed and extended, in addition to binary classification functions, to also handle multi-classification and regression functions. The support vector machines, besides being mathematically solid, are considered to perform very well when applied to real-world cases and are considered to be one of the best tools for machine learning and data mining (Wang, 2005, Chapter 1).

In essence, the support vector machine can be divided into three parts, depending on the data at hand. The optimal hyperplane that perfectly classifies the data into two groups is the most

basic form. This can only be applied to data sets where the classes can be linearly separated and is also known as the hard margin support vector machine or the maximal margin hyperplane. The soft margin hyperplane, also called the support vector classifier, is an extension that allows for some misclassification. The support vector classifier can in turn be extended to accommodate non-linear class boundaries, which is what is typically referred to as the support vector machine (Cortes & Vapnik, 1995; Hu & Kim, 2012; Hastie et al., 2009, Chapter 12).

4.5.2 Theoretical illustration of support vector binary classification

As previously mentioned, when classifying within supervised learning, the distinction is made between the known training data and the unknown test data. The training data consist of objects, where each object contains a class label and several features. Based on the features of the objects, they belong to a certain known class (Hsu et al., 2003). In order to classify the object into the correct class, a classifier mechanism needs to be constructed. In this case, using the training data, a support vector machine model is developed to perform the classification. When the support vector machine classifier has been trained (on the training data), the goal is to use it to predict which class the observations in the test data belong to.

4.5.3 The hard margin support vector machine or maximal margin classifier

The maximal margin classifier is the simplest support vector machine classifier in the sense that it is a linear classifier that is used when the data can be divided perfectly into two classes (James et al., 2013, Chapter 9).

In order to demonstrate, consider a set of training data consisting of *n* training observations in a *p*-dimensional space. Then the *np*-matrix X is as follows:

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ . \\ x_{1p} \end{pmatrix}, x_2 = \begin{pmatrix} x_{21} \\ \vdots \\ . \\ x_{2p} \end{pmatrix}, \dots\dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ . \\ x_{np} \end{pmatrix}$$

In the binary classification case, the training observations can be classified into either of two known different classes, commonly denoted as -1 and 1. Thereby the classes can be expressed as $(y_1, \dots, y_n) \in \{-1,1\}$. Then there exists a classifier in the form of a hyperplane that separates the observations according to class belonging (James et al., 2013, Chapter 9). A

separating hyperplane has the attribute that:

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) > 0, for\ all\ i = 1, \ldots, n$$

where $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients of the hyperplane and $y_i \in \{-1,1\}$

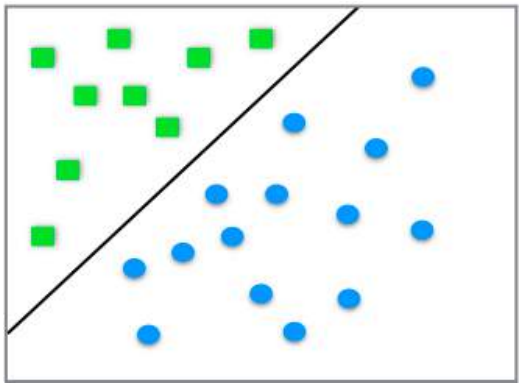

Figure 3. Classification of training objects.

As seen in figure 3, the training observations on one side of the hyperplane belong to one class while those on the other side belong to the other class (either class -1 or class 1, represented by the blue circles and green squares in the figure). The separating hyperplane in figure 3 acts as a linear decision boundary. As seen in figure 4, there exists not just one hyperplane that divides the data, between the observations from the two classes one can fit an infinite number of possible hyperplanes (James et al., 2013, Chapter 9).

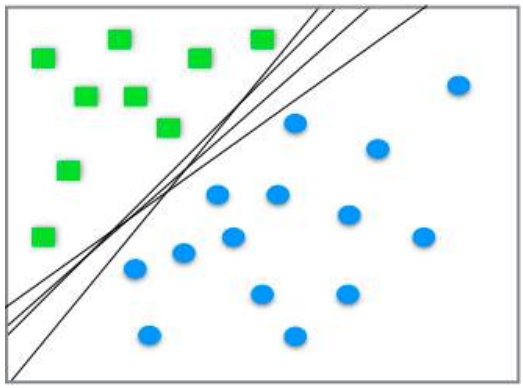

Figure 4. Multiple possible separating hyperplanes.

Out of all possible ones, a hyperplane that acts as a successful classifier is the maximal margin hyperplane, which is found by solving the following optimization problem:

$$\text{Maximize } M$$

$$\text{Such that}$$

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_{i,}(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M, for\ all\ i = 1, \ldots, n$$

where $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients of the maximal margin hyperplane. The constraints guarantee that every test object is located on the correct side of the hyperplane at distance M or further away, where M is the margin from the hyperplane (James et al., 2013, Chapter 9). As M is the margin from one side of the hyperplane, the total margin is 2M (Hastie et al., 2009, Chapter 12).

When the maximal margin hyperplane is found it can be used to classify the test data by predicting which class a test object belongs to. An observation of the test data is a p-vector of observed features $x^* = (x_1^* \ldots x_p^*)^T$ belonging to either class -1 or class 1. A test object x^*, is classified into class -1 if the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \ldots + \beta_p x_p^*$ is negative and into class 1 if the sign is positive (James et al., 2013, Chapter 9).

The maximal margin hyperplane is the hyperplane that has the farthest minimum distance to the observations, which is illustrated in figure 5. The observations that have the farthest minimum distance from this hyperplane are called support vectors. Their positions are highlighted by the two dotted lines seen in figure 5, which also signify the width of the margin. The lines perpendicular to the hyperplane emphasize the distance between each support vector and the hyperplane. If the support vectors are moved, the position of the hyperplane will change. This is contrary to what happens when moving any other observations, as there then is no effect on the position of the hyperplane. The support vectors are hence the points that decide the position of the hyperplane (James et al., 2013, Chapter 9).

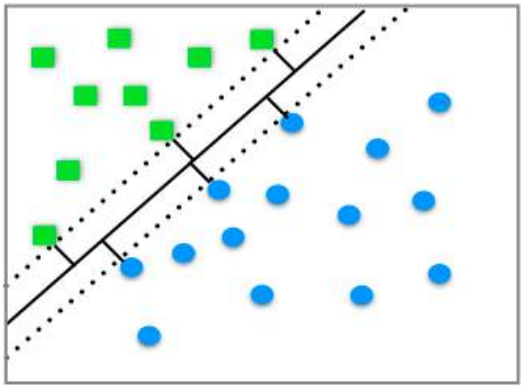

Figure 5. Maximal margin hyperplane.

The distance of the observations from the hyperplane can be seen as a measure of how much certainty there is in the classification. In the training data, if the support vectors are far from the hyperplane, then there is a large margin, and, ideally, the margin also will be large on the test data, leading to the test observations being classified correctly. The further the test object x^* is located from the hyperplane, the more certain the classification is. On the other hand, if the support vectors are close to the hyperplane, the margin will be smaller. This leads to less confidence concerning the correctness of the classification. Using the maximal margin classifier is generally a successful way to classify when it is possible to find a separating hyperplane, though, when p is large there might be problems with overfitting the data (James et al., 2013, Chapter 9).

Often, however, there does not exist such a hyperplane that exactly separates the two classes. Then there is no solution to the maximal margin hyperplane optimization problem with $M > 0$. In such cases, a hyperplane that nearly separates the classes can be used, which is referred to as using a soft margin or the support vector classifier (James et al., 2013, Chapter 9).

4.5.4 The soft margin support vector machine or the support vector classifier

The support vector classifier is an extension of the maximal margin classifier that can be used when it is not possible or desirable to separate the classes exactly. At times, a classifier based on a separating hyperplane might only have a tiny margin. In such a case, the confidence to correctly predict class is lower and the sensitivity to changes in individual observations is increased. A solution is to use a support vector classifier, which does not classify perfectly as it allows for training observations either to be on the wrong side of the margin or on the wrong side of the hyperplane. Hence, using a support vector classifier leads to increased robustness to individual observations and improved classification of the majority of the

training observations. Solving the following optimization problem gives the support vector classifier:

$$\text{Maximize } M$$

$$\text{Such that}$$

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_{i,}\left(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}\right) \geq M(1 - \varepsilon_i), for\ all\ i = 1, \ldots, n$$

$$\text{Where } \varepsilon_i \geq 0 \text{ and } \sum_{i=1}^{n} \varepsilon_i \leq C$$

and $\varepsilon_1, \ldots . \varepsilon_n$ are called the slack variables that permit individual observations to be located on the wrong side of the hyperplane or margin. When $\varepsilon_i = 0$, then the *i*th test object is on the correct side of the hyperplane. In the case of $\varepsilon_i > 0$, the object is on the wrong side of the margin and when $\varepsilon_i > 1$, the object is on the wrong side of the hyperplane. The parameter C is usually referred to as the tuning parameter. When C is equal to 0, then $\varepsilon_i = \ldots = \varepsilon_n = 0$ and the classifier is identical to the maximal margin hyperplane (James et al., 2013, Chapter 9).

If the value of C is greater than zero, the maximal number of observations on the wrong side of the hyperplane is equal to C. In other words, higher values of C mean that there is a wider margin with more violations of the margin. The data fit is less strict and the classifier is possibly more biased but with less variance. Lower values of C allow for less violations, a narrower margin and the classifier fits the data well. This potentially leads to low bias but high variance. The value of C is often chosen by cross-validations (James et al., 2013, Chapter 9), which is explained in section 4.6.

In figure 6, one can see what happens to the margin when one has two different values of C. To the right, C has a higher number meaning that there is a wider margin with more observations on the wrong side of the margin. To the left, where C has a lower value, the margin is narrower and fewer observations are on the wrong side.

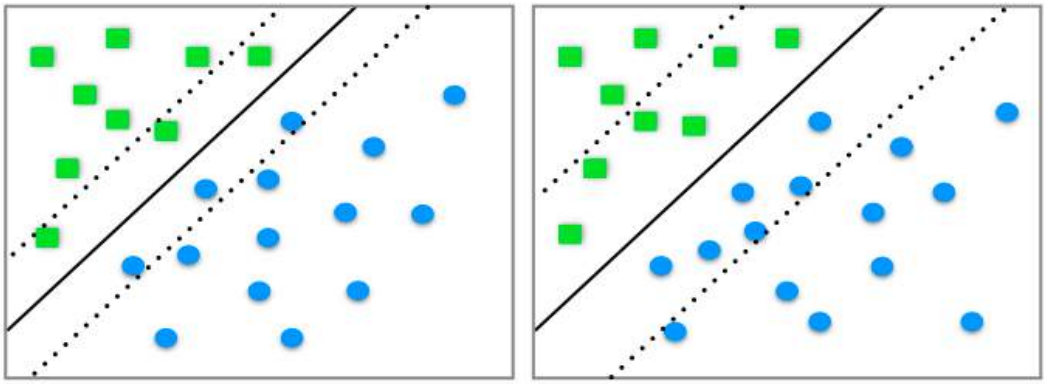

Figure 6. Different values of the parameter C.

As before, the only observations that affect the position of the support vector classifier are the support vectors; the observations that lie on the margin or on the wrong side of the margin for their class. As higher values of C lead to a larger margin, it, depending on the nature of the data, probably also leads to an increased number of support vectors (James et al., 2013, Chapter 9).

As the support vector classifier depends on the support vectors, which generally constitute only a small part of the training data, the classifier is fairly robust to the behaviour of observations far away from the hyperplane (James et al., 2013, Chapter 9).

4.5.5 The non-linear case, the support vector machine and the use of kernels

In cases where the data set cannot be separated by a linear classifier, a non-linear one is needed. A classical way to achieve this is by adding attributes to the data that are non-linear functions of the original data, which leads to the change from a linear to a non-linear classification algorithm. By doing this, current linear classification algorithms can be used in the expanded feature space while non-linear ones are produced in the original input space. This method of non-linear mapping is associated with two possible problems; overfitting due to the exponential dimensional increase of the feature space and practical calculation issues (Bennett & Campbell, 2000).

In the case when using the support vector machine, these problems are usually more or less overcome. As long as there is a suitable value of C, the overfitting problem is generally not an issue as the support vector machine uses margin maximization (for more details regarding overfitting and underfitting, see section 4.6). Furthermore, by using kernel functions, the computational complexity is reduced (Bennett & Campbell, 2000).

Before moving forward, a short description of kernel methods and kernel functions is appropriate. Kernel methods, of which the support vector machine is one, in essence, consist of two parts; one that accomplishes the mapping of the input data into the vector space called the feature space and one that is the learning algorithm aimed at uncovering linear patterns in this feature space (Shawe-Taylor & Christianini, 2004, Chapter 2).

By using kernel functions the input data are non-linearly mapped into a higher dimensional feature space where it is possible to define a similarity measure based on the inner products. In the feature space a linear classifier is used while it is non-linear in the original input space. This classifier is only expressed by the inner products of the data. The kernel function facilitates the possibility to operate in the input space leading to the inner products of the feature space not needing to be assessed, which makes calculations much easier (Jakkula, 2006).

Moving back to the case of the support vector machine, it happens to be the case that the solution to the support vector classifier optimization problem solely involves the inner products of the objects. The inner product for two objects x_i and x_{i*} is:

$$\langle x_i, x_{i*} \rangle = \sum_{j=1}^{p} x_{ij} x_{i*j}$$

Hence, the support vector classifier can be represented as:

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

where there is one parameter α_i per training object. The inner product between the new object x and each of the training objects x_i has to be calculated. Regarding α_i, if the training object is not a support vector, α_i is zero. If S is the support vectors, the solution function is of the form:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

So, only the inner products are needed in representing the linear classifier $f(x)$ and calculating its coefficients (James et al., 2013, Chapter 9).

A generalization of the inner product of two objects can be written as $K(x_i, x_{i^*})$ where K is the kernel function that quantifies their similarities. A linear kernel is the same as the support vector classifier, which is the case when $(x_i, x_{i*}) = \sum_{j=1}^{p} x_{ij} x_{i*j}$. The linear kernel quantifies the similarities of a pair of observations using Pearson correlation (James et al., 2013, Chapter 9).

In order to classify data that are not linearly separable, the support vector machines use kernel functions instead of adding attributes to the data that are non-linear functions of the original data, which was the classical way of doing it. The main idea is still as has been outlined earlier; the input vectors are transformed into high-dimensional feature vectors where the training data are linearly separable. A separating hyperplane is constructed which, in the transformed feature space, becomes a linear function while being a non-linear function in the input space (Hu & Kim, 2012). The transformation of the input vectors is illustrated in figure 7, where the non-linearly separable input space is seen to the left, the middle shows the transformation into a higher dimensional feature space where linear separation is possible, while to the right, the non-linear separation in the input space is illustrated (Brereton & Lloyd, 2010).

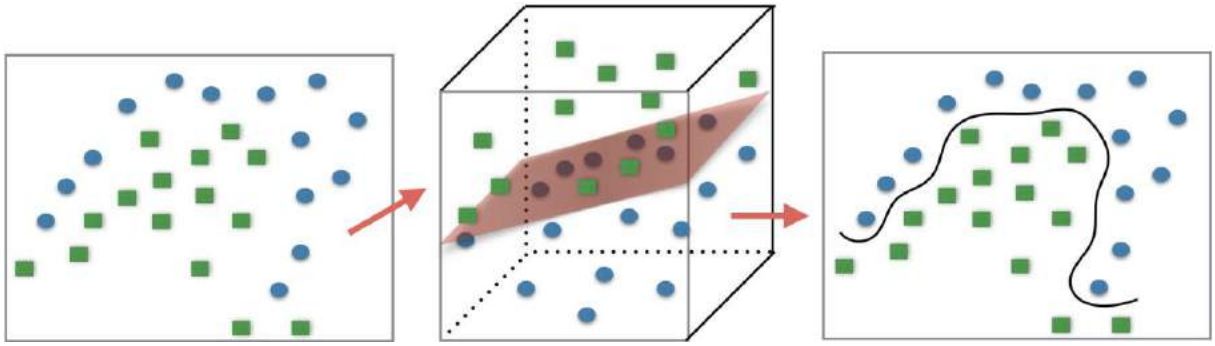

Figure 7. Transformation of the input space (adopted from Brereton & Lloyd, 2010).

Depending on the nature of the data, the kernel to choose is the one that best captures the decision boundary. Examples of two commonly used kernels are the polynomial kernel and the radial kernel, described below. The radial kernel is also commonly referred to as the radial basis function (RBF) kernel or the Gaussian RBF kernel (Ben-Hur & Weston, 2010; Hastie et al., 2009, Chapter 12; James et al., 2013, Chapter 9). For simplicity, henceforth, solely the term radial kernel is used in this thesis.

The polynomial kernel of degree d can be seen below, where d is the kernel parameter, a positive integer. If d is equal to 1 it is equivalent of the linear kernel.

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

The radial kernel has the following form:

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

where γ is the kernel parameter, a positive constant (James et al., 2013, Chapter 9).

The kernel parameters, together with the tuning or soft margin parameter C, are usually referred to as the hyperparameters. As previously brought up, the parameter C affects the width of the margin. The kernel parameters instead affect the flexibility of the classifier (or decision boundary). In the case of the polynomial kernel, as mentioned, a degree of 1 gives the linear kernel while increasing the value of d leads to more flexibility and bend to the classifier. Regarding the radial kernel, low values of γ give a classifier that is almost linear while increasing γ leads to more curvature of the classifier (Ben-Hur & Weston, 2010). Figure 8 gives an idea of how the two non-linear kernels work. To the left is an example of a polynomial kernel separating the data and to the right, an example of what a radial kernel might look like. In these two examples the values of d and γ respectively are quite high as the classifiers do not resemble a linear classifier and have quite an amount of curvature.

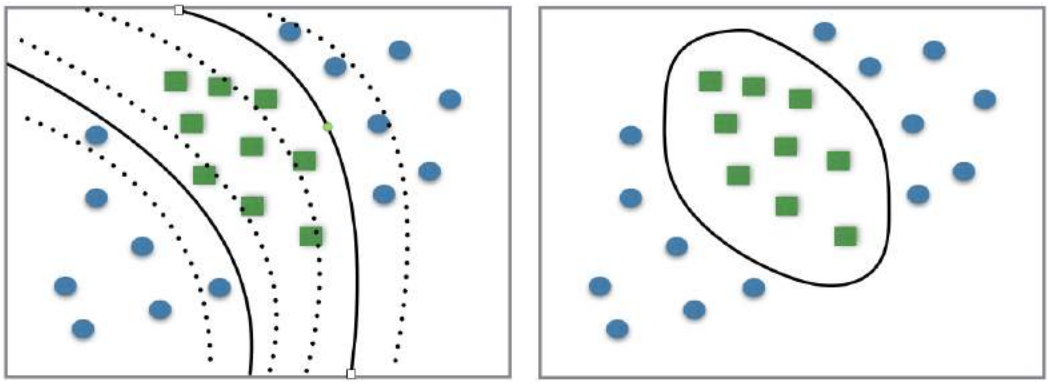

Figure 8. Polynomial and radial kernels

To summarize this section, the choice of classifier to apply depends on the nature of the data. Theoretically, following the just presented outline of the workings of the support vector machine, if the data are linearly or nearly linearly separable, the choice should fall upon the

maximal margin classifier or the support vector classifier. In the case of non-linear data, the support vector machines with its non-linear kernel functions are suitable. In practice, however, kernel functions are used regardless of the nature of the data. In cases where the data are linearly or nearly linearly separable, a linear kernel function is used and tuned appropriately. When the data are non-linear, a suitable non-linear kernel is applied.

Regarding the choice of the kernel function to use, whether it is the linear, polynomial, radial kernel or any other kernel, it depends on the data and is a process of trial and error in order to find the one that is the most suitable (Ben-Hur & Weston, 2010). In this thesis, the kernel functions that are applied and evaluated against each other are the linear, polynomial and radial kernels. The reason for using these particular ones is that they are the most commonly used ones and, with the right tuning, are capable of classifying most data sets (see for example James et al., 2013, Chapter 9; Ben-Hur & Weston, 2010).

4.6 Prediction accuracy and cross-validation

The usual way to evaluate a classifier is by its prediction (classification) accuracy, which is the percentage of correct classifications out of the total number of classifications (Kotsiantis, Zaharakis & Pintelas, 2007).

In the case of the random forest, the prediction accuracy is optimized by tuning the number of randomly selected features tried at each split (Breiman, 2001). Regarding the support vector machine, the tuning of the classifier is achieved by adjusting its parameters. Depending on which kernel is used, there are different hyperparameters that have to be tuned. As mentioned, three different kernels are investigated in this thesis, namely the linear, the polynomial and the radial kernel. Starting with the linear kernel, the only parameter to tune is C. Regarding the polynomial kernel, the parameters to tune are C and d while in the case of the radial kernel the parameters are γ and C. The aim is to choose the parameter values that lead to the classifier predicting the test data with as high accuracy as possible (one wants high accuracy regarding the test data, which is not always desirable for the training data) (Hsu et al., 2003). Also important to consider, when deciding on an appropriate value of the parameter C, is that it may have an effect on whether the model underfits or overfits the data (James et al., 2013, Chapter 9). Different values of C affect the margin's width and give a trade-off between maximizing the margin and minimizing the errors. Choosing values that are too high lead to overfitting while too low values lead to underfitting, which potentially leads to an

oversimplified model being used (Alpaydin, 2010, Chapter 13). When there is overfitting, the generalizability of the support vector machine disappears with the consequence being misleading results. Although results might be decent on a particular training data set, the performance of the classifier cannot be generalized to the test data. In addition to tuning C inappropriately, overfitting might happen if an unsuitable kernel function is chosen (Han & Jiang, 2014).

In order to illustrate the problems caused by overfitting consider the two simple examples seen in figure 9 and figure 10. To the left of figure 9, a classifier that is overfitted on the training data is seen, while to the right, the unsuccessful classification of the test data when using the overfitted classifier is shown. Looking at figure 10, to the left, a more suitable classifier is fitted on the training data. To the right, when using the more appropriate classifier, the classification of the test data is more successful (Hsu et al., 2003).

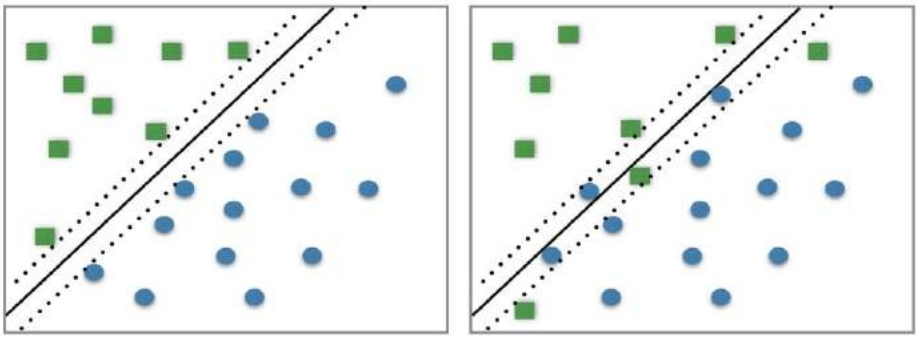

Figure 9. Overfitting the data (adopted from Hsu et al., 2003)

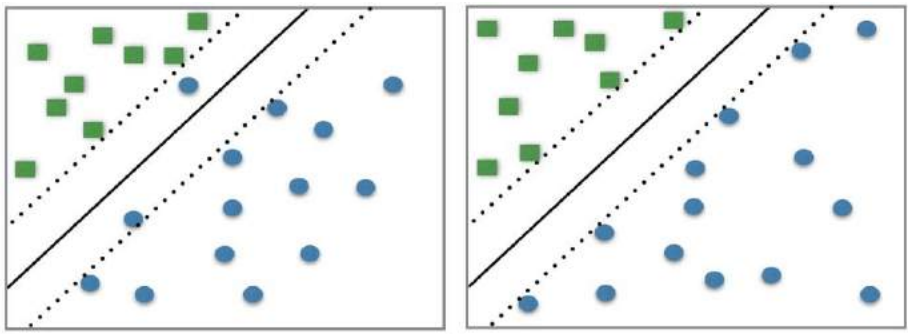

Figure 10. Using a more suitable classifier (adopted from Hsu et al., 2003)

The three most widely used methods for tuning and calculating the prediction accuracy of a classifier are the two-one method, cross-validation and leave-one-out cross-validation. In the first method, the training data are divided so that two-thirds is used for training and one-third is used for performance estimation (Kotsiantis et al., 2007).

In the case of cross-validation (also called v-fold or k-fold cross-validation), the training set is divided into v equally sized subsets. When equal division is not possible, some of the folds get an extra observation. Then, a classifier trained on v-1 subsets (or folds), is used to test the remaining subset sequentially. By doing this, each object of the entire training set is predicted once and the percentage of correctly classified data is the cross-validation accuracy. Regarding the support vector machine, the parameter values that give the highest cross-validation accuracy based on receiver operating characteristic (ROC) values (further explained in section 4.9) are the ones to choose. As mentioned earlier, the choosing of the parameter value C using the process of cross-validation can alleviate problems of overfitting (Hsu et al, 2003). In the case of the random forest it is not strictly necessary to use cross-validation as the algorithm performs a parallel cross-validation with the out-of-bag samples. As previously mentioned, about one third of the observations are not included in a given bootstrap sample. The algorithm then automatically lets the Bth classification tree predict the left-out observations for the bootstrap sample the tree was generated from. The out-of-bag assessment of performance and the cross-validation procedure has been proven to give quite similar, unbiased, results (Svetnik et al., 2003; Breiman & Cutler, n.d.). However, for the sake of comparison, the same method of cross-validation as for the support vector machine is used.

An extreme case of v-fold cross-validation is the leave-one-out cross-validation where in each fold, training is done on n-1 observations and validation is performed on one observation in each fold. The test classification error is then averaged over all folds. The result is an unbiased estimator with high variance due to the data sets in all folds being very similar apart from one observation. When taking the bias and variance trade off into account a recommendation is using five or ten folds (James et al., 2013, Chapter 5). In regards to this recommendation, the chosen method, in this thesis, for optimizing classification accuracy is to use the five-fold cross-validation.

There are two main drawbacks associated with the cross-validation procedure when comparing different models. One is that the classification accuracy is dependent on the random dividing of the data into folds. To alleviate any problems with this and enable reliable comparisons between the random forest and the support vector machine, the cross-validation process has to be repeated several times and the results thereafter averaged (Salkind, 2010,

Cross-Validation). Therefore, the five-fold cross-validation process in this thesis is repeated twenty times.

Another issue is that the cross-validation process is data-driven, meaning that a model that works well when tested on one set might not always work as well on another. Therefore, ideally, when comparing models, before deciding which one is the best, the models should be applied on several test sets (Salkind, 2010, Cross-Validation). Otherwise, questions regarding the generalizability of the results can be raised. Unfortunately, in this thesis, only one test set is available and it is therefore not possible to further investigate this by applying the models to additional sets. However, both the random forest and the support vector machine are known to generalize well even when trained on small training sets (see for example Cortes & Vapnik, 1995; Rio & Zha, 2004; Ham, Chen, Cramford & Ghosh, 2005; Mountrakis, Im & Ogole, 2011).

4.7 Statistical software

The k-means clustering method is performed using the statistical software SAS. The main reason to use SAS, in this case, is that the clustering method is relatively simple and straightforward to conduct in this software.

In the cases of the random forest and the support vector machine, the software used is R. The random forest technique was developed for use in R by Breiman (Breiman & Cutler, n.d) and includes all the relevant packages concerning the random forest. In SAS, on the other hand, the random forest procedure is not as developed. Regarding the support vector machine, the necessary tools to apply the method are available in both SAS and R. However, as a large amount of research papers and books (see for example James et al., 2013; Statnikov et al., 2008; Karatzoglou et al., 2006) applying and dealing with both the random forest and the support vector machine use R, the choice of using R for both methods seems reasonable. The specific R packages used in the case of the random forest and the support vector machine are the Caret, kernlab, pROC and randomForest packages.

4.8 The parameter C, the cost parameter and the radial kernel parameter

In the just mentioned examples illustrating the theoretical foundations of the support vector machine, the tuning parameter C that adjusts the margin of the classifier has been mentioned several times. In the presented mathematical formulations, this parameter is a constant (Hastie

et al., 2009) that is regarded as a budget parameter, where higher values allow for more violations of the margin and lower values allow for less. However, in practice, when computing the support vector machines with linear and non-linear kernels in the statistical software R, the parameter C is replaced by a cost parameter that penalizes or specifies the cost of violating the margin. This cost parameter is a constant that is tuned in R in order to achieve the best possible classification performance. Choosing higher values of the cost parameter create a narrower margin, leading to fewer support vectors located on the margin (or violating the margin) while lower values lead to the opposite (James et al., 2013, Chapter 9). In order to clarify, higher values of the cost parameter lead to a smaller margin while lower values lead to a larger margin (Ben-Hur & Weston, 2010). Comparing with the budget parameter C, both affect the classifier's width of the margin albeit in opposite directions as in the case of the budget parameter, higher values lead to a larger margin and lower lead to a smaller margin. Important to note, is that the cost parameter is also frequently referred to as the parameter C or the soft margin constant, which can be misleading (see for example Hsu et al., 2003; Hastie et al., 2009, Chapter 12; Ben-Hur & Weston, 2010).

The scale of the cost parameter has no meaningful interpretation, which is why it is more intuitive to use alternative formulations such as the parameter v, which controls the fraction of the support vectors and of the margin errors (Ben-Hur, Ong, Sonnenburg, Schölkopf & Rätsch, 2008), or the above mentioned budget parameter C. This being said, the software R uses the cost parameter, and it is this parameter that is actually tuned in order to optimize the classifier. Therefore, in the subsequent sections of this thesis, when mentioning a parameter that affects the margin's width or the cost in connection with the support vector machine, henceforth, the parameter in question is the cost parameter.

In the case of the radial kernel, as mentioned, the kernel parameter that decides the curvature of the classifier is denoted γ in the previous theoretical section. Using γ is a common way to represent this kernel parameter, however, an alternative way is to use the symbol σ. These two symbols are not identically interchangeable, although in essence they have the same effect on the radial kernel. The symbol σ, comparing with γ, affects the radial kernel's width and curvature in the opposite direction. Hence larger values of σ lead to less curvature and more resemblance to a linear kernel while smaller values lead to more flexibility and greater curvature. The Caret package in R uses σ, denoted in practice as sigma (Karatzoglou, Smola, Hornik & Karatzoglou, 2015; Ben-Hur et al., 2008). Therefore, in the following sections,

when mentioning the radial kernel parameter, sigma is used. In the case of the polynomial kernel, the Caret package also includes a scaling parameter. This parameter normalizes patterns in the data without actually altering the data itself (Karatzoglou et al., 2015).

4.9 Choice of algorithm setup and evaluation techniques

The data set used in this thesis, discussed in section 3.1, consists of 33 observations. As mentioned, the data are divided into what is referred to as the training data and the test data. Commonly, the distinction between the training data and the test data is achieved by randomly separating the available data into two groups where about half to one-third of the data, depending on the size of the data set, are considered as training data while the rest is treated as test data (Salkind, 2010, Cross-Validation). In this case, the data set is divided in such as way that the test data contain 11 observations while the training data contain 22.

The unsupervised learning method applied, the k-means clustering technique, divides the data and forms two classes based on existent patterns in the data, where one class is denoted as class 1 and the other as class 2. In order to determine whether or not the k-means clustering is successful, the cubic clustering criterion is used. Values above 2 or 3 indicate that the clustering is capturing some of the multivariate structure in the data (Cleland, Rothschild & Haslam, 2000).

In the case of the support vector machine, as mentioned, the three kernels that are evaluated are the linear, the radial and the polynomial kernel. The standard procedure to follow regarding which kernel to use is to start with the linear kernel and evaluate how it performs. The next step is to use non-linear kernels, in this case the radial and the polynomial, and assess how they perform in comparison with each other and the linear kernel. The one that usually performs better of these two non-linear kernels is the radial one. Within bioinformatics, using a linear kernel often gives the best results (Statnikov et al., 2008; Ben-Hur & Weston, 2010). In such cases, where there are often relatively few observations and many features, as in for example microarray data sets, using the polynomial or radial kernels might lead to overfitting in higher dimensions (Ben-Hur & Weston, 2010). In the results, cross-validated and optimally tuned support vector machines with linear, radial and polynomial kernels are compared and evaluated. As overfitting is of concern, the same comparison of optimally tuned kernels is done on a noised-up version of the original data set where a random number from a normal distribution, with zero mean and 0.5 standard

deviation, is randomly added to 30 per cent of the measured protein levels. The choice of selecting a zero mean, a standard deviation of 0.5 and adding noise to 30 per cent of the measured protein levels is more or less arbitrary. The reasoning behind choosing these values is that the aim is to only slightly modify the data in order to examine whether there are any signs of overfitting or not. Feature importance measures are not provided automatically in R for the support vector machine, however, it is possible to extract the most important features based on the receiver operating characteristic (ROC) value (briefly explained below) of each feature.

Regarding the optimization of the random forest algorithm the process is quite straightforward. The number of trees to use and the number of features tried at each split are the two things to consider. There is a diminishing effect in terms of accuracy for each added tree, with the algorithm stabilizing at around 200 trees (Hastie et al., 2009, Chapter 15). In this case, 2500 trees are used, well above the stabilizing point, as the power of a modern computer means that the computational time it takes to add that many trees is negligible. The only real parameter to tune is the number of features tried at each split. The algorithm sequentially tests 1 to 50 randomly chosen features tried at each split and then chooses the optimal number in terms of the ROC value. If there are ties i.e different number of features tried give the same highest ROC value, the lowest optimal number is used. Regarding feature importance measures, this is provided automatically in the setup of the randomForest package in R.

In order to make an accurate assertion concerning the relative performance of the support vector machine and the random forest, a measurement that measures performance in a way that allows for comparison is needed. The receiver operating characteristic (ROC) is a suitable measurement as it works well in cases of binary classification. The measurement is commonly used in medical research where classification is a diagnosis (positive or negative), and with an uneven number of observations in the classes. It is in fact a measurement frequently used when determining the validity of biomarkers for evaluating a binary outcome (Kumar & Indrayan, 2011). The total area under the ROC curve is a measurement of the classification performance of the algorithm, referred to as the area under the curve (AUC). The larger the area below the curve is, a value between 0.5 and 1, the more accurate the classification of the observations in the test set is. The ROC curve is a plot with sensitivity on the y-axis and 1-specificity on the x-axis. Using the class notation of the k-means clustering in order to

illustrate, sensitivity is the proportion of observations with class belonging equal to 1 that are correctly classified and specificity is the proportion of observations with class belonging equal to 2 that are correctly classified. A perfect test means that both are equal to 1. Anything above the diagonal line indicates that the algorithm performs better than what is achieved by random chance. The diagonal line, specificity + sensitivity = 1, represents the situation where an algorithm is useless and the classification outcome is completely random and equally likely. The classification capabilities of an algorithm are quantified by the AUC, where a perfect score on the test set is equal to 1 and the worst possible is 0.5 (Bewick, Cheek & Ball, 2004). Generally, AUC values above 0.9 are considered excellent while between 0.8-0.9 is considered good. A value between 0.7-0.8 is regarded as fair and between 0.6-0-7 is poor (Tape, 2006). The AUC values are accompanied by corresponding confidence intervals, which further enables evaluation of the classification accuracy.

ROC values are also used in the tuning process of the random forest and the support vector machine. In the case of the random forest, the ROC is used in order to decide which parameter value to choose while in the case of the support vector machine, the ROC is used to decide which kernel to use.

Finally, certainty regarding the reliability of the results needs to be established in order to make sure that the results are not just a function of the randomness associated with assigning observations to folds. Therefore, on the training data, a twenty times repeated five-fold cross-validation is performed and results are averaged across the repeats.

5. Results

In this section, to start with, the results of the k-means clustering are briefly reported. Then the results of the classification of the data set in regards to the random forest and the support vector machine are presented. The two techniques are compared and the method that is shown to be the superior classifier is thereafter used to construct a final parsimonious model based on feature importance measures. Results from using the five and ten most important features are reported. The performance of this parsimonious model is then compared to the original classifications performed by the best performing method.

5.1 The k-means clustering

The results of the cluster analysis plotted against two principal components are shown in figure 11. Examining the figure, it is evident that the k-means clustering is successful in dividing the observations into two clusters, where an observation's cluster belonging is denoted by either the number 1 or 2. The two classes that the observations can belong to are hence class 1 and class 2. Further inspection of the figure reveals that it is possible to fit a straight line between the two clusters and thereby perfectly separate them.

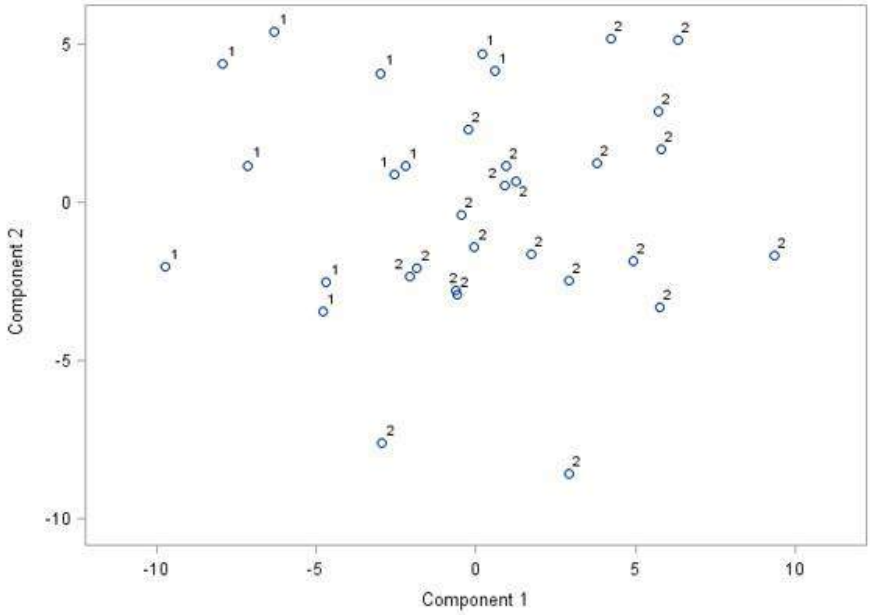

Figure 11. The results of the cluster analysis with group belonging illustrated by the observation markers and principle components on the y and x-axis.

The cubic clustering criterion from the k-means cluster analysis is equal to 10.32, which is safely above the threshold of around 2 to 3, meaning that the goal of deriving a binary dependent variable from the underlying structure is fulfilled.

5.2 The support vector machine

The three support vector machine kernels are optimally tuned on the training data and the one that performs the best is thereafter applied to the test data. Starting with the radial kernel, the receiver operating characteristic (ROC) is displayed as a function of the two tuning parameters sigma and cost, as seen in the two diagrams in figure 12. After tuning, the support vector machine with the radial kernel has an optimal sigma value = 0.006 and cost = 36. The tuning gives an area under the curve (AUC) value of 0.983.

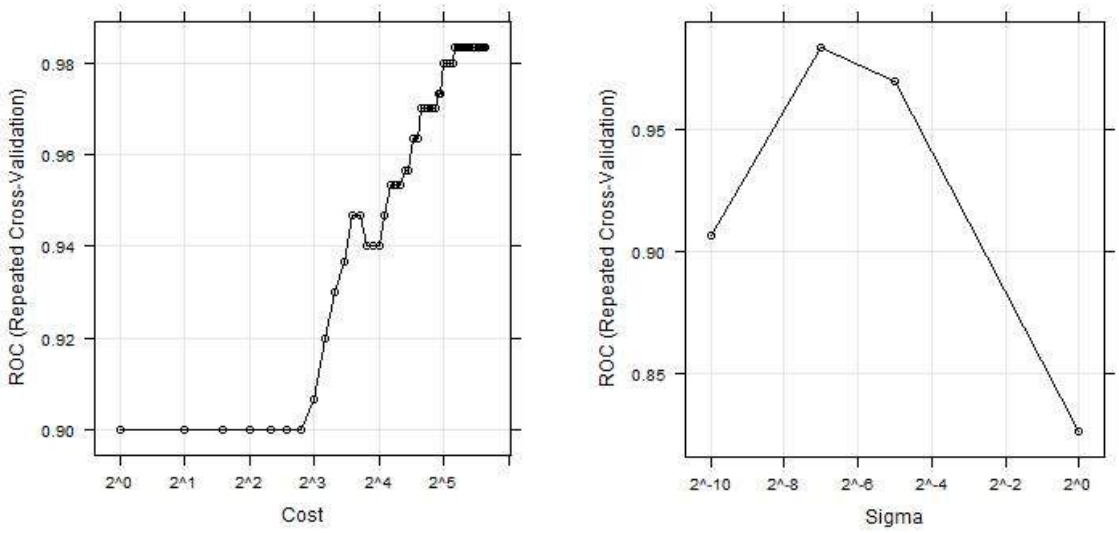

Figure 12. ROC displayed as a function of the tuning parameters sigma and cost for the radial kernel.

In figure 13, the ROC is plotted as a function of the cost parameter for the support vector machine with a linear kernel. The cost is the only parameter subject to tuning when using a linear kernel. The optimal tuning gives a cost = 0.5, with an AUC value of 0.983.

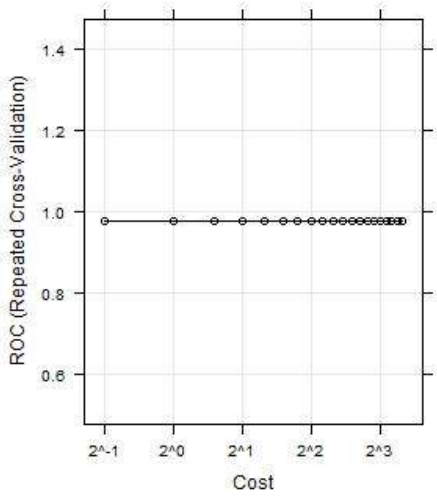

Figure 13. The ROC displayed as a function of the tuning parameter cost for the linear kernel.

Lastly, in figure 14, the ROC is shown as a function of the tuning parameters of the polynomial kernel: cost, degree and scale. The optimal tuning gives a degree = 1, scale = 1, cost = 0.5 with a ROC value of 0.983. That the optimal polynomial kernel is of degree 1 and has cost = 0.5 means that it is equivalent to the linear kernel. In table 1, a summary of all kernel parameter values as well as AUC values is displayed.

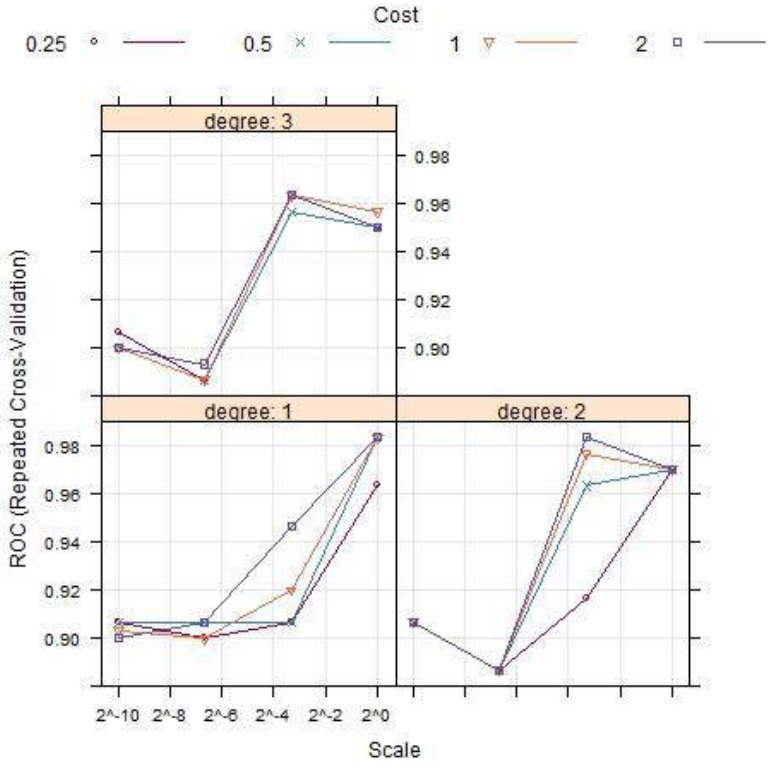

Figure 14. ROC displayed as a function of tuning parameters cost, scale and degree for the polynomial kernel

Linear kernel		**Radial kernel**			**Polynomial kernel**			
Cost	AUC	Cost	Sigma	AUC	Cost	Degree	Scale	AUC
0.5	0.983	36	0.006	0.983	0.5	1	1	0.983

Table 1. Optimal parameter values and the AUC value for each kernel

In figure 15, the confidence intervals (95 %) of the ROC for the three support vector machine models are shown. Examining the confidence intervals shows that the models perform equally well on the training data set and it is therefore unclear which of them outperforms the others. However, as previously reported, the radial kernel has the highest cost parameter of the three, meaning that misclassifications are to a greater degree penalized with the likely consequence being overfitting.

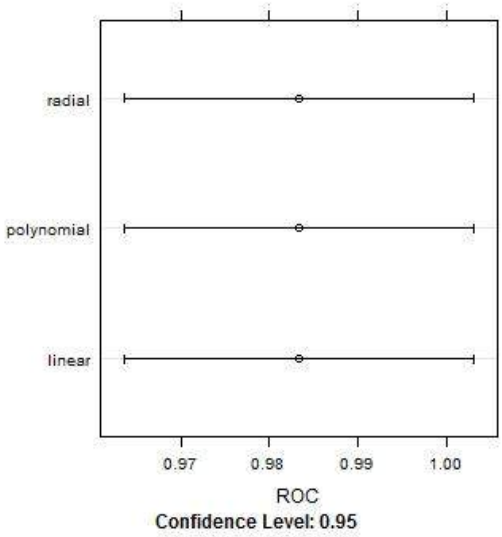

Figure 15. Confidence intervals (95 %) of the ROC values for the three support vector machine models.

In order to further investigate whether there is overfitting or not, the three different support vector machines are applied to, and evaluated on, a noisier version of the training data set. In figure 16, the confidence intervals for the optimally tuned support vector machines applied on the noisy data set are displayed.

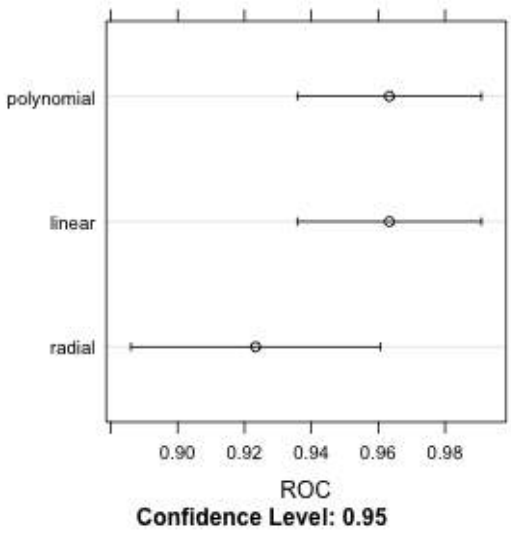

Figure 16. Confidence intervals (95 %) of ROC values for the three support vector machine models on the noisy training data.

It is evident by figure 16 that the optimally tuned radial kernel performs worse than the polynomial and linear kernels. This is an indication that the radial kernel is indeed overfitted on the original training data. In light of these results, the radial kernel is discarded. As the

optimal polynomial kernel is linear and identical to the linear kernel performancewise, the linear kernel support vector machine is henceforth used and compared to the random forest on the test data.

5.3 The random forest

In the case of the random forest, the only tuning parameter to consider is the number of features tried at each split when creating the decision trees. In figure 17, the ROC is plotted as a function of the number of features tried at each split.

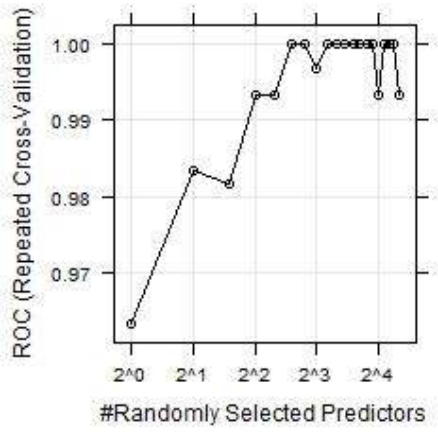

Figure 17. The ROC plotted as a function of the number of randomly selected features for the random forest.

The optimal number of features is chosen based on the ROC value. If there are ties, the lowest optimal number of features is chosen in order to reduce the correlation between the trees. As seen in figure 17, the optimal number of features tried at each split is 6. The number of trees used when training the algorithm is 2500 as further increasing the number of trees beyond this point does not have any impact on the results apart from computing time.

5.4 Comparisons and model choice

To start with, the performance on the training data set of the optimally tuned random forest and support vector machine with a linear kernel is compared. The results show that the random forest outperforms the support vector machine on all evaluation measures. The mean, as well as the minimum and maximum values from the training process of the ROC, sensitivity and specificity for all models are found in table 2. These results are the average across all folds of the cross-validation procedure from the training process. Studying the means of the different measurement, the random forest has a slight edge over the support vector machine in terms of the ROC value, as well as sensitivity and specificity.

	RF			*SVM*		
	Min	**Mean**	**Max**	**Min**	**Mean**	**Max**
ROC	0.833	0.997	1	0.667	0.99	1
Sens.	0	0.85	1	0	0.78	1
Spec.	1	1	1	0.333	0.96	1

Table 2. Minimum, mean and maximum accuracy values for the random forest (RF) and the support vector machine (SVM).

The confidence intervals for the evaluation measurements are found in figure 18. That the random forest performs better is supported by its narrower confidence intervals for the ROC value, the specificity and the sensitivity. The random forest has a specificity value of 1 across all folds in the cross-validation, which is why no confidence interval is reported. The confidence interval for sensitivity of the random forest is relatively narrower with a higher lower and upper bound.

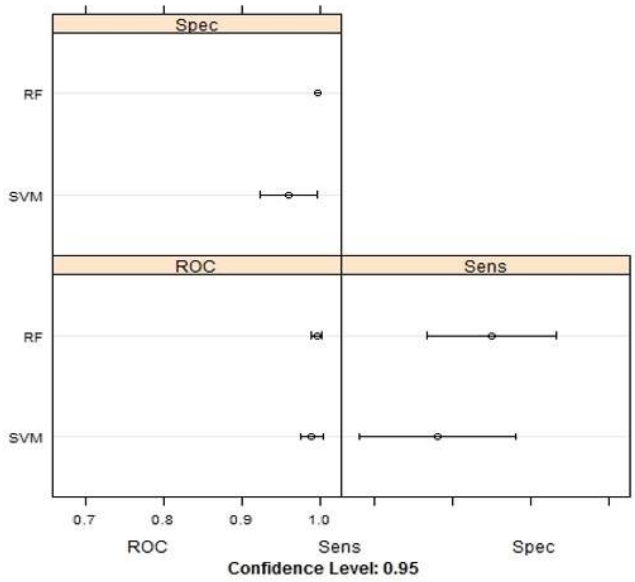

Figure 18. Confidence intervals of the ROC, specificity and sensitivity for the random forest and the support vector machine based on the training of the algorithms.

The next step is to compare how the random forest and the support vector machine actually perform on the test data. The main measurement used is the AUC value. In figure 19, based on their performance on the test set, the ROC curves for both algorithms are shown. In the

case of the random forest, the AUC value is 1 whilst it is 0.9643 for the support vector machine. The interpretation of this is that, overall, the random forest does a better job when classifying the observations in the test set as a value of 1 means that all observations are correctly classified.

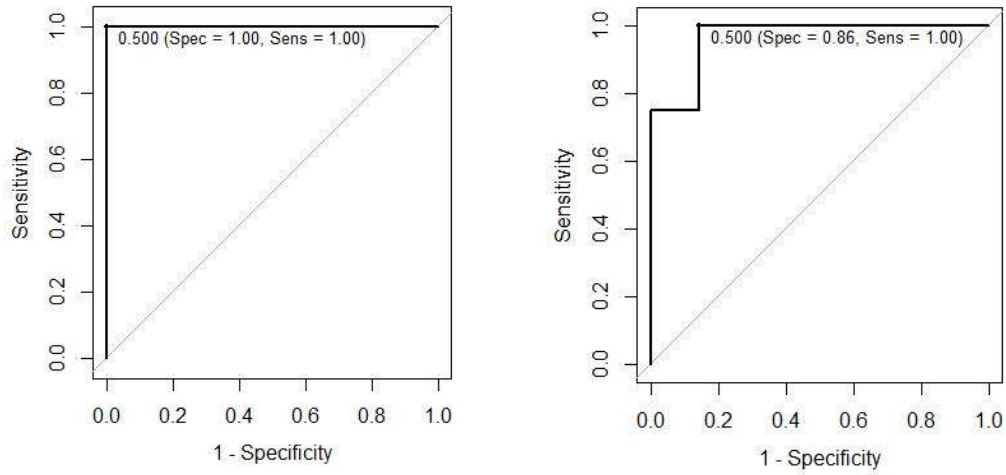

Figure 19. ROC curves for the random forest to the left with AUC=1 and the support vector machine to the right with AUC = 0.9643.

In table 3, the classification accuracy (percentage of correct classifications) and corresponding confidence interval based on the results on the test set, are found. Note that the p-value displayed, reported in parenthesis, is a test whether the accuracy is significantly higher than the no information rate (NIR), which is 0.6364 in this sample. The NIR is the largest class, which in this case is class 2, as the percentage of the total data. As seen in table 3, the random forest correctly classifies all 11 observations in the test set while the support vector machine correctly classifies 10 observations. Apart from this, a noticeable difference between the models becomes evident when considering their lower bounds for the confidence intervals of accuracy. The random forest model has a lower bound of 0.751 whilst the support vector machine has a lower bound of 0.587.

	RF	***SVM***
Accuracy	1 (0.007)	0.909 (0.051)
95 % CI	0.7151, 1	0.587, 0.998

Table 3. Accuracy, as the proportion of correctly classified observations in the test set, and the corresponding confidence intervals. P-value in parenthesis of hypothesis Accuracy > NIR.

As the random forest model does a better job in classifying the observations in the test data set the conclusion is that the random forest is, in this case, a better model. Hence, the parsimonious model will be based on the random forest.

5.5 Feature selection and performance

In this section, the performance of the random forest using the top ten as well as the top five features is evaluated. Furthermore, a comparison is made with the random forest using all features. In figure 20, a plot is displayed of the top ten features in falling order in terms of explanatory power, from the random forest algorithm, when classifying observations. The higher importance value of a feature, the more important the feature is in the context of classifying an observation correctly.

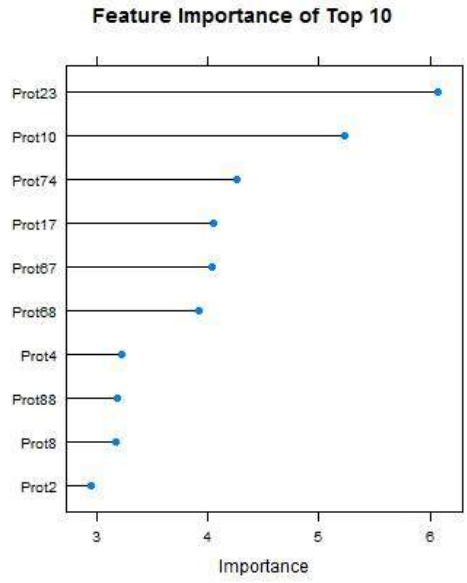

Figure 20. Gini index for measurement of feature importance in the random forest algorithm.

First, the performance on the training set of the random forest using different number of features is evaluated. In figure 21 the confidence intervals of the evaluation measurements are reported. When using both the top ten features and the top five, the random forest still performs very well. The main difference is the increased uncertainty of the ROC when using only five features. The lower bound is still above 0.95 though. Also, the sensitivity when using five and ten features is slightly higher compared to when using all features. In table 4, the minimum, the mean and the maximum of the evaluation measurements from the training data are reported.

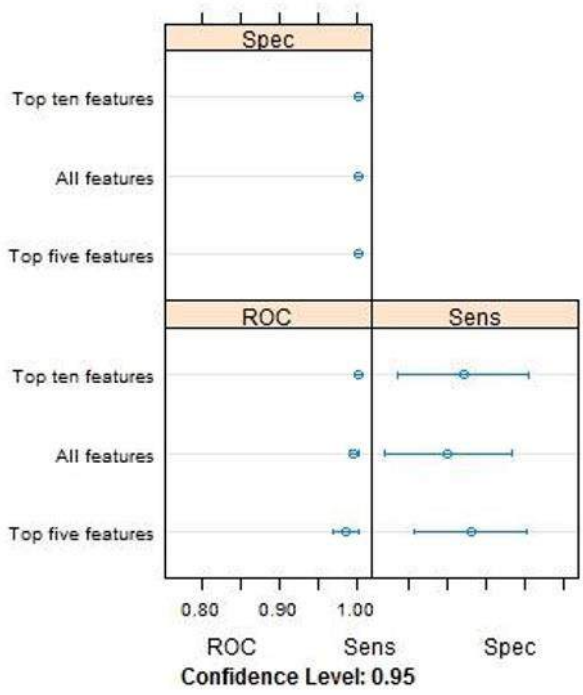

Figure 21. Confidence intervals for the random forest when using five, ten and all features respectively.

The random forest with the top ten features performs the best in terms of the ROC and the specificity and is tied best when it comes to sensitivity. Using only the top five features performs comparably well also, although it has the lowest minimum value of the ROC.

	RF all			***RF top 10***			***RF top 5***		
	Min	**Mean**	**Max**	**Min**	**Mean**	**Max**	**Min**	**Mean**	**Max**
ROC	0.833	0.997	1	1	1	1	0.667	0.9867	1
Sens.	0	0.85	1	0	0.88	1	0	0.88	1
Spec.	1	1	1	1	1	1	1	1	1

Table 4. Minimum, mean and maximum accuracy values for the random forest with different number of features.

Lastly, the classification accuracy on the test data is reported. As displayed in table 5, when using the top five features the random forest still classifies all observations in the test set correctly. This indicates that a random forest with only five of the original 89 features does, in this sample, as good a job in terms of classification as the random forest with all features.

	RF all	*RF top 10*	*RF top 5*
Accuracy	1 (0.007)	1 (0.007)	1 (0.007)
95 % CI	0.7151, 1	0.7151, 1	0.7151, 1

Table 5. Accuracy, as the proportion of correctly classified observations in the test set, and corresponding confidence intervals. P-value in parenthesis of hypothesis Accuracy > NIR.

Based on the test results from table 5, the random forest with feature selection works just as well on the test set as the random forest without feature selection.

6. Discussion and analysis

The k-means clustering method clearly shows that it manages to identify patterns and underlying structures in the data set as well divide as it into two different classes. The separation of the data is clearly successful as there are two distinct clusters as seen in figure 11 in the results. As mentioned earlier, the actual meaning of the two classes is unknown and the division of the data into the classes is solely performed in order to facilitate the application of the two supervised learning techniques.

Starting with the support vector machine, due to the nature of the separation, all three kernels tested are capable of performing well when classifying the observations into either of the two classes. As has been mentioned earlier, research suggests that within bioinformatics, where it is common to have more features than observations, the linear kernel tends to perform the best. In cases where the data is non-linear, in general, the radial kernel is the most successful. Due to this data set having many more features than observations a likely suitable kernel is therefore the linear one.

The optimization and cross-validation process on the training data shows that all three kernels, when optimally tuned, are capable of separating the data equally well in regards to the prediction (classification) accuracy expressed as AUC values. Identical are also the corresponding confidence intervals of the AUC values. However, worth noting is that the optimal polynomial kernel is of degree 1, meaning that it is equivalent of a linear kernel. Furthermore, the radial kernel has a low value of sigma, resulting in a flexible classifier having substantial curvature with little resemblance to the other two kernels.

Another apparent difference between the radial kernel and the other two kernels is the different optimal values of the cost parameter. The linear and the polynomial kernels both have a cost parameter value of 0.5 while the radial kernel has a value of 36. The polynomial kernel has a scale parameter of value 1, which just means that no scaling has taken place.

In regards to the radial kernel, generally, in cases where there are more features than observations, research suggests there is a risk of overfitting the data. Normally this applies to the polynomial kernel as well, however as it is in practice a linear kernel in this case, it is treated as a linear kernel. As previously mentioned, having too high values of the cost

parameter lead to overfitting and the comparatively higher value of the radial kernels cost parameter could be a sign of just this. The identical cost values of the linear and the polynomial kernel is logical as both the classifiers are linear and in fact equal.

By solely considering the results of the training procedure, as all three kernels (or two as the linear and polynomial are equal) perform equally in terms of AUC values and confidence intervals, there is no obvious reason to choose one over the other as a classifier of the test data. In order to further investigate any differences between them, the trained kernels are tested on a version of the data set with added noise, as been previously explained. The reasoning here is that the k-means clustering creates two classes that are too cleanly separated. As a consequence, any of the three classifiers can too easily be fitted between the classes, thereby, rendering any comparisons between the kernels redundant. Additionally, if there actually is overfitting in the case of the radial kernel, it might be expressed as the kernel performing worse on the noised-up data set, compared to the other kernels. The results of applying the classifiers to the noised-up data set show that the radial kernel does indeed perform worse than the linear and polynomial kernels, which could be an indication of overfitting. As the linear and polynomial kernels are identical and slightly outperform the radial kernel, the linear kernel is used on the test data.

As the training of the random forest algorithm is straightforward, discussing the process further is unnecessary. When comparing the results of the random forest and the support vector machine using a linear kernel on the training data, the random forest performs slightly better. However, this does not necessarily mean that the random forest performs better when applied to the test data. As mentioned, in the case of the support vector machine, allowing for a certain amount of misclassifications on the training data can result in a better classifier on the test data. Yet, examining the results of the application of the two classifiers on the test data, the random forest does indeed classify with higher accuracy. The support vector machine manages to correctly classify 10 out of 11 observations while the random forest manages to correctly classify all 11. That the support vector machine has a lower bound of the confidence interval of its AUC value further confirms the greater accuracy of the random forest. Important to note is that, although the support vector machine is inferior to the random forest in this case, it still achieves a classification accuracy above 0.9, which is regarded as an excellent result.

As this result is based on classifying a test data set only consisting of 11 observations and where both classifiers perform very well, applying them to another, preferably larger, test data set in order to investigate if the results still hold is an interesting notion worth exploring if the possibility had existed. Nevertheless, on this data set the random forest is the best classifier and is therefore used to construct a parsimonious model.

In regards to selecting the appropriate features for the parsimonious model, when comparing the training results of the random forest using five, ten and all features, the three perform very similarly when considering their evaluation measures. The only result that appears to be slightly surprising at first sight, is that the random forest with ten features is optimized in such a way that it classifies all training observations correctly. However, this is just interpreted as, when the random forest uses ten features, the algorithm is optimized when it classifies all training observations correctly.

Moving over to classifying the test set, the results show that selecting the five most important features when it comes to affecting the classification, gives as high classification accuracy as using ten or all features. This indicates that a parsimonious model only having five features manages to classify the observations of the test set as accurately as using all the features. As in the case when comparing the support vector machine and the random forest, employing the random forest on a test set with more observations in order to explore whether the same selected features as well as the same number of features are relevant is an interesting idea.

As has been touched upon above, there are several reasons to why it is an interesting extension to this thesis to apply and evaluate these random forest and support vector machine algorithms to additional protein biomarker data sets. Another reason is to further investigate the classifiers' generalizability. As has been mentioned before, both the random forest and the support vector machine tend do generalize well, even when trained on limited training sets. The results of this thesis show that the classification performance of both the random forest and the support vector machine is excellent in the case of the test data, indicating that the generalizability of the classifiers is good. Notwithstanding the signs of generalizing well, due to the relatively few observations in both the training and the test data set and that the cross-validation procedure is data-driven, the generalizability could still be questioned. Therefore, in order to further explore this issue, applying these tuned classifiers to additional, preferably larger data sets, is a conceivable notion.

7.Conclusions

In this thesis, the two supervised learning techniques, the random forest and the support vector machine, are compared and evaluated in the context of binary classification of a protein biomarker data set. Finally, in regards to the best performing classifier, a parsimonious model is derived where the most important features are included. This thesis is a two-stage process as the data set originally only consisted of features. Preceding the stage of comparing the models, using k-means clustering, a binary class variable was created, which is a construct of the underlying structure of the data. This variable enables the comparison of the two models performance on the test set using the ROC values and their respective confidence intervals. To recapitulate, the research questions in this thesis are the following:

(1) *How successful are the two supervised learning techniques, the random forest and the support vector machine, in the case of classifying the specific data set provided and how do the two methods compare to each other?*

(2) *In regards to which method performs the best, based on feature importance measures, how does a final parsimonious model perform?*

In this thesis, the optimally tuned random forest, with 6 features tried at each split, outperformed the optimally tuned linear kernel support vector machine with cost = 1, in terms of classification accuracy on the test data. Important to note is that all support vector machines performed exceptionally well with ROC values above 0.9. When extracting the ten most important features in the random forest algorithm for classification, the results show that the random forest performed exceptionally well using both ten features and five features. Although the uncertainty of the ROC when using only five features is slightly higher compared to the case when using all features, the lower bound of its confidence interval is still above 0.95. This indicates that only using five features is more than sufficient in order to successfully classify the observations in the test set.

This thesis is, at best, a rough start when working with unsupervised and supervised learning techniques within the context of this protein biomarker data set. The results indicate that the random forest outperforms the support vector machine, however, would this conclusion stand if there existed additional classes or perhaps a third class for outliers? Identifying completely new classes by more closely investigating the multivariate structure of the data set using other

unsupervised learning methods is a possible subject for further research. Furthermore, other supervised learning techniques such as neural networks, and the shrunken centroid method can be included in the analysis.

As has been previously touched upon, although they generally generalize well, another interesting extension of this thesis is to further explore the capabilities and generalizability of the random forest and the support vector machine by applying them to additional protein biomarker data sets. If the case is that the classifiers perform well on other data sets, it is a sign that they have good generalizability within the specific area of classifying observations based on the protein biomarkers used in this thesis.

References

Alpaydin, E. (2010). *Introduction to machine learning*. MIT press.

Babu, M. M. (2004). Introduction to microarray data analysis. *Computational Genomics: Theory and Application*, 225-249.

Ballman, K. V. (2008). Genetics and Genomics Gene Expression Microarrays. *Circulation*, *118*(15), 1593-1597.

Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., & Rätsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Comput Biol*, *4*(10), e1000173.

Ben-Hur, A., & Weston, J. (2010). A user's guide to support vector machines. In *Data mining techniques for the life sciences* (pp. 223-239). Humana Press.

Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah?. *ACM SIGKDD Explorations Newsletter*, *2*(2), 1-13.

Bewick, V., Cheek, L., & Ball, J. (2004). Statistics review 13: receiver operating characteristic curves. *Critical care*, *8*(6), 508.

Biau, G., Devroye, L., & Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *The Journal of Machine Learning Research*, *9*, 2015-2033.

Bramer, M. (2007). *Principles of data mining* (Vol. 131). London: Springer.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Breiman, L. & Cutler, A, (2004), Berkley University, Program rf5new. Retrieved 2015-12-10 from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_examples/prog.f

Breiman, L. & Cutler, A, (n.d.), Berkley University, Random Forests. Retrieved 2015-11-12 from
https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.I. (1984). *Classification and regression trees*. Belmont, Calif.: Wadsworth.

Brereton, R. G., & Lloyd, G. R. (2010). Support vector machines for classification and regression. *Analyst*, *135*(2), 230-267.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., ... & Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, *97*(1), 262-267.

Byun, H., & Lee, S. W. (2002). Applications of support vector machines for pattern recognition: A survey. In *Pattern recognition with support vector machines* (pp. 213-236). Springer Berlin Heidelberg.

Caetano, S., Aires-de-Sousa, J., Daszykowski, M., & Vander Heyden, Y. (2005). Prediction of enantioselectivity using chirality codes and classification and regression trees. *Analytica Chimica Acta*, *544*(1), 315-326.

Camm, J., Cochran, J., Fry, M., Ohlmann, J. & Anderson, D. (2014). *Essentials of Business Analytics* (1st edition). Cengage Learning, Florence, United States

Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (pp. 161-168). ACM.

Cleland, C. M., Rothschild, L., & Haslam, N. (2000). Detecting latent taxa: Monte Carlo comparison of taxometric, mixture model, and clustering procedures. *Psychological reports*, *87*(1), 37-47.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine Learning Techniques for Multimedia* (pp. 21-49). Springer Berlin Heidelberg.

Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, *88*(11), 2783-2792.

Cutler, A., & Stevens, J. R. (2006). [23] Random Forests for Microarrays. *Methods in enzymology*, *411*, 422-432.

Díaz-Uriarte, R., & De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, *7*(1), 3.

Eckardt, N. A. (2004). Cutting Edge Transcriptome Analysis: It's All about Design. *The Plant Cell*, *16*(9), 2249-2251.

Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

El-Naqa, I., Yang, Y., Wernick, M. N., Galatsanos, N. P., & Nishikawa, R. M. (2002). A support vector machine approach for detection of microcalcifications. *Medical Imaging, IEEE Transactions on*, *21*(12), 1552-1563.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, *16*(10), 906-914.

Ham, J., Chen, Y., Crawford, M. M., & Ghosh, J. (2005). Investigation of the random forest framework for classification of hyperspectral data. *Geoscience and Remote Sensing, IEEE Transactions on*, *43*(3), 492-501.

Han, H., & Jiang, X. (2014). Overcome support vector machine diagnosis overfitting. *Cancer informatics*, *13*(Suppl 1), 145.

Hastie, T., Friedman, J., & Tibshirani, R. (2009). *The elements of statistical learning* (2nd edition). Springer: Springer Series in Statistics.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. Technical report. Department of computer science and information engineering. *National Taiwan University, Taipei*. Retrieved 2015-12-12 from https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf

Ishwaran, H., & Kogalur. U. B. (2015). Package "randomForestSRC" 2015. Retrieved 2015-12-10 from https://cran.r-project.org/web/packages/randomForestSRC/randomForestSRC.pdf

Jakkula, V. (2006). Tutorial on support vector machine (svm). *School of EECS, Washington State University*. Retrieved 2015-12-08 from http://www.ccs.neu.edu/course/cs5100f11/resources/jakkula.pdf

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. New York: springer.

Jia, S., Hu, X., & Sun, L. (2013). The Comparison between Random Forest and Support Vector Machine Algorithm for Predicting β-Hairpin Motifs in Proteins. *Engineering*, *5*(10), 391.

Karatzoglou, A., Meyer, D., & Hornik, K. (2006). Support vector machines in R. *Journal of Statistical Software, 15(*9*).* 1-28.

Karatzoglou, A., Smola, A., Hornik, K., & Karatzoglou, M. A. (2015). Package 'kernlab'. Retrieved 2015-12-14 from https://cran.r-project.org/web/packages/kernlab/kernlab.pdf.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Informatica, 31*(3), 249-268.

Kumar, R., & Indrayan, A. (2011). Receiver operating characteristic (ROC) curve for medical researchers. *Indian pediatrics*, *48*(4), 277-287.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*(1), 169-186.

Mountrakis, G., Im, J., & Ogole, C. (2011). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *66*(3), 247-259.

Pharma Consulting Group. (2015). About us. Retrieved 2015-11-27 from http://clinical.pharmaconsultinggroup.com/about-us/

Rios, G., & Zha, H. (2004). Exploring Support Vector Machines and Random Forests for Spam Detection. In *CEAS July 2004*. Retrieved 2015-12-04 from http://ceas.cc/2004/174.pdf
Salkind, N. J. (Ed.). (2010). *Encyclopedia of research design* (Vol. 1). Sage.

Sarstedt, M., & Mooi, E. (2014). *A concise guide to market research: the process, data, and methods using IBM SPSS statistics*. Springer.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Statnikov, A., Wang, L., & Aliferis, C. F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, *9*(1), 319.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, *43*(6), 1947-1958.

Tape, T. G. (2006). Interpreting diagnostic tests. *University of Nebraska Medical Center*. Retrieved 2015-12-11 from http://gim.unmc.edu/dxtests/ROC3.htm.

Templin, M. F., Stoll, D., Schrenk, M., Traub, P. C., Vöhringer, C. F., & Joos, T. O. (2002). Protein microarray technology. *Drug Discovery Today*, *7*(15), 815-822.

Von Luxburg, U., & Schölkopf, B. (2008). Statistical learning theory: models, concepts, and results. *arXiv preprint arXiv:0810.4752*.

Wang, L. (2005). *Support Vector Machines: theory and applications* (Vol. 177). Springer Science & Business Media.

Yu, H., & Kim, S. (2012). SVM Tutorial—Classification, Regression and Ranking. In *Handbook of Natural Computing* (pp. 479-506). Springer Berlin Heidelberg.