Stockholms universitet

Logistic Regression versus Support Vector Machines

Alexander Nöu

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2018:20
Matematisk statistik
Juni 2018

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2018:20**
http://www.math.su.se

Logistic Regression versus Support Vector Machines

Alexander Nöu*

June 2018

Abstract

In binary classification, the objective is to identify if an observation belongs to one of two classes, on the basis of data consisting of observations with known classes. Logistic regression and support vector machines are two of several classification methods, where the former is a traditional statistics method, and the latter is part of the closely related field of machine learning. The purpose of this thesis is to analyze and compare these methods, both theoretically and practically.

In the theoretical part, each method is described in detail, from concept to model fitting. Despite their different approaches to the classification problem, it turns out that the parameters for both methods can be obtained by minimizing an objective function, consisting of a loss and penalty function. The logistic regression loss and the support vector machine loss behave similarly, and as a result, they often have similar prediction accuracies.

The practical part consists of four different experiments, where in each experiment, a number of binary data sets are simulated. The aim is to analyze how each classifier performs on different types of data, by varying a number of parameters that characterize a data set.

The results coincided with the theory, showing that the predictive power of logistic regression and the support vector machine was close to equal (a few minor differences were observed). It was also shown that each method can be modified in order to obtain some of the advantages of the other method, making them even more alike.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: alexander.lacson.nou@gmail.com. Supervisor: Ola Hössjer and Disa Hansson.

Acknowledgements

I would like to thank my supervisors Ola Hössjer and Disa Hansson for their valuable guidance, suggestions and feedback during the writing of this thesis.

Contents

1 Introduction

In statistics and machine learning, *classification* is the process of predicting the category (or class), that an observation belongs to, on the basis of data where the category of each observation is known. For example, identifying one of several medical conditions or predicting the winner of a horse race both represent multiclass classification problems.

There exists a number of methods for classification. Different methods suit well for different types of situations, and different types of data. This thesis will focus on binary (two classes) classification, where the goal is to compare and analyze two popular classifiers, namely *logistic regression* and the *support vector machine*, on a variety of simulated data sets.

1.1 Background

While logistic regression (LR) comes from classical statistics, support vector machines (SVMs) take a geometric approach on classifying data. The method gained a lot of attention when it was introduced in the mid 1990s [8], and is today one of the more popular classification methods in machine learning communities, whereas LR is the commonly used method in statistics. Even though their approaches to the classification problem are very different, there is actually a connection between the two (which is shown in section 2.4). According to [8], SVMs may yield slightly better results when classes are well separated, while for more overlapping data, logistic regression is often preferred. In general, their prediction accuracies are supposedly similar on most types of data.

The goal of the thesis is to dissect the theory behind these two methods, and discuss their strengths, weaknesses and similarities. We will consider aspects such as predictive power, interpretability and computational complexity. The practical part of the project aims to investigate how SVMs and LR perform on different types of data, by varying a number of parameters that characterize data.

1.2 Outline

The thesis begins in section 2, presenting the theory for LR and SVMs. It is followed by a description of the performance measures that are being used in the simulation study, and a method for parameter tuning. The

theory section ends by showing how the two classifiers are related theoretically (section 2.4). Section 3 describes the practical part of the thesis. It presents a method for simulating binary data sets, and describes how the final models are fitted for each method. The results of the simulations are thereafter presented in section 4, and section 5 contains a discussion of the advantages/disadvantages that come with each method.

2 Theory

In binary classification, the response variable Y is categorical with two possible outcomes (typically 0 and 1, or -1 and 1), and the p explanatory variables $\mathbf{X} = (X_1, \ldots, X_p)^T$ can be either continuous or categorical (we stick to continuous variables in this project). The following section covers the theory for how logistic regression and support vector machines predict the outcome of Y for new observations.

2.1 Logistic Regression

The theory for logistic regression models is taken from [1], unless stated otherwise.

Let Y take either of the values 1 and 0, and let

$$\pi(\mathbf{x}) = P(Y = 1 \mid \mathbf{X} = \mathbf{x}) = 1 - P(Y = 0 \mid \mathbf{X} = \mathbf{x}).$$

The multiple logistic regression model is

$$\pi(\mathbf{x}) = \frac{\exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)}{1 + \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)} = \frac{\exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}{1 + \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}, \tag{2.1}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ contains the effect parameters for all explanatory variables. Alternatively, taking the log odds, or logit, the model can be formulated as the linear combination

$$\text{logit}[\pi(\mathbf{x})] = \log \frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p. \tag{2.2}$$

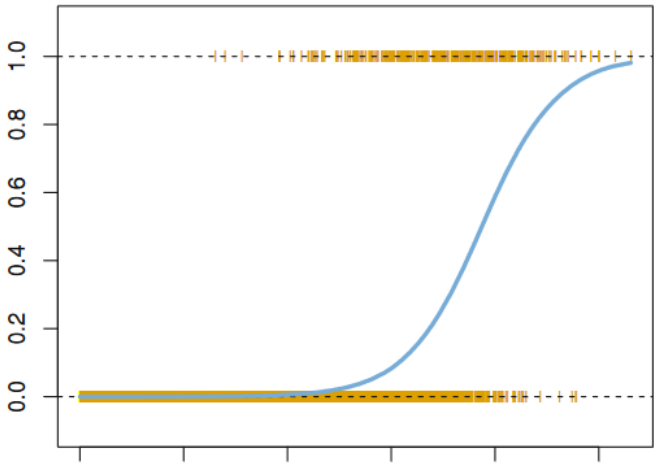

Figure 1: *The logistic regression model, fitted to some data. The orange ticks represent the binary outcomes (*0 *or* 1*) of the given data, and the curve is the logistic regression model's estimated probabilities for class* 1*. The figure is from [8].*

Thus, the LR model provides the conditional probabilities for the outcome of Y based on the predictor variable values $\mathbf{x} = (x_1, \ldots, x_p)^T$, and some parameters β_0 and $\boldsymbol{\beta}$ that need to be estimated. For $p = 1$, the estimated probabilities can be illustrated as an S-shaped curve fitted to the given data (Figure 1). The coefficients β_i are interpreted as the increase in conditional log odds when increasing x_i by one unit, holding the other x_j fixed.

Since the LR model provides estimated probabilities, it does not classify observations. However, the model can easily be turned into a classifier if we choose a cutoff c such that if

$$P(Y = 1 \mid \mathbf{X} = \mathbf{x}^*) > c$$

we assign the new observation $\mathbf{x}^*$ to class 1, otherwise to 0. The choice of c may vary, depending on the situation, and the data [8]. For example, the focus of this project lies on prediction accuracy, where data will be simulated in such a way that the outcome of the response variable is balanced. Therefore, a cutoff $c = 0.5$ is being used throughout the simulations. This means that we choose the class that has the highest probability. Using (2.1) and (2.2), the classification condition

$$\pi(\mathbf{x}^*) = \frac{\exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}^*)}{1 + \exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}^*)} > 0.5,$$

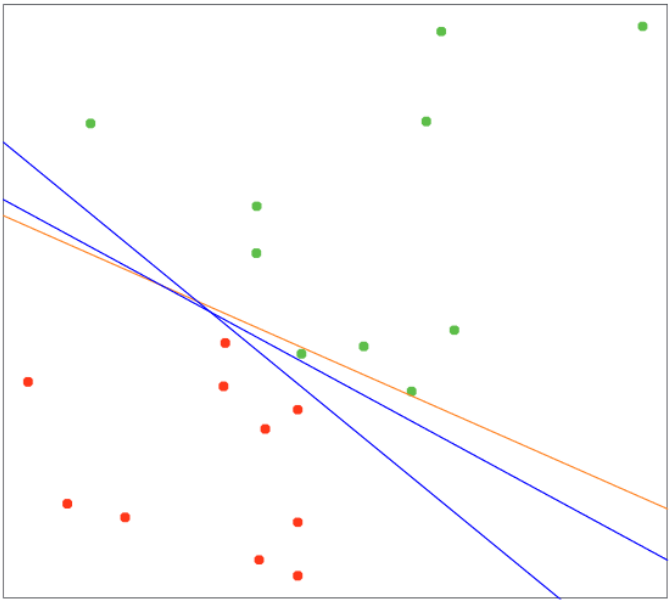

Figure 2: *The figure shows three different decision boundaries in the two-dimensional feature space. The binary outcome of the given observations is represented by the color of each point. The figure is from [7].*

can be rewritten as

$$\text{logit}[\pi(\mathbf{x}^*)] = \beta_0 + \boldsymbol{\beta}^T\mathbf{x}^* > 0.$$

The observation $\mathbf{x}^*$ is identified as class 1 for a positive log odds, and as 0 for a negative one. The class of an observation therefore depends on a linear combination of the explanatory variables. This is called a *linear classifier* [7]. The easiest way of illustrating a linear classifier is by plotting the predictors against each other in what is known as the *feature space* (feature is the term used for predictor variables in the field of machine learning). In the feature space, a *decision boundary* $\beta_0 + \boldsymbol{\beta}^T\mathbf{x} = 0$ corresponds to a hyperplane, and for $p = 2$ the boundary is a line (Figure 2). Which side of the boundary a new observation falls on decides its predicted class.

2.1.1 Fitting Logistic Regression Models

For a given data set (often called *training data*), let $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ denote the values of each predictor, and let y_i denote the outcome, for observation $i = 1, \ldots, N$. Only using continuous predictor variables results

in one outcome per setting $\mathbf{x}_i$. In addition, since the response is binary, $y_1, \ldots, y_N$ are drawn from independent binomial distributions with $n = 1$ trials. The likelihood function is

$$
\begin{aligned}
L(\beta_0, \boldsymbol{\beta}) &= \prod_{i=1}^{N} \pi(\mathbf{x}_i)^{y_i} [1 - \pi(\mathbf{x}_i)]^{1-y_i} \\
&= \prod_{i=1}^{N} \left(\frac{\pi(\mathbf{x}_i)}{1 - \pi(\mathbf{x}_i)} \right)^{y_i} \prod_{i=1}^{N} [1 - \pi(\mathbf{x}_i)] \\
&= \exp \left(\sum_{i=1}^{N} y_i \log \frac{\pi(\mathbf{x}_i)}{1 - \pi(\mathbf{x}_i)} \right) \prod_{i=1}^{N} [1 - \pi(\mathbf{x}_i)] \\
&= \exp \left[\sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right] \prod_{i=1}^{N} \left[1 + \exp \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right]^{-1},
\end{aligned}
$$

where we used the LR model ((2.1) and (2.2)) in the last equality. The log likelihood becomes

$$
\sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) - \sum_{i=1}^{N} \log \left[1 + \exp \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right]. \tag{2.3}
$$

Through maximization of the log likelihood (2.3), the intercept β_0 and the coefficients $\boldsymbol{\beta}$ are estimated. This is done by first setting the partial derivatives equal to 0:

$$
\begin{aligned}
\frac{\partial \log L(\beta_0, \boldsymbol{\beta})}{\partial \beta_0} &= \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} \frac{\exp \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} \right)}{1 + \exp \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} \right)} = 0, \\
\frac{\partial \log L(\beta_0, \boldsymbol{\beta})}{\partial \beta_j} &= \sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} x_{ij} \frac{\exp \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} \right)}{1 + \exp \left(\beta_0 + \sum_{k=1}^{p} \beta_k x_{ik} \right)} = 0,
\end{aligned}
$$

$j = 1, \ldots, p$. The likelihood equations are then expressed as

$$
\begin{aligned}
&\sum_{i=1}^{N} y_i - \sum_{i=1}^{N} \hat{\pi} = 0, \\
&\sum_{i=1}^{N} y_i x_{ij} - \sum_{i=1}^{N} \hat{\pi} x_{ij} = 0, \quad j = 1, \ldots, p,
\end{aligned}
$$

where $\hat{\pi}_i = \exp\left(\hat{\beta}_0 + \sum_{k=1}^{p} \hat{\beta}_k x_{ik}\right) / \left[1 + \exp\left(\hat{\beta}_0 + \sum_{k=1}^{p} \hat{\beta}_k x_{ik}\right)\right]$ are the maximum likelihood estimates of the conditional probabilities $\pi(\mathbf{x}_i)$. These $p + 1$ equations are nonlinear and can be solved with the *Newton-Raphson* algorithm or coordinate descent methods [7].

2.1.2 Shrinkage Methods

In model selection, one seeks for a subset of variables that makes the model easier to interpret and possibly yields higher prediction accuracy. Methods such as *forward-* and *backward-stepwise selection* [8] are commonly used and may work well when the number of predictors is not too large and when variables are not highly correlated. Because of the discrete steps involved, where variables are either included or excluded, these methods often suffer from high variance and the prediction error might not be reduced compared to the full model [7]. Since this study focuses on predictive power, where the simulated data will be of both high dimension and correlations, we will instead be using a combination of *shrinkage methods* (also called *regularization methods*), namely *ridge regression* and the *lasso*. According to [7], they are more continuous, which results in lower variance.

Ridge regression [10] combats overfitting and collinearity by restricting the sizes of the regression coefficients. Instead of maximizing the log likelihood (2.3), we introduce a penalized version:

$$\sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right) - \sum_{i=1}^{N} \log\left[1 + \exp\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)\right] - \lambda \sum_{j=1}^{p} \beta_j^2,$$

where the ridge parameter λ controls the amount of shrinkage of coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$. The inputs are normally standardized before optimizing the penalized log likelihood, and the intercept is left out of the penalty term. In [10], it is explained that a large number of predictors, and/or highly correlated data will result in unstable parameter estimates, but through shrinking of the coefficients toward 0, we obtain a more stabilized system that yields estimates with smaller variance. Ridge regression, however, does not shrink parameters exactly to 0, resulting in all variables being kept in the model.

The lasso [7], or LASSO (least absolute shrinkage and selection operator), also puts a constraint on the coefficients through a penalty term. For logistic regression, the function that is to be maximized is

$$\sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) - \sum_{i=1}^{N} \log \left[1 + \exp \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right] - \lambda \sum_{j=1}^{p} |\beta_j|.$$

Unlike ridge regression, the lasso may shrink coefficients to be exactly 0 and thus works for variable selection.

A combination of ridge regression and the lasso, called *elastic net*, will be used in the simulations. Elastic net works as a shrinkage and selection method, where the log likelihood is maximized with the following penalty term [5]:

$$\sum_{i=1}^{N} y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) - \sum_{i=1}^{N} \log \left[1 + \exp \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right] - \lambda P_\alpha(\boldsymbol{\beta}),$$

where

$$P_\alpha(\boldsymbol{\beta}) = \sum_{j=1}^{p} [(1-\alpha)\beta_j^2 + \alpha|\beta_j|].$$

The lasso corresponds to $\alpha = 1$, while for ridge regression $\alpha = 0$. The optimal choice of λ and α is obtained through cross-validation (section 2.3). The R package *glmnet* [6] provides a function for logistic regression with the use of elastic net, where the penalized log likelihood is maximized with coordinate descent methods.

2.2 Support Vector Machines

This section introduces the second classifier of the thesis; support vector machines [2]. The theory is taken from [7] and [8], if nothing else is mentioned. In section 2.1, a decision boundary in the two-dimensional feature space was illustrated (Figure 2). We will once again use this setting for illustrations, as it helps understanding the concept behind SVMs.

2.2.1 Maximal Margin Classifier

As with logistic regression, let Y be the binary response variable, but with outcomes 1 and -1, and let $\mathbf{X} = (X_1, \ldots, X_p)^T$ be the explanatory variables.

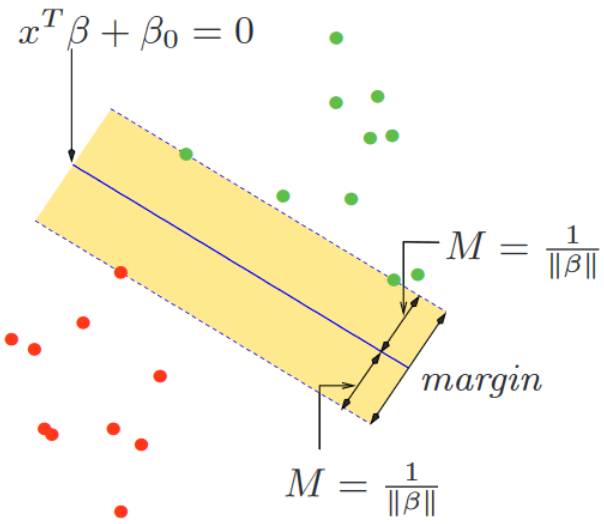

Figure 3: *The figure shows the case where two classes are perfectly separable. The solid line represents a decision boundary for the maximal margin classifier, and the broken lines are the margin borders. The figure is from [7].*

We begin by considering the simplest case, where two classes can be perfectly separated by a hyperplane (a line for $p = 2$, Figure 3). This separating hyperplane (an infinite of such hyperplanes exists) has the property that for some vector $\boldsymbol{\beta}$ and scalar β_0,

$$\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} > 0 \quad \text{if } y_i = 1$$

and

$$\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} < 0 \quad \text{if } y_i = -1,$$

for all observations $i = 1, \ldots, N$. These can equivalently be formulated as

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) > 0.$$

Note that this is not a condition for classification, but a property of the separating hyperplane. We can, however, directly use the hyperplane as a

decision boundary to create a linear classifier: if $f(\mathbf{x}^*) = \beta_0+\beta_1 x_1^*+\ldots+\beta_p x_p^*$ is positive, we assign the new observation $\mathbf{x}^*$ to class 1. If $f(\mathbf{x}^*)$ is negative, we assign it to -1. For $p = 2$, we classify $\mathbf{x}^*$ based on whichever side of the line it lands on.

Since there will exist an infinite amount of separating hyperplanes, one naturally wishes to seek out the "optimal" decision boundary. The *maximal margin classifier* (or optimal separating hyperplane) is a linear classifier that maximizes the distance between the hyperplane and the closest observations from both classes (Figure 3). This distance is called *margin* (hence the name of the classifier), and the closest observations are *support vectors*. The maximal margin hyperplane only depends on the support vectors and would therefore not be affected by moving the other observations as long as they do not cross the margins.

Before getting to the maximization of the margins, some linear algebraic properties need to be listed. For any point $\mathbf{x}_0$ lying in the hyperplane $\beta_0 + \boldsymbol{\beta}^T\mathbf{x} = 0$ (Figure 4),

$$\boldsymbol{\beta}^T\mathbf{x}_0 = -\beta_0. \tag{2.4}$$

It directly follows that for any two points $\mathbf{x}_0$ and $\mathbf{x}_1$ lying in $\beta_0 + \boldsymbol{\beta}^T\mathbf{x} = 0$, $\boldsymbol{\beta}^T(\mathbf{x}_0 - \mathbf{x}_1) = 0$, which means that $\boldsymbol{\beta}^* = \boldsymbol{\beta}/||\boldsymbol{\beta}||$ is a vector normal to the the surface of the hyperplane. The distance of any point $\mathbf{x}$ to the hyperplane (negative for points of class -1) can be written as

$$\boldsymbol{\beta}^{*T}(\mathbf{x} - \mathbf{x}_0) = \frac{1}{||\boldsymbol{\beta}||}(\boldsymbol{\beta}^T\mathbf{x} - \boldsymbol{\beta}^T\mathbf{x}_0) = \frac{1}{||\boldsymbol{\beta}||}(\boldsymbol{\beta}^T\mathbf{x} + \beta_0), \tag{2.5}$$

where we first projected $\mathbf{x} - \mathbf{x}_0$ onto $\boldsymbol{\beta}$, then used (2.4) in the last equality. The optimization of the margin can now be formulated as

$$\begin{aligned} &\max_{\beta_0,\boldsymbol{\beta}} \ M \\ &\text{subject to } \frac{1}{||\boldsymbol{\beta}||} y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq M, \ i = 1, \ldots, N. \end{aligned}$$

The signed distance from any point to the hyperplane was given in equation (2.5). The constraints for the optimization problem therefore ensure that no points are inside the margin, which is to be maximized. By rewriting the constraint as

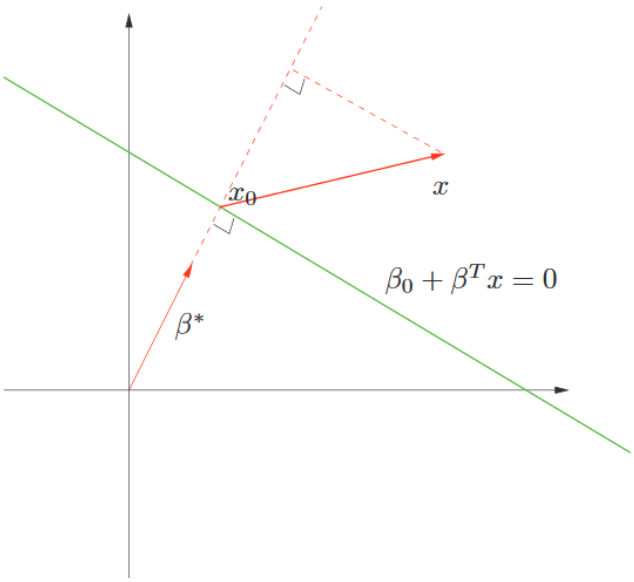

Figure 4: *The linear algebra of a hyperplane. The figure is from [7].*

$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M||\boldsymbol{\beta}||,$$

we see that for any $\boldsymbol{\beta}$ and β_0 satisfying the inequality, any positively scaled multiple does as well. This means that we can arbitrarily choose $||\boldsymbol{\beta}|| = 1/M$. Thus, we wish to maximize a margin with thickness $1/||\boldsymbol{\beta}||$, which is equivalent to minimizing $\frac{1}{2}||\boldsymbol{\beta}||^2$ (for mathematical convenience). Finally, the optimization problem is rewritten as

$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}} \ \frac{1}{2}||\boldsymbol{\beta}||^2 \\ &\text{subject to } y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1, \ i = 1, \ldots, N. \end{aligned} \tag{2.6}$$

This convex optimization problem can be simplified by the use of Lagrange multipliers $\alpha_1, \ldots, \alpha_N$, which is a method for constrained optimization [14]. First, we set up the Lagrange function

$$L = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1], \tag{2.7}$$

and thereafter set the derivatives to zero:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0,$$
$$\frac{\partial L}{\partial \beta_0} = -\sum_{i=1}^{N} \alpha_i y_i = 0.$$

Substituting $\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$ in (2.7) gives us

$$\begin{aligned} L &= \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k\right) - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k \\ &\quad - \sum_{i=1}^{N} \alpha_i y_i \beta_0 + \sum_{i=1}^{N} \alpha_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k. \end{aligned} \tag{2.8}$$

This is called a *Wolfe dual* problem [16]; a simpler optimization problem than the original one, and the solution to the original problem can be found by instead maximizing (2.8), subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$\alpha_i [y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1] = 0,$$

and $\alpha_i \geq 0$, $i = 1, \ldots, N$. From the constraints, we see that if $\alpha_i > 0$, then $y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) = 1$ and $\mathbf{x}_i$ is on the margin border. The maximal margin solution $\hat{\boldsymbol{\beta}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i$ therefore depends on a linear combination of only the support vectors, and the classifier is the function

$$\hat{f}(\mathbf{x}^*) = \mathbf{x}^{*T} \hat{\boldsymbol{\beta}} + \hat{\beta}_0 = \mathbf{x}^{*T} \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i + \hat{\beta}_0 = \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}^{*T} \mathbf{x}_i + \hat{\beta}_0,$$

where the class of $\mathbf{x}^*$ is decided by the sign of $\hat{f}(\mathbf{x}^*)$.

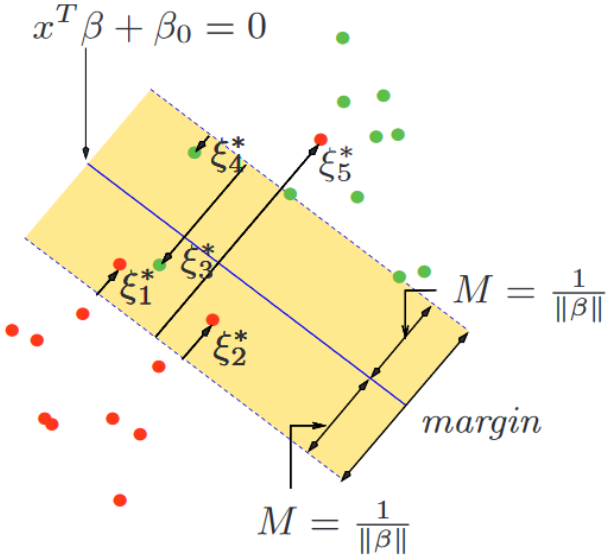

Figure 5: *The nonseparable case. The solid line represents a decision boundary for the support vector classifier. Points ξ_i^* are on the wrong side of the margin by a distance $\xi_i^* = M\xi_i$. The figure is from [7].*

2.2.2 Support Vector Classifier

The maximal margin classifier can be used when classes are perfectly separable. However, the more common case is when classes are overlapping and no such classifier exists. By introducing a *soft margin*, which is to be maximized while allowing some observations to be on the wrong side of the margin, or even on the wrong side of the hyperplane, we get what [8] calls the *support vector classifier* (also known as *soft margin classifier*).

Let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^T$ denote the proportional distances of which each observation is on the wrong side of the margin (Figure 5). An observation is therefore on the right side of the margin if $\xi_i = 0$. If $\xi_i > 1$, the distance is larger than the margin, which means that the observation is on the wrong side of the decision boundary. As with the maximal margin classifier, we seek to maximize a margin M, subject to a set of constraints. The constraints

$$\frac{1}{||\boldsymbol{\beta}||} y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq M(1 - \xi_i), \ i = 1, \ldots, N,$$

once again ensure that each observation is at least a distance away from the hyperplane. For the support vector classifier, the distance does not need to

be the entire margin, since $M(1-\xi_i)$ allows some proportional distance ξ_i to be on the wrong side of the margin. Just as in (2.6), with $M = 1/||\boldsymbol{\beta}||$, the optimization problem is reformulated as

$$\begin{aligned} &\min_{\beta_0,\boldsymbol{\beta}} \ \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to } \xi_i \geq 0,\ y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) \geq 1-\xi_i,\ i=1,\ldots,N. \end{aligned} \tag{2.9}$$

The term $C\sum_{i=1}^{N}\xi_i$ can be interpreted as a form of penalty on the proportional distances on the wrong side of the margin. A larger C results in larger punishment, hence a smaller total sum of errors is allowed. The optimal value for the cost parameter C is obtained through cross-validation (section 2.3). The Lagrange function is

$$L = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i, \tag{2.10}$$

where α_i and μ_i are the multipliers. This function is to be optimized with respect to $\boldsymbol{\beta}$, β_0 and ξ_i. For $\boldsymbol{\beta}$ and β_0, setting the derivatives to zero leads to the same equations as for the maximal margin Lagrange function:

$$\boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i \tag{2.11}$$

and

$$\sum_{i=1}^{N}\alpha_i y_i = 0. \tag{2.12}$$

For the last set of partial derivatives we get

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0,\ i=1,\ldots,N. \tag{2.13}$$

By substituting (2.11), (2.12) and (2.13) into (2.10), the Wolfe dual problem becomes (look back at (2.8))

$$
\begin{aligned}
L &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k + (\alpha_i + \mu_i) \sum_{i=1}^{N} \xi_i \\
&\quad - \alpha_i \sum_{i=1}^{N} \xi_i - \mu_i \sum_{i=1}^{N} \xi_i \\
&= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k. \qquad (2.14)
\end{aligned}
$$

Once again, this is a simpler convex quadratic programming problem compared to the original optimization problem. Equation (2.14) is maximized under constraints (2.12), (2.13) and

$$0 \leq \alpha_i \leq C,$$

$$\alpha_i [y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - (1 - \xi_i)] = 0,$$

$$\mu_i \xi_i = 0,$$

$$y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \geq 0,$$

$i = 1, \ldots, N$. From (2.11), the support vector classifier's solution for $\boldsymbol{\beta}$ is written as

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i, \qquad (2.15)$$

and the decision boundary becomes

$$\mathbf{x}^T \hat{\boldsymbol{\beta}} + \hat{\beta}_0 = \mathbf{x}^T \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}_i + \hat{\beta}_0 = \sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{x}^T \mathbf{x}_i + \hat{\beta}_0 = 0. \qquad (2.16)$$

As with the maximal margin classifier, $\alpha_i > 0$ means that

$$y_i (\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) = 1 - \xi_i. \qquad (2.17)$$

Observation i is therefore a support vector, meaning that it lies on the margin border or on the wrong side of the border. As a result, $\hat{\boldsymbol{\beta}}$ is a linear combination of the support vectors. Equation (2.17) also tells us that β_0 can be solved by using any of the support vectors on the margin border ($\alpha_i > 0$ and $\xi_i = 0$). In addition, it follows from the constraints $\mu_i \xi_i = 0$ and (2.13) that observations on the wrong side of the margin ($\xi_i > 0$) have $\alpha_i = C$.

2.2.3 Support Vector Machines and Kernels

So far, the maximal margin classifier has been introduced for when classes can be perfectly separated by a linear decision boundary, and the extension, the support vector classifier, is used for overlapping classes. The final classifier expands on these methods even further, by generalizing for cases when classes can not be separated well with a linear boundary.

The idea is to enlarge the feature space to a higher dimension where there exists a linear decision boundary. This can be done, for example, by adding higher-order polynomial terms or by transforming the predictors. In the original feature space, the transformations translate to a nonlinear decision boundary (Figure 6). However, problems such as overfitting and heavy computations may occur when adding too many features. The support vector machine is a solution to these issues.

Consider the transformations $\mathbf{h}(\mathbf{x}_i) = [h_1(\mathbf{x}_i), \ldots, h_q(\mathbf{x}_i)]^T$ of the p predictors $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$. The support vector classifier's Lagrange optimization problem (2.14), using the transformed predictors, is

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{h}(\mathbf{x}_i)^T \mathbf{h}(\mathbf{x}_k),$$

with a decision boundary (2.16)

$$\sum_{i=1}^{N} \hat{\alpha}_i y_i \mathbf{h}(\mathbf{x})^T \mathbf{h}(\mathbf{x}_i) + \hat{\beta}_0 = 0.$$

It turns out that the optimization problem and the decision boundary only depend on

$$\mathbf{h}(\mathbf{x}_i)^T \mathbf{h}(\mathbf{x}_k) = \langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_k) \rangle,$$

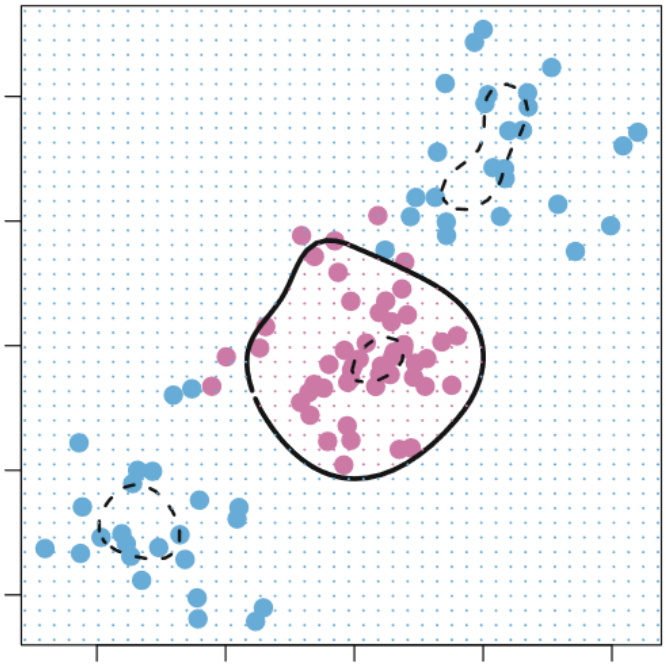

Figure 6: *Two classes separated by a nonlinear decision boundary. The figure is from [8].*

which is the inner product of the transformed predictors. This means that we do not need to find the actual transformations $\mathbf{h}(\mathbf{x}_i)$. Instead, all we need is knowledge of the inner product, called kernel function:

$$K(\mathbf{x}_i, \mathbf{x}_k) = \langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_k) \rangle.$$

Some of the more popular kernels are

$$\begin{aligned} \text{dth-Degree polynomial:} \quad & (1 + \langle \mathbf{x}_i, \mathbf{x}_k \rangle)^d, \\ \text{Radial basis:} \quad & \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_k||)^2. \end{aligned}$$

Kernels are outside the scope of this thesis, since focus will be on data where linear boundaries fit well. The linear kernel $K(\mathbf{x}_i, \mathbf{x}_k) = \mathbf{x}_i^T \mathbf{x}_k$ (the support vector classifier) is therefore sufficient.

Finally, the decision boundary for the support vector machine is

$$\sum_{i=1}^{N} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) + \hat{\beta}_0 = 0.$$

There are computational advantages of using SVMs and kernels, instead of simply enlarging the feature space. The transformations do not need to be computed, which can be difficult when the transformed space is of very high dimension. For some kernels, the feature space may even be infinite-dimensional. One only needs to compute $K(\mathbf{x}_i, \mathbf{x}_k)$ for all distinct pairs i, k, which can be done without working in the transformed feature space.

If the transformations are complex enough, it is likely that a decision boundary that perfectly separates the classes can be found. This is not necessarily a good thing since it may lead to overfitting and thus will not be able to classify new data well. The cost parameter C prevents this by allowing a certain amount of overlapping between classes. By finding the optimal values for the cost- and kernel parameters, the support vector machine manages to balance the transformation of the predictors to capture systematic nonlinear behaviour, and the amount of overlapping classes caused by variance.

2.3 Parameter Tuning

When choosing the penalty parameter for the logistic regression, or the cost- and kernel parameters for the support vector machine, one wishes to find the values that provide the best performing model. This section describes two measures of performance for binary classification (section 2.3.1 and 2.3.2), and a method for finding the optimal parameter values (section 2.3.3).

2.3.1 Misclassification Rate

The misclassification rate, or error rate, is the proportion of incorrectly classified observations made by a model [8]. For the outcomes y_i, $i = 1, \ldots, N$, the misclassification rate is

$$\frac{1}{N} \sum_{i=1}^{N} I(y_i \neq \hat{y}_i),$$

where $\hat{y}_i$ is the predicted class of observation i, and $I(y_i \neq \hat{y}_i)$ is an indicator variable that equals to 1 if the observation is incorrectly classified, and 0 if it is classified correctly.

2.3.2 AUC

For binary classification, there is an alternative performance measure that uses *receiver operating characteristics* (ROC) graphs.

Let the outcome of an observation be labeled as positive or negative. If an observation is positive and a model predicts it as positive, we define it as a *true positive* [4], and if it is predicted as a negative we call it a *false negative*. Moreover, if an observation is negative and also is predicted as such by a model, we define it as a *true negative*, otherwise it is a *false positive*. The *true positive rate* and *false positive rate* can now be defined as

$$\text{TP rate} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

and

$$\text{FP rate} = \frac{\text{False positives}}{\text{True negatives} + \text{False positives}}.$$

A ROC graph is a two-dimensional space where the true positive rate is plotted against the false positive rate (Figure 7). In Figure 7, coordinates $(0, 0)$ correspond to a classifier that classifies zero observations as positive. As a result, none of the positive observations will be classified correctly, and zero negative observations will be incorrectly classified. On the other hand, $(1, 1)$ results in a classifier that only predicts positive outcomes. All positive observations will therefore be correctly classified while all negative ones are misclassified.

By changing a classifier's threshold for when to assign an observation to the positive class, we receive different points on the ROC graph, and if it is varied enough (conceptually from $-\infty$ to ∞), a ROC curve may be computed. The performance of a classifier can be quantified by calculating the *area under the ROC curve* (AUC).

A diagonal line on the ROC graph, having the area 0.5, corresponds to a classifier that randomly guesses the outcome. Classifiers having an AUC value of 1 classify perfectly.

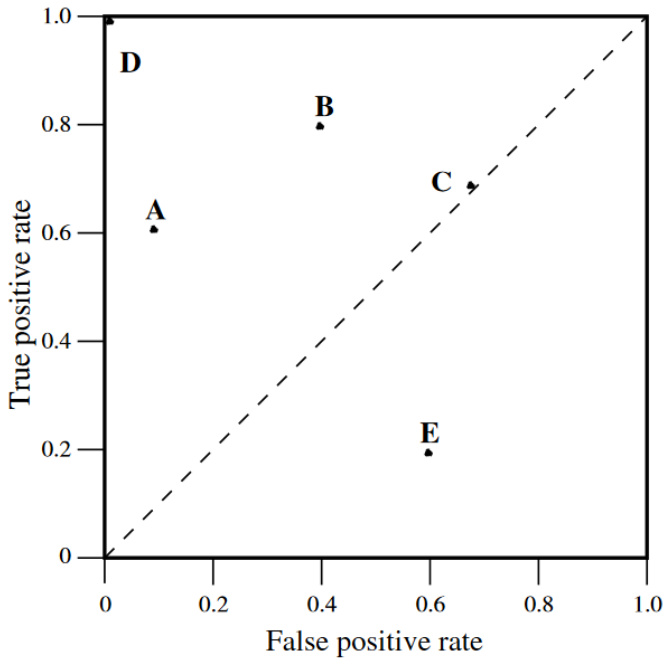

Figure 7: *A ROC graph showing the performance of different classifiers. The figure is from [4].*

2.3.3 Cross-Validation

Dividing observations into a training set and a validation set is a common approach when assessing the performance of a model. The training set is used to fit a model, which is then validated on the other set by some performance measure. The idea is that one does not want to validate a model on the same data that is used for the model fitting process, since that would provide no information on the model's ability to predict new observations and could lead to overfitting the model on the training set. The downside with this approach is when data sets are relatively small, since one has to leave out a portion of the data for validation, which means that there are less observations to fit the model with. In addition, the results may differ, depending on which observations that are in the training and validation set.

K-fold cross-validation [8] extends the idea of splitting up the data, by dividing the observations into K roughly equal-sized parts. We let $K-1$ parts of the data be the training set, and after fitting the model it is used to predict the kth part that was left out. The procedure is repeated for $k = 1, \ldots, K$. Using misclassification rate as measure, the K-fold cross-validation estimate is

$$\mathrm{CV}_{(K)} = \frac{1}{K}\sum_{k=1}^{K}\mathrm{MR}_k,$$

where MR_k is the misclassification rate for the model that was trained on all parts of the data except for the kth part. The choice of K depends on factors such as the size of the data set and if the fitting procedure is computationally intensive. For example, $K = N$, called *leave-one-out* cross-validation, requires the training of a model to be repeated N times, which could be very expensive computationally. Standard choices for K are 5 and 10 [8].

For the classification methods in this thesis, the model selection procedure consists of finding the optimal values for one or several parameters, such as the penalty parameter for shrinkage methods or the cost parameter for SVMs. We seek to minimize

$$\mathrm{CV}(\alpha)_{(K)} = \frac{1}{K}\sum_{k=1}^{K}\mathrm{MR}(\alpha)_k,$$

which is the cross-validation estimate for a model fitted with the tuning parameter α [7]. The value $\hat{\alpha}$ that minimizes the average prediction error is used to fit the final model on all of the data.

2.4 Relationship Between the Classifiers

Let $f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0$ and $\mathbf{y} = (y_1, \ldots, y_N)^T$. It can be shown that the optimization problem (2.9) for the support vector classifier can be reformulated as

$$\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{N} \max\left[0, 1 - y_i f(\mathbf{x}_i)\right] + \lambda||\boldsymbol{\beta}||^2,$$

where λ is a tuning parameter [8]. This is a more general form, consisting of a loss function $L(\mathbf{X}, \mathbf{y}, \boldsymbol{\beta})$ that quantifies the fit of a model on some data $(\mathbf{X}, \mathbf{y})$, and a penalty $P(\boldsymbol{\beta})$ on the model parameters. For support vector classifiers,

$$L(\mathbf{X}, \mathbf{y}, \boldsymbol{\beta}) = \sum_{i=1}^{N} \max[0, 1 - y_i f(\mathbf{x}_i)]$$

is called *hinge loss* and

$$\lambda P(\boldsymbol{\beta}) = \lambda ||\boldsymbol{\beta}||^2$$

is the ridge penalty term, where λ controls the penalty effect.

The penalized logistic regression (section 2.1.2) can also be written on the "loss + penalty" form:

$$L(\mathbf{X}, \mathbf{y}, \boldsymbol{\beta}) = \sum_{i=1}^{N} [\log\{1 + \exp[f(\mathbf{x}_i)]\} - y_i f(\mathbf{x}_i)],$$

with a shrinkage method penalty term, and the loss function being the negative log likelihood. These two methods are therefore not so different from each other, as it may appear at first. They do have different loss functions, but they turn out to be quite similar as well; [8] demonstrates the similarity in behaviour between the SVM hinge loss and the logistic regression loss (also called *binomial deviance*) in Figure 8, which is why logistic regression and SVMs often yield similar results. Note that, in Figure 8, in order to compare the loss of each observation, the outcome is set to $y_i \in \{-1, 1\}$, which results in a slightly different binomial log likelihood than the one that is presented above.

One difference, however, is how the parameters for SVMs only depend on observations on the margin border or on the wrong side of the margin (section 2.2.2). This can be seen in Figure 8, where the loss for observations having $y_i f(\mathbf{x}_i) \geq 1$ (i.e. support vectors (2.17)) is zero. The logistic regression loss is not exactly zero anywhere, but the loss is very small for observations that are on the right side, and far away from the decision boundary.

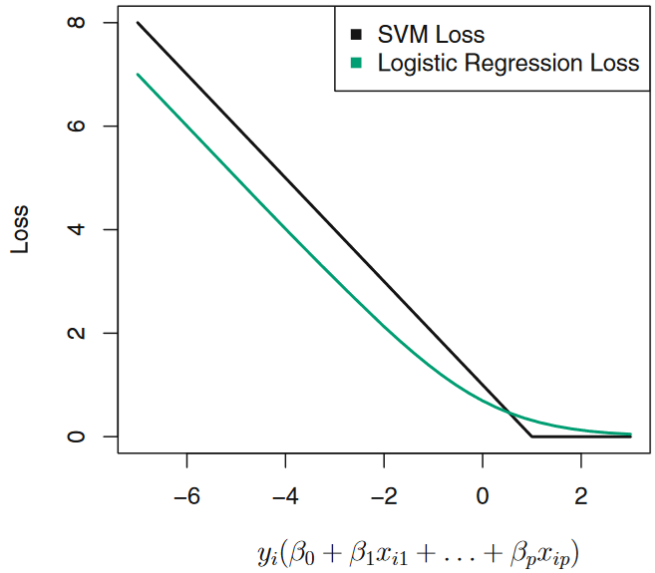

Figure 8: *The hinge loss and logistic regression loss of an observation, plotted against* $y_i f(\boldsymbol{x}_i)$*. The response variable is set to the outcomes* 1 *and* -1*. The figure is from [8].*

3 Simulation and Modeling

This section describes the simulation and model fitting process. Logistic regression and the support vector machine will be evaluated and compared on a number of simulated data sets. The study consists of four larger experiments, where in each experiment, the goal is to isolate and vary some of the parameters. The entire simulation study is done in R.

3.1 Simulation Process

The following method is common for simulating binary data:

For each observation $i = 1, \ldots, N$, the p predictor variable values $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^T$ are randomly generated from a multivariate normal distribution having the probability density function

$$f(\mathbf{x}) = \left(\frac{1}{2\pi}\right)^{p/2} \frac{1}{\sqrt{\det(\boldsymbol{\Sigma})}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right],$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix. The binary outcomes y_i are generated by plugging $\mathbf{x}_i$ and some effect vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ into the logistic regression model (2.1):

$$\pi(\mathbf{x}_i) = \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}{1 + \exp(\boldsymbol{\beta}^T \mathbf{x}_i)},$$

where the intercept is left out to balance the outcomes of the two classes. The generated probabilities are then used to simulate y_i from a Bernoulli distribution with probability $\pi(\mathbf{x}_i)$. Through variation of N, p, $\boldsymbol{\mu}$, Σ and $\boldsymbol{\beta}$, different attributes can be isolated for comparison of performance between the classification methods.

3.2 Experiments

3.2.1 Number of Observations and Predictors

In the first experiment, the aim is to analyze how the classifiers perform when varying the number of observations N and explanatory variables p. Six different settings will be used, where N varies in the range of 200 and 1000, and p is between 20 and 500.

As described in the previous section, predictor values are randomly generated from a multivariate normal distribution. For the first set of simulations, we let the predictors be independent standard normal variables. In other words, values for the mean vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^T$ are

$$\mu_1 = \ldots = \mu_p = 0,$$

and the covariance matrix is the identity matrix. Furthermore, the effect parameters $\boldsymbol{\beta}$ are set to

$$\beta_1 = \ldots = \beta_p = 0.5.$$

3.2.2 Correlated and High-Dimensional Data

The second experiment focuses on high-dimensional data sets, where the predictors are correlated. In each case, p will be at least as large as N.

For the simulation of predictor values, we follow [9] and [13], and construct a covariance matrix where the covariances between variables are based on their indices:

$$\boldsymbol{\Sigma} = \begin{bmatrix} c^{|1-1|} & \dots & c^{|1-p|} \\ \vdots & \ddots & \vdots \\ c^{|p-1|} & \dots & c^{|p-p|} \end{bmatrix},$$

where $0 < c < 1$. As a result, the diagonal elements, that represent the variances, are all equal to 1. The other elements will now represent the correlation between two variables, and also the covariance. Elements that are close to the diagonal will have a higher correlation, while elements where the difference between indices is large will have lower correlation. The mean vector and the effect parameters are defined as in the first experiment (section 3.2.1). The correlation parameter c is chosen to be 0.5 and 0.9.

3.2.3 Magnitude of Predictor-Effects

So far, ratios between N and p, and correlation between variables have been explored. In the following experiment, we vary the magnitude of the effect that a predictor has on the response variable.

The setting used in the last experiment is used here as well ($c = 0.5$), with the difference that $\boldsymbol{\beta}$ is altered in several ways. In the first case, 20% of the variables are randomly picked and set to have $\beta = 0.5$, while the rest of the predictors are set to $\beta = 0.01$. The idea is to investigate how each method handles data where a majority of predictors have no, or close to no effect on the response variable. For the second case, each effect parameter is drawn from a uniform distribution between 0.01 and 3.

3.2.4 Separation of Classes

The final experiment explores the separability of classes. By adjusting $\boldsymbol{\beta}$ and $\boldsymbol{\mu}$, different data sets are generated, where the separation of classes varies from perfectly separated to very overlapping. The two-dimensional case ($p = 2$) is chosen for this experiment, since it is easier to visualize the overlap between classes. The effect parameters and mean values for the two predictors are defined as:

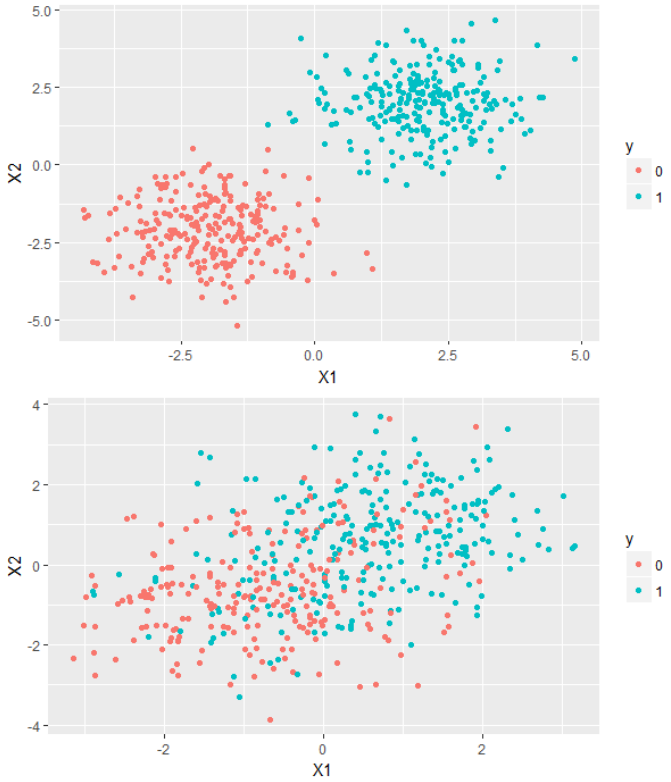

Figure 9: *Generated data sets in R. The first data set is when* $\beta_1 = 10$, $\mu_1 = 2$, *and the classes are perfectly separated. The second data set is when* $\beta_1 = 0.5$, $\mu_1 = 1$, *and the classes have a lot of overlap.*

$$\beta_2 = \beta_1,$$
$$\mu_2 = -\mu_1,$$

where $\beta_1 \in \{0.1, 0.5, 1, 5, 10\}$ and $\mu_1 \in \{1, 2\}$. This way, a larger β_1 and μ_1 results in a more separated data. Figure 9 demonstrates two of the settings in the simulations. For all settings, $N = 500$ and the predictors are independent.

3.3 Model Fitting

For each data set, $2N$ observations are generated. Half of the observations are used to train and validate the models, and the other half is left out for testing of the final models.

3.3.1 Support Vector Machines

The R package *e1071* [11] provides tuning of the cost parameter and kernel parameters for support vector machines. Since the predictor values are simulated from a normal distribution and thereafter run through the logistic regression model, classes can be linearly separated (not perfectly). The linear kernel (the support vector classifier) will therefore be optimal. The cost parameter is obtained by searching through a sequence of values between 0.1 and 20. The parameter that yields the lowest mean misclassification rate through 10-fold cross-validation (section 2.3.3) is used in the final model.

3.3.2 Logistic Regression

Parameter tuning for the logistic regression model can be done with the R package *Glmnet* [6]. By searching through a grid of values for the elastic net parameter α and the penalty parameter λ, the optimal values are once again obtained through 10-fold cross-validation, using the misclassification rate as performance measure. The sequence of values for α are chosen as seven numbers between 0.01 and 0.99. For λ, Glmnet computes its own sequence of 100 values.

4 Results

After retrieving the final models, they are evaluated by predicting the classes of a test data set of size N. The whole procedure is repeated 50 times, and the mean and standard deviation for the misclassification rates (MR) and AUC-values are calculated. Tables 1-6 present the mean difference in percentage points for MR and AUC (standard deviation in parentheses). For MR, the difference in means is the SVM minus LR, and the opposite for AUC. That way, for both performance measures, a positive value corresponds to LR having better prediction accuracy, and a negative value means that LR performs worse. The full results are presented in the appendix (Tables 7-12).

N	p	MR diff (%)	AUC diff (%)
1000	20	0.01 (0.26)	-0.01 (0.26)
1000	200	0.86 (0.28)	0.86 (0.28)
1000	500	0.80 (0.27)	0.80 (0.27)
200	500	-2.35 (0.74)	-2.52 (0.75)
100	20	1.24 (1.01)	1.24 (1.04)
100	50	1.00 (0.92)	0.89 (0.90)

Table 1: *Experiment 3.2.1. The objective is to investigate if there exist any differences in performance, when varying the number of observations and the number of predictors.*

N	p	MR diff (%)	AUC diff (%)
500	5000	-3.63 (0.43)	-3.61 (0.43)
500	1000	-0.76 (0.38)	-0.76 (0.38)
500	500	0.56 (0.39)	0.56 (0.40)

Table 2: *First case of experiment 3.2.2, where* $c = 0.5$*. The experiment consists of high-dimensional data sets with correlated predictor variables.*

N	p	MR diff (%)	AUC diff (%)
500	5000	-2.57 (0.43)	-2.58 (0.43)
500	1000	-0.03 (0.29)	-0.01 (0.29)
500	500	0.15 (0.24)	0.14 (0.23)

Table 3: *Second case of experiment 3.2.2, where* $c = 0.9$*. The experiment consists of high-dimensional data sets with highly correlated predictor variables.*

N	p	MR diff (%)	AUC diff (%)
500	5000	-2.00 (0.41)	-2.12 (0.43)
500	1000	1.03 (0.44)	1.05 (0.44)
500	500	2.12 (0.37)	2.12 (0.37)

Table 4: *First case of experiment 3.2.3, which analyzes data with a lot of noise (80% of the predictors have almost no effect on the response variable).*

N	p	MR diff (%)	AUC diff (%)
500	5000	-2.90 (0.51)	-2.94 (0.51)
500	1000	-0.90 (0.43)	-0.89 (0.42)
500	500	0.51 (0.38)	0.51 (0.38)

Table 5: *Second case of experiment 3.2.3. Each effect parameter is drawn from a uniform distribution between* 0.01 *and* 3*.*

β	μ	MR diff (%)	AUC diff (%)
10	2	0.02 (0.02)	0.02 (0.02)
5	1	0.12 (0.17)	0.12 (0.17)
1	1	-0.32 (0.36)	-0.34 (0.36)
0.5	1	0.03 (0.38)	-0.00 (0.38)
0.1	1	0.26 (0.61)	0.11 (0.55)

Table 6: *Experiment 3.2.4. Two-dimensional analysis ($p = 2$) with different separations between classes.*

5 Discussion

5.1 Predictive Power

Section 4 (Tables 1-6) presents the comparison in prediction accuracy between the methods, in terms of the difference in mean misclassification rate (MR) and AUC. According to the theory in section 2.4, the support vector machine (SVM) and logistic regression (LR) often predict similarly. From the results in this project, the classifiers turned out to have almost identical predictive power.

Overall, the difference in mean MR and AUC is close to 0 percentage points. A few minor differences were observed:

In Table 1, the SVM has a slightly lower mean MR by 2.35 percentage points, when the number of observations is $N = 200$ and the number of predictors is $p = 500$. For most settings, however, the difference in prediction accuracy is less than 1 percentage point, which means that the classifiers behave similarly for uncorrelated data with different ratios between N and p. The second experiment (Table 2 and 3) explored correlated data with a large number of predictors. For $N = 500$, $p = 5000$, the SVM has a 2.57 and 3.63 percentage point lower MR. This coincides with the result in Table 1, suggesting that LR may perform somewhat worse when the number of

predictors is much larger than the number of observations. Table 4 and 5 present the results of the third experiment. Once again, the SVM performs slightly better when $N = 500$, $p = 5000$. In Table 4, where most of the predictors are set to have almost no effect on the response variable, LR has a lower MR for $N = 500$, $p = 500$ (2.12 percentage points). Lastly, in Table 5, where the separation of classes was varied, the difference in prediction acccuracy is negligible.

There are probably numerous other experiments that would be of interest, but since this project is on an undergraduate level, where both time and knowledge is limited, restrictions on the simulation process have been made. For example, the parameters could have been varied a lot more, or isolated and combined in other ways. It would also be interesting to analyze different probability distributions, and testing other methods for simulation of binary data.

Moreover, with the same reasoning as before, and for computational reasons, the fitting process might not be optimized. The value 10 was used in the K-fold cross-validations for all data sets, regardless of their sizes. Also, because of the large running time to fit the models 50 times for each setting, the search grid for parameter tuning was quite coarse.

To conclude, the results in section 4 are very consistent; in terms of misclassification rate and AUC, the predictive power of logistic regression and the support vector machine are close to identical. When the number of predictors is much larger than the number of observations, the SVM performed slightly better. However, the differences are very small, and the fact that the modeling and simulation process was limited needs to be taken into consideration before drawing any definite conclusions. In addition, none of the papers that are referred to in this thesis mentions that SVMs performs better in such cases. In [8], it is stated that the SVM may yield better results when classes are well separated. With that in mind, a possibility is that the simulated data sets with a large number of predictors resulted in classes that are well separated.

There are ways to combine several classifiers and potentially increase prediction accuracy. *Ensemble learning* is the idea of constructing a set of classifiers [3], and predict new observations on the basis of some combination of their individual predictions. If the individual classifiers predict some observations differently, a combination of them could potentially capture the strengths of each classifier and result in a lower error rate than the individual predictions.

In our case, the SVM and LR have very similar performance, and this sug-

gests that they also predict the same observations correctly. If a combination of the two does not capture any differences in prediction, we will not obtain an ensemble classifier that yields a higher prediction accuracy. If they, however, do predict observations differently and still provide such similar results, an ensemble method could improve the predictive power.

5.2 Interpretability

One is often more interested in the interpretation of a model, rather than focusing on classification accuracy. For example, estimating the probability of a medical condition being present and obtaining the possible main causes is more informative than simply classifying if a person has the medical condition or not. That is when the logistic regression model has the advantage of being a probabilistic model; it produces estimated probabilities for the outcome of a response variable, and the parameters can easily be interpreted in terms of the log odds. Model selection methods such as forward selection might be preferable, as one would want to overlook the smaller details provided by a more complex model in order to capture the primary effects and getting the bigger picture, while also saving time (potentially) with a less computationally intensive model selection process.

Classification is more often used when the objective is to automate a process, and each observation needs to be assigned to one of several classes. The field of machine learning contains a number of methods that provide discrete outputs, such as SVMs and *k-nearest neighbours* [7]. But just as the logistic regression model can be modified into a classifier, the outputs of these methods can be transformed into probabilistic outputs. *Platt scaling* [12] does exactly this, by fitting a logistic regression model to the classification scores (the linear combination of the predictor values and the effect parameters) of the classifier. For SVMs, the distance between the decision boundary and an observation can be thought of as a probability; observations that are far away from the decision boundary are assigned to the right class with higher certainty.

There are also variable selection methods for the SVM [15], which could improve the interpretation, and/or improve generalization performance.

5.3 Nonlinearity

The experiments of this thesis focused on data where observations were best classified by a linear decision boundary, but support vector machines

(section 2.2.3) are generalized to handle separation of classes with nonlinear boundaries. With the use of kernels, the SVM's way of transforming the predictors to higher dimensions has great computational advantages.

At first, the idea of using kernels was thought to be unique to the SVM [8], but it has since been shown that the they can be applied to other methods as well, such as logistic regression. Despite this, kernels are most commonly used with SVMs, and the more popular kernel functions are usually available in standard SVM packages (for example, the package used in this thesis [11]). Why is kernel logistic regression not as popular? It is explained in [17] that it is computationally more expensive to apply kernels to LR. Additionally, the estimated parameters for SVMs only depend on the support points, which also results in advantages when coding the algorithms. Another possible reason, is that the SVM is seen as a more modern method (20 years old), whereas logistic regression is a traditional statistical tool.

5.4 Conclusion

In section 2, we focused on the theory of logistic regression and support vector machines. The first approach is probabilistic, and maximizes a binomial log likelihood in order to estimate the parameters of the linear classifier. The support vector machine maximizes the margin, which is the distance between the "closest" observations from two different classes, while allowing some observations to be on the wrong side of the margin.

At first glimpse, these two methods seemed to be completely unalike. Not only do their approaches to the classification problem seem very different, but they also come from different scientific fields. LR is a method that even people outside of statistics are familiar with, but not all statisticians have heard of SVMs, which is a member of a group of machine learning algorithms.

In section 2.4, it was shown that the optimization problem of the penalized LR model and the support vector classifier, both could be reformulated as a more general form, consisting of a loss function and a penalty term. Furthermore, Figure 8 demonstrated the similarity in behaviour between the SVM hinge loss function and the LR loss function. Consequently, the two methods provide similar classifiers in many cases, and therefore have similar predictive power. The results in section 4 confirmed this, with the difference in prediction accuracy being close to 0 percentage points in all experiments. There was, however, a small difference when the number of predictors was much larger than the number of observations, suggesting that

SVMs perform slightly better in such settings (section 5.1 explains why that might not be the case).

To make these two methods even more alike; section 5.2 and 5.3 explained how each classifier can be modified in order to obtain attributes of the other classifier. Which method should we then use? If interpretation is of primary interest, LR would require less work since it naturally produces probabilities, and there are very straight forward methods for selecting the main predictor variables (such as forward selection). For data sets with nonlinear behaviour, the SVM is the easy choice. Kernels are included in standard SVMs packages for programming languages such as Python and R, and applying them to SVMs has some computational advantages over kernel LR.

In light of these arguments, the simple answer is pick the method that best suits the situation. This thesis has, however, solely focused on binary classification. Both methods can also be applied to multiclass classification, but this we have not explored.

6 Appendix

6.1 Results

The full results are presented in Tables 7-12. For logistic regression (LR) and the support vector machine (SVM), the misclassification rate and AUC are displayed (the mean, with standard deviation in the parentheses), along with parameter values that have been varied.

		Misclassification Rate		AUC	
N	p	LR	SVM	LR	SVM
1000	20	0.21 (0.01)	0.21 (0.01)	0.79 (0.01)	0.79 (0.01)
1000	200	0.14 (0.01)	0.14 (0.01)	0.86 (0.01)	0.86 (0.01)
1000	500	0.20 (0.01)	0.21 (0.01)	0.80 (0.01)	0.79 (0.01)
200	500	0.38 (0.04)	0.35 (0.03)	0.62 (0.04)	0.65 (0.03)
100	20	0.27 (0.05)	0.29 (0.05)	0.72 (0.05)	0.71 (0.05)
100	50	0.26 (0.05)	0.27 (0.04)	0.74 (0.05)	0.73 (0.04)

Table 7: *Experiment 3.2.1. The objective is to investigate if there exist any differences in performance, when varying the number of observations and the number of predictors.*

		Misclassification Rate		AUC	
N	p	LR	SVM	LR	SVM
500	5000	0.36 (0.02)	0.33 (0.02)	0.64 (0.02)	0.67 (0.02)
500	1000	0.22 (0.02)	0.21 (0.02)	0.78 (0.02)	0.79 (0.02)
500	500	0.17 (0.02)	0.17 (0.02)	0.83 (0.02)	0.83 (0.02)

Table 8: *First case of experiment 3.2.2, where* $c = 0.5$*. The experiment consists of high-dimensional data sets with correlated predictor variables.*

		Misclassification Rate		AUC	
N	p	LR	SVM	LR	SVM
500	5000	0.20 (0.02)	0.18 (0.02)	0.80 (0.02)	0.82 (0.02)
500	1000	0.09 (0.01)	0.09 (0.01)	0.91 (0.01)	0.91 (0.01)
500	500	0.06 (0.01)	0.06 (0.01)	0.94 (0.01)	0.94 (0.01)

Table 9: *Second case of experiment 3.2.2, where* $c = 0.9$*. The experiment consists of high-dimensional data sets with highly correlated predictor variables.*

		Misclassification Rate		AUC	
N	p	LR	SVM	LR	SVM
500	5000	0.39 (0.02)	0.37 (0.02)	0.61 (0.02)	0.63 (0.02)
500	1000	0.27 (0.02)	0.28 (0.02)	0.73 (0.02)	0.72 (0.02)
500	500	0.22 (0.02)	0.24 (0.02)	0.78 (0.02)	0.76 (0.02)

Table 10: *First case of experiment 3.2.3, which analyzes data with a lot of noise (80% of the predictors have almost no effect on the response variable).*

		Misclassification Rate		AUC	
N	p	LR	SVM	LR	SVM
500	5000	0.36 (0.03)	0.33 (0.02)	0.64 (0.03)	0.67 (0.02)
500	1000	0.23 (0.02)	0.22 (0.02)	0.77 (0.02)	0.78 (0.02)
500	500	0.17 (0.02)	0.18 (0.02)	0.83 (0.02)	0.82 (0.02)

Table 11: *Second case of experiment 3.2.3. Each effect parameter is drawn from a uniform distribution between* 0.01 *and* 3*.*

		Misclassification Rate		AUC	
β	μ	LR	SVM	LR	SVM
10	2	0.00 (0.00)	0.00 (0.00)	1.00 (0.00)	1.00 (0.00)
5	1	0.03 (0.01)	0.03 (0.01)	0.97 (0.01)	0.97 (0.01)
1	1	0.16 (0.02)	0.16 (0.02)	0.84 (0.02)	0.84 (0.02)
0.50	1	0.28 (0.02)	0.28 (0.02)	0.72 (0.02)	0.72 (0.02)
0.10	1	0.45 (0.03)	0.46 (0.03)	0.55 (0.03)	0.55 (0.03)

Table 12: *Experiment 3.2.4. Two-dimensional analysis ($p = 2$) with different separations between classes.*

References

[1] Agresti, A. (2002). *Categorical Data Analysis*, second edition. John Wiley & Sons, Inc., Hoboken, New Jersey.

[2] Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3), 273-297. Kluwer Academic Publishers, Boston.

[3] Dietterich, G. T. (2000). Ensemble Methods in Machine Learning. *Lecture Notes in Computer Science*, 1857, Multiple Classifier Systems, 1-15. Springer, New York.

[4] Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8), 861-874. Elsevier.

[5] Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1-22. American Statistical Association.

[6] Friedman, J., Hastie, T., Tibshirani, R. and Simon, N. (2018). *Lasso and Elastic-Net Regularized Generalized Linear Models*. https://cran.r-project.org/web/packages/glmnet/glmnet.pdf

[7] Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, second edition. Springer, New York.

[8] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017). *An Introduction to Statistical Learning*. Springer, New York.

[9] Krona, E. (2017). *A Simulation Study of Model Fitting to High Dimensional Data using Penalized Logistic Regression*. Stockholms universitet. https://www.math.su.se/publikationer/uppsatsarkiv/

[10] Le Cessie, S. and Van Houwelingen, J. C. (1992). Ridge Estimators in Logistic Regression. *Journal of the Royal Statistical Society*, series C, 41(1), 191-201. Wiley, Hoboken, New Jersey.

[11] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. and Leisch, F. (2017). *Misc Functions of the Department of Statistics, Probability Theory Group, TU Wien*. https://cran.r-project.org/web/packages/e1071/e1071.pdf

[12] Platt, C. J. (2000). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, 61-74. MIT Press, Cambridge, Massachusetts.

[13] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society*, series B, 58(1), 267-288. Wiley, Hoboken, New Jersey.

[14] Vapnyarskii, I. B. (2001). Lagrange multipliers. *Encyclopedia of Mathematics*, Kluwer Academic Publishers, Boston.

[15] Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. and Vapnik, V. (2000). Feature Selection for SVMs. *Advances in Neural Information Processing Systems 13*, 647-653. MIT Press, Cambridge, Massachusetts.

[16] Wolfe, P. (1961). A Duality Theorem For Non-linear Programming. *Quarterly of Applied Mathematics*, 19(3), 239-244. Brown University, Providence, Rhode Island.

[17] Zhu, J., Hastie, T. (2002). Support Vector Machines, Kernel Logistic Regression, and Boosting. *Lecture Notes in Computer Science*, 2364, Multiple Classifier Systems, 16-26. Springer, New York.