UPTEC X09 033
Examensarbete 30 hp
November 2009

Comparing classification accuracy between different classification algorithms

Jonas Gisterå

UPPSALA UNIVERSITET

Bioinformatics Engineering Program

Uppsala University School of Engineering

UPTEC X 09 033	**Date of issue 2009-10**

Author

Jonas Gisterå

Title (English)

Comparing classification accuracy between different classification algorithms

Title (Swedish)

Abstract

In this study the performance of four classification algorithms; *k*-Nearest Neighbors, Linear Discriminant Analysis, Support Vector Machine and Random Forest was analysed when applied on SELDI-TOF-MS data. No conclusions could be made about any algorithm performing better than the rest of the algorithms. The classification results seem to be more dependent of the datasets than the classification algorithm used.

Keywords

Classification algorithms, data mining, mass spectrometry, SELDI-TOF-MS, supervised classification, feature selection.

Supervisors

Michal Lysek
MedicWave AB

Scientific reviewer

Tomas Olofsson
Department of Engineering Sciences, Uppsala University

Project name	Sponsors
Language **English**	Security
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **41**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Comparing classification accuracy between different classification algorithms

Jonas Gisterå

Sammanfattning

Masspektrometri är en teknik som kan användas till att bestämma massan för olika molekyler i en lösning. Ett exempel på masspektrometri är SELDI-TOF-MS, en metod där molekyler accelereras genom ett rör tills de når fram till en detektor. Information om hur snabbt molekylerna tar sig till detektorn kan användas till att uppskatta molekylernas massa. Resultatet presenteras i form av ett diagram med olika toppar. Idealt motsvarar varje topp en molekyl.

Förbättringar inom masspektrometri har på senare tid gjort det möjligt att analysera stora komplexa biomolekyler och ett av de mest spännande forskningsområdena är att studera proteiner. Målet med denna forskning är att hitta skillnader mellan spektra från friska och sjuka patienter vilket skulle kunna vara användbart vid diagnos av olika sjukdomar. För att kunna skilja på dessa två typer av spektra krävs algoritmer som kan bestämma om ett nytt spektrum kommer från en sjuk eller en frisk patient. Denna klassificering baseras på kunskap om hur tidigare spektra sett ut.

Förhoppningen är att man i framtiden ska kunna ta ett blodprov från en människa, köra det i en masspektrometer och slutligen använda en mjukvara som medel att ställa diagnos av en viss sjukdom. På detta sätt skulle exempelvis cancer kunna upptäckas innan det bildats en tumör och behandling av sjukdomen kunna påbörjas på ett tidigt stadium, vilket skulle öka patienternas chans att tillfriskna.

Syftet med detta examensarbete är att jämföra hur bra olika algoritmer är på att klassificera spektra från personer som är antingen friska eller sjuka. Totalt utvärderades klassificeringsalgoritmerna på data från fyra olika masspektrometristudier; en studie med patienter med diagnosen multipel skleros, en studie avseende äggstockscancer och två studier med bröstcancerpatienter. Förbehandlingssteg och användning av olika metoder för att kunna analysera stora mängder data utfördes på ett sådant sätt att jämförelse av algoritmerna skulle vara möjlig. Resultaten i den här undersökningen visade ingen signifikant skillnad mellan klassificeringsmetoderna. Däremot verkar typen av data, det vill säga vilken sjukdom patienterna hade, vara av större betydelse än vilken algoritm som används.

Examensarbete 30 hp
Civilingenjörsprogrammet Bioinformatik

Uppsala universitet oktober 2009

Table of contents

Abbreviations

BC1	Breast cancer dataset 1
BC2	Breast cancer dataset 2
C	Cost parameter for support vector machines
CV	Cross validation
Da	Daltons
FS	Feature selection
k	Number of neighbors in k-nearest-neighbor
k-NN	*k*-nearest-neighbor
LDA	Linear discriminant analysis
m/z	Mass over charge
mtry	Number of created variables in random forest
ntree	Number of trees in random forest
MS	Mass spectrometry
MSC	Multiple Sclerosis dataset
OC	Ovarian cancer dataset
p	Number of features selected in feature selection
RF	Random forest
RMS	Root Mean Square
SELDI	Surface enhanced laser desorption/ionization
SVM	Support vector machines
TOF	Time of flight
γ	Gamma parameter in support vector machines

1. Introduction

Research within the field called proteomics which Berrar, Granzow and Dubitzky define as 'the study of proteins, protein complexes, their localization, their interactions, and posttranslational modifications' has changed a lot recently. In contrast to traditional research where one protein was studied at a time, nowadays data mining methods are used to analyse the activity of thousands of proteins simultaneously. This development has been made possible due to faster computers and improvements of techniques within mass spectrometry. Surface enhanced laser desorption/ionization time of flight mass spectrometry (SELDI-TOF-MS) is one of those newer techniques which can be used for studying proteins in human samples (Glish & Vachet 2003). The SELDI-TOF-MS technique was used in almost 100 articles published in 2003 and it is still widely used.

Detection of diseases using classification of SELDI-TOF-MS data is a very interesting field of research in proteomics (Radlak & Klempous 2007). The main idea of classification in this area is to find patterns in protein content of different samples, which can be used for discriminating samples from healthy patients from those who have a certain disease. Examples of such kinds of patterns could be that the healthy patients lack some proteins that are present in the diseased patients or that the amount of certain proteins is different for the two types of patients. The protein content of different samples is analysed by studying mass spectra obtained from mass spectrometry. Each mass spectrum contains peaks of various heights which are situated along the horizontal axis. These peaks correspond to molecules in the analysed sample. The heights of the peaks give information about how many of a particular molecule there is in the sample whereas their horizontal location indicates how large the molecules are. The methods used for extracting useful information from mass spectra, containing up to hundreds of thousands measured intensities each, are called classification algorithms. These algorithms build models based on a set of spectra with known disease status (either healthy or diseased) which can be used for classifying the status of other spectra. In the future such models will hopefully be accurate enough to use for diagnosis purposes. Then diseases could be treated at an earlier state and in case of cancer perhaps even before a tumour has started to grow. It is important that the models created by the classification algorithms are very accurate. Otherwise there is a risk that healthy people are misclassified and treated with unnecessary medication which is costly and possibly have adverse psychological effect for the people involved. Of course there is also a risk that patients are misclassified as healthy when they are ill which potentially would be even more serious since their health status could worsen.

In this study the predictive performance of four different classification methods, *k*-Nearest Neighbors (*k*-NN), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM) and Random Forest (RF), are compared. The classification is performed on mass spectrometry datasets from ovarian cancer, multiple sclerosis and breast cancer studies. The dimensionality of the data requires the data to be analysed by using strategies able to cope with the large amount of features per spectrum and the small total amount of spectra. A statistically stringent framework is used to compare the classification algorithms in order to exclude that the difference in performance could be due to chance.

The organisation of this paper is as follows. The background of both mass spectrometry and the data mining techniques used for extracting useful information from proteomic data is presented in section 2. In section 3 the datasets and methods used in this particular study are described; why they were chosen and how they were implemented. The results obtained using

the methods and design in section 3 is presented in chronological order in section 4. Finally, the results of this study and possible future improvements are discussed in section 5.

1.1 Aim

The aim of this project is to analyse the performance of classification algorithms *k*-Nearest Neighbors, Linear Discriminant Analysis, Support Vector Machine and Random Forest used on four different SELDI-TOF-MS datasets received by MedicWave.

To be able to fulfil the aim of this study the following question is to be answered:

- does anyone of the classification algorithms perform significantly better than the other algorithms when applied on MedicWave's four datasets?

2. Background

This section presents the general concepts used in proteomic classification studies. First the technique which has made this type of proteomic studies possible, high-throughput mass spectrometry is described. Thereafter the data mining part is presented, which is the use of computers to find interesting patterns in data. Here the classification problem is stated and also which techniques that are commonly used to solve the problem without being algorithm specific.

2.1 Mass spectrometry

Mass spectrometry is a powerful technique which can be used in a fast and sensitive way to study almost any molecule (Glish & Vachet 2003). This fact makes the technique very useful in a field like proteomics where it identifies biologically important molecules. Different techniques of mass spectrometry have some analytical steps in common. They all include an ionization step of the studied molecules, a mass analyser and a detector. Despite its name, a mass spectrometer returns the proteins mass over charge (m/z) values and not their masses.

In mass spectrometry, the ionization step which is the conversion of chemical substances into charged gas molecules is a crucial component of the analysis. In the past, only compounds with low molecular weight could be analysed with mass spectrometry since the ionization had to be performed in two succeeding steps; first vaporization of the molecules and then ionization. Furthermore, it was only possible to study thermally stable molecules. The development of ionization makes it possible to analyse large biomolecules such as proteins. One example of a mass spectrometry technique with improved ionization steps is the SELDI-TOF-MS which is described in section 2.2.1.

To be able to measure m/z values with a detector, a mass analyser needs to be used on the ionized molecules. Time-of-flight (TOF) is an analyser accelerating the molecules through a tube by applying an electric field. Equally charged molecules achieve the same amount of kinetic energy. This energy transports molecules with small m/z values faster to the detector, than those with larger m/z values. Since the length of the tube is known and the time it takes for an ion to reach the detector can be measured, the m/z value can be determined by using the fact that it is inversely proportional to the velocity. An important property of TOF is the ability to handle large molecules and measure values of m/z up to several hundreds of thousands. By measuring the amount of molecules for small time steps the output of the analysis, a mass spectrum, can be created with m/z values on the horizontal axis and the corresponding intensity on the vertical axis.

2.1.1 SELDI-TOF-MS

Surface enhanced laser desorption/ionization time-of-flight mass spectrometry is useful when studying samples with proteins (Li et al. 2005). Samples can be taken from different types of bodily fluids such as cerebral spinal fluid, urine and blood. One of the main reasons of studying these samples is to find differences in protein content for two different types of patients; healthy and diseased. Various types of cancer and other serious diseases such as multiple sclerosis have been studied by analysing samples with SELDI-TOF-MS.

Proteins analysed with SELDI-TOF-MS technology binds to the active surface of small arrays, whereas the rest of the molecules are washed away. The proteins are then dried and crystallized together with a solution containing energy absorbing molecules. To convert the proteins into gaseous charged ions a laser beam is used on the array. The laser also accelerates the ions through a tube towards a detector that measures the TOF of the ions. The m/z values can then be determined. Molecules with known masses are used for calibration in order to find the coefficients of the equation that describes the relationship between measured TOF and m/z values.

2.2 Data mining in proteomics

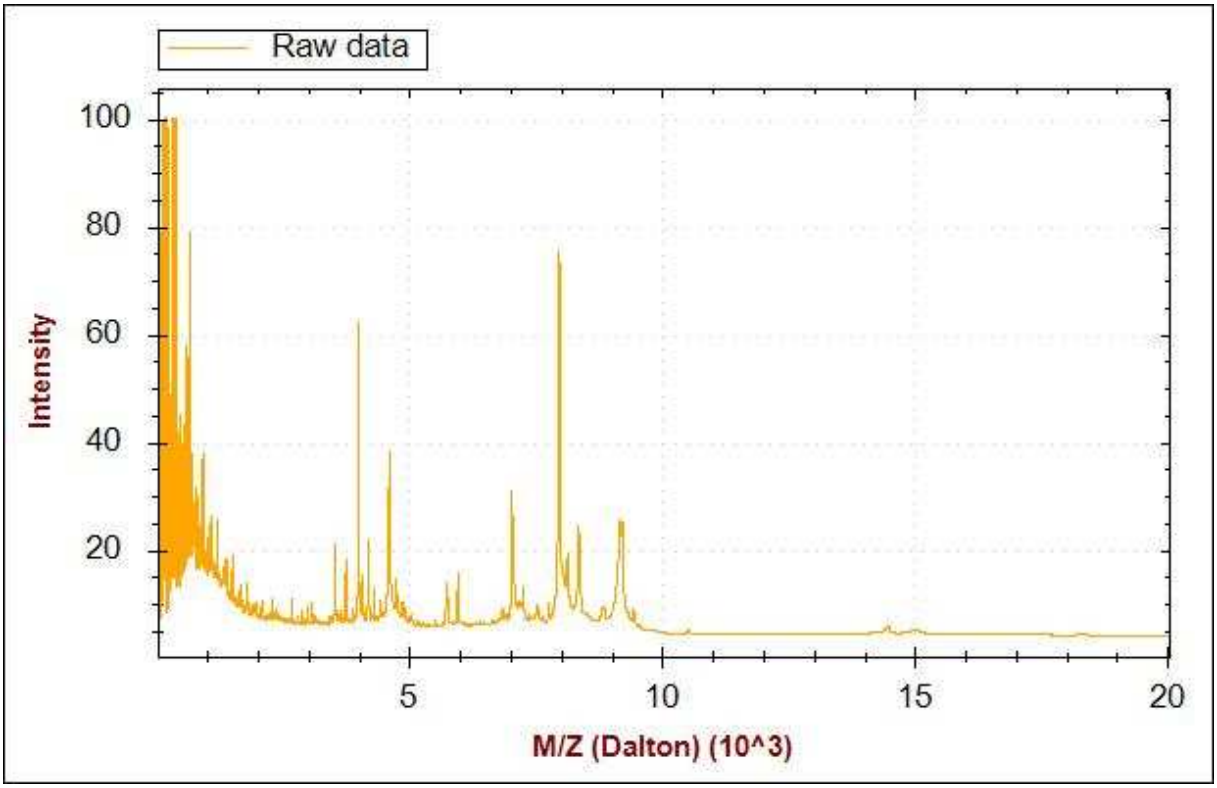
The raw data used in proteomic studies is different from the data used in classical data mining applications such as finance and retail (Berrar, Granzow & Dubitzky 2007). The main difference is that in proteomics there is a large amount of features which is measured for a small number of patients whereas in the classical the opposite situation is present, with a small number of features and a large number of cases.

2.2.1 Raw data

Datasets obtained from proteomic research usually consist of tens or hundreds of spectra, one spectrum for each patient in the study. Patients are either healthy or carriers of the disease of interest in the study. Both types are necessary in order to find proteomic patterns which successfully can discriminate the two types.

The raw data obtained from mass spectrometry analysis of a chemical solution can be visualized in a spectrum as shown below in figure 1. In a SELDI-TOF spectrum there are often 10 000 to 100 000 m/z values on the horizontal axis (Coombes, Baggerly & Morris 2007). In most cases, the mass spectrometer ionizes proteins into single charged ions and therefore the m/z values can be expressed in terms of masses instead of m/z values (Berrar, Granzow & Dubitzky 2007). The values referring to these masses form peaks along the horizontal axis at every point a mass occur at high intensity. It is then expected that these peaks correspond to proteins in the analysed solution. In mass spectra the mass is expressed in terms of Daltons (Da), which is the mass for a single molecule.

Fig. 1. Example of a SELDI-TOF spectrum from a patient with ovarian cancer.



Analysis of proteomic data is a "large *p*, small *n*" problem where *p* is the number of features and *n* is the number of cases. Translated into terms used for explaining proteomic datasets this means there are a large number of m/z values in every spectrum, but in total only a small number of spectra. The set of intensity values at one feature with values from all *n* spectra is called a protein expression profile.

2.2.1 Data pre-processing

Pre-processing of SELDI-TOF-MS raw data is used in order to prepare the data for being analysed in a classification study (Radlak & Klempous 2007). Well performed pre-processing steps are important in order to adequately analyse spectra and it can increase the classification performance. The techniques chosen for pre-processing varies between different studies and also the order in which the pre-processing steps are performed can differ (Coombes, Baggerly & Morris 2007). However all methods have one important property in common, the fact that they do not use information about the class of different spectra. One of the main problems when pre-processing mass spectra is to handle the noise, which is present in both the vertical and horizontal direction. If peaks and not the raw m/z values are used for classification some kind of algorithm detecting peaks is also needed. If there are missing values these can be handled using a few different approaches (Berrar, Granzow & Dubitzky 2007). One way is to simply ignore missing values which forces the data mining methods to handle the missing data. A large drawback of this approach is that it results in fewer possible classification methods to use. Another approach is to impute a new value, for instance 0 or the mean value of the feature where missing values occur. More sophisticated methods used in bioinformatics determine imputes by estimating the missing values using information about intensity values of other features. Examples of this are collateral missing value estimation (Sehgal, Gondal, & Dooley 2005) and weighted nearest-neighbour methods (Troyanskaya et al. 2001; Johansson & Häkkinen 2006). There is also an approach to handle missing values by deleting the cases

they appear in and this should only be used if the removed cases form a small proportion of the dataset.

2.2.2 Supervised classification

The aim of classification in proteomics is to predict which class different spectra with unknown class label belong to (Larrañaga et al. 2006). In binary classification of SELDI-TOF-MS data this means classifying spectra into one of the two groups with class labels called "Normal" and "Disease". This class prediction is performed using models which are built by classification algorithms. A large amount of classification algorithms has been applied on proteomic datasets and new ones are developed continuously in order to improve predictive performance. These algorithms base their decisions on spectra with known class labels. In the future it may be possible that these types of models could be based on stored patient data and be used in diagnosing of new patients.

In order to estimate the performance of classification algorithms only spectra with known class label can be included. This is because the estimation is based on the comparison of the predicted classes and the true classes. The ratio of how often spectra are successfully classified into one of the two classes is called accuracy and the opposite, how often spectra are misclassified is called the error rate. To evaluate the predictive performance of classification algorithms the data needs to be divided into learning and test sets. The learning set is used to build the models of the classification algorithms whereas the test set is used for testing the models.

Due to the large amount of features a process called feature selection can be used before data is passed to the classification algorithms. The aim of using feature selection is to determine which features are best at discriminating samples of different classes and only use those when training the classifiers. There are multiple objectives using feature selection on high dimensional proteomic data. Important ones are the decreased computational burden due to data reduction and also the reduced risk of overfitting (Datta & DePadilla 2006). Additionally including too many features decreases the predictive performances of the classifiers since a lot of irrelevant and noisy features worsen the classification (Liu, Li & Wong 2002).Techniques used for feature selection in proteomics are often divided into filters, wrappers and embedded methods (Hauskrecht et al. 2007; Saeys, Inza & Larrañaga 2007). The general idea of filters is to examine the features one at a time with a scoring function which tells if samples from different classes are differently distributed. A disadvantage with filters is that they do not take into consideration dependencies between features. This could imply that there exists an optimal combination of features which could not be discovered. The second type of method is called wrappers. Wrappers include a heuristic search over possible feature subsets and each one of these subsets is evaluated by training a classifier and validating on validation data. In contrast to filters, wrappers are able to find dependencies in data which may improve classification performance. On the other hand they are computationally intensive, dependent of the classifier and there is a risk of overfitting. The third variant is the embedded methods which integrate the feature selection into to the model building process. With these embedded methods it is hard to find a small subset of features but an advantage in comparison to wrappers is that they require less computation.

In order to estimate the predictive accuracy of a classification algorithm when the number of cases is small it is ideal to gather more data from new experiments and use it for testing the model (Simon 2007). This is seldom possible due to time and cost issues and therefore methods taking advantage of the available dataset have been developed, such techniques are

called resampling methods. These methods can be used for assessment of the predictive performance of classification algorithms. Different resampling methods use different strategies to partitioning the data, but they all construct multiple divisions into learning and test data. The error rate obtained on different test sets can then be used for determining the observed error rate. The observed error rate is used to estimate the true error rate, which is the expected rate at which completely new spectra would be misclassified. Using resampling methods can cause a difference between the estimated error rate and the true error rate, which can yield a result that is positively or negatively biased.

3. Materials and methods

This section first presents the datasets analysed, both the diseases tested and the dimensionality of the data. Then the methods used for preparing the data for the classification task are described. Also the design used for evaluating the classification algorithms is presented which is the main focus in this study. The chosen resampling strategy, feature selection process and the different classification algorithms with their parameters are also described.

3.1 Datasets

This study includes four SELDI-TOF-MS datasets: Breast Cancer 1 (BC1), Breast Cancer 2 (BC2), Ovarian Cancer (OC) and Multiple Sclerosis (MSC). Table 1 shows that the size of the four datasets varies a lot, both in terms of number of spectra and number of features. OC and MSC are the two largest datasets, with more than four times the number of spectra compared to the BC1 and BC2 datasets. The percentage of spectra from diseased patients lies between 32,7 and 64,3 %. The number of m/z values in the spectra are in the interval of 15 000 – 75 000 values and every spectrum has one target feature with information regarding the patient's status. The feature is called class label and assigns either "Normal" or "Disease".

Table 1. A table of the original datasets based on raw spectra. For each dataset the number of spectra from healthy and diseased people respectively is displayed and also the number of m/z values in each spectrum. The target feature determines if spectra come from healthy or diseased patients.

	Dataset			
	Breast Cancer 1	Breast Cancer 2	Ovarian Cancer	Multiple Sclerosis
Spectrum				
Normal	30	24	90	189
Disease	29	40	162	92
Total	59	64	252	281
Feature				
m/z	48480	52062	15154	74002
Target	1	1	1	1

3.2 Pre-processing of mass spectra

The Bioconductor project provides open source software for applications within bioinformatics (Gentleman et al. 2003). It is based on the R programming language and has several packages performing pre-processing of mass spectra. Some of those packages were tested in the beginning of this study, but later on Medicwave's Proteinmarker Detection Software™ was chosen to perform the pre-processing steps explained in sections 3.2.1-3.2.4.

For the evaluation of missing value estimation described in section 3.2.5 scripts were developed in R. The main ideas of the methods used for pre-processing are described but all details of the algorithms are not presented. All datasets analysed in this study were prepared for data analysis using the same pre-processing steps.

3.2.1 Valid m/z range

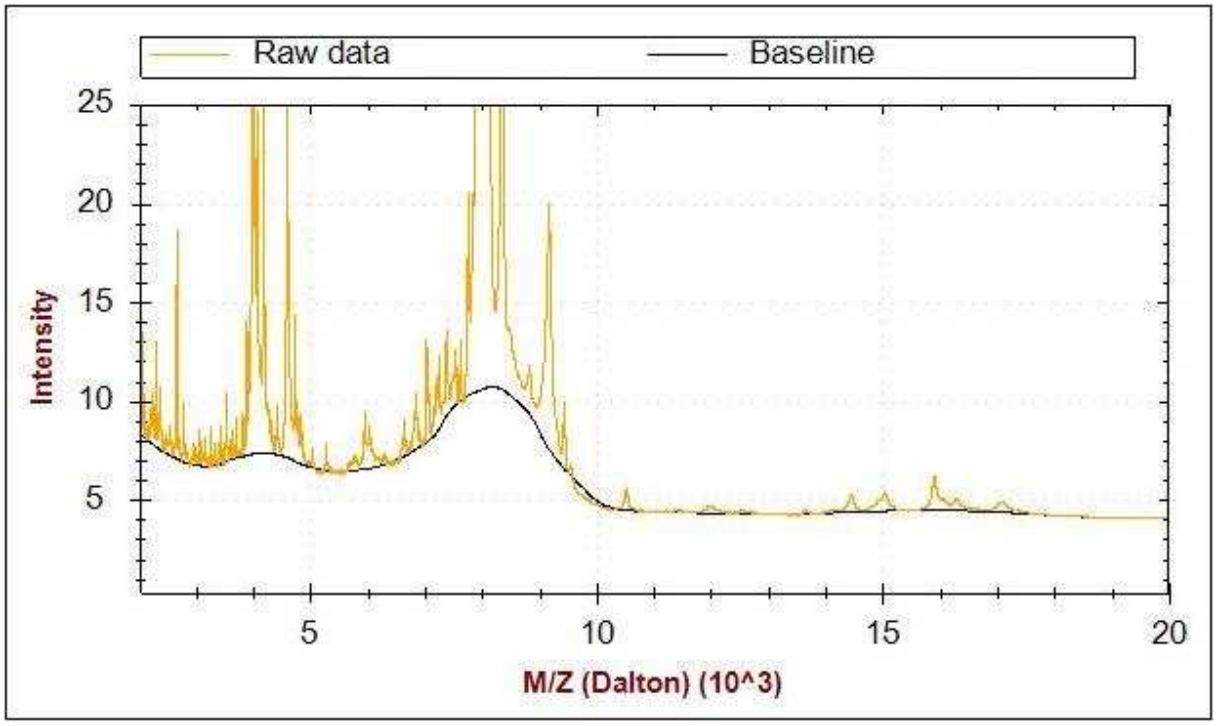
Due to chemical noise affecting the SELDI-TOF-MS analysis, peaks at low m/z values in spectra are not reliable (Li et al. 2005). This noisy behaviour can be observed in the spectrum in figure 1 in section 2.2.1. To be able to compare spectra, a valid m/z range needs to be mutual for all spectra. In this study all datasets where reduced to include m/z values within the range of 2 000 – 20 000 Daltons, as in the study by Jeffries in 2005.

3.2.2 Baseline subtraction

Raw SELDI-TOF-MS spectra contain a non-constant noise level along the entire range of m/z values which causes the peaks to be located on a baseline above the horizontal axis (Li et al. 2005). Baseline subtraction is the process used to handle the noise level by correcting the vertical shift of intensity values. Each spectrum in a dataset needs this correction, which is performed in two subsequent steps. The baseline of the analysed spectrum is first estimated and then subtracted from the spectrum. The resulting spectrum is a baseline adjusted spectrum with peaks approximately starting from the horizontal axis with intensity equal to zero.

In figure 2 an example of a raw spectrum from the OC dataset and the corresponding estimated baseline can be seen. The figure shows the non-constant shape of the baseline and the fact that the noise level is larger for small m/z values.

Fig. 2. An example of a raw spectrum and its estimated baseline.



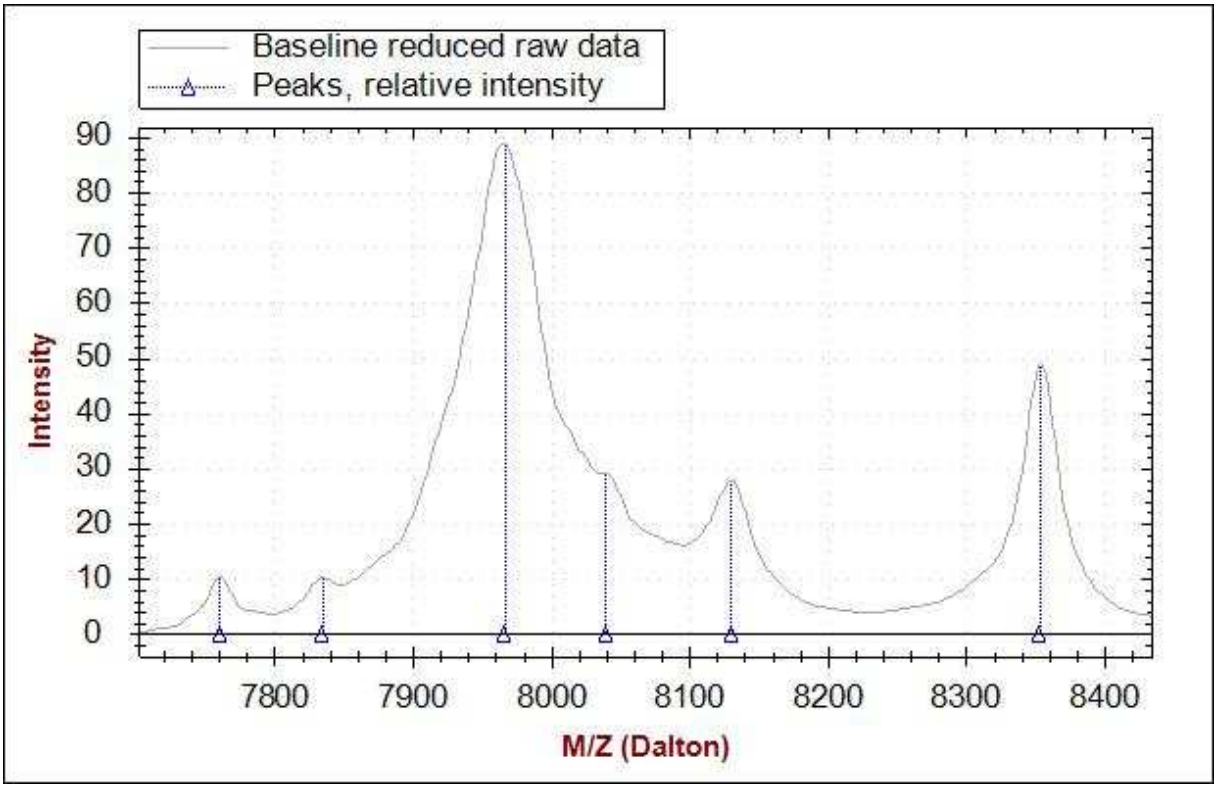
3.2.3 Peak detection

After obtaining a set of baseline adjusted spectra, the next step of pre-processing is to detect their peaks. This process is performed in order to determine the m/z values which could be

represented by proteins in the sample (Li et al. 2005). For each spectrum a peak detection algorithm searches through the spectrum after local maxima and the corresponding m/z values. Hopefully the intensities of the detected peaks are different for spectra coming from healthy and ill patients or perhaps there are peaks only found in one type of patients.

Figure 3 shows the same spectrum as shown in figure 2 but with removed baseline and zoomed in at the region with m/z values between 7700 and 8440 Da. There are settings in the Proteinmarker Detection Software™ which affects how many possible peaks that will be detected. Feature selection is performed on the data later in this study and therefore detection of a large number of peaks is acceptable. The noisy peaks will not be selected later and the risk of not including real peaks is reduced.

Fig. 3. A spectrum of detected peaks in the span of m/z values between 7700 and 8440 Da. The figure shows six possible peaks and their locations on the horizontal axis are marked with triangles.



3.2.4 Binning

The detection of peaks in each spectrum is not enough to be able to compare different spectra within the same dataset because there is a small variation in the m/z values. For example, assume one peak in spectrum A and one peak in spectrum B corresponds to the same protein. To be able to analyse correctly with data mining methods it is important that the peak in A and the peak in B are represented by the same feature. However this is not always possible due to small errors which can cause the location on the horizontal axis to vary with 0.3% around the m/z-value (Li et al. 2005). Binning is the process used in this study to handle the horizontal drift. In binning, all spectra are divided into small intervals of m/z values called bins. These are exactly the same for all spectra in the dataset and the peaks of each spectrum are then placed into the bins. Instead of representing the data with different m/z values the different bins are now the new features.

3.2.5 Handling of missing values

After binning, each spectrum only contains intensity values corresponding to the m/z values where the spectrum has detected peaks, whereas the rest of the data is missing. For a considerable amount of features, some spectra lack intensity values.

In this study the BC1 dataset was used to evaluate three different methods handling missing values. Two simple techniques where analysed, one called *Zero impute* where every missing value is replaced by zero and one called *Mean impute* where the missing values for a feature are replaced by the mean value of the other spectra's values for that feature. The third technique, called *KNNimpute* (Troyanskaya et al. 2001), calculates a value to impute based on the k most similar protein expression profiles which have non-missing values for the same feature as the missing value. The missing value is then replaced by the mean intensity of the k neighbors.

To analyse which one of *Zero impute*, *Mean impute* and *KNNimpute* is best at replacing missing values, 50 randomly chosen values in the dataset where removed. Parameter k in *KNNimpute* was set to 5 as in the study by Dudoit, Fridlyand & Speed (2002). Substitutes for the removed values were calculated with the three different methods mentioned above and for each one of them the root mean square (RMS) error was determined. The RMS error E_{RMS} for the imputed values $y = \{y_1, y_2, ..., y_n\}$ with original values $x = \{x_1, x_2, ..., x_n\}$ is given by

$$E_{RMS} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(x_j - y_j\right)^2} ,$$

where n denotes the number of removed values. The procedure of imputing values was repeated 100 times to get more reliable results. The results of the comparison of different missing value estimation methods are presented in section 4.1.

3.2.6 Pre-processed datasets

The pre-processing steps described in the previous sections were performed in order to improve the quality of the data and to reduce the number of features (see table 2). The amount of data is reduced with 82.4 – 93.2 % because of the choice of just including m/z values between 2 000 and 20 000 Da, and transformation of the peaks into bins instead of m/z values and transformation of the peaks into bins instead of m/z values. The pre-processed datasets will be used as input data to the comparison of classification algorithms explained in section 3.6.

Table 2. The dimensionality of the raw and pre-processed datasets.

	Dataset			
	Breast Cancer 1	Breast Cancer 2	Ovarian Cancer	Multiple Sclerosis
Data				
Raw	59 x 48480	64 x 52062	252 x 15154	281 x 74002
Pre-processed	59 x 5622	64 x 9392	252 x 1026	281 x 7465

3.3 Classification algorithms

In this study four classification algorithms were compared in means of their respective accuracy. The algorithms are *k*-Nearest-Neighbour, Linear Discriminant Analysis, Support Vector Machines and Random Forest and they have all shown good discriminating power in previous studies. Methods which differ a lot from each other and have different number of model parameters were chosen since it is hard to predict which algorithm will perform better in a classification task (Larrañaga et al. 2006). In microarray studies (Dudoit, Fridlyand & Speed 2002) simple methods based on discriminant analysis and nearest neighbors have shown better performance, when applied on datasets with few cases and a large number of features, compared to more complex methods. The algorithms used in this study are explained in further detail in the following sections (3.3.1-3.3.2).

3.3.1 *k*-Nearest Neighbors

In proteomic studies, the *k*-Nearest Neighbors (*k*-NN) is a commonly used classification algorithm. Let L denote a learning set and T the corresponding test set. To classify a test case t_i from T with unknown class label the *k*-NN algorithm calculates the distance between every learning case and the test case t_i. A common choice of distance measurement is the Euclidean distance which was used in a genomic study by Dudoit, Fridlyand and Speed (2002). When classifying mass spectra the Euclidean distance d between a test spectrum $t_i = p_1, p_2,..., p_n$ and a learning spectrum $l_i = q_1, q_2,..., q_n$ where p_i and q_i is the intensity values at peak i, is calculated as follow:

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \quad .$$

Each test case in T is classified based on the k number of learning spectra which are situated with the shortest distance to the test case. These learning spectra are called the k nearest neighbors and the majority of their class labels are used for determining the class label of the test case. A disadvantage using the simplest form of *k*-NN on large datasets is that it slows down when it has to calculate all distances between each test case and all the cases in the learning set (Larrañaga et al. 2006). However, development of new variants of the *k*-NN algorithm has made it possible to reduce the number of measured distances required for classification (Gil-Pita et al. 2007).

The *k*-NN algorithm used in this study comes from an R package called *class* and it uses the Euclidean distance to determine which cases are the nearest neighbors. The initial parameter values of the number of nearest neighbors used as input to the algorithm in this study were the following:

$k = 1, 4, 7,\ldots, 51$

3.3.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a linear method of classification used for two-class problems (Fung et al. 2005). The main idea of LDA classification is to maximise the separation of objects from two different classes (Lilien, Farid & Donald 2003). For a given learning set L this can be done by finding a hyperplane which maximise the ratio of between-class sums of squares B to the within-class cross-products W. The ratio can be written as $a'Ba / a'Wa$ where a is a vector with coefficients. In binary classification the hyperplane separating the two classes is a linear function in the one-dimensional discriminant-space. The function is called a linear discriminant function and it can be determined by finding the

eigenvalues and eigenvectors of $W^{-1}B$. Test cases can finally be classified into one of the two classes by projecting the test cases on the discriminant function.

The LDA implementation used in this study is a part of the R package called *MASS*. In contrast to the three other chosen classification algorithms LDA has no parameters that need to be tuned.

3.3.3 Support Vector Machine

Support Vector Machine (SVM) is a classification algorithm which has been used in several proteomic studies (Fung et al. 2005). The algorithm first map the cases used for learning into a high dimensional space. In a binary classification task, as the one in this study, the idea of using the SVM algorithm is to find a hyperplane which best separates cases from two classes. The hyperplane chosen is the one resulting in the largest margin between the nearest cases of the two classes. Two advantages when using SVM are that it is fast and it can be used on non-linear classification problems (Johansson & Ringnér 2007).

Hsu, Chang and Lin propose a procedure to get satisfactory SVM classification results on datasets with few features (2008). Since the feature selection in this study reduces the number of features significantly before passing the data to the SVM algorithm the author's suggestion works in this study as well. Therefore the SVM in this study use a radial basis function kernel which have two parameters; a cost parameter C and a kernel parameter γ. These two parameters need to be tuned. A search over all possible combinations of γ and C is very time-consuming and the authors suggest to first perform a search over $C = 2^{-5}, 2^{-3},..., 2^{15}$ and $\gamma = 2^{-15}, 2^{-13},..., 2^{3}$. By analysing a contour plot of the result a good region can be determined which can be partitioned in smaller steps in a second parameter search.

The SVM algorithm from the R package *e1071* was used in this study and it is a C/C++ implementation by Chih-Chung Chang and Chih-Jen Lin. The initial values used for tuning of the SVM parameters were the following:

$C = 2^{-5}, 2^{-2}, 2^{1},..., 2^{16}$
$\gamma = 2^{-15}, 2^{-12}, 2^{-9},..., 2^{3}$

3.3.4 Random Forest

Random forests are classifiers based on voting by an ensemble of tree-structured base classifiers (Breiman 2001). Using a collection of trees in this manner has shown to give better predictive performance in comparison to single tree classifiers when used in medical diagnosis applications. An important advantage with random forests is that they are stable against overfitting since the generalization error converges when the number of trees becomes large. The model obtained by random forests is complex and hard to interpret, but this disadvantage is of less importance in this study since the focus lies only on the predictive performance of the classification algorithms.

The random forest algorithm used for classification when studying a dataset D with p number of features and n number of cases works as described in the following paragraph. Let n_{learn} denote the number of cases in the learning set used for building the classifier and n_{test} denote the number of test cases. The algorithm has three parameters called *ntree*, *mtry* and *nodesize*. However the effect of parameter *nodesize*, which is the minimum size of terminal nodes in the trees, is negligible and not necessary to tune (Díaz-Uriarte & Alvarez de Andrés 2006). The algorithm starts by picking *ntree* number of bootstrap samples which all will be used to build

a classification tree. Each sample is formed by drawing n_{learn} number of cases with replacement from the learning data. The splits in each tree are not based on all p number of features; instead a random selection of *mtry* number of features is performed. All the *ntree* number of trees are then used to classify each test case. Finally all n_{test} test cases will assign a class label corresponding to the majority vote of the classification trees.

Since the *nodesize* does not need to be tuned, the two parameters to be optimized in random forests are *ntree* and *mtry*. As default value of parameter *mtry* the square root of the number of variables $\sqrt{p}$ is suggested and if more tuning is to be performed examining twice and half of the default value is recommended (Liaw & Wiener 2002). In this study the search space was expanded further by also including $\sqrt{p}$ multiplied by 2^2, 2^3 and 2^{-2}. Parameter *ntree* which is less important than parameter *mtry* (Díaz-Uriarte & Alvarez de Andrés 2006) was investigated for values ranging from 1 000 to 20 000. By studying a large span of values of *ntree*, the effects on the classification accuracy by using a larger number of trees when building the random forests can be explored.

The random forest implementation used in this study is based on the original Fortran code by Breiman and Cutler and it is available in an R package called *randomForest*. The initial parameter values used by the algorithm were the following:

ntree = (1 000, 5 000, 10 000, 20 000)
mtry = $2^i * \sqrt{p}$ rounded to the closest integer where i = -2, -1, 0, 1, 2, 3
nodesize = 1 (default value in the randomForest package for classification tasks)

3.4 Feature selection

In this study a filter based on the Wilcoxon rank sum test was used for feature selection. Filters are the most widely used techniques for feature selection in mass spectra analysis because they are fast and possible to use for a large amount of data (Saeys, Inza & Larrañaga 2007). In addition they are independent of the classification algorithm which makes them useful when comparing different classification algorithms. The features can be selected in two different ways, either a fixed number of features with best discriminatory power are chosen or the features achieving a value smaller than a chosen threshold, for example a p-value less than 0.05 (Hauskrecht et al. 2007). The former strategy was used in this study for different number of features p with the following values:

p = 5, 10, 15, 25, 35

The effect of adding permutation tests to the filter is analysed and described in section 3.4.2.

3.4.1 Wilcoxon rank sum test

The Wilcoxon rank sum test, also known as the Mann-Whitney test, is a statistical test which can be used for ranking features. The Wilcoxon rank sum test compares two unpaired groups of data and it is a non-parametric test which means no assumptions about the distribution of data are needed (Motulsky 1995). The two unpaired groups in this study are the group with healthy patients and the group with diseased patients. The test is two-sided because it does not matter which one of the two groups has higher intensity values, the only interesting aspect is to see is if there is a difference in any direction. The null hypothesis is that the distributions of the two groups are equal. First of all the cases are ranked according to their intensity values

starting with a rank equal to one for the feature with lowest intensity (Wild & Seber 2000). The sum of the ranks for each group is then calculated. In order to test if the two groups come from the same distribution a p-value can be determined using the sum of ranks and the number of cases in each group.

In this study the R function *wilcox.test* from the R package called *stats* was used when calculating p-values of all features according to the Wilcoxon rank sum test. The calculated p-values are then used to rank the features.

3.4.2 Filter with permutation test

The idea behind performing permutation tests is to assess the significance of results, if they are reliable or if they could be due to chance (Radmacher, McShane & Simon 2002). This assessment can be applied on classification results but it can also be used in the process of feature selection. Combining filter techniques with permutation tests can improve the classification accuracy when applied to binary classification tasks (Radivojac, Obradovic & Dunker 2004). In this case the features are first ranked by a filter and then permutation test is used to detect irrelevant features and only selecting those which are significant.

Any scoring metric can be assessed by permutation tests (Hauskrecht et al. 2007) and in this study the combination of a Wilcoxon rank sum filter used together with permutation test is evaluated. Since the pre-processed mass spectrometry datasets contains 1026-9392 variables compared to 6-49 variables in the article by Radivojac et al. (2004) the algorithm used in the comparison was modified. The modification implied that a fixed number of top scoring features were permutated. The effect of permutation tests was analysed by comparing the difference in estimated error rates using a Wilcoxon filter with and without permutation test. This analysis was performed by using 10-fold cross validation (described in 3.5.1) on the BC1, BC2, OC and MSC datasets. In each fold both techniques were used selecting features from the learning set and the impact of two different methods could be compared by looking at the average error rates of all folds. This 10-fold cross validation was repeated 15 times for each dataset.

The procedure of permutation test used in this study was performed as follows:

1) The p-values of all features in the learning data were ranked with a Wilcoxon rank sum filter.
2) The top 80 features were selected and analysed with permutation tests.
3) For each feature f_i , the class labels were randomly permuted 1 000 times.
   a) For each permutation a new p-value* was calculated with a Wilcoxon rank sum filter.
   b) The permutation p-value of feature f_i was given by how many times the p-value* was better than the original p-value divided by the number of permutations.
   c) Step 3 a) – 3 c) were repeated for all top 80 features.
4) The top 40 features ranked by the permutation tests were chosen.

The classification results of the 40 features selected by the described random permutation procedure were compared to the result when selecting the top 40 features obtained by the Wilcoxon rank sum filter. The comparison was performed by running created scripts in R. The results are presented in section 4.2.

3.5 Resampling

In this study 10-fold cross-validation (CV) was used for resampling. 10-fold CV is a special case of *k*-fold cross-validation with *k* set to 10. In an article by Molinaro, Simon & Pfeiffer (2005), some of the most common resampling strategies were compared on "large *p*, small *n*" datasets. One of their conclusions is that resubstitution and split-sample estimations are seriously biased. In contrast, 10-fold CV is found to be one of the least biased methods and, in addition, it requires much less computations than leave-one-out cross validation (LOOCV). One further disadvantage using LOOCV instead of 10-fold CV is that it is more variable (Ambroise & McLachlan 2002). The 0.632 bootstrap estimator is another resampling method with low bias, but it is computationally expensive (Braga-Neto & Dougherty 2004) and compared to 10-fold CV it performs worse on simulated datasets where feature selection is performed on each learning set (Molinaro, Simon & Pfeiffer 2005).

3.5.1 10-fold cross-validation

In 10-fold CV the analysed dataset D is divided into 10 different partitions denoted $d_1,d_2,\ldots,d_{10}$ where each d_i has approximately the same size. In each one of the 10 CV loops one part of the dataset is used as test set and the remaining parts are used as a learning set. The estimation of the prediction error is based on the average classification result achieved from the 10 different test sets. Each case, that is each spectrum in proteomic data, is used for testing once but is also used for building the classifier nine times. The use of stratified CV, taking into account the ratio of different classes in the original dataset and using those proportions in every fold in the CV, has shown to improve the estimating performance (Braga-Neto & Dougherty 2004). In this study stratified 10-fold CV was used as resampling strategy in both the internal and external loop described in section 3.6.1.

3.6 Comparing and evaluating classification algorithms

The framework used for comparing classification algorithms was programmed in R and the classification algorithms were obtained from available machine learning packages. A similar base design consisting of internal and external cross-validation as the one described by Berrar, Granzow & Dubitzky (2007) was used to evaluate the classification algorithms. In this study 10-fold CV was used for both the internal and the external CV and in contrast to the authors' illustrative example four different algorithms were analysed instead of one. The external CV along with feature selection, performed only on the data used for training the classifier, is needed in order to avoid selection bias (Ambroise & McLachlan 2002). To find optimal parameters not only the parameter space of the classification algorithms are explored, but also the number of selected features are investigated. This fact expands the search space with an extra dimension which needs to be applied on the subsets used to build the models (Saeys, Inza & Larrañaga 2007).

To make the comparison as fair as possible all datasets in this study were pre-processed using the same methods and parameters. In an article by Berrar, Bradbury & Dubitzky some further important steps to take into consideration when comparing classification algorithms are discussed (2006). They propose the use of identical test and learning sets for each classification algorithm, and in the same manner the training and validation sets are identical in this study. Furthermore they propose to use the same resampling method, to perform a parameter optimization, to use an external CV and to assess the performance with suitable tests. All these recommendations were taken into consideration in this study. Section 3.6.1 explains the design used in this study and section 3.6.2-3.6.3 describes how the evaluation is

divided into two runs to perform parameter optimization and also how the results are assessed.

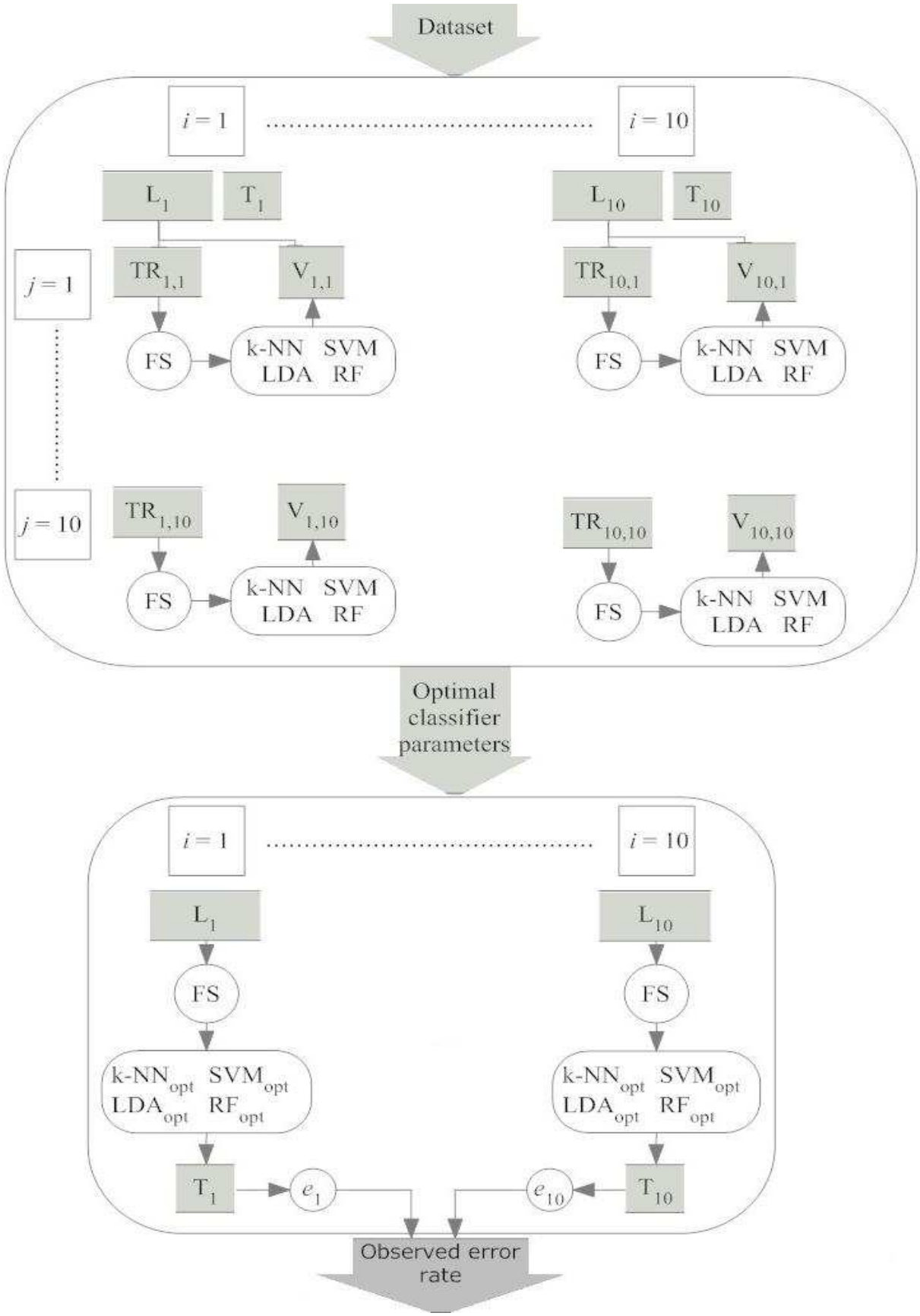
3.6.1 Nested cross-validation design

The design chosen for evaluating the four classification algorithms is shown in figure 4. This design was used to analyse all four datasets and in the following text the steps are described for one dataset denoted *D*. The classification algorithms were compared by using 10-fold CV in both external and internal CV loops. In the figure, index *i* denotes the folds in the external CV while *j* denotes the folds in the internal CV. Initially the input data *D* is divided into ten partitions with similar amounts of spectra from normal and diseased patients respectively. These partitions are the basis of the external CV and in each one of the iterations one partition is used as test set T_i whereas the rest are forming a learning set L_i. The division of *D* into T_i and L_i are shown for the first (i =1) and last iteration (i=10) of the external loop in figure 4.

How the classifications algorithms were evaluated on the test sets is described further in the next paragraph. In order to perform this evaluation though, optimal parameters first need to be found. The parameter tuning is performed with an internal CV on each one of the learning sets created by the external CV. The main purpose of the upper box in figure 4 is to determine optimal parameters of the classification algorithms and also the optimal number of features to select. The internal CV starts to divide each L_i into ten partitions in the same way as for the external CV, but this time the set containing one partition is a validation set V_i and the larger set is a training set TR_i. In figure 4 circles containing "FS" corresponds to the process of feature selection, which in each fold is performed by a Wilcoxon filter, ranking all the features based on the data in TR_i. To investigate how the number of features affects the classification results, the classification algorithms are evaluated with different number of features. For a given TR_i in the internal loop different sets of data are passed to the classification algorithms with the same subset of spectra but with varying amount of features. For each of these variants of TR_i all parameters of the classification algorithms are validated on a subset of V_i containing the same features as in TR_i. When all internal loops are finished the error rates for each unique combination of number of selected features and choice of model parameters are determined by adding the results from all different TR_i. The best parameter settings for each classification algorithm are passed to the lower box in figure 4.

Returning to the external CV, the predictive performance is evaluated when the optimal number of features and the optimal parameters are used to build classifiers. These classifiers are based on the different L_i and tested on the corresponding T_i. The partitions into L_i and T_i are the same in the upper as well as the lower box in figure 4. Feature selection with a Wilcoxon filter is based on the data in L_i, but this time only the optimal number of features is passed to the classification algorithms which only build models based on their optimal parameters. These ten models, one for each fold in the external CV, are tested on the corresponding T_i. For each classification algorithm the observed error rate is determined by adding the number of errors obtained in all of the outer cross-validation folds and then dividing the calculated sum by the total number of test cases. When using cross-validation the total number of test cases is equal to the number of spectra in the dataset, since all of them are used for testing once.

Fig. 4. The nested CV design. The upper box shows the parameter tuning using internal CV and the lower box shows how the observed error rates is determined using an external CV.



Explanations to the abbreviations in figure 4:

i	The fold of the external CV
j	The fold of the internal CV
L_i	Learning set in external fold *i*
T_i	Test set in external fold *i*
$TR_{i,j}$	Training in external fold *i* and internal fold *j*
$V_{i,j}$	Validation in external fold *i* and internal fold *j*
e_i	Error rate in external fold *i*
FS	Feature selection with a Wilcoxon rank sum filter
k-NN	Classification with k-Nearest Neighbor
LDA	Classification with Linear Discriminant Analysis
SVM	Classification with Support Vector Machine
RF	Classification with Random Forest

3.6.2 Identify smaller parameter regions

When comparing the performance of different classification algorithms optimal parameter settings need to be found for each algorithm. Dividing the parameter spaces into small steps and investigating all combinations requires a lot of computations. In order to save time the optimization can be divided into two steps. In step one all of the parameter intervals are divided into large steps and good parameter regions are then determined in which the search should continue. Useful regions are determined by analysing graphs showing how the error rates depend on the parameter values. In the second step the good region is examined more thoroughly by dividing it into even smaller steps.

The identifying of smaller parameter regions is done for all four datasets with the following initial parameter settings:

$k = 1, 4, 7, \ldots, 51$
$C = 2^{-5}, 2^{-2}, 2^{1}, \ldots, 2^{16}$
$\gamma = 2^{-15}, 2^{-12}, 2^{-9}, \ldots, 2^{3}$
ntree = (1 000, 5 000, 10 000, 20 000)
mtry = $2^i * \sqrt{p}$ rounded to the closest integer where $i = -2, -1, 0, 1, 2, 3$
$p = 5, 10, 15, 25, 35$

How the new parameter spaces are determined for the BC1, BC2, OC and MSC datasets respectively is shown in the section 4.3.1-4.3.4.

3.6.3 Final run and assessment of algorithms

In the final run the nested cross-validation design described in section 3.4.2 is used. The parameter spaces to be explored are the ones obtained from the previous run. The results from the final run are used for comparing the performances of the classification algorithms. Comparing the obtained values of observed error rate ε or the accuracy of different classification algorithms is not sufficient since ε is only an estimate of the true error rate τ (Berrar, Bradbury & Dubitzky 2006). Confidence interval for the true error rate τ with confidence level 1-α and observed error rate ε is given by

$$\tau \approx \varepsilon \pm \left(\frac{1}{2M} + \phi^{-1}\left(1-\frac{\alpha}{2}\right)\sqrt{\frac{\varepsilon(1-\varepsilon)}{M}} \right), \qquad (1)$$

where ϕ is the normal cumulative distribution function and M is the number of test cases. Berrar, Bradbury & Dubitzky (2006) propose a use of tests taking into consideration the resampling strategy and tests which corrects for multiple testing.

Correcting for the random variation caused by the chosen resampling strategy is performed to handle situations where a classifier by chance performs better than another one for a particular partitioning into learning and test sets. In this case the differences in performance in each fold of the CV need to be tested. Let p_{Ai} denote the error rate for classifier A and p_{Bi} denote the error rate for classifier B in the i-th fold of the CV and let the difference be denoted p_i where $p_i = p_{Ai} - p_{Bi}$. The average performance difference p_{avg} over the k folds in CV is given by

$$p_{avg} = \frac{1}{k}\sum_{i=1}^{k} p_i \quad .$$

Let the variance s^2 in error rates of the two models over the k folds be given by

$$s^2 = \frac{1}{k-1}\sum_{i=1}^{k}\left(p_i - p_{avg}\right)^2 ,$$

then the test statistic T_c correcting for the variance of the CV can be given by

$$T_c = \frac{p_{avg}}{\sqrt{\left(\frac{1}{k} + c\right)s^2}} \qquad (2)$$

where c is a correction term which can be set to the ratio of the number of learning cases to the number of test cases in a CV fold (Nadeau & Bengio 2003).

When comparing multiple classification algorithms correction for multiple testing is also needed. Let q be the number of classification algorithms, then the number of possible pairwise comparisons κ is given by

$$\kappa = \frac{\mathrm{q!}}{2!(\mathrm{q}-2)!} .$$

Since this study includes four classification algorithms there are six pairwise comparisons. The procedure used for multiple testing was the Bonferroni correction.

4. Results

The results of this study are presented in the following sections (4.1-4.4) in the same order as they where obtained. In section 4.1 the results of the evaluation of different methods replacing missing values is presented. The effects of using permutation tests in combination with a Wilcoxon rank sum filter is shown in section 4.2. Section 4.3 describes how parameter regions resulting in low error rates were found. Section 4.4 presents the main results of this study which is the comparison of different classification algorithms.

4.1 Evaluation of methods handling missing values

The three compared methods handling missing values were *Zero impute*, *Mean impute* and *KNNimpute* which were all tested on the BC1 dataset. In each of the 100 iterations all three methods imputed values to replace the 50 removed values. The results are presented in two different figures to make it easier to compare the RMS errors and to conclude which method to use in the classification study.

Fig. 5. Comparison of *KNNimpute* and *Zero impute* in terms of the RMS error.

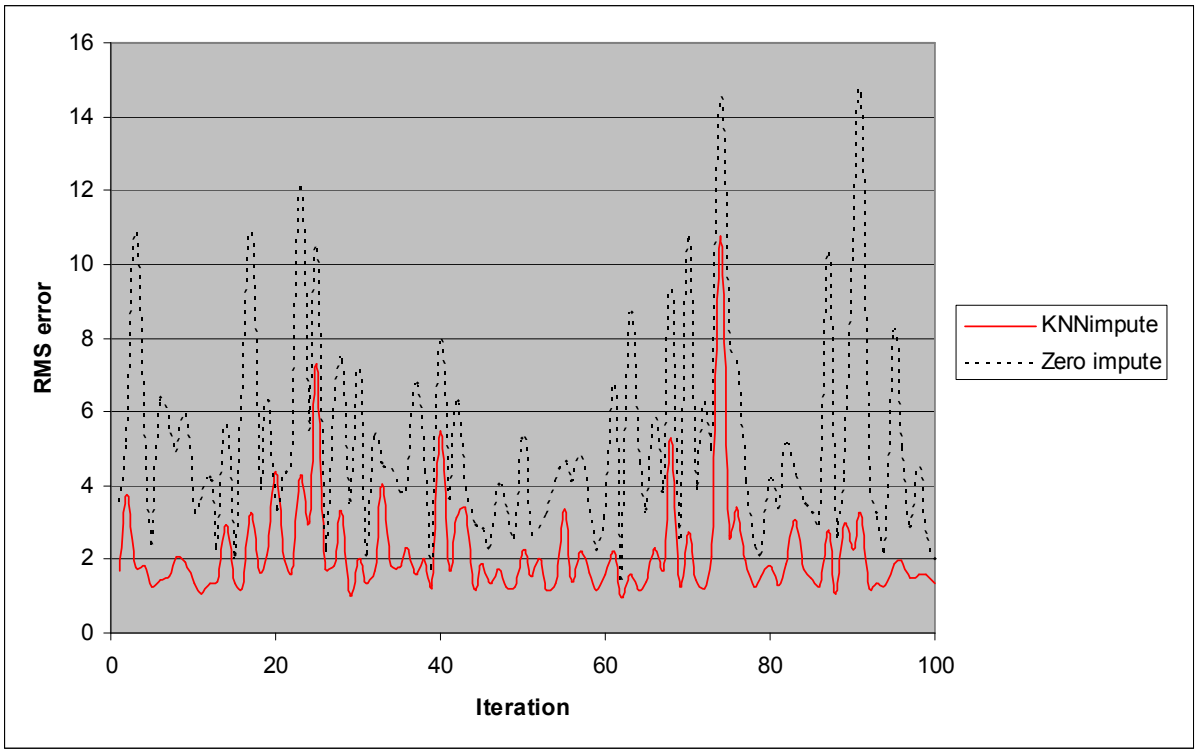
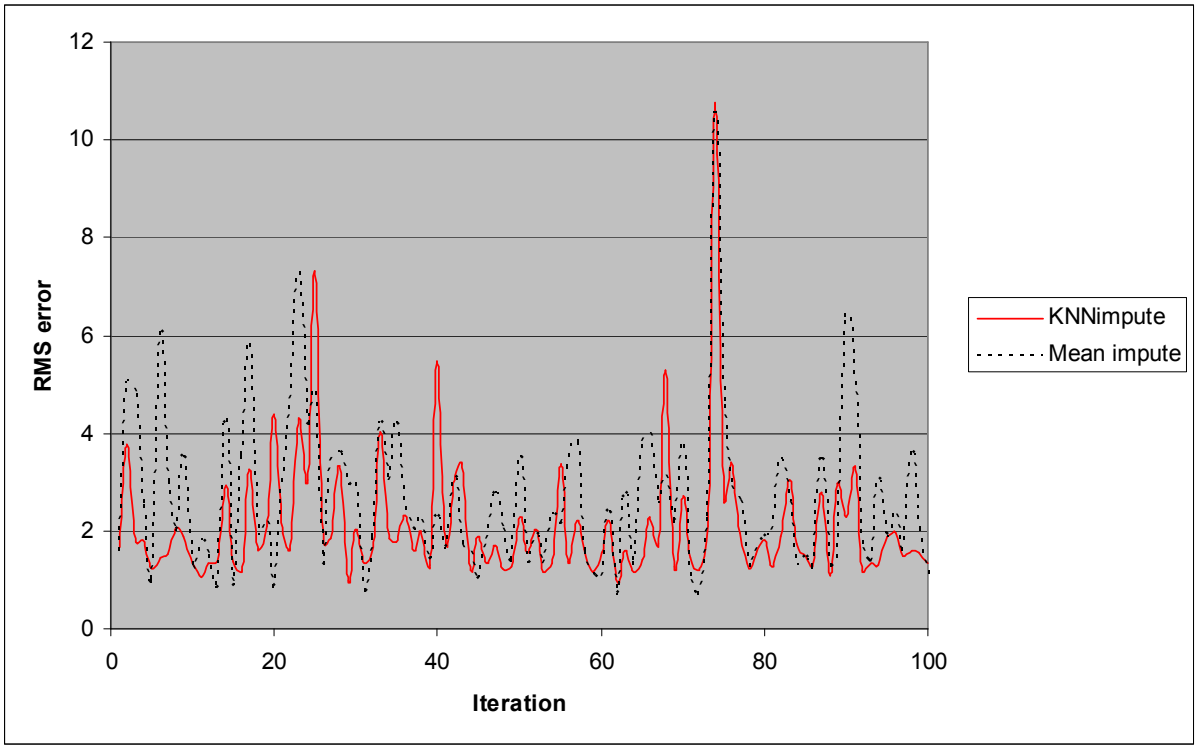


Figure 5 clearly shows that the resulting RMS error is lower for *KNNimpute* than for *Zero impute* in almost all iterations. To find the best method, *KNNimpute* is compared to *Mean impute* (see figure 6) whereas no comparison of *Zero impute* and *Mean impute* needs to be visualized.

Fig. 6. Comparison of *KNNimpute* and *Mean impute* in terms of the RMS error.



The difference between *KNNimpute* and *Mean impute* is not as obvious as in the former case, but *KNNimpute* seems to be more correct in this case also. The mean RMS errors over all 100 iterations are 2.13 for *KNNimpute*, 2.64 for *Mean impute* and 4.95 for *Zero impute*. Consequently the mean RMS error is 24% larger for *Mean impute* than for *KNNimpute*. Additionally *KNNimpute* achieves the lowest RMS error among the three methods in 69 % of the iterations. Thus *KNNimpute* was chosen to be the method handling the missing values in all four datasets in order to prepare them for further analysis.

4.2 Feature selection with or without permutation tests

The effect of permutation tests was analysed by comparing the difference in estimated error rate using a Wilcoxon filter with, as well as without, permutation tests. The evaluation was made using all the classifiers (*k*-NN, SVM, RF and LDA) with their default parameter values and 10-fold CV as resampling method. This procedure was repeated 15 times for each dataset (BC1, BC2, OC and MSC). The feature selection was only made on the learning sets and not the test sets. For each run totally 10 feature selection procedures were performed, one per fold in the cross validation.

For each combination of dataset and classification algorithm the effect of using permutation tests in the feature selection process is shown in table 3. In total, the best performance was achieved in 63 cases out of 240 when using permutation tests, whereas only using a filter was the best choice in 62 cases. In the remaining115 cases the performance was equal for the two strategies. The wanted effect, improvement in classification performance, was not significant in either case (with or without using permutation tests). Within these results, the results varied a lot for different classification algorithms and different datasets. In the classification study the Wilcoxon rank sum filter was used without permutation tests in the following sections (4.3-4.4) since evidence for improved performance was not found.

Table 3. The number of times Wilcoxon ranking without a permutation test achieves the best performance (lowest error rate) is shown in the first row for each classifier. The second row shows the number of times the feature selection with an additional permutation test results in better performance and the third row displays how many times there is no difference between the two. Each combination of dataset and classification algorithm can be summarised to 15 since it is the number of times the cross validation procedure was repeated.

		Dataset				
		BC1	BC2	OC	MSC	Total
LDA	LDA > LDA*	3	9	0	0	12
	LDA < LDA*	9	4	0	1	14
	LDA = LDA*	3	2	15	14	34
KNN	k-NN > k-NN*	10	8	0	1	19
	k-NN < k-NN*	4	6	0	0	10
	k-NN = k-NN	1	1	15	14	31
SVM	SVM > SVM*	4	8	0	0	12
	SVM < SVM*	7	5	0	0	12
	SVM = SVM*	4	2	15	15	36
RF	RF > RF*	5	7	3	4	19
	RF < RF*	8	2	10	7	27
	RF = RF*	2	6	2	4	14

Explanations to the symbols used in table 3:

*	indicates that permutation tests have been used with the Wilcoxon filter
>	method 1 > method 2 means that method 1 performed better than method 2
<	method 1 < method 2 means that method 1 performed worse than method 2
=	method 1 = method 2 means equal performance of the two methods

4.3 Identifying smaller parameter regions

As described in section 3.6.2, comparing different classification algorithms requires optimization of the parameters. The first search for smaller parameter regions is performed for all datasets with the following parameter settings:

$k = 1, 4, 7, \ldots, 51$
$C = 2^{-5}, 2^{-2}, 2^{1}, \ldots, 2^{16}$
$\gamma = 2^{-15}, 2^{-12}, 2^{-9}, \ldots, 2^{3}$
$ntree = (1\,000, 5\,000, 10\,000, 20\,000)$
$mtry = 2^{i} * \sqrt{p}$ rounded to the closest integer where $i = -2, -1, 0, 1, 2, 3$
$p = 5, 10, 15, 25, 35$

Parameter p above is the number of selected features and it is optimized for all four classification algorithms. The plots in sections 4.3.1-4.3.4 show the different error rates obtained for different classification algorithm parameters when the optimal value of p was used. The error rates obtained in the parameter tuning for different classification algorithms are not supposed to be compared to each other. This is because the intervals of different regions in the contour plots are of different size. The final comparison between methods is presented in section 4.4.

4.3.1 BC1

When analysing the BC1 dataset all four classification algorithms achieves the lowest error rate when picking the top 35 features ($p = 35$). In addition, every step including more features (all steps from $p = 5$ to $p = 35$) results in a lower error rate for all algorithms. For the final run the values of p are ranging from 26 to 35 to cover the most interesting values.

Figure 7 shows that for k-NN a minimum error rate is achieved when $k = 10$ and for increasing values of k the trend is that the error rate increase. For the final run, classification using $4 \leq k \leq 22$ was evaluated which corresponds to the span made up by the seven best error rates in the figure.

Fig. 7. k-NN parameter k ($p = 35$)

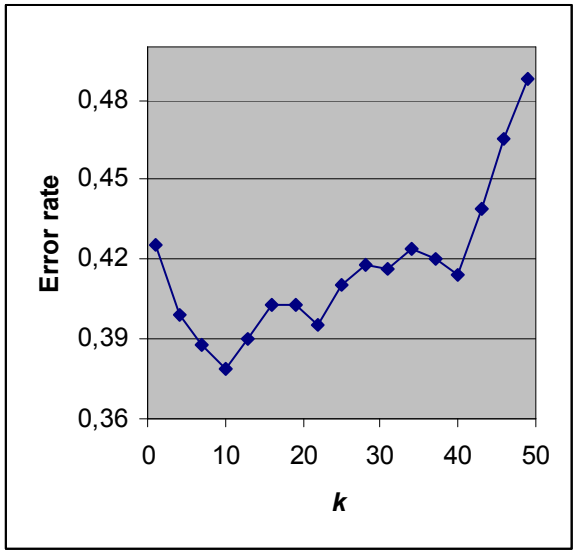
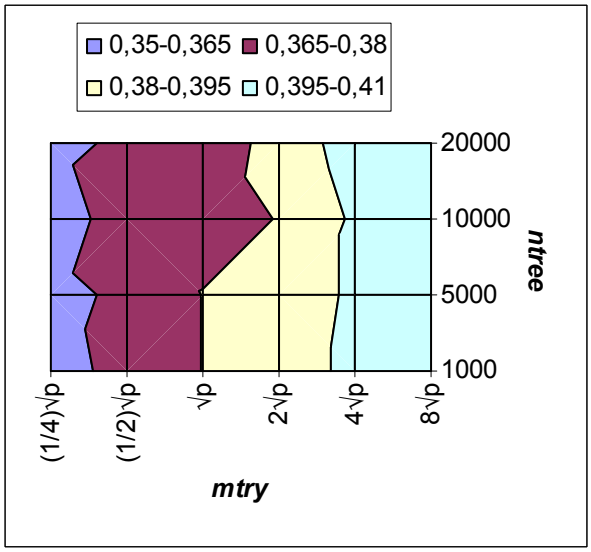


Fig. 8. RF parameters *ntree* and *mtry* ($p = 35$)



When using RF it seems like different tested values of parameter *ntree* do not affect the error rate much since there are almost parallel bands of different error rate regions depending only on the value of *mtry* (figure 8). The figure also shows that lower values of *mtry* give better performance. Since it is possible that values of parameter *ntree* larger than 20 000 or smaller than 1 000 may increase the performance an expanded search is performed with the following settings:

ntree = (100, 200,…, 1 000) & (2 000, 3 000,…,10 000) & (15 000, 20 000,…, 35 000)
mtry = $2^i * \sqrt{p}$ rounded to the closest integer where i = -2, -1

The result is presented in figure 9 and it shows that the performance is better for the lower value of *mtry*. When investigating the good regions further the lowest error rates are found for low values of *ntree*. The lowest error rate is obtained for *ntree* = 200 and in the final run parameter *ntree* ranges from 100 to 1 000 to cover the best region.

Fig. 9. Further analysis of RF parameters *ntree* and *mtry* (p = 35)

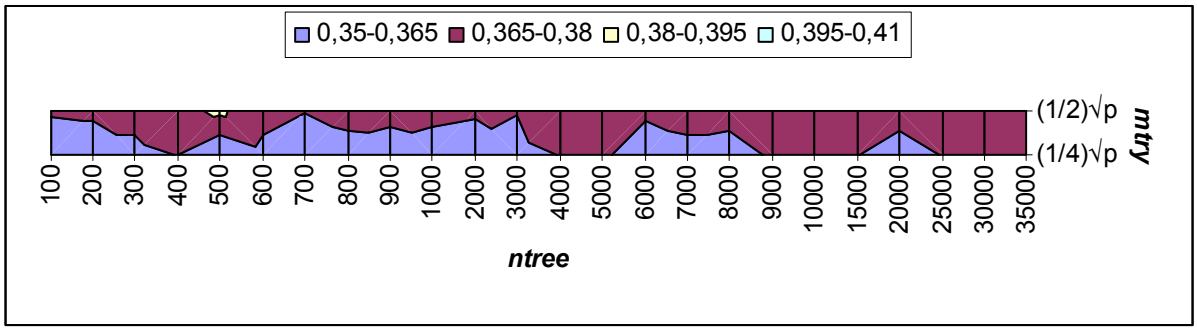
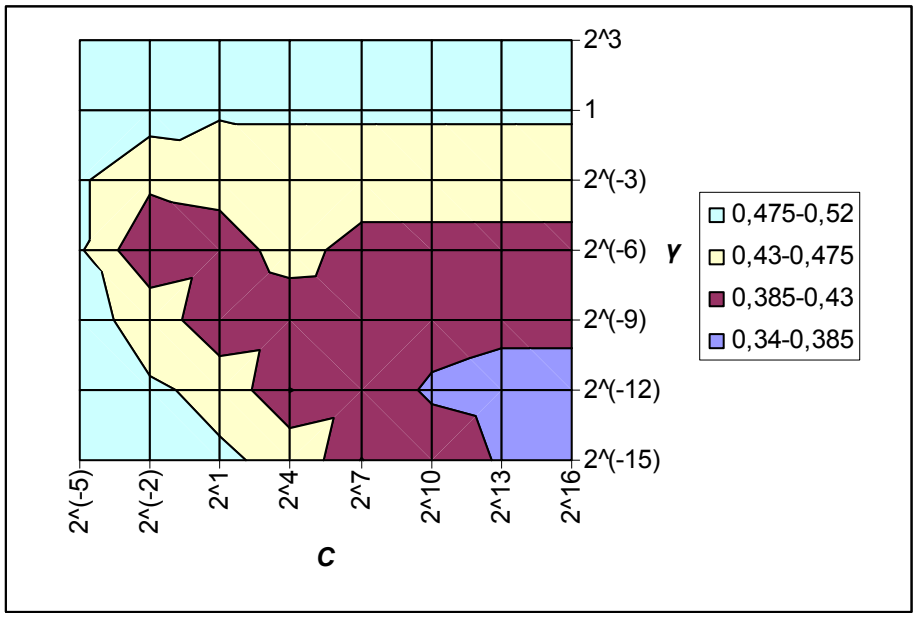


Figure 10 shows how the combination of parameters C and γ affect the error rate. The region with lowest error rate is found in the lower right corner where the values of γ are small and the values of C are large. Since this region is located along the boundaries it is possible that even better combinations of parameters exist outside the figure. Therefore it is necessary to expand the search space further and investigate more combinations of C and γ.

Fig. 10. SVM parameters C and γ (p = 35)



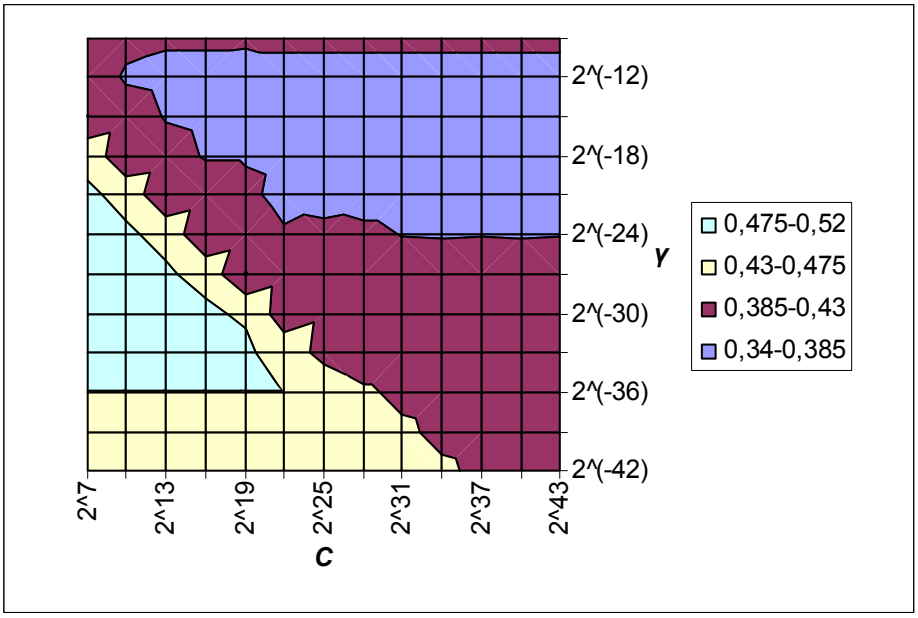
The values of the SVM parameters used in the additional run were the following:

$C = 2^{7}, 2^{10}, 2^{13}, \ldots, 2^{43}$
$\gamma = 2^{-42}, 2^{-39}, 2^{-36}, \ldots, 2^{-9}$

The result of the expanded search space is shown in figure 11. The six squares in the upper left corner are the same as in the lower right corner of figure 9. Parameter C and γ are evaluated for values ranging to 2^{43} = 8796093022208 and 2^{-42} = 0.0000000000002273737 compared to 2^{16} = 65536 and 2^{-15} = 0.00003051758 in the first run. For C ranging from 2^{28} to 2^{43} the error rates are constant in the blue region for the top four γ parameter values (2^{-12} to 2^{-21}). Therefore no additional search for lager values of C was performed. Let (C, γ) denote a combination of parameter C and γ. The lowest error rates are achieved for two combinations,

(2^{25}, 2^{-21}) and (2^{21}, 2^{-12}), which both are covered by the region used for optimization in the final run.

Fig. 11. Further analysis of SVM parameters *C* and *γ* (*p* = 35)



Parameter settings chosen for the final run used on BC1 were the following:

<u>*k*-NN</u>
$k = 4, 5, 6, \ldots, 22$
$p = 26, 27, 28, \ldots, 35$

<u>RF</u>
ntree = 100, 200, 300,…, 1 000
$mtry = 2^i * \sqrt{p}$ rounded to the closest integer where $i = -2, -3/2, -1$
$p = 26, 27, 28, \ldots, 35$

<u>SVM</u>
$C = 2^{16,0}, 2^{16,5}, 2^{17,0}, \ldots, 2^{28,0}$
$\gamma = 2^{-24,0}, 2^{-23,,5}, 2^{-23,0}, \ldots, 2^{-9,0}$
$p = 26, 27, 28, \ldots, 35$

<u>LDA</u>
$p = 26, 27, 28, \ldots, 35$

4.3.2 BC2

The dependency of number of features *p* is not as clear for the BC2 dataset as in the case of the BC1 dataset. For the BC2 dataset the optimal performance for LDA and SVM is achieved when 15 features are used while the *k*-NN and RF get lower error rates when using 25 and 35 features respectively.

In figure 12 it is visible that the lowest error rates for *k*-NN are achieved when *k* is equal to 13, 16 and 19. Both lower and higher values of *k* result in an increased error rate.

Fig. 12. *k*-NN parameter *k* (*p* = 25)

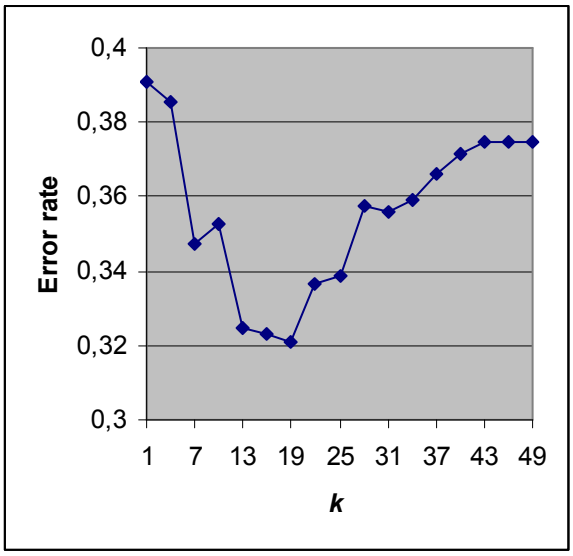
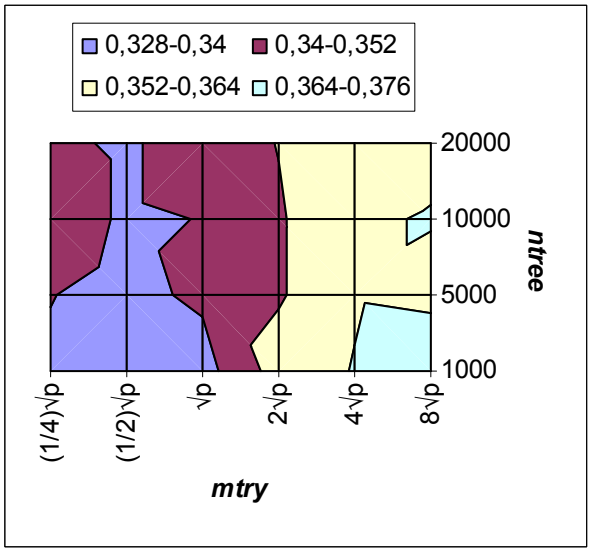
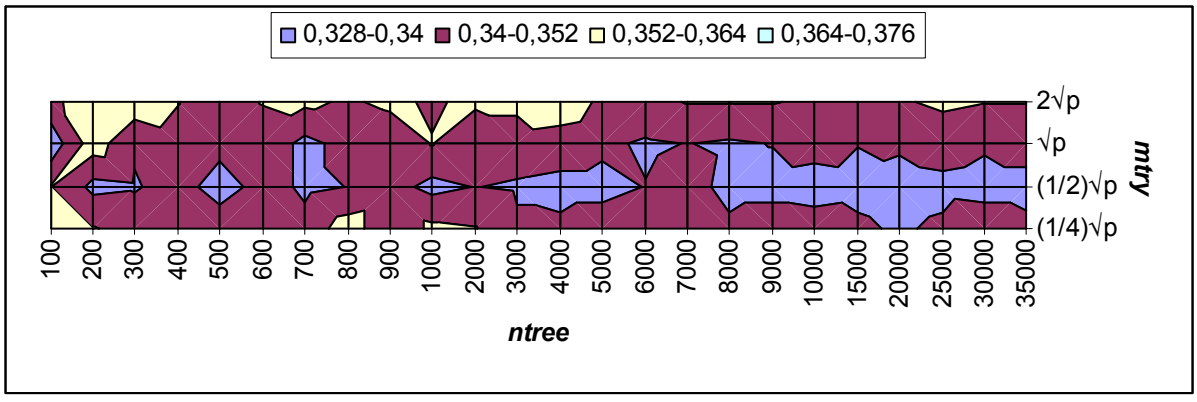


Fig. 13. RF parameters *ntree* and *mtry* (*p* = 35)



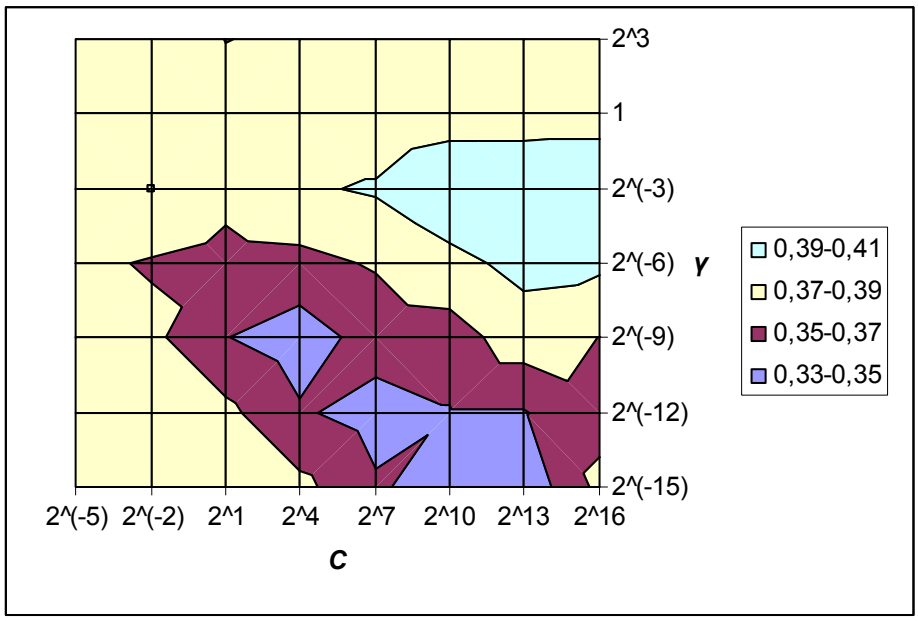
The contour plot for the RF algorithm shows that large values of *mtry* give large error rates (figure 13). The best performance is achieved for values of *mtry* around $(1/2)p^{-1/2}$. Figure 14 shows how the error rates depend on large quantity of different values of *ntree* for the top four values of *mtry*. Still the best performance is achieved when *mtry* is set to $(1/2)p^{-1/2}$. Even though the regions with low error rates are larger for large values of *ntree* the best performance is achieved for low values of *ntree*. The parameter values chosen for the final run is based on the analysis within the good regions and includes the lowest error rates.

Fig. 14. Further analysis of RF parameters *ntree* and *mtry* (*p* = 35)



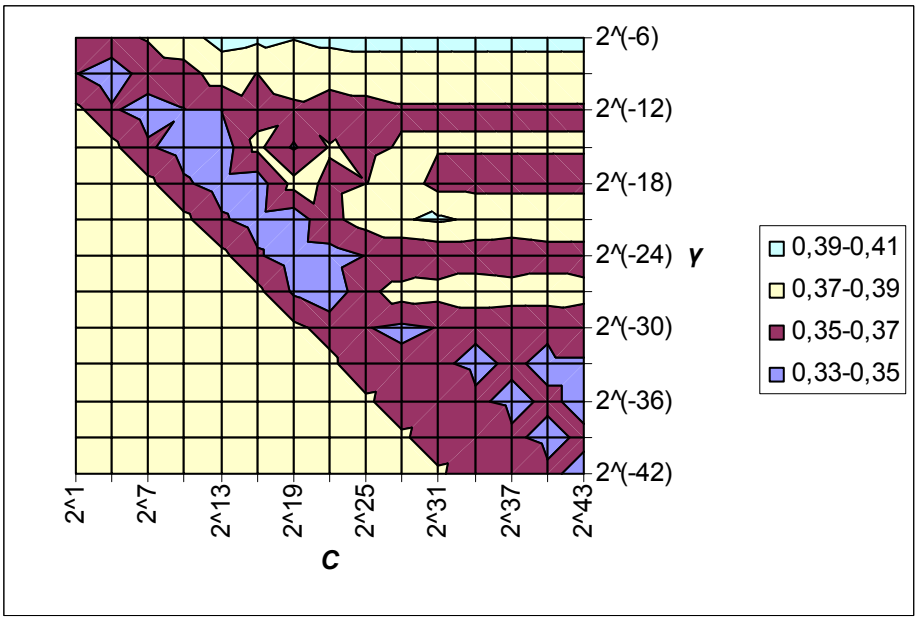
The combinations of SVM parameters show a similar pattern as for the BC1 dataset since large values of *C* and small values of γ give low error rates in the first run (figure 15).

Fig. 15. SVM parameters C and γ (p = 15)



Also for this dataset the search space needs to be expanded. The result using $C = 2^{1}, 2^{4}, 2^{7},..., 2^{43}$ and $\gamma = 2^{-42}, 2^{-39}, 2^{-36},..., 2^{-6}$ is shown in figure 16. The eleven lowest error rates are located in the largest good region which is situated between values of C ranging from 2^{4} to 2^{25}. Even though there are good regions located at the boundaries in the lower right corner no further expansion is needed. This is because a finer partitioning of error rates would not include any of the combinations to the right of $C = 2^{25}$ in the interval corresponding to the lowest error rates. The SVM parameters chosen for the final run include the parameter combinations which result in lowest error rates in the best region.

Fig. 16. Further analysis of SVM parameters C and γ (p = 15)



Parameter settings chosen for the final run used on BC2 were the following:

<u>*k*-NN</u>
$k = 13, 14, 15, \ldots, 25$
$p = 16, 17, 18, \ldots, 34$

<u>RF</u>
ntree = 100, 200, 300,…, 1 000
mtry = $2^i * \sqrt{p}$ rounded to the closest integer where $i = -2, -3/2, -1, -1/2, 0, 1/2, 1$
$p = 26, 27, 28, \ldots, 35$

<u>SVM</u>
$C = 2^{10,0}, 2^{10,5}, 2^{11,0}, \ldots, 2^{25,0}$
$\gamma = 2^{-30,0}, 2^{-29,5}, 2^{-29,0}, \ldots, 2^{-12,0}$
$p = 11, 12, 13, \ldots, 24$

<u>LDA</u>
$p = 11, 12, 13, \ldots, 24$

4.3.3 OC

When analysing the OC dataset LDA performs best with 15 selected features whereas the rest of the algorithms achieves the lowest error rates when only the top 10 features are selected.

The curve in figure 17 visualises how parameter *k* in *k*-NN affects the error rate. The curve shows a similar shape as the one obtained for the BC2 dataset. In this case the values used for parameter *k* were between 10 and 25 compared to values between 13 and 25 for the BC2 dataset.

Fig. 17. *k*-NN parameter *k* (*p* = 10)

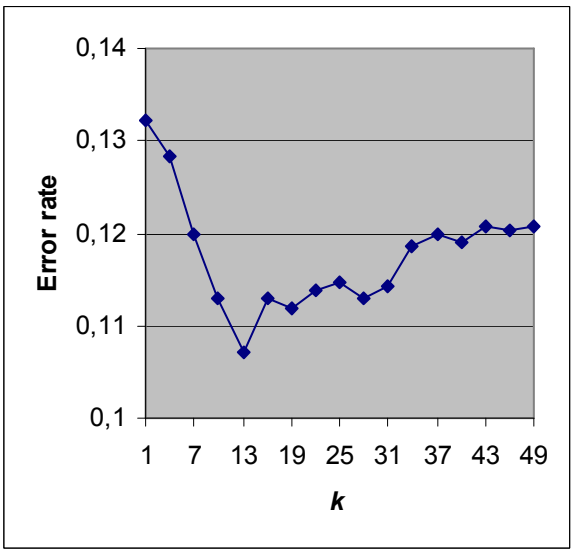
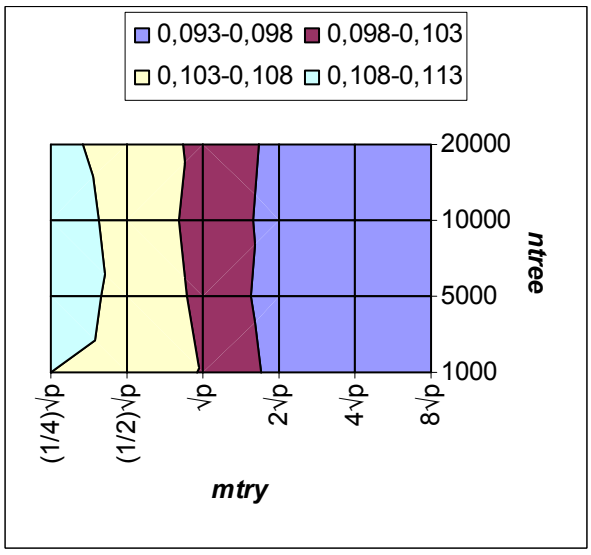


Fig. 18. RF parameters *ntree* and *mtry* (*p* = 10)



Again, the values of parameter *ntree* are of less importance compared to the values of parameter *mtry* (figure 18). As opposed to the other datasets the error rate is small for large values of *mtry* which imply the need of a further investigation of larger values of *mtry*. The factor prior to the square root of *p* was set to 16 and 32 in an expanded search which also

explored a large range of values of parameter *ntree* (figure 19). Larger factors do not need to be tested since the error rates increase for the largest values of *mtry* within the good region.

Fig. 19. Further analysis of RF parameters *ntree* and *mtry* (*p* = 10)

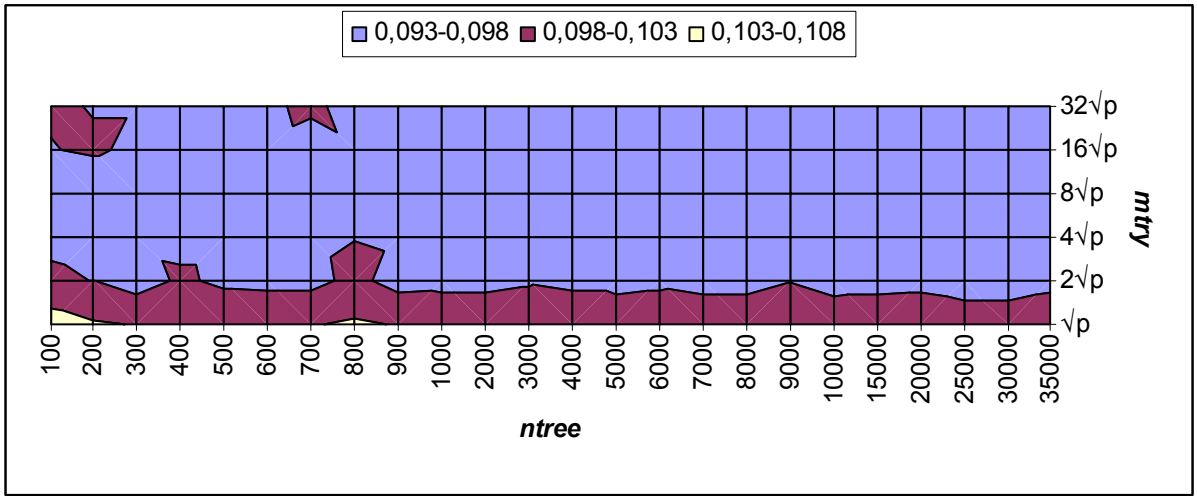
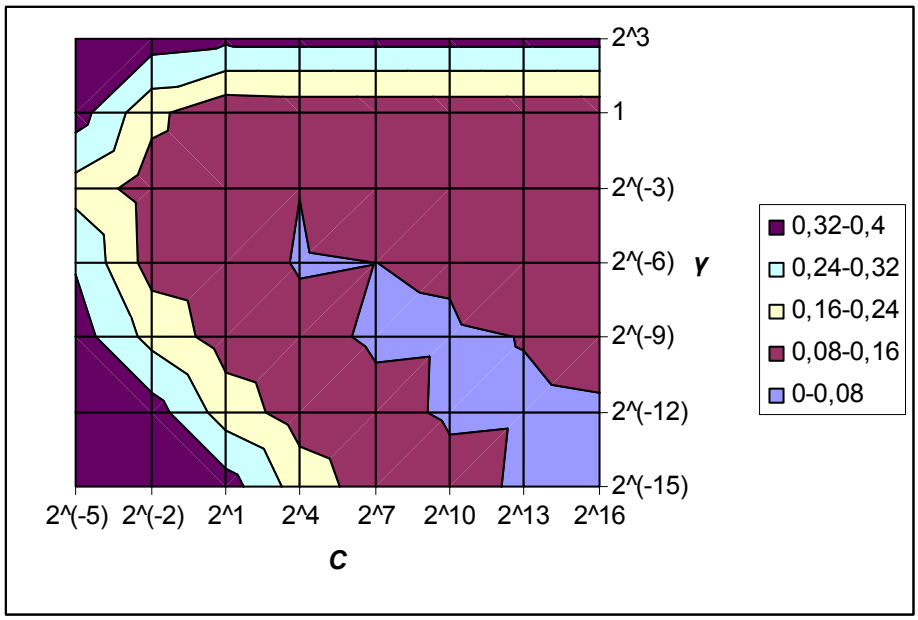


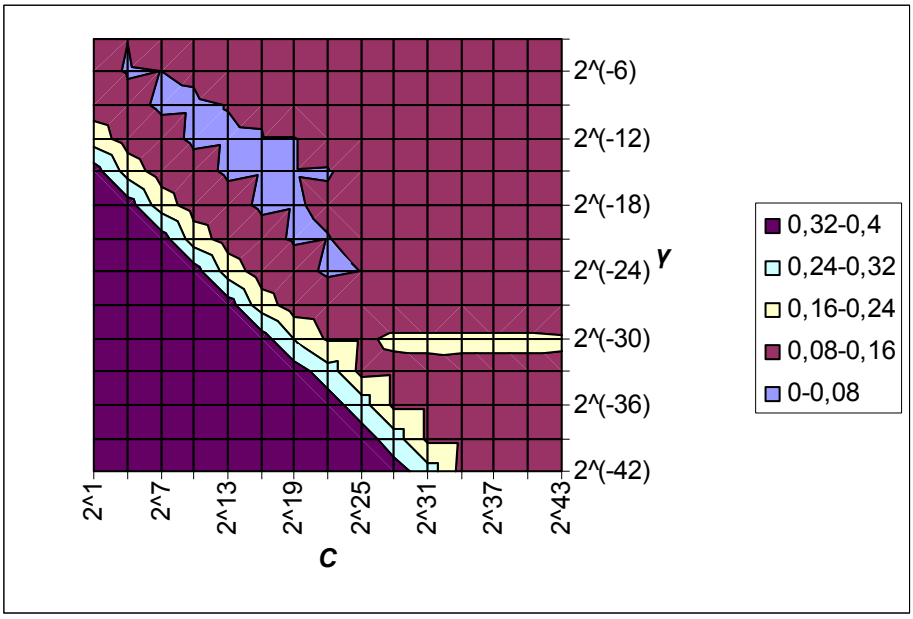
Figure 20 illustrates that also for the OC dataset the SVM parameters show a similar pattern as for the BC1 and BC2 datasets. The only difference is that the good region spans over a larger area.

Fig. 20. SVM parameters *C* and *γ* (*p* = 10)



The further analysis of SVM parameters shown in figure 21 was performed with $C = 2^{1}, 2^{4}, 2^{7},..., 2^{43}$ and $\gamma = 2^{-42}, 2^{-39}, 2^{-36},..., 2^{-3}$. This expanded search space was large enough as seen in figure 21. The lowest error rates are found in the upper left part of the good region and the two best combinations of *C* and *γ* are $(2^{7}, 2^{-9})$ and $(2^{10}, 2^{-12})$.

Fig. 21. Further analysis of SVM parameters *C* and *γ* (*p* = 10)



Parameter settings chosen for the final run used on OC were the following:

<u>*k*-NN</u>
$k = 10, 11, 12, \ldots, 25$
$p = 6, 7, 8, \ldots, 14$

<u>RF</u>
ntree = 950, 1 000, 1 250, 1 500, 1 750, 2 000
$mtry = 2^i * \sqrt{p}$ rounded to the closest integer where $i = 2, 5/2, 3, 7/2, 4$
$p = 6, 7, 8, \ldots, 14$

<u>SVM</u>
$C = 2^{4,0}, 2^{4,5}, 2^{5,0}, \ldots, 2^{13,0}$
$\gamma = 2^{-15,0}, 2^{-14,5}, 2^{-14,0}, \ldots, 2^{-6,0}$
$p = 6, 7, 8, \ldots, 14$

<u>LDA</u>
$p = 11, 12, 13, \ldots, 24$

4.3.4 MSC

The MSC dataset which is the dataset with the largest amount of spectra requires 25 features when using RF but only 5 features for the other three algorithms.

Low values of parameter *k* in *k*-NN results in high error rates, but with increasing values of *k* the error rate is decreasing until it fades out and remains almost constant from approximately $k = 25$ (figure 22).

Fig. 22. *k*-NN parameter *k* (*p* = 5)

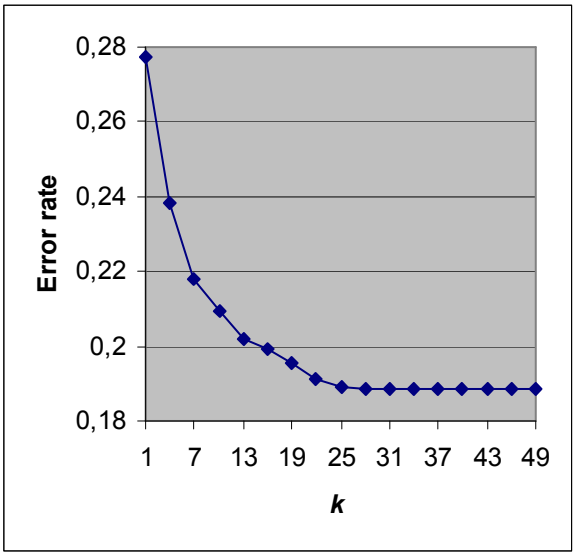
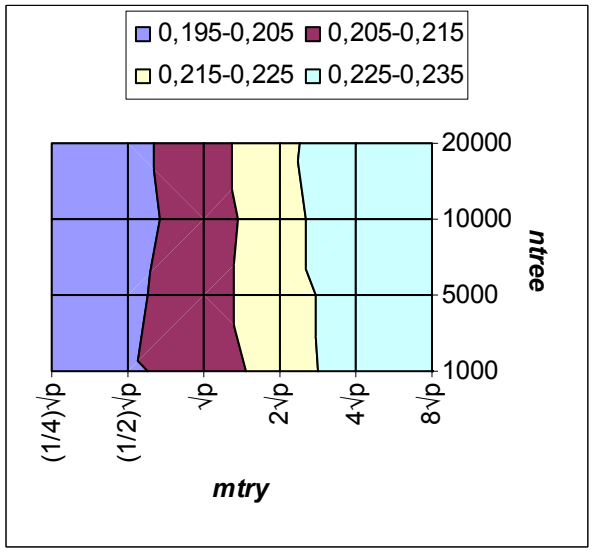
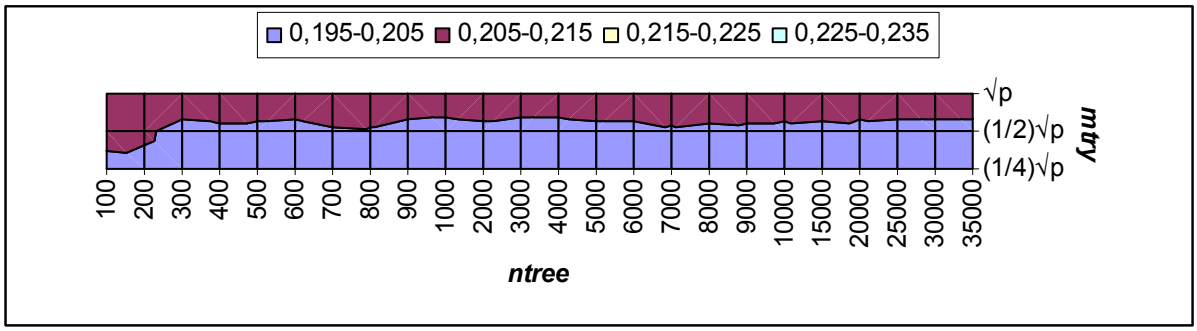


Fig. 23. RF parameters *ntree* and *mtry* (*p* = 25)



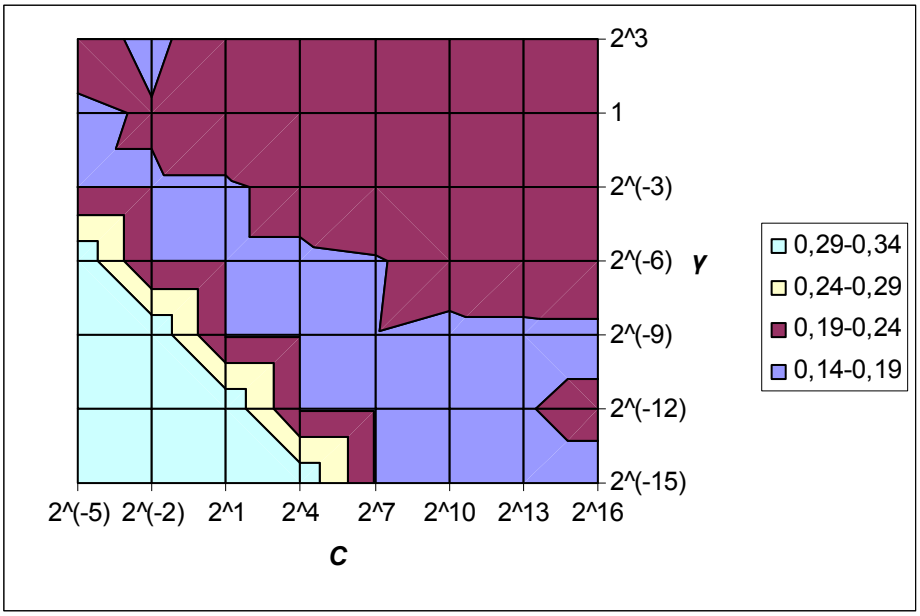
Not too surprisingly, the parameter *ntree* does not affect the results much in this dataset either (figure 23). Low values of *mtry* decrease the error rate. Figure 24 shows the impact of parameter *ntree* for the top three values of *mtry*. The dependence of parameter *ntree* is almost constant within the whole range of values except for the smallest values to the left in the figure.

Fig. 24. Further analysis of RF parameters *ntree* and *mtry* (*p* = 25)



For SVM the results differ a bit from previous datasets. In contrast to the other datasets the good region is more stretched and not only located in the lower right corner as seen in figure 25.

Fig. 25. SVM parameters C and γ (p = 5)



Since there is a good region along the entire diagonal including both the upper left and lower right corner, the search space must be expanded in both corners. The expanded search was divided into two runs due to the size of the dataset and the large area to explore. Figure 26 shows the result of the first run investigating the search space which was expanded along the upper left corner for $C = 2^{-17}, 2^{-14}, 2^{-11},..., 2^{7}$ and $\gamma = 2^{-12}, 2^{-9}, 2^{-6},..., 2^{12}$. The result of the second run exploring the lower right corner is shown in figure 27. In this case the $C = 2^{7}, 2^{10}, 2^{13},..., 2^{31}$ and $\gamma = 2^{-30}, 2^{-27}, 2^{-24},..., 2^{-6}$ were investigated.

Fig. 26. Analysis of small C and large γ (p = 5)

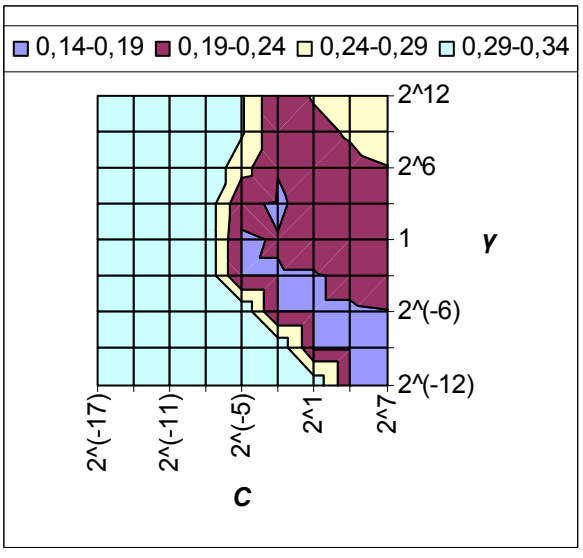
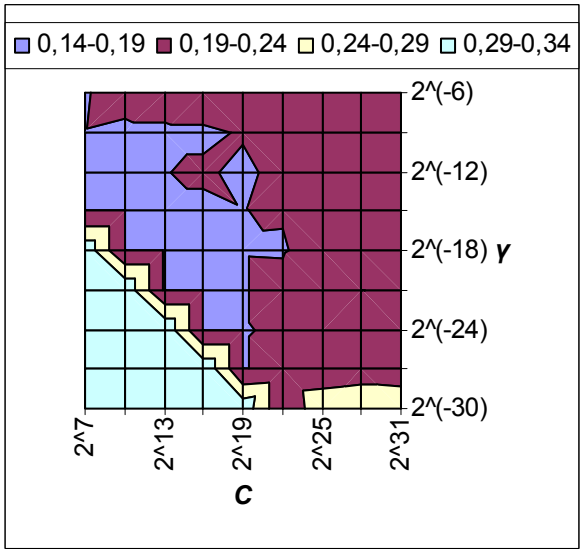


Fig. 27. Analysis of large C and small γ (p = 5)



Within the best regions in figure 26 and 27 the lowest error rate is found for point (2^{-2}, 2^{3}) and the second best point at (2^{-5}, 2^{0}) which are located close to each other. the final run cover these two points and the surrounding area.

Parameter settings chosen for the final run used on MSC were the following:

<u>*k*-NN</u>
$k = 25, 26, 27, \ldots, 35$
$p = 2, 3, 4, \ldots, 9$

<u>RF</u>
$ntree = (7500, 8000, 8500)$
$mtry = 2^i * \sqrt{p}$ rounded to the closest integer where $i = -2, -1$
$p = 16, 17, 18, \ldots, 34$

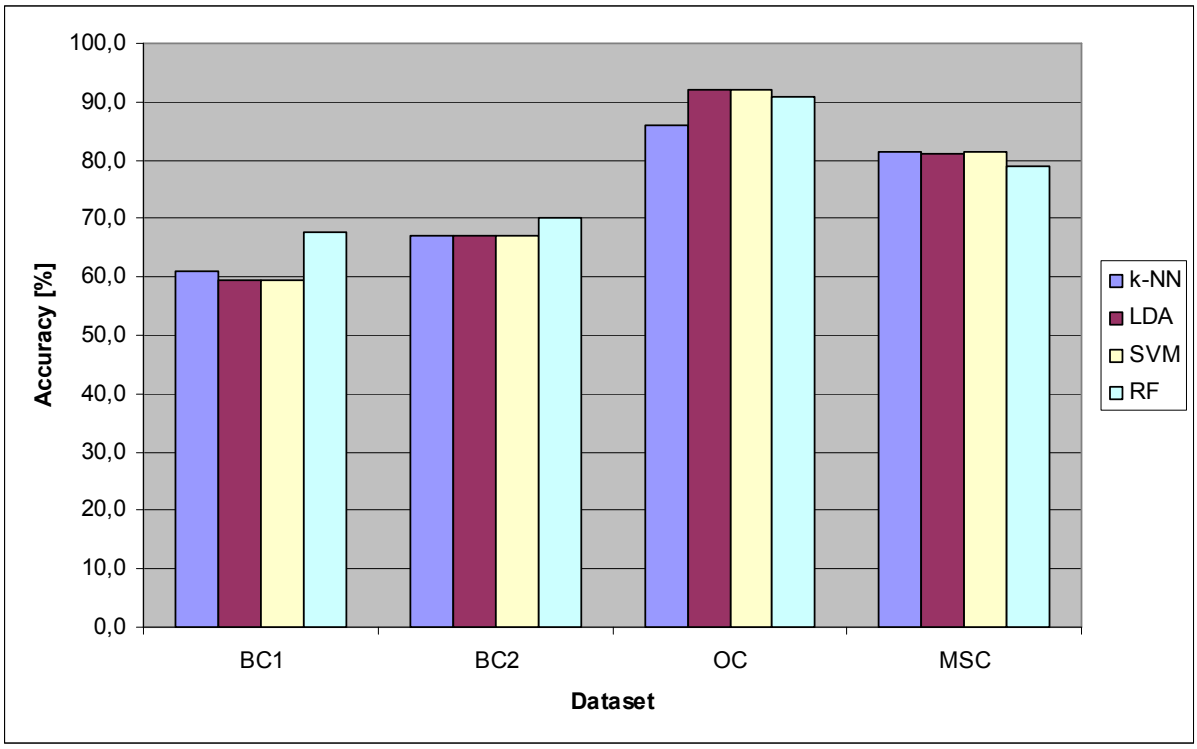
<u>SVM</u>
$C = 2^{-8,0}, 2^{-7,5}, 2^{7,0}, \ldots, 2^{1,0}$
$\gamma = 2^{-3,0}, 2^{-3,5}, 2^{-4,0}, \ldots, 2^{6,0}$
$p = 2, 3, 4, \ldots, 9$

<u>LDA</u>
$p = 2, 3, 4, \ldots, 9$

4.4 Final run

In the final run, the parameters were tuned within the identified smaller parameter regions obtained in sections 4.3.1-4.3.4. The nested CV design explained in section 3.6.1 was used in the final run for tuning the parameters and for determining the observed accuracies. Figure 19 shows the accuracies obtained for the four different classification algorithms when applied on the four datasets used in this study. For the BC2 dataset, three of four algorithms show no difference in performance; all result in an accuracy of 67.2%. The variation in performance of different classification algorithms is small for all the studied datasets. If only looking at the observed accuracies it seems like RF is the method which overall best classifies spectra as normal or diseased. Figure 19 also indicates that spectra from the two BC datasets are the most hard to classify whereas spectra from the OC dataset is most easily classified. All algorithms except *k*-NN achieve accuracies above 90% on the OC dataset.

Fig. 28. Observed accuracy of four different classification algorithms used on all datasets.



Equation 1 from section 3.6.3 was used for determining the true error rates with confidence intervals. Table 4 shows the results after converting them into estimations of the true accuracies instead of error rates. The narrowest intervals were obtained for the OC and MSC datasets. These are the largest datasets and in addition they were the two datasets for which the classification algorithms achieved best observed accuracy values.

Table 4. Estimated true accuracy with 95% confidence intervals.

	Dataset			
	BC1	BC2	OC	MSC
Algorithm				
k-NN	61,0 ± 13,3	67,2 ± 12,3	86,1 ± 4,5	81,5 ± 4,7
LDA	59,3 ± 13,4	67,2 ± 12,3	92,1 ± 3,5	81,1 ± 4,8
SVM	59,3 ± 13,4	67,2 ± 12,3	92,1 ± 3,5	81,5 ± 4,7
RF	67,8 ± 12,8	70,1 ± 12,0	90,9 ± 3,8	79,0 ± 4,9

The test statistics correcting for the variance caused by the CV were calculated for all six pairwise comparisons of classification algorithms for each dataset using equation 2. The corresponding p-values did not show any evidence that one algorithm outperformed the other ones since the lowest obtained p-value was 0.101. The lowest p-value was obtained for the comparison between *k*-NN and SVM in the OC dataset. The comparison of *k*-NN and SVM also in the OC dataset gave the second lowest p-value of 0.110. The correction for multiple testing using a threshold p-value of 0.05 divided by the number of pairwise comparisons is not needed since any significant difference was not found between the algorithms. To find one

algorithm which is most accurate for all different types of datasets an additional correction for the multiple datasets is likewise needed.

5. Discussion

This study compares the predictive performance of the *k*-NN, LDA, SVM and RF classification algorithms when analysed on four different datasets. No conclusions could be made about any algorithm performing better than the rest of the algorithms. Interestingly there was no significant difference between any two algorithms either. The accuracy seems to be more dependent of the datasets than the classification algorithm used. This is because the variation between accuracies of different algorithms is small compared to the difference in accuracies between the datasets. More datasets need to be studied in order to conclude if differences between datasets are due to random variations or if they are dependent on the studied disease. For example, if analysing a larger number of breast cancer and ovarian cancer datasets it would be possible to determine if ovarian cancer is easier to classify.

Some interesting findings have been made on the way to the final results of the classification study. Starting with the pre-processing step, the *KNNimpute* algorithm for estimating missing values is significantly better than the two other analysed methods. Further investigations comparing *KNNimpute* with other complex methods would be interesting in order to see whether the estimations could be improved. Another finding is that the use of permutation tests together with filters during the feature selection process does not affect the predictive performance much. In addition the permutation tests are very time-consuming due to large number of additional computations. However, using it to analyse results could be useful since the permutation tests are performed once instead of 110 times as in the case with nested cross validation (100 times on the training sets and ten times on the learning sets).

During the tuning of parameters some trends were discovered. The random forest parameter *ntree* is not that important, only small differences in error rates are achieved when varying *ntree*. Using small values of *ntree* is a good choice since it saves time and no improved classification performance is obtained when using large values. In further analysis a greatly reduced computational time for RF makes it possible to use more time for optimising parameters of other classification algorithms which are more positively affected by optimisation.

A bit surprisingly, the OC and MSC datasets obtained the best performance when a small number of features were selected. Selecting only five features was the best choice for three of the four classification algorithms when evaluated on the MSC dataset.

It is relatively easy to add new methods or replace existing ones in the scripts used in this study. This makes it possible to continue using the framework and analysing more datasets, or to test other algorithms.

6. Acknowledgements

I want to thank Michal Lysek and Tobias Persson for formulating the idea of this study and for giving me the chance to do my master degree project at MedicWave AB.

I also want to thank the scientific reviewer of this project, Tomas Olofsson who is a senior lecturer at the Department of Engineering Sciences, Signal and Systems Group Uppsala University.

7. References

Ambroise, C. & McLachlan, G.J. 2002, 'Selection bias in gene extraction on the basis of microarray gene-expression data', *PNAS*, vol. 99, no. 10, pp 6562-6566.

Berrar, D., Bradbury, I. & Dubitzky, W. 2006, 'Avoiding model selection bias in small-sample genomic datasets', *Bioinformatics*, vol. 22, no. 10, pp. 1245-1250.

Berrar, D., Granzow, M. & Dubitzky, W. 2007, 'Introduction to genomic and proteomic data analysis' in W. Dubitzky, M. Granzow & D. Berrar (eds), *Fundamentals of Data Mining in Genomics and Proteomics*, Springer, New York, pp. 1-37.

Braga-Neto, U.M. & Dougherty, E.R. 2004, ´Is cross-validation valid for small-sample microarray classification´, *Bioinformatics*, vol. 20, no. 3, pp. 374-380.

Breiman, L. 2001, ´Random forests´, *Machine Learning*, vol. 45, no. 1, pp. 5-32.

Coombes, K.R., Baggerly, K.A. & Morris, J.S. 2007, 'Pre-processing mass spectrometry data' in W. Dubitzky, M. Granzow & D. Berrar (eds), *Fundamentals of Data Mining in Genomics and Proteomics*, Springer, New York, pp. 79-102.

Datta, S. & DePadilla, L.M 2006, 'Feature selection and machine learning with mass spectrometry data for distinguishing cancer and non-cancer samples', *Statistical Methodology*, vol. 3, no. 1, pp. 79-92.

Díaz-Uriarte, R. & Alvarez de Andrés, S. 2006, ´Gene selection and classification of microarray data using random forest´, *BMC Bioinformatics*, vol.7, no. 3, viewed 28 April 2009, http://www.biomedcentral.com/content/pdf/1471-2105-7-3.pdf.

Dudoit, S., Fridlyand, J. & Speed, T.P. 2002, 'Comparison of discrimination methods for the classification of tumors using gene expression data', *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 77-87.

Fung, E.T., Weinberger, S.R., Gavin, E. & Zhang, F. 2005, 'Bioinformatics approaches in clinical proteomics', *Expert Rev. Proteomics*, vol. 2, no. 6, pp. 847-862.

Gentleman, R., Rossini, A.J., Dudoit, S. & Hornik, K. 2003, *The Bioconductor FAQ*, BioConductor, viewed 1 June 2009, http://www.bioconductor.org/docs/faq/.

Gil-Pita, R., Rosa-Zeura, M, Vicen-Bueno, R. & López Ferreras, F. 2007, 'A new algorithm for fast search of the k nearest patterns', Proceedings of the 15th European Signal Processing Conference, Poznan, Poland, pp. 1887-1891.

Glish, G.L. & Vachet, R.W. 2003, 'The basis of mass spectrometry in the twenty-first century', *Nature Reviews Drug Discovery*, vol.2, no. 2, pp. 140-150.

Hauskrecht, M., Pelikan, R., Valko, M. & Lyons-Weiler, J. 2007, 'Feature selection and dimensionality reduction in genomics and proteomics' in W. Dubitzky, M.

Granzow & D. Berrar (eds), *Fundamentals of Data Mining in Genomics and Proteomics*, Springer, New York, pp. 149-172.

Hsu, C.W, Chang, C.C. & Lin, C.J. 2008, *A practical guide to support vector classification*, Department of Computer Science; National Taiwan University, Taipei, viewed 7 April 2009, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Jeffries, N. 2005, 'Algorithms for alignment of mass spectrometry proteomic data', *Bioinformatics*, vol. 21, no. 14, pp. 3066-3073.

Johansson, P. & Häkkinen, J. 2006, 'Improving missing value imputation of microarray data by using spot quality weights', *BMC Bioinformatics*, vol. 7, no. 306, viewed 20 April 2009, http://www.biomedcentral.com/content/pdf/1471-2105-7-306.pdf

Johansson, P. & Ringnér, M. 2007, 'Classification of genomic and proteomic data using support vector machines' in W. Dubitzky, M. Granzow & D. Berrar (eds), *Fundamentals of Data Mining in Genomics and Proteomics*, Springer, New York, pp. 187-202.

Larrañaga, P., Calvo, B., Santana, C., Bielza, C., Galdiano, J., Inza, I., Lozano, A., Armañanzas, R., Santafé, G., Perez, A. & Robles, V. 2006, 'Machine learning in bioinformatics', *Briefings in bioinformatics*, vol. 7, no. 1, pp. 86-112.

Li, X., Gentleman, R., Lu, X., Shi, Q., Iglehart, J.D., Harris, L. & Miron A. 2005, ´SELDI-TOF mass spectrometry protein data´, in R. Gentleman, C. Carey, W. Huber, R. Irizarry & S. Dudoit (eds), *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, Springer, New York, pp. 91-109.

Liaw, A. & Wiener, M. 2002, ´Classification and regression by randomForest´, *R News*, vol. 2, no. 3, pp. 18-22, viewed 29 April 2009, http://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf.

Liu, H., Li, J. & Wong, L. 2002, 'A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns', *Genome Informatics*, no. 13, pp. 51-60.

Lilien, R. H., Farid, H. & Donald, B.R. 2003, 'Probabilistic disease classification of expression-dependent proteomic data from mass spectrometry of human serum', *Journal of Computational Biology*, vol. 10, no. 6, pp. 925-946.

Molinaro, A.M., Simon, R. & Pfieffer, R.M. 2005, ´Prediction error estimation: a comparison of resampling methods´, *Bioinformatics*, vol. 21, no. 15, pp. 3301-3307.

Motulsky, H. 1995, *Intuitive biostatistics*, Oxford university press, New York.

Nadeau, C, & Bengio, Y. 2003, 'Inference for the generalization error', *Machine Learning*, vol. 52, no. 3, pp. 239-281.

Radivojac, P., Obradovic, Z., Dunker, A.K., Vucetic, S. & 2004, ´Feature selection filters based on the permutation test´, in J-F. Boulicaut, F. Esposito, F. Giannotti & D. Pedreschi

(eds), *Machine Learning: ECML 2004*, Springer, Berlin Heidelberg, pp. 334-346.

Radlak, M. & Klempous, R. 2007, 'SELDI-TOF-MS pattern analysis for cancer detection as a base for diagnostic software', in A. Gelbukh & A.F. Kuri Morales (eds.), *MICAI*, Springer-Verlag, Berlin Heidelberg, pp. 1132-1142.

Radmacher, M.D., McShane, L.M. & Simon, R. 2002, ´A paradigm for class prediction using gene expression profiles´, *Journal of Computational Biology*, vol. 9, no. 3, pp. 505-511.

Saeys, Y., Inza, I. & Larrañaga, P. 2007, 'A review of feature selection techniques in bioinformatics', *Bioinformatics*, vol. 23, no. 19, pp. 2507-2517.

Sehgal, M.S.B, Gondal, I. & Dooley, L.S. 2005, 'Collateral missing value imputation: a new robust missing value estimation algorithm for microarray data', *Bioinformatics*, vol. 21, no. 10, pp. 2417-2423.

Simon, R. 2007, 'Resampling strategies for model assessment and selection' in W. Dubitzky, M. Granzow & D. Berrar (eds), *Fundamentals of Data Mining in Genomics and Proteomics*, Springer, New York, pp. 173-186.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. & Altman, R.B. 2001, 'Missing value estimation methods for DNA microarrays', *Bioinformatics*, vol. 17, no.6, pp. 520-525.

Wild, C. & Seber, G. 2000, *Chance encounters: a first course in data analysis and inference*, John Wiley & Sons, New York

All figures included in the report are produced by the author between February and September 2009.