Universiteit Utrecht

Universiteit Utrecht

Bachelor thesis Artificial Intelligence

Heterogeneity in psychiatric disorders:
Using machine learning to predict development of mood disorders in bipolar offspring

Suzan Stempher
3865010

15 ECTS

1st supervisor
Dr. Hugo Schnack

2nd supervisor
Prof. dr. Yoad Winter

July 3, 2017

Abstract

Heterogeneity in psychiatric disorders complicates the application of machine learning techniques in psychiatry. Various causes in the brain can lead to the same mental disorder, partitioning the patient data into several groups. In a classification task, it may be impossible to separate the patients from the rest with a linear boundary. Several methods with potential to deal with heterogeneity in data were discussed and applied to simulated data with a heterogeneous patient group. Support vector machines (SVMs) with a radial-basis kernel and artificial neural networks (ANNs) were able to separate the groups well, using a non-linear separation boundary.

As a proof of principle, these machine learning methods where then used in a regression task performed on real-world data, aiming to predict age based on brain volumes ($n = 501$). Significant improvements compared to the linear method were found, with the best model (ANN) having a mean absolute error of 4.3 years. Finally, classification models were trained on a clinical data set of children of bipolar parents ($n = 140$) to predict development of mood disorders and bipolar disorder in 12 years. The linear and non-linear machine learning approaches performed similarly; this may be caused by a lack of heterogeneity in this particular data set. Still, accuracies ranging from 71 to 77% in predicting mood disorders and 76 to 80% in predicting bipolar disorder were reached. With follow-up research on new data sets, these promising results may be improved in order to apply the prognosis models in practice.

Keywords: machine learning, psychiatry, heterogeneity, bipolar disorder, brain age, support vector machines, neural networks.

Contents

Chapter 1

Introduction

1.1 Machine learning in psychiatry

The field of psychiatry is relatively new to machine learning techniques. The first studies to use pattern recognition were published around 1991 (Van der Knaap et al. [1], Maes et al. [2]), but it took some years before this technique was picked up by other researchers. At the moment, the use of statistical learning methods is a very promising approach for classification tasks. These methods, like support vector machines and especially deep neural networks, are growing in popularity when it comes to applying artificial intelligence.

With the rise of machine learning as a method to construct models which predict and classify automatically, scientists are exploring whether these promising techniques can contribute to the psychiatric field as well. Currently, patients are being diagnosed by professionals that weigh symptoms and use standardized questionnaires and guidelines to come to a conclusion. One goal of applying machine learning to psychiatry is to improve diagnosis by making use of classification models. Ultimately, such models could make diagnosing more accurate and more efficient.

Another goal is to improve prognosis. For many psychiatric disorders, it is essential to start treatment as early as possible. Using machine learning, we may be able to predict future diagnoses and severeness of illness. Patients at high risk for a disease can then be monitored and potentially start with medication to slow down the progression of illness.

Furthermore, research is being conducted towards acquiring more knowledge of the possible presence of biomarkers in the brain underlying the psychiatric disorders. With machine learning, especially applied to neuroimaging, patterns in the brain may be discovered that are hard to find otherwise.

However, these goals are not easy to achieve in the psychiatric field; it is a very challenging domain for statistical learning. Many studies suffer from a relatively low sample size compared to other areas. In well-known machine learning areas like image recognition or artificial intelligence in games, large data sets are usually available. In psychiatry, in contrast, it is costly and difficult to acquire data. Patients and controls willing to participate need to be recruited, interviewed and possibly scanned. For longitudinal studies, follow-ups are required, with the risk of participants dropping out of the study. Combining existing data sets can be problematic, since designs and measurement standards vary between studies.

The true labels of psychiatric data, usually diagnoses defined by specialists, pose another challenge. Due to differences in interpretation, differing guidelines and human error, we deal with a silver standard rather than a gold standard. A percentage of at least 10% mislabeled cases can be assumed when working with diagnoses, depending on the disease (Regier et al. [3]). This is a limiting factor in the maximum accuracy that can be achieved with predictive models. The symptom variables in clinical data are determined by human experts as well and may also suffer from mislabeling.

Lastly, a problem that arises in the domain of psychiatry is the presence of heterogeneity

in data. Different causes or abnormalities in the brain may lead to the same mental disease, subtyping patients of the same illness into subgroups. These subgroups may be positioned in a way that makes straightforward linear separation impossible, as shown in figure 1.1.

1.2 Problem of heterogeneity

The focus in this study will lie on how to deal with heterogeneity in psychiatric data when it comes to classification tasks. Linear methods such as support vector machines and linear regression are often used in psychiatry, as they suffer less from small sample sizes and are relatively easy to interpret. They assume, however, that the classes in the data can be separated linearly. Because of possible heterogeneity in the symptoms or causes of the disorders, this does not have to be the case. The use of models that are capable of separating non-linearly can improve the accuracy on heterogeneous data. It may also help to discover patterns in the heterogeneous nature of the diseases, which are hard to detect by hand. Therefore, in this study, different approaches to handle heterogeneity are investigated to study their potential in improving classification.

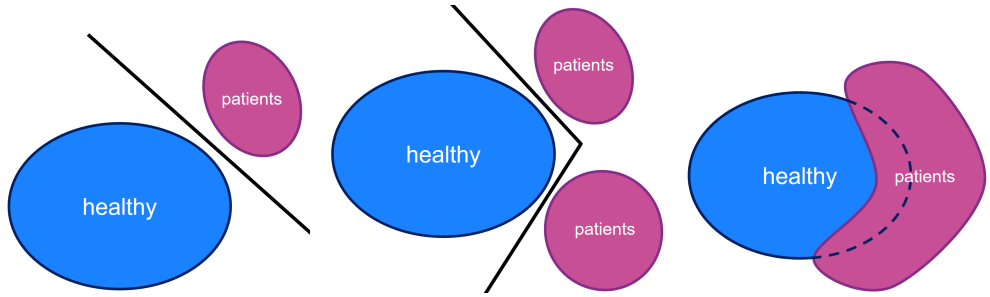


Figure 1.1: Schematic view of the problem of heterogeneity. a) The situation that is often assumed, where the patients and healthy controls are linearly separable. b) Situation with multiple distinct disease groups, where the optimal separation between the groups is non-linear. c) Situation where the patient subgroups overlap with each other and with the healthy controls as well. This scenario is most likely the closest to reality.

1.3 Place in artificial intelligence

Machine learning, a technique part of the broad field of artificial intelligence, is applied to a relevant and practical clinical problem in this thesis. Apart from the use of computer science to construct models, psychological interpretation of the models plays an important part. Also, the balance between requirements like performance, interpretability and the false positive/negative ratio is more delicate than in most machine learning areas in artificial intelligence. Since human data is involved, ethical issues are concerned as well. When is the accuracy high enough to apply a model in practice? How many false positives are permitted and what is the cost of a mistake? Also a few neuroscience topics are touched, especially in chapter 4, where volumes of brain regions are used to train predictive models. So all in all, this study combines issues from the cognitive and the technical side of artificial intelligence.

1.4 Thesis structure

In chapter 2, various machine learning approaches which may be suitable for tackling heterogeneity are described and analyzed in detail. In chapter 3 these methods are applied to 2-dimensional simulated data containing artificially added heterogeneity to test the potential of the approaches. The most promising techniques are thereafter applied to real-world clinical data in chapters 4 and 5. Firstly, clinical brain volume data ($n = 501$) were used to a person's predict age as proof of principle of the proposed methods. Subsequently, the prognosis of children at high familiar risk for bipolar disorder ($n = 140$) was predicted. In each of these two chapters, the training process, results and implications are described in terms of the method, results and discussion sections respectively.

In appendix A, additional tables with results from chapter 5 can be found for reference. Appendix B consists of the user manual for the `R` script `NNSVM` that was written for the purpose of this study and which may be reused in other machine learning studies in the psychiatric domain.

Chapter 2

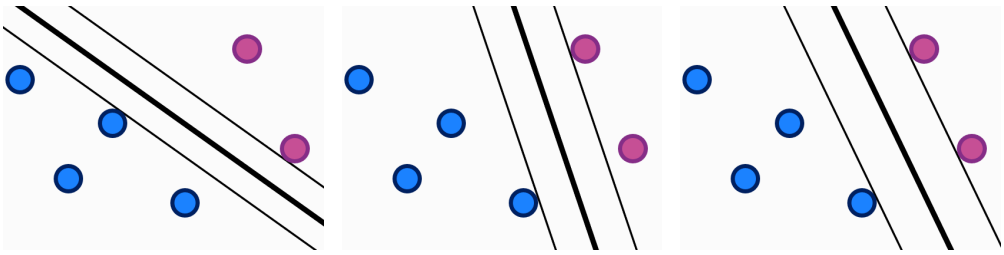
Different approaches

In this chapter, a number of classification approaches are discussed. Support Vector Machines, especially the linear ones, are widely used in psychiatry. Artificial Neural Networks are popular in the general field of machine learning. Normative modeling and Hydra are approaches that were specifically designed with the problem of heterogeneity in mind.

2.1 Support vector machines

Support vector machines (SVMs) are one of the most widely used machine learning technique used in neuroimaging, especially for classifying schizophrenia (Wolfers et al. [4]). The concept was proposed in 1992 (Boser, Guyon, and Vapnik [5]) and finds a decision boundary in feature space, partitioning the space into two groups. Feature space in machine learning is defined by the data columns (characteristics of the samples that are used to train a model). In figure 2.1 three examples of points in 2-dimensional feature space are shown. Each axis represents one variable in the data.

SVMs find a decision boundary in feature space by fitting a hyperplane[1] that has a maximal margin around it containing no data points. Soft-margin SVMs allow some points to lie within the margin under a certain penalty to allow for overlapping data. When a non-linear decision boundary is necessary, the kernel trick can be used to transform the feature space to higher dimensions. Since SVMs are easy to use, do not suffer from local minima and have possibilities for both linear and non-linear classification, they are a popular method for classification tasks.



(a) (b) (c)

Figure 2.1: Three possible linear separations, with their corresponding margins. The width of the margin in (c) is the maximum possible in this example, which is considered optimal.

[1] n-dimensional generalization of a line

2.1.1 Background

In this section, the basics of SVMs, kernels and cross validation are explained. Readers familiar with these concepts may skip to a further section.

The idea of SVMs is based on the concept that a decision boundary should be chosen in such way that the distance to the groups it is trying to separate is maximized. Since data points are usually noisy and unseen data points may deviate from points in the training set, this approach tries to leave as much space as possible between the decision boundary and the groups. This concept is visualized in figure 2.1.

Only in figure (c) from 2.1 does the margin have a maximal width. The line cannot be tilted in any way to increase the distance to the nearest data points. The points touching the decision boundary that determine the margin are called the support vectors. In figure 2.1c, three support vectors are present. The decision boundary is defined by a set of weights, one for each feature in the data. The task is to find a set of weights that maximizes the margin of the decision boundary.

Hard-margin SVM

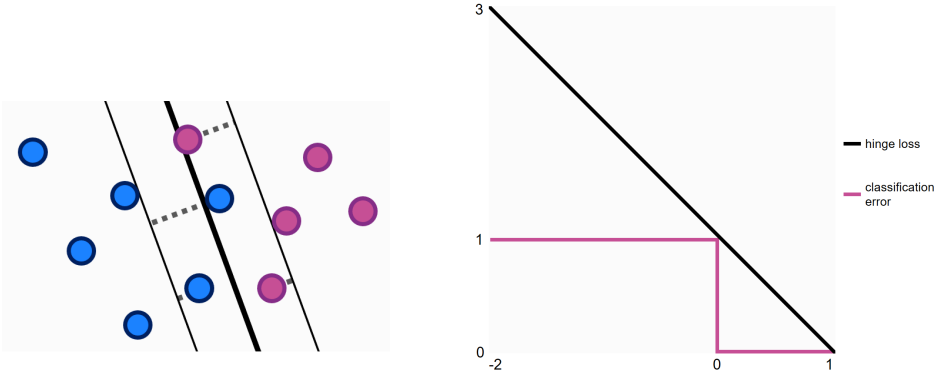
Data points can be described as vectors in d-dimensional feature space. Let $\mathbf{x} \in \mathbb{R}^d$ be a data point vector where d is the amount of features, and let $\mathbf{w} \in \mathbb{R}^d$ be the vector of weights that needs to be learned. The intercept is called b, yielding a signal of $\mathbf{w}^T\mathbf{x} + b$. If y_n is the correct label belonging to a data point x_n, the point is classified correctly when

$$y_n \cdot (\mathbf{w}^T\mathbf{x} + b) > 0.$$

The resulting weight vector is made unique by requiring that the minimum signal of the data points is exactly 1:

$$\min_{n=1,\dots,N} \; y_n \cdot (\mathbf{w}^T\mathbf{x} + b) = 1.$$

With this constraint, the size of the margin of the decision boundary is equal to $\frac{1}{\|\mathbf{w}\|} = \frac{1}{\sqrt{\mathbf{w}^T\mathbf{w}}}$. To maximize the margin, $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized under the constraint of $y_n \cdot (\mathbf{w}^T\mathbf{x} + b) \geq 1$ for $n = 1, ..., N$. The choices for $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ (easy to differentiate) and minimization (common in optimization problems) are for convenience purposes. This minimization problem is convex, allowing for solving with algorithms like gradient descent.



(a) (b)

Figure 2.2: a) A soft-margin SVM, allowing for points lying within the margin and possibly for misclassifications (points lying beyond the decision boundary). The distance of points located within the margin to the margin edges (drawn with dashed lines) is minimized by the algorithm. b) The hinge loss function. The horizontal axis represents the distance of a data point to the decision boundary (positive when located on the correct side, negative otherwise), the vertical axis represents the penalty (black) and the classification error (pink). The margin is 1 in this example.

Soft-margin SVM

The described procedure only works for linearly separable data, while the vast majority of data sets is not. To apply the concept of SVMs to data sets with possible overlap, it needs to be extended and allow for misclassifications under a certain penalty.

In figure 2.2a an example of this concept is shown. There are four points that lie within the margin, and one of them (blue) is classified incorrectly. Figure 2.2b shows the hinge loss function, which penalizes data points for lying in the margin. Points lying outside the margin (on the correct side) get no penalty, and the further away they get from the margin edge closest to their group, the higher the penalty gets. The pink function shows the classification error, which is equal to 1 when a data point lies on the wrong side of the decision boundary and equal to 0 otherwise.

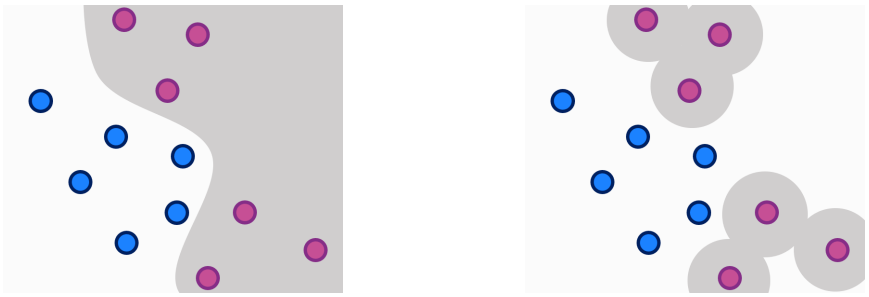
Let there be N data points and let ξ_n be the hinge loss of the n^{th} data point. Now $C \cdot \Sigma_{n=1}^{N} \xi_n$, an estimated in-sample error multiplied by a parameter C, has to be minimized in addition to $\frac{1}{2}\mathbf{w}^T\mathbf{w}$. So for the soft-margin SVM, the optimization problem is:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \cdot \sum_{n=1}^{N} \xi_n \\ \text{under the constraint} \quad & y_n \cdot (\mathbf{w}^T\mathbf{x_n} + b) \geq 1 - \xi_n \\ & \xi_n \geq 0 \qquad\qquad \text{for } n = 1, ..., N \end{aligned}$$

The parameter C controls the degree of regularization. A high C results in a smaller margin, eventually approximating a hard-margin SVM. This reduces the error on the training sample, but risks overfitting. A lower C allows for a larger margin, making the model typically more generalizable, although it increases the error on the train data.

Kernels

The SVMs that were discussed until now are linear models, having a decision boundary in the form of a linear hyperplane. Linear SVMs are often used because they are easy to implement and robust to overfitting with their limited flexibility in fitting to the data. But if the data is truly not linearly separable, models having more freedom to fit to the data are preferred. Using a kernel function, it is possible to transform feature space to higher dimensions, making the data points easier to separate. A linear hyperplane will still be used to separate the groups, but it results in a non-linear decision boundary in the standard feature space.



(a) Low γ (b) High γ

Figure 2.3: Schematic view of SVM classifiers with the RBF kernel for two different values of γ. a) With a lower value of gamma, the classifier constructs a more generalizable model. The lower the value, the more rigid the boundary becomes. Values extremely close to zero approximate a linear SVM, making the use of the kernel redundant. b) Higher values make the model fit to the data more precisely, where very high values result in 'islands' around the points of one group, usually a case of strong overfitting.

A commonly used non-linear kernel is the radial basis function kernel $K(x, x') = \exp(-\gamma\|x - x'\|^2)$. Instead of just the cost parameter C, a second parameter γ has to be tuned. The effect that γ has on a trained model is demonstrated in figure 2.3.

2.1.2 Double nested cross validation

K-fold cross validation (CV) is a common method for estimating the performance of a model. The train set is partitioned into K equal subsets (called folds), and a model is trained K times leaving out each of the folds once for validation (the left out folds). The process is shown schematically in figure 2.4.

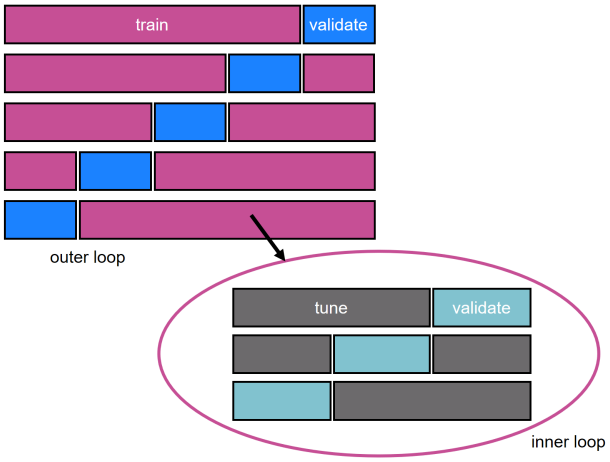


Figure 2.4: Schematic figure of double nested cross validation, with five outer loops and three inner loops. A model is trained on the pink fold and validated on the blue left out fold five times. During each of those training processes, another loop is performed on the pink data fold to train with different parameters on the tune set and validate them on the validation set.

Since different values of parameters influence the performance of the SVMs, tuning the parameters is required. This can be done by training models with varying parameter values on a train set and selecting the best performer on the test set. However, choosing parameters based on the same data that the models are validated on may lead to overfitting. Therefore parameters need to be tuned in a double nested cross validation loop. In each of the performance measuring loops (called the 'outer loops'), a new cross validation process is performed on the train fold to make sure the parameter performance is not measured on the same data as the overall model performance.

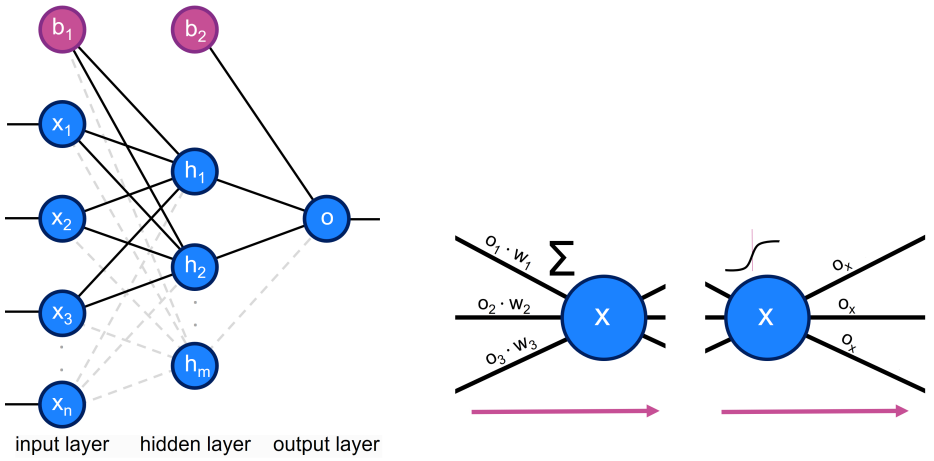
2.1.3 Conclusion

Linear support vector machines are models that are relatively easy to interpret, as the resulting weight vector shows the precise linear relation between the input features and the predicted outcome. Their robustness allows them to be used in situations with many features. Non-linear kernels like the radial-basis function can be used in non-linear situations, but will not result in an interpretable weight vector. They certainly have the potential to find heterogeneous structure in the data however, and are therefore a good non-linear candidate to compare to the linear SVMs.

2.2 Artificial neural networks

Artificial neural networks are a well-known approach in machine learning. They gained wide popularity in 1974 when the backpropagation algorithm was introduced (Werbos [6]). By sending the error on the output back into the network, networks were now able to contain multiple hidden nodes which enabled them to learn non-linear functions. Nowadays, neural networks are still growing in popularity under the term of deep learning, which denotes networks with multiple layers to model different abstraction levels of learned features.

Neural networks are relatively hard to interpret compared to other learning algorithms. After the training process, all we get is a set of weights defining the strength of each connection in the model. These connections do not have a one-to-one correspondence to the features, like the weights in Support Vector Machines do. This makes it hard to determine which features are important to the model. The model's performance can be measured easily, but how that performance is achieved is a more difficult question.



(a) Structure of a neural network with n inputs, one hidden layer with m nodes and a single output node.

(b) Input to a node.

(c) Output from a node.

Figure 2.5: A feed-forward neural network. a) Global architecture of a network. b) The input to each node (except for the input nodes) is calculated by multiplying each of the outputs of the nodes in the preceding layer with their corresponding weights and taking the sum. c) The output from a node is calculated by applying the activation function to the node's input. The output is sent to each connection going away from the node.

2.2.1 Background

The basics and mechanics of feed-forward neural networks are described in this section.

The common architecture of a feed-forward network is shown in figure 2.5a. The network has an input layer of n nodes $(x_1, ..., x_n)$ and may have one or more hidden layers with nodes (here one hidden layer with m nodes $(h_1, ..., h_m)$ is present). The number of hidden nodes is usually lower than the number of input nodes and in binary classification there is typically one output node (node o in the figure). Each node, except for the output node, is connected to all nodes in the next layer. Each connection between two nodes has a weight. The model is trained by learning the values of all the weights in the network.

Neural networks learn a function that maps values $x_1, .., x_n$ $(n \in \mathbf{N})$ to one (or more) output value o. For given inputs, the output of a network can be calculated by sending all inputs and outputs of nodes from left to right through the network, hence the term feed-forward. The bias nodes of each layer (here b_1 and b_2) do not have inputs, but send out a

constant output of a single value. The nodes $x_1, ..., x_n$ get the feature values as input and output these unchanged.

All the hidden nodes take the sum of all the outputs of preceding nodes multiplied by their weights ($o_i \cdot w_i$ in figure 2.5a) and apply a user defined activation function over it to create an output value in the interval $(0, 1)$ (though other intervals are possible). The sigmoid or hyperbolic functions are commonly used as the activation function. The output nodes may use the activation function as well, or just produce linear output in case of regression.

Before training, initial weights are chosen randomly or can be defined by the user. The train examples are fed to the network, and the network output is compared to the correct label. An error function is used, commonly $\frac{1}{2}(y - y')^2$ for true label y and predicted value y'. The errors are propagated back to the network from right to left, updating the weights in the opposite direction of the error derivative. This learning process is called backpropagation (Werbos [6]).

The feeding of the training examples and updating of weights continues until the stopping criterion is met. Examples of stopping criteria are a maximum amount of iterations or a minimum decrease of the error.

2.2.2 Conclusion

In the psychiatry domain, not only the output of a model but also an understanding of why it works and which features are informative is essential. We do not yet have a profound understanding of the biomarkers underlying various mental diseases. A disadvantage of neural networks is that they are relatively hard to interpret. They do not result in a vector of weights which shows how the increase of a feature influences the output, like in linear SVMs. Neural networks also have a tendency to suffer from overfitting. The number of features and hidden layers used should be low enough compared to the amount of samples to get a reasonable convergence. However, its high complexity grants the freedom to find more complex relations in the data that are hard or impossible to detect by linear learning algorithms.

Though neural networks are harder to interpret, their great flexibility may be what is necessary to find heterogeneous patterns in data. For data sets with a reasonable sample size and number of features, they certainly have potential.

2.3 Hydra

An approach that was invented to deal with heterogeneity in data was presented by Varol, Sotiras, and Davatzikos [7]. SVMs were extended to a more general framework in order to do binary classification and and find subtypes in the patient data simultaneously. The objective is not only to distinguish patients from the healthy controls, but also to model the heterogeneous subgroups in the patient cohort.

2.3.1 General approach

The authors propose a non-linear semi-supervised machine learning algorithm called HYDRA, which combines multiple SVM classifiers to create a convex polytope[2] that separates the healthy controls from the heterogeneous group of patients. The dimensions of heterogeneity can be determined by varying the number of estimated hyperplanes.

The article explains that a regular linear SVM is capable of separating two homogeneous groups, assuming there is a single pattern that distinguishes them. Since the real-world data is likely to be heterogeneous instead, this will lead to a very small margin. Having more freedom to fit to the data, a non-linear kernel may fix this. However, the use of such kernels will not result in the identification of different subgroups in the disease that underlie the

[2] the n-dimensional generalization of a polygon

heterogeneity. Therefore, according to the authors, single linear SVMs will not suffice if we want to subtype the disorder.

Hydra considers all sets of K hyperplanes to use for separating the classes. The requirement is that every hyperplane classifies all the members of the positive class (the homogeneous class, corresponding to the healthy samples) and at least one negative (heterogeneous class) sample correctly. Every two negative samples that have been assigned to the same hyperplane are considered to be part of the same subgroup, augmenting the problem to a clustering task. Figure 2.6a shows a schematic example of Hydra using two hyperplanes to seperate the heterogeneous patient group from the healthy subjects.

The aim is to maximize the average margin of all involved SVMs, as opposed to just the margin of a single SVM. After training the classifier, we can predict the class of a new sample $\mathbf{x}$ by taking the sign of the minimum of the prediction scores for all classifiers:

$$y^* = sign(\min_j \mathbf{w}_j^T \mathbf{x} + b_j)$$

Since positive samples are classified as such by every hyperplane and negative samples are assigned to the hyperplane with the lowest prediction score, taking the minimum will yield desirable classifications for each sample. Multiple runs with different initializations of the algorithm are necessary due to local minimums, arising from the non-convex nature of the problem.

2.3.2 Application

In the article, the Hydra algorithm has been tested on both simulated and real-world data. Using Hydra on the simulated data, a significant performance improvement was found when using three hyperplanes instead of one. For higher numbers of planes, the performance decreased very slightly, which suggests that Hydra discovered correctly that the data consisted of three subgroups, according to the authors.

On the real-world data ($n = 300$), using Hydra did not result in significant performance improvement compared to fitting a single hyperplane. On the other hand, the aim of increasing the margin was achieved which suggests the Hydra model was more robust. The article states that this implicates that Hydra had successfully found heterogeneous structures in the data.

While the Hydra method might help to identify those patterns, the reported results on the clinical data did not show improvement over regular linear SVMs. High performances (AUC) were achieved on a decent sample size, but the classifier should be tested on an independent test sample to see whether the high performance and increased margin will be maintained. An independent sample may contain more or different degrees of heterogeneity than the set used in the article.

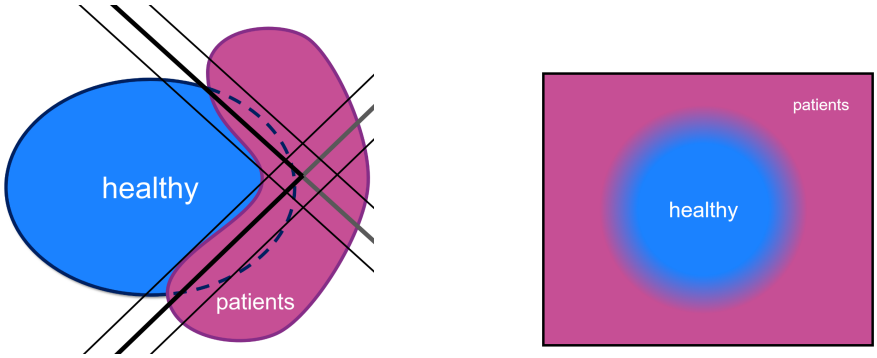
The authors used the Rand index[3] to look for the optimal number of hyperplanes. The highest stability in clustering was found when using three planes, suggesting that the patient group consisted of three subgroups. For higher numbers, these groups were only divided into smaller clusters, implying a hierarchy. Backing the claim of three subgroups, distinct brain patterns for these groups were found in the MRI, suggesting they represent different variations of the disease.

2.3.3 Conclusion

The use of the Hydra algorithm has not shown an increase of the performance on clinical data. It needs more investigation and should be applied to varying sorts of data sets before a conclusion can be drawn. Though the additional information on group structure in the data is beneficial, a disadvantage is that the value of K for the amount of hyperplanes must be

[3]The adjusted Rand index gives the similarity between different data clusterings. A high index means that the variance in clusterings was low, yielding a more stable clustering.

chosen beforehand. It should be guessed by looking at the data, or various values must be tried adding another tuning task with dangers of overfitting.



(a) Hydra (b) Normative modeling

Figure 2.6: Schematic view of normative modeling and Hydra approach. a) Hydra combines multiple linear SVMs to better accommodate to heterogeneity. Each patient can be assigned to the closest hyperplane to obtain structure in the patient group. b) With normative modeling, by default the healthy cohort will be modeled. Outliers to this model are considered patients.

2.4 Normative modeling

To handle heterogeneity in clinical groups, Marquand et al. [8] propose a modeling which rejects the classical notion of case and control groups and judges all individuals against the same scale. The approach aims for a mapping of variation within the whole sample and treats outliers on the scale as pathological as patients.

The authors argue that the classic, widely used case-control approach is not ideal in the domain of psychiatry because of the heterogeneity in the data. Since biological tests for diagnosing mental diseases are not available in this domain, clinical classification is used where patients are classified according to their deviations from patterns in healthy subjects. The case-control approach assumes two well-defined, distinguishable groups which might in fact not exist in many psychiatric disorders. In practice, groups overlap, are composed of subgroups with different properties or are completely diffuse (fig 1.1).

2.4.1 General approach

The proposed normative modeling approach can be applied to a study population consisting of both healthy and clinical subjects, or healthy subjects entirely. Having disease symptoms should correspond to having extreme values within the distribution on an individual level.

The first step of composing a normative model is using clinical features to predict brain structure or functions, using Gaussian process regression. Measures of confidence for these prediction are computed, which can be used to determine how well a subject fits the norm. The predicted norm shows the typical trajectory while the measures of confidence translate to contour lines around it. Subsequently, a normative probability map is computed for each individual, in which for every brain region the deviation from the normative model is determined. Every subject is then summarized by its extreme values, namely those that differ most from the normative trajectory. The computational details can be found in the article as well as a visual overview of the described approach (fig 2, page 554).

Figure 2.6 shows a very schematic view of the concept of normative modeling. The population is modeled such that the healthy people fall on the typical trajectory (blue area) and the outliers from this norm are classified as patients (pink area).

2.4.2 Application

To study the normative approach in practice, the authors used a large group of healthy samples ($n = 288$) who completed a functional MRI task and were measured on having clinical symptoms. In the article, the means of the extreme values of each subject are compared to their symptoms. Having higher extreme values corresponds to having a larger deviation from the normative model according to the theory. Most subjects fitted the normative model well, as expected from a healthy sample group. Some of those, however, did have high symptoms but still fitted the norm.

It can be debated whether the subjects which had high symptoms should actually have deviated from the model and classified as belonging to the disease group, or whether they are examples of heterogeneity in the healthy group. The results show some interesting structure in the group, but no overall correlation between deviating from the normal spectrum and having symptoms was found; this correlation was only present in subgroups of the data.

2.4.3 Conclusion

The proposed notion of normative modeling is a very interesting view and should be explored further. The results were not completely convincing yet, but the approach is a good step in the direction of handling heterogeneity. The concept that case and controls groups are not well-defined and may exist as a spectrum should be kept in mind. Though for classification and diagnosing purposes, our goal remains to decide between being healthy and heaving a disease. Normative modeling can be used to reveal structure in a healthy or mixed population and to find the degree of heterogeneity, but the shown results are not stable enough to use for classification alone.

2.5 Conclusion

All four discussed methods are able to deal with heterogeneity in a certain way. SVMs with RBF kernels and neural networks are capable of producing curved decision boundaries, Hydra classifies by placing several linear hyperplanes around the healthy class and normative modeling lets go off the notion of two distinguishable groups altogether and applies a norm based on the healthy group to the whole sample population. While the latter approach introduces a promising concept for modeling heterogeneity, it does not hold much potential for constructing classifiers yet. Since a goal of this study is to improve the accuracies of the classification of mental disorders, the other three methods will be assessed further in the next chapter to evaluate their potential on heterogeneous data.

Chapter 3

Simulated data

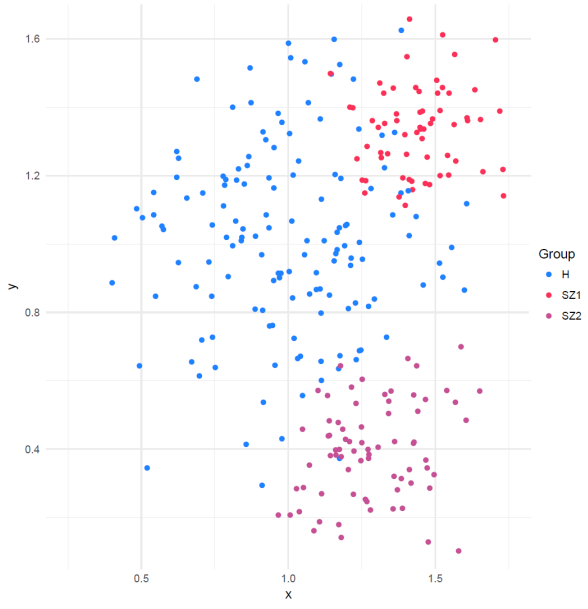
Before subjecting the proposed classification approaches to real-world clinical data, a generated data set was used to simulate the idea of heterogeneity and to discover whether the non-linear methods are capable to use their freedom to fit to these heterogeneous groups.

3.1 Data set

A data set with two Gaussian distributed features was generated, containing 200 healthy control samples ('H'), and 200 (schizophrenia) patients. The patient group was divided into two subgroups ('SZ1' and 'SZ2') to simulate heterogeneity and to make the ideal separation boundary non-linear. The patients were labeled 1 regardless of which subgroup they belong to, while the healthy subjects were given label -1. The data was randomly divided into a train set (70%) and a test set (30%), containing respectively 280 and 120 samples.

	mean		sd	
	x	y	x	y
H	1.00	1.00	0.28	0.28
SZ1	1.45	0.15	0.15	0.15
SZ2	1.30	0.40	0.15	0.15

(a) Means and variances of each group in the data (both train and test), for Gaussian distributed variables x and y.



(b) Simulated train data set ($H = 143$, $SZ1 = 64$, $SZ2 = 73$)

Figure 3.1: Generated data set.

The data set is visualized in figure 3.1. The small overlap between the heterogeneous subgroups is exaggerated; groups in real-world clinical data are likely to overlap more. The healthy group has a higher variance than each patient group. The means and variances used to generate the data can be found in table 3.1a.

In the next sections, linear SVMs, RBF SVMs, neural networks and simulation of Hydra will be trained and validated on this data to assess their potential on heterogeneous data and visualize the differences between their decision boundaries. Subsequently the resulting models will be tested on the test set that was kept aside to make a final comparison.

3.2 Method

The `LibSVM` library for `R` (called `e1071`) was used to train SVMs with linear and RBF kernels. For the neural networks, the library `neuralnet` was used. Functionality for cross validation and validation of the models was added. All models were trained with 11-fold cross validation, with an additional double nested 5-fold loop for the SVMs to tune the parameters C and γ (of which the latter only applies to RBF kernels). The odd number 11 was chosen to possibly allow for taking medians[1] instead of means. The parameter ranges for tuning of both C and γ were powers of 2, ranging from 2^{-3} to 2^3. The SVM with the best performing parameters in the inner loop was selected to predict the label of the left out fold in the outer loop.

The neural networks were feed-forward networks with a single hidden layer containing two hidden nodes. The sigmoid function $\theta(s) = \frac{1}{1+e^{-s}}$ was used as the activation function, applied to the output of all the hidden layer node except for the output node. Resilient back-propagation with weight backtracking (Riedmiller and Braun [9]) was used, as implemented in the library `neuralnet`. The start weights were randomly initialized by the algorithm in every repetition. The threshold of the partial derivatives of the error function was set at 0.01 as a stopping criterion, which means the algorithm kept training until the error of the model did not reduce with more than 1% at every iteration step.

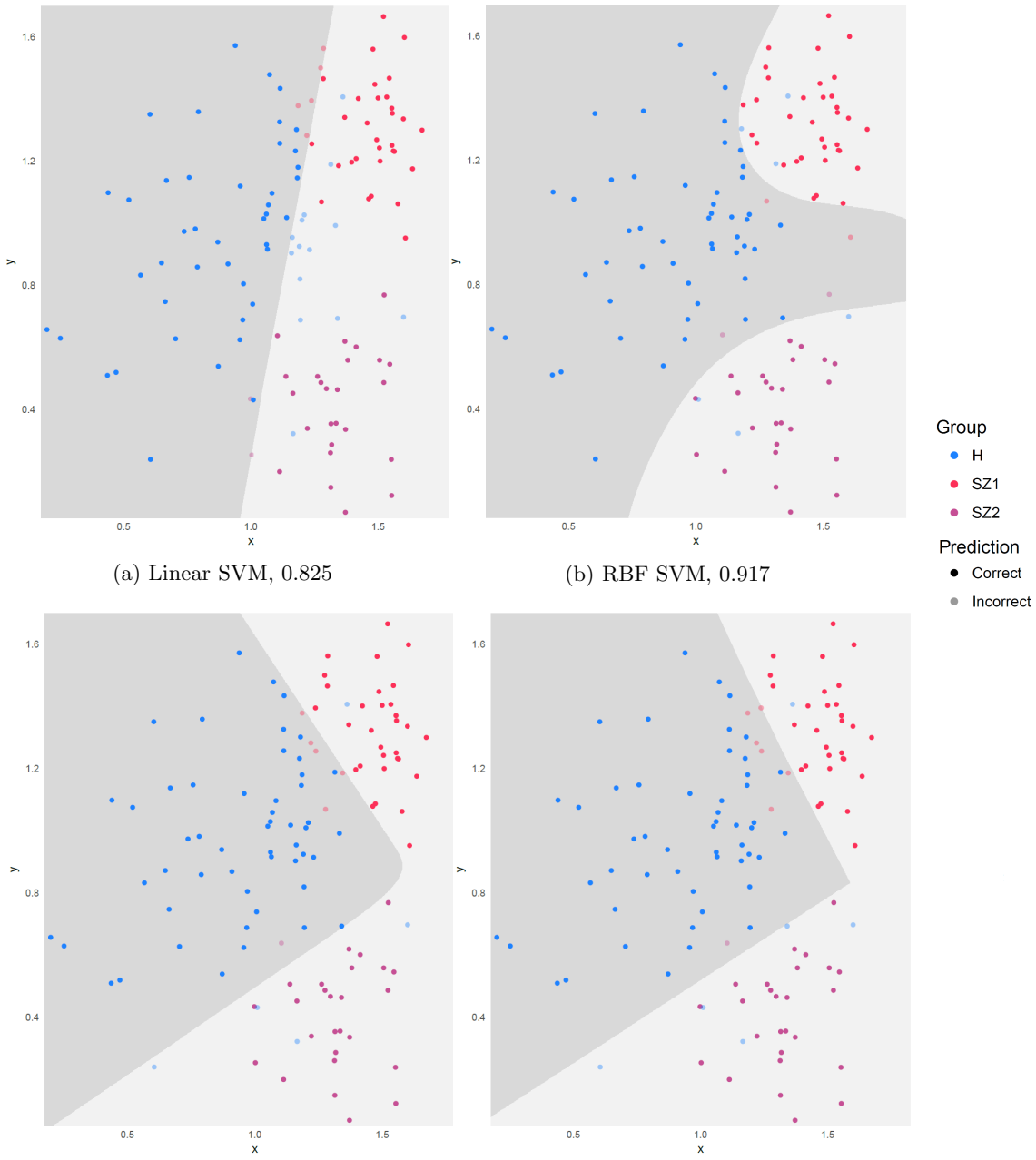
Additionally, two linear SVMs were trained, where each was given all the healthy subject data and only one of the patient subgroups to simulate the idea of Hydra. Each of these two SVMs was trained using 11-fold cross validation. The resulting two hyperplanes were then combined to one model by classifying all points that were labeled 1 by at least one of them as belonging to the patients. The results may differ from a real Hydra implementation, but will likely be similar given the distinct separation between the patient subgroups in this particular data set.

Since cross validation does not result in a single ready to use model, model selection was carried out looking at the 11 trained models in the cross validation process and picking the one having the median (middle scoring) performance on its left out fold. Considering that the best performance usually is due to a stroke of luck on the validation data, the median performer is a more robust choice. Therefore the median model of each approach was selected to be evaluated on the test set.

3.3 Results

The validation accuracy (mean of the accuracies on the left out CV folds) of the linear SVMs was 0.843 (SD=0.06). The model with the median performance had an accuracy of 0.840 on its left out fold and 0.825 on the test data set, with parameter setting $C = 2$. The non-linear RBF SVMs achieved a validation accuracy of 0.935 (SD=0.05). Their median SVM scored 0.960 on its left out fold and 0.917 on the test set, with parameters $C = 0.5$ and $\gamma = 8$. The parameter choices that were made in the tuning process were fairly steady among the 11 folds. During the training of the linear SVMs, $C = 2$ was chosen 7 times and in the RBF SVMs $C = 0.5$ was chosen 5 times and $\gamma = 8$ in all 11 models.

[1] The median value of data is the central value, located in the middle when the values are sorted. Half of the values are smaller than the median, and half of the values are greater.



(a) Linear SVM, 0.825

(b) RBF SVM, 0.917

(c) ANN, 0.908

(d) Hydra, 0.891

Figure 3.2: Simulated test data points with fitted decision boundary of the four attempted approaches. The accuracies of the shown models on the test data are shown. The dark gray areas were labeled -1 by the models, the light gray areas 1. The non-transparent points were classified correctly.

The Hydra simulation model achieved an accuracy of 0.891 on the test set. The validation accuracies of the two individual (median) Hydra SVMs were 0.900 and 0.989, trained on SZ1 and SZ2 respectively. The extremely high accuracy on the SZ2 group shows shows that those patients were easy to separate from the healthy subjects, which is also visible in the train data (fig 3.1b). The neural networks had a validation accuracy of 0.911 (SD=0.06), with the median network (shown in figure 3.3) performing 0.920 on its left out fold and 0.908 on the test set.

Figure 3.2 visualizes the decision boundaries of the discussed median models on the test set. The one linear model (a) lacked freedom to to fit to the heterogeneity present in the patient group. In order to classify most of the patients correctly, many healthy subjects were classified as patients as well. In contrast, the RBF SVM (b) was capable of fitting directly to the patient group. It formed two islands, leaving space between the two patients groups, unlike the neural network (ANN) and Hydra simulation did ((c), (d)). The close fit to the patient group of the RBF SVM was most likely caused by the high γ value ($\gamma = 8$) that was chosen in the inner CV loops. The decision boundary of the neural network was similar to that of the Hydra simulation, with a slightly different slope and an added curve. The difference in slopes made the ANN classify two more data points correctly (one healthy and one SZ1 subject).

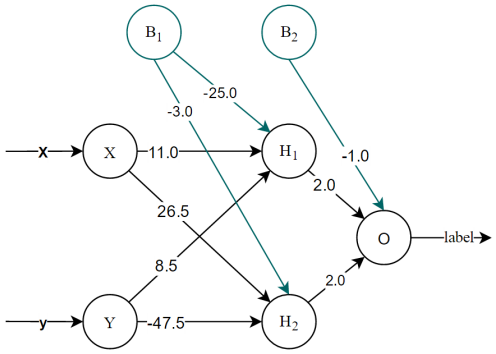
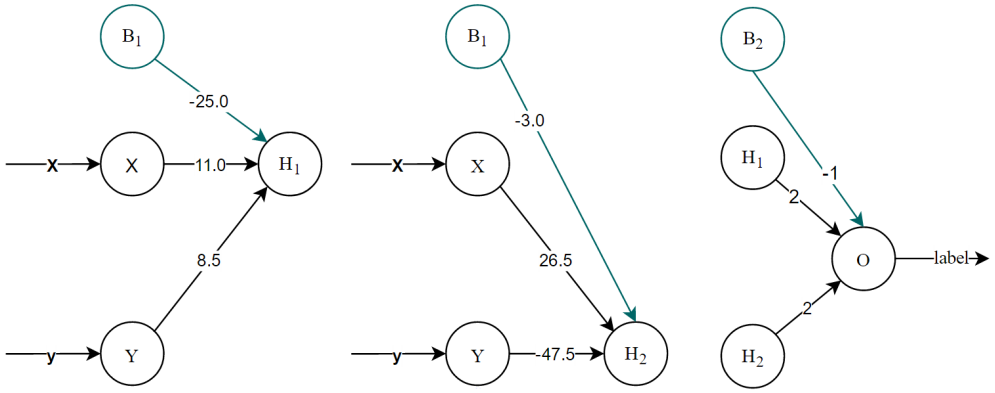


Figure 3.3: Network corresponding to the model presented in figure 3.2c. The weights were rounded to the nearest half for convenience.

3.4 Interpretation of a 2-input network

As discussed earlier, the interpretation of a classifier is important in psychiatry. In this section, the neural network in figure 3.3 will be analyzed to get an impression of how the network and its weights form the decision boundary shown in figure 3.2c. In order to do this, the network was split into three parts (fig 3.4): a) the first layer with its connections to the first hidden node, b) the first layer with its connections to the second hidden node and c) the hidden layer with all connections to the output node. The equations that are produced by nodes H_1 and H_2 and how they are combined in the output node O will be examined.

Every hidden and output node (H_1, H_2 and O) takes the sum of its inputs multiplied by their corresponding weights. The hidden nodes also apply the sigmoid activation function $\theta(s) = \frac{1}{1+e^(-s)}$ to scale the input sum to a value in the interval $(0, 1)$.



(a) $H_1 = \theta(11x + 8.5y - 25)$ (b) $H_2 = \theta(26.5x - 47.5y - 3)$ (c) $O = 2H_1 + 2H_2 - 1$. The label is $+1$ when $H_1 > 0.5$ or $H_2 > 0.5$.

Figure 3.4: The network in figure 3.3 split into three parts. a) All nodes with connections to node H_1. b) All nodes with connections to node H_2. c) All nodes and connections directly linked to the output node O. This shows how the output of a) and b) are combined into the final output.

Nodes H_1 and H_2 both compute a function with two variables as can be seen in figure 3.4a and 3.4b. The outputs of these functions are then combined in figure 3.4c. In this particular example, both the outputs of H_1 and H_2 are weighted equally with weights 2, and a bias of -1 is added. The labels H_1, H_2 and O will from now on denote the *output values* of the corresponding nodes, giving rise to the following formula for the total output O:

$$O = 2H_1 + 2H_2 - 1 \tag{3.1}$$

The predicted classification label will be $sign(O)$, with the decision boundary lying at $O = 0$. What values of H_1 and H_2 will result in a classification of $+1$? If we equate O (3.1) to zero and then rewrite the formula, we find the equation

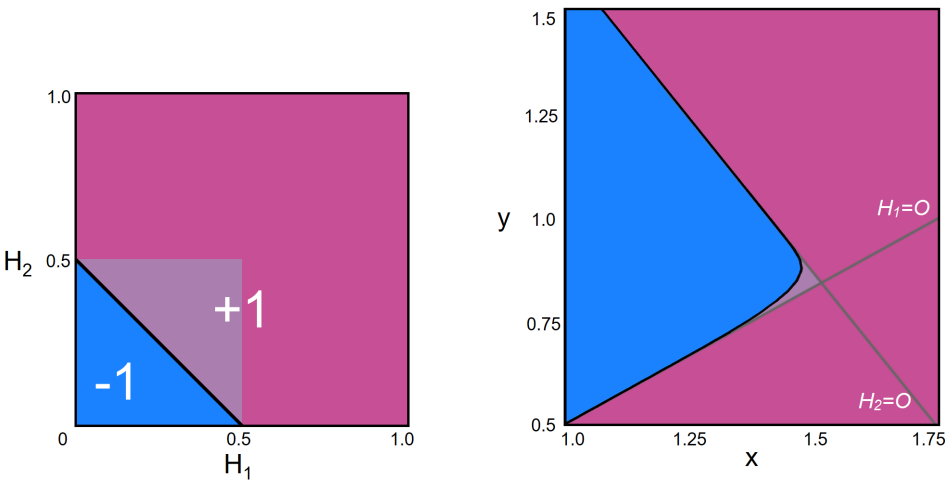
$$H_2 = \frac{1}{2} - H_1 \tag{3.2}$$

which is shown in figure 3.5a. Points in the purple area denote a positive output O, since $H_2 > \frac{1}{2} - H_1$ and therefore $2H_1 + 2H_2 - 1 = O > 0$, and points in the blue area get a negative output value analogously. When *either* H_1 or H_2 is greater than 0.5, the predicted label will always be positive as shown in the bright purple area. This means that the network displayed in 3.4c represents the OR function, on the assumption that an output of 0.5 or higher for a hidden node encodes for a positive label.

Thus for the hidden node outputs H_1 and H_2, the value 0.5 is a threshold defining the decision boundary:

$$H_1 = \theta(11x + 8.5y - 25) = 0.5 \tag{3.3}$$
$$H_2 = \theta(26.5x - 47.5y - 3) = 0.5 \tag{3.4}$$

The lines corresponding to these equations have been plotted in figure 3.5b. As expected, these lines are the main components of the decision boundary. Again, in the purple areas $H_2 > \frac{1}{2} - H_1$ holds, while in the bright purple area additionally $H_1 > 0.5 \ \vee \ H_2 > 0.5$ holds. The curve between these lines in the true decision boundary (black curve in fig 3.5a) is determined by values of H_1 and H_2 that are both smaller than 0.5 but still lie above the decision boundary (the dim purple area in fig 3.5b).



(a) $H_2 > \frac{1}{2} - H_1$ yields label +1, otherwise we get -1. When $H_1 > 0.5$ or $H_2 > 0.5$, label +1 is guaranteed.

(b) The decision boundary $O = 0$ (black), which is composed by the lines $H_1 = 0.5$ and $H_2 = 0.5$ (gray).

Figure 3.5: Relation between the hidden node outputs H_1 and H_2 and the decision boundary.

So in conclusion, the network with two hidden nodes has learned two separating lines, which were combined into one non-linear decision boundary by the weights on connections to the output node. Parallels can be drawn between this process and the Hydra method described in section 2.3, which also works with decision boundaries that are a combination of linear hyperplanes. An advantage of the neural network approach over Hydra is that the number of hyperplanes does not have to be selected in advance: if the user provides enough hidden nodes for the network to find a good fit to the data but not too many to prevent overfitting, neural networks will find the optimal boundary themselves. Also, multiple bends in different directions are possible, whereas Hydra is only capable of constructing a convex polytope around the healthy controls to separate them from the rest.

3.5 Discussion

On both the train data (CV) and the test data set, the RBF SVM had the highest accuracy. The ANN model came as a close second. It is however impossible to determine which method is the most suitable based only on this 2-feature data. The results do show that improvement of performance can be achieved by using non-linear methods on heterogeneous data. The proposed approaches were able to fit to non-linear patterns in the data.

All the methods (except for the Hydra simulation that was not assessed with CV) performed poorer on the test set than during cross validation. This may be caused by some remarkable differences in the random placement of the data points. In the train data, several healthy subjects were placed between the two patient subgroups, which did hardly occur in the test set. Additionally, the SZ2 group happened to be separated very well from the healthy controls in the train set (as was pointed out by the extremely high accuracy of the SZ2 SVM in Hydra), while this was not the case for the test set. Furthermore, due to small dependencies in the cross validation process, performances tend to drop slightly on test sets.

The difference between decision boundary of the RBF SVM and the other boundaries is the most profound. The RBF model isolated the two patient groups, creating three areas in the (visible) space, while the other methods divided the space into two areas and kept a connection between the patient groups. It is hard to say which approach is preferable, especially since this a simulated case. It depends on the expected amount of healthy subjects lying between the patient groups and on the interpretation of the feature variables. The RBF

SVM seems to have the most fitting freedom in this case. Adding more hidden nodes to the ANN will allow it to fit more degrees of non-linearity, but again at risk of overfitting.

It should be noted that the notion of heterogeneity present in the data set was strongly idealized. In real-world clinical data, subgroups may exist but will not be as distinct and overlap more with both each other and the healthy subjects. It is plausible that non-linear patterns caused by heterogeneity are present in real-world data, but most likely not in the form of easy identifiable groups.

3.5.1 Conclusion

On a simplified simulated data set with a heterogeneous patient group, the use of non-linear methods in a binary classification task improved the accuracy considerably (from roughly 83% to 91%). The best performer was the RBF SVM, which also showed the most fitting freedom in its resulting model. Hydra, in contrast, did not outperform any of the other non-linear methods. The neural networks were able to construct a similar model with a relatively small amount of nodes. Furthermore, Hydra produces rather unnatural decision boundaries with sharp cuts while the other methods do not suffer from this.

The potential of RBF SVMs and neural networks as possible solutions for the heterogeneity problem in clinical data will therefore be investigated further on real-world data in the next chapters.

Chapter 4

Brain volume data

4.1 Introduction

As a proof of principle of the proposed methods, a clinical data set was used to evaluate whether these approaches are feasible and may yield promising performances on real-world data. In a regression task, a set of brain volumes was used to predict a person's age. A similar task was performed by Schnack et al. [10], using neuroimaging data (MRI scans) to construct a brain age model based on healthy subjects. This model was then applied to schizophrenia patients, showing that these patients have older brains (3.36 years on average) and also suffer from accelerated brain aging.

In this study, brain volume data was used instead of the MRI data to see if models based on a few features (instead of the thousands present in neuroimaging) are able to achieve similar performance in predicting age. The volume of the brain is known to decrease with age. The process of gray matter loss starts at early adulthood, while shrinkage of white matter areas typically start around middle age (Fotenos et al. [11]). Accordingly, a set of the volumes of these brain areas may be informative enough to perform an age regression task.

4.2 Method

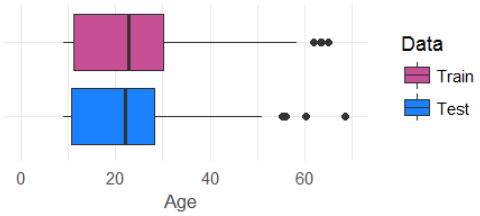
4.2.1 Data

The brain volume data set describes the volumes of brain areas in healthy subjects ($n = 503$) aged between 9 and 68 years old, discriminating between big brain, small brain (both white and gray matter) and ventricles. Additionally, sex and total iq were available as well. All the variables that were present in the data set are presented in table 4.1a, along with the abbreviations that will be used further on in this chapter.

The volume data was normalized using z-transformation, which consists of subtracting the mean and dividing by the standard deviation of the variables to get means that are approximately zero and standard deviations of one. The iq was transformed by $(iq - 100)/15$ and age by $(age - 15)/55$). The data set was then randomly split into a train set (n=400) and a test set (n=103), ensuring that the sets had similar distributions for age (figure 4.1b).

ic	intra-skull volume (intracranium)
tb	total brain volume
bb	big brain (cerebrum) volume
bbgm	big brain gray matter volume
bbwm	big brain white matter volume
cb	small brain (cerebellum) volume
cbgm	small brain gray matter volume
cbwm	small brain white matter volume
vlat	lateral ventricle volume
v3	third ventricle volume
age	age in years
sex	0 female, 1 male
iq	total iq

(a) Features in the volume data set. All volumes are in cm^3.



(b) Boxplots of age variable in train data and test data sets.

Figure 4.1: Features and distribution of age in the data set.

4.2.2 Training

To compare the different algorithms, linear SVMs (Lin), RBF SVMs (RBF) and neural networks (ANN) were trained on the volume data set. 11 identical runs were performed, each consisting of an 11-fold cross validation procedure. This resulted in models consisting of 11 sets of 11 CV models, making a total of 121 models per algorithm (Lin, RBF, ANN).

The linear and RBF SVMs were trained with parameter ranges of 0.001, 0.0025, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 2, ..., 10 for both C and γ. Due to random tie breaking in the inner cross validation loop (where the parameters are tuned), identical runs of the algorithm do not necessarily have to result in identical performances, though the variation is expected to be less in the SVMs than in the neural networks that work with random initialization.

The neural networks were feed-forward networks using resilient backpropagation with weight backtracking. They had one hidden layer with two nodes and a threshold of 0.0005 as a stopping criterion (or a maximum of 400000 steps). Random weights were chosen at the initialization of each network, resulting in different networks each run of the algorithm.

4.2.3 Performance measures

The error that a model makes is the (absolute) difference between its predicted age and the true age of a subject. The performance was measured with the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), linear correlation coefficient (corr) and R^2, which denotes the amount of variance present in the true labels that is explained by the model. All these measures were calculated on the models' predictions on the left-out folds to assure that validation was only carried out on unseen data points.

4.2.4 Feature selection

Not all features present in the data set were suitable or informative for the purpose of predicting age. The *iq* is usually not an indicator for age and was left out during training due to mediocre effects on the results. As a separate feature, *sex* is not an informative feature for age either, but it may be in combination with other brain volumes. The total brain, cerebrum ('big brain') and cerebellum ('small brain') features were left out since they do not hold any extra information when their respective gray and white matter volumes are already present in the data.

The first models were trained using all eight features that remain, namely *ic*, *bbgm*, *bbwm*, *cbgm*, *cbwm*, *vlat*, *v*3 and *sex*. Trying all $2^8 - 1$ non-empty subsets of these eight would take

too much time and is out of the scope of this study. By solely looking at the validation results, different combinations of features were attempted, to see which feature set was the most informative in predicting age. The performances of the best feature sets are shown in the results. Based on these results, a final choice of features was made to train a model on the full train set and apply it to the test set.

4.2.5 Weights

To reveal which features influenced the predictions the most, the weights of the final models are reported in the result section. For linear SVMs, a weight for each feature follows directly from the model. Since RBF SVMs do not yield meaningful weights due to the nature of the radial-basis kernel, no RBF weights are reported. Neural networks have weights on their connections, but they cannot be interpreted in the same linear way as the linear SVM weights.

To still get an idea of the features' influence on the networks, ANN weights were calculated by taking the sum of the weights on connections going from a feature's input node to the hidden layer (in figure 3.3 this would be 11+26.5 for x and 8.5-47.5 for y). A problem with this approach is that the sign of the weights may be flipped further in the network by a negative weight going from the hidden node to the output. Therefore, the sign of that weight was considered, flipping the sign of the original weight if it was negative.

The weights of both the linear SVMs and the ANNs were scaled to a sum of 100 (ignoring minus signs) to make sure the proportions were equal between models. The mean was taken over the absolute values of these weights to get an idea of the average influence each feature had and to prevent positive and negative weights canceling each other out. To still show which features had a mostly positive or negative influence on the model output, the result section will show which features had more negative weights than positive ones.

4.2.6 Ensembles

In many machine learning applications, the primary issue is to show the performances of a model. But if it is our aim to eventually apply the models practice, a final model that is ready to use is required. Cross validation does not yield a single applicable model, but it is possible to select one (like the approach in chapter 3) or to construct an ensemble.

Each execution of the training algorithm resulted in 11 models, trained on $\frac{10}{11}^{\text{th}}$ parts of the train data. For linear SVMs it is possible to average the weights and combine these models into one, but this is not the case for the RBF kernel and the neural networks. Therefore ensembles were composed; a subject was classified by taking the predictions of the models that were not trained on that particular subject. Every run contains exactly one such model, in which the subject was part of the validation fold. The mean or median of those 11 predictions was taken to get a more robust model. Extreme predictions (less than 0 or more than 80 years) were omitted out before taking the median or mean. These values occurred sporadically and only in the neural networks, typically when the input of a node fell on the steep part of a sigmoid function that was approximating a step function.

4.3 Results

There were two feature sets that performed particularly better than other combinations, namely the feature set with all eight features (section 4.2.4) and a set with six features that omitted the ventricle volumes ($vlat$ and $v3$). Results for both these sets (referred to as the 8-feature and 6-feature sets) are presented in this section.

4.3.1 Validation results

In figure 4.1 the validation scores of the three learning algorithms can be found, for both feature sets. The performances are based on all validation predictions of each run, averaged

over all 11 runs.

The RBF and ANN models performed significantly better than the linear model ($p < .05$, over 11 runs) on all values. The RBF model outperformed the ANN, but only significantly so on the correlation and R^2 values and the 8-feature RMSE. These validation scores suggest that the RBF SVM produced the best models on this data set, with a mean absolute error of 4.332 years over the validation predictions. The standard deviation over the MAEs of the 11 runs was 0.022.

Comparing the results of the different feature sets does not reveal which one is superior. The correlation and R^2 values were very similar between the sets and did not differ significantly. The only significant differences were found on the RMSE and MAE values of the Lin and RBF models. There was no overall pattern of improvement or decline visible in the performances between the sets.

	Validation results			
	RMSE	MAE	Cor	R^2
8-feature set				
Lin	6.419	4.677	0.866	0.751
RBF	6.045	4.329	0.882	0.777
ANN	6.200	4.342	0.875	0.765
6-feature set				
Lin	6.400	4.698	0.867	0.751
RBF	6.081	4.283	0.881	0.776
ANN	6.127	4.332	0.877	0.770

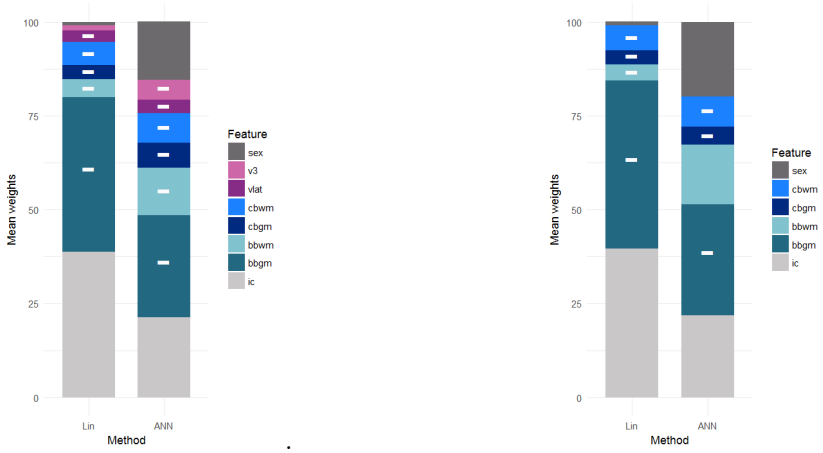
Table 4.1: **Average validation scores** of the linear (Lin) and RBF SVMs (RBF) and the neural networks (ANN), over the validation predictions of cross validation 11 runs. The RMSE and MAE values are measured in years.

The results do show that leaving out the ventricle volumes (*vlat* and *v*3) does not diminish the performances significantly (except for the Lin MAE and RBF RMSE). These volumes were apparently not informative enough, which can be explained by the fact that these values tend to suffer from measuring noise more than the other volumes. Their means in the data were 12.79 and 0.84 cm^3 respectively, while the other means range from 46.38 (*cbgm*) to values as large as 1474.71 (*ic*). Therefore slight measuring errors of the same order had a greater impact on the ventricle volumes than on the large volume regions. This makes the ventricles less reliable for predicting purposes. The correlation between the strength of the linear SVM weights (discussed in the next section) and the means of their corresponding features in the data was 0.839, even though the data was scaled before training. This shows indeed that the volumes of larger brain areas were generally of greater influence in the models.

4.3.2 Weights

The influence of the weights of each feature is shown in figure 4.2. For both feature sets, *ic* and *bbgm* were the two most important features, though this pattern was clearest in the linear models. The decrease of these feature weights in the ANN models was compensated by the *sex* feature, which was far more prominent there.

As expected from the linear model, the weights were very consistent between the 121 models, with standard deviations ranging from 0.26 for the 6-feature *vlat* to 1.19 for the 8-feature *bbgm*. The *sex* feature was the only one that did not have the same sign for all weights, being positive in 78 cases (6-feature) and 99 cases (8-feature) out of the 121 models.



(a) 8-feature models

(b) 6-feature models

Figure 4.2: Scaled absolute weights of the features of the Lin and ANN models. RBF models do not produce meaningful weights and are therefore not included. The absolute weights of all 121 models were scaled to a sum of 100 and then averaged and scaled again. The minuses show which features had a negative weight in at least half of the 121 models. The Lin weights show how much the model output increases in terms of the feature inputs, in contrast to the ANN weights that cannot be interpreted linearly and were computed in a custom way, described in the method section.

Much more variation was present in the ANN model weights, with standard deviations ranging from 4.18 for *v3* to 15.9 for *sex* (both 8-feature). This shows that the proportions between weights variated a lot between networks. When looking at the absolute values of the weights among the 121 models, the correlation between *ic* and *bbgm* was 0.74 in the 8-feature models while the correlation of *sex* with these was -0.70 for both. This means that *ic* and *bbgm* tended to have a big or small influence together and that *sex* filled in this gap in models where they had low weights.

4.3.3 Ensembles

Using the approach described in the method section, ensembles were used as well to calculate validation scores. Taking the mean instead of the median of the 11 predictions yielded the best results, therefore the performances are shown for the mean ensembles. See table 4.2.

	Ensemble validation scores			
	RMSE	MAE	Cor	R2
8-feature set				
Lin	6.417	4.675	0.867	0.751
RBF	6.028	4.308	0.882	0.779
ANN	5.919	4.102	0.886	0.785
6-feature set				
Lin	6.399	4.697	0.867	0.751
RBF	6.053	4.260	0.882	0.778
ANN	5.980	4.236	0.883	0.780

Table 4.2: **Ensemble validation scores**. The predictions were acquired by taking the left out fold predictions of all 11 runs and taking the mean for each subject. These performances therefore represent one value, so no significance tests were carried out. The RMSE and MAE values are measured in years.

The neural networks have improved themselves the most compared to the single model scores in table 4.1, as was expected given their high variation in models. They also performed best on all measures for both feature sets, with the best mean absolute error being 4.102 years. The RBF ensemble follows closely, leaving a gap to the linear model which did not benefit from the ensemble, as expected from its low variation between runs.

Based on the results of both the individual and ensemble models, the non-linear models are the best performers on this data set. The results do not show a clear superiority of one feature set over the other, though the 8-feature set performed slightly better on the ensembles. Following the principle of Occams Razor, the simplest model should be preferred in cases of similar performance. Therefore the 6-feature model will be used to produce the final models.

4.3.4 Test results

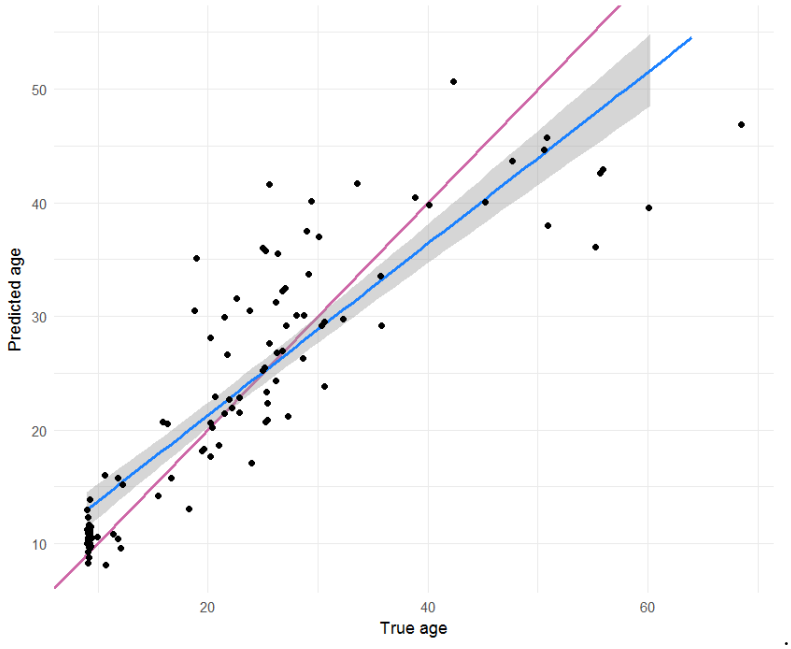
Using the 6-feature set, RBF SVMs and ANNs were trained on the full train data set ($n = 400$), without cross validation. They were then subjected to the previously unused test set ($n = 103$) to get an idea of the generalizability to unseen data. Again, 11 times 11 models were trained to give rise to 11 ensembles combining 11 models each. Both the performances of the 121 individual models and the 11 ensembles are displayed in table 4.3.

	Test scores			
	RMSE	MAE	Cor	R2
Individual models				
Lin	7.170	4.876	0.844	0.712
RBF	6.584	4.349	0.871	0.758
ANN	6.630	4.493	0.867	0.752
Ensembles				
Lin	7.170	4.876	0.844	0.712
RBF	6.575	4.342	0.871	0.759
ANN	6.416	4.342	0.876	0.767

Table 4.3: **Test scores** of the RBF SVM (RBF) and the neural networks (ANN), on 11 models assessed on the test set (n=103). The individual model scores are an average over the 121 models.

The performances have dropped compared to the validation results, with decreases of one to two tenths of years in mean absolute errors. This relatively slight drop compared to the validation performances suggests that the models are fairly capable of generalizing, though it must be stated that this test set is not a fully independent set and does not contain between-sample heterogeneity. Apart from that, similar patterns to those in the validation scores are visible, with the ANN being the best performer and the linear SVM having the weakest performance.

Using the ensemble has made the performances more stable, especially for the ANNs. The standard deviation of the RMSE values of the individual ANN models was 0.72, while it was only 0.09 for the ensemble RMSEs. Also, the ANN was the only method with a significant ($p < .05$) improvement when using ensembles compared to individual models. All differences between ensemble method scores were significant for given measures as well, except for the MAE of the RBF and ANN. Note that the linear model performed identically on both the individual models and the ensembles up to three decimals, but there were differences beyond these decimals.



(a)

Figure 4.3: The predicted age plotted against the true age of 103 test samples, together with the regression line (blue) and the ideal $y = x$ line (purple). The predictions were based on the first ANN ensemble model. RMSE=6.32 years, MAE = 4.30 years, sd of absolute difference = 4.66.

Figure 4.3 shows a plot of the predictions made by the first ANN ensemble model, compared to the true chronological age of the 103 test set samples. The relatively large group of 9 year olds was predicted well, with predictions ranging from 8.3 to 13.9 years and an MAE of only 1.49 compared to 5.20 for all the other subjects. At higher ages, more deviation from the true age is visible; especially after the age of 50 the predictions were consistently too low (ranging 36.0-46.9). Moreover, the highest prediction that was made was 50.7 years, while the oldest subject present in the train set was 64.9 years old.

4.4 Discussion

4.4.1 Performance differences due to age

The plot in figure 4.3 showed that the ANN ensemble is better at predicting children (aged 9 to 17) than adults. This can be explained by looking at the big brain gray matter volumes and dividing them by the intracranial volumes to compensate for skull sizes. Figure 4.4 plots these values against age. The curved, non-linear trend of the gray matter loss could explain why the non-linear models outperformed the linear ones. The figure also shows that the children and especially the 9 year olds had significantly more gray matter than the older subjects and were easier to predict using this feature. The regression curve shows the trend of the gray matter volume loss, which stagnates around the age of 40. After this age, the amount of gray matter hardly decreases anymore. In follow-up studies, models could be trained on data with exclusively adult subjects to find out whether this will enable the models to improve performance on these subjects.

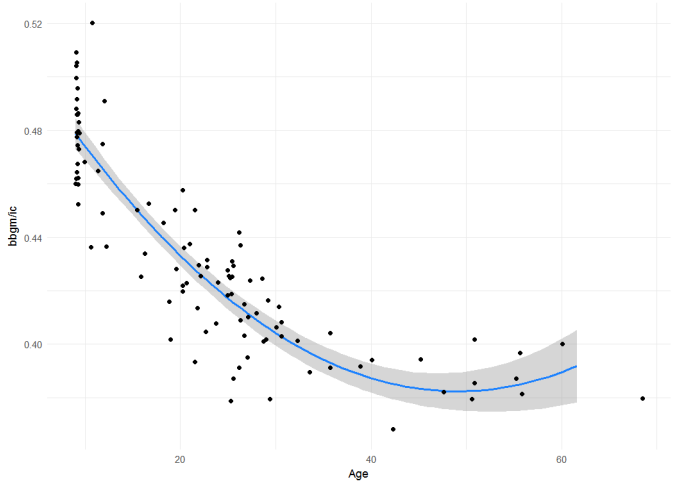


Figure 4.4: The big brain gray matter volumes divided by the intracranial volumes ($bbgm/ic$) of the 103 subjects, plotted against their age. Linear correlation coefficient=-0.81.

4.4.2 Weight interpretation

Though the intracranial volume *ic* hardly had any effect on age by itself (correlation coefficient of -0.003), it had a high positive influence in the models, having the second highest weight. It is likely that *ic* works as a compensator for the brain volumes, since having a bigger cranial volume indicates that the brain was probably larger to begin with and has reduced more compared to people with smaller *ic* volumes. Remarkably, the *ic* influence is of similar size of *bbgm* in both the Lin and ANN models, though they both decreased considerably in the ANN models. This suggests that the compensating effect of *ic* is mostly linked to big brain gray matter.

In most of the models, *sex* had a positive weight. This is most likely due to the fact that women have a smaller head on average, with an *ic* volume of 1392 compared to 1539 for men (a significant difference, $p < .05$). This means that when two persons of different sex have the same brain volumes, the man is presumably older since his brain would have had to shrink more to get to the same stage. Also note that the males present in the data set had an average age of 24.2 while the women were 22.5 years old, which is another reason for the *sex* weight to be positive; a form of overfitting that is hard to avoid.

Omitting the ventricle volumes *vlat* and *v*3 from the feature set did not change the distribution of the weights heavily. Their role was replaced by *cbwm* in the Lin models and *sex* and *bbwm* in the ANNs.

Overall, the weights show that there are definitely differences in the weight distribution between linear models and the neural networks. Again, it should be noted that the neural networks weights are not as straight forward as the linear ones and were calculated based on some assumptions as discussed in the method section. However, it is clear that the neural networks focused more on the *sex* feature than the linear SVMs did. The non-linearity of the networks may have allowed them to use the feature in a more effective way.

4.4.3 Comparison to earlier work

Some previous age predicting studies using brain features have been carried out, all using linear SVM regression. They used gray matter density maps based on MRI scans, in which the amount of gray matter tissue present in each voxel (3D-pixel) in the brain is registered. These voxel maps consist of more than 100 000 features, far more than the number of variables considered in this thesis.

The brain age model of Schnack et al. [10] ($n = 386$) had a mean absolute error of 4.31 years and an R^2 of 0.79 measured by cross validation. Tested on stronger signal MRI data (3-T), a MAE of 3.86 was achieved. Another study, by Franke et al. [12] ($n = 410$), also showed better results on the test set containing both lower (1.5-T) and higher (3-T) strength MRI data (MAE=4.61) than on the set containing only 1.5-T scans (MAE=5.44). In Koutsouleris et al. [13] ($n = 800$), a MAE of 4.6 and R^2 of 0.83 were achieved on a set containing both MRI strengths, measured with cross validation. The 3-T MRI scanners, having double signal strength, are less likely to suffer from noise and therefore presumably allowed these models to make preciser predictions.

The results of the non-linear models (ANN) in this thesis are comparable to those in the literature (MAE=4.236, R^2=0.780 with cross validation, MAE=4.342, R^2=0.767 on test set). The linear ones (MAE=4.876, R^2=0.712 on test set), however, show weaker performance than most results in the previous studies, that were achieved with linear models as well. Koutsouleris et al. [13] mentioned that non-linear kernels were tried but did not improve the performance, while significant improvements were found in this thesis. The high number of features in the MRI data may be problematic for non-linear methods and lead to overfitting.

The brain volume data containing only 6 or 8 features turned out to be informative enough to achieve similar performance to earlier studies on high number feature data. The use of non-linearity was needed to attain these comparable results. A limitation is that the previous studies all used data of subjects that were at least 16 year olds, while the best performances in this study were achieved on the younger subjects aged 9-18 years old. Further research on adult brain volume data should be conducted to ensure that these performances are truly comparable.

4.4.4 Conclusion

In conclusion, decent performances were achieved compared to previous studies, using only a few features. The non-linear methods performed significantly better than the linear models, which demonstrates that the use of these methods is feasible on clinical data. However, the train data set was relatively large in terms of the psychiatry field, which is particularly beneficial to the non-linear methods. Since predicting age is a regression task, no conclusions regarding heterogeneity can be drawn yet. The main goal of this chapter, assessing the potential of the proposed machine learning techniques, was nonetheless achieved.

Chapter 5

Bipolar offspring data

5.1 Introduction

A clinical data set from the domain of psychiatry, possibly containing heterogeneous patterns, was used to assess the techniques. The set contains data from children of parents with bipolar disorder that were followed for 12 years. A study on this set showed that these children have an increased risk of developing a depression disorder, or bipolar disorder to a lesser extent (Mesman et al. [14]). The goal in this chapter is to predict the prognosis of these children, using only the baseline data. Predicting which participants will develop a depression or bipolar disorder can help with deciding whether treatment or medication should be provided in early stages.

5.2 Method

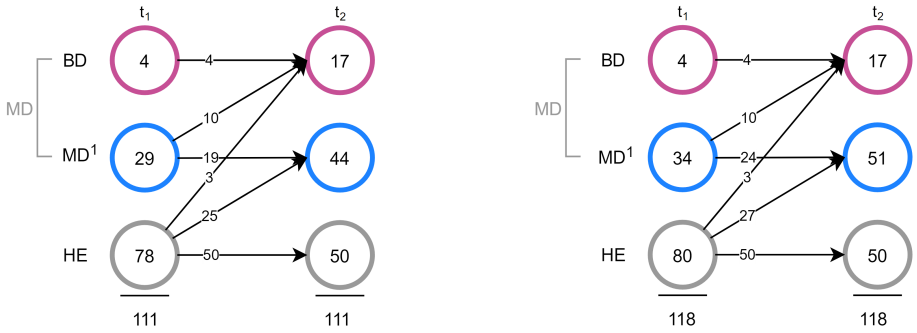
5.2.1 Data

The Dutch bipolar offspring cohort is a data set of 140 children aged between 12 and 21 years, from 86 different families that were recruited in the years 1997-1998. Each participant had at least one bipolar parent at baseline. The offspring was followed for twelve years total, with follow-up interviews taken 1, 2, 5 and 12 years after recruitment. 108 out of the 140 participants completed all the follow-ups. Details on the recruitment procedure of the data can be found in Wals et al. [15].

Various features of the participants were available in the data set. Demographic information like age at baseline, sex, total iq, sex of the bipolar parent and social economic status (SES) were documented. Lifetime DSM-IV (Diagnostic and Statistical Manual of Mental Disorders) diagnoses and the presence of a large range of symptoms were determined at each assessment. In this thesis the focus will lie on predicting bipolar disorder and mood disorders in general. Mood disorders (MD) are psychiatric illnesses with depression as a major symptom, as specified in the DSM-IV. They include Major Depressive Disorder (MDD), Bipolar Disorder type I and II (BD), Substance-Induced Mood Disorder (SIMD) and Dysthymic Disorder. To make the distinction clear between mood (including BD) and unipolar mood (excluding BD), the latter will be referred to as MD^1 (=MD-BD).

Figure 5.1 shows transitions in the two subsets of the KBO set that were used for training. Both sets contain the 107 participants that completed the whole 12-year track. Since none of these subjects rehabilitated (a transition from MD to HE or from BD to MD^1), the assumption was made that this was also the case for the participants that were diagnosed but left the study early. Therefore three participants that were diagnosed with bipolar disorder before t_2 were included, yielding the BD set. For predicting MD, seven participants that left the study before t_2 but after they were diagnosed with mood were added additionally. The

group listed in the figure as healthy (HE) does actually contain other diagnoses that were not categorized as mood disorders by the DSM-IV.



(a) BD set (b) MD set

Figure 5.1: Transitions in the two subsets used for training. The 'healthy' group stands for participants that had no mood disorders (neither bipolar nor unipolar), but possibly other disorders. a) The BD set ($n = 111$) that was used for predicting bipolar disorder at t_2. At t_1, 7 participants in the healthy group had an anxiety disorder, 10 had any other DSM-IV diagnosis than mood or anxiety and 62 had no diagnosis. b) The MD set ($n = 118$) that was used for predicting mood disorders (either bipolar or unipolar) and multiclass prediction at t_2.

5.2.2 Training

In this study, only baseline (t_1) and 12-year follow-up (t_2) were considered. All the data used for training was taken from t_1, and the labels to be predicted were taken from t_2.

Firstly two binary classification tasks were performed, by predicting diagnosis at t_2. Mood disorders (MD) were classified against non-mood (HE) and subsequently bipolar disorders (BD) against non-bipolar (HE+MD^1). Analogous to chapter 4, these classifications tasks were performed using linear SVMs (Lin), radial-basis SVMs (RBF) and neural networks (ANN). Finally, a multiclass neural network was constructed to attempt to predict HE, MD^1 and BD simultaneously.

All the features from the data that were considered during the training process can be found in table A.1 of the appendix. Like with the brain volume data, not all combinations could be tried, hence various combinations were attempted and the features that were consistently part of better performing models were selected for the final models. Table 5.1 shows the subset of the features that ended up in one of the final models.

Among those features were two sum scores, one for depression and one for mania, which are the sums of varying symptoms falling under their category. Those symptom features were scaled 1 to 3, with 1 meaning not present, 2 mildly present and 3 highly present. The ones that were found to have a significant effect on developing bipolar or mood disorder (Mesman et al. [16]) were all considered. They are the first eight features in the table. The sum scores they are part of were used as well, in addition with the total iq. It should be noted that the machine learning techniques will treat all the features as continuous variables, while the symptom features are ordinal; the difference in symptom severity between score 1 and 2, and between score 2 and 3 does not have to be equal.

Feature	Description	Range	Mean	SD
selfc	marked self-consciousness	1-3	1.35	0.61
tens	marked feeling of tension/unable to relax	1-3	1.46	0.66
death	recurrent thoughts of death	1-3	1.43	0.67
suic	suicide ideation	1-3	1.20	0.54
insom	middle insomnia	1-3	1.11	0.40
elat	elation, expansive mood	1-3	1.18	0.45
decslp	decreased need for sleep	1-3	1.13	0.41
raceth	racing thoughts	1-3	1.15	0.45
ssDep	sum score depression	21-50	25.49	6.05
ssMan	sum score mania	4-16	4.65	1.81
tiq	total intelligence quotient	72-152	114.10	14.89

Table 5.1: Features used during training process with their abbreviation and a description. The range shows the minimum and maximum values present in the data. All these features were measured with integer values. The data in this table was extracted from the BD set ($n = 111$) as shown in figure 5.1a. These are the features that ended up in the final models; all considered features can be found in appendix table A.1

.

Performance measures

The performance of the models was mainly measured with the accuracy, which is denoted by the amount of true positives and true negatives divided by the total number of subjects. As some models were trained on imbalanced class data, the balanced accuracy ($bAcc$) was reported as well and treated as the main measure. The balanced accuracy ensures that both classes contribute equally to the final percentage.

Other measures that were used are the sensitivity and specificity. The sensitivity (also known as recall) is the percentage of patients that were classified as such by the model, denoting the model's ability to identify the illness. The specificity is the percentage of healthy subjects that were classified correctly. A lower specificity means that more healthy participants were diagnosed falsely. The sensitivity and specificity are not uniformly defined for multiclass models. In this study, they were computed for each class opposed to the other two combined.

The threshold that decides what prediction values are labeled as +1 or -1 is set a zero by default. Shifting it to other values will alter the sensitivity/specificity ratio, yielding a curve (the ROC curve, explained further in the next section). The area under this curve (AUC) measures the performance of a model across these different thresholds and will also be reported in the results. A value of 1 is optimal, while a value of 0.5 denotes a model that performs at chance level.

The weights of the linear SVMs and the ANNs scaled to a sum of 100 are reported. They were calculated according to the same method as in chapter 4 (section 4.2.5).

Bipolar/mood models

Given their good performance on the brain volume data, only ensembles were considered. Once again, 11 identical runs were performed, including an 11-fold cross validation procedure each. Every subject was then predicted by combining the 11 models (one in each run) that were not trained on that particular subject. The mean was taken over their predictions to get a classification. For the SVMs double nested cross validation was performed, with 5 inner loops to tune the parameters. Both the cost and the gamma parameters had a range of 0.001, 0.0025, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 2, ..., 10. The neural networks consisted of one hidden layer with two hidden nodes. A threshold of 0.0001 was used as the stopping criterion.

To deal with the vast imbalance between the classes in the classification of bipolar disorder (BD against HE+MD[1]), class weights were added to the SVMs and an adjusted error function to the neural network. In both cases, classifying a bipolar subject incorrectly was penalized more heavily ($\frac{94}{17} \approx 5.53$ times) than a non-bipolar one.

The models were trained on data in which some of the subjects were already diagnosed at t_1 with the disorder that the model was trying to predict at t_2. In order to avoid further reduction of the sample size, these baseline-cases were maintained during training and left out in the testing phase to prevent bias. Since they usually have more severe symptoms and are therefore easier to classify, omitting them in the testing phase may result in a skewed balance between the classes. Assuming they are all classified correctly, the sensitivity will go down while the specificity remains unaltered. Also, omitting subjects during testing means that models are evaluated on data with a different composition than the set they were trained on. This means that the default threshold of 0 that the models fitted on may not show the best performance on new data anymore. Therefore a shift of the threshold is proposed.

The performance of a model for various threshold values can be visualized by an ROC curve. An example can be viewed in figure 5.4 of the result section. An ROC curve plots all the sensitivity and specificity value pairs of a model along the spectrum of possible thresholds. The x-axis represents the 1-specificity values, the y-axis the sensitivities. The closer the curve gets to the top-left corner, the better and more balanced the performance will be.

New thresholds were calculated by selecting the two sensitivity/specificity value pairs with the shortest euclidean distance to the ideal point (1,1) of the ROC curve. Since selecting the single best sensitivity/specificity ratio based on the test data may result in performances that are too optimistic, the mean of the thresholds belonging to these pairs was calculated. This results in a threshold located between two local maxima. Figure 5.4 shows this in action, with the black dots representing the two selected local maxima and the blue dot the proposed threshold, located in a high performing region of the curve but not on a local maximum. In the result section, performances will be shown for both the unaltered threshold and the proposed thresholds.

Multiclass model

The multiclass model aims to predict BD, MD[1] and HE simultaneously. A network with two hidden layer nodes and two output nodes was trained. The output of the first output node decides between being healthy and having a mood disorder, while the second output node decides between unipolar mood (MD[1]) and bipolar disorder (BD). A more complex network with one output node per class was considered, but this adds even more weights and requires scoring to decide on a predicted label. Since the `R` package only allowed one error function for all output nodes, the class imbalance was only corrected for the mood class as opposed to the healthy class.

The coding used for the outputs nodes is visualized in figure 5.2. This particular coding places the BD class the furthest away from HE and views BD as an extension of MD[1] by putting them closer together. Given the binary outputs of two nodes, there are four class options and one remains empty. No data points in the train data were assigned to the empty class (coded as (-1,1)), but the resulting network may classify subjects to it.

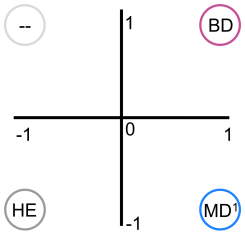


Figure 5.2: Coding of labels for the multiclass neural network. Horizontal axis represents the first output node, the vertical axis the second.

In the results, the performance will be shown on all subjects, on a subset without the BD patients of t_1 and on that same subset leaving out the patients diagnosed with MD that did not transition to BD, assuming that these were easier to predict.

5.2.3 Network interpretation

To get some insight in the mechanics of the neural network models, their weights and node inputs/outputs were analyzed. The output of each hidden node was multiplied by their weight to the output node. The output node bias was divided by the number of hidden nodes and added to those outputs. The resulting values show the contribution each hidden node makes to the final output. The correlations between these hidden node outputs and the feature inputs were computed, which resulted in a correlation vector for each hidden node. These vectors provided information on what kind of features each hidden node focused on.

The analyses were only performed on the multiclass model, having a slightly more complex structure than the binary neural network classifiers. The correlations of the first network of the first run, and the mean over the correlations of all 121 models were calculated. In models in which more than half of the correlations were negative, their signs were flipped given the arbitrariness of signs due to the weights.

There was a presumption of a recurring role that each of the two hidden nodes played in the models, but their order in the network is always arbitrary. K-means clustering ($K = 2$) was performed on the correlation vectors to determine whether this pattern was truly present. In all but one of the 121 models, the vectors of the two hidden nodes were assigned to different cluster centers, suggesting that there was indeed a similarity between the roles of the hidden nodes among models. In the model of which the vectors were assigned to the same centroid, the tie was broken by determining which vector lied the furthest away from both centroids and assigning it to its closest centroid, giving the remaining vector to the other centroid.

In addition, to show patterns in the influence each hidden node had on the final output, the absolute values of the four weights on connections between the hidden layer and the output layer were sorted increasingly. Then for each of the 121 networks, the mean over the absolute weight values in each sorting rank was taken.

5.2.4 Checking for non-linearity

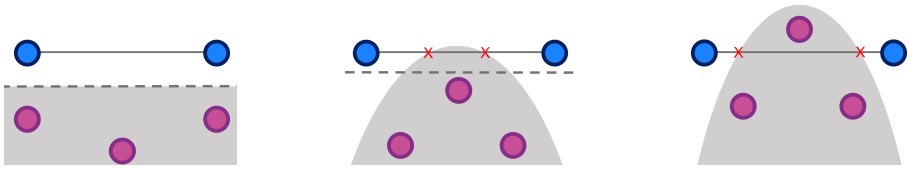
One of the goals of this study is to explore the potential of non-linear methods as opposed to the established linear ones. To see whether in fact the proposed approaches make use of their non-linear freedom, the models were inspected by looking at the non-linearity they conveyed in two ways: firstly, the non-linearity in the whole feature space was assessed. Subsequently, only the classifications of the data points was considered, ignoring the decision boundary.

In feature space

To assess the non-linearity in feature space, per single model or run ensemble, the subjects that it was trained on were represented as points in feature space. Depending on the feature set of the model, duplicate data points may exist and were left out. A (parametric representation of a) line was drawn between each pair of remaining points. The label was predicted on 100 equal divided points on those lines. For each line, the amount of times the predicted label switched from -1 to 1 or vice versa was documented.

In a perfectly linear situation (fig 5.3a), lines between points with different labels will switch exactly once and those between same-label points will never switch. When the optimal decision boundary is non-linear, multiple switches per line become possible but will not necessarily arise (fig 5.3b). However, when the points are labeled in a way that is impossible to separate linearly, at least one multicrossing line must be present (fig 5.3c).

The degree of non-linearity was measured with a percentage ($wPerc$) by taking the distances of all lines that switch label at least twice, divided by the sum of all line distances. A longer line has more opportunities to change labels and contributed less to the percentage in this way. The mean of the weighted percentage was taken over the 11 ensembles (one ensemble for each run of the algorithm) and over all 121 individual models. The percentage of models/ensembles that had a non-zero $wPerc$ are reported as well.



(a) Linearly separable (b) Linearly separable, but optimal non-linear separation (c) Not linearly separable

Figure 5.3: a) The data points have been separated in a linear way, which is optimal in this situation. The line between the blue points does not cross the decision boundary. b) The data points can be separated linearly (dotted line), but a non-linear boundary allows for a wider margin and may therefore generalize better. As a result, the predicted label changes twice on the line between the blue points. c) The data points are not linearly separable. Since this specific kind of classification (dichotomy) can only be achieved with a non-linear separation, there must be a line between the points that crosses the boundary at least twice.

In the classifications

The approach for feature space discussed above reveals forms of non-linearity, but does not differentiate between the situations in (b) and (c) from figure 5.3, while (c) is the only of the three that depicts a classification that could not be realized by a linear boundary. To find out whether the use of non-linear methods did in fact result in a classification of the data points that was otherwise impossible, the predictions of the models were taken as the correct labels for the data the models were trained on. An SVM with a high cost parameter ($C = 10000$) to approximate a hard-margin SVM was trained and tested on this data. Anologous to the feature space approach, the accuracy of the 11 runs and all 121 models individually are reported. Additionally, the accuracy of the ensemble over all runs and the accuracy of the feature data set with true labels are reported. The ensemble over all runs corresponds to the final models of predicting mood and bipolar disorder.

Models with perfectly linearly separable predictions ((a) in fig 5.3), are able to achieve an accuracy of 1. The higher the accuracy, the more linearly separable the predicted classifications were. Since the models were trained on different features sets, their feature spaces differ and the degree of non-linearity may vary.

5.2.5 Three approaches on same features

Selecting the best set of features for each approach influenced the performance and amount of non-linearity more than was expected. Besides, the majority of the features used were ternary, resulting in many subjects cluttering together in feature space. To make comparison easier, posthoc three more mood models (Lin, RBF, ANN) were trained on the same feature set. The most common features from the best mood models were chosen (self, tens and death), in addition with the depression sum score to create more diversity in feature space. The results will be discussed briefly.

5.3 Results

5.3.1 Performance

The main results are presented in this section; for additional and extended results, refer to appendix A.

Mood Disorders

The results of the best performing ensemble classifiers for mood disorder can be found in table 5.2. For each model, the upper row shows the results that came directly from the algorithm. The subjects that were already diagnosed with mood at t_1 were excluded from testing; the results including these subjects can be found in the appendix (table A.2). The second row shows the results with a shifted threshold, in order to restore the sensitivity/specificity balance. For the linear and neural network model the mean weights averaged over the 121 submodels and scaled to a sum of 100 are shown.

	Scores						Features						
	Thresh	bAcc	Acc	Sens	Spec	AUC	ssDep	ssMan	tiq	selfc	tens	death	decslp
Lin													
	.	.730	.763	.600	.860	.777	35.1	18.5	.	.	23.8	18.9	3.7
	-.061	.743	.763	.667	.820	.777							
RBF													
	.	.727	.750	.633	.820	.702	.	.	.	✓	✓	✓	.
	-.122	.727	.750	.633	.820	.702							
ANN													
	.	.710	.713	.700	.720	.745	.	.	8.5	13.4	22.1	56.0	.
	.636	.767	.800	.633	.900	.745							

Table 5.2: Performances of the mood ensembles. Results are shown for the default and proposed threshold values.

		True	
		MD	$\overline{\text{MD}}$
Pred	MD	18	7
	$\overline{\text{MD}}$	12	43

(a) Lin

		True	
		MD	$\overline{\text{MD}}$
Pred	MD	19	9
	$\overline{\text{MD}}$	11	41

(b) RBF

		True	
		MD	$\overline{\text{MD}}$
Pred	MD	21	14
	$\overline{\text{MD}}$	9	36

(c) ANN

		MD	$\overline{\text{MD}}$
Pred	MD	20	9
	$\overline{\text{MD}}$	10	41

(d) Lin (thres=-.061)

		MD	$\overline{\text{MD}}$
Pred	MD	19	9
	$\overline{\text{MD}}$	11	41

(e) RBF (thres=-.122)

		MD	$\overline{\text{MD}}$
Pred	MD	19	5
	$\overline{\text{MD}}$	11	45

(f) ANN (thres=-.636)

Table 5.3: Confusion matrices for the mood disorder classifiers from table 5.2. $\overline{\text{MD}}$ denotes the non-mood group. The first row shows the tables with the default threshold, the bottom row shows the results after the threshold adaptation.

Without the threshold adaptations, the linear model reached the best accuracies and AUC, making it the best classifier on the validation data. In contrast, the sensitivity/specificity ratio was more imbalanced than in the other models, favoring the specificity. Moving the threshold relieved this imbalance slightly. Although the linear model seems to perform best, the difference in the accuracies is marginal.

The main features of the linear model are the depression and mania sum scores, along with death thoughts and tension. These features comprise symptoms for depression (ssDep, death), bipolar disorder (ssMan) and anxiety disorder (tens), which the first two are the main components of the mood disorder group. Decreased need for sleep did not play a big role,

but leaving it out resulted in a slightly less performing model. The most influencing feature in the neural networks were the death thoughts. The ANN was the only model that included the iq as a feature, which played a small role with 8.5% influence.

Adapting the threshold improved the sensitivity/specificity ratio of the linear model, but did nothing for the RBF and made the balance of the ANN worse. However, a perfectly balanced ratio is not necessarily desired in all situation. Depending on the intended application of the classifier, a higher specificity or sensitivity may be preferred.

The confusion matrices corresponding to these models are presented in table **??**. The best performing linear model had a relatively low sensitivity in the default situation, where only 18 out 30 of future mood patients were prognosed correctly. A shift in threshold to restored the balance to classifying 20 patients correctly.

A ready to use linear SVM model for predicting mood can be found in the appendix (table A.6). It shows the weights and intercept for one of the models trained during cross validation, that had a median accuracy performance on the left-out fold.

Bipolar Disorder

For the predictions of bipolar disorder against the rest, the performances are presented in figure 5.4. The values are reported in the same way as the mood disorder results.

In both the unchanged and changed threshold situations, the neural network achieved the best accuracies. In contrast, its AUC was the lowest, which makes the model less flexible for adapting the threshold. As an effect, the proposed threshold did not alter the classification. The best AUC belongs to the RBF model, having a reasonable accuracy performance as well.

	Thresh	bAcc	Acc	Sens	Spec	AUC	ssDep	ssMan	suic	insom	elat	decslp	raceth
Linear SVM													
	.	.709	.897	.462	.957	.822	20.9	8.7	29.7	.	15.8	18.0	6.9
	-.586	.800	.822	.769	.830	.822							
RBF SVM													
	.	.765	.879	.615	.915	.870	.	.	✓	✓	✓	✓	.
	-.602	.800	.822	.769	.830	.870							
Neural Network													
	.	.804	.888	.692	.915	.715	.	.	41.9	.	32.7	25.4	.
	-.445	.804	.888	.692	.915	.715							

Table 5.4: Performances of the bipolar ensembles. Results are shown for the default and proposed threshold values.

		True	
		BD	$\overline{\text{BD}}$
Pred	BD	6	4
	$\overline{\text{BD}}$	7	90

(a) Lin SVM

		True	
		BD	$\overline{\text{BD}}$
	BD	8	8
	$\overline{\text{BD}}$	5	86

(b) RBF SVM

		True	
		BD	$\overline{\text{BD}}$
	BD	9	8
	$\overline{\text{BD}}$	4	86

(c) NN

	BD	$\overline{\text{BD}}$
BD	10	16
$\overline{\text{BD}}$	3	78

(d) Lin (thres=-0.586)

	BD	$\overline{\text{BD}}$
BD	10	16
$\overline{\text{BD}}$	3	78

(e) RBF (thres=-0.602)

		True	
		BD	$\overline{\text{BD}}$
Pred	BD	9	8
	$\overline{\text{BD}}$	4	86

(f) ANN (thres=-0.445)

Table 5.5: Confusion matrices for the bipolar disorder classifiers from table 5.4. $\overline{\text{BD}}$ denotes the non-bipolar group. The first row shows the performance given by the models itself, the bottom row shows the results for the same models with a shifted threshold.

The linear model suffered most from leaving out the t_1 diagnosed patients and had the most imbalanced sensitivity/specificity ratio. After correcting this with the proposed threshold, the three models had very similar accuracies. Note that the performances of the bipolar classifiers were generally better than those of the mood classifiers. In a paired one-sided t-test, the differences were significant ($p < .05$) for the accuracy values, but not for the AUCs.

Looking at the features, the linear SVM focused primarily on the depression sum score, suicide thoughts, elation and decreased need for sleep. The first two of those features are linked to depression, the latter two to mania, which are the two characteristic phases of bipolar disorder. Part of these features were present in the other models as well.

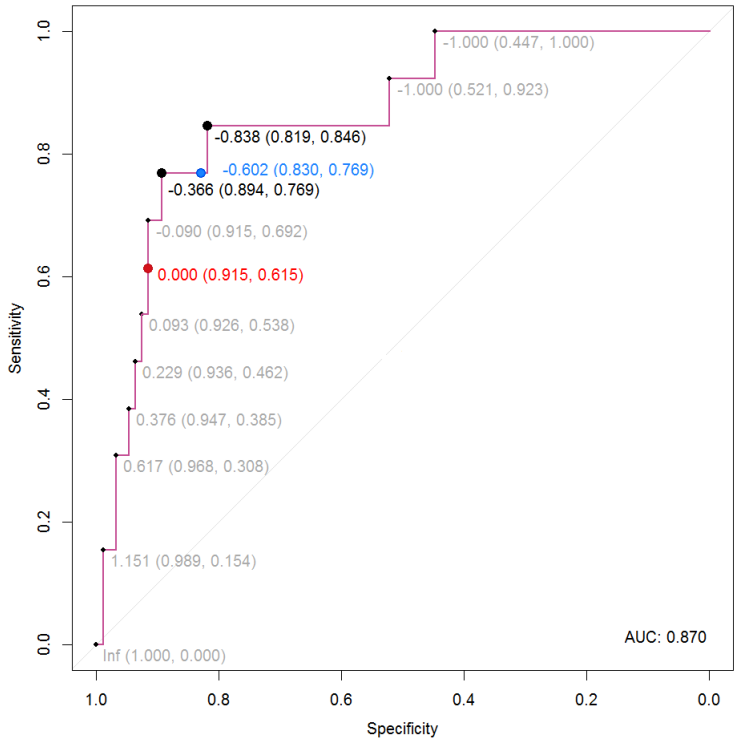


Figure 5.4: ROC curve of the RBF SVM bipolar model. The default threshold of 0 is displayed in red, the proposed threshold in blue. The sensitivity and specificity of each threshold are shown in pairs (spec, sens) behind the thresholds.

Figure 5.4 shows the ROC curve of the RBF SVM ensemble, which had an AUC of 0.870. The default threshold of 0 (red) lies more on the specificity side of the curve, resulting in a skewed balance. The algorithm for the proposed threshold computed the two threshold points lying closest to the top-left (black), and chose the mean (blue) as a new value. In this example, this approach resulted in a threshold located in a more balanced area of the curve without choosing a local maximum. The method did not work out as well in all cases; in some models, the new threshold did not alter the ratio at all (RBF mood model and ANN BD model) or even made they balance worse (ANN mood model).

Like with predicting mood, a linear SVM model to predict bipolar disorder can be found in the appendix (table A.11) as well. It shows the weights and intercept for one of the models trained during cross validation that had a median accuracy value on the left-out fold.

Multiclass Model

The performance of the multiclass model can be found in table 5.6 and the corresponding confusion matrices in table 5.7. The first row in the performance table shows the result for

all subjects including those diagnosed with BD or MD^1 at t_1. To lessen bias, those diagnosed with BD were omitted in the middle row since those never change diagnosis between t_1 and t_2. In the bottom row, patients with MD^1 that remained MD^1 were left out additionally, although they are less trivial to predict due to their potential of shifting to BD.

When all subjects are considered, one subject was classified to the fourth 'empty class'. This means that the first output node in the network classified this subject as being healthy as opposed to having MD, while the second node classified it to BD rather than MD^1. This combination was not present in the train data.

The unbalanced accuracies were reasonable and hardly changed under leaving out the baseline-cases. The balanced accuracies suffered however, since only BD and MD subjects were taken out. This made their classes even smaller, increasing their influence on these accuracies.

The balance between sensitivity and specificity especially was skewed for BD, probably due to the fact that the error function could only be adjusted to one smaller class during training. When higher sensitivities are required for BD, the threshold of the second output node which codes between MD^1 and BD can be adjusted. On the downside, this may direct more subjects to the empty class as well.

	bAcc	Acc	SensBD	SpecBD	$SensMD^1$	$SpecMD^1$	SensHE	SpecHE
all subs	.649	.667	.588	.894	.659	.791	.700	.803
no BD_{t_1} subs	.633	.664	.538	.894	.659	.778	.700	.789
no BD_{t_1}, no $MD^1_{t_1,t_2}$ subs	.599	.636	.538	.920	.560	.778	.700	.684
	ssDep	ssMan	tens	death	suic	elat	decslp	raceth
mean scaled absolute weights	17.7	16.6	11.7	8.3	15.8	6.8	14.0	9.1

Table 5.6: Performances and weights (scaled to a sum of 100) of the multiclass model.

		True		
		BD	MD^1	HE
Pred	BD	10	6	4
	MD^1	3	29	11
	HE	3	9	35
	–	1	0	0

(a) All subs

		True		
		BD	MD^1	HE
Pred	BD	7	6	4
	MD^1	3	29	11
	HE	3	9	35

(b) No BD_{t_1}

		True		
		BD	MD^1	HE
Pred	BD	7	2	4
	MD^1	3	14	11
	HE	3	9	35

(c) No BD_{t_1} & no $MD^1_{t_1}$ in the MD^1 cat.

Table 5.7: Confusion matrices for the multiclass model, corresponding to the rows in table 5.6.

It is clear from the confusion matrices in table 5.7 that most of the baseline-cases were predicted correctly, as expected. The three subjects with bipolar disorder at t_1 were all classified correctly, and out of the 19 t_1 unipolar mood patients that remained MD^1, 15 were predicted correctly. All of the seven rightly predicted bipolar patients were already diagnosed with mood disorder at t_1. All three healthy subjects of t_1 that developed bipolar disorder before t_2 were classified incorrectly (two prognosed as staying healthy and one as developing mood).

5.3.2 Network interpretation

In figure 5.5 a network out of the multiclass model can be found, with bar plots of correlations between input features and the outputs of the hidden nodes for 1) the displayed network and 2) averaged over all 121 networks.

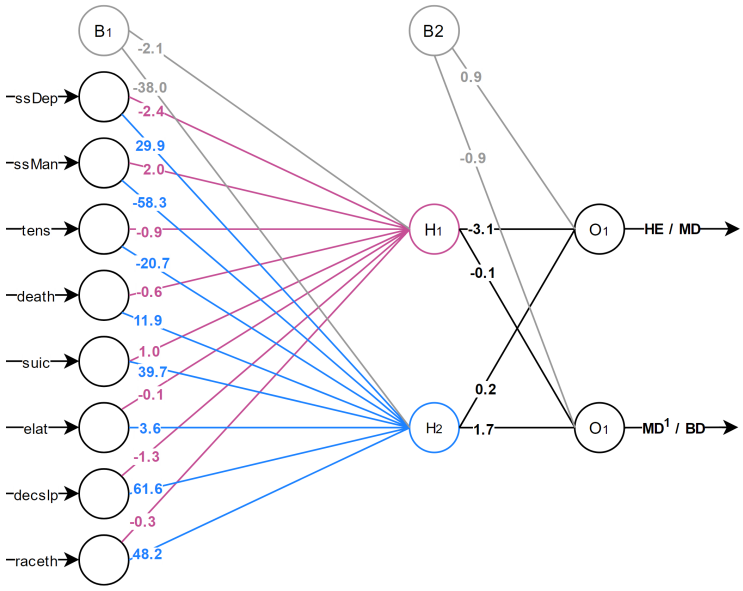


Figure 5.5: Multiclass network with weights (first CV network from first run). The first output node decides between HE (-1) and MD (1) and the second between MD[1] (-1) and BD (1). For example, having BD is coded as MD in output node 1 and BD in output node 2. The combination HE in output node 1 and BD in output node 2 does not code for a class and was not present in the train set.

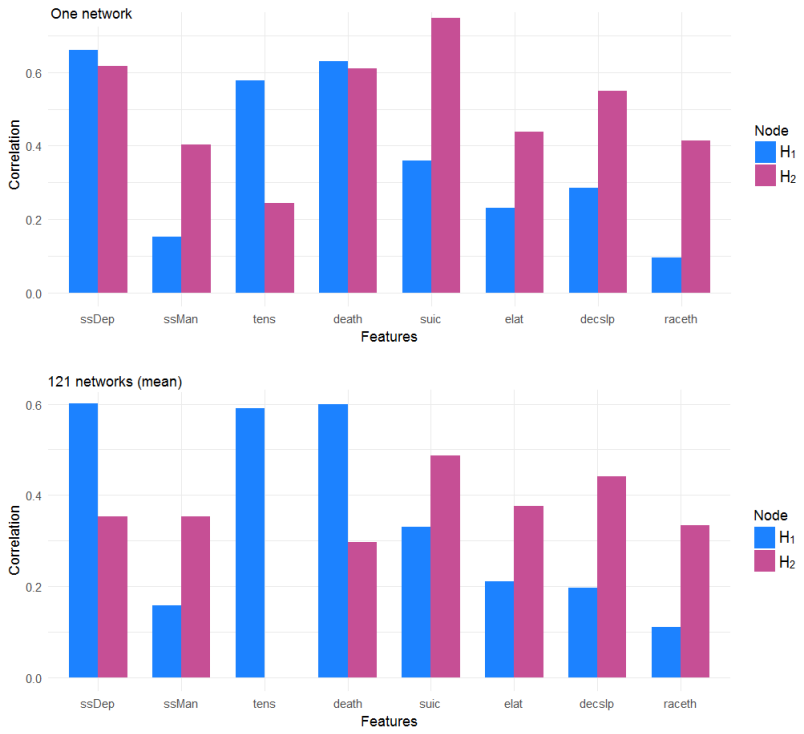


Figure 5.6: Correlation of hidden node outputs with the feature inputs. The upper figure shows the values for the specific network shown in (a) and the lower figure the mean values over all 121 networks.

In the featured model (top figure of 5.6), both nodes focused on features linked to depression (sum score depression and death thoughts). The first node (H_1) was also influenced strongly by tension, which is linked to mood disorders as well (Mesman et al. [16]). Besides the mood features, the second node (H_2) focused on decreased need for sleep, racing thoughts, elation, suicide ideation and the mania sum score more than the first node did.

This pattern can be found averaged over all models as well (bottom figure of 5.6). The first node focused on the mood features more than on the others, while the second node tended to incorporate all features except for tension. The second node outputs correlated stronger with the bipolar features than those of the first node. Seemingly, the hidden nodes code more or less in the same way as the output nodes, suggesting that the hidden layer does not add much value to the networks. All differences between H_1 and H_2 were significant ($p < 0.05$).

In the displayed network (fig 5.5), the weights between the hidden layer and output layer reveal that each output node relied heavily on a single hidden node, making the hidden layer seem superfluous in this particular model. Over all 121 models, the mean of the lowest weight between the hidden and output layer in each network was 0.19, while the means over the second lowest, second highest and highest weight were 0.86, 1.72 and 2.32 respectively (standard deviations of 0.12, 0.63, 0.62, 0.91). These differences are all significant. It shows that all networks had at least one weight out of the four that hardly influenced the output node compared to the other weights. The second lowest weight was very low in some networks as well, as is clear from its standard deviation. In 87 out of the 121 networks, the two lowest weights were attached to connections coming from a different hidden node, implying that the each output node relied mostly on a single hidden node instead of both.

Posthoc, another model was trained on the same features and parameters, leaving out the hidden layer completely. The results can be found in the appendix (tables A.12 and A.13). The model performed similar on the unipolar mood and healthy class (one more mood subject was classified correctly), but classified two bipolar patients as unipolar that were classified correctly in the hidden layer model, making the balanced accuracy drop a little. Without the diagnosed patients, a balanced accuracy of .562 and an accuracy of .625 was reached.

5.3.3 Non-linearity

In feature space

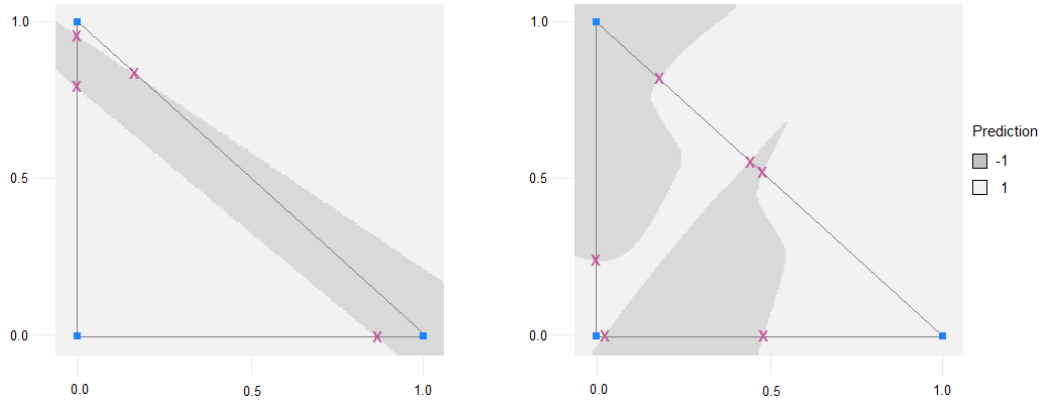
	ensembles (11)			all models (121)		
	wPerc	wPerc0	frac	wPerc	wPerc0	frac
Mood						
Lin	0.000	0.000	0/11	0.000	0.000	0/121
RBF	0.000	0.000	0/11	0.005	0.322	2/121
ANN	1.141	1.245	11/11	3.040	7.508	49/121
Bipolar						
Lin	0.000	0.000	0/11	0.000	0.000	0/121
RBF	0.182	0.334	5/11	0.670	1.530	55/121
ANN	0.279	1.533	2/11	0.856	2.073	50/121

Table 5.8: Degree of non-linearity in models' feature space. wPerc is the mean balanced percentage of multicrossing lines, wPerc0 is the mean taken over those percentages greater than zero and frac is the fraction of models that had any multicrossing line. The results are shown for the 11 ensembles and 121 single models separately.

In table 5.8 the results of examining the degree of non-linearity in the feature spaces are shown. Naturally, all linear models show no non-linearity at all. In the mood models, the RBF SVMs hardly show any either. In the ensemble none was found, and of all 121 models only two contained multicrossing lines. The ANN models score high percentages, and all

ensemble models contained non-linear separations. In the bipolar models, both the RBF and ANN models contained considerable amounts of non-linearity, though the fractions of non-linear ANN ensembles drops.

It applies to all single model fractions that the majority of the models contained no multicrossing lines at all. Even the wPerc0's, that only consider models with multicrossing lines, hardly rise over 2 percent except for the mood neural network.



(a) One of the lines between the the points is multicrossing (single mood ANN).

(b) Two multicrossing lines, of which one crosses three times (ensemble mood ANN).

Figure 5.7: Planes through feature space containing, three data points. The planes were scaled such that the vector points lied at (0,0), (0,1) and (1,0). The dark gray areas were labeled -1 (healthy), the light areas +1 (patients).

In classifications

The mean accuracies of linear, high-C SVMs trained on the classifications of all single models and ensembles are shown in table 5.9. An accuracy lower than one reveals that the high-C SVM was not able to separate the classified points perfectly, suggesting that the dichotomy contained non-linearity.

Though non-linearity in the classification of single linear models is impossible, some was found on the outer loop predictions of the 11 runs. Since they are each composed of 11 models, single points may be classified differently by these models, yielding possible non-linear classifications. The same goes for the true ensemble, which in fact showed only a non-linear classification for the linear mood ensemble.

The data accuracies approximate how linearly separable the data were to begin with. The number is different for the various models since they were trained on different feature sets. These accuracies correlate fairly strongly with the the 11 run accuracies (0.736) and the accuracies of the 120 models (0.650). Comparing the data accuracies to the percentages in table 5.8, high correlations with the wPerc columns were found as well (-0.620 and -0.621). This suggests that the choice of the feature set influenced the amount of non-linearity that was used by the models.

The mood neural network had the most submodels with non-linear classifications and also showed relatively low accuracies. Since this was not the case for the bipolar network, it is probably due to its data being the least linearly separable (together with the ANN).

	runs (11)		models (121)		ens (1)	data	unique
	acc	frac	acc	frac	acc	acc	points
Mood							
Lin	0.958	11/11	1.000	0/121	0.975	0.839	60
RBF	0.997	3/11	1.000	0/121	1.000	0.805	22
ANN	0.910	11/11	0.975	33/121	1.000	0.763	90
Bipolar							
Lin	0.992	7/11	1.000	0/121	1.000	0.919	44
RBF	0.991	7/11	1.000	2/121	1.000	0.892	24
ANN	1.000	0/11	0.999	5/121	1.000	0.892	11

Table 5.9: Results of training linear SVMs on the data that the models were trained on, with the models' predictions as correct labels. A high cost parameter ($C = 10000$) was used to approximate a hard-margin SVM. The fraction (frac) shows the number of models that had an accuracy lower than one, implying a non-linearly separable classification. The unique points are the number of unique data points that were present the data. This number varies for different feature sets. A lower accuracy (acc) suggests a less linearly separable classification.

5.3.4 Same feature models

Extensive results of the models all trained on the same feature set are presented in the appendix (tables A.14, A.15, A.16 for performance, feature space non-linearity and classification non-linearity respectively). As expected, the accuracies were slightly poorer than that of the models that had their own unique feature sets. The neural network and RBF had a better balanced accuracy than the linear SVM, but only after the threshold adaptation. Overall, the performances between the approaches are similar, especially with the default threshold.

The same-feature RBF and ANN models scored higher on most of the non-linearity values (in the feature space analysis) than the regular mood models. An increase of non-linearity use compared to the regular models is visible, possibly caused by the increased number of unique data points (from 22 to 62 for the RBF for example). The same pattern is detectable in the analysis of the non-linearity in the models' classifications.

5.4 Discussion

5.4.1 Performances

The mood classifiers were able to predict with balanced accuracies in the range of 71-77%, while bipolar disorder was classified 76-80% correctly (excluding the linear SVM which scored 71% caused by a large class imbalance). Threshold adaptations (discussed more thoroughly in section 5.4.3) brought the balanced accuracies to the higher ends of these ranges, but these values may be too optimistic. No overall patterns of remarkable differences in performance between the linear SVMs and the non-linear approaches were found. In the bipolar models, the non-linear methods did improve the performance, but only when the thresholds remained unaltered.

To our knowledge, this is the first study to apply machine learning techniques to predict prognosis of bipolar offspring. There are no similar studies to compare the performances with. To get a realistic idea of the performance in practical use, a completely independent test set is required. Acquiring such a set is difficult given the specifics and longitudinal nature of this study. Accuracies are likely to drop on an independent test set due to between-sample heterogeneity and may vary more due to sampling effects (Schnack and Kahn [17]).

The observed accuracies may be high enough to classify which offspring has a higher risk of developing mood or bipolar disorder in the future. Psychiatrists may want to monitor

these children more cautiously. To use the models to decide on immediate treatment like medication, very high specificities nearing 100% are vital to make sure that participants who are not going to develop the disorder are not put through unnecessary treatment. The thresholds can be adapted to reach higher specificities, but the sensitivities will drop. In the best bipolar classifier (ANN), only two patients were correctly identified when no false positives were allowed. The specificities were fairly high in the bipolar models in table 5.5, but this is partly due to the imbalance between the number of controls and patients. If these models would decide on providing treatment, at least 40% of the treated participants would get medication needlessly. Note, however, that patients that did not develop bipolar disorder before the 12-year follow-up may still get a diagnosis later in life.

When considering the multiclass network as a binary classifier (by combining classes), bipolar disorder was classified with 71% balanced accuracy and mood disorder with 62% (leaving out all bipolar and mood baseline-cases respectively). This is considerably lower than the individual models' performances, so it seems that the extra information that was given to the model did not help to improve the classification performance of these individual classes. However, deciding between three classes simultaneously is a more complex task than binary classification, leaving more room for errors. It may not be fair to compare the accuracy directly to the binary models. Also, the chosen encoding for the labels in 2D space (fig 5.2) may not correspond to reality. Perhaps BD should be closer to HE instead of MD[1], or a completely different network architecture would be more suitable. A network with three outputs could be attempted, selecting the highest outputting node as the predicted label.

The same-feature mood classifiers performed worse than the regular mood ones, with balanced accuracy ranges of 65%-71%. These models did not reveal a best performing model either; the RBF scored highest after a threshold move while the linear model performed best on the default threshold. This shows that the use of different features for the three approaches was not the sole reason that no substantial difference between the methods was found.

5.4.2 Leaving out subjects

A limitation of the data set that was used has to do with the longitudinal nature of the data. The data consists of subjects with various diagnoses already present at t_1, some of which the model is trying to predict at t_2, adding a fair amount of bias. The data also suffers from dropouts. All shown models were trained on data including these baseline-cases, in order to spare the sample size and leave potential heterogeneity in the data. The group of patients that these models are trying to predict can be viewed as a heterogeneous group of people that will have a diagnosis 12 years in the future, of which some are further in their illness trajectory than others. From this point of view, diagnosed patients of t_1 that are already ill are extreme cases on this spectrum. Although including them will generate bias, leaving them all out may fail to catch some characteristics of the disorder.

Additionally, the features that were used to prognose are directly related to the labels that the models are trying to predict. Leaving out diagnosed patients during training means omitting subjects with severe symptoms, making it more difficult for the models to find a pattern. On the other hand, subtle patterns in patients that have not been diagnosed yet are the most valuable in practical use. Further research on the effect of including and excluding diagnosed patients in prognosing models is required.

With the multiclass model it is more complicated to decide which subjects should be considered during evaluation. The most rigorous option in the result section omitted both the BD subjects from t_1 and the MD$^-$1 subjects from t_1 that did not transition to BD. This may be considered too harsh, since some of the mood patients of t_1 did transition to BD in the data set, so they are not trivial to prognose. But since no patients in the data rehabilitated, they are easier to predict than participants that were healthy in the beginning (who have three options instead of two).

Ideally, models should be trained on a data set consisting solely of patients that have not been diagnosed with any disease that is to be predicted. Such a set with decent sample size

is not available yet and is hard to obtain, especially when requirements like having a bipolar parent or being available for follow-ups have to be met.

5.4.3 Threshold adaptation

The solution for the problem of baseline diagnoses that was proposed in this study, was to leave out the baseline-cases in the validation phase. Since mostly all of these cases are predicted correctly due to their severe symptoms, omitting them only lowers the sensitivity, resulting in too little correctly predicted cases. Adapting the threshold can restore this balance, but must be approached with caution. Firstly, selecting the most convenient threshold based on the data that the model is testing on, leads to overfitting. Secondly, the models are fitted in such way that the default threshold value of 0 yields the best generalizability. This goes especially for the margin of the SVMs.

To reduce overfitting, the mean of the two most convenient threshold points was chosen instead of the best one. However, this is a rather arbitrary method, and more research on the effect of improving the sensitivity/specificity balance by adapting the threshold is required. Other options may be omitting the baseline-cases after all and accepting a lowered sample size to make sure the models and results are not biased, or accepting the baseline cases as correctly predicted samples and viewing them as a heterogeneous group in the data. The latter option is problematic when the models are only going to be applied to subjects without any diagnoses, since the sensitivity will be misleading and turn out to be lower in practice.

Instead of compensating for the baseline-cases during testing, it would be better to do this in the training phase. It is possible to include the cases during training, while giving them less strength to update the model. For SVMs this could be done by applying a more forgiving hinge loss function to the baseline-cases, lowering the influence they have on the fitted hyperplane. In the neural networks, it is possible to penalize errors on the baseline-cases less heavily by modifying the error function, so that the weights are be adjusted less strongly for these subjects. Research is required to discover to what extent the subjects diagnosed at baseline should influence the models to get a good balance in the performance.

In the multiclass network results, no threshold adaptation was suggested. It is possible to change the balance in the results by changing the thresholds of at least one of the two output nodes. This is more complex than in the binary models since subjects will be moved from two classes in the direction of the two others (possibly including the empty class). Adapting the threshold of the first output node will move patients back and forth between the healthy/empty and the mood side, while the second output threshold will move them between non-bipolar and bipolar/empty (see figure 5.2).

This mechanic makes it harder to put more emphasis on a certain class, without unwillingly moving subjects to another class as well. The relatively low sensitivities for BD in the multiclass results can be increased by lowering the second output threshold. This will move subjects from MD[1] to BD, but will likely also move healthy subjects to the empty class unintentionally. The presence of a class that is incorrect for any subject complicates it more; a transition to the empty class will always end up in an incorrect classification, while this does not have to be the case for other classes. A different network architecture with an output node for each class may be considered in future research on thresholds in multiclass situations.

5.4.4 Feature selection

In this study, the final features were selected by choosing the models that performed best. Though the differences in performance between various well-performing feature sets were not substantial in this case, in further research feature selection should be carried out in an additional cross validation loop to exclude any bias in the features. For each model that is trained (on the train fold, see figure 2.4), a cross validation procedure should be added in which models with various feature sets are trained and tested. For SVMs it is more complex,

since a tuning loop has to be carried out additionally for each of those models. A disadvantage of this way of feature selection is that it results in K models with different features (for K outer loops). To get an ensemble or final model, a voting system needs to be added to select which feature set should be used.

In both the mood and bipolar models, the best performing linear SVM included more features than the other approaches did. Also, in contrast to the others, it included total sum scores instead of only the symptom feature components. This may indicate that the linear SVM prefers a data set with more distinct points, since the sum scores have more distinct values than their components. Besides, as was explained in chapter 2, linear SVMs suffer less from overfitting due to their limited freedom in fitting. Therefore the linear SVMs were able to include more information without reducing the (cross validated) accuracy. Note that the best multiclass model included the most features (8) of all models, which was probably possible because of the extra label information the network has.

The results show that using non-linear methods on clinical data is feasible and that they are able to perform at least as well as the linear method, but extra features should be added with great caution. For equal sample sizes, the non-linear methods methods should be regulated by training them on less-dimensional feature spaces to achieve similar performances.

5.4.5 Non-linearity

The amount of non-linearity as shown in table 5.9 correlated strongly with the linear separability of the feature set it was trained on. This suggests that the choice of features influenced the amount of freedom that was used by the non-linear methods more than the choice for these approaches itself. Besides, the table revealed that creating an ensemble of linear SVMs may result in a non-linear model. This means that when using ensembles, linear SVMs might also be a resource in tackling heterogeneity. Their ability is still limited to combining linear hyperplanes however, making their freedom smaller than fully non-linear techniques.

The values in the non-linearity tables of the same-feature models (table A.15, A.16) cannot be influenced by differences in the data, but still no clear difference between the RBF and ANN models was found. The same-feature RBF showed considerably more non-linearity than the regular model however, which indicates that a higher amount of unique data points (22 in the regular model as opposed to 62 for the same-feature model) allows for more linearity. In the bipolar offspring data, many features that correlated well with the labels consisted of a small range of distinct values or were even ternary. These ternary features turned out to be the most informative, but reduced different subjects to the same data points in feature space, making many participants indistinguishable for the models.

In the data used for this study, the problem of small-range features could be solved by only using sum scores instead of the components, at the expense of the accuracy. The fact that the models performed best on features that reduce the amount of unique data points may indicate that they prefer data that is more linearly separable to begin with. The mood neural network that included the total iq, of which many distinct values are present in the data, suggests this link between unique data points and non-linearity as well. It had the highest amount of unique data points (90) and by far the highest percentages in the feature space non-linearity (wPerc 3.040, wPerc0 7.508). The fact that the RBFs and ANNs did not perform considerably better than the linear SVMs also indicates that the use of non-linearity did not make a big impact on the results.

As can be seen in the extensive bipolar performance table (A.7) in the appendix, the non-linear methods did outperform the linear SVM in terms of balanced accuracy when all subjects were included. There may be some heterogeneity in the bipolar subjects that is caught by those methods. This effect was not found in the mood table A.2, where the linear SVM performed best.

5.4.6 Further research

In addition to the suggestions that were already made, the following is proposed for further research. Firstly, other approaches in classifying mood and bipolar disorder can be attempted. Models could be trained only on the patients with mood disorder of t_1 to find which of these will also develop bipolar disorder. Leaving out the healthy subjects (which may never develop a disorder) may help the models to focus on early bipolar patterns. However, if subjects diagnosed with BD at t_1 are to be left out to prevent bias, only 29 mood patients of t_1 are available. Preliminary results of such an approach were promising, achieving a balanced accuracy of 80% with an RBF SVM trained on those 29 subjects. This is a slight improvement compared to the bipolar models with unaltered thresholds, but a larger sample size and more extensive training is required to draw conclusions.

Another possibility is to construct a two-step model where firstly a classifier is built to separate healthy participants from mood patients, to subsequently apply a second model to filter the bipolar patients from those classified as mood. The accuracy of the current mood models have to be improved before such a two-step model would be feasible.

The longitudinal nature of the data set was not represented in the features that the models trained on; it was only present in the fact that the labels consisted of diagnoses that were made at 12 years follow-up while the feature were extracted from baseline data. To give the models information on the pace of which symptoms are changing, the features of 1-year follow-up could be added. This may enable the models to distinguish between patients with moderate symptoms that remain stable and those who are worsening quickly. Note that if additional follow-up information is added for every existing feature, the amount of features is doubled. Therefore this method is recommended for the linear methods.

Chapter 6

Conclusion

The application of artificial intelligence techniques to build predictive models on psychiatric data showed promising results. The age predicting models performed at least as well (MAE of 4.3 years) as models in previous studies, regardless of the lower number of features. In follow-up research, the brain age models could be applied to brain volumes of patients suffering from mental disorders to find out whether their brains have aged more rapidly than those of healthy persons. This effect was confirmed by Schnack et al. [10], and could be supported by the brain volume models as well.

The prognosis models trained on the bipolar offspring data achieved encouraging performances as well. Decent accuracies between 70% and 80% were reached, despite specific issues in the data like the baseline-cases, the longitudinal nature and the limited value ranges of features, that complicated the training and testing process. The predictive models may be used to determine which children are at high risk of developing a disorder, so they can be monitored more intensively. To make the models suitable for deciding on treatment, higher specificities need to be reached. Follow-up research could comprise adding new features such as longitudinal variables or merging data sets from several bipolar studies.

Significant performance increases were achieved with the construction of non-linear models on the brain volume data. The curved trend of the gray matter loss was presumably the reason for this gain. On the bipolar offspring data, no convincing performance increases were found using non-linear models as opposed to the linear method. Examination of the RBF SVM and ANN predictions revealed that these methods did use their flexibility to draw non-linear boundaries. This suggests that there was not enough heterogeneity in the data to make a difference.

Likely, there are differences in the extent to which heterogeneity is present in clinical and MRI data. Another possibility for a follow-up study could be to apply the same methods to neuroimaging and look for heterogeneous patterns in the brain instead of clinical variables. Such an approach would pose new challenges, such as the large number of features in MRI data. Feature selection and reduction would play a more important role, to ensure that the RBF SVMs and particularly the neural networks are still feasible to train.

In conclusion, the concept of heterogeneity should always be kept in mind in classification tasks on psychiatric data. Non-linear approaches should be studied and applied more often to deal with possible heterogeneity in the data. Caution is needed when the number of features grows or the sample size shrinks, but both data studies demonstrated that non-linear machine learning techniques are able to predict at least as well and possibly better than the linear method, even when sample sizes are relatively small. This provides hope for the future, as these non-linear techniques hold the power to fit to the heterogeneity in psychiatric diseases and thereby tackle one of the main issues that stands in the way of applying machine learning classifiers in medical practice.

References

[1] M.S. Van der Knaap et al. “Pattern recognition in magnetic resonance imaging of white matter disorders in children and young adults”. In: *Neuroradiology* 33.6 (1991), pp. 478–493.

[2] M. Maes et al. “A clinical and biological validation of the DSM-III melancholia diagnosis in men: Results of pattern recognition methods”. In: *Journal of Psychiatric Research* 26.3 (1992), pp. 183 –196.

[3] D.A. Regier et al. “DSM-5 field trials in the United States and Canada, Part II: test-retest reliability of selected categorical diagnoses”. In: *American journal of psychiatry* 170.1 (2013), pp. 59–70.

[4] T. Wolfers et al. “From estimating activation locality to predicting disorder: A review of pattern recognition for neuroimaging-based psychiatric diagnostics”. In: *Neuroscience and Biobehavioral Reviews* 57 (2015), pp. 328–349.

[5] B. Boser, I. Guyon, and V. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: COLT ’92 (1992), pp. 144–152.

[6] P. Werbos. “Beyond regression: new tools for prediction and analysis in the Behavioral Sciences”. PhD thesis. Cambridge, MA: Harvard University, 1974.

[7] E. Varol, A. Sotiras, and C. Davatzikos. “HYDRA: Revealing heterogeneity of imaging and genetic patterns through a multiple max-margin discriminative analysis framework”. In: *NeuroImage* (2016).

[8] A.F. Marquand et al. “Understanding heterogeneity in clinical cohorts using normative models: beyond case-control studies”. In: *Biological Psychiatry* 80.7 (2016), 552–561.

[9] M. Riedmiller and H. Braun. “A direct adaptive method for faster Backpropagation Learning: the RPROP Algorithm”. In: *IEEE International Conference on Neural Networks* (1993), pp. 586–591.

[10] H.G. Schnack et al. “Accelerated Brain Aging in Schizophrenia: A Longitudinal Pattern Recognition Study”. In: *American Journal of Psychiatry* 173.6 (2016), pp. 607–616.

[11] A.F Fotenos et al. “Normative estimates of cross-sectional and longitudinal brain volume decline in aging and AD”. In: *Neurology* 22.64 (2005).

[12] K. Franke et al. “Estimating the age of healthy subjects from T1-weighted MRI scans using kernel methods: Exploring the influence of various parameters”. In: *Neuroimage* 50 (2010), pp. 883–892.

[13] N. Koutsouleris et al. “Accelerated brain aging in schizophrenia and beyond: a neuroanatomical marker of psychiatric disorders”. In: *Schizophrenia Bulletin* 40.5 (2014), pp. 1140–1153.

[14] E. Mesman et al. “The Dutch Bipolar Offspring Study: 12-Year follow-Up”. In: *American Journal of Psychiatry* 170.5 (2013), pp. 542–549.

[15] M. Wals et al. “Prevalence of psychopathology in children of a bipolar parent”. In: *Journal of the American Academy of Child & Adolescent Psychiatry* 40 (9 2001), pp. 1094–1102.

[16] E. Mesman et al. “Baseline dimensional psychopathology and future mood disorder onset: findings from the Dutch Bipolar Offspring Study”. In: *Acta Psychiatrica Scandinavica* (2017).

[17] H.G. Schnack and R.S. Kahn. “Detecting neuroimaging biomarkers for psychiatric disorders: sample size matters”. In: *Frontiers in Psychiatry* 7.50 (2016).

Appendix A

Additional results

A.1 Features

Feature	Description	Range	Mean	SD
age	age at baseline (in years)	11.7-21.0	16.09	2.77
sex	sex	0-1	0.54	0.50
tiq	total intelligence quotient	72-152	114.10	14.89
parbd	bipolar type of parent (I or II)	0-1	0.26	0.44
parsx	sex of bipolar parent	0-1	0.41	0.49
depld	depression load in family	0.5-6.2	1.66	1.25
subld	substance use load in family	-1.6-2.4	-0.76	0.60
ses	social economic status	1-9	4.90	2.04
ssDep	sum score depression	21-50	25.49	6.05
ssMan	sum score mania	4-16	4.65	1.81
ssExt	sum score externalizing behavior	12-22	13.91	2.59
ssAnx	sum score anxiety	23-39	27.77	4.02
ssSub	sum score substance use	6-13	7.78	1.92
ssPsy	sum score psychosis	12-19	13.13	1.62
selfc	marked self-consciousness	1-3	1.35	0.61
tens	marked feeling of tension/unable to relax (ssAnx)	1-3	1.46	0.66
death	recurrent thoughts of death (ssDep)	1-3	1.43	0.67
suic	suicide ideation (ssDep)	1-3	1.20	0.54
insom	middle insomnia (ssMan)	1-3	1.11	0.40
elat	elation, expansive mood (ssMan)	1-3	1.18	0.45
decslp	decreased need for sleep (ssMan)	1-3	1.13	0.41
raceth	racing thoughts (ssMan)	1-3	1.15	0.45

Table A.1: All features that were considered for the models trained on the bipolar offspring data set. Features where the ranges are listed with integers, do in fact only contain integer values. The sum score features are sums of symptoms features (with possible values of 1, 2 and 3). The data in this table was extracted from the BD set ($N = 111$) as shown in figure 5.1a.

A.2 Predicting Mood Disorders

	Thresh	wAcc	Acc	Sens	Spec	AUC	ssDep	ssMan	tiq	selfc	tens	death	decslp
Linear SVM													
all subs	.	.812	.805	.765	.860	.876	35.1	18.5	.	.	23.8	18.9	3.7
	.028	.812	.805	.765	.860	.876							
no bd subs	.	.730	.763	.600	.860	.777							
	-.061	.743	.763	.667	.820	.777							
RBF SVM													
all subs	.	.807	.805	.794	.820	.820	.	.	.	✓	✓	✓	.
	.224	.807	.805	.794	.820	.820							
no bd subs	.	.727	.750	.633	.820	.702							
	-.122	.727	.750	.633	.820	.702							
Neural Network													
all subs	.	.772	.780	.824	.720	.809	.	.	8.5	13.4	22.1	56.0	.
	.636	.805	.818	.735	.900	.809							
no bd subs	.	.710	.713	.700	.720	.745							
	.636	.767	.800	.633	.900	.745							

Table A.2: Performances mood ensembles (including results on all subjects).

A.3 Linear model (MD)

	Train			Validation				
Cost	Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.500	.805	.794	.820	.806	.799	.811	.820	.812
.500	.830	.813	.852	.798	.768	.848	.802	.795
.500	.836	.816	.864	.807	.799	.830	.807	.801
.750	.834	.818	.856	.790	.768	.833	.792	.786
.500	.831	.818	.850	.798	.805	.800	.797	.792
.500	.828	.813	.848	.798	.786	.815	.800	.793
.250	.833	.812	.863	.815	.786	.864	.820	.812
.500	.831	.804	.866	.782	.768	.808	.782	.776
.500	.830	.804	.866	.790	.786	.811	.790	.784
.500	.830	.812	.854	.807	.805	.83	.807	.801
.250	.825	.790	.872	.798	.768	.848	.802	.795
.500	.828	.809	.856	.799	.785	.827	.802	.795

Table A.3: Mean results of 11 linear SVM model runs. For the parameters, the median was taken. The bottom row shows the means over this table, with again the median value for the parameters.

A.3.1 RBF model (MD)

		Train			Validation				
Cost	γ	Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.250	.250	.829	.859	.787	.764	.843	.683	.807	.801
.500	.250	.805	.794	.820	.806	.799	.811	.807	.801
.250	.250	.803	.785	.826	.798	.781	.811	.800	.793
.500	.250	.805	.794	.820	.806	.799	.811	.807	.801
.500	.250	.805	.794	.820	.806	.799	.811	.807	.801
.250	.250	.803	.796	.812	.806	.799	.811	.807	.801
.500	.250	.803	.787	.826	.789	.763	.811	.792	.786
.500	.250	.805	.794	.820	.806	.799	.811	.807	.801
.250	.250	.803	.796	.814	.798	.799	.788	.797	.792
.250	.250	.802	.784	.826	.806	.799	.811	.807	.801
.250	.250	.805	.794	.820	.806	.799	.811	.807	.801
.250	.250	.806	.798	.817	.799	.798	.797	.804	.798

Table A.4: Mean results of 11 RBF SVM model runs. For the parameters, the median was taken. The bottom row shows the means over this table, with again the median value for the parameters.

A.3.2 ANN model (MD)

Train			Validation				
Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.901	.765	.926	.883	.727	.914	.764	.766
.828	.862	.781	.773	.831	.727	.762	.767
.825	.854	.784	.747	.781	.720	.740	.740
.832	.870	.779	.740	.831	.658	.722	.734
.828	.859	.785	.774	.812	.742	.764	.766
.835	.856	.806	.747	.796	.705	.737	.740
.835	.854	.808	.764	.827	.705	.752	.758
.832	.857	.798	.748	.794	.703	.737	.740
.836	.840	.832	.772	.815	.742	.764	.766
.841	.843	.839	.789	.827	.758	.782	.783
.829	.859	.787	.764	.843	.683	.749	.760
.838	.847	.811	.773	.808	.732	.752	.756

Table A.5: Mean results of 11 ANN SVM model runs. The bottom row shows the means over this table.

A.3.3 Example of a linear model (MD)

	ssDep	ssMan	tens	death	decslp	b
weights	2.422	-1.283	1.049	0.721	0.187	1.328

Table A.6: A linear model which can be directly applied to predict a data point, by taking the dot product of the feature weights (*ssDep* to *decslp*) with the data point vector and adding the intercept b. This was a linear mood model trained in the first cross validation fold of the first run. It was selected because its validation accuracy was equal to the median value of all 121 models' validation accuracy.
These five features were all measured on a scale 1-3. They were normalized by subtracting the mean and dividing by the standard deviation of the MD set ($n = 118$, fig 5.1a)

A.4 Predicting Bipolar Disorder

	Thresh	wAcc	Acc	Sens	Spec	AUC	ssDep	ssMan	suged	drsla	uitbu	versb	versn
Linear SVM													
all subs	.	.773	.901	.588	.957	.862	20.9	8.7	29.7	.	15.8	18.0	6.9
	-.586	.827	.829	.824	.830	.862							
no bd subs	.	.709	.897	.462	.957	.822							
	-.586	.800	.822	.769	.830	.822							
RBF SVM													
all subs	.	.810	.883	.706	.915	.895	.	.	✓	✓	✓	✓	.
	-.385	.859	.883	.824	.894	.895							
no bd subs	.	.765	.879	.615	.915	.870							
	-.602	.800	.822	.769	.830	.870							
Neural Network													
all subs	.	.840	.892	.765	.915	.777	.	.	41.9	.	32.7	25.4	.
	-.445	.840	.892	.765	.915	.777							
no bd subs	.	.804	.888	.692	.915	.715							
	-.445	.804	.888	.692	.915	.715							

Table A.7: Performances bipolar ensembles (including results on all subjects).

A.4.1 Linear model (BD)

	Train			Validation				
Cost	Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.0050	.914	.659	.961	.912	.682	.958	.802	.835
.0100	.913	.678	.955	.902	.682	.947	.797	.812
.0050	.911	.677	.953	.893	.682	.937	.792	.792
.0075	.911	.677	.953	.912	.682	.958	.802	.835
.0050	.912	.617	.965	.893	.591	.958	.743	.805
.0075	.909	.666	.953	.893	.591	.948	.768	.797
.0075	.914	.665	.959	.902	.636	.958	.773	.821
.0075	.910	.640	.958	.902	.636	.958	.773	.821
.0050	.912	.624	.964	.902	.636	.958	.773	.821
.0050	.911	.677	.953	.912	.682	.958	.802	.835
.0050	.914	.659	.961	.912	.682	.958	.802	.835
.005	.912	.658	.958	.903	.653	.954	.784	.819

Table A.8: Mean results of 11 linear SVM model runs. For the parameters, the median was taken. The bottom row shows the means over this table, with again the median value for the parameters.

A.4.2 RBF model (BD)

		Train			Validation				
Cost	γ	Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.250	.025	.903	.725	.935	.884	.682	.924	.845	.803
.100	.050	.904	.765	.929	.875	.727	.903	.805	.758
.500	.025	.900	.759	.926	.902	.818	.924	.845	.803
.100	.025	.902	.759	.928	.893	.818	.913	.840	.787
.250	.025	.907	.754	.935	.893	.773	.924	.816	.789
.250	.050	.903	.782	.924	.875	.727	.903	.805	.758
.100	.050	.901	.759	.927	.884	.727	.924	.786	.773
.250	.025	.904	.754	.931	.875	.682	.913	.781	.757
.075	.050	.902	.753	.929	.866	.727	.903	.776	.742
.075	.050	.900	.725	.932	.884	.727	.914	.810	.773
.100	.025	.903	.725	.935	.884	.682	.924	.786	.773
.100	.025	.903	.751	.930	.883	.735	.915	.809	.774

Table A.9: Mean results of 11 RBF SVM model runs. For the parameters, the median was taken. The bottom row shows the means over this table, with again the median value for the parameters.

A.4.3 ANN model (BD)

Train			Validation				
Acc	Sens	Spec	Acc	Sens	Spec	bAcc	AUC
.903	.725	.935	.884	.682	.924	.810	.773
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.883	.727	.914	.810	.773
.901	.765	.926	.883	.727	.914	.810	.773
.901	.765	.926	.883	.727	.914	.810	.773
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.893	.773	.914	.840	.787
.901	.765	.926	.883	.727	.914	.810	.773
.901	.761	.927	.889	.748	.915	.826	.781

Table A.10: Mean results of 11 ANN SVM model runs. The bottom row shows the means over this table.

A.4.4 Example of a linear model (BD)

	ssDep	ssMan	suic	elat	decslp	raceth	b
weights	0.187	0.106	0.253	0.173	0.185	0.078	-0.608

Table A.11: A linear model which can be directly applied to predict a data point, by taking the dot product of the feature weights (*ssDep* to *raceth*) with the data point features and adding the intercept b. This was a linear BD model trained on the second cross validation fold of the first run. It was selected because its validation accuracy was equal to the median value of all 121 models' validation accuracy.
The ssDep values in the data were ranged from 21 to 50, and the other five features were all measured on a scale 1-3. They were all normalized by subtracting the mean and dividing by the standard deviation of the BD set ($n = 111$, fig 5.1b).

A.5 Multiclass model (no hidden layer)

	bAcc	Acc	SensBD	SpecBD	SensMD^1	SpecMD^1	SensHE	SpecHE
all subs	.617	.658	.471	.926	.682	.731	.700	.803
no BD$_{t_1}$ subs	.589	.654	.385	.926	.682	.714	.700	.789
no BD$_{t_1}$, no MD$^1_{t_1,t_2}$ subs	.562	.625	.385	.960	.600	.714	.700	.684
	ssDep	ssMan	tens	death	suic	elat	decslp	raceth
mean scaled absolute weights	17.3	16.7	10.9	8.9	15.3	7.3	14.4	9.2

Table A.12: Performances and weights (scaled to a sum of 100) of multiclass model consisting of networks without hidden layer, as opposed to the results in 5.6.

		True		
		BD	MD^1	HE
Pred	BD	8	5	2
	MD^1	5	30	13
	HE	3	9	35
	–	1	0	0

(a) All subs

		True		
		BD	MD^1	HE
Pred	BD	5	5	2
	MD^1	5	30	13
	HE	3	9	35

(b) No BD$_{t_1}$

		True		
		BD	MD^1	HE
Pred	BD	5	1	2
	MD^1	5	15	13
	HE	3	9	35

(c) No BD$_{t_1}$ & no MD$^1_{t_1}$ in the MD^1 cat.

Table A.13: Confusion matrices of the multiclass model without hidden layer, corresponding to the rows in table A.12.

A.6 Models on same features (mood)

	Thresh	bAcc	Acc	Sens	Spec	AUC	ssDep	seflc	tens	death
Linear SVM										
	.	.697	.738	.533	.860	.757	32.6	17.2	23.5	26.7
	-.788	.653	.650	.667	.640	.757				
RBF SVM										
	.	.683	.713	.567	.800	.742	✓	✓	✓	✓
	-.409	.717	.738	.633	.800	.742				
Neural Network										
	.	.677	.713	.533	.820	.734	49.1	21.5	9.2	20.2
	-.413	.707	.700	.733	.680	.734				

Table A.14: Performances of models trained on an identical feature set (predicting mood disorder). Patients diagnosed with mood disorder at t_1 are omitted in the results.

	ensembles (11)			all models (121)		
	wPerc	wPerc0	frac	wPerc	wPerc0	frac
eqLIN	0.000	0.000	0/11	0.000	0.000	0/121
eqRBF	1.365	1.365	11/11	1.436	2.446	71/121
eqANN	1.206	1.896	7/11	0.589	1.047	68/121

Table A.15: Degree of non-linearity in feature space, of models trained on an identical set of features. wPerc is the mean balanced percentage of multicrossing lines, wPerc0 is the mean over the those percentages greater than zero and frac is the fraction of models that had any multicrossing line. The results are shown for the 11 ensembles and 121 single models separately.

	runs (11)		models (121)		ens (1)	data
	acc	frac	acc	frac	acc	acc
eqLIN	0.976	11/11	1.000	0/121	1.000	0.805
eqRBF	0.965	11/11	0.985	37/121	0.975	0.805
eqANN	0.935	11/11	0.992	43/121	0.932	0.805

Table A.16: Results of training linear SVMs on the data the models with identical feature sets were trained on, with the models' prediction as correct labels. A high cost parameter ($C = 10000$) was used to approximate a hard-margin SVM. The fraction (frac) shows the amount of models that showed an accuracy lower than one, implying a non-linearly separable classification. A lower accuracy (acc) suggests a less linear separation.

Appendix B

NNSVM User Manual

An R framework called NNSVM was built around the pre-existing libraries neuralnet and libsvm (known as e1071) to allow for a more smooth training process. It was specifically designed for this study, with the domain of psychiatry in mind. It combines the implementations of SVMs and neural networks in a coherent interface; functionality to train models with (double nested) cross validation, get analyses of SVM and ANN weights and test ensembles was added. Access to the R script and the full documentation document can be requested from the author or the first supervisor.

This user manual contains a few examples, showing how to use the NNSVM script to train and test neural networks and SVMs using cross validation. This is not a documentation document. For further information and formal descriptions of all parameters and return values, please refer to the documentation document.

It is recommended by the author to use the development environment RStudio[1], which has functionality to view and scroll through data frames and view all parameters in the global environment.

B.1 Loading the script

To start off, make sure the R packages neuralnet[2] and e1071[3] (LIBSVM for R) are installed. This can be done by calling the following commands in R, which is only necessary the first time:

```
> install.packages(neuralnet)
> install.packages(e1071)
```

Load the needed packages:

```
> library(neuralnet)
> library(e1071)
```

Load the allFunctions script by calling the source function with the path to the file (with escaped backslashes):

```
> source(C:\\Users\\Example\\allFunctions.R)
```

All the functions in the script are now available to use. Read the documentation file for an overview of all available functions with their exact use, parameters and return values. The following sections consist of a few examples of how to use the functions to train on data and get results.

[1] https://www.rstudio.com/
[2] https://CRAN.R-project.org/package=neuralnet
[3] https://CRAN.R-project.org/package=e1071

B.2 Read in data

Use the built-in `R` functions `read.table` (for a tab separated text file) or `read.csv` (for a comma separated file) to read in your data file, which must located in the directory of your current path. The path can be inspected and modified with `getwd()` and `setwd()` respectively.

```
> data <- read.csv(‘‘yourData.csv", header = TRUE, sep = ",")
```

To use the NNSVM training functions, it is essential that the first column of the data frame contains the true labels and the other columns represent the features that you would like to train with. If you would like to only use, for example, the first five columns, the seventh column and the ninth column from your data frame, modify your data with (where `c()` stands for concatenation):

```
> data <- data[,c(1:5, 7, 9)]
```

Or just the first 40 rows:

```
> data <- data[1:40,]
```

It can be convenient to store the column with the true labels for later use:

```
> labels <- data[,1]
```

It is recommended to scale the data by hand and keep the `scale` parameters at their default value of `FALSE` during training. Otherwise, the weights produced by the linear SVMs may not be accurate.

B.3 Training

B.3.1 Neural network

This function call trains a neural network with one hidden layer containing three nodes ($h = 3$), using 10-fold cross validation ($cross = 10$). Ten models will be trained on the 9 train folds and validated on the left out folds. In each epoch, training terminates when the derivative of the function decreases with at most 0.001 or when the amount of training iterations exceeds *stepmax*. The type `regression` makes sure the output will be unaltered, as opposed to `classification` which will use the sign function to get -1 and 1 outputs.

```
> nn.model <- nnTrain(data, type="regression", h=3, threshold = 0.001,
stepmax = 400000, cross = 10)
```

The result of the training process is stored in `nn.model`, and consists of four components which can be accessed with the `$` operator. `nn.model$resultTable` returns a data frame with results and information about the 10 cross validation runs, and is automatically printed at the end of the full training run. Call `nn.model$results` to get the means of that result table. `nn.model$nn` contains a list of the 10 trained networks. Use the subscript `[[i]]` to access the different networks:

```
> nn.model$nn[[1]] # first model
> nn.model$nn[[2]] # second model, etc...
```

These models are of type `"nn"` from the `neuralnet` package. See the documentation of the `neuralnet()` function for more information on the values that can be extracted from `nn` objects.

Error function

It is possible to alter the error function in cases of class size imbalances. By default the error function is `function(x,y){(1/2 * (y - x)^2}`, but to penalize errors made on the smaller class more heavily, the function should be multiplied by the proportion of the class sizes (larger class divided by the smaller): `function(x,y){((50/10)^(0.5*(y+1))) * 1/2`

`* (y - x)^2}` for a class with 10 and one with 50 subjects, where the small class has label -1. The exponent of $0.5 * (y + 1)$ is added to ensure the multiplication will only be carried out when the true label is -1. To reverse the labels, use -y in the exponent instead of y.

B.3.2 Support vector machines

To train an SVM, use `svmTrain`. Here are examples for a `linear` and an `RBF` SVM:

```
> svm.model <- svmTrain(data, kernel = "linear", type="regression", cost
= c(1:6), innerCross = 5, outerCross = 10)
```

```
> svm.model <- svmTrain(data, kernel = "radial", type="regression", cost
= c(1:6), gamma=c(0.01, 0.1, 1, 2), innerCross = 5, outerCross = 10)
```

A vector of values for cost C and γ are supplied. These are the values that are tried during the grid search in the inner cross validation loop. *InnerCross* denotes the number of folds in this inner loop, where parameter optimization is carried out.

Analogous to the neural networks, the models can be accessed with:

```
> svm.model$svm[[1]] # first model
> svm.model$svm[[2]] # second model, etc...
```

These models are of type `"svm"` from the `e1071` package. Refer to the documentation of the `svm()` function for more information on the values that can be extracted from them. Additionally, functionality to easily extract weights and the intercept b was added in `NNSVM`:

```
> svm.model$svm[[5]]$weights
          ic        cbgm        cbwm        vlat         sex
0.02805278 -0.08782604 -0.0009609036 0.03464588 0.01460566
```

```
> svm.model$svm[[5]]$b
0.1274092
```

These weights are only meaningful for the `linear` kernel and only when the parameter `scale` was set to `FALSE` during training! Any scaling should be done on the data set by hand to get a sensible result for the weights.

Class weights

When the class sizes are imbalanced, it is possible to add class weights. A higher weight for a class means that errors made on subjects belonging to that class are penalized more heavily, which results in the model working harder to classify them correctly. Usually the ratio between the classes can be used by dividing the size of the bigger class by the size of the smaller class (i.e. a weight of 5 when the sizes are 10 and 50). Supply only the class weight of the class that is to be compensated for, the other will be 1 by default: `> svm.model <- svmTrain(data, ..., classWeights = c("-1" = 5))`

In this example, the class with label -1 gets a weight of 5 and the class with label 1 will get the default weight of 1.

B.4 Predicting

Now we want to see how the trained models perform. In these examples the `nn.model` is used, but predicting and performance measuring works likewise for the `svm.model`. To predict with a single model (the first one) from the `nn.model` list, do:

```
> model.pred <- predictModel(model$nn[[1]], data[,-1])
```

Note that we omit the first column from the data, which contains the correct labels and should not be provided in the testing phase. For SVMs, it is possible to get the decision values instead of the sign values by adding `decisionValues=TRUE` as a parameter.

To request a vector of the predictions of each subjects that were made on the left out folds during cross validation:

```
> valPred <- nn.model$valPred
```

It is also possible to predict with the ensemble of the 10 trained CV models. Such an ensemble will make a prediction with each provided model, and return the mean of these prediction as a final value. It is also possible to take the median instead by adding the parameter `fun="median"`.

```
> nn.pred <- predictEnsemble(nn.model$nn)
```

B.5 Performance

The variable `nn.pred` contains a vector now with the predictions for each row in the data. To get an idea of the model performance, we can use the following functions to get the RMSE, MAE, correlation and R^2 measures. Note that these measures are for regression models. The model and true labels must be supplied:

```
> rmse(nn.pred, data[,1])
6.3112531
> mae(nn.pred, data[,1])
4.683251
> cor(nn.pred, data[,1])
0.8663163
> r2(nn.pred, data[,1])
0.7505039
```

If you trained a classification model instead of using regression, you can use the *confusion matrix* to get performances, with the function `predTable`. Supply the predictions followed by the true labels.

```
> nn.tab <- predTable(nn.pred, data[,1])
> nn.tab
      true
 pred   1   -1
   1    7    5
  -1    3   35
> acc(nn.tab)
0.84
> bAcc(nn.tab)
0.7875
```

Use the functions `acc` for the normal accuracy (diagonal divided by total) and `bAcc` for a balanced accuracy, where each class influences the performance equally. The class with label 1 was predicted less accurately in this example, and it is also the smallest class. Therefore the balanced accuracy is lower than the unbalanced version.

B.6 Suggestions for parameters

One of the only ways we can influence the training process, is by choosing the parameters well. The best choice of parameters depends heavily on the data set. Here are a few rules of thumb, but these should be used with great caution. Always experiment by using cross validation and comparing results on the validation folds.

B.6.1 Neural networks

- `Threshold` The network will keep training until the derivative of the error function does not decrease with more than the threshold value in percentage. If this constraint is not met before the stepmax is reached, the network will not be labeled as 'converged' and will be discarded and trained again. Therefore this value is very important! Always try a few values to see how long it takes before a certain threshold is reached. Sensible values are in the order of 0.0001, 0.001 or 0.01. The lower the value, the more time is needed, thus a higher stepmax will be necessary too. When adding more hidden nodes, this value should be lowered to achieve the same training duration as before.

- `Stepmax` This is the maximum amount of steps the algorithm will take. If the threshold is not reached before, the network will be discarded and retrained. Choose a high value to make sure the network has enough time to converge, but extreme values will make the process very lengthy. Values between 100000 and 1000000 are common, but for bigger networks with more than 3 to 5 nodes, the values may need to be much bigger to ensure convergence.

- `H` The optimal number of hidden nodes depends heavily on the amount of inputs (features) in the data set. Typically, it is lower than the number of inputs and higher than the number of outputs. The `NNSVM` script was intended for relatively small networks with hidden node amounts between 2 and 10, but feel free to experiment. It is possible to provide a vector (like c(4, 10)) to get a second layer, but attempting this is at the user's risk and has not been tested thoroughly. Some functionality may fail.

- `InnerCross and OuterCross` For the number of inner CV folds, a value between 5 and 10 seems sensible. Leave-one-out CV is also possible. The same goes for the number of outer folds, though it is typically equal or greater than the number of inner folds.

B.6.2 Support Vector Machines

- `Cost` The cost parameter influences the size of the margin in the SVM. Large values of C will resemble a hard-margin SVM, allowing as less misclassifications as possible, but typically also resulting in overfitting. A commonly used value is 1, but the cost parameter can be tuned by providing a vector of values. Try a few values closer to zero and larger values as well, since the best value depends on your data. Start with something like c(0.001, 0.01, 0.1, 1:5) and see which order of magnitude is chosen most often by the algorithm.

- `Gamma` The γ parameter influences how close the RBF SVM will fit to the data. Lower values will result in a looser fit that is more robust to overfitting and higher values may improve the accuracy but at risk for a less generalizable model. Like with C, multiple orders of magnitudes should be tried in the inner cross validation loop.

B.7 Common errors

There are a few common error situations that can occur while using the `NNSVM` script or the `neuralnet` and `e1071` packages. Watch out for these situations:

- `Error in neurons[[i]] %*% weights[[i]] : non-conformable arguments`
  This occurs when you provide a data frame which differs from the data that the neural network was trained on. Make sure the column with the true is omitted for predicting, and that it is present for training. Also make sure that the data set contains the same features in the same order as in the training phase.

- `Error in svm$coefs : object of type 'closure' is not subsettable`
  This is a similar situation, but for SVMs. It usually means that the provided data frame contains the wrong columns: check whether the label column should be present or if any unneeded columns have been added.

  It may also mean that you are trying to add a column using $ to an object that is of a different type than a data frame (e.g. a matrix). Use `class` to check the type of the object, and use `as.data.frame()` to convert it (back) to a data frame.