Universiteit Leiden
Opleiding Informatica

Finding and visualizing patterns in Borderline Personality

Disorder fMRI images

Name:	Jelle van Mil & Chris Onderwater
Date:	26/07/2016
1st supervisor:	Dr. Ir. F.J. Verbeek
2nd supervisor:	C.C. van Schie, MSc

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Finding and visualizing patterns in Borderline Personality Disorder fMRI images

Jelle van Mil & Chris Onderwater

Abstract

The Borderline Personality Disorder is a mental illness with among other things a great negative impact on the social life of a person. Because of developments in functional MRI it is now possible to search for altered patterns of activation in different parts of the brain.

For this research a number of subjects have been analysed using classification methods. The goal of the analysis is to find differences in brain patterns between diagnosed and undiagnosed subjects. The data we analyse was gathered by performing fMRI scans. During these fMRI scans the subjects received multiple visual stimuli. In this research the data is analysed using a data-driven approach.

Using decision tree classification for analysing the data shows much potential, although it is rather depending on the preprocessing and feature selection. By distributing the processes over a cluster the computational challenges are faced. The practice of using region growing to select areas in the brain is promising and introduces many possibilities. But region growing is not the only interesting topic for further research: using more specific regions and using more advanced classification methods could also increase insight in the Borderline Personality Disorder.

We worked on this BSc project as a team of two people. During the first part of the research we worked strictly together to develop the fundamental processes required for the project, after this we both focused on different parts of the research. The project was done at LIACS in collaboration with Clinical Psychology from Leiden University.

Acknowledgements

This BSc research project has been an interesting and informative time which we could not have done without the help of some people. First of all we would like to thank our supervisor Dr. Ir. Fons Verbeek for the intensive guidance and his help when needed. It was a pleasant experience that we could always walk by his office to discuss the current difficulties regarding the project. We would also like to thank our domain experts Charlotte van Schie and Prof.dr. B.M. Elzinga for the pleasant and valuable collaboration. Without their help and their data this project would not be possible.

The members of the Imaging and Bioinformatics research group have also been of great help, in particular we value the lengthy discussions with Koen Griffioen and Yuanhao Guo. Koen Griffioen also assisted and helped us with the 3D visualizations, which we really appreciate.

As a last note, we are grateful that we could use the LLSC cluster for the computational analysis. Thanks to the LLSC sysadmins who made it possible for us to use the LLSC, in particular Chris Rietveld. Moreover we would like to thank CZ for their support in LLSC.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Borderline Personality Disorder (BPD) is a mental illness that is characterised by mood swings, a negative self image, affect dysregulation and unstable interpersonal relationships. The disorder affects about 1-2% of the general population, up to 10% of the psychiatric outpatients and is more often diagnosed in the female gender. From the patients with BPD up to 10% commit suicide, which is almost 50 times the percentage of the general population [38].

Current focus in the BPD research field revolves around finding neurobiological causes and finding better and more cost-effective treatment methods [38]. What factors are causal to borderline is only partly known, although it is known that genetic factors and adverse events like sexual abuse during the childhood give rise to the development of BPD. Because of developments in functional magnetic resonance imaging (fMRI) it is possible to search for patterns of activation in different parts of the brain, which can help to find neurobiological characteristics of BPD. This creates possibilities to deal with the issue of *representation* [41], the search for areas in the brain that contain information about BPD.

In this multidisciplinary research we will search for areas of similar voxels based on patterns in their internal intensity time series. This research is done in collaboration with domain experts Charlotte van Schie (Msc.) and Prof. dr. B.M. Elzinga from Leiden University Faculty of Social Sciences, department of Clinical Psychology. The study is performed in the research group Imaging and Bioinformatics of LIACS. The fMRI data we analyse are fMRI images gathered by the domain experts during a *social feedback task* where subjects received different kinds of stimuli during an fMRI scan. In this way we can explore if BPD patients in the research group Imaging and Bioinformatics have different brain activation patterns during those moments of exposure than a group of healthy controls.

Conventional fMRI research focuses on analysing individual isolated voxels and performing statistical tests on these voxels to define if it contains significantly more or less activation than normal [41]. This is also the methodology the domain experts use when analysing the scans. By creating a model based on the moments of stimuli and the hymodynamic response function [33], the correspondence between the model and a voxels real behaviour is calculated. In contrast to this methodology, in this multidisciplinary research we will search for areas of similar voxels based on patterns in their internal intensity time series. Our research will therefore explore the possibilities of applying conventional data mining techniques to fMRI data, so that we can define patterns in the brain that show a distinction between a group of BPD patients and a healthy control group.

1.1 Research question

To discover the possibilities of applying data mining techniques to fMRI data, we defined the following research question: *To what extent can non-borderline and borderline subjects be classified into those two groups by applying machine-learning paradigms on their functional MRI data?*

1.2 Sub-questions

In order to answer the research question, we defined the following sub-questions:

1. *Which preprocessing steps are required for proper classification?*
   Classifiers are not always able to cope directly with the original characteristics of a data set. To deal with this it is necessary to identify these characteristics of the data and to properly preprocess the used data.
2. *Which features are relevant?*
   To describe the data multiple features can be calculated. This can be done in multiple ways and some features may have more predictive value than others.
3. *How can we handle the processing of large files?*
   The dataset consists of many dimensions. This has to be reduced to a single set of features. The amount of data that needs to be processed introduces new challenges in managing the computation times.
4. *How can we divide the brain into spatial areas which can be used for classification?*
   To classify the data we need to divide the brain into areas with similar activation patterns.
5. *Do the classification results improve when using more specific areas?*
   Both large and small areas can be used to identify features. We want to identify if there is a difference in results when in using more specific regions.

6. *How can we deal with a large amount of resulting features?*

   Even after aggregating features, many features remain per subject. We want to identify if classifiers perform better if we perform a feature selection step first.

7. *Do the classification results improve when using other classification algorithms than a decision tree?*

   Multiple classification methods exist for multiple purposes. Classification methods that perform significantly better or worse need to be addressed to identify the potential of other classifiers.

8. *How can we interpret and visualize the results?*

   The data consists of many dimensions and the processing consists of many steps. Methods need to be generated to improve the visualization process so that we can improve the understanding of the data and address errors in the process.

1.3 Thesis Overview

In following chapter we will first discuss the materials and methods we used, divided into four domain context aspects, the tools we used and the methodology we followed. After this we will discuss the results, where a standard experiment will be carried out and all the research sub-questions will be investigated. In the last chapters we conclude the research and discuss interesting areas for future work.

Chapter 2

Material & Methods

In this chapter first some context information about borderline, the brain, fMRI and the collection, structure and management of our data is given. After this the tools and methodology we use will be discussed.

2.1 Context

2.1.1 Borderline

Borderline is a personality disorder that is characterized by instability in affect regulation, impulse control, interpersonal relationships [35] and self image [38]. Factors that contribute to the disorder are genetics, but also physical and sexual abuse during childhood. The disorder affects 1-2% of the general population, of which about 10% commits suicide [38]. There are four symtoms in psychopathology that help to identify Borderline patients [38].

1. Disturbed affect. Patients with Borderline disorder experience intense emotions, such as anger, rage, terror. Another characteristic is the intense mood swings patients can experience.

2. Disturbed cognition. This consists of overvalued ideas of being bad, depersonalisation, delusions and hallucinations.

3. Impulsivity. Patients may harm themselves or exhibit other forms of impulsivity such as driving recklessly, eating irregularly or spending sprees.

4. Unstable relationships. One key aspect of this is fear of abandonment. Another aspect is that close relationships are often of low quality.

2.1.2 Brain

The brain functions as the center of the nervous system, it stores and processes information from the outside world that is delivered by multiple input sources. The brain performs a number of important functions: information processing, perception, motor control, arousal, homeostasis, motivation, learning and memory [4].

Anatomy

There are two general types of brain tissue, white matter and grey matter. Grey matter consists of synapses, most of the processing of the brain happens in these areas. White matter consists of fibers and its function is to connect grey matter areas [11].

The brain is commonly divided into three different areas. These are the Cerebrum, the Cerebellum and the brainstem [5]. The brainstem is the most primal part of the brain, which evolutionary developed first. It is connected to the spinal cord. The brain stem is associated with regulating the central nervous system, it maintains consciousness and regulates the sleeping cycle. The Cerebellum is another part of the brain that is located in the lower back. Its most important function is motor control, but it also supports some cognitive functions. Finally there is the Cerebrum. This controls conscious thoughts. The Cerebrum is often divided into four major lobes. These are the Frontal lobe, Parietal lobe, occipital lobe and the temporal lobe. Whereas the frontal lobe handles the most complex thoughts, the parietal lobe processes many sensory data. The Occipital lobe mostly processes sensory information. The limbic system is the part of the brain surrounding the brain stem [16]. This part of the brain is mostly important for emotion, motivation and learning.

Borderline in the brain

Previous research has concluded that in certain areas of the brain Borderline patients have different activation patterns compared to healthy people [42]. Areas that are smaller in BPD patients are the Hippocampus, the Orbitofrontal Cortex, and the Amygdala. The Prefrontal Cortex contains some areas that often are less active in Borderline patients [42].

2.1.3 Functional MRI

In order the understand the Borderline disorder, the functions of different parts in the brain have to be addressed. One of the relatively new solutions to get insight into the brains activity is the functional magnetic resonance imaging technique (fMRI). Functional MRI is based on the same principles as MRI and needs a strong magnetic field to function. Magnetic resonance imaging does not measure the direct neuronal activity, but instead measures a proxy: an estimation of the local blood flow. The fMRI technique relies on the assumption that neuronal activity and cerebral blood flow are correlated. Although this association is the foundation of functional neuroimaging, the physiological basis of this connection is still incompletely understood [30]. A typical output of an fMRI scan is a full 3 dimensional brain image consisting of voxels. A voxel is a sample element in a three dimensional space which contains information [43]. In the case of fMRI, a voxel contains the brains intensity values over time. Those activity values are absolute activity levels which can differ per person. According to [36], a voxel typically contains a few million neurons and tens of billions of synapses. The amount of neurons and synapses per voxel depends on the resolution of the scan and therefore on the real size of the voxels. In our fMRI data the voxel size is 2.75mm (x-axis) by 2.75mm (y-axis) by 3.0250001mm (z-axis).

2.1.4 Data

The raw data used in this research is provided by our domain experts. In this section we will explain the data gathering process and the structure of the data to give more context to the other chapters in this thesis.

Data collection

In research done by our domain experts, 107 women were scanned in a functional MRI scanner during a so called *Social Feedback task*. This research has been approved by the medical ethical committee (METC) of the LUMC. All subjects have given a written consent to the study. These women can be divided in three groups: a group of 37 subjects without the Borderline personality disorder (*Healthy Controls* group), a group of 23 subjects who are characterized by having an above average lack of self-confidence (*Insecure people* group) and a group of 47 subjects which are diagnosed with Borderline personality disorder (*Borderline Personality Disorder* group).

Before the patient lied down in the fMRI scanner the person first got an introduction into the study after which the person was interviewed for a duration of around 10 minutes. After this a questionnaire about their state of self-esteem, anger and tension on a scale from 0 to 100 was filled in by the subjects. During the scan the *Social Feedback task* was performed, see Figure 2.1 for an illustration of the process. In this Social Feedback Task 45 stimuli words from different categories were shown on a screen in front of the subject: 15 negative words, 15 positive words and 15 neutral words. The stimuli words were shown in random order, with the only limitation that the categories should alternate during this task. After each stimuli there was a moment for the person to rate his or her own self-esteem. In total this process of giving stimuli and rating the self-esteem took around 5 minutes in total. After the scan the state of the persons self-esteem, anger and tension was rated again by themselves and the session ended with a debriefing.

This resulted in a data repository of 108 folders (1 example subject folder and 107 real subjects folders). This data was stored on a local hard drive at the FSW building. To access this data we transferred the data using File Transfer Protocol (FTP) to the LLSC (see section 2.2.3) server. The exact structure of this data can be found in cf appendix A.

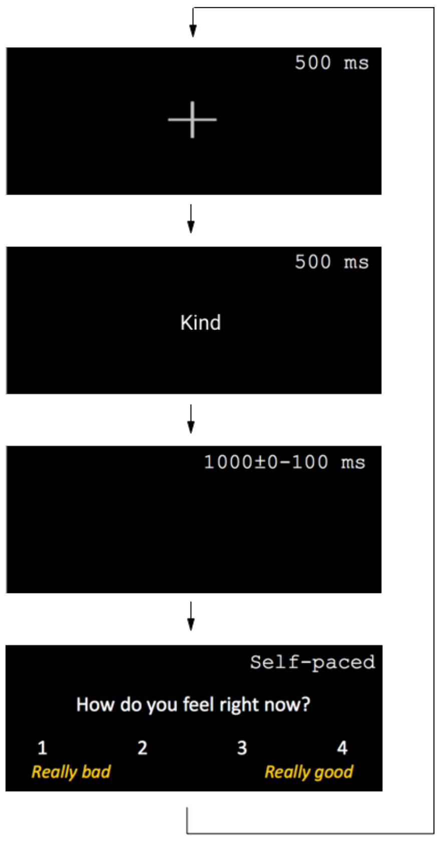
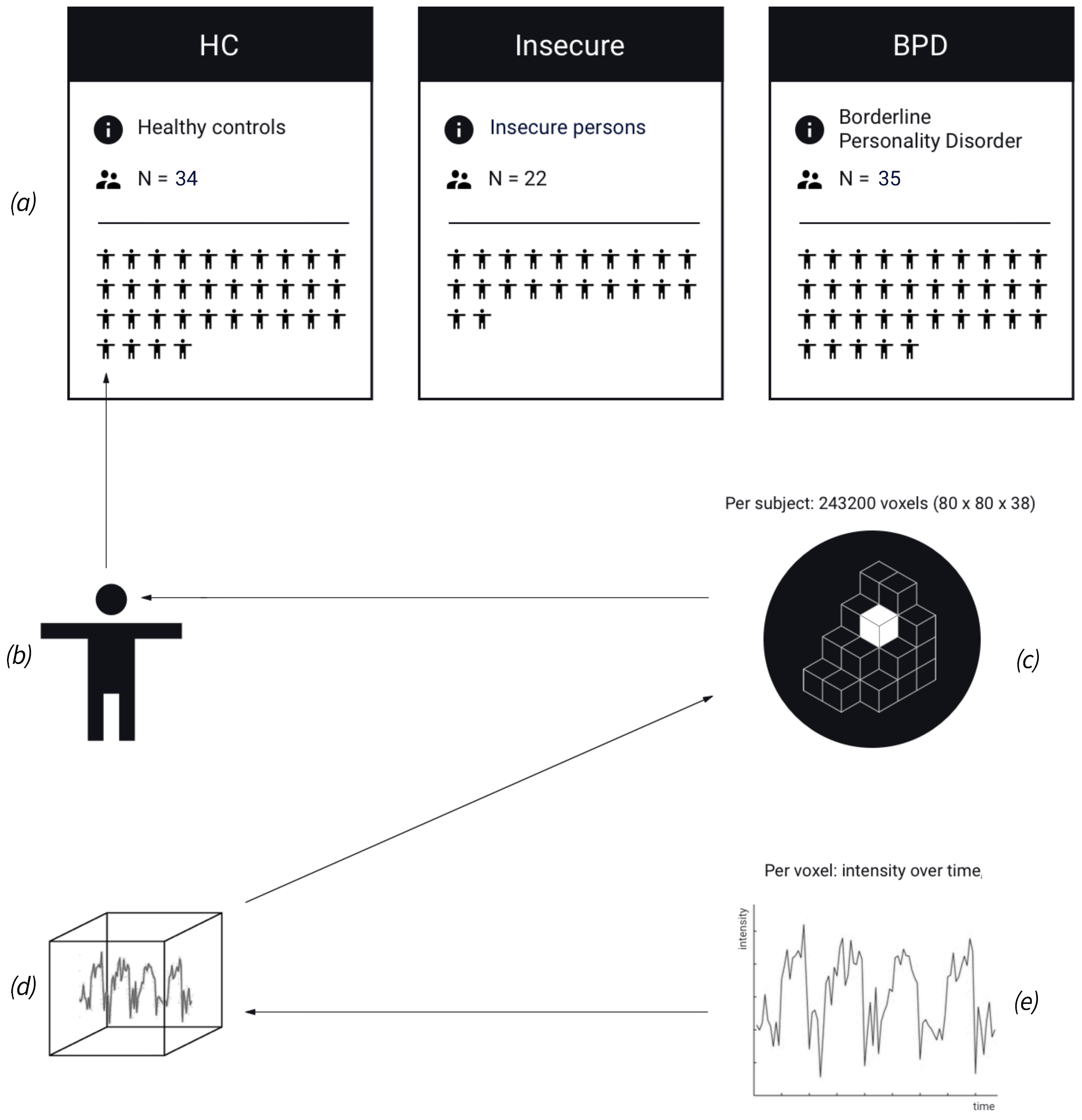


Figure 2.1: Social feedback task process [47]. Adjusted with consent of domain experts.

Data description

The dataset that is used in this study consists of 91 subjects in total, which is less than the 107 mentioned in section 2.1.4. This is due to the fact that not all scans were successfully completed, which in turn can be explained by the fact that some subjects stopped the scanning process prematurely or due to the fact that there was an error with the scanner which resulted in an incomplete scan. The subjects we could not use for the analysis with the corresponding reason can be found cf appendix A.2. A schematic overview of our final data can be seen in Figure 2.2. For every subject we have a functional MRI file. This file consists of 80 voxels on the x-axis, 80 voxels on the y-axis and 38 voxels on the z-axis. Every voxel contains a time series which specifies the absolute intensity values over time.

The fMRI files of our final dataset are in the NifTI-1 file format [20] and range from 56 megabytes to 115 megabytes in size. When using the chopped versions of the files the size ranges from 25 to 70 megabytes. The chopped files differs from the original files in that the time frames where nothing happens in the scan are chopped out. In the NifTI-1 file format the first three dimensions are reserved for the spatial dimensions x, y and z respectively. The fourth dimension is specified for the time-dimension [40]. The data can therefore be represented as a 4-dimensional array. The coordinates used in this paper are based on the NifTI-1 file format coordinates system. These x, y and z coordinate values range from zero to respectively the height, width or length of the brain image.


HC
Healthy controls
N = 34
Insecure
Insecure persons
N = 22
BPD
Borderline Personality Disorder
N = 35
(a)
Per subject: 243200 voxels (80 x 80 x 38)
(b)
(c)
Per voxel: intensity over time
intensity
time
(d)
(e)

Every group (a) consists of multiple subjects. For every subject (b) there is an fMRI scan which consists of multiple voxels (c). Each voxel (d) in this scan contains a time series. This time series (e) describes the intensity over time in that voxel.

Figure 2.2: Overview of the data

Data management

In order to manage the data and to separate all different files, file naming conventions are used by our domain experts cf app A.1. Because the fMRI brain scans can potentially reveal personal and private information a person may not want to know or may not want to be public [45], data privacy is also an important data management aspect of this project. To protect the privacy of the subjects, all scanner files are renamed to include an anonymous identifier which only reveals the identifier number and the group the scan belongs to. To carefully work with the data, we only access the data on a private folder on the LLSC (see section 2.2.3 for more information about the LLSC).

2.2 Tools

For the analysis of the fMRI data we use a number of tools that are publicly available. These tools are selected based on their utility for this project. In most cases we preferred tools that are relatively easy to use and portable over more complex but more memory and time efficient tools.

2.2.1 Python

Python [23] is a general purpose scripting language for which many additional scientific libraries are available. It is widely used in scientific research. The main reason to use Python instead of other languages is the availability of packages to process fMRI data. The python libraries we use can be found in Table 2.1.

package	*description*
SciPy [24]	SciPy offers a number of open source scientific packages for Python.
Numpy [21]	Numpy is an open source Python package that provides a powerful N-dimensional array structure and a number of mathematical functions.
Pandas [22]	Pandas is an open source package that provides high performance data structures and data analysis tools.
NiBabel [19]	An open source library that provides input output functionality for numerous fMRI formats.
svgwrite [26]	An open source library that provides functions to create SVG images by defining primitive shapes.

Table 2.1: Python packages

2.2.2 JavaScript

JavaScript is a scripting language available in most modern web browsers. It can mainly be used to make web pages interactive. jQuery [14] is a general JavaScript library that simplifies a number of JavaScript functionalities. The JavaScript language with the jQuery library enables us to create interactive image viewer for our results.

2.2.3 LLSC

The LLSC (*LIACS Life Science Cluster*) is a cluster computer of the LIACS [15] (*Leiden Institute of Advanced Computer Science*). It consists of a maximum of 48 cores. At this moment each core uses Debian [7] version 3.2.63. It is able to distribute tasks using the TORQUE grid engine [27]. The cluster uses a separate file server to take care of the file reading and writing on the cluster.

2.2.4 FSL

FSL [44] (*FMRIB Software Library*) is a set of tools to preprocess and analyse fMRI data. The tools are available for Windows, Linux and Mac Os X. The tools can be accessed using either command line interfaces or via a graphical user interface. The tools we use can be found in Table 2.2. We use FSL because it offers a whole pre-processing pipeline for fMRI data.

package	*description*
FSLview [10]	A 3th dimensional viewer of fMRI data.
MCFlirt [9]	Motion correction for fMRI data.
BET [2]	Brain Extraction Tool, a tool to separate the brain from non-brain tissue in fMRI data.
FSLutils [8]	A number of utilities to process and convert Nifti files.

Table 2.2: FSL packages

2.2.5 Visualization tools

For the visualization process a number of tools are selected. The tools serve different steps in the visualization process. The visualization tools we use can be found in Table 2.3.

package	*description*
ImageMagick [12]	ImageMagick is a set of command line tools used to convert .SVG images to PNG.
Matlab [17]	Matlab is used to convert .PNG images to .STL models. First, areas of different colors will be extracted. These areas are then exported to STL models. This is done using the script `stlwrite.m` [25].
Blender [3]	Blender is used to import STL models. To each region, a material and a color is assigned. These are then rendered to images and videos that are used in the research.

Table 2.3: Tools for visualization

2.2.6 Weka

Weka [28] is a data mining software application that offers the classification pipeline and contains a large number of machine learning algorithms. In our research it is used to classify the data and to perform feature selection.

2.3 Methodology

The methodology we use to analyze the fMRI data can be described in 5 main steps and is an iterative process. In order to compare different choices and configurations in this study, we decided to compare everything with a *standard configuration*. An illustration of the experiments using this methodology can be found in figure 2.3. Because we are mainly interested in finding differences between the BPD group and the HC group, we limit the scope by not including the insecure persons in the study. In every step of the methodology we will discuss what the step means for the process and what particular configuration we use for the standard configuration. In the following subsections the steps will be discussed. The exact configuration options and file descriptions can be found in cf appendix C.

2.3.1 Preprocessing

The files we received from the domain experts are raw fMRI files. In order to compare the different subjects with each other, multiple preprocessing steps are required. We will use FSL commands to execute this preprocessing and we will use the LLSC in combination with Python to call these commands. In our main configuration we will perform the following preprocessing steps: slice timing correction, brain extraction, intensity normalisation, spatial smoothing and registration to a standard image using the commands listed in cf appendix B.1. The parameter values for the preprocessing can be found in the commands cf appendix B.

2.3.2 Time series analysis

After the proprocessing has been performed for every subject, we analyze the preprocessed fMRI scan to summarize every voxels time series dimension into some key descriptors, which we call *voxel-features*. The features will be calculated using Python in combination with Python packages. To load the fMRI images into Python we use the NiBabel package, to perform operations on it we use Numpy and to calculate the features we mainly use the scientific Python package SciPy. The computation will be done on the LLSC in order to reduce the computation time. In our standard configuration we use the complete set of features with low intercorrelations as seen in Figure 3.11.

2.3.3 Feature selection: determining average areas

If we would use all the voxel-features of all the voxels for classification, we would get an unusable amount of features. Take for example the case that there are 10 voxel-features for every voxel, the total amount of features would then calculate to $80 * 80 * 38 * 10 = 2432000$. This would be too many for any conventional classification algorithm and would introduce the curse of dimensionality, which means that computational problems and problems with overfitting could arise [46]. In order to reduce this feature amount we do not use the voxel-features individually, but we perform a feature-selection procedure by taking the averages of groups of voxels. These groups of voxels are defined based on the similarity in the spatial and/or feature space. After this we use these average features per area as final features. All of the calculating needed for configuring the average areas is done using the Python language. In our standard configuration we determine these groups of voxels by using region growing.

2.3.4 Classification

After the voxels are grouped into regions and therefore the features are aggregated into features per region, we split up the data in test and training distributions using leave-one-out cross-validation. As a last step we use the training set entries to train the model, which results in a decision tree which can map the entries of the test set to the particular group. For all the final classification we use Weka (see section 2.2.6). For the standard configuration we choose to use the J48 algorithm [13], which is a Weka implementation of the C4.5 decision tree algorithm. This algorithm chooses the attributes to split on based on which attributes most effectively splits the training data. The splitting criterion used in C4.5 is the information gain. Information gain measures the gain in entropy after a particular split on a particular variable is made, where entropy measures the impurity of a group of examples. We use a decision tree because it gives much insight into which features the algorithm uses.

All the models we create will be described by the accuracy score, the kappa statistic, a confusion matrix and where possible a visualization of the model. The accuracy score describes the percentage with which the model predicts the class correctly and the kappa statistic compares this observed accuracy with the expected accuracy. The structure of the given confusion matrix is described in table 2.4 where *True Positive* describes the amount of times the algorithm correctly classifies a subject as having BPD, where *True Negative* describes the amount of times the algorithm correctly classifies a subject as being healthy, where *False Positive* is defined as the amount of times the algorithm incorrectly classifies a healthy subject as having BPD and where *False Negative* is defined as the amount of times where the algorithm incorrectly classifies a BPD subject as being healthy. The visualization of the decision tree model is made using the visualize model function in Weka. In this visualization the features the algorithm uses are represented using ovals and the resulting classes are represented as rectangles. In those rectangles also two numbers are given. The left number is the resulting amount of BPD subjects in the training data after the split, the right number is the resulting amount of HC subjects in the training data after the split.

	Predicted as BPD subject	Predicted as HC subject	Total
Real BPD subject	*True Positive*	*False Negative*	*Total real BPD*
Real HC subject	*False Positive*	*True Negative*	*Total real HC*
Total	*Total predicted BPD*	*Total predicted HC*	*Total all subjects*

Table 2.4: Confusion matrix template

2.3.5 Visualization

Finally, we will visualize the results using different methods so that the results of the research will be better interpretable. First we export the data to second dimensional .SVG images using the Python svgwrite-package. This is done by exporting each slice along the Z-axis of a voxel-feature array. Areas can be indicated in the exporter, if a voxel falls into an area it will be colored. For each area, a color is assigned out of 6 colors. After this, we create a viewer using JavaScript to view the exported images in. We will also generate a number of images of 3D models by exporting the results to STL models using Matlab. After this, these models will be rendered using Blender.

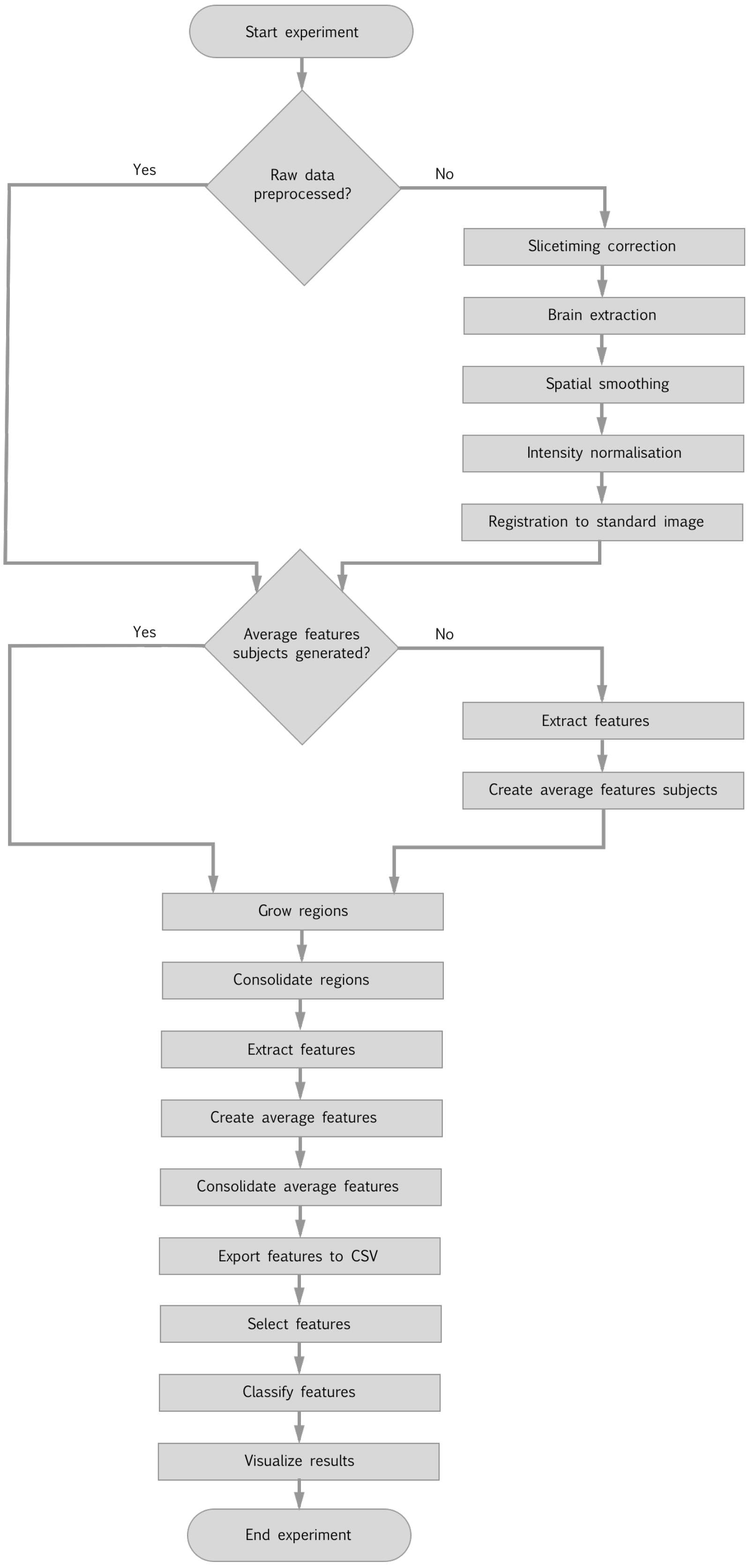


Figure 2.3: Illustration of the analysis process

Chapter 3

Experiments and results

In this chapter we will discuss the performed experiments and results. All of the sections in this chapter are linked to a particular sub-question, except one: the standard configuration experiment. In that section we discuss the results of the standard test we configured and in all the other sections we explain our way of thought and make variations on the standard configuration.

3.1 Standard configuration

To have an anchor point for all the other sections in this chapter, we first perform an analysis with a basic configuration which is purely based on our intuition of *what is reasonable* and purely serves as an anchor point for the other experiments and only gives an indication on how well our analysis performs. The standard configuration consists of the configuration seen in Table 3.1.

The results of this test can be seen in Table 3.2 and Table 3.3. The resulting tree and the regions used in that tree are shown in Figure 3.1 and 3.2.

Topic	*More information*	*Configuration*
Preprocessing	Section 3.2	Slice timing correction, brain extraction, intensity normalisation, spatial smoothing and registration to standard image.
Voxelfeatures	Section 3.3	`standardDeviation`, `skewness`, `kurtosis`, `maxPeak_skewness`, `maxPeak_kurtosis`, `peaks`, `peaksIntervalStd` and `peaksAvg`.
Way of aggregating	Section 3.5	Region growing with: • maximum *eucledian distance* $= 15$. • regions grown in average brain of healthy subjects on all of the above mentioned features. • seed points: `(71,72,42),(66,75,29), (48,91,53), (45, 47, 47), (64, 72, 32), (26, 72, 32), (36, 64, 47), (45, 60, 41), (45, 31, 55), (47, 29, 66), (52, 32, 6)`. • feature restrictions*: `standardDeviation`, `skewness`, `kurtosis`, `peaks`, `peaksAvg`, `peaksIntervalStd`, `peaksStd`.
Feature selection	Section 3.7	Attribute selection with information gain value as evaluator and Ranker as search method with the number of values to select is set to 15
Classification algorithm	Section 3.8	J48 algorithm using leave-one-out cross-validation and with the minimal number of examples in a leaf set to 5.

* It should be noted that due to human error the region growing in the standard experiment does not use the `maxPeak_kurtosis` and `maxPeak_skewness` as feature restrictions while we do use these features in the classification. The coefficient of variation of those features in the found regions (see Table 3.20) is less than the feature restriction used and less than the overall coefficient of variation, so it should not have a large impact on the regions.

Table 3.1: Configuration standard test

Accuracy	71.0145%
Kappa statistic	0.4192

Table 3.2: Results standard test

	Predicted BPD	Predicted HC	Total
Real BPD	27	8	35
Real HC	12	22	34
Total	39	30	69

Table 3.3: Resulting confusion matrix standard configuration

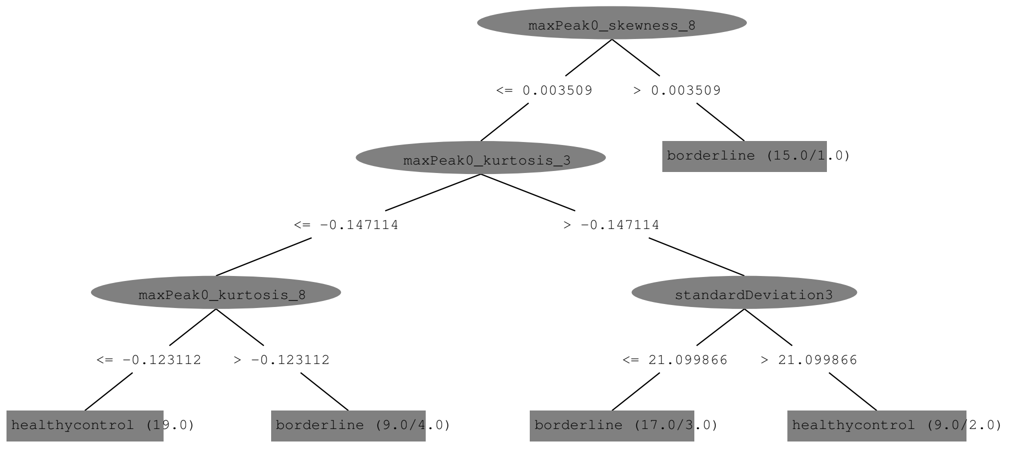


Figure 3.1: Resulting decision tree standard configuration

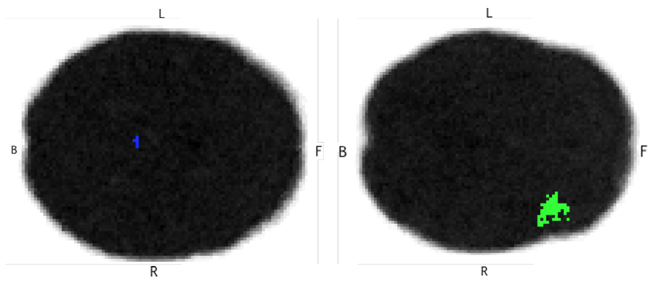


Figure 3.2: Areas used in decision tree with standard configuration at $z = 47$ and $z = 32$ respectively

3.2 Preprocessing steps

In this section we answer the question: "Which preprocessing steps are required for proper classification?". The initial dataset consisted of both raw fMRI images as well as preprocessed images. Since the already preprocessed images are not adjusted for slice timing differences and are not registered to a standard image, we need to improve the preprocessing to apply these methods. We analyse the effects of preprocessing on our final experiment to make sure these and other steps do not negatively influence our results. The steps that are considered are:

- Slice timing correction

  An fMRI scanner creates multiple 2D images of the brain that are stacked to generate a 3D image. Each slice is recorded at a different time, the last slice is recorded 2.2 seconds later than the first slice. This means that different areas in the fMRI image contain data from different time points. One method to cope with this is to perform slice timing correction. Slice timing correction is a method that shifts and interpolates the time series of each slice such that the slices are comparable. In our data the time between all the slices is 2.2 seconds. The scanning process is configured as regular down, which means that the scanner scans the brain from top to bottom.

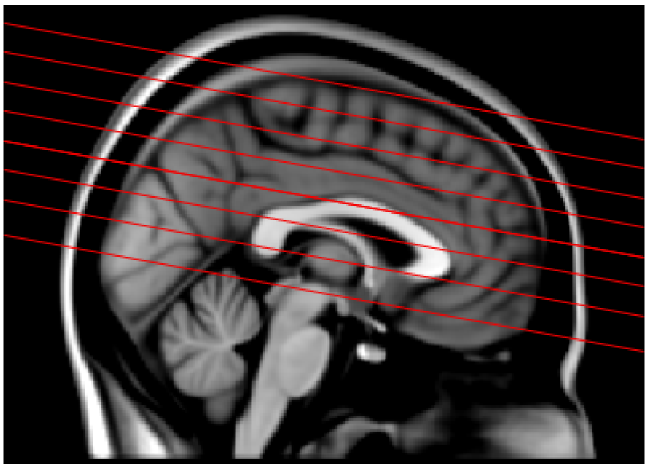


Figure 3.3: Illustration of time slices

- Brain extraction.

  Brain extraction is a method to separate the brain from non-brain tissue. This step is necessary to make sure the brain will not be classified based on blood flow in non-brain areas. We perform brain extraction using the standard settings of BET from FSL.

- Spatial smoothing

Raw fMRI data contains a lot of noise [41]. Spikes in the intensity of one voxel does not necessarily mean there is more activation in a certain area. A method to deal with this issue is to smooth every timeframe of the image spatially by convolving the timeframes with a Gaussian kernel [6]. The full width at half maximum (FWHM) is an estimate of the amount of smoothing performed on the image. In our standard configuration we use a FWHM of 2.12mm. After performing spatial smoothing, every voxels time series will be a weighted average of its own values and the values of its neighbours. Neighbours that are further away will have a lower weight, defined by the height of the Gaussian kernel in that voxel.

- Intensity normalisation
  Intensity normalisation is a method to normalize the time series of each subject. After applying this, the mean intensity of the time series of each voxel will be at the same level [34]. As a result, subjects cannot be classified by differences in absolute intensity levels. Intensity normalisation works by applying:

$$newIntensity = (intensity - min)\frac{newMax - newMin}{max - min} + newMin \tag{3.1}$$

  where max and min are the current maximum and minimum of the time series and $newMax$ and $newMin$ are the desired new maximum and minimum values.

- Registration to a standard image
  Naturally different persons have different shapes of brains. This introduces new problems in the classification process since the same position in different scans does not represent the same area in the brain. A method to deal with this is registering the scan of every subject to a standard image. The standard image consists of an average scan based on the brains of 152 subjects [1]. After an image is registered to the standard image, its dimensions change from 80 (x-axis) * 80 (y-axis) * 38 (z-axis) to 90 (x-axis) * 109 (y-axis) * 90 (z-axis).

3.2.1 Classification using raw data

The differences between feature selection on raw and preprocessed data will be measured using cubical regions (see section 3.5.1 for more information about cubical regions) because region growing is not sensible with unregistered images (see section 3.5.2 for more information about region growing). The results of this test, using the same parameters as in the standard configuration can be found in table 3.4 , table 3.5, figure 3.4 and figure 3.5.

One problem that occurred when applying a simple classification test using cubical regions was that all classification areas were located at the edge of the brain. There could be multiple explanations for this,

Accuracy	60.8696%
Kappa statistic	0.2176

Table 3.4: Results simple square classification

	Predicted BPD	Predicted HC	Total
Real BPD	21	14	35
Real HC	13	21	34
Total	34	35	69

Table 3.5: Confusion matrix simple square classification

maybe borderline patients have a different shape of brains on average or the shape of the skull is different on average. Because the result would be unreliable, it is important to register the images to a standard image so that every image will have the same dimensions.

3.2.2 Standard experiment without spatial smoothing

Spatial smoothing is applied to the data to reduce noise and to let each voxel represent an weighted average value of itself and its neighbors. We have performed an experiment to address if applying spatial smoothing improves the standard configuration experiment. The results can be found in table 3.6, table 3.7, figure 3.6 and figure 3.7.

Accuracy	56.5217%
Kappa statistic	0.1317

Table 3.6: Results no spatial smoothing

3.2.3 Standard experiment without intensity normalisation

Intensity normalisation is applied to normalize the mean intensity of all voxels in a volume. We have performed an experiment to address if applying intensity normalisation improves the standard configuration experiment. The results can be found in table 3.8, table 3.9, figure 3.8 and figure 3.9.

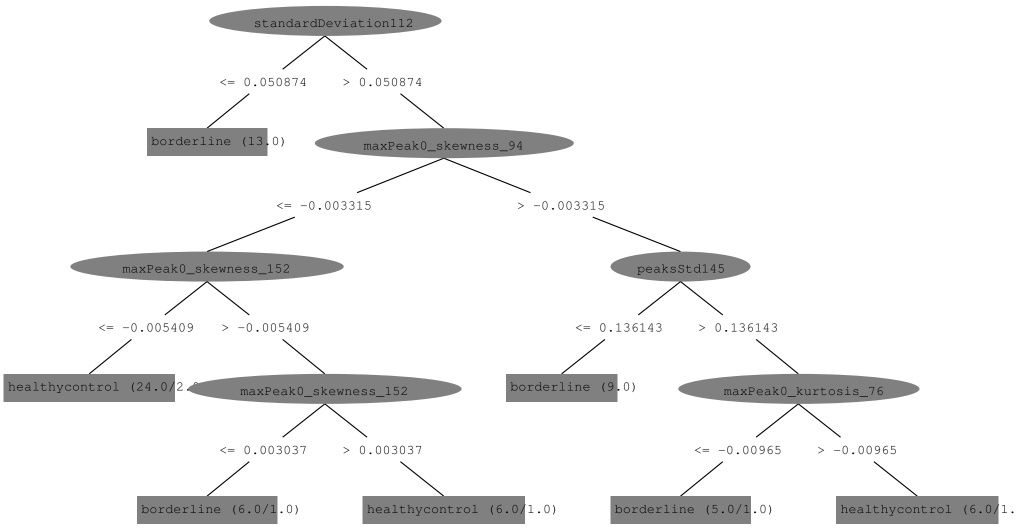


Figure 3.4: Decision tree, without preprocessing using simple square regions

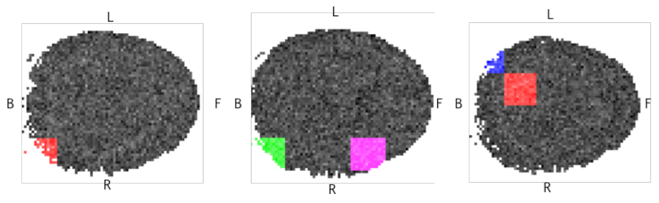


Figure 3.5: Areas classified without preprocessing using simple areas at z = 24, z = 26, z = 18 respectively

	Predicted BPD	Predicted HC	Total
Real BPD	18	17	35
Real HC	13	21	34
Total	31	38	69

Table 3.7: Confusion matrix no spatial smoothing

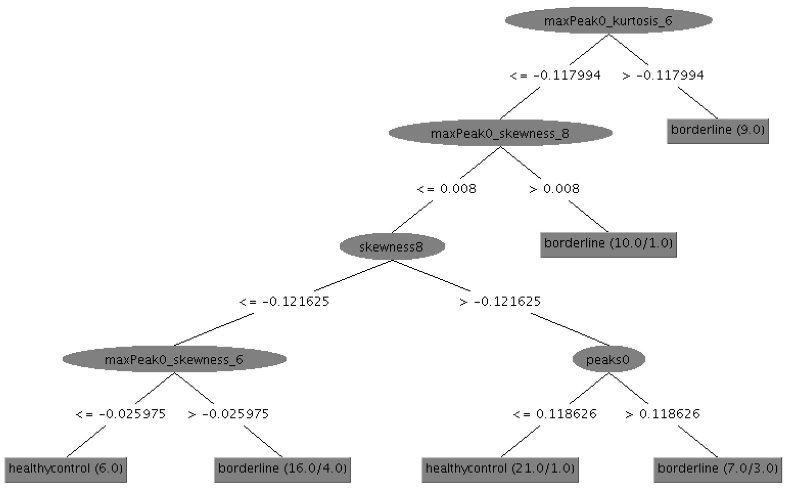


Figure 3.6: Decision tree, without spatial smoothing

Accuracy	40.5797%
Kappa statistic	-0.1812

Table 3.8: Results no intensity normalisation

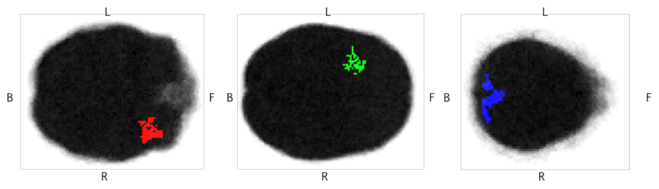


Figure 3.7: Areas classified without spatial smoothing at z = 27, z = 36, z = 69 respectively

	Predicted BPD	Predicted HC	Total
Real BPD	7	28	35
Real HC	13	21	34
Total	20	49	69

Table 3.9: Confusion matrix no intensity normalisation

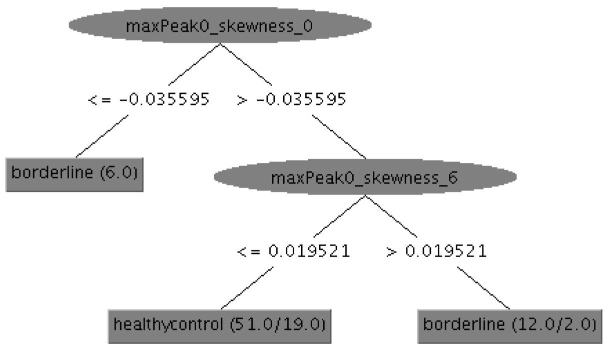


Figure 3.8: Decision tree, without intensity normalisation

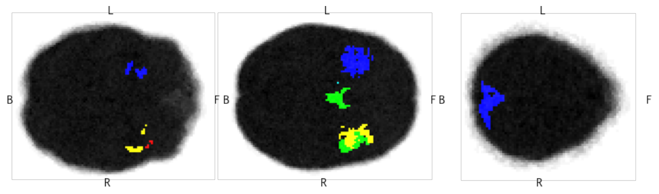


Figure 3.9: Areas classified without intensity normalisation at z = 28, z = 37, z = 67 respectively

3.3 Features

In this section we will discuss the problem of selecting key features to describe our data. We will do this by answering the following sub-question: "Which features are relevant?".

For extracting the features from the data, multiple levels of abstraction were possible: extracting a set of features for a whole brain at once, extracting a set of features for particular areas in the brain or extracting a set of features for every voxel in the brain. Based on the method for pattern analysis in fMRI described in [41] and with modularity in mind, we choose to extract a set of features for every voxel in the fMRI scans. In this way it is always possible to aggregate the features per voxel into features per area or features for the whole brain at once.

According to [39], high dimensionality is a big problem when mining in time series data. To prevent this problem from occurring, feature extraction should be applied to compress the time series into some key descriptors. By doing this only the most important information will be kept and the noise will be removed. In the process of translating the data into features and picking *relevant* features out of the set of possible features, we defined the following criteria for a *relevant* set of features:

- The features should have a high correlation with the target.
- The features should have a low correlation with the other features in the feature set.
- The features should be as easy interpretable as possible.

As seen in Section 2.1.4 and Figure 2.2, every voxel contains a time series describing the intensity over time. The amount of time points per time series in our dataset ranges from 122 to 248 (with average of 163.5 and standard deviation of 19.1). To compress those time series into some key descriptors we in collaboration with our domain experts came up with the following voxel-features:

1. Average intensity (`averageIntensity`): measures the average y-value of the time series using

$$AverageIntensity = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.2}$$

   where n is the amount of timepoints and $x_1, x_2, \ldots, x_n$ is the y-value of a timepoint.

2. Standard deviation (`standardDeviation`): measures the standard deviation in the y-value of the time series

$$StandardDeviation = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.3}$$

   where n is the amount of timepoints, x_1, x_2, ..., x_n are the observed y-values of the timepoints and $\overline{x}$ is the average y-value.

3. Maximum value (`maximumValue`): measures the maximum y-value of the time series.

4. Minimum value (`minimumValue`): measures the minimum y-value of the time series.

5. Skewness [18] (`skewness`): measures the Fisher-Pearson coefficient of skewness of the total time series. The skewness indicates the 'tailedness' of a distribution, where a value > 0 indicates that there is more weight in the left tail of the distribution and a value < 0 indicates that there is more weight in the right tail of the distribution. The used skewness is defined as follows:

$$Skewness_x = \frac{\mu_3}{\sigma^3} \tag{3.4}$$

where μ_3 is derived from the third order central moment and σ is the standard deviation of the time series.

6. Kurtosis [18] (`kurtosis`): measures the kurtosis of the total time series using Fisher's definition. The kurtosis indicates the 'peakedness' of a distribution, where (using Fisher's definition) a value < 0 indicates a lower kurtosis than the normal distribution and a value > 0 indicates a kurtosis higher than the normal distribution. The used kurtosis can be defined as follows:

$$Kurtosis_x = \frac{\mu_4}{\sigma^4} \tag{3.5}$$

where μ_4 is derived from the fourth order central moment and σ is the standard deviation of the time series.

7. Skewness of the highest peak (`maxPeak0_skewness`): Measures the skewness of the highest peak. To calculate the highest peak every local maxima is compared to their neighbouring two local minima. Then the height of the peak is defined by the average of the y-value difference with its left local minima and the y-value difference with its right local minima. The highest peak is defined as the peak with the greatest height.

8. Kurtosis of the highest peak (`maxPeak0_kurtosis`): Measures the kurtosis of the highest peak.

9. Peaks (`peaks`): measures the amount of local maxima per timeframe:

$$Peaks = \frac{x_{voxel}}{y_{subject}} \tag{3.6}$$

where x_{voxel} is the total amount of local maxima in that voxels timesieres and $y_{subject}$ is the total amount of timeframes for that subjects scan. When determining local maxima, no threshold for the size of the peak is set so every local maxima is included.

10. Standard deviation of distance between peaks (`peaksIntervalStd`): measures the standard deviation of the distance between the local maxima. For calculating this feature first a list containing the x-values of the local maxima is created. After this a new list is created containing the distance between those values and finally the standard deviation is calculated from this list with distances. A value of 0 indicates a that the peaks are perfectly uniformly distributed and a value > 0 indicates a less uniformly distribution.

11. Average intensity of peaks (`peaksAvg`): measures the average y-values of the local maxima.

12. Standard deviation of intensity values of peaks (`peaksStd`): measures the standard deviation in y-value of the local maxima.

To give an proxy of the correlation between all the features, we create the correlation matrix for the average of the features for an area grown with region growing (see Section 3.5.2) with seed point $x = 71$, $y = 72$, $z = 42$. It should be noted that the different correlation matrices show the same behaviour. The resulting correlation matrix can be seen in Figure 3.10. Based on this correlation matrix and Table 3.10 we decided to drop four of the high correlating features: `peaksStd`, `maximumValue`, `minimumValue` and `averageIntensity` in our standard configuration. The resulting correlation matrix is shown in Figure 3.11

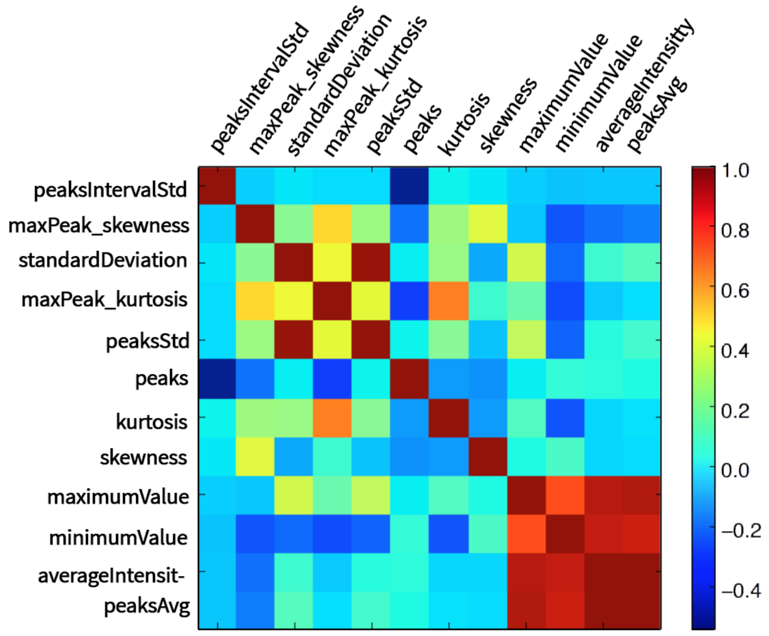


Figure 3.10: Correlation matrix with high correlating features

3.3.1 Subset of features

To investigate the importance of several features we test the standard configuration with subsets of features instead of all the features. We choose these subsets based on the characteristics of the features. One way to divide the features according to their characteristics is to divide them into features that say something about the

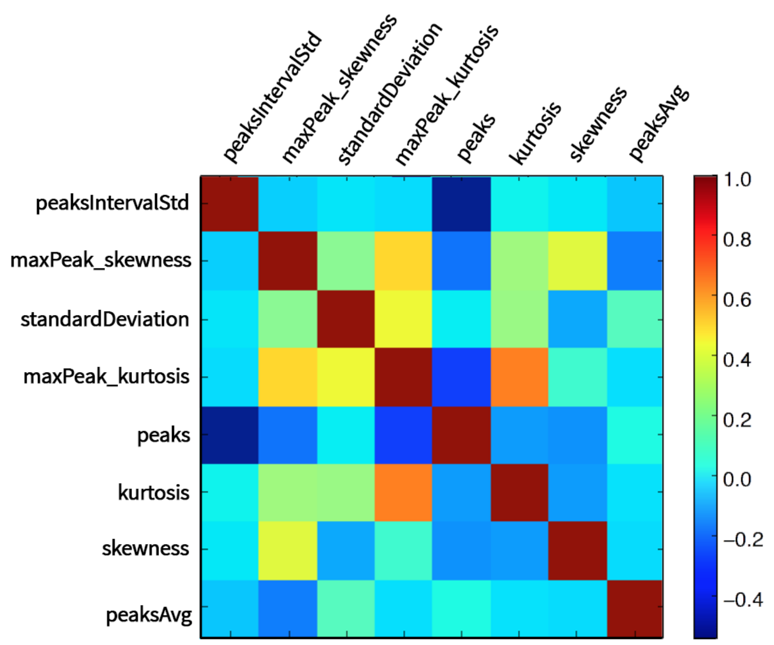


Figure 3.11: Correlation matrix without high correlation features

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
(1)	1.00	-0.05	-0.01	-0.03	-0.03	-0.54	0.02	-0.00	-0.05	-0.08	-0.07	-0.07
(2)	-0.05	1.00	0.23	0.51	0.26	-0.22	0.27	0.41	-0.07	-0.28	-0.22	-0.20
(3)	-0.01	0.23	1.00	0.44	0.99	0.01	0.26	-0.12	0.37	-0.24	0.09	0.13
(4)	-0.03	0.51	0.44	1.00	0.42	-0.34	0.69	0.09	0.16	-0.30	-0.06	-0.02
(5)	-0.03	0.26	0.99	0.42	1.00	0.02	0.22	-0.07	0.34	-0.25	0.06	0.10
(6)	-0.54	-0.22	0.01	-0.34	0.02	1.00	-0.15	-0.17	0.01	0.08	0.06	0.04
(7)	0.02	0.27	0.26	0.69	0.22	-0.15	1.00	-0.15	0.12	-0.28	-0.04	-0.01
(8)	-0.00	0.41	-0.12	0.09	-0.07	-0.17	-0.15	1.00	0.05	0.11	-0.04	-0.02
(9)	-0.05	-0.07	0.37	0.16	0.34	0.01	0.12	0.05	1.00	0.77	0.94	0.96
(10)	-0.08	-0.28	-0.24	-0.30	-0.25	0.08	-0.28	0.11	0.77	1.00	0.93	0.91
(11)	-0.07	-0.22	0.09	-0.06	0.06	0.06	-0.04	-0.04	0.94	0.93	1.00	1.00
(12)	-0.07	-0.20	0.13	-0.02	0.10	0.04	-0.01	-0.02	0.96	0.91	1.00	1.00

Table 3.10: Correlation values between features where (1) is `peaksIntervalStd`, (2) is `maxPeak_skewness`, (3) is `standardDeviation`, (4) is `maxPeak_kurtosis`, (5) is `peaksStd`, (6) is `peaks`, (7) is `kurtosis`, (8) is `skewness`, (9) is `maximumValue`, (10) is `minimumValue`, (11) is `averageIntensity`, (12) is `peaksAvg`

peaks of the time series (`peaks`, `maxPeak_kurtosis`, `maxPeak_skewness`, `peaksAvg`, `peaksIntervalStd`) and features that are more focused on the whole time series (`standardDeviation`, `kurtosis`, `skewness`). The results of the test with the first subset of features can be seen in Table 3.11, Table 3.12 and in Figure 3.12. The results of the test with the second subset of features can be seen in Table 3.13, Table 3.14 and in Figure 3.13.

Accuracy	55.0725%
Kappa statistic	0.0986

Table 3.11: Classification results using only *peak-features*: `peaks, maxPeak_kurtosis, maxPeak_skewness, peaksAvg, peaksIntervalStd`

	Predicted BPD	Predicted HC	Total
Real BPD	23	12	25
Real HC	19	15	34
Total	42	27	69

Table 3.12: Confusion matrix using only *peak-features*: `peaks, maxPeak_kurtosis, maxPeak_skewness, peaksAvg, peaksIntervalStd`

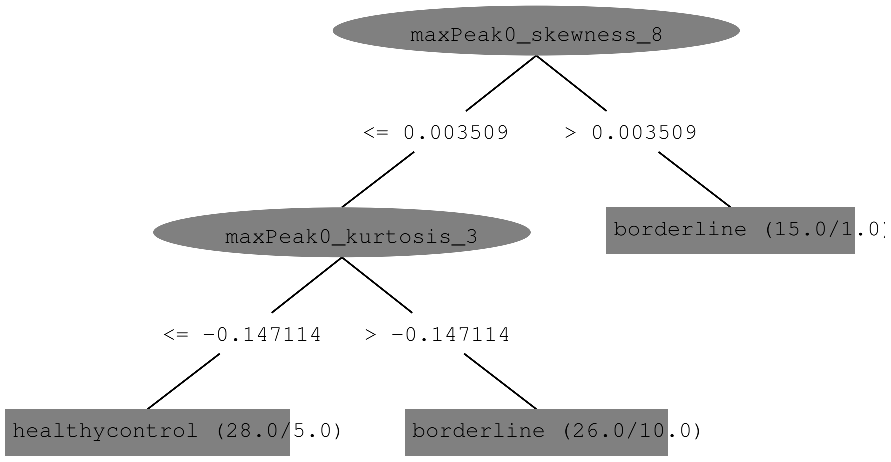


Figure 3.12: Resulting decision tree using only *peak-features*: `peaks`, `maxPeak_kurtosis`, `maxPeak_skewness`, `peaksAvg`, `peaksIntervalStd`

Accuracy	56.5217%
Kappa statistic	0.1389

Table 3.13: Classification results using only *non-peak features*: `standardDeviation, kurtosis, skewness`

	Predicted BPD	Predicted HC	Total
Real BPD	8	27	35
Real HC	3	31	34
Total	11	58	69

Table 3.14: Resulting correlation matrix using only *non-peak features*: `standardDeviation, kurtosis, skewness`

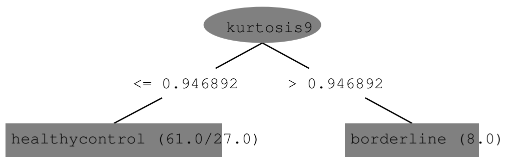


Figure 3.13: Resulting decision tree when using only *non-peak features*: `standardDeviation, kurtosis, skewness`

3.4 LLSC

In this section we will answer the question: "How can we handle the processing of large files?" Processing the data is resource intensive. For each voxel, there is a one dimensional list where voxel-features will be extracted from. After extracting the voxel-features, averages of areas also need to be calculated. The amount of information to be processed caused two problems:

- The computation time was very high, even with 100% CPU load processes lasted longer than a day.
- The amount of memory used was so high that processes would fall back into swap memory. This caused the processes to significanlty slow down.

3.4.1 The LLSC cluster

To run jobs on the LLSC cluster, we implemented two scripts that were able to run Torque commands from the command line. The scripts implement the job-script required by Torque engine. The job script describes the amount of memory used, the amount of processors allocated and the file to be execute. The amount of memory to be allocated is a dynamic value so that we could assign more memory to jobs that demand it. There are three kinds of jobs:

- `extract_features`. This job extracts the features of one subject and stores results in the output directory. To each job, 4GB of RAM and one processor is allocated. The processing of all these jobs takes around 2 hours in total, depending on the settings.
- `create_avg_features`. This job creates the average features of all areas for one subject. 4GB of RAM and one processor is allocated to each job. Results are stored in the output directory. The processing of these jobs takes around half an hour in total, depending on the settings.
- `region_growing` This is a job that does the region growing. Unlike the other jobs, it does not generate results for each subject. Instead, it generates the resulting region in the average image of all healthy subjects for one seed point. Because it is very memory intensive, 8GB of RAM and one processor are allocated. Results are stored in the output directory. The processing of this jobs takes around 10 minutes in total, depending on the settings.

For each subject or seed point, the job can be started by calling the command line tool *qsub*. This is a command line tool from Torque that submits a job cf appendix B.2 to the Torque engine .

3.5 Regions

As seen in section 3.3 we generate a set of features for every voxel.

Feature selection is valuable to building a classifier on fMRI data. It involves the decision which voxels will be included in the classifier [41]. In order to get a workable amount of features, we need to aggregate the features into averaged regions to reduce the amount of features. Therefore in this chapter we answer the question: "How can we divide the brain into spatial areas which can be used for classification?".

In order to answer this question we came up with two different methods. Both methods take the average feature values of an area consisting of multiple (> 0) voxels. The two methods are:

1. Dividing the brain into regions of cubical shape each having roughly the same shape and volume.
2. Dividing the brain into regions of variable shape and volume where the shape and volume is based on the similarity of the features.

For both methods the process of assigning voxels to the regions is different, but the the process of aggregating the voxel-features to features per region is exactly the same. How we configure which voxels belong to which area will be explained in subsections 3.5.1 and 3.5.2. For aggregating the voxel-features to features per area, we take the average voxel-feature values of all the voxels in the particular area, as shown in Figure 3.14. Given that every voxel contains n voxel-features (b) and that there are m areas, the total amount of features (f) in our featureset calculates to $n * m$. So every basic feature essentially splits up in m features f. This process is also illustrated in Figure 3.14.

3.5.1 Cubical regions

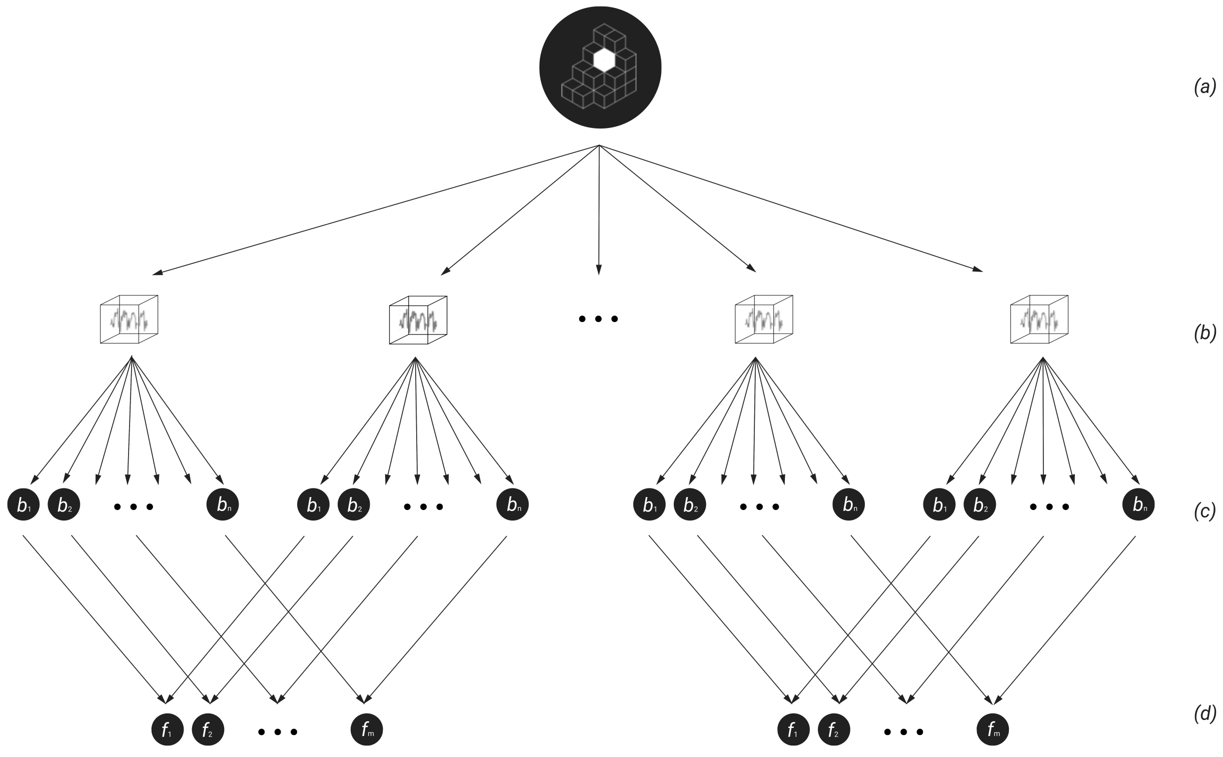
The first way to determine the areas is dividing the whole brain (featurespace) into cubical regions. The amount of areas is defined in our settings file, where the amount of areas on the different axis can be specified individually.

The area number a voxel belongs to is determined with the following function:

$$area = area_x + (area_y * numberOfAreas_x) + (area_z * numberOfAreas_x * numberOfAreas_y) \tag{3.7}$$

where the subscript x, y and z defines the area the variable refers to, where $numberOfAreas_{axis}$ is the amount of areas specified on for particular axis and where $area_x$, $area_y$ and $area_z$ are defined as:

$$area_{axis} = \left\lfloor \left(\frac{x}{length_{axis}}\right) * numberOfAreas_{axis} \right\rfloor \tag{3.8}$$



First for every voxel (b) in the brain (a) a set of voxel-features (c) is calculated from the time series in a voxel. After this those voxel-features of every voxel are aggregated into features per area (d) by taking the average of all the voxel-features in an area.

Figure 3.14: Feature extraction process

where x is the current voxel coordinate on the particular axis, $length_{axis}$ is the length of the particular axis and $numberOfAreas$ is the amount of areas specified for the particular axis.

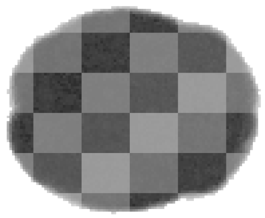


Figure 3.15: Resulting regions when using cubical regions at $z = 40$ and $numberOfAreas_x = numberOfAreas_y = numberOfAreas_z = 6$

This process has one main disadvantage with several consequences. Firstly, the division is very arbitrary because it is not based on any knowledge from the data. As a consequence of this, the standard deviations for the features in the areas are relatively high, so calculating the average values leads to the loss of a relatively high amount information. Another consequence of this is that it selects areas which go through all kinds of brain matter. Also a consequence is that some areas are on the edge of the brain and may contain very few valid voxels which results in features evaluating to zero for some subjects in some regions. This makes the data mining algorithm less reliable because the decision tree will then mine decision rules based on those empty values. A possible reason that the algorithm uses the features which has zero values for some subjects is because of the many features and chance. When there are many features containing zero values, there is a high chance that in some of those features the values are only zero for subjects of one class. The algorithm will then use this feature because it (misleading) provides a very good split on the data. To see this last disadvantage in practice, see the decision tree in Figure 3.16 and some of the areas the decision tree used in Figure 3.17. It also leads to a relatively low classification accuracy score of 62.6866% with a kappa statistic measure of 0.2546 as seen in Table 3.15 and Table 3.16.

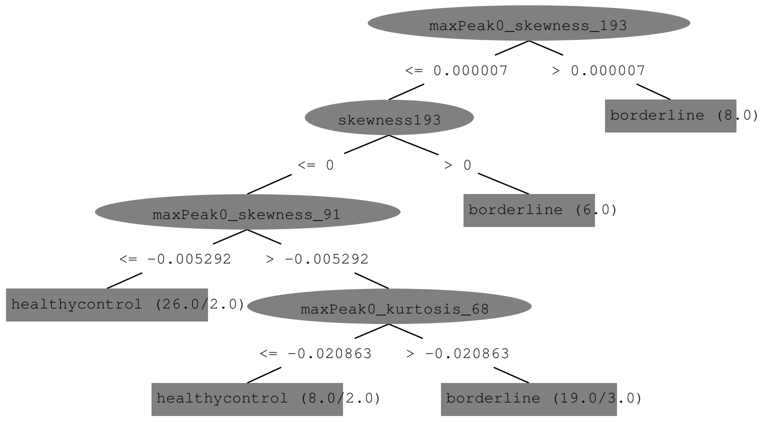


Figure 3.16: Decision tree, using standard settings using cubical regions

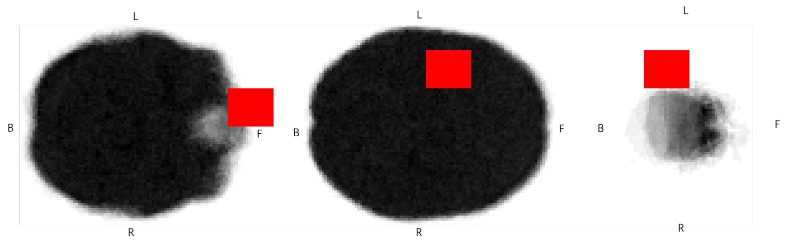


Figure 3.17: Areas used (visualized in red) in decision tree in Figure at z = 27, z = 42, and z = 76 respectively

Accuracy	62.6866%
Kappa statistic	0.2546

Table 3.15: Results cubical regions

3.5.2 Region growing

To overcome these disadvantages of cubical regions we use the principles of region growing [48] to find areas that contain relatively similar voxels. According to [41], a way to select features is to limit the analysis to specific anatomical regions. By using this method we can combine anatomical regions of interest with the feature characteristics.

Region growing performs a segmentation of an image with respect to a set of seed points. Given n voxels

	Predicted BPD	Predicted HC	Total
Real BPD	20	4	24
Real HC	11	22	33
Total	31	26	67

Table 3.16: Confusion matrix, cubical regions

in the 3D space as seed points, the algorithm starts with those seed points as regions, say $R_1, R_2, \cdots, R_n$. The algorithm will then expand those regions based on the criteria that regions can only expand to direct neighbouring voxels and only to those who satisfy to some homogeneity criteria. Using this method the algorithm finds spatial connected areas which are more homogeneous than when using cubical regions. With the process of seeded region growing higher level knowledge of the data can easily be incorporated into the technique through the choice of seeds [29]. Such knowledge can be what a region of interest is and what irrelevant noise is.

To perform this growing of regions we use the following steps:

1. Start with n arbitrary seed points, every seed point represents a region $R_1, R_2, \cdots, R_n$.
2. For all neighbours of R_i recursively add the neighbour to R_i if neighbour satisfies to chosen criteria.
3. If the neighbour does not satisfy to chosen criteria, proceed with at step 2 with other neighbour.
4. The growth of the region stops if none of the neighbours satisfies to the criteria.

For the first step we need to determine which seed points to use. We used two different approaches for this. The first approach is to use seed points from Regions of Interest (ROI). Those ROI are chosen based on the results from [47] from our domain experts. The seed coordinates used can be found in Table 3.17. It also must be said that due to human error we forgot to use the seed point with coordinates (19,47,33). The second approach is to use every center of the cubical regions as seed points while discarding the seed points outside of the brain.

Area name	*Hemisphere / position*	*Coordinates (x, y ,z)*
Superior Parietal lobe	Left	(47,29,66)
Inferior frontal gyrus	Left	(71,72,42)
Frontal Pole	Left	(48,91,53)
Cingulate gyrus	Posterior	(45, 47, 47)
Precuneus	Left	(64, 72, 32)
Caudate Nucleus	Right	(26, 72, 32)
Insula	Right	(36, 64, 47)
Thalamus	Left/Right	(45, 60, 41)
Orbitofrontal cortex	Left/Right	(45, 31, 55)
Insula	Left	(47, 29, 66)

Table 3.17: Seed points ROI

In step two we define a neighbour as a 6-connected neighbour (also see Figure 3.18). 6-connected neighbours are neighbours to voxels that touches one of their faces. Other options are 18-connected and 26 connected. To reduce complexity reasons in this process 6-connected neighbours are used. When translated to a coordinate system, the 6-connected neighbours of a voxel with coordinates (x, y, z) have the coordinates $(x \pm 1, y, z)$, $(x, y \pm 1, z)$ and $(x, y, z \pm 1)$.

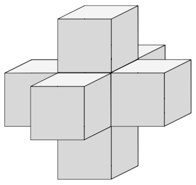


Figure 3.18: 6-connected voxels

Because every voxel in our dataset which is subject to growing contains multiple features, the similarity criterion we chose is based on the features in a voxel. For the voxel-features we first determine the coëfficient of variation ($\mathcal{C}_v$) computed over all the voxels in the brain (see Table: 3.18) using:

$$\mathcal{C}_v = \frac{\sigma}{\mu} \tag{3.9}$$

where σ is the standard deviation and μ is the mean of the feature. We then have grown the regions with the standard deviations as similarity criterion. With the kurtosis feature we allow 1.5 times the standard deviation. This is because we saw that the algorithm mainly used kurtosis to stop growing when using 1 time the standard deviation for kurtosis.

Voxel-feature	*Coefficient of variation ($\mathcal{C}_v$)*
standardDeviation	0.676652499253
skewness	3.4962250355
kurtosis	1.1097566506
peaks	0.4035803111283
maxPeak_skewness	170.406475662
maxPeak_kurtosis	0.76828261191
peaksAvg	0.671799922562
peaksIntervalStd	0.403311772235

Table 3.18: Voxel-features with coefficient of variation

Because we are mainly interested in finding local areas and not in finding areas containing voxels in all different places of the brain we also calculate a diminishing value which makes the similarity criterion more strict when the voxel is further away from the seed voxel. This diminishing value (k) is based on the euclidean distance of voxel V with respect to the seed point voxel S using:

$$k = (euclideanDistance_{pref} - euclideanDistance_{current}) / euclideanDistance_{pref} \tag{3.10}$$

where $euclideanDistance_{pref}$ is a parameter for the maximum euclidean distance and $euclideanDistance_{current}$ is the current euclidean distance between V and S.

We then use this standard deviation as a similarity criterion. The voxel V may only be added to the region R

with seed point S if for all features the following holds:

$$V_{featurevalue} \leq S_{featurevalue} + k * \left| \mathcal{C}_v * S_{featurevalue} \right| \tag{3.11}$$

$$V_{featurevalue} \geq S_{featurevalue} - k * \left| \mathcal{C}_v * S_{featurevalue} \right| \tag{3.12}$$

The classification results of using region growing in the classification process are shown in the standard configuration in section 3.1.

3.5.3 Multiple experiments with region growing

Analysing the resulting regions of our region growing implementation can be interesting for a couple of reasons. The first one is to validate that the algorithm results in regions we expect based on our intuition and knowledge of the brain. For example, it is expected that the regions grown with region growing do not pass through different brain tissues because the time series should look very different in the various brain tissues. Another reason it is interesting to look at the regions for analysis purposes, for example looking at the size and shape of the regions. Differences in regions grown with the same seed points but with different average brains (average healthy brain versus average borderline brain) could reveal differences between the two groups. Therefore we performed a couple of experiments. In the following experiments we describe and analyse the found regions. To analyse the regions we describe the single regions with the amount of voxels in a region as the *Voxel amount*, the axis of the minimum bounding box as B_x, B_y and B_z and the space filling of the minimum bounding box as *Space filling* with the following formula:

$$\textit{Space filling} = \frac{\textit{Voxel amount}}{B_x * B_y * B_z} \tag{3.13}$$

We compare the different regions by calculating the $\Delta Amountofvoxels$ which is the absolute difference in voxel amount of two regions. And we also calculate the *Relative size* of region a in comparison with region b which we define as:

$$\textit{Relative size} = \frac{\textit{Voxel amount}_a}{\textit{Voxel amount}_b} \tag{3.14}$$

Region growing with standard configuration

For validating the grown regions we first use region growing with the standard configuration. A 3D-vizualisation of the resulting regions can be found in Figure 3.19. As can be seen, the regions differ in size and in shape. In Table 3.19, the details of the particular regions are shown.

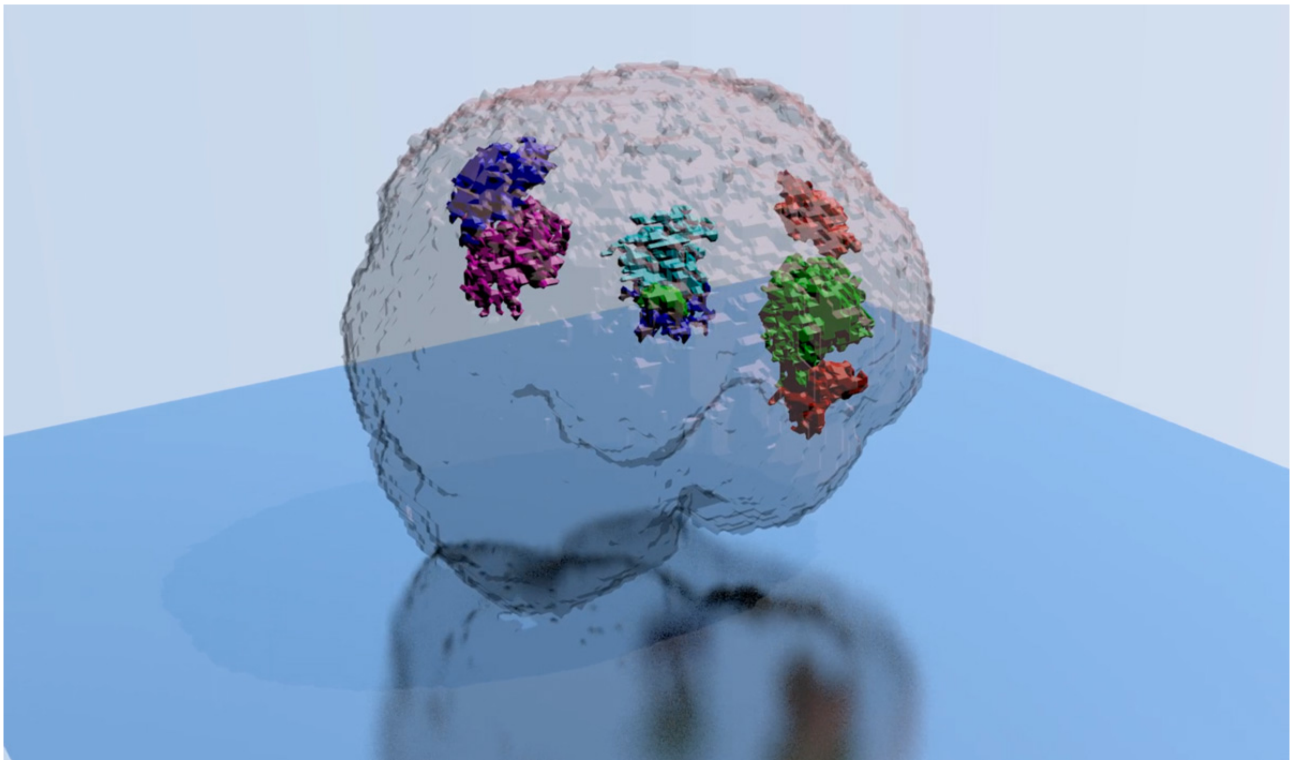


Figure 3.19: Results from region growing with standard configuration visualized in 3D

Area	*Seed point (x,y,z)*	*Voxel amount*	B_x	B_y	B_z	*Space filling*
Superior Parietal lobe	(47, 29, 66)	787	22	18	15	0.132
Inferior frontal gyrus	(71, 72, 42)	1828	16	24	22	0.216
Frontal Pole	(48, 91, 53)	658	18	15	18	0.135
Cingulate gyrus	(45, 47, 47)	6	4	2	2	0.375
Precuneus	(45, 31, 55)	1254	18	19	20	0.183
Caudate Nucleus	(36, 64, 47)	777	14	21	18	0.147
Insula	(26, 72, 32)	473	17	17	13	0.126
Thalamus	(45, 60, 41)	119	5	10	6	0.397
Orbitofrontal cortex	(66, 75, 29)	671	17	16	16	0.154
Insula	(64, 72, 32)	2	1	1	2	1.0

Table 3.19: Details of regions grown with standard configuration

We also calculated the average coefficient of variation in the resulting regions for all the voxel-features. In Table 3.20 it can be seen that for all the voxel-features the average C_v is lower than the C_v for the whole brain.

Voxel-feature	*Average C_v in regions*
standardDeviation	0.178366533024
skewness	1.4986423829
kurtosis	0.511034648179
peaks	0.02391007466
maxPeak_skewness	10.0144106125
maxPeak_kurtosis	0.598082794186
peaksAvg	0.0857761533346
peaksIntervalStd	0.0516851267675

Table 3.20: Average coefficient of variation per voxel-feature in the found regions

Region growing with average borderline scan

In this experiment we grow the regions with the same parameters as in the standard configuration, but now with the average scan of BPD subjects as a basis. The resulting visualization is shown in Figure 3.20 and the resulting analysis of the regions can be found in Table 3.21.

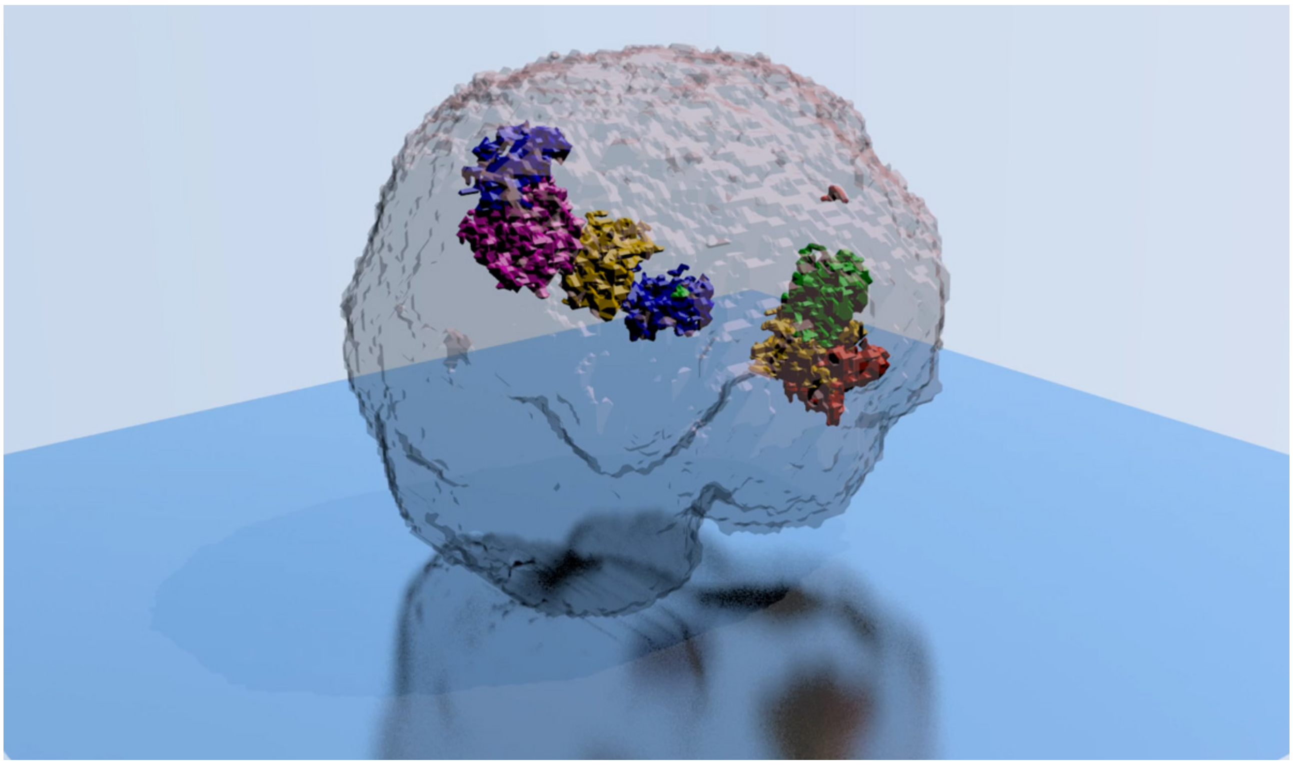


Figure 3.20: Results from region growing in avarage BPD brain visualized in 3D

Region growing difference between BPD and HC

To see the differences between the grown areas from Figure 3.19 and Figure 3.20 we also calculated the differences between the regions, see Table 3.22

Region growing without euclidean distance restriction

To validate if the region growing does not grow regions through different types of brain matter, we also performed a experiment where we ignored the euclidean distance restriction. In this experiment the standard configuration settings are used with the only variation that the homogeneity criteria is based on 0.5 times the coëfficient of variation to limit the growth of the areas in another way. The results are shown in Figure 3.21 where it can be seen that the regions follow some kind of pattern. This could indicate that it follows the natural patterns of the brain.

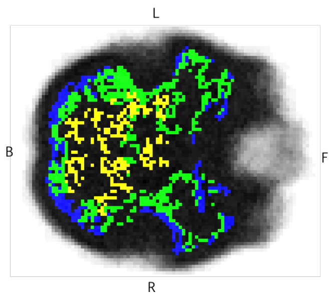


Figure 3.21: Regions grown without the euclidean distance restriction at z = 25

Region growing with different subsets of features as restrictions

To see how the peaks differ in size and shape if the regions are grown with different subsets of features as similarity criteria on the different groups, we split up the features into two subsets:

- Subset 1 describes the peak data and contains the following features: `peaks`, `peaksAvg`, `peaksIntervalStd`, `peaksStd`.
- Subset 2 describes the overall time series and contains the following features: `standardDeviation`, `kurtosis`, `skewness`.
- Subset 3 describes the maximum peaks of the time series and contains the following features: `maxPeak0_kurtosis`, `maxPeak0_skewness`.

Area	*Seed point (x,y,z)*	*Voxel amount*	B_x	B_y	B_z	*Space filling*
Superior Parietal lobe	(47, 29, 66)	581	20	16	13	0.14
Inferior frontal gyrus	(71, 72, 42)	828	14	18	18	0.183
Frontal Pole	(48, 91, 53)	19	5	3	3	0.422
Cingulate gyrus	(45, 47, 47)	815	17	17	15	0.188
Precuneus	(45, 31, 55)	1447	19	18	20	0.212
Caudate Nucleus	(36, 64, 47)	9	3	5	2	0.3
Insula	(26, 72, 32)	701	17	19	13	0.167
Thalamus	(45, 60, 41)	16	4	4	3	0.333
Orbitofrontal cortex	(66, 75, 29)	713	16	16	16	0.174
Insula	(64, 72, 32)	1420	19	19	16	0.246

Table 3.21: Details of regions grown with average BPD brain

Area	*Seed point (x,y,z)*	*Δ Amount of voxels*	*Relative size*
Superior Parietal lobe	(47, 29, 66)	206	1.355
Inferior frontal gyrus	(71, 72, 42)	1000	2.208
Frontal Pole	(48, 91, 53)	639	34.632
Cingulate gyrus	(45, 47, 47)	809	0.007
Precuneus	(45, 31, 55)	193	0.867
Caudate Nucleus	(36, 64, 47)	768	86.333
Insula	(26, 72, 32)	228	0.675
Thalamus	(45, 60, 41)	103	7.438
Orbitofrontal cortex	(66, 75, 29)	42	0.941
Insula	(64, 72, 32)	1418	0.001

Table 3.22: Differences between Table 3.19 and Table 3.21

Area	*Seed point (x,y,z)*	*Voxel amount HC*	*Voxel amount BPD*
Superior Parietal lobe	(47, 29, 66)	1033	1440
Inferior frontal gyrus	(71, 72, 42)	3835	3425
Frontal Pole	(48, 91, 53)	1663	1832
Cingulate gyrus	(45, 47, 47)	2565	2134
Precuneus	(45, 31, 55)	4938	4026
Caudate Nucleus	(36, 64, 47)	1943	1444
Insula	(26, 72, 32)	3080	2883
Thalamus	(45, 60, 41)	128	120
Orbitofrontal cortex	(66, 75, 29)	1306	1230
Insula	(64, 72, 32)	3156	2625

Table 3.23: Comparison between regions grown on the average of the healthy controls and on the average borderline subjects with subset 1

The result is shown in Table 3.23, Table 3.24 and Table 3.25.

Area	*Seed point (x,y,z)*	*Voxel amount HC*	*Voxel amount BPD*
Superior Parietal lobe	(47, 29, 66)	1236	892
Inferior frontal gyrus	(71, 72, 42)	2119	840
Frontal Pole	(48, 91, 53)	689	19
Cingulate gyrus	(45, 47, 47)	6	985
Precuneus	(45, 31, 55)	1276	1486
Caudate Nucleus	(36, 64, 47)	1076	9
Insula	(26, 72, 32)	498	728
Thalamus	(45, 60, 41)	1547	20
Orbitofrontal cortex	(66, 75, 29)	988	1034
Insula	(64, 72, 32)	2	1714

Table 3.24: Comparison between regions grown on the average of the healthy controls and on the average borderline subjects with subset 2

Area	*Seed point (x,y,z)*	*Voxel amount HC*	*Voxel amount BPD*
Superior Parietal lobe	(47, 29, 66)	1210	601
Inferior frontal gyrus	(71, 72, 42)	1322	5
Frontal Pole	(48, 91, 53)	1225	836
Cingulate gyrus	(45, 47, 47)	2756	2439
Precuneus	(45, 31, 55)	3013	2902
Caudate Nucleus	(36, 64, 47)	2721	2631
Insula	(26, 72, 32)	2666	1
Thalamus	(45, 60, 41)	10	945
Orbitofrontal cortex	(66, 75, 29)	2416	2022
Insula	(64, 72, 32)	1954	2648

Table 3.25: Comparison between regions grown on the average of the healthy controls and on the average borderline subjects with subset 3

3.6 Zooming in

In this section we will answer the question: "Do smaller, more specific, areas improve results compared to larger areas?" A general problem in our research is that we need to find a balance between having a manageable amount of features and using features which are too general. In other sections we focus on analysing parts in the whole brain. These areas will be used for further region growing in this section. We will focus on the two best regions classified in the standard configuration experiment. These are grown in the Cingulate gyrus and the Orbitofrontal cortex.

3.6.1 Determining seed points

We implemented the option to generate seed points from the centers of square regions. In this section we will generate seed points for num_areas_x = 30, num_areas_y = 30, num_areas_z = 30. This will generate $12^3 = 1728$ different points. The standard configuration experiment uses seed points determined by our domain experts. The regions which we selected from the standard configuration experiment will be applied as a mask to the generated seed points. Only points of which the position exists in the mask will be grown.

3.6.2 Zooming in on Cingulate gyrus

The experiment uses the same parameters as the standard configuration experiment. The region mask is defined by the region Cingulate gyrus based on the borderline average brain, since this region almost did not grow on the healthy control groups average features. An illustration of the region mask can be found in figure 3.22. The similarity criterion for region growing is set to 0.8 times the coefficient of variation to reduce the size of the grown regions. The maximum Euclidean distance is set to 15. The results can be found in table 3.26, table 3.27, figure 3.6 and figure 3.24.

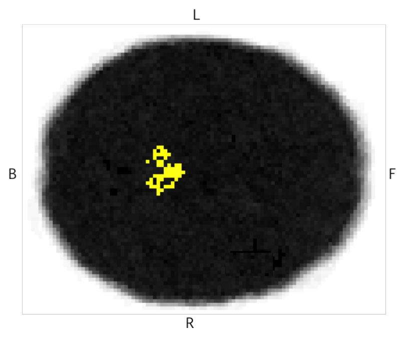


Figure 3.22: Cingulate Gyrus grown in average features Borderline at $z = 47$

Accuracy	68.1159%
Kappa statistic	0.3664

Table 3.26: Results zooming in Cingulate gyrus

3.6.3 Zooming in on Orbitofrontal cortex

The second area selected in the classification process of the standard configuration is grown from inside the Orbitofrontal cortex. Unlike the area grown from inside the Cingulate gyrus, this area did grow well in both the average healthy control group features and the average borderline group features. The experiment uses the same parameters as the standard configuration experiment. The region mask contains the voxel coordinates of the voxels which are part of the Orbitofrontal contex area in our standard configuration. The regions from the average healthy control group features are selected to grow in. The similarity criterion for region growing is set to 0.8 times the coefficient of variation to reduce the size of the grown regions. The maximum Euclidean distance is set to 15. The results can be found in table 3.28, table 3.29, figure 3.25, figure 3.26.

3.6.4 Zooming in on Cingulate gyrus and Orbitofrontal cortex

We wanted to know if the results improve if we zoom in at both areas at once. The experiment uses the same parameters as in section 3.6.2 and section 3.6.3, but uses a mask that contains both regions. The results can be found in table 3.30, table 3.31 and figure 3.27.

	Predicted BPD	Predicted HC	Total
Real BPD	16	19	35
Real HC	3	31	34
Total	19	50	69

Table 3.27: Confusion matrix zooming in Cingulate Gyrus

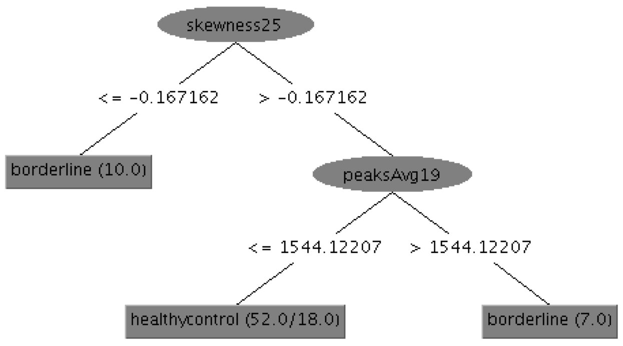


Figure 3.23: Decision tree, zoom in Cingulate Gyrus

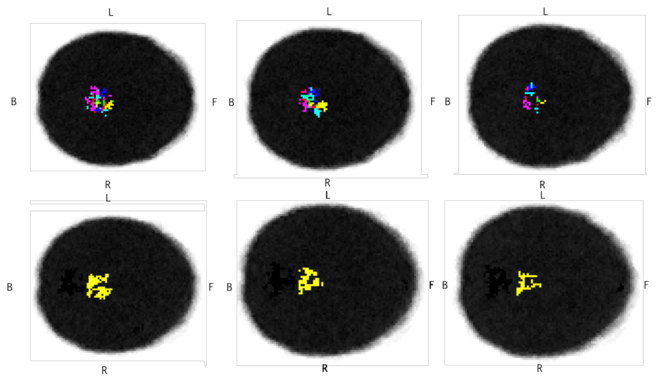


Figure 3.24: Top: Classified areas Cingulate Gyrus. Bottom: Area mask Cingulate Gyrus. Positions: $z = 51$, $z = 52$, $z = 53$

Accuracy	52.1739%
Kappa statistic	0.0429

Table 3.28: Results zooming in Orbitofrontal cortex

	Predicted BPD	Predicted HC	Total
Real BPD	19	16	35
Real HC	17	17	34
Total	36	33	69

Table 3.29: Confusion matrix zooming in Orbitofrontal cortex

Accuracy	76.8116%
Kappa statistic	0.5365

Table 3.30: Results zooming in Orbitofrontal cortex and Cingulate gyrus

	Predicted BPD	Predicted HC	Total
Real BPD	26	9	35
Real HC	7	27	34
Total	33	36	69

Table 3.31: Confusion matrix zooming in Orbitofrontal cortex and Cingulate gyrus

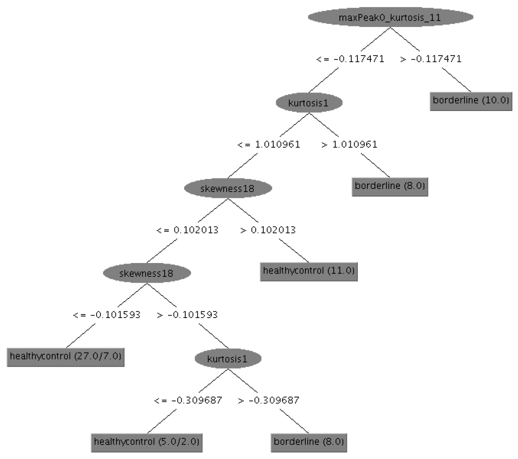


Figure 3.25: Decision tree, zoom in Orbitofrontal cortex

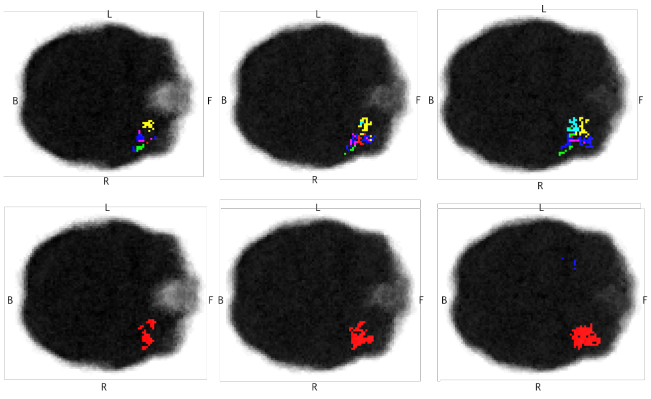


Figure 3.26: Top: Resulting areas when growing in Orbitofrontal cortex. Bottom: Area mask Orbitofrontal Cortex. Positions: z = 26, z = 27, z = 28

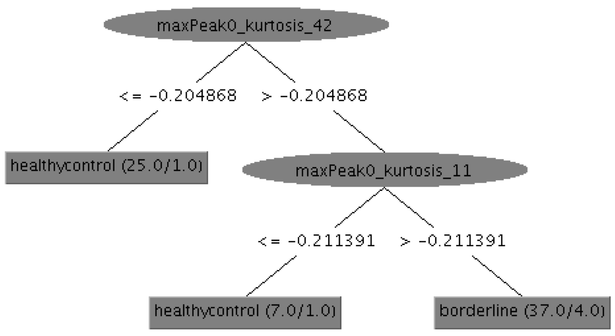


Figure 3.27: Decision tree, zoom in Orbitofrontal cortex and Cingulate Gyrus

3.7 Feature selection

Even after aggregating voxels into regions (see Section 3.5), many final features remain for every subject, which gives rise to the curse of dimensionality [46]. In this section we will look at the possibilities of feature selection in order to decrease the final amount of features used in the classification algorithm and hereby increase the performance of the classifier. Therefore in this chapter we focus on the question: "Do the classification results improve when using more specific areas?".

3.7.1 Feature selection using weka

With the standard configuration there are still 80 feature variables left after aggregating the voxels into regions. With a subject amount 69, this means more features than subjects are present. In order to reduce this amount of features we use Weka's attribute selection tools.

Attribute selection in Weka has two main arguments to configure: the evaluator to use and the search method to use.

For the evaluator we consider several options:

- A subset evaluator, which checks different subsets and evaluates the performance of all those subsets.
- Attribute evaluators, which evaluates the single attributes by their individual properties with respect to the classification. We consider the correlation-, gain ratio and info gain attribute evaluators.
- Principal components, which converts the features into n principal components using orthogonal transformation.

.

For the search method, three options are available:

- BestFirst: The BestFirst algorithm uses greedy hillclimbing augmented with backtracking to search for attribute subsets.
- GreedyStepwise: GreedyStepwise performs in the attribute subset space a greedy forward or backward search.
- Ranker: Ranker ranks the features based on their individual properties.

A comparison of using different feature selecting methods in the standard configuration experiment is shown in Table 3.32. It should be noted that CfsSubsetEval is only compatible with BestFirst and GreedyStepwise and the other evaluators are only compatible with the Ranker search.

Evaluator	*Search method*	*Items to select*	*Accuracy score*	*Kappa statistic*
CfsSubsetEval	BestFirst	N.A.	65, 2174%	0.3007
CfsSubsetEval	GreedyStepwise	N.A.	65, 2174%	0.3007
CorrelationAttributeEval	Ranker	10	60, 8696%	0.2143
CorrelationAttributeEval	Ranker	15	49, 2754%	-0.0126
CorrelationAttributeEval	Ranker	20	49, 2754%	-0.0117
GainRatio	Ranker	10	65, 2174%	0.3024
GainRatio	Ranker	15	71, 0145%	0.4192
GainRatio	Ranker	20	63, 7681%	0.2755
InfoGain	Ranker	10	73, 913%	0.4777
InfoGain	Ranker	15	71, 0145%	0.4192
InfoGain	Ranker	20	63, 7681%	0.2755
PrincipalComponents	Ranker	10	46, 3768%	-0.0704
PrincipalComponents	Ranker	15	53, 6232%	0.0691
PrincipalComponents	Ranker	20	55, 0725%	0.1001

Table 3.32: Comparison between different feature selection methods

3.7.2 No feature selection

To see if the feature selection step is actually needed, we also perform the experiment without feature selection. The results can be found in table 3.33,table 3.34 and Figure 3.28. The results indicate that feature seelction is a required step in the process.

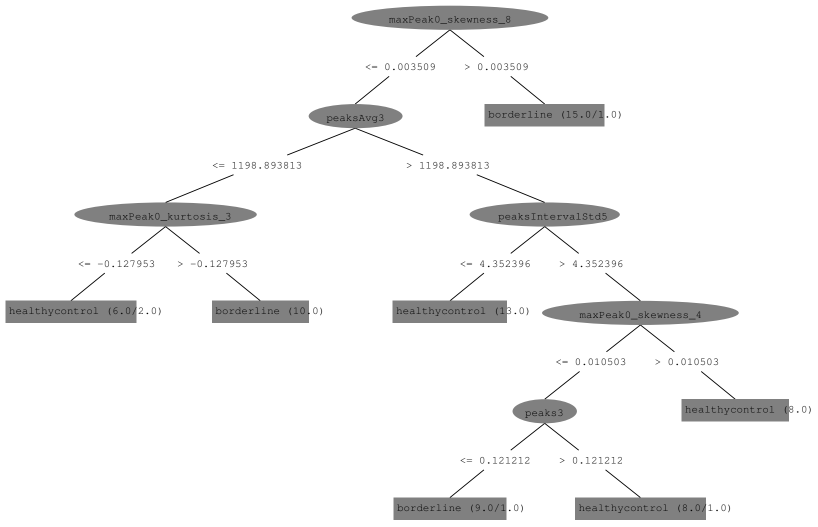


Figure 3.28: Decision tree, no feature selection

Accuracy	47.8261%
Kappa statistic	-0.0419

Table 3.33: Results no feature selection

	Predicted BPD	Predicted HC	Total
Real BPD	15	20	35
Real HC	16	18	34
Total	31	38	69

Table 3.34: Confusion matrix no feature selection

3.8 Other classification algorithms

In this section we ask the question: "Do the classification results improve when using other classification algorithms than a decision tree?". We first try the standard decision tree algorithm with different parameters and then try different classification algorithms.

3.8.1 Tuning C4.5

In our case, changing the parameter values of Weka's J48 algorithm does not change the model it creates most of the time, the only parameter changes that result in different models are changes in *minNumOfObj* and *doNotMakeSplitPointActualValue*, the results are shown in Table 3.35.

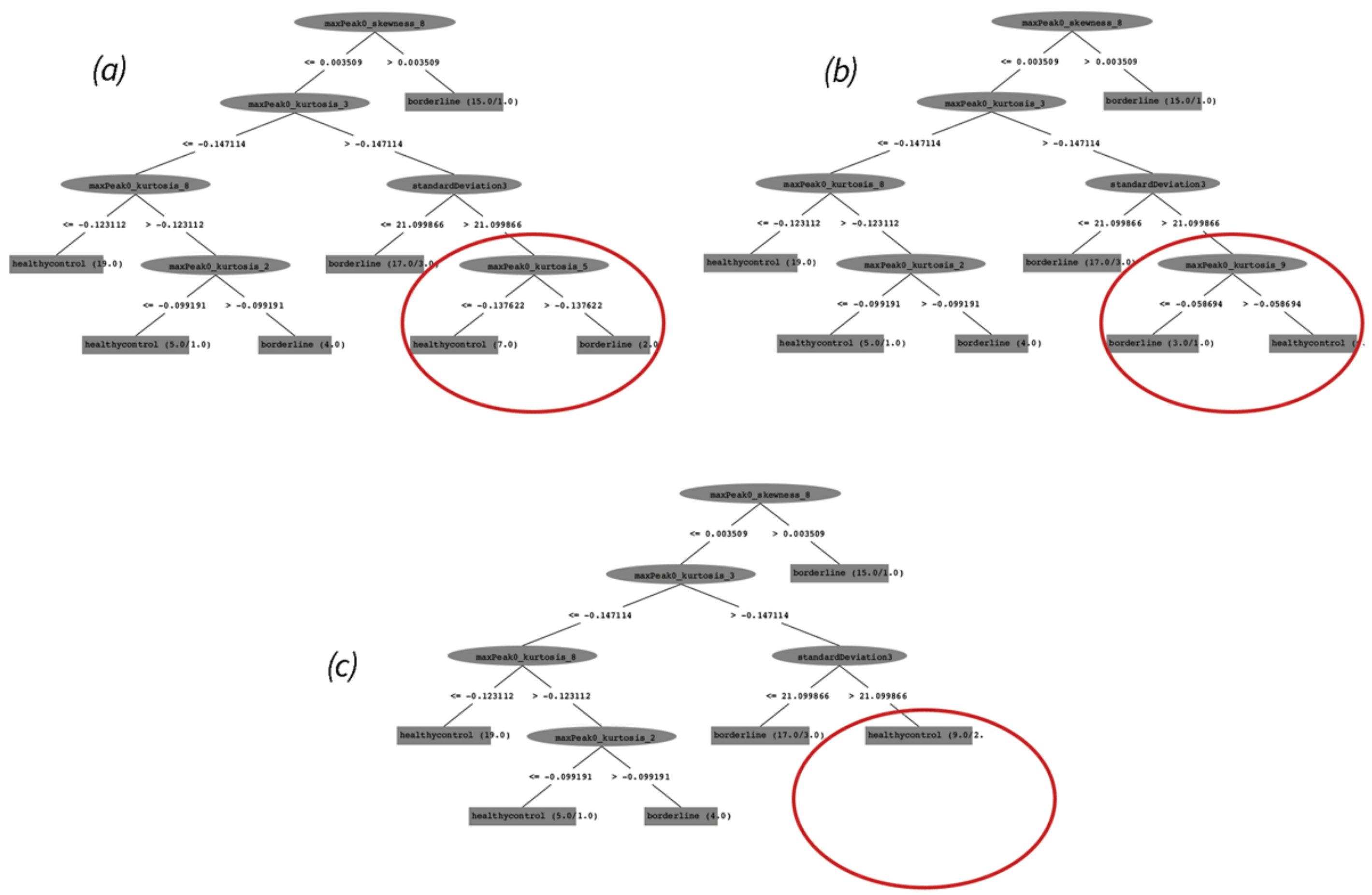
As illustrated in Figure 3.29, the algorithm uses more pruning when increasing the *minNumOfObj* parameter. In (a) and (b) the decision tree splits in a different way on the *maxPeak_kurtosis* feature. In (a) the algorithm assigns a higher kurtosis value to BPD, where in (b) the algorithm assigns a higher kurtosis value to the HC group. Then in (c) the algorithm drops that sub-tree.

3.8.2 Other classification algorithms

In this section the effect of using different classification algorithms is tested. Based on the different characteristics we consider the following algorithms: k-nearest neighbour, random-forest, classification via regression, multi-layer perceptron.

Parameter name	*Parameter value*	*Size of the three*	*Accuracy score*	*Kappa statistic*
minNumOfObj	2	13	65.2174%	0.3042
minNumOfObj	3	13	65.2174%	0.3042
minNumOfObj	4	11	72.638%	0.4489
minNumOfObj	12	7	73.913%	0.4803
minNumOfObj	16	3	65.22%	0.3024
doNotMakeSplitPointOnActualValue	True	9	71.0145%	0.4192
doNotMakeSplitPointOnActualValue	False	9	72.4638%	0.448

Table 3.35: Different parameter settings C4.5



In (a) the resulting tree with *minNumOfObj* = 2 is illustrated, in (b) the resulting tree with *minNumOfObj* = 3 is illustrated and in (c) the resulting tree with *minNumOfObj* = 4 is illustrated. The red circles illustrates the increased pruning when increasing the minimum number of objects at a leaf criteria.

Figure 3.29: Parameter tuning of *minNumOfObjects* prunes the tree

Random-forest

The random forest algorithm [37] is an example of an algorithm that generates many classifiers and aggregates their results. In a random forest each tree is constructed using a different sample of the data. Also the construction is different from a normal dicision tree: in a random forest each node is split using the best among a subset of randomly chosen predictors, instead of splitting using the best split among all predictors. The individual trees are then aggregated using majority votes. A disadvantage of this is that it does not return an easily interpretable model because it only returns the results of the majority vote of all the trees. Also the computation time for the random-forest is much longer than for a single decision tree because it needs to generate multiple trees. The results of the Random Forest algorithm with 1000 iterations can be found in Table 3.36.

k-Nearest Neighbour

In the k-Nearest Neighbour algorithm [31] the classification of examples is based on the class of their nearest neighbours. The amount of neighbours to consider is a parameter. Because the examples are at runtime evaluated with respect to the training examples, the training examples need to be in memory at runtime. A nearest neighbour needs a distance measure to define the definition of *nearest*, by default the euclidean distance is used in Weka. The computation time of this algorithm depends on the amount of features to take into account and on the used distance measure, but in our experiments the computation of the algorithm was almost instant so this will not be a problem. k-Nearest Neighbour is largely dependent on strict feature selection because when useless attributes are given as input for the algorithm, the algorithm threats them as equally important to more useful attributes. The k-Nearest Neighbour algorithm has also has the disadvantage that discovering knowledge from its results is difficult because it does not return a rule based model. We first perform the algorithm with the same feature selection as in the standard configuration, see Table 3.36. In order to take the algorithms characteristics into account, we also perform a test where we scale down the number of input attributes even more. We do this by using the same feature selection as in the standard configuration, but this time with *numToSelect* = 3. Only the three highest scoring features in the feature selection process remain. The resulting attributes are: `maxPeak0_skewness_8`, `maxPeak0_kurtosis_3` and `maxPeak0_kurtosis_5` . The results of this test can be found in Table 3.36.

Classification via regression

Classification via regression [32] is a type of decision tree where at the leaves a linear regression function is present. For every class one regression model is built. Because this algorithm needs to built multiple regression models the computation takes more time than when using C4.5, but in our tests the computation was always less than 0.2 seconds so this will not be a problem. The results can be found in Table 3.36.

Multi-layer perceptron

The multilayer perceptron algorithm available in Weka classifies instances using backpropagation. It uses a multilayer perceptron which is a type of neural network with multiple layers. The computation of a neural network can potentially take a great amount of time, but with this sample size it took less than 0.2 seconds so this will not be a problem with this sample size. It should be noted that a multi-layer perceptron has a great number of parameters, so extensive parameter tuning could potentially create better models. The results of the multilayer perceptron can be found in Table 3.36.

Algorithm	*Additional info*	*Accuracy*	*Kappa statistic*
Random Forest	-	71.0145%	0.4202
k-Nearest Neighbour	number of attributes = 15	68.1159%	0.3633
k-Nearest Neighbour	number of attributes = 3	79.7101%	0.5945
Classification via regression	-	76.8116%	0.5361
Multilayer perceptron	-	68.1159%	0.3627

Table 3.36: Results of different classification algorithms

3.9 Visualization

In this section we will discuss the visualization of steps in the processing of the data. The sub question that will be answered is: "How can we interpret and visualize the results?"
One problem of classification multidimensional data is that it is hard to interpret modifications. It should be possible to visualize pre-processing, processing and selected areas in a 3D image of the brain. The 3D brain image viewer provided with FSL, FSLView [10], is useful to identify the success of preprocessing steps. Unfortunately, it is not able to show values of different voxel-features or to mark areas in different colors. Because visualizing this information is useful in this project, we have implemented an image viewer that is able to visualize areas and multiple features. We also render a number of polygon mesh models of the voxel data.

3.9.1 SVG image exporter

The SVG image exporter can import feature files. Both the files per subject with the subjects voxel-features from the *featureExtraction/* directory and the average voxel-features file can be imported. The SVG image exporter exports the image as 2D images, where each layer is another slice on the Z-axis. It is also possible to specify a *region list file*, which contains all the regions. Voxels in this region will be colored. It is also possible to color square regions. A list of feature names can also be supplied, which makes it possible to export multiple features from the feature file. For each feature, a different set of images will be created. The filename of each SVG image is as follows: `sequenceF_Z.svg` , where *F* is the identifier of the feature, and *Z* is the Z-position of the image. Which feature name belongs to a feature identifier can be found in the file `featureF.txt`.

3.9.2 Javascript image viewer

The Javascript image viewer is able to navigate through the image files which are generated by the SVG image exporter. It uses jQuery for registering key events and AJAX calls. Navigation happens through the W, A, S, D keys. The A and D keys are used to move between different features, the W decreases the Z-position and the S key increases the Z-position. Each time one of these keys is pressed, the F and Z values are updated and according to the new F and Z values, a new picture is loaded with an AJAX call.

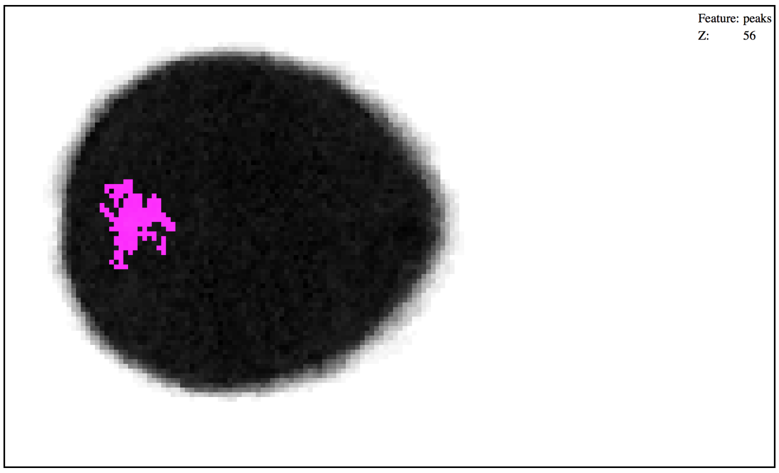


Figure 3.30: Javascript image viewer

3.9.3 Creating polygon mesh models from voxel data

The JavaScript viewer is only able to show one layer of the image at a time. Because of this, it is hard to imagine what regions look like from other angles than the top-down view. We will build a number of 3D polygon mesh models using the exported SVG images.

First the .SVG images are loaded into Matlab. Then for each color channel a separate stack is created. Using the stlwrite [25] library, the separate stacks are then exported to STL models.

With help of our 3D modelling expert Koen Griffioen we also created 3D models of our data. Blender is able to import STL models. Each STL model is imported using blender, and a color and material is assigned to each separate model.

Chapter 4

Conclusion and discussion

In this chapter we conclude this research. First in section 4.1 we provide the answers to the sub-questions. Then in section 4.2 we will answer the main research question.

4.1 Research sub-questions

Question 1: *Which preprocessing steps are required for proper classification?* Multiple preprocessing steps are considered in this project. Because the fMRI data is complex and might contain a lot of noise, it turns out preprocessing the data is a required step in order to compare subjects with each other. A limitation of this research is that not all the variations of preprocessing steps were tested, although we showed that if spatial smoothing or intensity normalisation are not applied, the standard configuration performed worse. If no preprocessing steps are performed the classification results could not be trusted because the classifier tends to select areas near the edges of the brain. Because the size and orientation of the brains differ, registering the images to a standard image is required in order to be able to compare the subjects.

Question 2: *Which features are relevant?* In section 3.3 we described eight voxel-features which each convert the voxels time series into a single variable. As seen in chapter 3, classification models based on those features achieve a far better than random accuracy. It is safe to say that the features which describe the greatest peak in a subjects scan are important features for the models. Further research into these peaks could potentially reveal more information and improve the analysis.

Question 3: *How can we handle the processing of large files?* Doing all of the computation needed for the analysis of the fMRI scans on single computers quickly proved to be questionable. Due to computational and memory reasons another approach was required. By distributing the analysis over the LLSC we had the opportunity to analyse multi-dimensional large files in a practical timeframe. This illustrates the opportunities supercomputers give the scientific world these days. The main steps in our analysis where we needed the cluster was when we had to perform computations for every single voxel or for every group of voxels.

Question 4: *How can we divide the brain into spatial areas which can be used for classification?* The step of dividing the brain into spatial areas is a direct consequence of the multi-dimensionality of our data. By trial and error, we showed that in our research it is beneficial to take the characteristics of the data into account when forming areas. In this project we use the region growing technique as a way to form areas that takes this into account. By applying this region growing on our data, a whole new research focus emerged. We can conclude that there are differences in regions grown based on the average scan of our BPD subjects and on the average regions grown based on HC subjects, although the strength of this conclusion is limited by the relatively small sample size. By applying the region growing technique not only on one single feature, but on a set of features we found regions which are similar in the spatial and feature space.

Question 5: *Do the classification results improve when using more specific areas?* We can conclude that using more specific areas can, but does not necessarily, result in a better model. A combination of the areas Cingulate gyrus and Orbitofrontal cortex seemed more promising to us than zooming in on the individual regions.

Question 6: *How can we deal with a large amount of resulting features?* In section 3.7 we discuss the feature selection we perform on the final features. We can conclude from this that the final feature selection is an essential step in the analysis. All the tested feature selection methods result in a better performance then when we do not perform feature selection.

Question 7: *Do the classification results improve when using other classification algorithms than a decision tree?* Almost all of the classification in this whole project is done using decision trees. From section 3.8 we may conclude that when the goal is to achieve an as accurate as possible prediction model, looking at other classification techniques is beneficial. In particular the k-Nearest Neighbour algorithm shows much potential for increasing the accuracy of the model, although it needs to be noted that this algorithm is very dependent on rich feature selection. With k-Nearest Neighbour the algorithm result can also be interpreted, which is a important characteristic in this research. The classification via regression and the random forest also show some potential although their difference with a binary decision tree algorithm like J48 is not that significant and they are less easy to interpret. Finally the multi-layer perceptron does not perform well and could potentially be resource intensive, although it must be said that a higher subject amount can potentially make this algorithm more interesting.

Question 8: *How can we interpret and visualize the results?* Visualization has been an important part of this project. Although visualization does not directly result in new information, it improves the understandability of the data. The visualisation in 2D serves as a fast and precise way to interpret the data, whereas the visualizations in 3D offer the visualization from multiple perspectives.

4.2 Research question

To answer our main research question: *"To what extent can non-borderline and borderline subjects be classified into those two groups by applying machine-learning paradigms on their functional MRI data?"* we review the whole research and the sub-questions. As a whole we showed that by applying machine learning, the different subjects can be classified at a far better than random chance. By using decision trees it was possible to gather knowledge from the data, especially in combination with the region growing technique. But it must be pointed out that there are some remarks to this conclusion. Exhaustive preprocessing and feature engineering are required in order to make the data useful for machine learning purposes, which makes the machine-learning very dependent on these steps. The decision trees give us some insight into the data, but we are aware of the limitations of this method. The k-Nearest Neighbour algorithm with some parameter tuning has shown to be the best classifier so far. Most of the trees in this project were small decision trees so only a small fraction of the data was eventually used and marked as important. We may conclude that this research area gives much research opportunities and there should definitely be looked at improving the data-driven analysis of fMRI data in relationship to BPD. In our future work we explain some conditions that will contribute.

4.3 Future work

Our research has explored a number of possibilities to analyse fMRI datasets of healthy and diagnosed Borderline Personality Disorder subjects. We have discovered that it is possible to distinguish these two groups to a certain extent, but there is still much to be investigated. We propose a number of topics for further research:

- Increasing the sample size. The classification process was limited by the small sample size. When the sample size increases, more of the combinations of features which describe BPD will be present. This will improve the resemblance of our sample with respect to the population, so that the models will be more realistic. The relatively small sample size also kept us from extensively using more sophisticated classification algorithms like neural networks.
  Increasing the sample size is very resource intensive, so although it would be interesting for further research it is also not likely to be achieved in the near future.
- Zooming in on more sets of areas in the brain. In this research we only focused on two areas, trying

more different combinations of areas could potentially improve the results.
Currently, zooming in on an area is a very long process. Mostly because the mask needs to be generated and applied. Testing more combinations is feasible but it would take a lot of time to generate tests for all anatomical areas in the brain.

- Making the analysis more resource and time efficient. In this project we did not focus on creating an efficient process, we only focused on creating a functional process. Making the process more efficient and even more automated would reduce the time it takes to analyse the data and to gain more knowledge from it.
  The current code is already quite efficient. Although it loops over all voxels in the brain, the voxel-features over the time series are calculated using compiled functions. Therefore, making the code even more efficient would probably require the code to be rewritten in a more efficient programming language such as C++. This process would take a lot of time.

- Improving the region growing technique. From the technical aspect we used a rather basic version of the region growing technique. It could be extended in order to reduce the coefficient of variation inside a region even more.
  Improvements in region growing would be an easy method to extend this research.

- Associating the regions with anatomical regions. By even more connecting the regions found with region growing with the existing domain knowledge of the brain, the analysis could provide a better understanding of the BPD subjects.
  Anatomical regions can be exported from FSLview. Applying these regions as a map would be an easy method to obtain more insight in anatomical regions.

- Using the characteristics of the grown region differences between BPD and healthy subjects as features for classification. This could potentially also be a method to analyse this kind of data. It would be very interesting to see if using these characteristics would create valuable models.
  Applying this would require a major addition to the codebase. The region growing method should be applied to the extracted features of each subject. Features would have to be calculated from the generated regions.

- Our data set consisted in fact of three groups: A healthy control group, a BPD group and an insecure group cf figure 2.2a. In this research, we only focused on the healthy control group and the BPD group. It could be interesting to see how the methods of our research would perform on the insecure group.
  Analysing this new group would be relatively easy. The same code base could be used, but changes in configurations would have to be made to analyse this third group. A possible experiment could be to perform the analysis with the healthy controls and the insecure group or with the borderline subjects

and the insecure group. It would be interesting to see how good the resulting models can discriminate between these groups. Another possible experiment could be to create a model based on the BPD and the HC groups, and then test this model on the insecure group.

- Applying image classification methods, such as the bag of features method, to the image of voxel-features. In our research we have only used averaged areas to classify. Sophisticated black-box methods could possibly improve classification results. Implementing this would require major additions to the code base.

- Improving the visualization process. The visualization process is very time consuming and has many manual steps. It would be interesting to find methods that more easily create 2D images and 3D models from the data.
  Improving the 2D and 3D visualization process would be relatively easy. The code to export the voxel data to 2D SVG images could easily be refactored to use multiple threads. For 3D visualization, the Matlab script for segmentation of colored areas could be automatically executed. Blender has support for Python files and could be automatized to take the required steps to create the final 3D model. Other visualization tools such as Amira could also improve the 3D visualization.

- Taking different sorts of stimulus into account. Our research does not extract separate features for different sorts of stimuli shown during the fMRI scan. There could potentially be differences in accuracy between different kinds of stimuli. Creating separate features for different kinds of stimuli could be an interesting topic for further research. For example, the whole time series could be divided into separate fragments per stimuli. For each segment separate features could then be calculated.
  The main issue with extracting multiple stimuli from the time series is that the computation time significantly increases. Researchers will have to deal with very long computation times or have to figure out methods to improve the efficiency of the voxel-feature extraction process. Another issue is that there are 45 stimuli in total, and the amount of timeframes per time series consists of 163.5 on average, which results in only $3,63$ timeframes per stimuli. Associating a peak with a specific stimuli is a different task with only this amount of timeframes per stimuli. Analysing a different task with less stimuli per timeframe should be considered.

- Connecting the predictions from the classifier with other knowledge. Other knowledge about the subjects than just the fMRI scan is available, which creates the opportunity to connect the predictions of the classifiers with this information. This can potentially give us more information about the reasons why some subjects are predicted incorrectly and potentially the model can improve when taking contextual features into account.

Workload justification

This project has been an intensive collaboration between Jelle van Mil and Chris Onderwater, both doing roughly the same amount of work. The beginning of the project mostly consisted of reading about the domain and analysing the current analysis workflow. During this time we worked together in the same environment, both reading all the papers and other documents. After this phase, we started with building the fundamental code-base we could use in our research. We separated this work into small chunks and intensively discussed decisions regarding the project. After the fundamentals of the project were finished, we both focused on different more specific subjects. Jelle focused on studying which features are relevant, how the brain can be divided into areas, how we can deal with a great amount of features and which classification algorithms show potential. Chris focused on discovering which preprocessing steps are relevant, how large files can be processed, if using more specific areas improve results and how the visualization of the data can be done.

Appendix A

Data

Prefix / postfix	Desription
_chop	Only the brain volumes which are relevant, so the volumes where nothing happens are absent in this file.
example	Specifies that this file only contains 1 example volume out of the total set of volumes
_func_data	Specifies that this file contains a functional brain image
filtered_func_data	Specifies that this file contains a functional brain image which is al- ready preprocessed
_std	File applies to standard space
_SF	File contains the fMRI social feedback scan
_brain	Specifies that this file contains an already brain extracted image
_brain_mask	Specifies that this file contains a mask (zeros and ones) for the brain which specifies what is seen as brain and what is seen as no brain by the brain extraction
_brain_overlay	Specifies that this file contains a overlay image for the brain which specifies the contour of the brain extraction. Mainly used to check manually check the brain extraction process.
hires	Specifies that this file is a high resolution image
_Neg	Specifies that this file contains data for the negative stimuli
_Neu	Specifies that this file contains data for the neutral stimuli
_Pos	Specifies that this file contains data for the positive stimuli
Q	Specifies that this file contains data for the question stimuli

Table A.1: Naming conventions data

Subject ID	*Reason*
B000	Example folder
B125	Noisy scan, unable to preprocess
B126	Noisy scan, unable to preprocess
B133	Noisy scan, unable to preprocess
B216	Scan not present
B222	Scan not present
B302	Scan not present
B305	Scan not present
B309	Scan not present
B315	Scan not present
B319	Scan not present
B320	Scan not present
B326	Scan not present
B331	Scan not present
B339	Scan not present
B342	Scan not present
B344	Scan incomplete
B345	Scan not present

Table A.2: Reasons why some subjects are not usefull in the analysis process

Appendix B

Commands

Preprocessing step	*Configuration line*
Slice timing correction	fsl4.1-slicetimer -i inputFiles -o outputFile -r 2.200000 –down
Brain extraction	fsl4.1-bet inputFile outputFile -F
Intensity normalisation	fsl4.1-fslmaths inputFile -ing 1000 outputFile
Spatial smoothing	fsl4.1-fslmaths inputFile -kernel gauss 2.12 -fmean outputFile
Registration to standard image	fsl4.1-flirt -in inputFile -ref referenceFile -init matrixFiles -applyxfm -out outputFile

Table B.1: Commands used to perform preprocessing

Code line	*Description*
`#!/bin/bash`	Specifies which shell is used
`#PBS -k o`	Specifies that the output logs should be generated in the user directory
`#PBS -l nodes=1:ppn=1 walltime=900:00`	Specifies number of nodes, number necessary of processors and the required time for the job
`#PBS -l mem=4000mb`	Specifies the amount of memory required for the job. We use 4000mb for feature extraction and 8000mb for region growing.
`#PBS -N create\_avg\_features\_B337`	Specifies a job name
`#PBS -j oe`	Specifies that the error stream and the output stream of the job should be merged
source /home/fswkp/pythonenv/venv/bin/activate	Activates the virtual environment on the LLSC cluster
python /home/fswkp/bep24052016/torque_job.py create_avg_features B337	Calls the job script

Table B.2: Example Torque script

Appendix C

Documentation software

C.1 Settings

settings.py is the global settings file where all settings are specified. We will briefly discuss the meaning of all settings.

- *num_areas_x, num_areas_y, num_areas_z*. These settings specify the amount of cubical regions that are generated along the corresponding axis. So 2, 2, 3 will respond to a split along the x and y axis, and a triple split along the z-axis. The total amount of cubical regions will be $2 * 2 * 3 = 12$
- *brainSize*. This specifies the resolution of the supplied brain images.
- *executionDirectory*. This specifies the directory where the program is located.
- *outputDirectory*. The output directory specifies the directory where results will be stored.
- *dataRepository*. This specifies the folder where the data respository can be found. The data repository contains a folder for each subject containing the brain scan of this subject.
- *analysisFilename*. Specifies the file name of the brain scan that will be used in the scan. The string `SUBJECTID` will be replaced by the subject id of each subject.
- *regionListFile*. Specifies the region list file. This is used to generate average features for each region.
- *numFourierVariables*. Specifies the number of fourier variables that will be generated.
- *zoomRange*. Specifies the range where the cubical regions will be created in. It also limits the feature extraction to this area.

- *deTrend*. Applies deTrend to the intensities in voxel time series.
- *paintAreaPureRed*. Paint areas pure red in the .SVG image exporter. So the intensity of pixels painted in an area will always be set to 100%.
- *differentColorRegions*. Create regions with different colors for each regions. There are six different colors and every area automatically gets a color assigned.
- *amountPeaksAnalyse*. This specifies the amount of peaks that will be analysed in depth by calculating the kurtosis and the skewness.
- *regionGrowing*. Toggles region growing functionality. If false, cubical regions will be used.
- *squareRegions*. Toggles square regions functionality. If true it will generate a seed points in the centers of each generated square region.
- *maxRegionSize*. Defines a spatial limit where regions can grow into. 16 means that the region can only grow 8 voxels in each direction.
- *regionsSeeds*. Specifies the region seeds for region growing if squareRegions is false.
- *regionMask*. Applies a mask to the region growing seeds. Only seeds that are in this mask will be grown.
- *std*. This specifies the amount of standard deviations that define the threshold of each feature in region growing.
- *featureStandardDeviations*. This is a dictionary of all the features and their standard deviations.
- *featureRestrictions*. This is a dictionary of all the features and their threshold. A threshold of 0.1 means that the feature may be 10% lower or higher than the value at the seed point, otherwise the voxel will not grow further.
- *maxEuclideanDistance*. This defines the maximum Euclidean distance. If the euclidean distance in the region growing algorithm is at this position, the thresholds will be zero and the algorithm cannot grow any further. The feature restrictions linearly decrease from the starting point towards the maxEuclideanDistance.

C.2 Files

Filename	*Description*
1growRegions.py	Script that grows the regions in the average features of the subjects.
2consolidateRegions.py	Script that consolidates the outputs of different regions into a single region list file.
3extractFeatures.py	Script that extracts the features from the time series of each subject.
4createAverageFeatures.py	Creates average features from the extracted voxel features.
5consolidateAverageFeatures.py	Consolidates the different average features files into a single list.
6exportFeaturesToCsv.py	Exports the extracted average features per area to CSV.
areaToXYZ.py	Prints the X, Y, Z values of a certain area as well as its dimensions.
arrayManipulation.py	Contains functions to create a 2D layer along the Z-axis from voxel-features images.
averageFeaturesSubjectsBorderline.pic	The average features of subjects that have Borderline, as calculated in this research using the in this research provided data.
averageFeaturesSubjectsHealthy.pic	The average features of subjects from the healthy control group, as calculated in this research using the in this research provided data.
createAverageFeatures.py	Contains functionality to create average features per area.
createAverageFeaturesSubjects.py	Calculates the average features of multiple subjects.
events.py	Contains functionality to extract time events. This is not used in this study.
exportImageSequence.py	Script that can export a 3D voxel-features image to SVG image slices along the Z-axis.
featureExtraction.py	Contains functionality to extract voxel-features from subjects.
fsl.py	Script to apply preprocessing to data from all subjects.
getAreaCenters.py	Return the area centers of generated square regions.
growRegionsStd.py	Calculates the standard deviation of a voxel-features throughout the whole brain.
mask.pic	Mask generated and used for zooming in on areas.
regionGrowing.py	Contains functions for region growing.
regionList.pic	Contains a list of regions, which will be used to create average features per region.
selectSeeds.py	Contains the restrictions for region growing seed selection.
settings.py	Contains the settings used in all other files.
svg.py	Contains functionality to convert data in voxel-feature files to SVG images.
torque.py	Contains functions to generate .job files for the Torque engine.
torqueJob.py	Contains the jobs executed by Torque engine.

Table C.1: Files in the execution directory

Filename	*Description*
settingsUsed.py	A duplicate of the settings.py file from the home directory.
regionGrowing/regionList.pic	Contains the coordinates of the found regions. Consists of a list of lists, where the most inside list contains the x, y, z values.
featureExtraction/B*.pic	The * is the unique identifier for the subject. The file consists of a 3 dimensional list where every element is a dictionary of the form {*voxel-feature:value*}
featuresAverage/features_B*.pic	The * is the unique identifier for the subject. The file consists of a single dictionary of the form {*average-feature:value*}.
csv/*.csv	* is the identifier name for the experiment. The file contains the final features in csv file format where the columns are features and the rows are subjects.

Table C.2: Files in the output directory

Bibliography

[1] Atlases - fslwiki. http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Atlases. Accessed: 2016-06-23.

[2] BET brain extraction tool, a tool to separate the brain from non-brain tissue in fmri data. http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/BET. Accessed: 2016-06-07.

[3] Blender blender foundation is a dutch public-benefit corporation, established to support and facilitate the projects on blender.org. http://www.blender.org. Accessed: 2016-06-07.

[4] Brain basics: Know your brain. http://www.ninds.nih.gov/disorders/brain_basics/know_your_brain.htm. Accessed: 2016-06-22.

[5] Brain structure. http://www.indiana.edu/~busey/Q301/BrainStructure.html. Accessed: 2016-06-22.

[6] Chapter 6 - the analysis of fmri data. http://users.fmrib.ox.ac.uk/~stuart/thesis/chapter_6/section6_2.html. Accessed: 2016-06-23.

[7] Debian linux based operating system. https://www.debian.org/. Accessed: 2016-06-07.

[8] FSLutils a package from fsl that contains utilities. http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Fslutils. Accessed: 2016-06-07.

[9] FSLview motion correction for fmri data. http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/MCFLIRT. Accessed: 2016-06-07.

[10] FSLview viewer of fmri data. http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/fslview. Accessed: 2016-06-07.

[11] Grey and white matter. http://www.indiana.edu/~p1013447/dictionary/greywhit.htm. Accessed: 2016-06-22.

[12] ImageMagick a software suit to create, edit, compose or convert bitmap images. http://www.imagemagick.org. Accessed: 2016-06-07.

[13] J48. `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html`. Accessed: 2016-06-22.

[14] jQuery a javascript library. `https://jquery.com`. Accessed: 2016-06-07.

[15] Leiden Instute of Advanced Computer Science. `http://www.liacs.nl`. Accessed: 2016-06-07.

[16] Limbic system. `http://www.indiana.edu/~p1013447/dictionary/limbic.htm`. Accessed: 2016-07-20.

[17] Matlab a technical programming language. `http://nl.mathworks.com/products/matlab/`. Accessed: 2016-06-07.

[18] Measures of skewness and kurtosis. `http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm`. Accessed: 2016-06-22.

[19] NiBabel open source package that provides read and write access to multiple fmri formats. `http://nipy.org/nibabel/`. Accessed: 2016-06-07.

[20] NIfTI-1 Data Format. `http://nifti.nimh.nih.gov/nifti-1`. Accessed: 2016-06-07.

[21] Numpy fundamental open source python package that provides a n-dimensional array object and has linear algebra and random number capabilities. `http://pandas.pydata.org`. Accessed: 2016-06-07.

[22] Pandas open source package that provides high-performance data structures and data analysis tools. `http://pandas.pydata.org`. Accessed: 2016-06-07.

[23] Python programming language. `http://www.python.org`. Accessed: 2016-06-07.

[24] SciPy a python package including a number of open-source packages for scientific research. `https://www.scipy.org`. Accessed: 2016-06-07.

[25] stlwrite a matlab plugin to export patches and surfaces to stl. `https://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-filename--varargin-`. Accessed: 2016-06-22.

[26] Svgwrite open source package that contains tools to write primitive figures to a svg image. `https://pypi.python.org/pypi/svgwrite/`. Accessed: 2016-06-07.

[27] Torque Grid Engine open source resource manager for clusters. `http://www.adaptivecomputing.com/products/open-source/torque/`. Accessed: 2016-06-07.

[28] Weka data mining software in java. `http://www.cs.waikato.ac.nz/ml/weka/`. Accessed: 2016-06-07.

[29] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, 1994.

[30] R. B. Buxton. *Introduction to functional magnetic resonance imaging: principles and techniques*. Cambridge university press, 2009.

[31] P. Cunningham and S. J. Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, pages 1–17, 2007.

[32] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

[33] K. J. Friston, P. Fletcher, O. Josephs, A. Holmes, M. Rugg, and R. Turner. Event-related fmri: characterizing differential responses. *Neuroimage*, 7(1):30–40, 1998.

[34] R. C. Gonzalez and R. E. Woods. Digital image processing. *Nueva Jersey*, 2008.

[35] J. G. Gunderson. Disturbed relationships as a phenotype for borderline personality disorder. *American Journal of Psychiatry*, 164(11), 2007.

[36] S. A. Huettel, A. W. Song, and G. McCarthy. *Functional magnetic resonance imaging*, volume 1. Sinauer Associates Sunderland, 2004.

[37] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[38] Z. M. C. S. C. L. M. M. . B. M. Lieb, K. Borderline personality disorder. *The Lancet*, 364(9432):453–461, 2004.

[39] F. Mörchen. Time series feature extraction for data mining using dwt and dft, 2003.

[40] A. K. Nolan Nichols. The NIFTI file format. `https://brainder.org/2012/09/23/the-nifti-file-format/`, 2012. Accessed: 2016-06-07.

[41] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in cognitive sciences*, 10(9):424–430, 2006.

[42] A. ONeill and T. Frodl. Brain structure and function in borderline personality disorder. *Brain Structure and Function*, 217(4):767–782, 2012.

[43] M. Rouse. What is a voxel? `http://whatis.techtarget.com/definition/voxel`, 2007. Accessed: 2016-06-07.

[44] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobnjak, D. E. Flitney, et al. Advances in functional and structural mr image analysis and implementation as fsl. *Neuroimage*, 23:S208–S219, 2004.

[45] S. Tovino. Confidentiality and privacy implications of functional magnetic resonance imaging. *Scholarly Works*, Paper 393, 2005.

[46] W. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011.

[47] C. van Schie. Presentation notes: Dealing with interpersonal evaluations: Affective and neural responses in borderline personality disorder, 2016.

[48] F. Verbeek. Lecture notes image analysis in microscopy: Segmentation: 1. `http://iammv.liacs.nl/lectures/IA2016-lecture06.pdf`, 2016. Accessed: 2016-06-14.