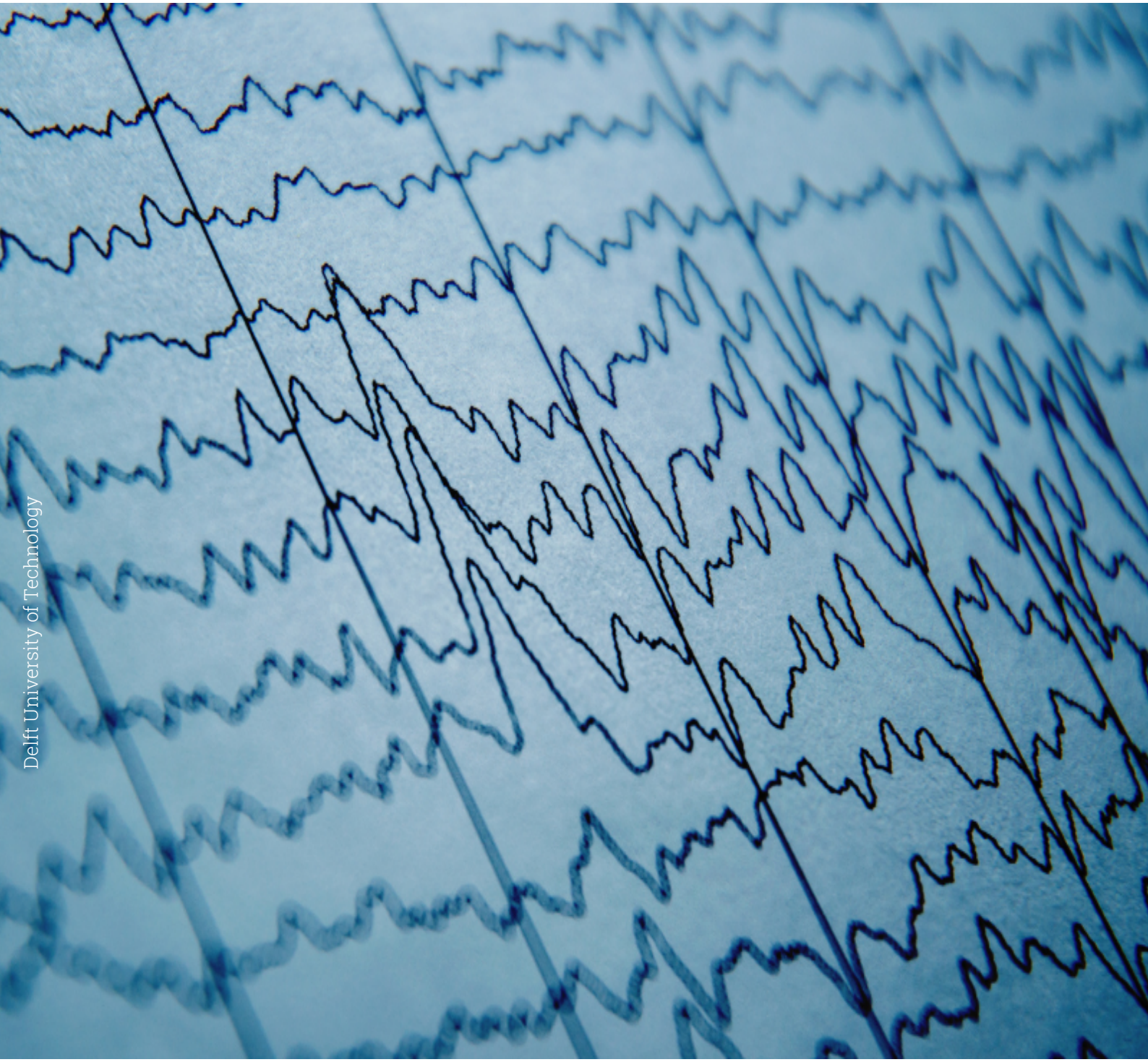
Predictive Analysis of Anti-NMDA-Receptor Encephalitis

using a Random Forest Classifier on EEG Data

Applied Mathematics
F.I.M. Lückerath



Delft University of Technology

TU Delft

Erasmus MC
University Medical Center Rotterdam

Predictive Analysis of Anti-NMDA-Receptor Encephalitis

using a Random Forest Classifier on EEG Data

by

F.I.M. Lückerath

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 4, 2023 at 10:00 AM.

Student number:	4551680	
Project duration:	November 14, 2022 – July 4, 2023	
Thesis committee:	Prof. dr. ir. G. Jongbloed,	TU Delft, supervisor
	Dr. R. van den Berg,	Erasmus MC, supervisor
	Prof. dr. ir. M. van Gijzen,	TU Delft, thesis committee

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Erasmus MC
University Medical Center Rotterdam

Preface

This thesis marks the end of my master's degree in Applied Mathematics with a specialisation in Mathematical Data Science. The research I have conducted for the past eight months is a final touch to my academic journey. Throughout my years at the EEMCS faculty, I have gained extensive knowledge in mathematics and, ultimately, focused on the discipline of data science. It fills me with excitement to conclude my degree with a research project that demonstrates practical application of mathematics and data science in a socially relevant challenge. Over the last eight months, I have realized that the relevance of a project ignites my motivation to thoroughly explore the subject. Despite various challenges throughout the research, I have enjoyed the research process immensely and recognized it as a valuable opportunity for academic and personal growth.

Firstly, I want to express my gratitude to my thesis supervisors, Geurt Jongbloed and Robert van den Berg, for their guidance throughout this research. Thank you for the continuous motivation that has encouraged me to explore new paths and navigate through this research. This research was carried out in collaboration with Erasmus MC and I would like to extend my appreciation to both supervisors for fostering the interdisciplinary collaboration between TU Delft and Erasmus MC. It is truly inspiring to witness the synergy between both fields. Lastly, I would like to thank Martin van Gijzen for being part of my thesis committee and I appreciate your time and effort.

Last but not least, I thank my parents whose support has been invaluable throughout my seven years of studying at TU Delft. They are my biggest inspiration. Furthermore, I am incredibly grateful to my brothers and friends for their unwavering support. In particular, I cannot thank my friends who were graduating alongside me enough. The encouragements and discussions we shared during coffee breaks and lunches made the eight months all the more enjoyable. As one era comes to a close, a new chapter begins. With the learned lessons of the past few years and the support of my family and friends, I embrace this transition with confidence.

F.I.M. Lückerath
Rotterdam, June 2023

Summary

During the initial phase of diagnosis, patients with anti-NDMA-receptor encephalitis (anti-NMDARE) often experience severe symptoms that significantly impact their quality of life. Anti-NDMARE is an autoimmune disorder affecting the brain, with electroencephalography (EEG) playing a vital role in diagnosis and treatment. Identifying EEG patterns associated with positive or negative prognosis is crucial for adjusting treatment intensity. Improved understanding of diagnosis, prognosis and treatment could enhance the quality of life for anti-NDMARE patients. This thesis aimed to analyse the EEG data with Machine Learning (ML) to predict which patients exhibit positive recovery after 12 months of standard treatment.

To predict the outcome after 12 months, a Random Forest (RF) classifier was constructed using available EEG features. The EEG dataset exhibited a clustered structure due to multiple values for each patient's EEG features. Three approaches were considered to handle this clustering: ignoring clustering, reducing clustering to independent observations, and explicitly accounting for clustering. The first two options were explored in this research. Another prominent challenge encountered early in the research was the class imbalance, which was addressed by under- and oversampling the dataset.

For the simulation sets, under- or oversampling did not yield the desired effect, as the normal sets demonstrated comparable or even superior performance compared to the the under- and oversampled sets. However, under- and oversampling improved the performance scores for the real dataset. Reducing the clusters to independent observations did not achieve high performance scores compared to ignoring clustering, both in the simulation and real data cases. Furthermore, in both cases, RF models using the EEG sets outperformed those using principal component analysis (PCA) on the clustered EEG set.

Although the performance metrics scores were not yet optimal, important features for determining class labels were identified, providing a good understanding of the dataset. Mean Decrease in Impurity (MDI) and SHAP algorithm highlighted the significance of connectivity-related features in the reduced clustering to independent observation setting. The relevance of these features became evident upon calculating the mean, minimum, or maximum. In the EEG setting, MDI emphasized the importance of the features `deltapower`, `sampleentropy` and `occipital`-related features. These features remain important in the reduced set. SHAP, in addition to prioritizing the same features, offered insights into how specific features contribute to the prediction of a specific observation, enhancing interpretability.

The challenges for the RF classifier in the case of anti-NDMARE are class imbalance and accurate classification of the minority class. Under- and oversampling techniques successfully improved classification of minority class observations for the original EEG set. Concluding, this set is strongly encouraged to be utilized over all sets when aiming to classify EEG features. However, this set overlooks the clustering aspect, leaving room for optimization in future research to address this limitation. Additionally, it is recommended to explore the potential of a Convolutional Neural Network (CNN) for accurate classification of raw EEG signals. Its exploration was beyond the scope of this research.

Contents

Abbreviations

AI	Artificial Intelligence
anti-NMDARE	Anti-NMDA Receptor Encephalitis
CNN	Convolutional Neural Network
CV	Cross-Validation
DL	Deep Learning
DT	Decision Tree
ECG	Electrocardiogram
EEG	Electroencephalogram
GradCAM	Gradient-weighted Class Activation Mapping
ICC	Intraclass Correlation
IQR	Interquartile Range
MDI	Mean Decrease Impurity
ML	Machine Learning
NN	Neural Network
OOB	Out-of-Bag
PC	Principal Component
PCA	Principal Component Analysis
RF	Random Forest
RNN	Recurrent Neural Network
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machine
XAI	Explainable Artificial Intelligence

1 Introduction

This thesis aims to explore effective classification methods for predicting the outcome after 12 months of anti-N-methyl-D-aspartate-receptor encephalitis (anti-NMDARE) patients based on available EEG features.
During the initial stage of diagnosis, patients with anti-NMDARE often experience severe symptoms that significantly impact their quality of life. However, after 12 months, a majority of patients having received standard treatment is well recovered. The factors that contribute to the severity of the disease remain unclear. Is it possible to predict which patients will exhibit positive recovery after 12 months of standard treatment after diagnosis?

1.1. Anti-NMDA-receptor encephalitis background

Anti-NMDARE is an auto-immune disorder that affects the brain. It occurs when the body's immune system produces antibodies that mistakenly attack NMDA receptors in the brain. The disease has only recently been discovered and categorized, prior to which patients were labeled as psychotic. The majority of anti-NMDARE patients are women of childbearing age or children [1]. Diagnosis and treatment is still challenging, because patients with anti-NMDARE have a diverse array of symptoms[2]. Early diagnosis and immediate immunotherapy can significantly improve the outcome [3]. Due to its simplicity and quick results, electroencephalography (EEG) has the potential to serve as a valuable tool for diagnosing anti-NMDARE. An EEG recording measures the electrical activity of the brain by attaching electrodes to the patient's head. Currently, the signals in the EEG recordings are manually analyzed by clinicians to identify any patterns and to diagnose cases of anti-NMDARE. This approach is labor-intensive and requires 9 to 12 months training. This is where EEG evaluation using Machine Learning (ML) can prove to be valuable. Furthermore, ML might analyse the EEG recordings more accurately and specifically. Typically, EEG recordings exhibit a non-specific pattern characterized by a diffuse slowing of background activity [4]. The rhythm speed can be considered as a rough measure for classification. An EEG displaying normal rhythms is associated with a favorable outcome, whereas a delayed EEG is associated with a poor outcome.
For this research, the EEG recordings are summarized in multiple features that describe the signal. By focusing on whether and when patterns appear in the EEG features of patients with anti-NMDARE, there is an opportunity for enhancing the diagnosis, description and differentiation of anti-NDMARE [2]. The ability to identify which EEG patterns relate to a positive or negative prognosis is crucial. By distinguishing which patients have a favorable or unfavorable long-term (12 months) outcome, the intensity of the treatment can be adjusted accordingly. This approach ensures that the immunosuppresive medication treats the patients effectively while minimizing the risk of severe side effects.

1.2. Machine Learning in healthcare: background information

ML has emerged as a powerful tool in healthcare that can help medical professionals diagnose and treat patients. ML involves constructing or training statistical models using real-world data to predict outcomes or classify observations. By analyzing high-dimensional data, ML can identify patterns that are not known to the specialists (yet). Rahmani et al. recognize the opportunities of ML methods in

improving the quality of healthcare in general [5]. The adoption of ML technologies realizes cost reduction, effective drug discovery and improvement of therapeutic results [5]. However, [5] also stresses that practitioners should identify the strengths and weaknesses of the ML methods in order to improve healthcare decision-making. ML has increasingly been utilized as effective tool that provides medical diagnoses and treatment recommendations. Craik et al. specify this for EEG data particularly and say that automatically classifying the EEG signals reduces the dependence on trained professionals and makes EEG usage more practical [6].

The new generation of healthcare practitioners is encouraged to embrace the ML challenge. They should acquire the skills to understand, develop and control the ML techniques to improve patient care [7]. According to Rubinger et al. healthcare professionals demand more interpretability, model performance, and reliability compared to other domains [8]. The authors emphasize that the quality of data fed into the ML algorithm is crucial for accurate assessments. Rajula et al. [7] state the challenge of statistical inference of large data sets in the setting of healthcare. This challenge in ML methods is to identify genuine patterns, prevent false classifications and make confident predictions on possible diagnosis and treatments. Consequently, ML methods emphasize not only predicting the outcome as accurately as possible, but also understanding the relationship between variables.
Rajula et al. [7] found that ML in the medical field is a rising topic and that the number of articles being published in the areas of diagnostics and drug discovery is growing exponentially. Rahmani et al. [5] continue that most research on ML methods in healthcare is about the field of diagnostics and that there are very few papers in the treatment field that use ML techniques [5]. This thesis attempts to examine the application of machine learning algorithms in the case of anti-NMDARE and assess their potential in improving the treatment of patients.

Loh et al. [9] explain the low acceptance of Artificial Intelligence (AI) in healthcare applications, because ML methods are seen as black boxes. AI includes a broad range of technologies and approaches, including ML, that are capable of performing tasks that typically require human intelligence. Explainable AI (XAI) is a field within AI that focuses on making AI and thus ML models more interpretable for practitioners, so that they can trust their decisions and better understand how they work. Loh et al. also say that visualization of XAI can be beneficial in supporting clinical decisions. The first high-quality paper on XAI was published in 2018, suggesting that the field of XAI is relatively new [9]. According to Confalonieri et al. [10], the initial ideas surrounding explainability in AI can be traced back to the mid-1980s. Recent advancements and achievements of XAI in ML technologies resurfaced in 2018. This thesis focuses on XAI as an important topic, as understanding how machine learning models work is becoming more crucial in many fields. By exploring XAI methods and their practical use, this thesis aims to contribute to the development of more trustworthy and dependable machine learning models

1.3. Framework for model development

Craik et al. [6] suggest three important steps in EEG classification tasks, artifact removal, feature extraction and classification. This thesis focuses on the latter, which is developing a classification model with the available EEG data. This research had no influence on the artifact removal and signal analysis techniques used to extract features from the EEG signals. According to Rahmani et al. [5] a good framework for model development in healthcare consists of five phases: problem definition, dataset, data preprocessing, ML model development and evaluation. In the same way, Chen et al. [11] outline the steps for developing an ML model for healthcare: ”selecting the appropriate problem, curating datasets, developing and evaluating model performance.”

This thesis follows those steps. In Chapter 2 various classification methods are explained and compared. Chapter 3 presents an overview of the anti-NMDARE data that is available for the classification method. Exploratory data analysis lets us make few assumptions on the outcome of the classification method. In Chapter 4, the Random Forest classifier is elaborated on and how the classifier can specifically be applied to the anti-NMDARE data. Chapter 5 includes the research design and techniques used for developing the Random Forest classifier for the anti-NDMARE data. Before we continue to the results of the anti-NMDARE data, we present a simulation study in Chapter 6. The results from the application and development of the Random Forest classifier on the anti-NMDARE data is presented

in Chapter 7, where the model's performance is evaluated. In Chapter 8, there are some important concluding remarks and suggestions for further research.

2

Classification

In this chapter, first a brief explanation of classification is given. Afterward, the most common classification methods used are discussed and compared. Why are specific methods better for our problem statement? And what are the characteristics of the methods we look at? After clarifying this, the decision for the classification method is further explained.

2.1. Introduction on classification

Classification is a task within the field of ML. A classification method aims to predict a qualitative response variable, like a class label, based on an input variable which is usually high dimensional. For example, given patient-specific info, does the patient classify as diabetic or non-diabetic? The output, or class label (eg. diabetic or non-diabetic), is usually denoted as y. The input variable is a set of features (eg. blood pressure, glucose levels, etc.), and is denoted by $X = (X_1, \ldots, X_p)$. Here, p denotes the number of features.

ML can be divided into supervised and unsupervised learning. When a supervised learning model is trained, it knows the response variables y. The goal is to find a classifier model f that predicts the class label $\hat{y}$ based on the observed features X for each individual patient as accurately as possible, [12]. Here, $f(X) = \hat{y}$ refers to the prediction of the class label, whereas y is the true class label. There are multiple ways to assess the quality of a classification model. The comparison of different methods is discussed in Section 2.3.

Formulation for the anti-NMDARE classification problem This thesis aims to develop a classification model that can predict the outcome of anti-NMDARE for different patients based on their EEG data.

The prognosis and severity of the disease is based on a clinical score called modified Ranking Scale (mRS) ranging from 0 to 6. An mRS of 0 means no symptoms up to an mRs of 6 means the patient is deceased. The clinical score is assessed at multiple time points. During the first six weeks, the majority of patients exhibit high scores. After 12 months, most patients are well recovered. Scores after 12 months from 0 to 2 are considered as a good outcome and score from 3 to 6 are defined as a bad outcome [1].

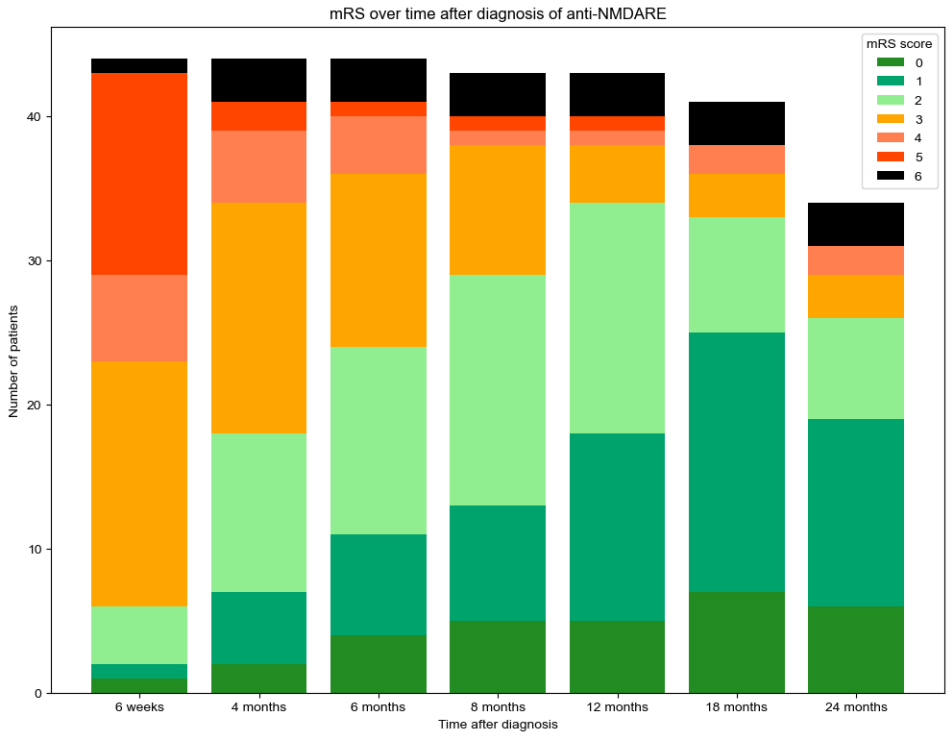


Figure 2.1: mRS scores for anti-NMDARE

We choose to use 0 and 1 by splitting the scores 0 to 6 into a favorable and unfavorable outcome. The output variable for this problem is therefore binary with values

$$y_i = \begin{cases} 1 & \text{if patient } i \text{ belongs to class with a negative outcome,} \\ & \text{meaning mRS scores of } 3 \text{ to } 6 \text{ after } 12 \text{ months} \\ 0 & \text{if patient } i \text{ belongs to class with a positive outcome,} \\ & \text{meaning mRS scores of } 0 \text{ to } 2 \text{ after } 12 \text{ months} \end{cases} \tag{2.1}$$

The input EEG data of the anti-NMDARE classification problem can be divided into two structures

1. Extracted features, in our case 29, from the epochs, which are segments of the EEG signals of patient i
$$X_i = (X_{i,1}, \ldots, X_{i,29})$$
2. The signals of raw EEG data per patient i

Each patient i has its own input features X_i and true label y_i. We have a total of 44 patients, represented by I. Each patient receives an output class label $\hat{y}_i$ after the model was fit. The classification methods need to be able to use these structures as input variables. Furthermore, there is data available that represents general characteristics such as age, gender or findings in MRI scans. This thesis is focused on the EEG signals. In Chapter 3, the dataset is described in more detail.

2.2. Characteristics

In comparing classification methods for the case of anti-NMDARE, it is essential to examine their characteristics in terms of accuracy, interpretability, robustness, and quality. We provide a brief overview of these characteristics. When discussing a model's performance, it is important to recognize that it is about a specific instance of a classification method, while the use of the term method typically refers to the classification method as a whole.

Accuracy

Accuracy is often used as a primary performance metric since it determines the model's ability to correctly classify observations. For quantifying the accuracy of a classification model f, the fraction of misclassified observations is computed

$$\frac{1}{I}\sum_{i=1}^{I}\mathbb{1}(\hat{y}_i \neq y_i) \tag{2.2}$$

Interpretability

Besides accurately predicting the class labels, the classification model must be interpretable in the case of anti-NMDARE. Since the output of our model is whether a patient has a positive or negative prognosis for the treatment, it should be clear how model $f(X)$ arrives at predicted outcome $\hat{y}$. ML often has the perception of being a black box, as it is often not clear why and how an ML model came to the prediction. Interpretability refers to how easy it is to understand the ML model and its prediction. Various techniques and methods, such as visualization, feature importance, and explanation models can help in understanding the classification models. In general, a more complex classification method is less interpretable. There is a tradeoff between flexibility and interpretability in various classification methods. As these methods become more flexible, their interpretability tends to decrease, as shown in Figure 2.2 [12].

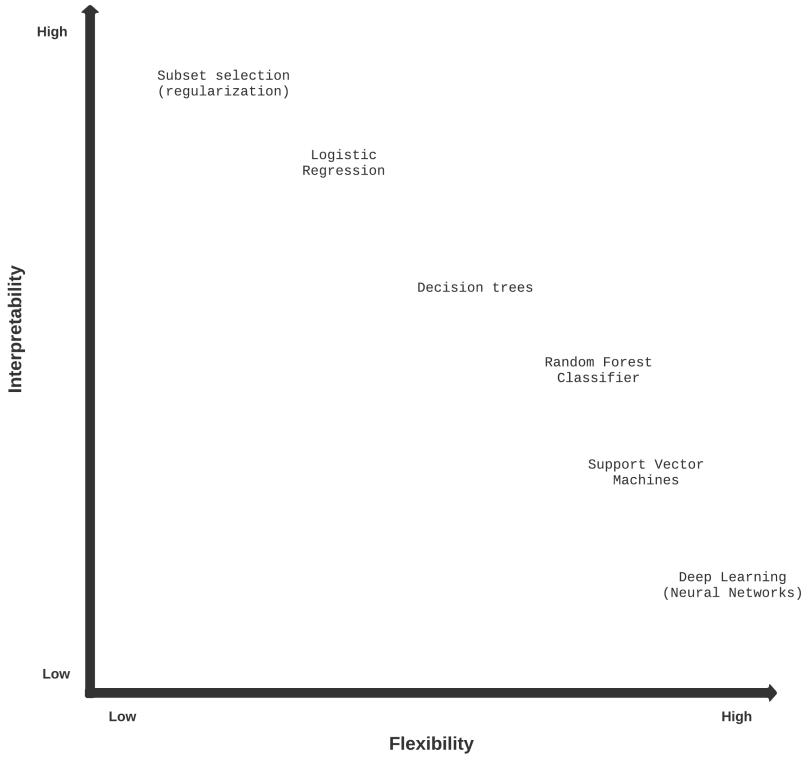


Figure 2.2: Tradeoff between flexibility and interpretability for various classification methods [12]

Breiman highlights the conflict between prediction, accuracy, and interpretability. He suggests that the best way to handle this conflict is to focus on predictive accuracy first and subsequently try to understand the underlying factors [13]. He further shifts the focus of the interpretability versus accuracy dilemma by saying that the objective of a model is accurate information on the relationship between the predictor variables and outcome, instead of interpretability.

Robustness

A classification method that is robust can handle outliers and minor variations in data with minimal impact on its performance. Small changes in the dataset or different initialization of tuning parameters,

the parameters that need to be set before a classification model is trained, can significantly impact the model's performance. Robust methods mitigate the influence of sensitivity within a model. Multiple definitions of robustness exist; however, when referring to robustness, we intend to follow the mitigation of sensitivity. By mitigating sensitivity, the stability and reliability of a model improves. The Rashomon effect, which shares similarities to robustness, is a concept Breiman discussed in [13]. The Rashomon effect implies that a slight alteration in the data leads to a change of model [13]. Ideally, a model performs the same on different datasets. The variance of a classification method can indicate how much a model changes when there is a modification in the dataset. High variance occurs when a small change in the training data leads to a substantial change in the model. More flexible models typically have higher variance as they adjusts to the specific dataset but not generalize on unseen data. However, these models have a low bias. Bias refers to the tendency of a model to make systematic errors in its predictions. A high bias implies underfitting, since the model's simplicity leads to a significant amount of error. A low bias might imply overfitting, which occurs when a model performs well on the training data but poorly on unseen data. Choosing the right model, there is a tradeoff between variance and bias. Ideally, a model achieves low values for both variance and squared bias. The challenges lies in identifying these models [12].

Quality

Performance metrics are considered to evaluate a model's quality. Caruana et al. [14] argue that it is important to compare models on multiple performance metrics. While a particular classifier may perform well on one metric, it may fall short on optimality when assessed using other metrics [14]. Performance metrics provide insights into the quality of a model including accuracy, sensitivity, specificity, precision, F1-score, and the ROC curve. As performance metrics are calculated after training a model, this section covers their explanation, while Chapter 7 discuss their usage.

Swathy et al. [15] argue that not only accuracy, Equation (2.2), should be considered as a performance metric, but also other metrics, depending on the demands of the diagnosis. The anti-NMDARE classification problem is a binary problem with class labels 0 and 1, see Equation (2.1). Binary classification models can be evaluated using a confusion matrix. Although it may seem counter-intuitive, the positive class in a binary classification problem (label 1 in Equation (2.1)) can be associated with the negative outcome, and vice versa. In the context of the confusion matrix, a negative prediction (label 0) often means low risk, whereas a positive prediction (label 1) is associated with high risk. The confusion matrix is displayed below.

		Actual class	
		class label 1	class label 0
Predicted class	class label 1	True Positives (TP)	False Positives (FP)
	class label 0	False Negatives (FN)	True Negatives (TN)

Table 2.1: Confusion matrix

True Positives are the observations that are classified as class 1, and also have actual class label 1.

False Positives are the observations that are classified as class 1, but have actual class label 0.

False Negatives are the observations that are classified as class 0, but have actual class label 1

True Negatives are the observations that are classified as class 0, and also have actual class label 0

In the healthcare sector, analyzing the misclassifications is often considered as more important than the correctly classified observations. Rudinger et al., [8], underline that in the healthcare sector where clinical decisions are made, the cost of model misclassification or error is high. Also Zhai et al., [16], state that higher sensitivity is more important for clinical applications, along the side of specificity. According to Hastie, the class-specific performance is generally emphasized in the domains of medicine and biology [12].

Specificity is defined as the ability of the model to correctly predict observations that have a negative class label, or favorable outcome after 12 months.

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{\text{\#patients correctly classified as negative, class } 0}{\text{\#patients with actual negative class label } 0} \tag{2.3}$$

A high value for specificity implies that the model is correctly classifying most of the negative class labels, as it is the percentage of negatives the model correctly classifies. A low value for specificity means that more patients who actually have a lower risk for a poor outcome are misclassified as high risk. The Type-I error is known as the False Positive rate, which equals $1 -$ specificity.

Sensitivity, or recall, on the other hand, is the ability of the model to correctly classify the patients that have a positive class label, or poor outcome after 12 months.

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{\text{\#patients correctly classified as positive, class } 1}{\text{\#patients with actual positive class label } 1} \tag{2.4}$$

A high value for recall implies that the model is correctly classifying most of the positive class labels. A low value indicates that the model misclassifies relatively many higher-risk patients to the lower-risk class. Here, $1-$sensitivity is the Type-II error. If reducing Type-II error is preferred, the aim is to optimize the recall metric or the True Positive rate. However, to get a comprehensive view of the performance of the classifier, both metrics are considered.

ROC curve is a graphical tool that visualizes the tradeoff between the True Positive (sensitivity) and False Positive (1-specificity) rate across all possible thresholds. Figure 2.3 displays an example of the ROC curve. The overall performance of a classifier can be summarized by the area under the ROC curve, also noted as ROC-AUC. A desirable ROC curve will hug the top left corner, indicating optimal performance. Therefore, a larger ROC-AUC generally indicates better classification performance [12]. The ROC-AUC can obtain a maximum values of 1.

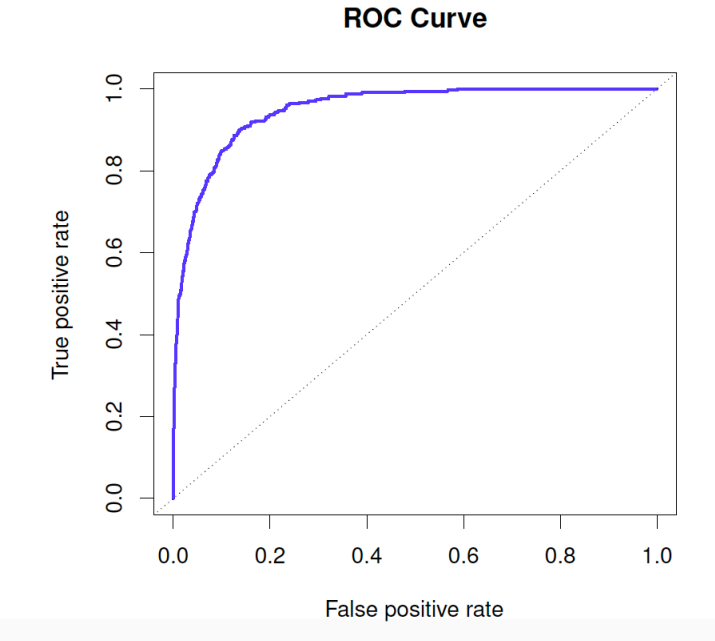


Figure 2.3: Example of ROC curve [12]

Precision is the fraction of correctly classified observations of the positive class out of all predicted positives. A high precision indicates a high level of confidence in the model's ability to predict the positive class labels. Precision is a commonly used metric when the goal is to minimize false alarms.

False alarms are instances that are predicted as positive but are actually negative.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{\#patients correctly classified as positive, class } 1}{\text{\#patients classified as positive, class } 1} \tag{2.5}$$

F-score is the harmonic mean between the recall and precision. It gives each metric the same weight. The F-score is often used for imbalanced datasets since it accounts for both false positives and false negatives.

$$\text{F-score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{2.6}$$

2.3. Classification methods

Before developing a classification model for the dataset considered in this thesis, existing methods are analyzed and evaluated in order to compare them. Within the field of classification, there are several ML methods to predict y. Rahmani et al. [17] formulated the advantages and disadvantages of various supervised methods, see Table A.1 in Appendix A. From this table, we decided to explore the following models for the anti-NMDARE problem: Decision Trees (DT), Neural Networks (NN), Random Forests (RF), and Support Vector Machines (SVM). The need for independent features in the Naive Bayes model makes the method unfavorable for our problem. Furthermore, k-Nearest Neighbors is regarded simple in the sense that it memorizes the training observations and is distance-based which may struggle to capture patterns present in the data. Considering the disadvantage of sensitivity to the data structures, we have decided not to include this method in our analysis. Additionally, logistic regression is included in the comparison since it is a widely recognized binary classification method.

Caruana et al. [14] state that it depends on the data which learning algorithm performs best. Hastie et al. [12] claim that data with few training observations require a model that demonstrates low variance. In the next paragraphs, we compare various classification methods for data structure (1) extracted features, and (2) raw EEG signals, which were previously defined.

According to Craik et al. [6], classification tasks in the field of neurology often employ common supervised learning techniques like SVMs and DTs. Conventional ML approaches face limitations when it comes to handling raw and unprocessed data [18]. In the case of the EEG data, the ML methods require features to be extracted from the EEG signals. Deep Learning (DL) methods, which are a specific field within ML, in general require less pre-processing steps and are able to use the raw EEG signals as input. NNs are the building blocks of DL methods [12].

Comparison

Logistic Regression

The linear classifier, logistic regression, is a popular learning algorithm. Logistic regression predicts the probability that an observation belongs to a particular class. Classification predictions made by linear classifiers rely on a linear combination of the features [19]. Logistic regression maps the linear combination of features to a probability scale using the logit function. Algani et al. [20] add that the linear structure of logistic regression makes it easier to interpret the pattern but may not be flexible enough to make accurate predictions. In cases where there is a substantial separation between the classes, the parameter estimates exhibit high instability, which causes challenges for interpretability [12]. Furthermore, the logistic regression method has its limitations when it comes to dealing with high-dimensional data.

SVM

Compared to logistic regression, SVMs can handle high-dimensional data well and do not assume a parametric relationship between features and class label [19]. However, SVMs require the classes to be linearly separable. The classification method aims to find a hyperplane, linear combination of features, that separates the classes perfectly. A soft margin can be added to the hyperplane to allow misclassifications, if a perfect separated boundary is not possible. Another variation of the SVM is using kernel functions that enlarge the feature space to higher dimensions in which the classes are linearly separable. A disadvantage of SVM is the need to define a kernel function which diminishes the interpretability of the model [19].

DTs and RFs

In a DT, features are used as internal nodes with branches representing a decision and each leaf node signifying the outcome. RFs leverages a combination of DTs to arrive at a prediction based on the majority vote of all DTs. Aggregating over multiple classification models improves accuracy and stability. However, Breiman [13] says that this multiplicity problem needs more attention, as each model can vary over the perspectives on model's interpretability and the significance of features. Caruana et al. [14] saw that RFs outperformed single DTs on all problems. Furthermore, RFs can provide high levels of accuracy and handle large datasets with a large number of variables. Moreover, RFs can automatically balance datasets [20]. Breimans research additionally shows that RFs are able to give more reliable information about the relationship between variables than logistic regression [13]. However, the interpretation of how the model arrives at its prediction is still challenging. Caruana et al. [14] found that across multiple performance metrics, tree-based models like RFs, a variant of the SVM, and neural nets were the strongest models. In an earlier paper of Caruana et al. [21], the authors concluded that similar to NNs and SVMs, bagged trees, trees that use bootstrap aggregation, are characterized as a reliable, versatile and high-performing classification method.

NN

A NN classifier is modeled after the structure and function of the human brain. The network consists of an input layer, (multiple) hidden layers and an output layer. The layers process and transmit information using neurons. The neurons use mathematical equations to combine and process the information from one layer to another. They eventually generate a final output for the output layer. According to Westreich et al. [19] an advantage of an NN compared to logistic regression is that it can deal with high-dimensional data. Individual features may have a slight influence on the probability of belonging to a specific class, while when considered as a group, they can effectively and accurately classify observations [19]. A disadvantage is that training an NN is still very hard and complex, making the predictions hard to interpret. The inner workings of the layers can make it challenging to understand how the output is connected to the input.

The paper of Hagan et al. [22] investigated ML methods for predicting cardiovascular disease. Their research is similar to ours since they work with ECG (electrocardiogram) signals which are reduced into extracted features with signal analysis. While their reduced ECG dataset contains sufficient data per patient, a simple clinical feature dataset set has a larger number of patients but only a few features per patient. The authors conclude that the set with a higher number of features lead to better classification performance as compared to a greater number of records with little features. From the ML models of Hagan et al. classification trees performed most accurately [22]. They used four different variations of the classification tree, such as Bootstrap Aggregation, RFs, Gradient Boosting.

Initially, NNs did not get the same level of attention they currently have in neural classification applications. However, with the growing availability of large EEG datasets, DL frameworks are now being utilized for classifying EEG signals. For the EEG signal data which can be considered as time series or image data, deep learning methods like Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) might be more suitable. CNNs and RNNs are variations of the neural network. A CNN combines convolution and pooling layers as hidden layers. Convolution layers identify small patterns in the image and pooling layers combine these patterns to form high-level features [12]. The probability of an output is influenced by the existence or non-existence of these higher-level features. RNN have feedback connections between the layers, which allow them to retain information from past inputs. This allows them to work well for language processing and time series prediction. Recently, advancements are made in CNN and RNN for medical applications. The key advantage lies in their ability, unlike other techniques, to automatically identify important features for prediction networks without requiring human intervention [15]. Deep learning methods such as CNN and RNN can use images, extracted features or signal values as input. Craik et al. [6] add that in comparison to other DL algorithms, CNNs can effectively manage the handling of a high number of EEG signals. Zhai et al. [23] used a version of CNN to classify ECG signals, which record the electrical activity of the heart and describe this as a common machine learning problem for time series signal classification with imbalanced classes. To solve a time series signal problem, various techniques can be used such as feature extraction or feature learning methods, including Convolutional Neural Networks (CNNs). For this research, feature extraction

techniques were applied to the EEG signals, but it is also possible to use a CNN on raw EEG signal data.

DL techniques offer a major advantage over conventional ML by processing data non-linearly, whereas most ML methods processes data linearly. Craik et al. [6] proved that using signal values as input, rather than images or calculated features, resulted in higher accuracy for CNNs. The authors suggest that this challenges the common belief that more pre-processing efforts lead to better classification results and highlight the potential benefits of direct signal input in CNN studies. Requiring less pre-processing of data is a major advantage of DL. However, due to the complex architecture and a large number of tuning parameters of DL methods, they are typically viewed as black box models, meaning they cannot provide an explanation for their outcomes.

Loh et al. identified three commonly used XAI methods, namely SHAP, LIME, and GradCAM (Gradient-weighted Class Activation Mapping) in their research [9]. The methods were applied to conventional ML and DL methods, where SHAP was mostly used on conventional ML methods and GradCAM on DL methods. GradCAM and SHAP are both XAI algorithms that support in understanding feature importance. Fauvel et al. find that using CNNs with the XAI method GradCAM not only have the potential to achieve high performance while minimizing complexity but also enable reliable explanations [24]. Loh et al. [9] performed a systematic review that focused on the use of ML and DL in the healthcare sector. They also summarized the performance of ML and DL models regarding accuracy, specificity, sensitivity, and AUROC. Their results showed that DL models performed better than conventional ML models in all four metrics, suggesting that DL models are better at accurately classifying data than the conventional ML models.

2.3.1. Conclusion

Each ML method has its advantages and disadvantages. However, by prioritizing interpretability, accuracy, and non-linear input data with high dimensionality, this thesis further analyzes and develops an RF classifier using the extracted features for the anti-NMDARE problem. The decision tree in Figure 2.4 illustrates our choice. The additional XAI algorithms that can be deployed are SHAP and GradCAM for a RF classifier and a CNN resp.. As a suggestion for further research, we also present the CNN classifier in combination with GradCAM for the data with raw EEG signals in Figure 2.4.

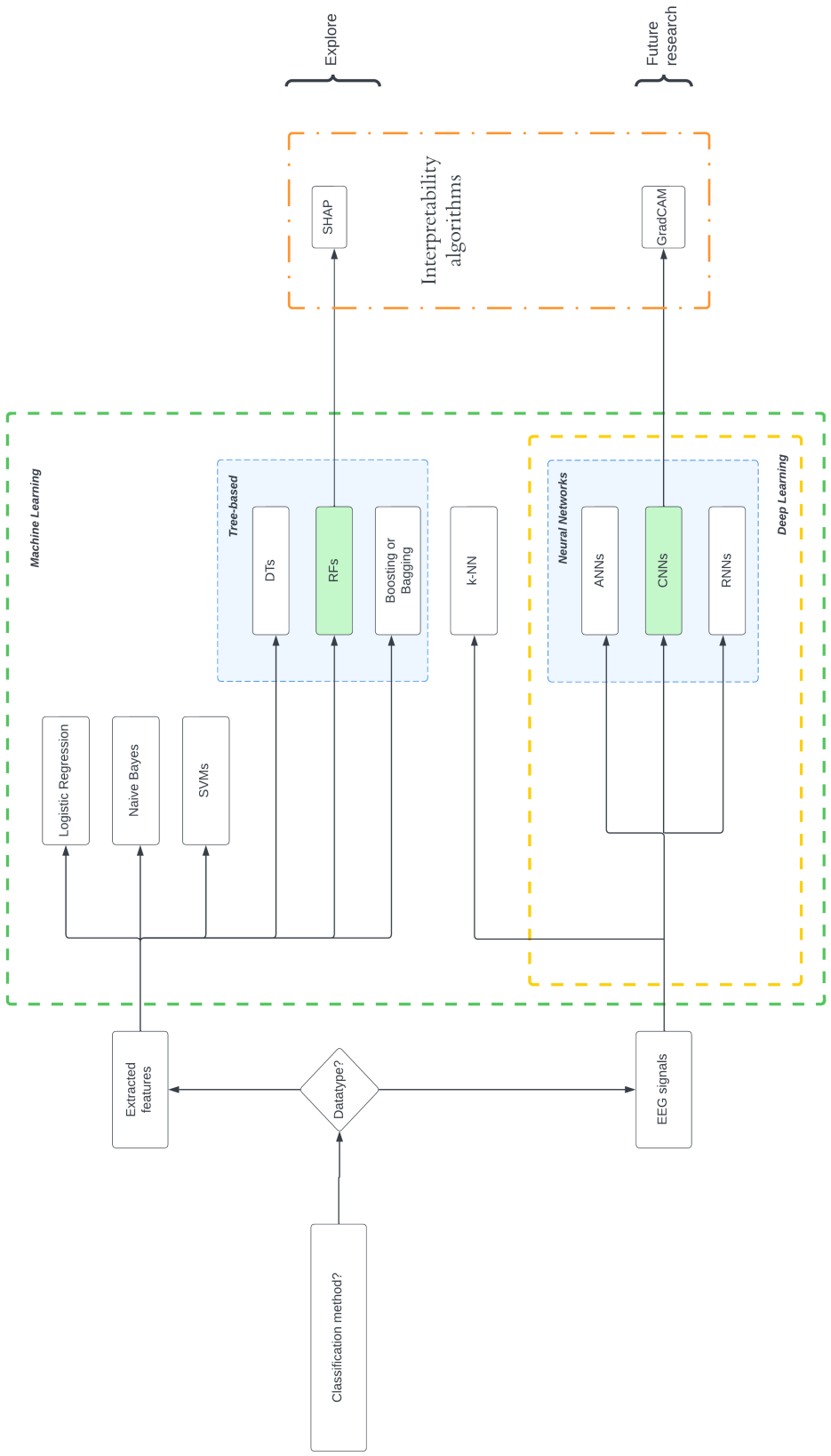


Figure 2.4: Decision flow: Classification methods overview

3 Data

This chapter provides a description of the EEG data available for this study. Some exploratory data analysis is conducted in this chapter to understand the features and their relationship to each other. Furthermore, we analyse the repeated measurements structure of the EEG data. Appropriate solutions are presented for this particular data format.

3.1. EEG dataset

EEG signals capture the electrical activity of the brain by attaching electrodes to the head. Due to its ease of performance and their capability to provide fast results, EEGs are considered a valuable tool for identifying anti-NMDARE [2]. An EEG recording from our dataset used from 16 to 25 electrodes to capture the signal.

Figure 3.1: Example of signals captured by attaching electrodes on the head [25]

Certain characteristics of an EEG support the diagnosis, description, and differentiation of anti-NMDARE. Freund et al. [2] have tried to define EEG characteristics that are specifically important in guiding the anti-NMDARE disease. The authors concluded that generalized slowing was the most common characteristic of the disease. When discussing a classification model, we refer to features rather than characteristics. In the data of anti-NMDARE, we define 29 features extracted from EEG signals. There are three different domains in which the features fall: frequency, complexity, and connectivity domains. The 29 features are assumed to accurately represent the different domains. Table B.1 in Appendix B presents an overview of the extracted features. It is our goal to determine which features improve the classification model.

For each patient, the EEG signals are cut into multiple fragments or epochs. Each epoch lasts 4 seconds, where consecutive epochs have an overlap of 3 seconds. We consider the multiple epochs

as multiple measurements per patient. The 29 useful features are computed per EEG epoch. For example, feature `deltapower` for patient i is computed for epoch 1, epoch 2, ..., epoch n_i, resulting in n_i values for `deltapower` for patient i.

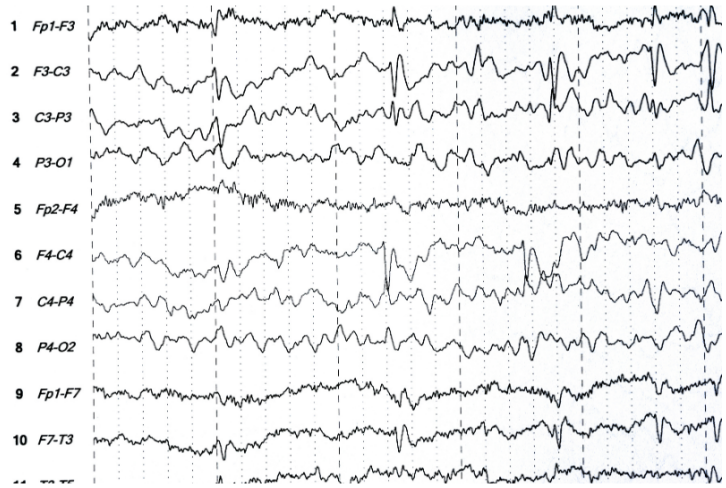


Figure 3.2: Example of an EEG recording [26]

The length of EEG recordings differs per patient. Furthermore, artifacts and redundant EEG epochs are removed from the dataset by the doctors. Therefore, each patient has a different number of useful EEG epoch, ranging from 42 to 3297 with a mean of 1064. The general structure of this type of EEG dataset is displayed in the table below, Table 3.1. The number of patients is denoted by I and the number of features by p.

Patient	Epoch	Label	Features		
1	1	y_1	$x_{1,1,1}$	$\dots$	$x_{1,1,p}$
1	2	y_1	$x_{1,2,1}$	$\dots$	$x_{1,2,p}$
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
1	n_1	y_1	$x_{1,n_1,1}$	$\dots$	$x_{1,n_1,p}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
I	1	y_I	$x_{I,1,1}$	$\dots$	$x_{I,1,p}$
I	2	y_I	$x_{I,2,1}$	$\dots$	$x_{I,2,p}$
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
I	n_I	y_I	$x_{I,n_I,1}$	$\dots$	$x_{I,n_I,p}$

Table 3.1: Structure of the EEG data

In the case of EEG data, the class label of a patient is determined based on the patient's condition after 12 months of the diagnosis of anti-NMDARE. The class label is therefore constant over the patient's EEG epochs. One EEG epoch does not tell us if the patient has a positive or negative prognosis for treatment after 12 months. That is why in Table 3.1 all epochs from the same patient i have label y_i. Mathematically, the data for each patient i is summarized as follows

$$\begin{aligned} &X_i = [\underline{x}_{i,1}, \dots, \underline{x}_{i,p}] \\ &\quad \text{with each } \underline{x}_{i,j} \in \mathbb{R}^{n_i} \\ &\quad \text{where } n_i \text{ is the number of EEG epochs of patient } i \end{aligned} \tag{3.1}$$

Each X_i represents a cluster for patient i with its measurements for features $1, \dots, p$. Combining the

data of all patients leads to the following matrix in $\mathbb{R}^{N\times p}$ with $N = \sum_{i=1}^{I} n_i$

$$X = \begin{bmatrix} \underline{x}_{1,1} & \underline{x}_{1,2} & \cdots & \underline{x}_{1,p} \\ \underline{x}_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \underline{x}_{I,1} & \cdots & \cdots & \underline{x}_{I,p} \end{bmatrix} \tag{3.2}$$

If we sequentially enter all measurements of each patient without differentiating between patients, we write

$$Z = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,p} \\ z_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ z_{N,1} & \cdots & \cdots & z_{N,p} \end{bmatrix} \tag{3.3}$$

This is simply matrix X, from Equation (3.2), where for $j \in [1, \ldots, p]$

$$\begin{aligned} \underline{x}_{1,j} &= [z_{1,j}, \ldots, z_{n_1,j}]^T, \\ \underline{x}_{2,j} &= [z_{n_1+1,j}, \ldots, z_{n_1+n_2,j}]^T \\ &\vdots \\ \underline{x}_{I,j} &= [z_{N-n_I,j}, \ldots, z_{N,j}]^T \end{aligned}$$

An overview to establish the parameters for our case is displayed below.

Number of patients	(I)	44
Total number of EEG epochs	(N)	46.832
Number of extracted features	(p)	29

Table 3.2: Parameters for anti-NMDARE dataset

Since in our case $N = 46.832$ and $p = 29$, this leads to an input matrix Z of size 46.832×29. Using feature reduction methods, this matrix can be reduced to smaller dimensions. The reduced feature space ideally improves the quality of a classifier. Feature reduction techniques are further discussed in Chapter 5.

3.2. Exploratory data analysis

To understand the EEG data and its features, some exploratory data analysis is conducted. This analysis includes the summary statistics such as mean, median, and standard deviation, correlation analysis, and analysis of the repeated measurements structure. These are supported by data visualizations, a useful tool for exploring data. Prior to analyzing the observations, it has been verified that there are no missing values in the EEG dataset provided for this research. Furthermore, all features are continuous variables.

3.2.1. Summary statistics

In this section, we delve into few summary statistics, which are derived from our EEG dataset comprising samples from 44 patients. It is important to acknowledge that we do not know the true population's summary statistics. Instead, we use estimated values obtained from our sampled dataset, denoted as $\hat{\mu}$, and compare them to the true value μ. It is crucial to avoid confusing this with $\hat{y}$, which represents the predicted class label.

The mean of feature j is its average value across all epochs, Equation (3.4).

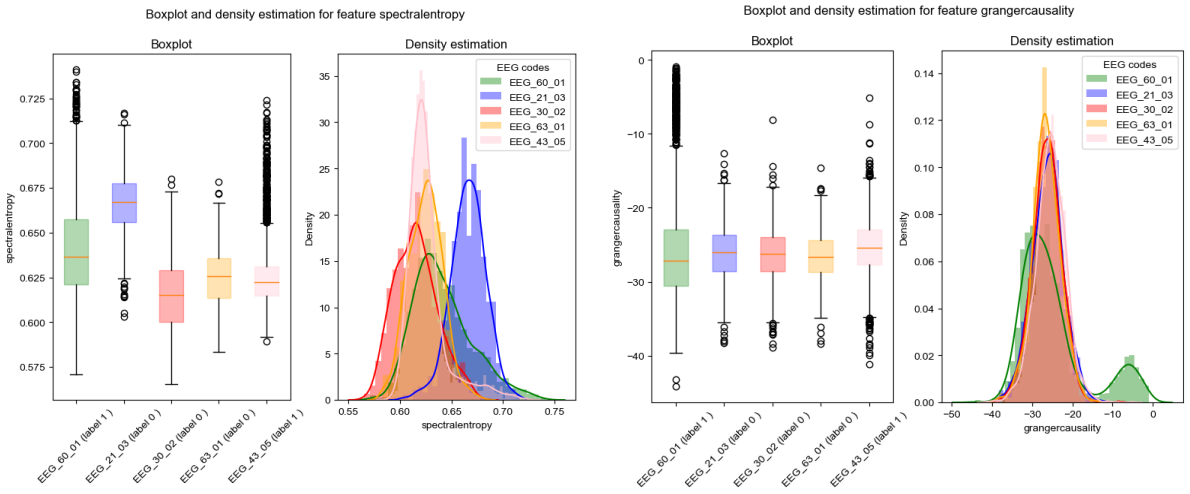
$$\hat{\mu_j} = \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \tag{3.4}$$

The median is the middle value of the data when it is ordered from smallest to largest. When the mean and median differ greatly, it might indicate a skew distribution with some extreme outliers. The standard deviation tells how much spread out a feature j is and is denoted by

$$\hat{\sigma_j} = \sqrt{\frac{\sum_{i=1}^{N}(x_{i,j} - \hat{\mu_j})^2}{N}} \tag{3.5}$$

It provides a measure on how much variability there is around the mean of feature j. The variance is simply obtained by taking the square of the standard deviation. The mean and standard deviation (std) plus the minimum and maximum of all EEG features is displayed in Table B.2 in Appendix B.

Boxplot A boxplot captures most of the summary statistics. The box represents the second (25th percentile) and third (75% percentile) quartile, note the end values as $Q1$ and $Q3$. The interquartile range (IQR) is the interval between the end values, $Q1$ and $Q3$. The IQR captures 50% of the data. The line in the middle of the box is the median value (notation $Q2$). Two lines extend the box no further than $\pm 1.5 \cdot (Q3 - Q1)$. The outliers of the dataset are denoted by the dots. Figure 3.3 shows the boxplot of two features with their estimated density.



(a)) spectralentropy boxplot and density estimation

(b)) grangercausality boxplot and density estimation

Figure 3.3: Boxplot and density estimation of spectralentropy and grangercausality for random 5 patients

Skewness Some patients exhibit a higher number of outliers compared to others. A lot of outliers might indicate a skewed distribution. The skewness coefficient from `stats.skew` quantifies the asymmetry of a sample and is computed as the Fisher-Pearson coefficient of skewness [27].

$$FP = \frac{m_3}{m_2^{\frac{3}{2}}}$$

where m_k is the sample k'th central moment defined by

$$m_k = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \hat{mu_j})^k$$

A positive value for the coefficient indicates more weight in the right tail, i.e. most values lay on the right side of the graph. A negative value for the skewness coefficient means more values for the feature lay on the left side. Figure 3.4 tells us that not all features of the data are normally distributed since some features have a high skewness coefficient. We will see that various statistical tests assume normality.

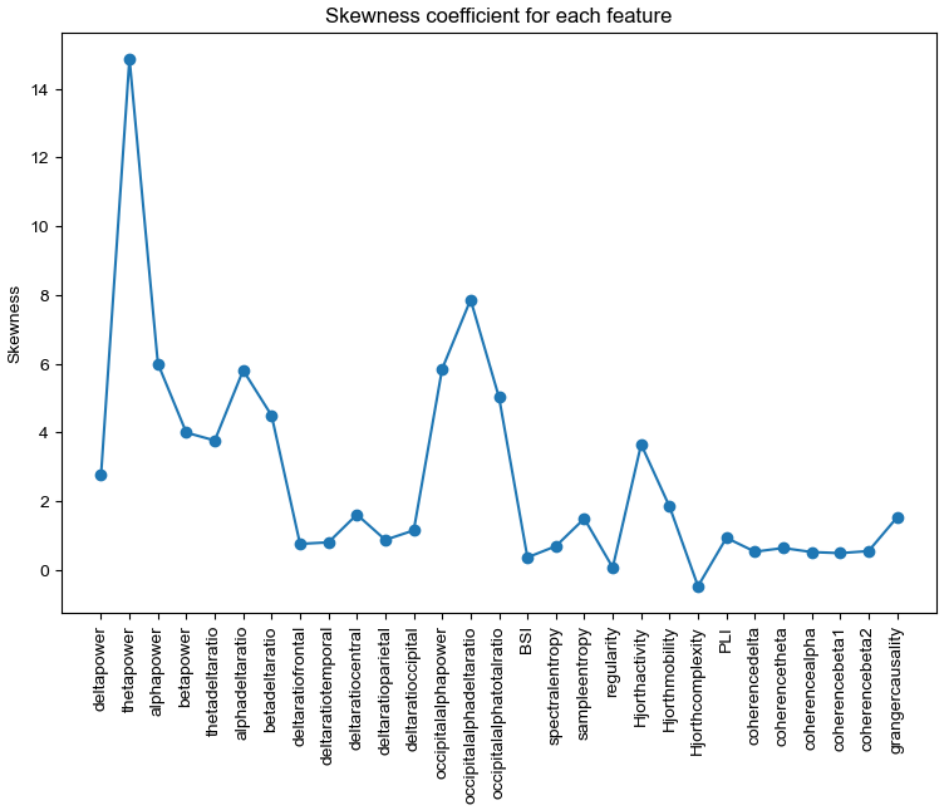


Figure 3.4: Skewness coefficient

When plotting the three features with the lowest and highest skewness coefficient, the coefficients appear to be correct as the features in the right figure are right skewed, see Figure 3.5.

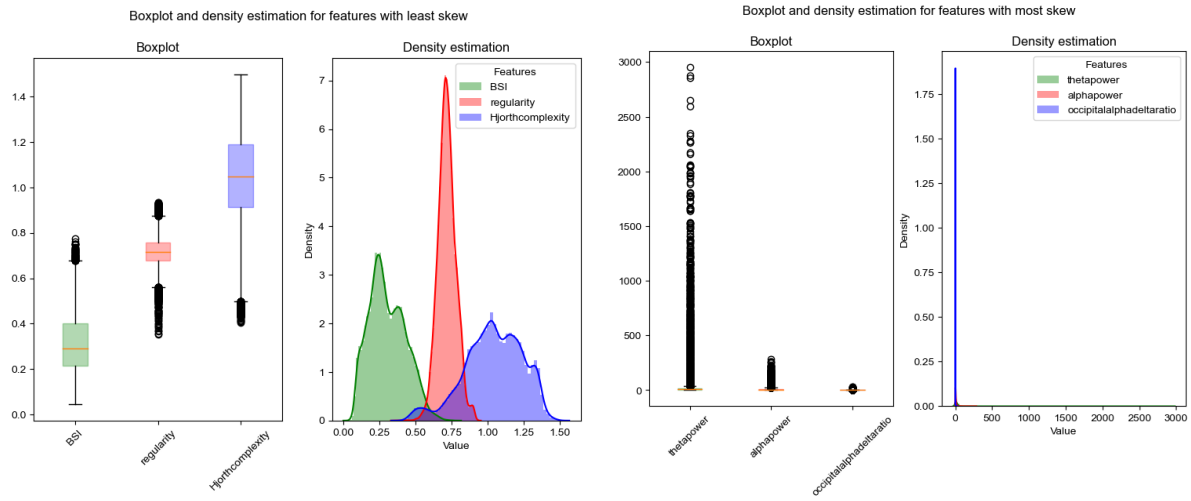


Figure 3.5: Boxplot and density estimation of least and most skewed features

Class balance Because we have two classes for classification, the class balance in the data is important to examine. Training on too little data from one class and too many of another might lead to overfitting. The classifier is then more trained on one type of class. Furthermore, the model's accuracy score may not accurately reflect the performance. The accuracy score has a baseline, which happens when all observations are classified to the most prominent class. In the case of anti-NMDARE, there are 35 patients with class label 0 and 9 with label 1, so 79.5% versus 20.5%. When we look at all EEG

epochs, there is a class imbalance of 73% (class 0) versus 27% (class 1). This needs to be taken into account for training the classifier. If we would create a classification model that classifies all patients to class 0, the model would reach an accuracy score of 79.5%.

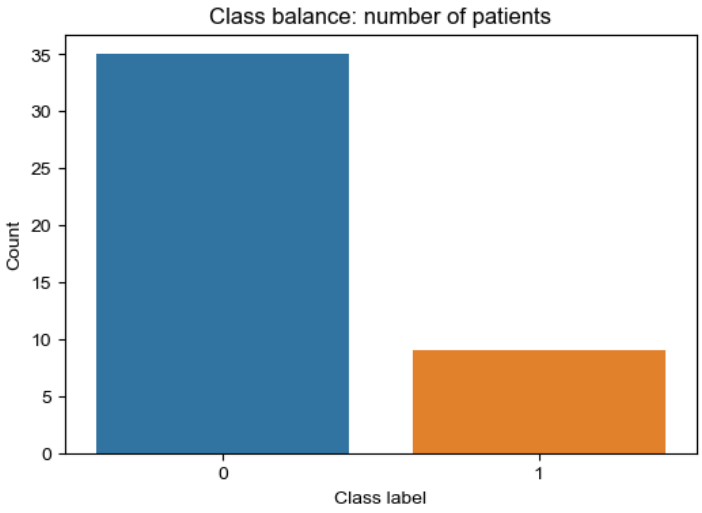


Figure 3.6: Balance of the two classes

3.2.2. Correlation analysis

Correlation helps determine the strength of the relationship between two variables. Pearson's correlation coefficient between two continuous features j and k is defined by

$$r_{j,k} = \frac{\sum_{i=1}^{N}(x_{i,j} - \hat{\mu_j})(x_{i,k} - \hat{\mu_k})}{\sqrt{\sum_{i=1}^{N}(x_{i,j} - \hat{\mu_j})^2}\sqrt{\sum_{i=1}^{N}(x_{i,k} - \hat{\mu_k})^2}} \tag{3.6}$$

The correlation coefficients ranges between -1 and 1, where both the extremes indicate a linear relationship between features. In the figure below, Figure 3.7, the correlation coefficients between all variables are color coded. The figure illustrates several strong relationships between features.

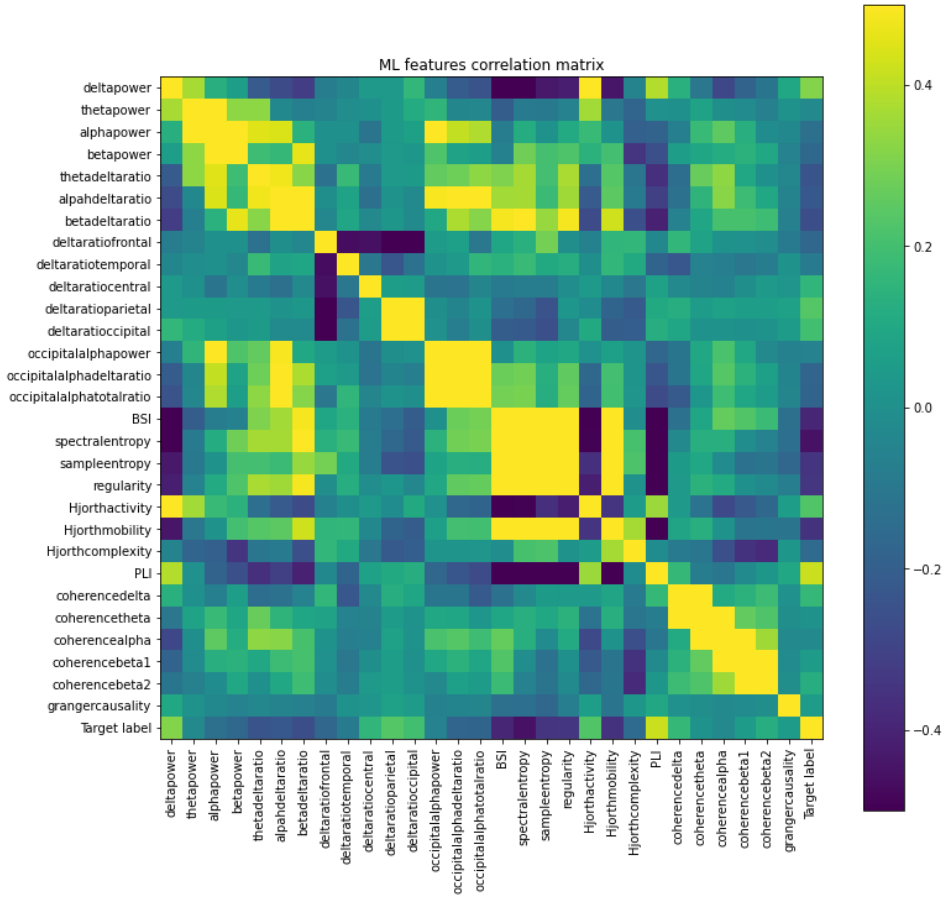


Figure 3.7: Correlation matrix of EEG features

The strongest relationship is between features `spectralentropy` and `Hjorthmobility` with a coefficient of 0.933.

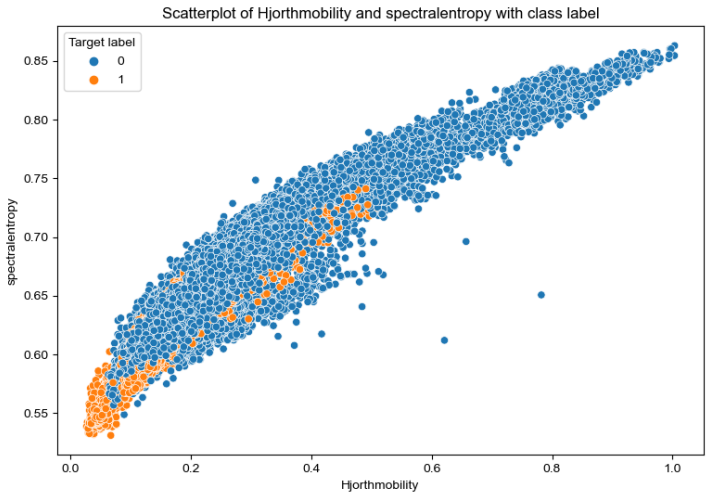


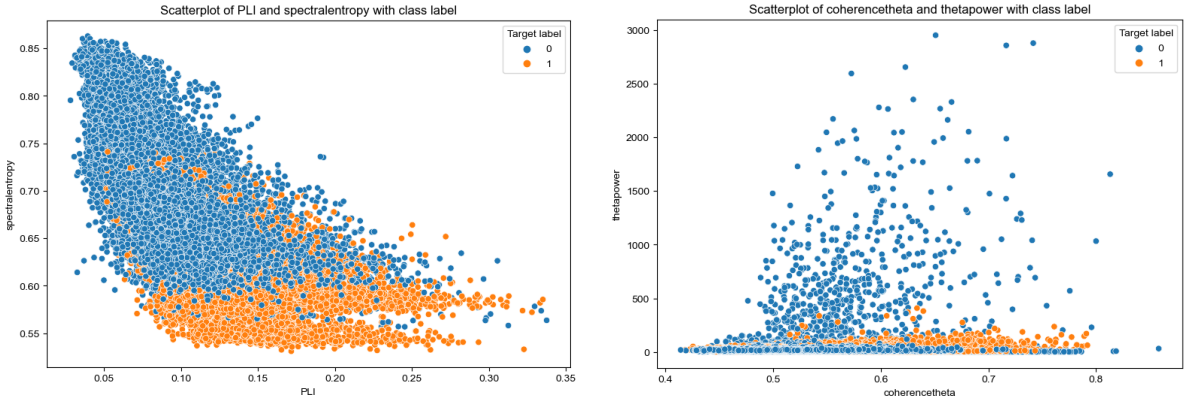
Figure 3.8: Scatterplot of two most correlated features

The four features that are most correlated with the target label are `spectralentropy`, `PLI`, `BSI`, and `Hjorthmobility` with correlation coefficients -0.45303, 0.4203, -0.3919 and -0.355. The correlation between categorical and continuous variables cannot be determined by the coefficient given in Equation (3.6). The Point-Biserial correlation coefficient quantifies the association between continuous feature

j and the binary label $\{0, 1\}$. The coefficient is given by Equation (3.7) and takes on a value between -1 and 1 [28].

$$r_{pb} = \frac{\hat{\mu}_j^{(0)} - \hat{\mu}_j^{(1)}}{\hat{\sigma}_j} \sqrt{\frac{N_0 N_1}{N(N-1)}} \tag{3.7}$$

where $\hat{\mu}_j^{(0)}$ and $\hat{\mu}_j^{(0)}$ are the sample means for class 0 and 1 resp.. The standard deviation over all observed feature values is $\hat{\sigma}_j$. It is expected that the variables with high correlation coefficient will play an important role in training the classifier as they provide valuable insights on the target label.



(a)) PLI against spectralentropy with class label

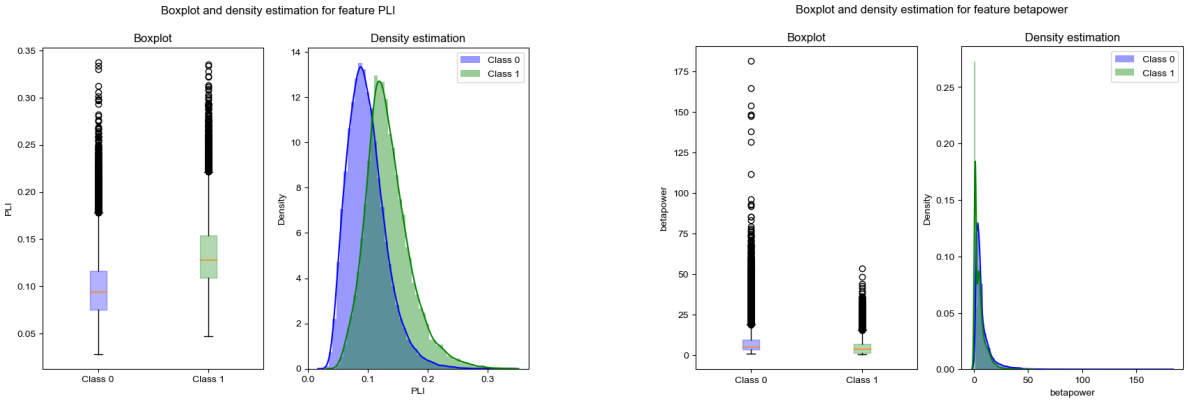
(b)) coherencetheta against thetapower with class labels

Figure 3.9: Two scatterplots with class labels: one representing highly correlated features, and one low correlated features

The four features that are least correlated with the target label are `coherencetheta`, `thetapower`, `coherencealpha`, and `grangercausality` with correlation coefficients 0.0096, -0.0286, -0.0290 and 0.0485.

3.2.3. Statistical tests

Apart from the correlation analysis, we test if there is a significant difference in feature values between the classes with label 1 and with label 0. Visually, we notice in Figure 3.10 that feature `PLI` might be significantly different for class means, as their means fall outside each other's IQR. For `betapower` it is more difficult to draw conclusions.



(a)) PLI boxplot and density estimation class 0 vs class 1 observations

(b)) betapower boxplot and density estimation class 0 vs class 1 observations

Figure 3.10: Boxplot and density estimation class 0 vs class 1 observations

Statistical tests examine the possible significant difference between the two classes. To determine which test to use, we formulate the characteristics of our dataset. Our EEG data consists of continuous features which are independent between patients and therefore between the classes with label 0 and 1.

So the feature values of `deltapower` of patient 1 are independent of the feature values of `deltapower` of patient 2. The features are not independent within-patients since the measurements are from consecutive EEG epochs. The feature value of `deltapower` in epoch 20 of patient 1 is not independent from epoch 1 within the same patient. The statistical tests such as an independent t-test or the Mann-Whitney U-test are not applicable due to the data dependency. In the next section, we examine the cluster-based structure of our EEG dataset to deal with the non-independence within patients.

3.3. Clustered data

Our dataset, defined in Equation (3.1), is cluster-based data, since we have multiple observed feature values per patient. The EEG epochs can be viewed as measurements over time per patient. The measurements from one patient are correlated. When there are repeated measurements in the data, the patients are considered as clusters [29]. We recognize that the interpretation of cluster-based data can vary in the context of ML. In this analysis, we treat the patients as clusters and examine these pre-defined clusters.

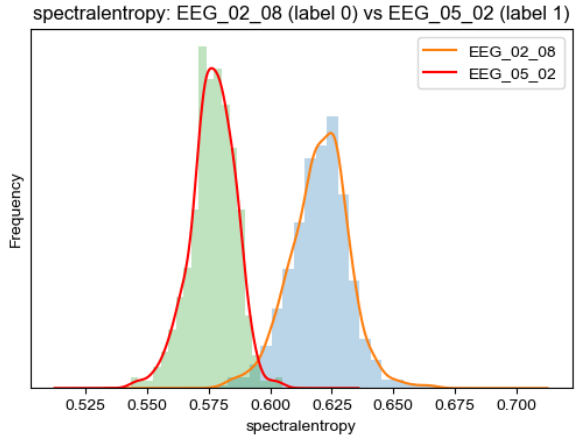
3.3.1. Statistical tests

To check if there is a significant difference in feature values between patients, we use statistical tests which reject or accept a null hypothesis. The hypotheses for this problem are formulated as

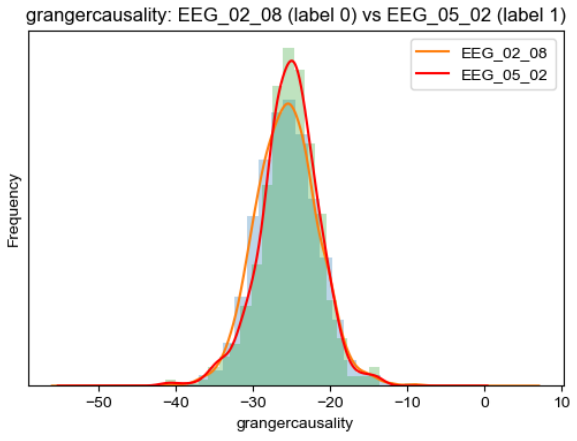
$$\begin{aligned} H_0 &: \text{ there is no difference for feature } j \text{ between the different patients } i \quad i \in \{1, \dots, 44\} \\ H_1 &: \text{ there is a difference for feature } j \text{ between the different patients } i \quad i \in \{1, \dots, 44\} \end{aligned} \tag{3.8}$$

The difference can be formulated as a difference in sample means, variance or distribution.

To first show visually that there might be significant differences between patients, we look at the histograms (with density estimation) of the features `spectralentropy` and `grangercausality` for patients with EEG codes `EEG_02_08` and `EEG_05_02` in Figure 3.11.



(a)) spectralentropy for EEG code 2 and 5



(b)) grangercausality for EEG code 2 and 5

Figure 3.11: Histogram and density estimation of spectralentropy and grangercausality for patients with EEG code 2 and 5

Since the histograms of `grangercausality` overlap entirely, we expect that there is no difference between patients. However, for `spectralentropy`, we expect to have a significant difference in feature values between patients, while their between-variance is large. This means that the null hypothesis for `grangercausality` is expected to be accepted, whereas for `spectralentropy` it would be rejected.
For various features the data is not normally distributed, as seen in Figure 3.4. When there is sufficient data available, the majority of the tests are resilient against the assumption of normality, though we have to be cautious interpreting the p-values. Both the one-way ANOVA test and the Kruskal-Wallis test reject the null hypothesis from Equation (3.8) for all features. The primary concern that undermines the p-value is the requirement for independence of data within groups. The EEG data are measurements from EEG epochs within each patient and so most statistical tests are unreliable.

Furthermore, the patients included in our research can be viewed as a random sample from a larger study population. We are not particularly interested in the difference between variances between patient 2 and 5, but we need to take these difference into account when we have repeated measures. As stated before, the repeated measures per patient are correlated. Therefore, it is crucial to determine whether certain patients have naturally lower values for a specific feature. This is called a random effect. We examine different approaches to the data definition by including the repeated measurements structure.

3.3.2. New definition for clustered data

A single variable for clustered data can be described as follows (this is a variation of [30], [31] and [32]),

$$\begin{aligned} &X_{i,j} = \mu_i + \epsilon_{i,j}, \\ &\quad \text{where } \mu_i \text{ is the target mean for patient } i \\ &\quad \text{which is independent across all patients,} \\ &\quad \text{and } \epsilon_{i,j} \text{ random fluctuations,} \\ &\quad \text{which are independent for all EEG epochs } j = 1, \ldots, n_i, \\ &\quad \text{across all patients } i = 1, \ldots, 44, \\ &\quad \text{and independent of } \mu_i. \end{aligned} \tag{3.9}$$

To give an example of this model, we adjust Table 3.1 for a single feature into Table 3.3

$X_{i,j}$	Mean	Fluctuation $\epsilon_{i,j}$	Model
$X_{1,1}$	μ_1	$\epsilon_{1,1}$	$\mu_1 + \epsilon_{1,1}$
$\vdots$		$\vdots$	$\vdots$
X_{1,n_1}		ϵ_{1,n_1}	$\mu_1 + \epsilon_{1,n_1}$
$X_{2,1}$	μ_2	$\epsilon_{2,1}$	$\mu_2 + \epsilon_{2,1}$
$\vdots$		$\vdots$	$\vdots$
X_{2,n_2}		ϵ_{2,n_2}	$\mu_2 + \epsilon_{2,n_2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$X_{I,1}$	μ_I	$\epsilon_{I,1}$	$\mu_I + \epsilon_{I,1}$
$\vdots$		$\vdots$	$\vdots$
X_{I,n_I}		ϵ_{I,n_I}	$\mu_I + \epsilon_{I,n_I}$

Table 3.3: cluster-based data description for a feature X

Shou et al. [32] denote σ^2_μ as the variance for μ_i across all patients and σ^2_ϵ as the variance for ϵ across all patients and repeated measures. Again, it is important to highlight that we do not know these variances, but we can estimate them using the available data by applying Equations (3.4) and (3.5) for variables μ and ϵ. In the figure below, Figure 3.12, an approximation for the density function is plotted for the mean and fluctuations of `spectralentropy` and `betapower`.

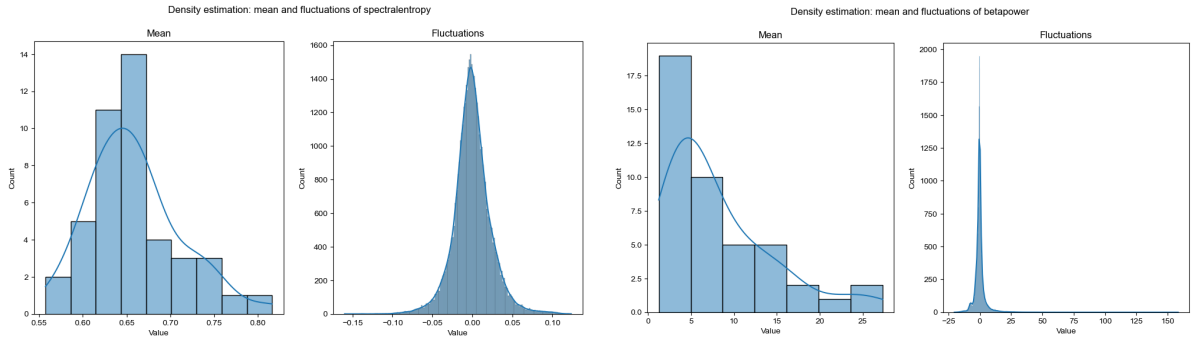


Figure 3.12: Histogram with density estimation of mean μ_i and fluctuations $\epsilon_{i,j}$ for features spectralentropy and betapower

Shou et al. [32] extended this model when applied to a study for image replicates. In our case, the number of voxels can be seen as the number of features (k). Mathematically,

$$
\begin{aligned}
&X_{i,j}(k) = \mu_i(k) + \epsilon_{i,j}(k), \\
&\qquad \text{where } \mu_i(k) \text{ is the target mean of feature } k \text{ for patient } i, \\
&\qquad \text{which is independent across all patients } i = 1, \ldots, 44, \\
&\qquad \text{and } \epsilon_{i,j}(k) \text{ random fluctuations for feature } k, \\
&\qquad \text{which are independent for all EEG epochs } j = 1, \ldots, n_i, \\
&\qquad \text{and across all patients } i = 1, \ldots, 44.
\end{aligned}
\tag{3.10}
$$

In this model, it also holds that $\mu_i(k)$ has variance $\sigma^2_\mu(k)$ and $\epsilon_{i,j}(k)$ has variance $\sigma^2_\epsilon(k)$.

Our goal is to analyze the clustered data for multiple patients using this model. This type of data involves considering both within-cluster and between-cluster variation, meaning within one patient and between patients. The existence of clustering introduces additional complexity that needs to be considered and addressed [33]. The types of variance need to be accounted for for better predictions. Analyzing the within-cluster variance ensures the model captures the correlation between the repeated measures. In repeated measures analysis, outcomes of two observations within the same cluster tend to be more similar than the outcomes of two observations from different clusters [33]. This type of variance violates the condition of independent observations most classification models, including Random Forests, require.

3.3.3. Solutions for clustered data

Galbraith et al., [34], have addressed multiple options to treat clustered data:

1. ignoring clustering
2. reducing clusters to independent observations
3. explicitly accounting for clustering

Ignoring clustering Since the design of our study has clustered data, the clustering cannot be simply ignored without acknowledging the consequences. According to Galbraith et al. [34] the impact of clustering depends crucially on the strength of intracluster correlation. Before we consider the option of ignoring clustering, we explain the intracluster coefficient (ICC). A greater intracluster coefficient indicates a stronger clustering effect. The coefficient depends on the distribution of the data, the number of clusters, and the number of observations per cluster. The ICC represents the similarity between measurements observed for the same patient [33]. If an ICC gets closer to 0, it means there is little to no clustering effect in the data. That is because the difference between μ_i's is negligible compared to the differences in observations. Liljequist et al. [31] presented a simplified theory of the ICC,

$$
\text{Intracluster Correlation Coefficient (ICC)} = \frac{\text{variance between groups}}{\text{total variance}} = \frac{\sigma^2_\mu}{(\sigma^2_\mu + \sigma^2_\epsilon)}, \tag{3.11}
$$

where σ_μ^2 and σ_ϵ^2 are the variances from the variables in Equation (3.9). We estimate the values for σ_μ^2 and σ_ϵ^2 based on our EEG data.

$$\begin{aligned} \hat{\sigma}_\mu^2 &= \frac{\sum_{i=1}^{I}(\mu_i - \hat{\mu})^2}{I} \\ \hat{\sigma}_\epsilon^2 &= \frac{\sum_{i=1}^{I}\sum_{k=1}^{n_i}(\epsilon_{i,k} - \hat{\epsilon})}{N} \end{aligned} \tag{3.12}$$

Here, $\hat{\mu}$ and $\hat{\epsilon}$ are the overall means of μ and ϵ and μ_i and $\epsilon_{i,k}$ the observed values from the data.

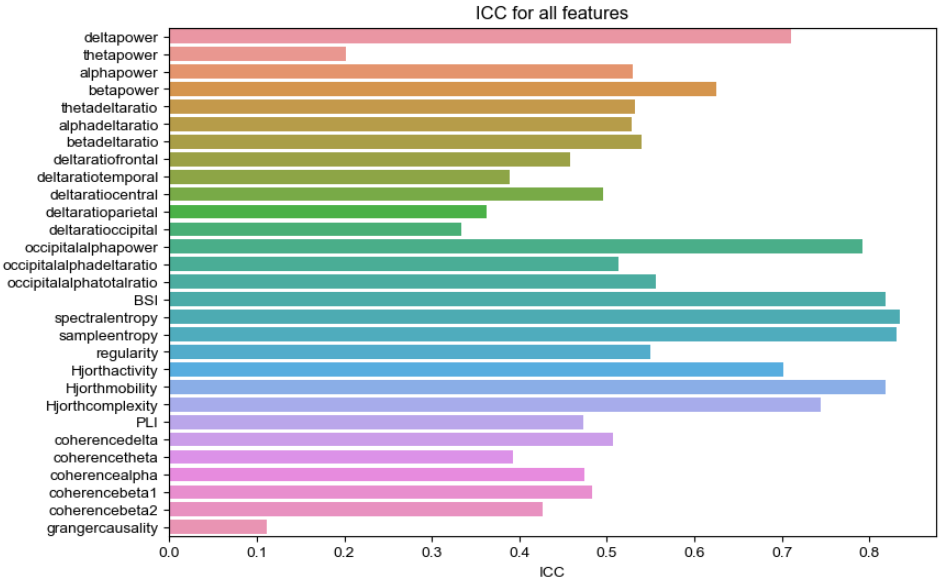
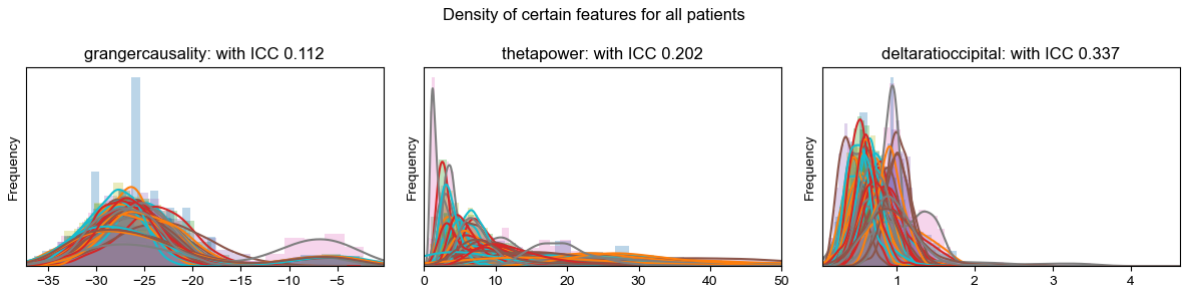
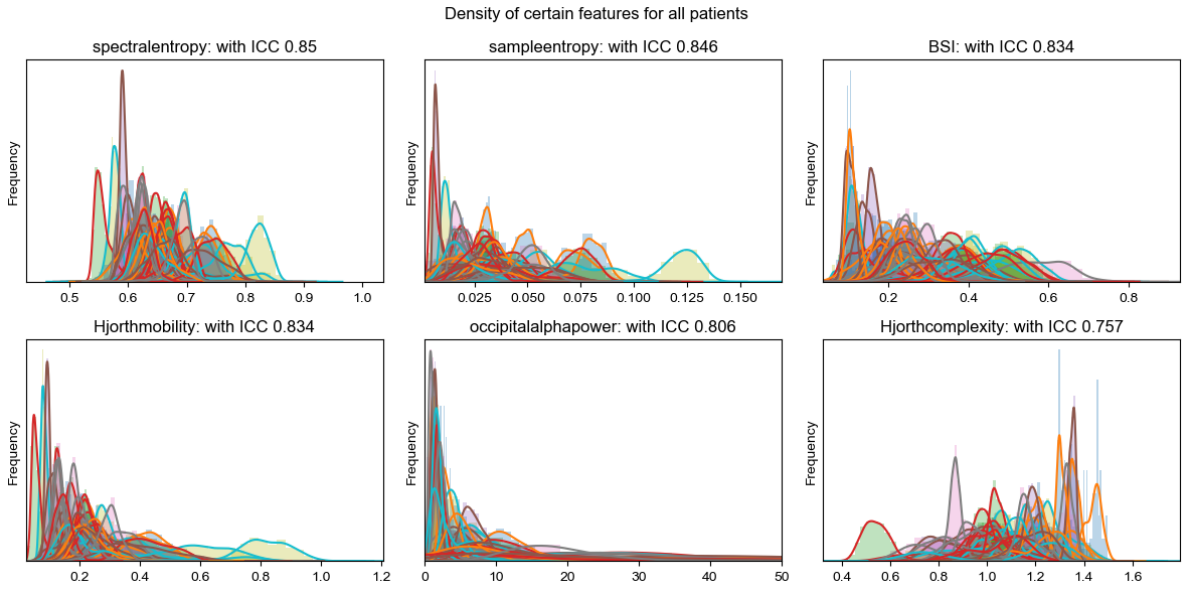


Figure 3.13: ICC per feature

In the case of anti-NMDARE, 17 features obtain an estimated ICC greater than 0.5 indicating a moderate clustering effect. Only 7 features have an estimated ICC greater than 0.7 providing good reliability on the clusters [35]. No feature has an estimated ICC above 0.9. In Figure 3.14, the clustering effect is visualized for various features. The figure shows that the densities of features with a high ICC are more dispersed compared to those with a low ICC. This suggests that there is a difference between the patients.



(a)) Estimated densities per patient of features with low ICC



(b)) Estimated densities per patient of features with high ICC

Figure 3.14: Densities of features with high and low ICC over all patients

Since we have multiple features that are measured and various measurements per patient, the model from Shou et al. [32] is more applicable. They extended the classical ICC to a measure for the high dimensional setting. They present an unbiased estimator for the average ICC for the model in Equation (3.10).

$$\text{average ICC } = 1 - \frac{1}{p}\frac{\sum_{i=1}^{I} n_i - 1}{\sum_{i=1}^{I}(n_i - 1)} \sum_{k=1}^{p} \frac{(x_{i,j}(k) - \hat{\mu}_i(k))^2}{\sum_{i=1}^{I}\sum_{j=1}^{n_i}(x_{i,j}(k) - \hat{\mu}(k))^2} \tag{3.13}$$

Furthermore, they introduce an image intraclass correlation coefficient (I2C2) which uses the multivariate setting from Equation (3.10). We leave the exploration of the multivariate setting for future research.

By ignoring clustering, the resulting data aligns with the definition stated in Equation (3.3). From this point forward, we refer to this particular dataset as EEG set Z.

Reducing clusters The alternative is to reduce the clusters such that observations become independent. Summary measures such as mean, max, min, and variance compile all EEG epochs from a high dimension into one dimension. Mathematically formulated, for every patient i, their X_i from Equation

(3.1) is summarized into four statistics for each feature k,

$$\begin{aligned} x_{i,\mu_k} &= \frac{\sum_{j=1}^{n_i} \text{feature k in epoch } j \text{ for patient } i}{n_i} \\ x_{i,\sigma_k^2} &= \frac{\sum_{j=1}^{n_i} (\text{feature k in epoch } j \text{ for patient } i - x_{i,\mu_k})^2}{n_i - 1} \\ x_{i,\text{min}_k} &= \min_{j=1,\ldots,n_i} x_{j,k} \\ x_{i,\text{max}_k} &= \max_{j=1,\ldots,n_i} x_{j,k} \end{aligned} \tag{3.14}$$

As a more robust alternative for the minimum and maximum, $x_{i,Q3+1.5\cdot IQR}$ and $x_{i,Q1-1.5\cdot IQR}$ can be used. These values are less sensitive to outliers.

This results in a reduced feature matrix $F_X \in \mathbb{R}^{44\times 116}$ with all observations (rows) independent. Hereafter, we call the reduced dataset F_X, defined in Equation (3.15).

$$F_X = \begin{bmatrix} x_{1,\mu_1} & \cdots & x_{1,\mu_{29}} & x_{1,\sigma_1^2} & \cdots & x_{1,\text{min}_1} & \cdots & x_{1,\text{max}_1} & \cdots & x_{1,\text{max}_{29}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{44,\mu_1} & \cdots & x_{44,\mu_{29}} & x_{44,\sigma_1^2} & \cdots & x_{44,\text{min}_1} & \cdots & x_{44,\text{max}_1} & \cdots & x_{44,\text{max}_{29}} \end{bmatrix} \tag{3.15}$$

Since our patients receive only one label, averaging does not introduce any issues like Hu et al. suggest. When the subjects have different category labels over their measurements, the averaging approach would be impossible, [30]. Luckily, we can summarize our EEG data from Z into the matrix F_X.

Explicitly accounting for clustering This option entails direct integration of the clustering effect into the classification model. The data does not have to be reformulated but its clustering structure will still be taken into account when building the classification model. This last option includes methods such as Generalized Linear Mixed Models or Generalized Estimating Equations [34]. GLMMs are an extension of Linear Mixed Models and can be used for continuous non-normal, binary, and categorical responses. LMMs include both fixed and random effects that account for the variation in the data. Mixed effect models are generally used in settings when there are within-patient and between-patient variations. The fixed effects represent the population-level relationships between the features and the outcome variable, whereas the random effects express the variation that is not explained by the fixed effects. The random effects justify the effects of unmeasured or unknown factors that vary between the different patients. The goal is to estimate the degree of variation within the different patients and incorporate this into the random effects. GLMM can be challenging for complex datasets since the relationship between predictors and outcome needs to be determined beforehand. The general form of a GLMM is

$$\begin{aligned} &y = X\beta + Z\gamma + \epsilon \\ &\text{with} \\ &y \in \mathbb{R}^N && \text{where } N \text{ is the number of EEG epochs} \\ &X \in \mathbb{R}^{N\times p} && \text{where } p \text{ is the number of features} \\ &\beta \in \mathbb{R}^{p\times 1} && \text{the fixed effects parameters} \\ &Z \in \mathbb{R}^{N\times q\cdot I} && \text{where } q \text{ is the number of random effects} \\ &\gamma \in \mathbb{R}^{q\cdot I\times 1} && \text{the random effects parameters} \\ &\epsilon \in \mathbb{R}^N \end{aligned} \tag{3.16}$$

The matrix of random effects Z is sparse, meaning it consists of mostly zeros. The columns represents the random effect for each patient and each row is an epoch. If an epoch j belongs to the patient i, $z_{j,i} = 1$. The use of random slopes allows for fluctuations in the fixed effects based on the patient. Including a random intercept allows for variations in the outcome for each patient. If a random intercept and random slope is included in the model, Z has size $N \times 2 \cdot I$. In our case, it is possible to consider a random effect for `spectralentropy`. A LMM would have the form of

$$y_i = X_i\beta + Z_i \cdot \gamma_{i,\texttt{spectralentropy}} + \epsilon_j$$

This leads to an extra parameter to estimate. The slope for `spectralentropy` varies per patient $\beta_{\texttt{spectralentropy}} + \gamma_{i,\texttt{spectralentropy}}$.

In case of a binary outcome, we need a link function $g(\cdot)$ that relates a binary outcome y to the linear model $X\beta + Z\gamma$. That is,

$$\begin{aligned} \mathbb{E}(Y = 1|\cdot) &= g^{-1}(X\beta + Z\gamma) \\ P(Y = 1|\cdot) &= g^{-1}(X\beta + Z\gamma) + \epsilon \end{aligned} \tag{3.17}$$

The link function that is most often used for a binary outcome is the logit function: $g(p) = \log(\frac{p}{1-p})$. Here, $\frac{p}{1-p}$ are the odds, which essentially is the ratio of an event happening to an event not happening. The log odds are useful in problems where a probability of two situations, p versus $1-p$, needs to be discovered. A probability p is the ratio of an event happening to everything else that can happen. The inverse of the logit function is $p = g^{-1}(\cdot) = \frac{\exp(\cdot)}{1+\exp(\cdot)}$. Leading to,

$$\begin{aligned} \text{logit}(p) &= X\beta + Z\gamma \\ p &= \frac{\exp(X\beta + Z\gamma)}{1 + \exp(X\beta + Z\gamma)} + \epsilon \end{aligned} \tag{3.18}$$

as a GLMM for binary outcomes.

Training a GLMM involves estimating the fixed and random effects. The GLMM is fitted to the training data using an optimization algorithm. These algorithms include an iterative method of maximum likelihood estimation or the Expectation-Maximization (EM) algorithm and are used to estimate the parameters in the GLMM. During this training stage, the estimation of the fixed effect parameters takes into account the random effects. In this sense, the random effects influence the estimation of the fixed effect parameters. When it comes to classifying test observations, the fixed effect parameters obtained from the training stage are used. The random effects for the test observations are not available, as they are specific to the training data.

A drawback of this method is that each repeated measure, or EEG epoch, requires its own label. In our dataset, the class label is determined based on all EEG epochs. For this thesis, we decide to primarily focus on the first two options: ignoring clustering (defined by Z from Equation (3.3) and reducing the clusters to independent observations (defined by F_X from Equation (3.15)).

3.4. Analysis of F_X data

Our exploratory data analysis at the beginning of this chapter captured information regarding the EEG set Z. We briefly analyze the correlation and statistical significance of the F_X set.

3.4.1. Correlation analysis

The features from F_X that are most correlated with the class label are `sampleentropy std`, `deltapower std`, `BSI mean`, `deltapower mean`, `regularity mean` and `coherencealpha min`. Their correlation coefficient with the class label range from 0.32 to 0.39. The five least correlated features are `coherencebeta1 std`, `coherencebeta2 max`, `deltaratioccipital max`, `deltaratiocentral std` and `deltaratiotemporal min`, with coefficients between 0.005 and 0.011.
It is noteworthy that when computing the correlations for dataset Z, `coherencealpha` showed the least correlation with the class label. However, in the correlation matrix of F_X, the minimum of `coherencealpha` becomes one of the top features correlated with the class label. Additionally, `BSI`, `PLI` and `spectralentropy` keep their high correlation with the target label.

This can medically be due to the fact that the presence of a specific coherence alpha value is more important than the values itself.

3.4.2. Statistical tests

By reducing the clusters to independent observations, we can test the relationship with the class label through an independent t-test. A two-sample independent t-test compares the means of two samples,

in our case the observations of patients with class label 0 and the ones with class label 1. This test uses a test statistic T that quantifies the strength of evidence against the null hypothesis that the two classes have equal means. The t-statistic is defined by

$$T = \frac{\hat{\mu}_1 - \hat{\mu}_0}{s\sqrt{\frac{1}{N_1} + \frac{1}{N_0}}} \tag{3.19}$$

where s is an estimator of the pooled standard deviation of the two samples [12]. The two terms, $\hat{\mu_1}$ and $\hat{\mu_0}$ are the two sample means of group 0 and 1. Also, N_1 and N_0 are the number of observations per group. The mathematical definition of s is

$$s = \sqrt{\frac{(N_1 - 1)\sigma_1^2 + (N_0 - 1)\sigma_0^2}{N_1 + N_0 - 2}}$$

Again, σ_1^2 and σ_0^2 are the estimators of the variances per group.
If the absolute value of T is large enough, there is evidence against $H_0 : \mu_0 = \mu_1$. To determine what value is large enough, we test the value of T against a $\mathcal{N}(0, 1)$ distribution. If the probability, known as the p-value, of observing a value greater than the calculated test statistic T is less than the significance level α, we reject the null hypothesis. Based on our data, the function `ttest_ind` can tell us to reject or accept the null hypothesis.

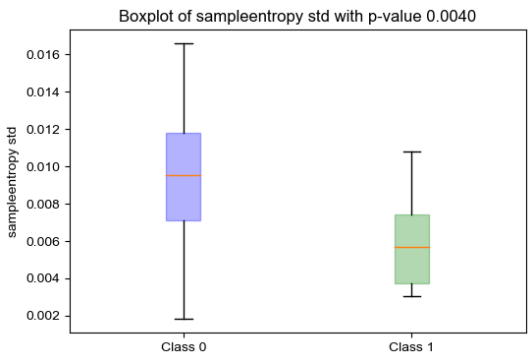
Before we test is there is a significantly difference between the two classes, we check the assumptions that need to hold. Namely, the data needs to be normally distributed and independent, and homogeneity of variances needs to hold. Homogeneity of variances means that the variances of each patient being compared are equal. In Figure 3.4, we saw that not all features are normally distributed. However, the test is robust to normality violations when there is sufficient data available. If the data is very non-normal, a non-parametric test might be more suitable.

Testing normality with `stats.normaltest` shows us that for class 0, 75 0f 116 features are not normal and for class 1 32 0f 116 features, where the latter might be affected by the limited amount of data. Only 7 features do not conform to the assumption of equal variances.
In Table 3.4, we present the test statistic T and its corresponding p-value for each feature for which we reject the null hypothesis (p-value $< \alpha = 0.05$).

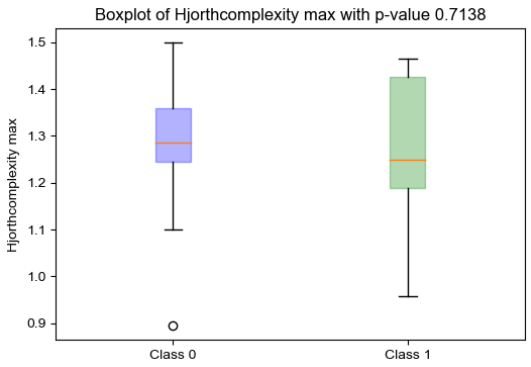
Feature	T	p-value
betadeltaratio mean	-3.006	0.004
BSI mean	-3.365	0.003
regularity mean	-2.753	0.014
betadeltaratio std	-2.935	0.005
spectralentropy std	-2.188	0.04
sampleentropy std	-3.329	0.004
alphadeltaratio min	-2.823	0.007
betadeltaratio min	-2.521	0.018
occipitalalphadeltaratio min	-2.818	0.007
PLI min	2.357	0.03
coherencetheta min	-2.311	0.03
alphadeltaratio max	-2.479	0.018
betadeltaratio max	-3.122	0.003
BSI max	-2.209	0.04

Table 3.4: Statistical significant features of F_X: t-statistic and p-value

Plotting the difference between the two classes of features `sampleentropy std` and `Hjorthcomplexity max`, Figure 3.15, we notice a bigger difference for `sampleentropy std` (p-value $= 0.004$) than for `Hjorthcomplexity max` (p-value $= 0.71$). We expect `sampleentropy std` to play a bigger role in predicting the class labels for the patients with anti-NDMARE based on this p-value.



(a)) Boxplot sampleentropy std for F_X with p-value for T-test



(b)) Boxplot Hjorthcomplexity max for F_X with p-value for T-test

Figure 3.15: Boxplots to show the statistical significance for sampleentropy std and not for Hjorthcomplexity max

Due to the violation of the normality assumption in some features, the Mann-Whitney U-test is also applied to F_X. The Mann-Whitney U-test is considered as the non-parametric equivalent to the two-sample independent t-test. It tests the equality of two groups. This test is advised when the data is not normally distributed. Indeed, continuous data, independence and similarity of distribution shapes are both required to be satisfied. The test statistic U of the Mann-Whitney U-test is a combination of two sub test statistics U_0 and U_1.

$$U = \min(U_0, U_1) \tag{3.20}$$

Each sub statistic U_i uses the adjusted rank-sum of sample i, denoted as R_i. The adjusted rank-sum is computed by ranking all observations from both samples together.

$$U_0 = N_0 N_1 + \frac{N_0(N_0+1)}{2} - R_0$$
$$U_1 = N_0 N_1 + \frac{N_1(N_1+1)}{2} - R_1$$

For N_0 and N_1 large enough, we can approximate the statistic U with a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The mean and variance are estimated by

$$\mu = \frac{N_0 N_1}{2}$$
$$\sigma^2 = \frac{N_0 N_1 (N_0 + N_1 + 1)}{12}$$

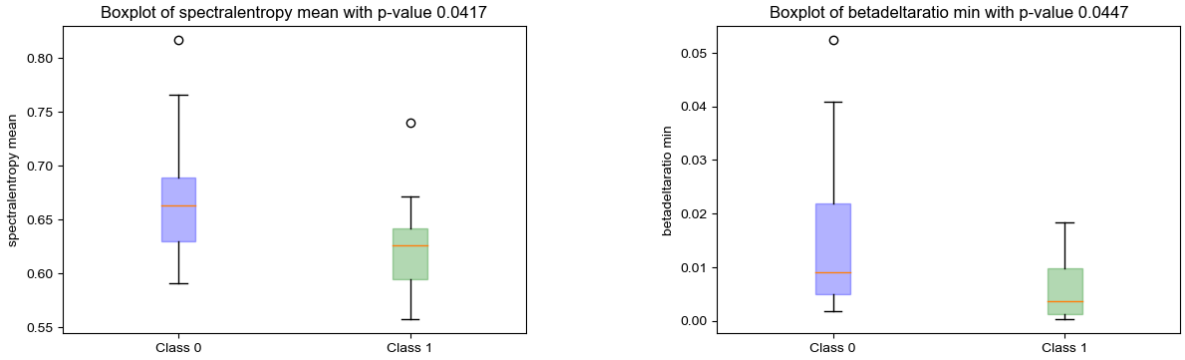
To compute a p-value, we check the probability of obtaining a value of U for which holds $U < -|u|$ or $U > |u|$ according to the normal distribution $\mathcal{N}(\mu, \sigma^2)$. Here u is the observed test statistic. This test is performed by using the function `stats.mannwhitneyu` in Python.

In Table 3.5, the values for the computed U statistic are displayed that have a p-value below 0.05. The features that show statistically significant differences between patients with class label 0 and label 1 for both the independent t-test and the Mann-Whitney U-test are emphasized in green.

Feature	U	p-value
deltapower mean	242	0.015
alphadeltaratio mean	86	0.034
betadeltaratio mean	88	0.045
occipitalalphadeltaration mean	82	0.029
BSI mean	82	0.029
spectralentropy mean	87	0.042
sampleentropy mean	81	0.027
regularity mean	84	0.034
Hjorthmobility mean	85	0.036
PLI mean	228	0.042
alphadeltaratio std	82	0.029
occipitalalphadeltaratio std	83	0.031
sampleentropy std	67	0.009
deltapower min	244	0.012
alphadeltaratio min	71	0.012
betadeltaratio min	88	0.045
occipitalalphadeltaratio min	75	0.017
coherencetheta min	87	0.042
alphadeltaratio max	82	0.029
occipitalalphadeltaratio max	82	0.029

Table 3.5: Statistical significant features of F_X: test statistic U and p-value. *Green features are also statistically significant for the independent t-test.*

Figure 3.16 shows the difference between classes for features `spectralentropy mean` and `detadeltaratio min`.



(a)) Boxplot spectralentropy mean for F_X with p-value for U-test

(b)) Boxplot betadeltaratio min for F_X with p-value for U-test

Figure 3.16: Boxplots to show the statistical significance for sampleentropy mean and betadeltaration min

4

Random Forest Classifier

In Chapter 2, we identified the Random Forest classifier as one of the most promising classification methods in the case of anti-NMDARE. In this chapter, the Random Forest classifier is described and analyzed with respect to the EEG data. The required parameters for a Random Forest model are discussed.

4.1. Definition

A Random Forest (RF) classifier is an ensemble method that combines multiple Decision Trees (DTs) into a final prediction and is therefore a tree-based method. The method was first introduced by Leo Breiman [36]. Ensemble methods can construct a classifier that produces a probability of class membership since it combines multiple classifiers [19]. In the case of ensemble Algorithms, the underlying classifier is not required to be highly accurate. Some terminology for ensemble methods are bagging, boosting or the random forest classifier. Bagging is short for bootstrap aggregation. This means random subsets (with replacement) are sampled from the training data.

4.1.1. Decision Trees

Tree-based methods segment the predictor space into simple regions [12]. Certain splitting rules are used to divide the predictor variables into multiple regions. At each node in the tree, a decision rule is formulated based on a feature. For an example of a DT with corresponding segmentation of regions see Figure 4.1.

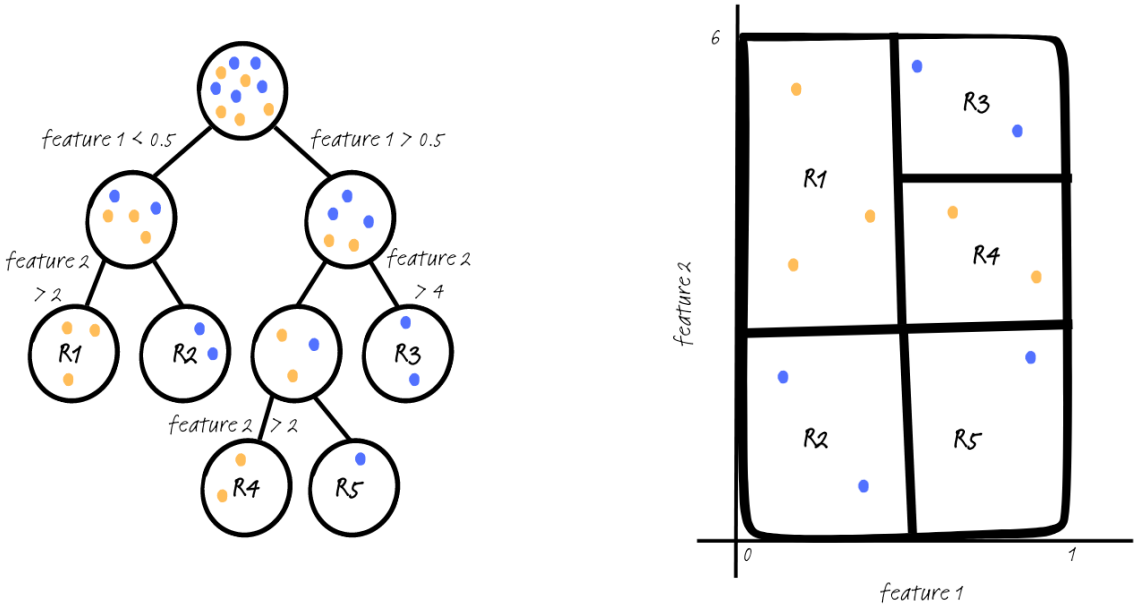


Figure 4.1: Example of a DT with corresponding segmentation of regions

At the final nodes, called leaves, a class label is assigned to the observation. A new observation from the test set belongs to the class which occurs the most in the region or leaf to which the observation is assigned. The DTs, and also the RF, are first trained on a training set. The observations from the training set determine the splits of the DTs as well as the class labels in all leafs.

The accuracy of a DT, or an RF, which is the proportion of misclassified observations can be measured by the error rate,

$$\frac{1}{I}\sum_{i=1}^{I}\mathbb{1}(\hat{y}_i \neq y_i) \tag{4.1}$$

The error rate can be computed for the test or training set, measuring the test or training error. Although accuracy in terms of the error rate is important for finding the best DTs, other measures may be preferred when training the DTs. At each node, the optimal feature for splitting the data has to be determined. The Gini index and the entropy are two common measures to determine this optimal feature. They are called impurity measures, namely both measures seek to produce the purest node split. This means that the splitting rule separates the dataset into two regions, each of which contains mostly observations from one class. Since DTs consider both classes when learning how to split the nodes, they perform well on imbalanced data.
The Gini index is defined by

$$G_m = \sum_{k=1}^{K}\hat{p}_{m,k}(1-\hat{p}_{m,k}) \tag{4.2}$$

in case of the 2-class classification: $G_m = 2 \cdot p_{m,1}(1-p_{m,1})$

The Gini index is a measure across the K classes. In node m, $\hat{p}_{m,k}$ is the proportion of observations belonging to class k. Smaller values for the Gini index are preferred as it indicates that the nodes are more pure. For the two-class case, when $\hat{p}_{m,1}$ is close to 0 or 1, one of the two factors in $\hat{p}_{m,1} \times (1-\hat{p}_{m,1})$ is small, hence the summation is small. It means that if an observation in the child node is randomly sampled, there is a low likelihood it will be misclassified. When the probability of the two classes are both close to 0.5, an observation from the child node can be randomly sampled from both classes.

An alternative measure, entropy is given by

$$D_m = -\sum_{k=1}^{K} \hat{p}_{m,k} \log(\hat{p}_{m,k}) \tag{4.3}$$

in case of the 2-class classification: $D_m = -\hat{p}_{m,1} \log(\hat{p}_{m,1}) - (1 - \hat{p}_{m,1}) \log(1 - \hat{p}_{m,1})$

Likewise, for the entropy holds that for $\hat{p}_{m,k}$ close to either zero or one, the entropy is small and therefore indicates a better split. In Figure 4.2, we see that both measures have their maximum value at $p = 0.5$. Both represent impurity well and can be used to train the DTs. One benefit of Gini impurity compared to entropy is its faster training time, making it computationally more efficient. Given the small number of observations in our dataset, this reasoning can be disregarded. Hastie et al. mention that both measures usually provide very similar results [37]. The performance of the two measures can depend on the specific dataset and therefore it is recommended to consider the impurity measures as a tuning parameter.

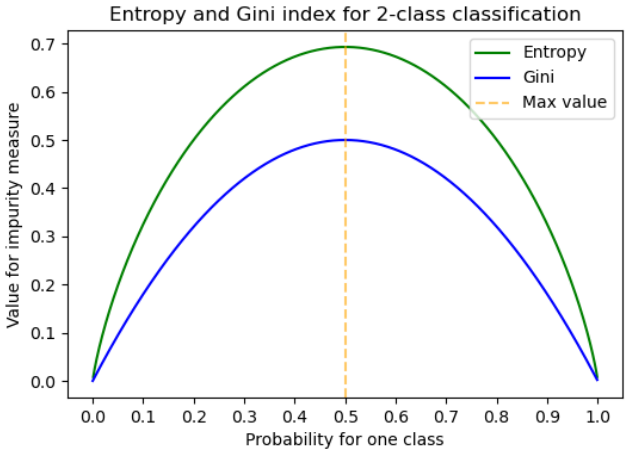


Figure 4.2: Entropy and Gini index values for $p_{m,k}$

Both measures are generally used when the quality of a split in the decision tree needs to be evaluated. To measure the accuracy of the entire tree, the classification error rate suffices.

4.1.2. Random Forests

Because a small change in data might result in a completely different decision tree, averaging multiple DTs will decrease the variance of the classification model. To decrease the variance, RFs use a technique called bagging [36]. With bagging, which stands for bootstrap aggregation, random subsets (with replacement) of the training data are sampled, where each bootstrapped subset is used for training a single DT. The final class label for an observation is determined by a majority vote among all DTs. By averaging the predictions from the DTs, a RF classifier reduces the impact of outliers or noisy observations that might lead to high variance in a single DT. Furthermore, the aggregation reduces the risk of overfitting. This risk means the model performs well on the training data, but poorly on new and unseen data. If a DT in the forest demonstrates high variance, indicating overfitting to its subset of the training data, another DT mitigates this issue by being trained on a different subset. The technique bootstrap aggregation is visualized in Figure 4.3.

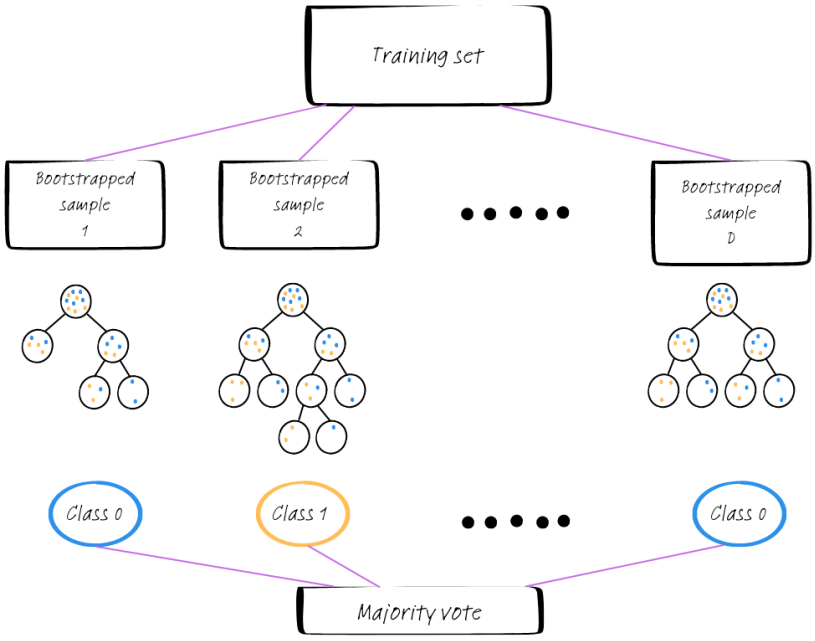


Figure 4.3: Bagging for RF classifiers

Besides decreasing the variance, RFs aim to decorrelate the DTs that are used for the final decision. DTs that use the same set of features and training data are correlated. By randomly selecting a subset m features from the p available features at each split, RFs can ensure the DTs do not use the same strong features. From these m features, the optimal split is determined. Including feature randomness ensures a strong predictor feature is not used in all bagged trees and so not all bagged trees behave similarly [12]. By using bagging and feature randomness, RFs create a less correlated forest of DTs. The DTs are however not independent due to the fact that they use the same features and training observations. This is limited by bagging and introducing feature randomness. Breiman proved that the error is bounded by the strength of a forest and the dependence between trees [36]. Strength essentially means the ability to correctly classify the observations.

Algorithms The RF method can be summarized in a few key Algorithms. Firstly, a number of DTs are generated using subsets of the training data that are randomly selected with replacement. This is called bootstrapping.

Algorithm 1 Training a Random Forest Classifier

Require: $D > 0$ decision trees
Require: input data X with output y
1: **for** $1 \leq d \leq D$ **do**
2: Randomly generate a bootstrapped data set of patient id's: $I_d \subset I$ (with replacement) and take X_{I_d} as training set for tree d where X_{I_d} is all data from the patients in I_d
3: Store I_d^c as out-of-bag samples from bootstrapped data
4: Go through DT Algorithm (2) with X_{I_d} as training set
5: Store as decision tree DT_d
6: **end for**
7: All DT_d form a trained Random Forest

In training each individual DT in the RF on a subset of the training set X, a random subset of features is selected. The optimal splitting criterion is either based on the Gini index or entropy. Each DT stops to grow when a stopping rule s is reached. Commonly used stopping rules to prevent overfitting in random forests are:

- the maximum depth,

- the minimum number of samples required to split a node,
- the minimum number of samples required to be at a leaf node.

Algorithm 2 Training a Decision Tree with feature randomness

Require: training set X with output y
Require: stopping rule s and subset size m
1: **while** stopping rule s does not hold **do**
2: Consider a random subset of features, $Z \subset P$ with Z of size m and P the set of features
3: Choose the feature from Z that best separates the dataset X w.r.t. class labels y using Gini index or the entropy
4: **end while**

Once the decision trees are trained, the RF can decide on the predicted class label for test set X_{test} by taking a majority vote on the predictions of all the individual decision trees.

Algorithm 3 Testing a Random Forest Classifier

Require: input data X
1: **for** $1 \leq d \leq D$ **do**
2: Go through decision tree DT_d with input X
3: Appoint class label $\hat{y}_d$
4: **end for**
5: Assign the majority class label

$$\hat{y} = \begin{cases} 0 & \text{if } \frac{1}{D}\sum_{d=1}^{D} \hat{y}_d \leq 0.5 \\ 1 & \text{if } \frac{1}{D}\sum_{d=1}^{D} \hat{y}_d > 0.5 \end{cases}$$

To evaluate and compare different RFs, the out-of-bag (OOB) error can be used as a performance metric. In Algorithm 1, the out-of-bag samples were stored in I_d^c. The classification error of the OOB samples is named the OOB error.

Algorithm 4 Evaluating a Random Forest Classifier

1: $OOB \leftarrow 0$
2: **for** $i \in \bigcup_{d=1}^{D} I_d^c$ **do**
3: Go through the decision trees that did not use X_i in training them
4: Assign majority class label $\hat{y}_i$ to X_i
5: $OOB = OOB + \mathbb{1}_{\hat{y}_i \neq y_i}$
6: **end for**
7: OOB error $= \frac{OOB}{|\bigcup_{d=1}^{D} I_d^c|}$

The OOB error is an unbiased estimate of the generalization error. Breiman stated that the OOB error accurately represents an error for the test set that has the same size as the training set [36].

4.1.3. Tuning parameters

The tuning parameters are the settings of a classification method that are specified before training the model. In Algorithm 2, the number of features m used per split is a tuning parameter, as is the stopping rule s. Algorithm 1 requires a decision on how many decision trees the RF consists of. Increasing the number of decision trees in a RF can improve the accuracy but can also lead to a computational expensive model. The aim of tuning parameters is finding the best combination to improve the performance of the classification model and to reduce over- and underfitting. Grid search is a method that evaluates different combinations of tuning parameters and select the one with best performance. This method are further discussed in Chapter 5. To conclusion, the tuning parameters of the RF classifier which we focus on are

- `max_features` (m)
- stopping rule s:
  - `min_samples_split`
  - `min_samples_leaf`
  - `max_depth`
- `n_estimators` (D)
- splitting rule: Gini or entropy

The optimal combination of these tuning parameters for our dataset is determined in Chapter 7. Algorithm 4 can assess the different tuning parameter settings.

4.2. Random Forest applied to EEG data

As explained in Section 3.3, our dataset includes clustered data. This means our goal is to compare variations of the Random Forest classifier or develop a clustered-based Random Forest classifier. We look at incorporating clustered data into a Random Forest classifier based on the three approaches of Galbraith [34]. The approaches (ignoring clustering, reducing clusters to independent observations and explicitly accounting for clustering) are discussed in Section 3.3.

4.2.1. Ignoring clustering

In this case, the assumption is that all EEG epochs are considered to be independent of each other. This corresponds to the definition of Z in Equation (3.3). Each row, or EEG epoch, of features is analyzed separately as input for the classifier.

$$\begin{aligned} &\hat{y}_k = f(z_k) \in \{0, 1\} \\ &\quad \text{where } k = 1, \ldots, 46.832 \\ &\quad \text{and } z_k = [z_{k,1} \ldots z_{k,29}] \end{aligned} \tag{4.4}$$

Here, f represents the random forest classifier. For patient i, a majority vote is computed for all EEG epochs belonging to the patient, i.e.

$$\hat{y}_i = \frac{\sum_{k=1}^{n_i} f(z_k)}{n_i} \in [0, 1]$$

If $\hat{y}_i > 0.5$, patient i will be classified in class with label 1.

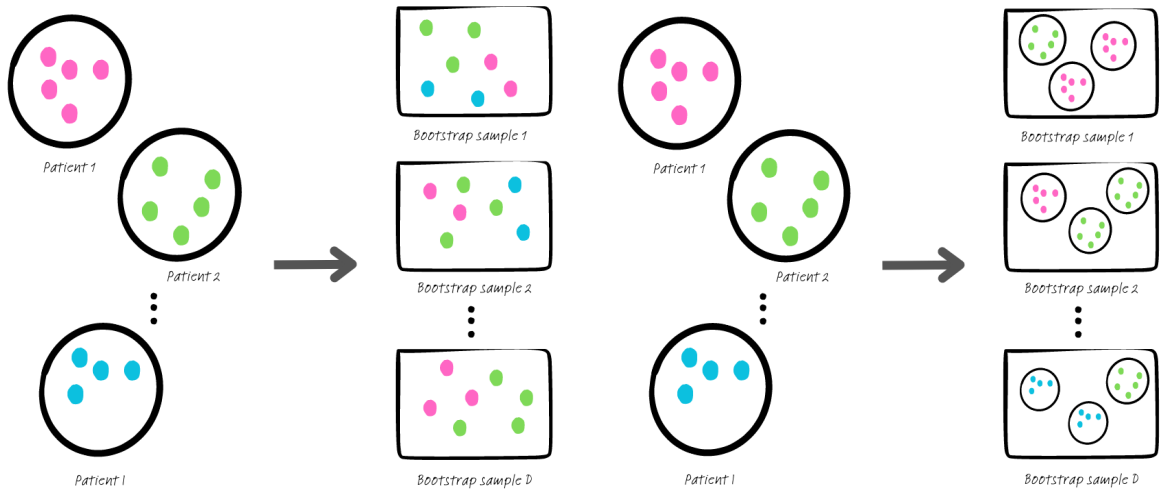
4.2.2. Reducing clusters to independent observations

Another solution that we proposed in Section 3.3 is to reduce the clusters in a way that they become independent observations. The matrix F_X, Equation (3.15), can easily be used as input data for the Algorithms.

4.2.3. Explicitly accounting for clustering

The data structures for this solution were discussed in Chapter 3 and their analysis was suggested for further research. In this section we provide a brief overview of several solutions for the RF classifier.

One solution proposed by Hu et al. [30] for a clustered-based RF classifier is subject-level bootstrapping. They select one measurement per patient to train the RF, perform the classification at the measurement level, and compute the majority vote per patient. This solution is visualized in Figure 4.4.



(a)) Measurement-level bootstrap for RF

(b)) Subject-level bootstrap for RF

Figure 4.4: Clustered-based bootstrapping for RF [30]

Another approach, suggested by Calhoun et al. [38], is the Repeated Measures Random Forest (RMRF) Algorithm. It incorporates a robust Wald statistic as the splitting criterion and applies an acceptance-rejection criterion to reduce the computational intensity and variable-selection bias. To address the dependence in the data within subjects, their Algorithm employs subsampling the data by subjects and uses a robust Wald statistic.
Hu et al. [30] reviewed extensions of the 'standard' RF method for clustered-based data and created an overview of these extensions. The overview can be found in [30]. For binary outcomes, such as our classification problem, they proposed the Binary Mixed Model forest as a solution to the clustered-based RF. This method combines the RF classifier with the Generalized Linear Mixed Model (GLMM) methodology and is discussed by Speiser in [29]. Implementing an RF model into a GLMM allows the BiMM to handle interactions among features and high-dimensional settings with nonlinear relationships. Other solutions proposed by Hu et al. [30] focus on quantitative outcomes. However, these models can potentially be adapted for our binary classification problem by using the logit function described in Equation (3.18). This functions transforms the quantitative outcome into a probability, which can be employed for classification problems.

5

Methodology

This chapter discusses the process of exploring the RF classifier for our EEG problem. An in-depth examination of the key techniques and approaches are provided in this chapter. Our methodology is based on the ML-based scheme of Rahmani et al. [5] who proposed the following phases: problem definition, dataset, data preprocessing, ML model development and evaluation. We have implemented these phases in order to generate reliable and valid research findings specific to this classification problem. It is important to note that the analysis of the data for better understanding its structure and characteristics is an essential part of the overall process. The analysis of the data is addressed in the preceding Chapter 3. Continuing with this chapter, we proceed to explain the remaining steps. Figure 5.1 (on the next page) outlines the three main components of this research: data preprocessing, model training and model evaluation.

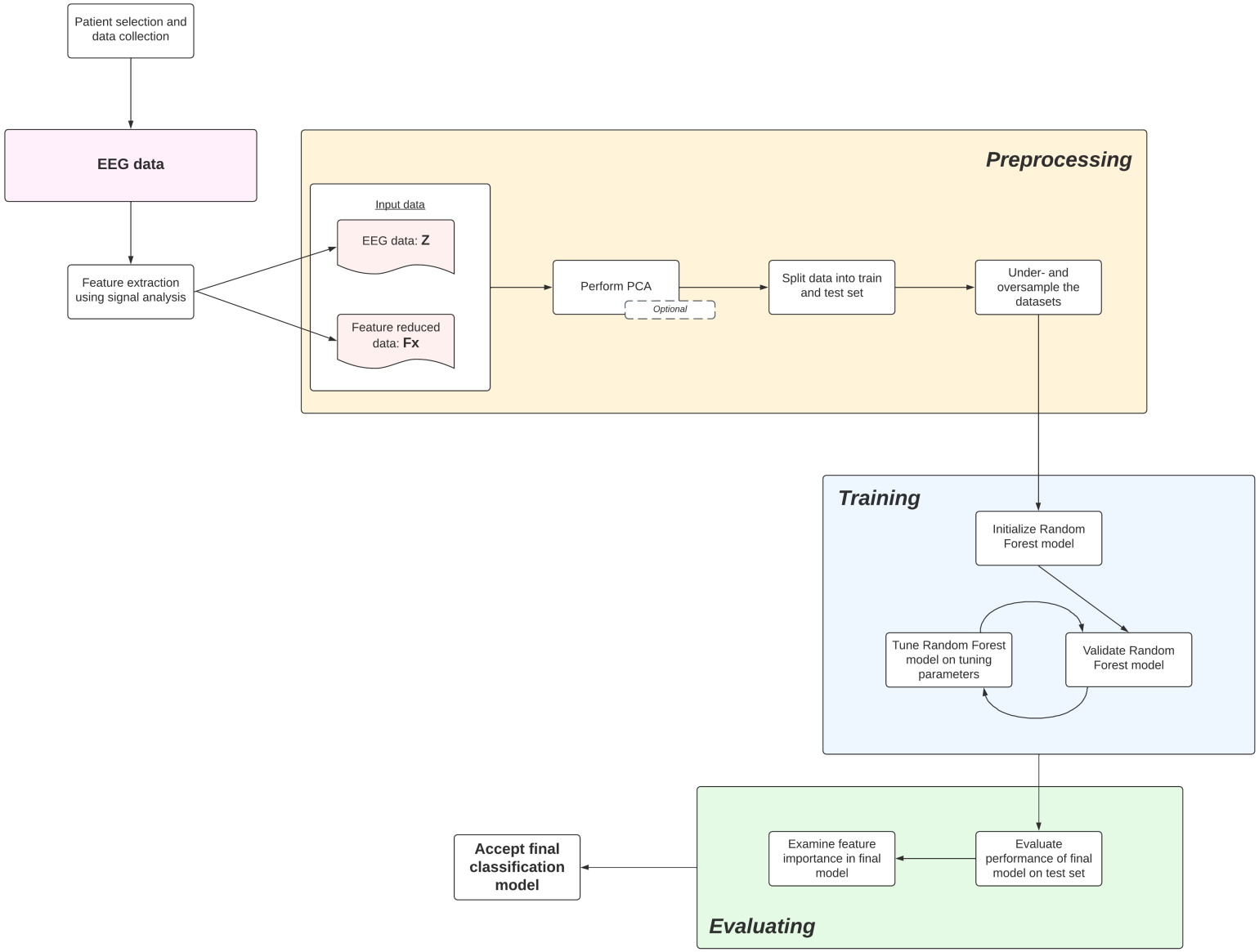


Figure 5.1: Decision flow: Classification methods overview

5.1. Preprocessing

Data preprocessing is an essential step in ensuring the datasets' quality for the classification model. Data preprocessing includes outlier detection, handling missing values and normalizing or standardizing the data. Rhamani et al. call these steps data cleaning methods [5]. There are no missing values in our dataset and normalizing or standardizing the data is unnecessary for the Random Forest classifier as it is not distance based. We did not emphasize on the outlier detection, as outliers are considered important in the context of this dataset. High or low values of certain features might be good indicators for a favorable or unfavorable outcome after the 12 months.
Additionally, data reduction methods might be employed when the data is high-dimensional. High-dimensional data is very common in healthcare. As the number of dimensions increases, a classification model faces challenges in effectively capturing meaningful patterns and generalizing on the data [39]. Additionally, there is a higher likelihood of the model incorporating redundant features that do not contribute significantly to its predictive ability. Therefore, high dimensions weaken the performance of ML algorithms [17]. Data reduction methods should focus on preserving relevant and informative features [17]. Feature reduction methods map the original feature space to a lower dimensional space while keeping the effective information from the original features [39]. The data reduction method that we focus on is Principal Component Analysis.

5.1.1. Principal Component Analysis

In Section 3, we have seen various formulations of our dataset. The EEG dataset is indicated as matrix Z of size 46.832×29 from Equation (3.3). Summarizing the epochs of the EEG data per patient has generated a 44×116 matrix F_X, defined in Equation (3.15). High-dimensional matrices can be reduced to lower dimensions using Principal Component Analysis (PCA).

PCA is a feature extraction method that aims to find linear combinations of the variables, $X = (X_1, \ldots, X_p)$. Each $X_j \in \mathbb{R}^N$ represents a vector of values for feature $j \in \{1, \ldots, p\}$ for N observations. The coefficients in the linear combination are displayed in the vector $\phi = (\phi_1, \ldots, \phi_p)$. Any linear combination $Z_m \in \mathbb{R}^N$ is formulated as

$$\begin{aligned} Z_m &= \phi_m^T X = \sum_{j=1}^{p} \phi_{j,m} X_j \\ \text{with } &\sum_{j=1}^{p} \phi_{j,m}^2 = 1 \end{aligned} \tag{5.1}$$

The constraint of $\sum_{j=1}^{p} \phi_{j,m}^2 = 1$ is necessary to prevent the issue of an excessively large linear combination.

The fundamental principle of PCA is to maximize the variance of linear combinations Z_m's. By maximizing the variance, the uniqueness of the dataset is mostly captured. The most interesting information is typically stored in the spread or variance of the data. The linear combinations are called the principal components (PCs). In total, there can be $M = \min\{N - 1, p\}$ PCs [12]. The variance of each variable Z_m and the covariance between variables Z_j's is presented in the covariance matrix $S \in \mathbb{R}^{n \times n}$, consisting of values

$$s_{j,k} = Cov(Z_j, Z_k) = \frac{1}{N} \sum_{i=1}^{N} (Z_{i,j} - \hat{\mu}_j)(Z_{i,k} - \hat{\mu}_k)$$

where the observed means, $\hat{\mu}_j$, are calculated using the sample set in the linear combinations, Z_j.
The covariance is a measure of linear dependence. The sample variance of Z_m is equal to $\text{Var}(Z_m) = \text{Var}(\phi_m^T X) = \phi_m^T S \phi_m$. As PCA wants to maximize this variance under the constraint $\phi_m^T \phi_m = 1$, we formulate the optimization of the variance with a Lagrange multiplier λ_m:

$$\max J(\phi_m) = \phi_m^T S \phi_m + \lambda_m (1 - \phi_m^T \phi_m) \tag{5.2}$$

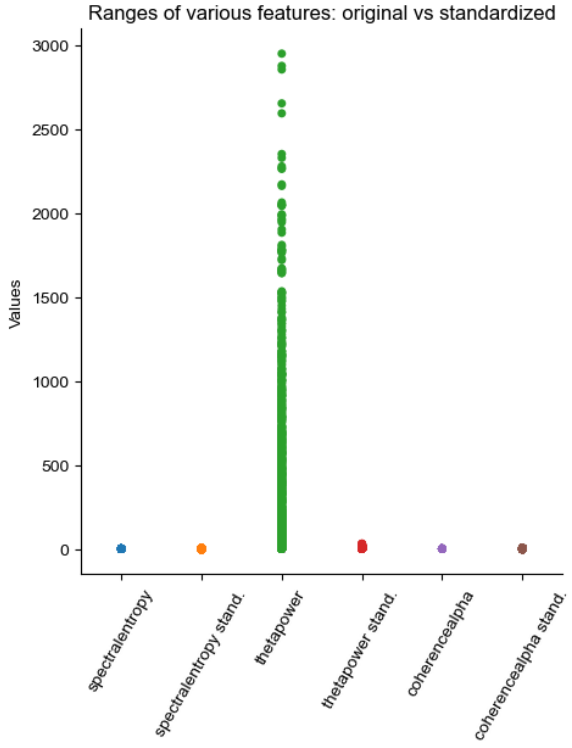
Optimizing for ϕ_m, we set the derivative to zero and find the same formulation of an eigenvalue problem.

$$\begin{aligned}\frac{\partial \phi_m^T S\phi_m + \lambda_m(1-\phi_m^T\phi_m)}{\partial \phi_m} &= 0\\ S\phi_m - \lambda_m\phi_m &= 0\\ S\phi_m &= \lambda_m\phi_m\end{aligned}$$

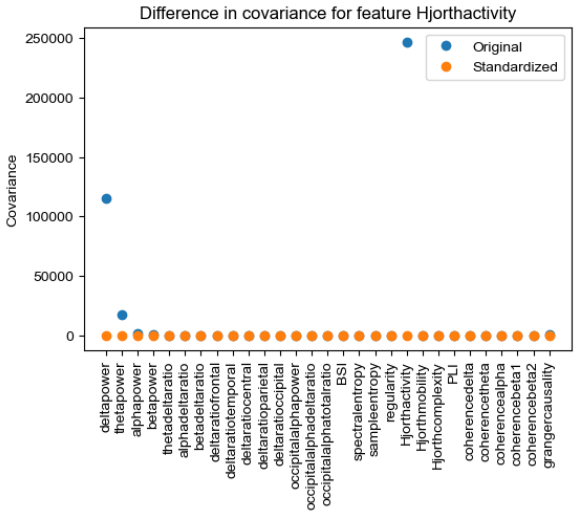
For a nonzero square matrix, in our case S, the vector equation $Sv = \lambda v$ is called an eigenvalue problem. The values $\lambda \neq 0$ and $v \neq 0$ are called eigenvalues and eigenvectors. The first M eigenvectors ϕ_m's with largest eigenvalues $\lambda_1, \ldots, \lambda_M$ are called the principal components [40]. All ϕ_m's are real since S is a symmetric matrix. We look for the largest λ_m and it's corresponding eigenvector ϕ_m. For this eigenvector and -value the variance of $Z_m = \phi_m^T X$ is the largest. This linear combination is called the first principal component, $\text{PC}_1 = \phi_1^T X$. In cases where certain features have a high value in PC_1, it is likely that the features correlated with them will have a lower value in PC_2. The reason is that the linear combination PC_2 has largest variance subject to the constraint that it is uncorrelated with PC_1 [12]. In a two-dimensional plane, uncorrelated vectors can be visually displayed as PC_2 is orthogonal to PC_1. Ultimately, M principal components can be constructed. Since the goal is to reduce the dimension of the dataset, the first k principal components are chosen such that the cumulative ratio of variance that is explained by the first k principal components, is larger than 90%. The cumulative ratio of explained variance, Equation (5.3), is obtained by sequentially summing up the explained variance of each principal component, PC_m, in the order of their occurrence. Equation (5.3) assumes the features X_j are centered around mean zero, and therefore so are the linear combinations Z_m.

$$\text{Explained variance by PC}_m = \frac{\frac{1}{N}\sum_{i=1}^{N} Z_{i,m}^2}{\sum_{j=1}^{p}\frac{1}{N}\sum_{i=1}^{N} X_{i,j}^2}, \tag{5.3}$$

Since PCA finds linear combinations of features $X = (X_1, \ldots, X_p)$ and uses the covariance matrix, it is important to scale the features to the same scale with a technique called standardization. In Figure 5.2a), we see for example that values of `thetapower` can be 1000 times larger than for BSI. The correlation matrix does not change under standardization, the covariance does, see Figure 5.2b).



(a)) Ranges of various features



(b)) Difference in covariance between original and standardized Hjortactivity

Figure 5.2: Why you need to standardize before PCA

Standardization is a technique where the features are scaled towards a distribution with mean zero and standard deviation 1. As the variance plays a crucial role in PCA, standardization of the features is preferred. Furthermore, standardization ensures the variables X_i are centered around mean zero. For each feature j, the mean μ_j and σ_j from Equations (3.4) and (3.5) are estimated. The feature values $x_{i,j}$ are assigned new values using the following equation

$$x_{i,j}^{\text{new}} = \frac{x_{i,j} - \hat{\mu}_j}{\hat{\sigma}_j}$$

5.1.2. Train and test split

In order to evaluate the performance and generalization ability of a classification model, we split our data into two distinct sets: a training and test set. The process of splitting data into a train and test split plays a fundamental role in assessing the ability to predict labels for unseen data. The training set is used for training and fitting the classification model. The testing set serves as an independent set that evaluates the performance of the model. To guarantee unbiased performance scores, it is crucial to maintain distinct training and test sets. By ensuring that the model has not been exposed to the data in the test set during training, we can avoid introducing bias into the performance metrics. Furthermore, we specifically separate the data before performing the under- and oversampling technique to prevent the possibility of using the same observation in both sets.

Python Two functions are created for splitting, namely `train_test(X,y)`, and `train_test_EEG(X,y,y_EEG)`. The separate function `train_test_EEG(X,y,y_EEG)` is created for the EEG data with repeated measurements. Chen et al. suggest that all observations of a patient should be in the same set [11]. The splitting process is stratified based on the class labels y. This approach guaranteed that both classes are proportionally represented in both the training and test set.

5.1.3. Class imbalance

For effective model development, it is essential that the dataset contains a sufficient number of data samples for each class. An effective approach to deal with class imbalance may involve augmenting the dataset with additional instances of the minority class or removing instances of the majority class from the dataset [11]. The impact on our model performance of under- and oversampling is explored in this research.

Undersampling uses all observations of the minority class. The total number of observations from the minority class is sampled from the majority class. This technique is implemented with the function `undersample()`. A separate function for handling the clustered EEG data was developed, ensuring that observations from each patient are sampled together, `undersample_EEG()`.
Oversampling, on the other hand, copies all observations from the majority class and samples with replacement from the minority class until the dataset contains the same amount of observations per class label. The functions `oversample()` and `oversample_EEG()` are used for this technique.

5.2. Training a Random Forest classifier

In Chapter 4 we presented the Random Forest classifier along with its algorithm, as described in Algorithm 1. In Python, this can be summarized using the `RandomForestClassifier()` function from sklearn [41]. By utilizing the `.fit()` and `.predict()` methods, we can train the model on the training set and make predictions on the test set using the trained model. The performance metrics are introduced in Chapter 2. For healthcare applications, we prioritize a higher score for sensitivity, see Chapter 2, as we want to classify all patients with class label 1 correctly.

5.2.1. Default Random Forest

To understand the training ability of a RF classifier for our datasets, we initiate the training by using a default parameter settings for the RF algorithm. The default setting is self-formulated.

5.2.2. Tuned Random Forest

The next step involves optimization of the RF model. In Section 4.1.3, the tuning parameters are discussed. They are the settings of a classification method that are specified before training the model. The parameters should be tuned so that a better RF model can be found. To find an improved version of the RF model, it is necessary to tune its parameters. By adjusting the tuning parameters, we aim to find the configuration that leads to higher scores for the performance metrics. In Python, the function `GridSearchCV` performs the parameter tuning. The function explores different combinations of parameter settings and identifies the best performing model within these combinations. For the case of anti-NMDARE, we have prioritized the F-score as important performance metric. The function identifies the model that performs best on this metric, followed by the other scores. The function `GridSearchCV` uses a technique called cross-validation. With cross-validation, a dataset X is divided into k folds, where $k-1$ folds are used for training the RF model, while the remaining fold evaluates the model's performance metrics. The technique is displayed in Figure 5.3.

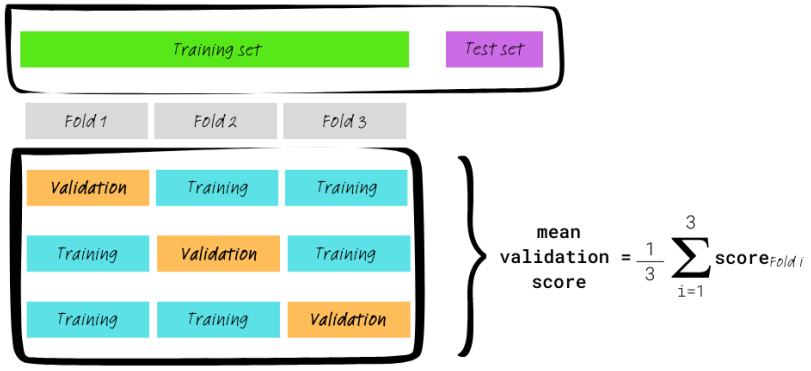


Figure 5.3: k-fold cross-validation technique used for tuning parameter tuning [42]

The function `GridSearchCV` calculates the mean performance score across the validation folds, providing an average score for the parameter setting using the training folds. The parameter setting with the highest mean F-score is chosen for the tuned RF model.

5.3. Evaluation

In the evaluation process, the strengths and weaknesses of the classification model are examined [5]. In Chapter 2, we identified four main characteristics of a classification method to evaluate: quality, accuracy, robustness and interpretability.

5.3.1. Quality

Since the quality of a classification model is measured by the performance metrics including accuracy, we do not consider accuracy as a separate subsection in this section. The performance metrics are discussed in Chapter 2.

Python Python has a function for each performance metric, `accuracy_score`, `recall_score`, `precision_score` and `F-score`. We defined the function `specificity_score` using `confusion_matrix`.

5.3.2. Robustness

Analyzing the robustness of our RF model, we gain insights in its behaviour and reliability. Our goal is to examine how our model performs in diverse scenarios. It is important to highlight that the statistical mechanism of RFs are still being explored and researched [43]. Many theoretical studies have mainly focused on simplified versions of the RF algorithm [44]. The mathematical properties are difficult to analyze due to the RFs' complex underlying mechanisms. Consistency assures that the classification model will improve its performance as more data becomes available. A consistent classifier converges to the underlying true function of the data (and their labels) as the number of observation increases.

Biau et al. [43] provide the proof that an averaged classifier of consistent classifiers is also consistent. This is a proof for simplified versions of the RF classifier. Gao et al. [45] add that the consistency of a RF depends on the consistency of individual random trees. The proof given by Breiman [36] state that RFs do not overfit as more trees are added to the RF. Furthermore, Breiman proves that the generalization error is bounded from above by a function based on the strength of the individual classifiers and the dependence between them [36]. A proof of consistency for our model requires a comprehensive analysis of the convergence properties as the sample size increases. Our focus does not lie here since we possess over a small sample size. We will conduct the analysis of robustness based on our specific data and model's performance.
Robustness refers to the model's ability to generalize on unseen data, see Chapter 2 for the definition. A robust model can handle various model's settings and changes in the data. By maintaining its performance in different initializations of the parameters, a robust model is less sensitive to changes and indicate a strong ability to generalize on unseen data. There are several ways to introduce variability in the training process of our RF classifier. This can be achieved by using different training sets with cross-validation or by trying out different parameter settings.

- Cross-validation
- Varying parameter settings

Cross-validation

To obtain a less biased estimate of the performance metrics, we use the technique called cross-validation [13]. The purpose of this technique is to train and evaluate the RF model k times, see Figure 5.3. We therefore obtain k scores for the performance metrics, allowing us to assess their variability. A score that is centered around a value might indicate a higher level of robustness. However, this is not the only aspect of robustness. In Chapter 2, we state that there is a tradeoff between bias and variance, aiming to generalize on unseen data.

Python The function `cross_validate(model, X, y, cv = k, fit_params, scores)` conducts a k-fold cross validation on X and y for model `model` with parameters `fit_params`.

Parameter settings

Grid search allows us to explore and examine various parameter settings. Ultimately, the best setting is identified based on scores of the performance metrics. By analyzing the results of the parameter grid, we can gain valuable insights into the robustness of the model and how the model performs in different settings.

5.3.3. Interpretability

In the case of anti-NMDARE, because the output of our model is whether a patient has a positive or negative prognosis for the treatment, it is essential that the classification model is explainable. It should be clear how the classifier arrives at their predictions. Our approach aligns with Breiman's path of initially selecting the model that achieves high scores in terms of the performance metrics. Subsequently, we try to understand which features the model identified as important, contributing to its predictive performance.

We focus on two different approaches on determining the most important features; Mean Decrease in Impurity (MDI) and through SHAP values. Yigit et al. [46] used two ways of measuring variable importance for an RF classifier; via Mean Decrease Gini (MDG) and via the SHAP algorithm. The MDG method evaluates the decrease in impurity according to the Gini index that results from splitting on a feature. MDI is a generalized version of MDG. The SHAP algorithm is a technique used for the explainability of ML models. SHAP, that originated from the concept of cooperative game theory, utilizes Shapley value to analyze the contribution of each feature to the final outcome. As stated by Loh et al. [9], SHAP is particularly suitable for tree ensemble ML techniques such as RFs and XGBoost.

Mean Decrease Impurity

The importance measure of a splitting variable in a RF is determined by the improvement in the split criterion at each split within each tree [37]. Per feature, the importance measures are averaged over

all trees in the forest [12]. The split criteria in our case are the Gini index or entropy, see Equations (4.2) and (4.3) in Chapter 4, depending on the dataset. It is worth noting that as correlation between features increases, the MDI becomes less effective in detecting the most relevant features [47]. Figure 5.4 visualizes the splitting qualities of two features. Examining the two splits, it is evident to see that feature 2 would receive a higher importance measure.

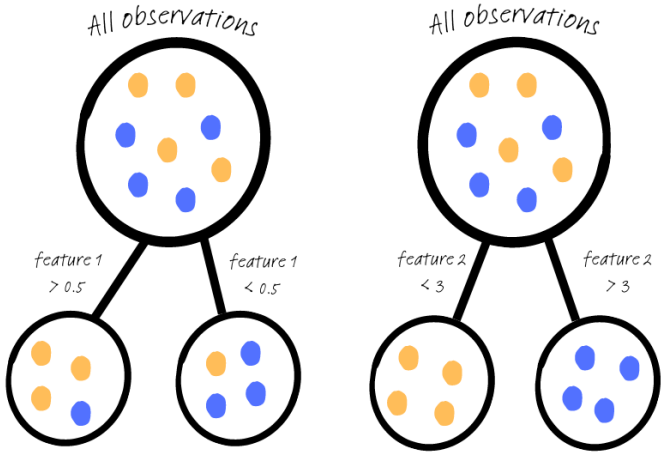


Figure 5.4: Visualization of feature importance using impurity measures [48]

Python The MDI is computed using the `.feature_importance_` property of the `RandomForestClassifier()`, after the RF is fitted. It is worth noting that the feature importance from scikit.learn is based on the training set [41]. The MDI is defined as

$$\text{Imp}(X_j) = \frac{1}{D}\sum_{d=1}^{D}\sum_{t\in d}\mathbb{1}_{X_j \text{ used for splitting node } t} \times \frac{N_t}{N}\Delta i(s,t) \tag{5.4}$$

where t is a node in tree d and N_t the number of samples in node t. The decrease in impurity of split s at node t is defined by

$$\Delta i(s,t) = i(t) - \frac{N_{t_L}}{N_t}i(t_L) - \frac{N_{t_R}}{N_R}i(t_R)$$

where node t is split into a left t_L and right t_R node [49]. The equations for the impurity measures for node t, $i(t)$, are given in Equations (4.2) and (4.3) in Chapter 4.

SHAP values

The Shapley Additive exPlanations (SHAP) algorithm is a powerful and interpretable method used for explaining the predictions of classification models. SHAP is based on the concept of cooperative game theory, where the Shapley values tells us how the payout of a game is distributed over the players. The prediction task for a single observation can be considered as the 'game' and the features as the 'players'. The 'payout' of the prediction task is the difference between the prediction value and the average prediction over all observations. The SHAP algorithm calculates the contribution of each feature by considering its effect on the prediction, helping us to understand feature importance in the classification model. The contributions per class are summarized in Shapley values. Shapley values are computed per feature and patient and defined per class. In the subsequent notations, the subscript 0 denotes the value corresponding to class label 0, while 1 indicates the values for class 1. Here, we will provide the explanation for class 0. Due to the symmetry of Shapley values, the values for class 1 can be obtained by making the values of class 0 negative. The predicted value, $f(x)^{(i)}$, of patient i for class 0 depends on the features. The contributions of features $j \in \{1,\dots,p\}$ on the prediction of patient i are summarized in Shapley values $\phi_{j,0}^{(i)}$. The values adhere to the following equation, Equation (5.5).

$$f_0^{(i)}(x) = \mathbb{E}(f_0(x)) + \sum_{j=1}^{p}\phi_{j,0}^{(i)}(x_{i,j}) \tag{5.5}$$

Here, $x_{i,j}$ represents the value of feature j of the patient i, $f_0^{(i)}$ is the prediction from an RF model expressed in a probability belonging to the specific class and $\mathbb{E}(f_0(x))$ is the expected value for the specific class. The output $f_0^{(i)}(x)$ can be estimated by `.predict_proba()` from the `RandomForestClassifier` function in Python.

The Shapley values, $\phi_{j,0}$, are calculated by Equation 5.6 [50].

$$\phi_{j,0} = \sum_{S \subseteq P \setminus \{j\}} \frac{|S|!(P - |S| - 1)!}{P!} [v(S \cup \{j\}) - v(S)] \tag{5.6}$$

The formula considers all possible subsets of features, denoted by S. For feature j, to calculate $\phi_{j,0}$ we iterate over all possible subsets of features that does not contain feature j. For these subsets, we compare the prediction when including feature j with the prediction ($v(S \cup \{j\})$ when excluding feature j ($v(S)$). This difference is calculated by $v(S \cup \{j\}) - v(S)$ and represents the marginal contribution of feature j to subset S. The Shapley value is a weighted average of these marginal contributions, weighted by the number of ways each subset can occur. This weight is give by

$$\frac{|S|!(P - |S| - 1)!}{P!}$$

All Shapley values add up to $v(P)$. The set function $v(\cdot)$ represents the prediction for feature values in the set S which is calculated by marginalizing over the features that are not included in S [51], i.e.

$$v(S) = \int \hat{f}(x_1, \ldots, x_p) d\mathbb{P}_{x \notin S} - \mathbb{E}(\hat{f}(X)) \tag{5.7}$$

A detailed explanation of Shapley values is provided in Appendix C.

A specific SHAP algorithm designed for tree-based models is TreeSHAP. TreeSHAP is an efficient method in estimating feature importance for tree-based models [51]. The Shapley value is computed per DT but due to their additive property easily averaged over all DT's in an RF model.

SHAP values are symmetrical, meaning that a positive value for class with label 1 implies a negative SHAP value for class with label 0. A positive SHAP value of feature j for patient i for class 1 indicates that feature j increases the probability of patient i belonging to class 1. The same feature decreases the probability of patient i belonging to class 0 by the same amount. SHAP feature importance (FI) for feature j is simply the average of all absolute Shapley values over all patients, Equation (5.8).

$$FI_j = \frac{1}{I} \sum_{i=1}^{I} |\phi_j^{(i)}| \tag{5.8}$$

Visualized, the feature importance of an RF model could resemble Figure 5.5

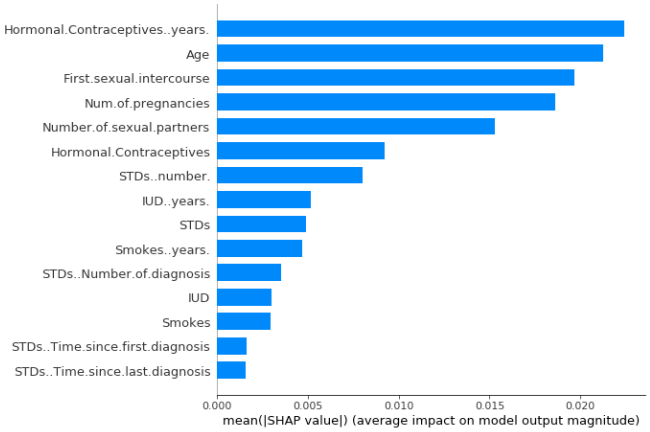


Figure 5.5: Example feature importance plot SHAP [51]

The figure indicates that in this example, the number of years with hormonal contraceptives is an important indicator for predicting the risk of cervical cancer. On average the feature added a value of 0.024 to the predicted risk. In Figure 5.6, a threshold is observed for years with hormonal contraceptives that guarantee a higher SHAP value for the predicted risk.

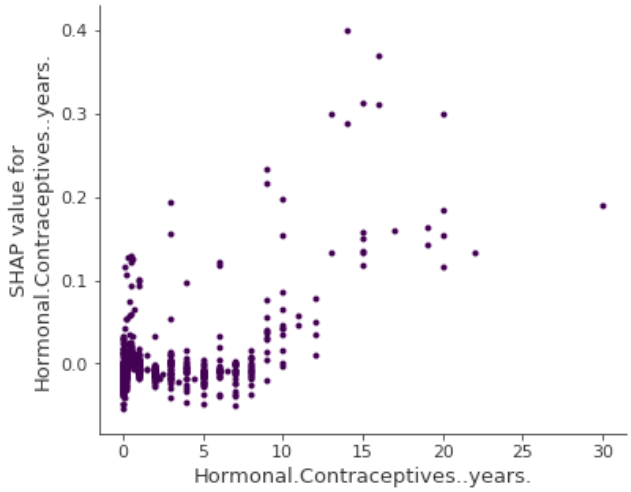


Figure 5.6: Example dependence plot SHAP

5.4. Code availability

The implementation scripts can be requested through the git repository at github/femkeluck. The EEG data from Erasmus MC is confidential and therefore not available. The scripts can be tested on the simulation data.

6

Simulation study

A simulation study allows us to investigate the complex structure of the RF classifier, evaluate its performance and make informed decisions in a controlled manner. The simulation analysis aims to better present the behavior and outcome of the RF classifier in the case of cluster-based data. In this chapter, we simulate a realistic dataset that closely resembles the EEG scenario from Chapter 3. In this chapter, we aim to gain valuable insights and explore possibilities to optimize the RF classifier by simulating the data of repeated measures.

6.1. Simulated data

The simulated data set that we use in this chapter has the same structure as our EEG data set from Chapter 3, i.e.

Patient	Epoch	Label	Features		
1	1	y_1	$x_{1,1,1}$	$\dots$	$x_{1,1,p}$
1	2	y_1	$x_{1,2,1}$	$\dots$	$x_{1,2,p}$
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
1	n_1	y_1	$x_{1,n_1,1}$	$\dots$	$x_{1,n_1,p}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
I	1	y_I	$x_{I,1,1}$	$\dots$	$x_{I,1,p}$
I	2	y_I	$x_{I,2,1}$	$\dots$	$x_{I,2,p}$
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$
I	n_I	y_I	$x_{I,n_I,1}$	$\dots$	$x_{I,n_I,p}$

Table 6.1: Structure of the EEG data

where I is the number of patients, n_i the number of EEG epochs of patient i and p is the number of features.
We simulate $p = 29$ features for $I = 44$ patients where each patient has a different number of EEG epochs n_i. The number of EEG epochs is randomly sampled from values ranging between 500 and 1500. To further simulate the dataset, we employed a two-step process. First, we sampled the class label for each patient i from a Bernoulli distribution with probability $p = 0.2$, ensuring a representation of the class imbalance from the real dataset. This means a patient receives class label 1 with a probability of 0.2 and label 0 with a probability of 0.8. An unfavorable outcome was considered if a patient received label 1. Consequently, the variables for these patients were sampled from the same distribution as those with class label 0, except for 6 features. These 6 features are indicated as informative features, while the remaining 24 features were redundant. The complete definition with distributions of the simulated dataset in Python can be found in Appendix D. Sampling the label first and then the features provides a more realistic representation of the underlying process. In the natural order of events, when a patient has a disease, they tend to experience specific side effects. It is not the case that certain features cause the patient to become sick. In certain scenarios where the features are primary driving

factors influencing the class label, the approach of first sampling the features and then assigning the class label might be more appropriate.

We check that the correlation matrix of the simulated data, Figure 6.1, does not show any unusual patterns. Additionally, it is important to note that the simulated data set intentionally incorporates class dependent features. However, the density of these features differ slightly so that the level of complexity observed in our real EEG dataset is mirrored, see Figure 6.2. We expect that the classification task for the simulated dataset poses a comparable level of difficulty. We approach this dataset in a similar manner as in Chapter 3; the data can be treated as independent EEG epochs in matrix Z, or summarized into clinically relevant features in matrix F_X. We use the following notations for the simulated sets: Z_{sim} and $F_{X\text{sim}}$

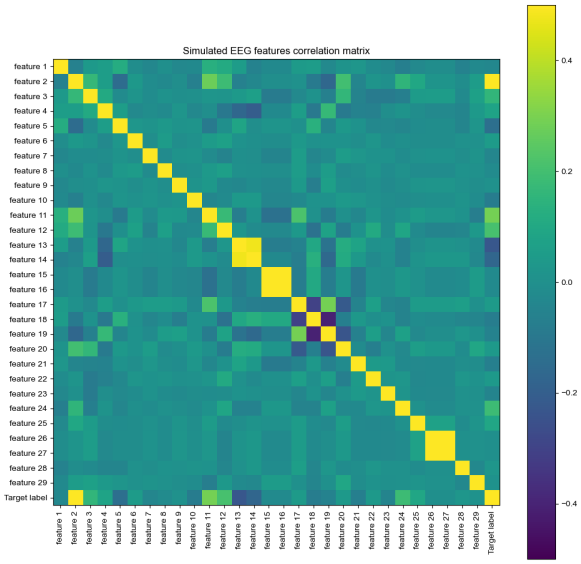


Figure 6.1: Correlation matrix of simulated EEG features

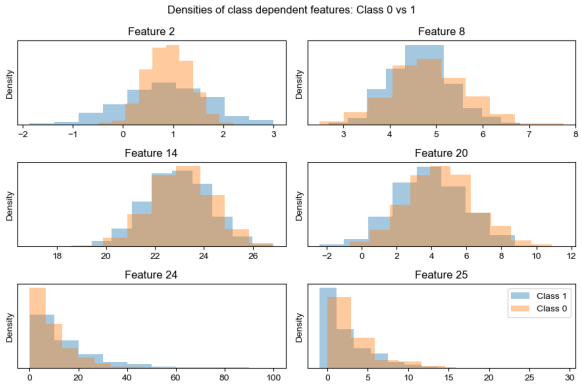


Figure 6.2: Density of class dependent EEG features in simulation

6.2. Preprocessing

Before initiating the training process of the classifier, we undertake several data preprocessing steps as outlined in Chapter 5.

6.2.1. Principal Component Analysis

For the EEG dataset, we apply PCA to assign weights to the relevant features. This step reduces the number of features in Z_{sim} from 29 to 14. Due to the fact that the number of features exceeds the number of observations in the $F_{X\text{sim}}$ dataset, we choose to not perform PCA on this dataset. PCA is performed early in the process, since it then has the ability to leverage all variables in order to compute covariances and identify the features that contribute the most to the overall variance within the dataset. Figure 6.3 shows the first two principal components of the entire EEG dataset and Figure 6.4 displays the corresponding weights assigned to the features in PC 1. The weights assigned to features 15 and 16 are justified due to their strong positive correlation. It is noteworthy that features exhibiting negative correlation receive opposing weights. Figure 6.1, representing the correlation matrix, reveals darker shades for features 17, 18, 19, and 20, indicating their negative correlation. Similarly, feature 4 demonstrates a darker shade in relation to features 13 and 14, resulting in the assignment of opposite weights to these features. On the other hand, features 10 and 21 exhibit low weights. This connects to the low covariances observed in the correlation matrix, where the entire row associated with these features appears to be close to a value of 0. Appendix E contains scatterplots of both the low and high covariance features, accompanied by additional explanations. Additionally, the density functions of these features are also presented in Appendix E.

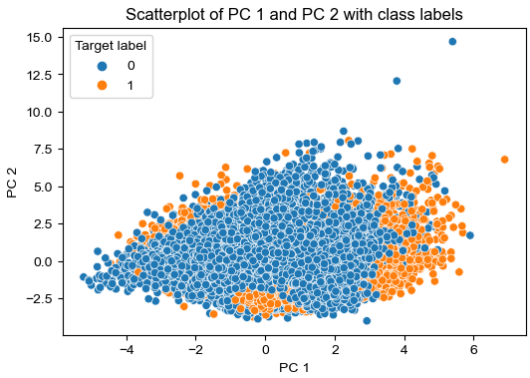


Figure 6.3: PC 1 and PC 2 of the EEG dataset Z_{sim}

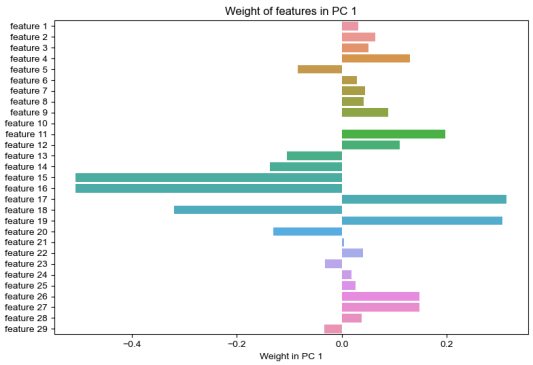


Figure 6.4: Weight per feature in PC 1

6.2.2. Train and test split

The proportion of each class present in both the training and test sets is highlighted in Figure 6.5. As mentioned in Chapter 5, the train and test split is done with stratification and based on the number of patients.

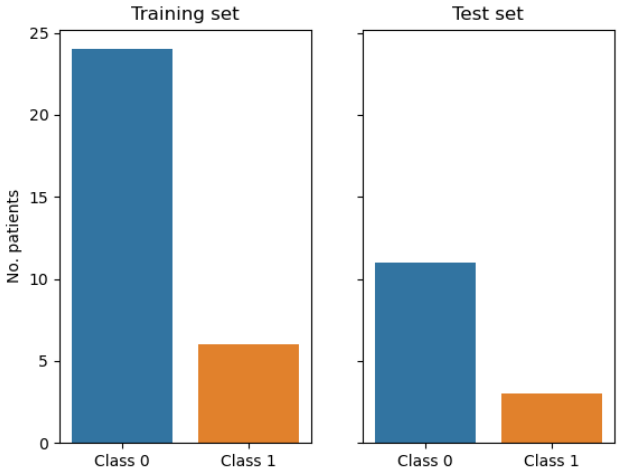


Figure 6.5: Train and test split for $F_{X\text{sim}}$, stratified on the class label

6.2.3. Class imbalance

The simulated dataset exhibits the same class imbalance as the original EEG dataset. The minority class with label 1 contains 9 patients, whereas the majority of all patients, namely 35, have class label 1. The techniques called under- and oversampling can solve this issue. An overview of the number of observations present in each dataset are displayed in Table 6.2.

	Number of observations			
	Train		Test	
Class label	0	1	0	1
Undersampled	6	6	3	3
Normal	24	6	11	3
Oversampled	24	24	11	11

Table 6.2: Overview of the number of observations in each dataset with class label

In Figure 6.6, we see the effect of under- and oversampling the training set of $F_{X\text{sim}}$. In Appendix E,

Figures E.3 and E.4, the effects of under- and oversampling on the EEG training set and on the EEG PCA set are displayed.

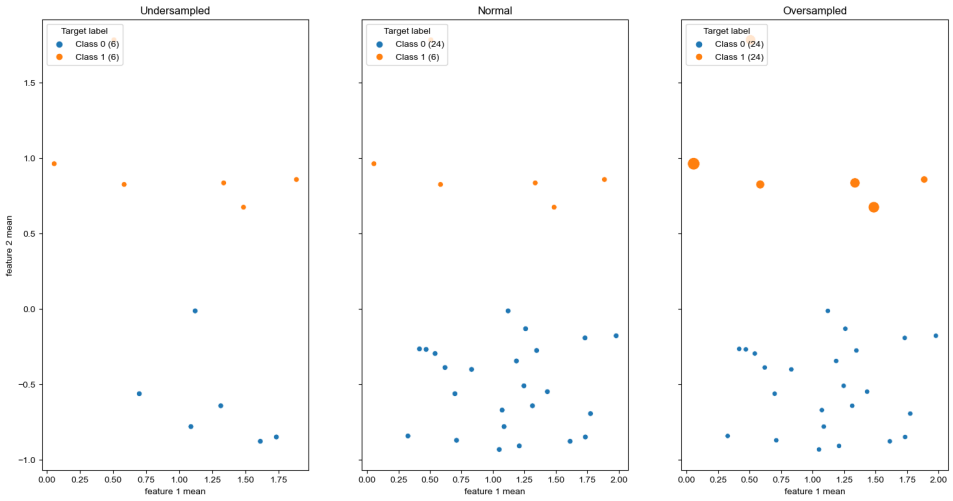


Figure 6.6: The effect of under- and oversampling compared to the normal extracted features training set of $F_{X\text{sim}}$ (larger dots indicate multiple instances in the set)

In the next few sections, we see the effect of under- or oversampling on different performance metrics.

6.3. Training a Random Forest classifier

6.3.1. Default Random Forest

To understand the training ability of a RF classifier for our datasets, we initiate the training by using the default parameter settings for the RF algorithm. In our case, we set the parameters according to Table 6.3.

Parameter	Setting
`max_depth`	`None`
`n_estimators`	100
`max_features`	$\sqrt{p}$
splitting rule	Gini
`min_samples_split`	2
`min_samples_leaf`	1

Table 6.3: Parameters in default setting

6.3.2. Tuned Random Forest

The defined grid for the tuning parameters of the $F_{X\text{sim}}$ dataset is outlined in Table 6.4. We maintain the `min_samples_split` and `min_samples_leaf` parameters the same as the default setting. The parameter grid already contains $4 \times 4 \times 3 \times 2 = 96$ combinations. Including an additional 4 options for both parameters would result in $96 \times 4 \times 4 = 1.536$ combinations, which can be computationally expensive to evaluate. Furthermore, also due to computational limitations, we have defined the grid differently for the EEG dataset, see Table 6.5. The EEG dataset consists of over 40.000 observations, making it computationally expensive to train an RF model 96 times with 100 to 250 trees. However, by trying out fewer parameter combinations, we have already observed that the model achieves impressive scores for the performance metrics.

Parameter	Grid
`max_depth`	[None, 3, 6, 9]
`n_estimators`	[100, 150, 200, 250]
`max_features`	[None, $\sqrt{p}$, 0.2]
splitting rule	Gini or entropy
`min_samples_split`	2
`min_samples_leaf`	1

Table 6.4: Parameter grid for tuning parameters $F_{X\text{sim}}$ set

Parameter	Grid
`max_depth`	[3, 6, 9]
`n_estimators`	[10, 20, 50]
`max_features`	[$\sqrt{p}$, 0.2]
splitting rule	Gini or entropy
`min_samples_split`	2
`min_samples_leaf`	1

Table 6.5: Parameter grid for tuning parameters Z_{sim} set

In the next section, we examine the difference between a default and tuned Random Forest classifier.

6.4. Evaluation

6.4.1. Quality

In our problem, we prioritized the performance metrics sensitivity and F-score, followed by accuracy. The scores for the performance metrics with a default RF for all three datasets are displayed in Table 6.6. In Appendix E, Figures E.5, E.6 and E.6 illustrate the difference between the predicted and true class labels of the test set observations associated with these scores.

Test set	Default RF						
Reduced $F_{X\text{sim}}$	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.5	0.333	0.667	0.5	0.571	0.5	0.583
Normal	0.857	1	0.333	1	0.5	0.667	0.9
Oversampled	0.681	1	0.364	1	0.533	0.682	1
EEG Z_{sim}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.833	0.667	1	0.75	0.857	0.833	1
Normal	1	1	1	1	1	1	0.999
Oversampled	0.727	1	0.455	1	0.625	0.727	1
PCA on EEG Z_{simPCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	1	1	1	1	1	1	0.983
Normal	0.929	1	0.667	1	0.8	0.833	0.972
Oversampled	0.727	0.909	0.545	0.857	0.667	0.727	0.997

Table 6.6: Scores performance metrics for reduced dataset ($F_{X\text{sim}}$), EEG (Z_{sim}) and PCA on EEG (Z_{simPCA}): default RF (in bold higher accuracy, sensitivity and F-score scores are emphasized)

Based on the observations from this table and its corresponding prediction figures, several notable findings emerge. Firstly, it is evident that the undersampled $F_{X\text{sim}}$ dataset demonstrates higher sensitivity than specificity. This implies that the model performs better in classifying prediction from the minority class, class 1, compared to class 0 within the test set. Conversely, the oversampled $F_{X\text{sim}}$ set exhibits the opposite pattern. These observations are commonly observed, as undersampling can result in loss of information from class 0, while oversampling can lead to duplicated observations from class 1, making it more challenging to generalize on the class 1 data. Another notable observation relates to the relatively low OOB score for the undersampled $F_{x\text{sim}}$ set. In certain instances, Python generated an error indicating that there were insufficient samples to provide an accurate OOB score. However, despite this limitation, the undersampled set still provides a good representation of the test accuracy score. In all three cases, the OOB score for the oversampled set does not adequately reflect the accuracy score. This may be due to the presence of duplicated observation from the minority class in the training set, which could be mistakenly utilized as OOB sampled while also being used for training a tree. This asks for further research on the implementation of the RF function in Python, `RandomForestClassifier()`.

Finally, an important observation is that the normal EEG set, Z_{sim}, outperforms all other sets and already shows optimal performance scores. In Figure E.6 in Appendix E, we notice that these high scores remain intact when tuning the RF. Also for the under- and oversampled EEG set Z_{sim}, tuning the classifier improves its performance. Recognizing the potential impact of a tuned RF classifier, we now compare all sets using a tuned RF. It is worth noting that the RF classifier is tuned individually for each dataset, meaning that each set has its own unique set of parameters as optimized configuration.

First, looking at the prediction differences between the default and tuned RF in Figures E.5, E.6 and E.7 of Appendix E, we see that all sets demonstrate either equal or improves scores, except for the undersampled and normal EEG PCA set. During the grid search for the EEG sets, the default parameters were not included. Therefore, it is not surprising that these particular sets exhibit lower scores compared to the default RF. It indicates those sets need more tuning on the parameters.

Having explored the difference between the default and tuned RF, we shift our focus to examining the performance of the individually tuned RF classifiers. The test scores for the tuned RFs are displayed in Table 6.7. The best parameter settings from the grid per dataset can be found in Appendix E. It is important to note that the test scores in the table were calculated using a separate test set that the model had not been exposed to previously. The table also displays the mean of validation sets of the cross-validation process. Although the validation sets used for the mean scores were not seen by the model during training, they were utilized to train the final model. The validation scores for the EEG dataset Z_{sim} are based on all epochs, so there is not yet a majority vote per patient established. The test set scores are based after taking the majority vote per patient. The scores for both datasets are displayed in Table 6.7.

Tuned RF	Mean of CV validation sets (in best setting)						
Reduced $F_{X\text{sim}}$	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.75	0.833	0.667	0.889	0.711	0.708	
Normal	0.933	1	0.667	0.667	0.667	1	
Oversampled	1	1	1	1	1	1	
	Separate test set						
Reduced $F_{X\text{sim}}$	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.667	0.667	0.667	0.667	0.667	0.667	0.75
Normal	0.929	1	0.667	1	0.8	0.833	1
Oversampled	0.818	1	0.636	1	0.778	0.818	1

Tuned RF	Mean of CV validation sets (in best setting)						
EEG Z_{sim}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.827	0.715	0.922	0.791	0.851	0.915	
Normal	0.766	0.894	0.519	0.725	0.598	0.761	
Oversampled	0.969	0.935	1	0.949	0.973	1	
	Separate test set						
EEG Z_{sim}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	1	1	1	1	1	1	0.978
Normal	1	1	1	1	1	1	0.978
Oversampled	1	1	1	1	1	1	1

Tuned RF	Mean of CV validation sets (in best setting)						
PCA on EEG Z_{simPCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.732	0.632	0.816	0.728	0.763	0.808	
Normal	0.843	0.951	0.470	0.750	0.551	0.792	
Oversampled	0.932	0.879	0.976	0.909	0.941	0.987	
	Separate test set						
PCA on EEG Z_{simPCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.833	0.667	0.75	0.333	0.857	0.833	0.876
Normal	0.929	1	0.667	1	0.8	0.833	0.937
Oversampled	0.955	0.909	1	0.917	0.957	0.955	0.969

Table 6.7: Scores performance metrics for reduced dataset ($F_{X\text{sim}}$), EEG (Z_{sim}) and PCA on EEG (Z_{simPCA}): tuned RF (in bold higher accuracy, sensitivity and F-score scores are emphasized)

For the $F_{X\text{sim}}$ set, we observe that the oversampled set has highest means of validations scores. Each validation set perfectly classifies all observations within its set. However, in the separate test set, oversampling exhibits lower scores than the normal set. It appears that the normal $F_{X\text{sim}}$ set better generalizes on unseen data.
Another special observation is the exceptional performance of all EEG Z_{sim} datasets, including the undersampled, normal and oversampled variants, when applied to their tuned RF. It suggest in this case, you can choose either set. In Figure E.6, we see however that for certain observations, the percentage of class 1 classified epochs is positioned near the boundary of 0.5. This indicates that not all epochs are classified correctly. Though the labels remained constant over all epochs, meaning that not all epochs are ought to contain the necessary information to classify a favourable or unfavourable outcome. Furthermore, in the table we do see lower mean scores for the normal EEG set. Examining the scores of all tuning parameters sets in Figure E.8, the sensitivity score only once exceeds a threshold of 0.7, while the sensitivity score in the test set is equal to 1. A good reason could be that the RF model used for the test set benefits from a larger number of training samples. This may contribute to a better performance compared to the RF model trained on 2-folds of the training set.

An intriguing conclusion that can be drawn is that the EEG PCA set exhibits poorer performance in all three scenarios (undersampled, normal and oversampled) when compared to the regular EEG set. Both sets' RFs are tuned on the same grid of parameters. Therefore, we cannot attribute the inferior performance of the EEG PCA set to a lack of access to better parameter settings. This is because PCA is a linear combination of data derived from the original EEG set, indicating that the potential for improved parameter settings exists for both sets.

Lastly, we mention the link between the OOB scores and the accuracy scores of the cross-validation sets. The OOB score is higher in all cases. Still in the case of our tuning, we set the maximum number of trees to 50 for the EEG sets. For a training set of 12 observations, if each bootstrapped set contains 12 samples, the probability that a specific observation occurs in less than 3 out of 50

6.4.2. Robustness

Cross-validation

The results of cross-validation on the entire $F_{X\text{sim}}$ and Z_{sim} set using the tuned RF can be found in Figure 6.7 and 6.8. The dots are the mean of the k scores and the upper- and lowerbound represent the maximum and minimum scores. We see that the normal Z_{sim} set can have a training and test set that on one hand reaches an F-score of value 1, but another fold reaches a score of 0. This shows that the tuned RF model perform differently on normal Z_{sim} sets that are slightly changed. In terms of sensitivity, precision and F-score, under- and oversampling the EEG set Z_{sim} ensures a smaller interval for validation scores. For the tuned RF for $F_{X\text{sim}}$ sets, we see mostly higher scores for the validation sets compared to Z_{sim}. The tuned RF for $F_{X\text{sim}}$ performs more constantly better on slightly different train and validation sets. Where the undersampled Z_{sim} set always appears to have enough samples from class 0 (specificity and precision of 1), the tuned RF of undersampled $F_{X\text{sim}}$ exhibits more uncertainty in classifying these class labels.

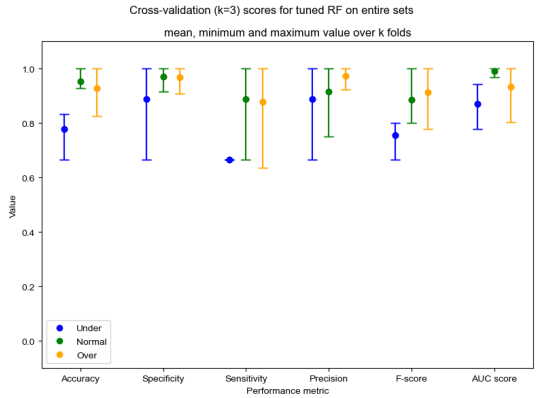


Figure 6.7: Cross-validation of default RF on entire $F_{X\text{sim}}$ set

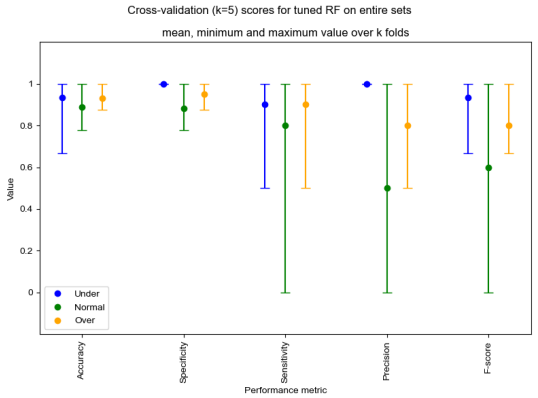
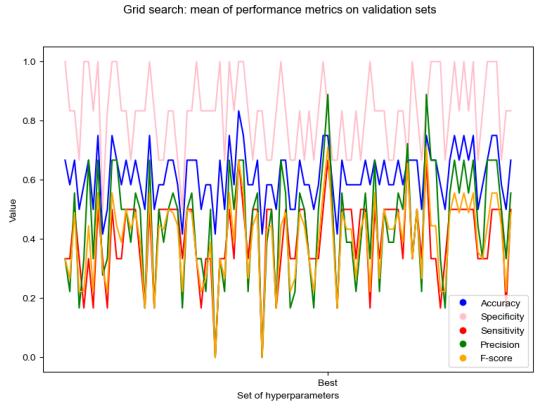


Figure 6.8: Cross-validation of tuned RF on entire Z_{sim} set

Parameter settings

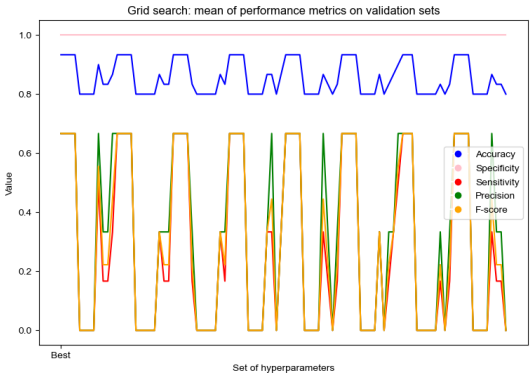
Figure 6.9 presents the mean scores of the three validation sets for each parameter setting. The primary goal is to identify the parameter set that yields the highest mean F-score. In the event of multiple maximum values, the method proceeds to search for other performance metrics' scores. Table 6.7 compares the maximum mean scores for the best parameter setting and the scores corresponding to the performance on the test set, which was unseen by the RF model during training. Figure 6.9 displays the mean scores of validation sets across various tuning parameter settings. We see that the scores of the RF model for undersampled dataset are more chaotic than the other sets. This reasoning seems logical because RF models trained on a smaller dataset may encounter challenges in generalization. With limited training data, different parameter settings can have a significant influence on their behavior. We can also tell that for the normal dataset the classifier has an accuracy never below 0.8, which is essentially the class proportions. Furthermore, the parameter settings have in the case of $F_{X\text{sim}}$ no influence on the specificity score, while it has an effect on the sensitivity and precision. This suggests that the training data contains sufficient amount of information on class labels 0. In the

oversampling case, we see that the scores drop down for some settings. It shows that it is essential to choose the right parameters for a qualified model. However, in case of oversampling, the RF classifier do not reach scores under 0.92 for these parameter sets, making it an attractive set to train any RF model on. Appendix E contains Figure E.8 with the grid search results of the EEG dataset Z_{sim} and Z_{simPCA}. In the normal sets of both Z_{sim} and Z_{simPCA}, we notice a relatively constant value for specificity which is a result of the class imbalance. The sets contain sufficient observations of class 0 to correctly classify these class labels. In the under- and oversampled sets, sensitivity has higher values than specificity, indicating the opposite for these sets: there is enough information on class labels 1. Figure E.3 and E.4 support this by the observation that both the undersampled and oversampled sets have a higher number of samples with class label 1 compared to label 0. This can attributed to the varying number of EEG epochs across different patients, where it is important to remember that under- and oversampling happens on patient-level and not on epoch-level. Once again, the minimum mean scores of the oversampled EEG set consistently remain above 0.8 indicating a high level of performance. In addition to this, the test scores were also optimal, as seen before.



(a)) Grid search for under $F_{X\,\text{sim}}$ set

(b)) Grid search for over $F_{X\,\text{sim}}$ set



(c)) Grid search for normal $F_{X\,\text{sim}}$ set

Figure 6.9: Different parameter settings for $F_{X\,\text{sim}}$ set

6.4.3. Interpretability

Mean Decrease in Impurity

In the figure below, we see that according to Mean Decrease in Impurity (MDI), `feature 2 mean` is the most prominent feature, followed by `feature 2 max`. This aligns with the definitions of our simulated data. Besides the second feature, features 24, 22, 15, 25 and 12 are incorporated in the feature importance graph and apparently ensure more pure splits in the decision trees. Only feature 24 was also made class dependent in the definition of our dataset. This indicates that besides considering the right features as important in determining the class label, the RF classifier assigns importance to 3 redundant features.

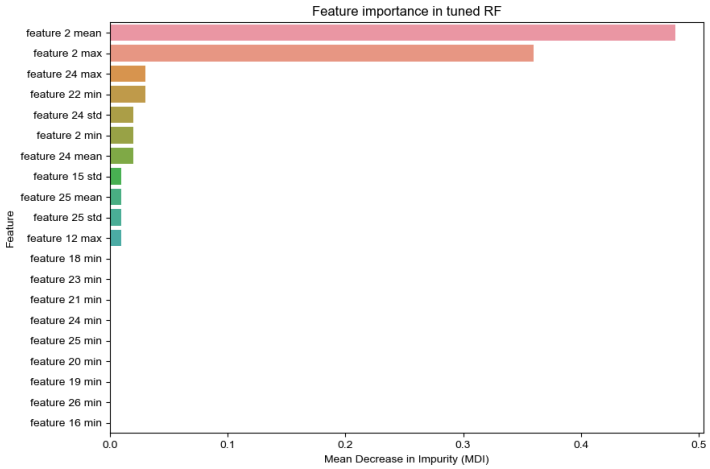


Figure 6.10: Feature importance in tuned RF for $F_{X\,\text{sim}}$

SHAP values

The SHAP values for the $F_{X\,\text{sim}}$ set in the tuned RF are displayed in Figure 6.11. We only see `feature 2 mean` and `feature 2 max` present in the figure, which suggests that SHAP demonstrates better filtering capabilities of class-dependent features, distinguishing them from random features more effectively. Both classes use the two features equally. This might indicate the confusion rate is high. Ideally, features are uniquely dedicated to one class. However, Figure 6.12 shows that there is a certain threshold for feature `feature 2 mean` according to the SHAP values.

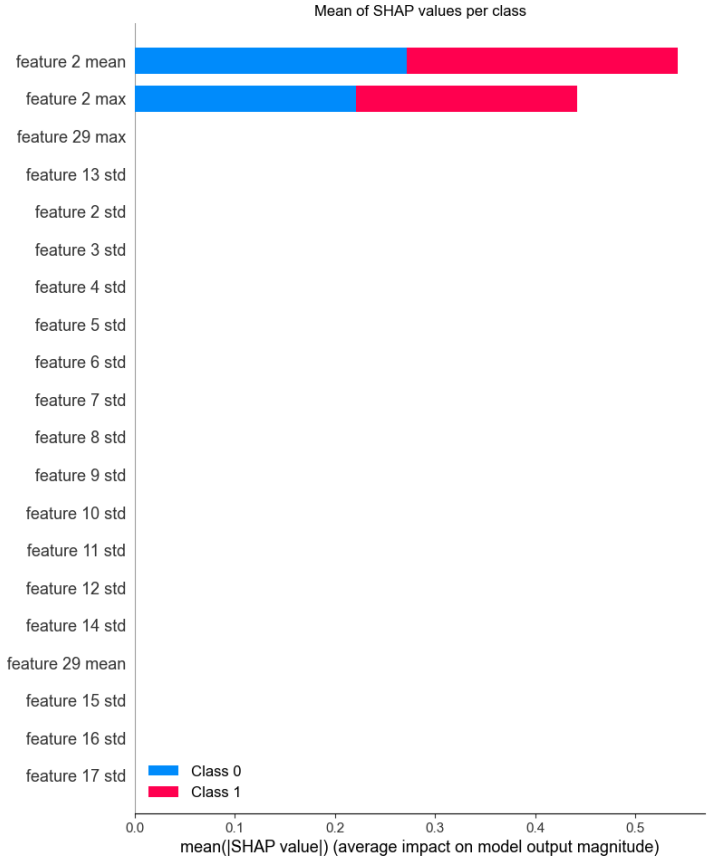


Figure 6.11: Mean of SHAP values of normal $F_{X\,\text{sim}}$: tuned RF

The figure is based on the class labeled as 1. Negative values for `feature 2 mean` receive positive

SHAP values, meaning the values are associated with class label 1. High values of the feature indicate class label 1. These values receive positive SHAP values which push `.predict_proba` closer to 1, indicating a high probability for class label 1. If you would plot the values based on class label 0, negative feature values push the observation to a higher probability for class label 0. This is because the SHAP values are symmetric. The threshold corresponds to Figure 6.6 where we have plotted the training observations of $F_{X\text{sim}}$.

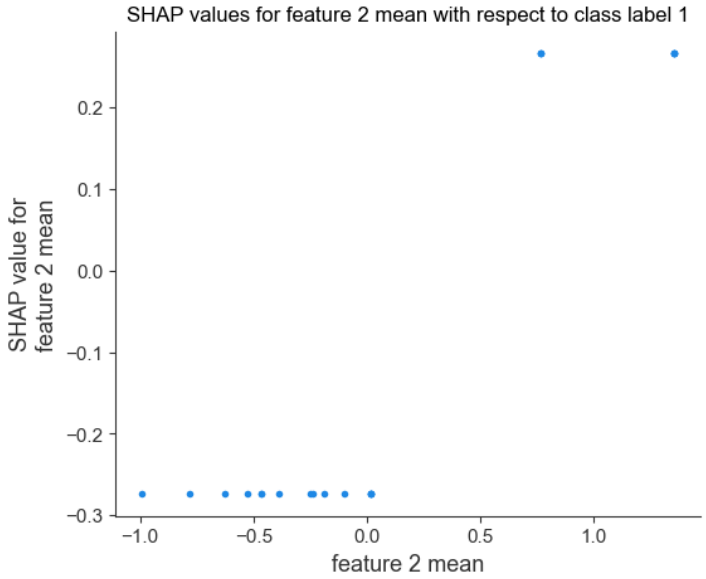


Figure 6.12: Threshold for feature 2 mean based SHAP values: oversampled $F_{X\text{sim}}$

We now use observation 28 as an example, which belongs to the class with label 1. In Figure E.5, patient 28 was misclassified for all three sets. With SHAP values we gain insights into the factors that led to this misclassification. The expected values for class 0 and 1 are 0.209 and 0.791 resp. In Figure **??**, we see that `feature 2 mean` and `feature 2 max` push the prediction of patient 28 away from the value of 0.791 to a predicted probability of 1. On the y-axis, the value of `feature 2 mean` for patient 28 is displayed and shows that the value is close to the threshold of Figure 6.12.

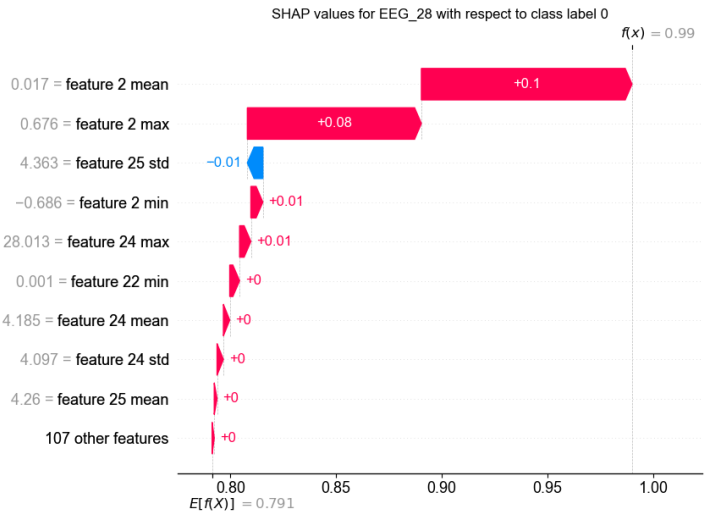


Figure 6.13: SHAP values for patient 28 in $F_{X\text{sim}}$

7

Results

This chapter provides a comprehensive description of the results obtained from our study. The analyses in this chapter elaborate on the methodology described in Chapter 5. The data is discussed in Chapter 3 and the classification method in Chapter 4. We continue this chapter with the following steps: data preprocessing, model development and evaluation.

7.1. Preprocessing

Before initiating the training process of the classifier, we undertake several data preprocessing steps as outlined in Chapter 5. PCA is performed early in the process, since it then has the ability to leverage all variables in order to compute covariances and identify the features that contribute the most to the overall variance within the dataset.

7.1.1. Principal Component Analysis

In the case of the data of anti-NMDARE, there are multiple set on which we can perform PCA, either on Z or F_X, see Equations (3.3) and (3.15). For the EEG dataset Z, we apply PCA to assign weights to the relevant features. Due to the fact that the number of features exceed the number of observations in the F_X dataset, we choose to not perform PCA on this dataset. Performing PCA on F_X may not represent the data well, as the principal components can only consists of the first I features. The set F_X suffers from a problem where the number of features exceeds the number of observations. Other feature selection methods may be better suitable.

We run PCA on the entire matrix Z of Equation (3.3) where the 29 columns of Z are reduced to k columns such that the explained variance is > 90%.

$$Z_{PCA1} = \begin{bmatrix} z_{1,PC1} & z_{1,PC2} & \cdots & z_{1,PCk} \\ z_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ z_{46832,PC1} & \cdots & \cdots & z_{46832,PCk} \end{bmatrix} \tag{7.1}$$

If the PCA is run on the dataset, the number of principal components, p, is 13 such that ex. var. > 90%, see scree plot in Figure 7.1.

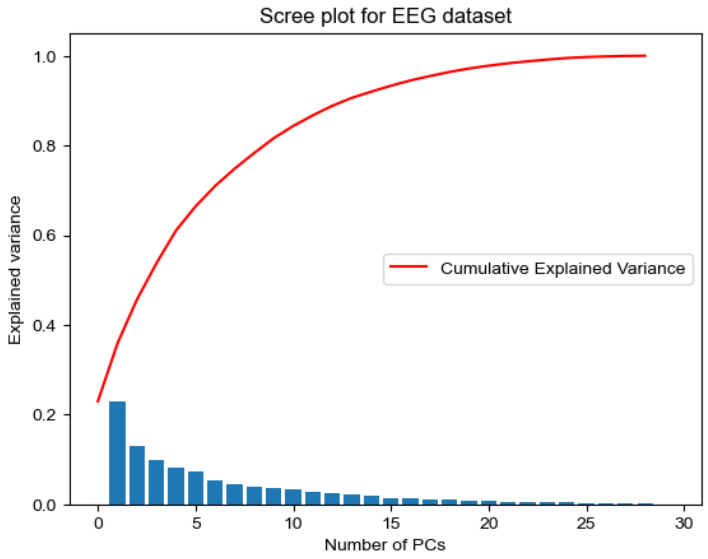


Figure 7.1: Scree plot for matrix Z

The scatterplots of the first two principal components shows there is no clear distinction between the two classes. Adding the third principal component has minimal effect on this observation.

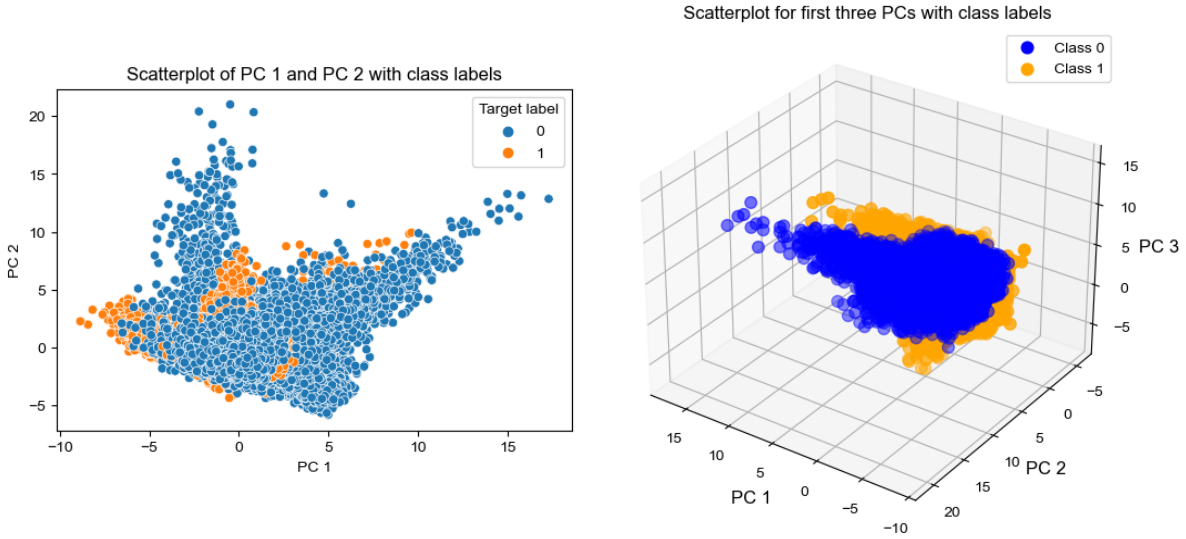


Figure 7.2: Scatterplots of the first two and three PCs with class labels

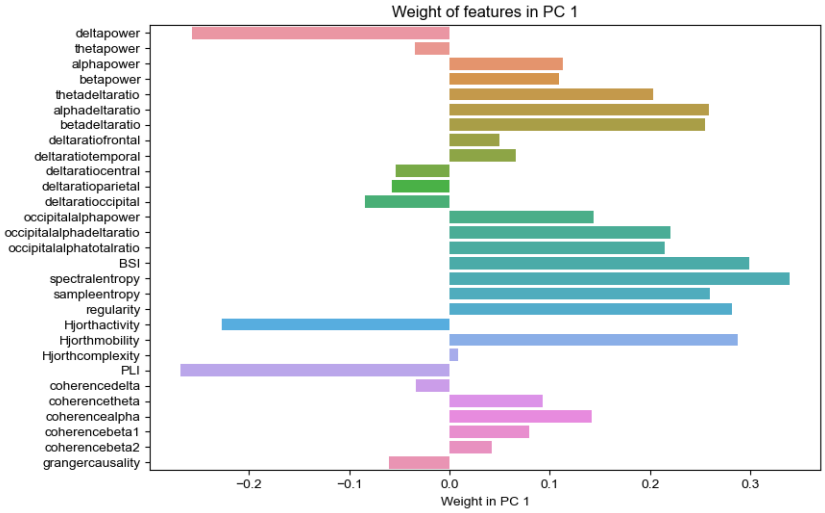
In the case of anti-NMDARE, the features `BSI`, `spectralentropy`, `sampleentropy`, `regularity`, `Hjorthmobility` and `PLI` had high absolute values of covariance in S, see Figure 3.7 in Chapter 3. Also the features `occipitalalphapower`, `occipitalalphadeltaratio` and `occipitalalphatotalratio` were highly correlated with each other. The presumption is that the coefficients for these feature variables receive a high value in ϕ_1. Plotting the weight per feature in the first principal component, the presumption that certain correlated features receive a high weight is confirmed. The weights of the five features that receive the most absolute weights in PC 1 and 2 are displayed in Tables 7.1 and 7.2

Feature	Weight
spectralentropy	0.399
BSI	0.299
Hjorthmobility	0.288
regularity	0.282
PLI	−0.268

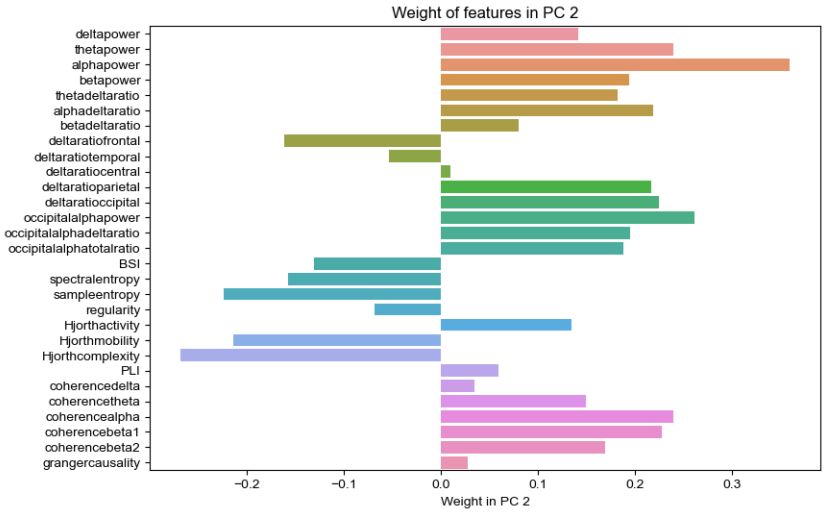
Table 7.1: Features with most absolute weight in PC 1

Feature	Weight
alphapower	0.360
Hjorthcomplexity	−0.268
occipitalalphapower	0.262
thetapower	0.240
coherencealpha	0.240

Table 7.2: Features with most absolute weight in PC 2



(a)) In PC 1



(b)) In PC 2

Figure 7.3: Weight per feature in the first two principal components

7.1.2. Train and test split

Figure 7.4 illustrates the train and test split. The set F_X is split in a train ($F_{X\text{train}}$) and test ($F_{X\text{test}}$) set of size 30 and 14 resp.. The split is done with stratification, such that each set has equal representation of labels 0 and 1. Furthermore, the EEG dataset, denoted as Z, is divided based on the patients, resulting in a split of 30 versus 14 as well. Each patient's epochs are included in either the training (Z_{train}) or test (Z_{test}) set.

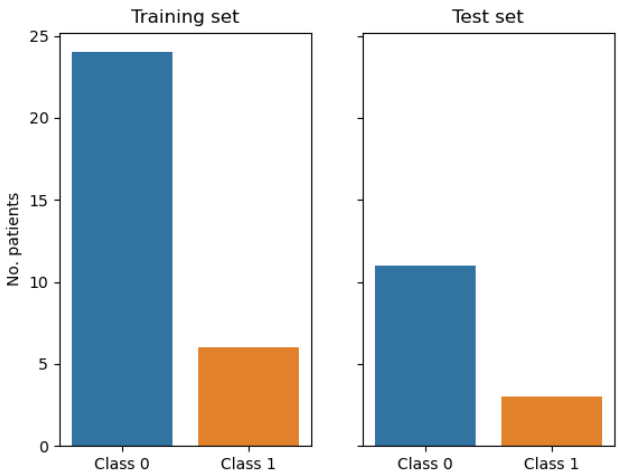


Figure 7.4: Train and test split

7.1.3. Class imbalance

The effect of under- and oversampling the F_X training set is depicted in Figure 7.5. We chose to perform under- and oversampling after splitting the data to ensure differentiation between the training and test set. The effects on the EEG training set and the EEG PCA training set is displayed in Appendix F. Visually, it appears that for the undersampled set of F_X it is most easy to make a distinction between classes 0 and 1, as you can draw a non-linear boundary between the observation.

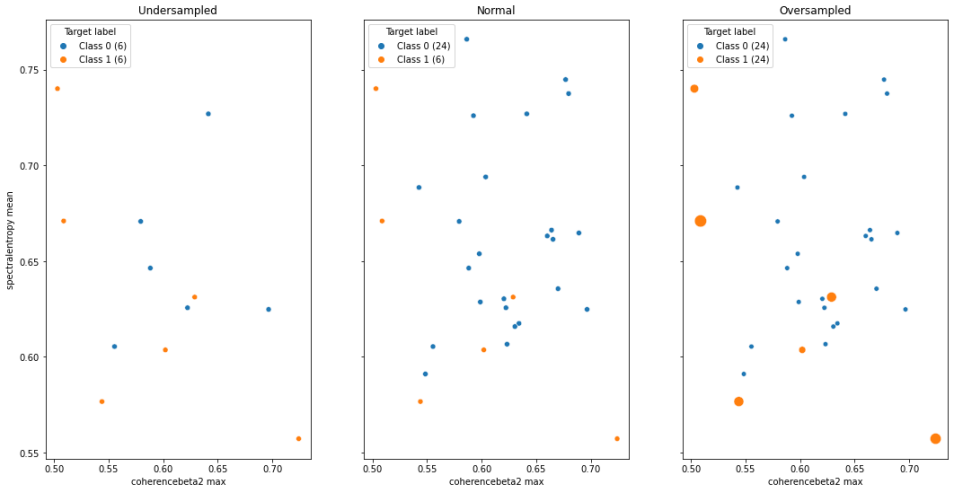


Figure 7.5: Effect of under- and oversampling for the F_X training set

7.2. Training a Random Forest classifier

7.2.1. Default Random Forest

A simple RF model is fit on each training set X_{train} and y_{train} with the same default parameters as in Table 6.3. To serve as a reminder of the parameters, the table is displayed below, Table 7.3

Parameter	Setting
`max_depth`	`None`
`n_estimators`	100
`max_features`	$\sqrt{p}$
splitting rule	Gini
`min_samples_split`	2
`min_samples_leaf`	1

Table 7.3: Parameters in default setting

7.2.2. Tuned Random Forest

Our focus shifts to tuning the model's parameters with the aim is to achieve improved results on the test sets. We performed `GridSearchCV()` with the same grid as in the simulation study of Chapter 6, displayed in Tables 7.4 and 7.5.

Parameter	Grid
`max_depth`	$[\texttt{None}, 3, 6, 9]$
`n_estimators`	$[100, 150, 200, 250]$
`max_features`	$[\texttt{None}, \sqrt{p}, 0.2]$
splitting rule	Gini or entropy
`min_samples_split`	2
`min_samples_leaf`	1

Table 7.4: Parameter grid for the tuning parameters F_X set

Parameter	Grid
`max_depth`	$[3, 6, 9]$
`n_estimators`	$[10, 20, 50]$
`max_features`	$[\sqrt{p}, 0.2]$
splitting rule	Gini or entropy
`min_samples_split`	2
`min_samples_leaf`	1

Table 7.5: Parameter grid for the tuning parameters Z set

We used $k = 3$ for the cross validation sets, meaning the training sets ($F_{X\text{train}}$ and Z_{train}) are split into 3 sets. Each time a fraction, $\frac{2}{3}$, of the sets are used as training set and $\frac{1}{3}$ as validation set. By setting $k = 3$, all validations sets have at least one observation with class label 1. Finally, the test sets ($F_{X_{\text{test}}}$ and Z_{test}) are predicted on the model with the best found parameters.

7.3. Evaluation

The default and tuned RF models are evaluated in the this section. Before we begin with the analysis, we present a random tree from the Random Forest Classifier. As the Random Forest classifier is an ensemble method of multiple DTs, we can visualize each individual tree. For the reduced dataset F_X, a random classification tree is displayed in Figure 7.6.

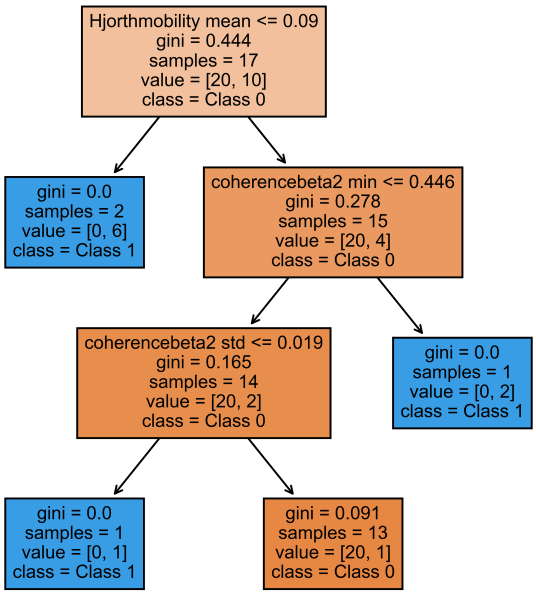


Figure 7.6: A tuned tree in the Random Forest classifier for F_X

Figure 7.7 presents a random DT applied to the EEG dataset Z. Unfortunately, due to its large size, the entire tree cannot be displayed in high quality within this paper. Figure 7.8 shows a split of the tuned tree, showing that the value 0.013 of `betadeltaratio` splits the parent node into two child nodes of size 2078 and 681. The number of unique samples in the node do not equal the sum of these values since we use bootstrap sampling with replacement so that not all values are unique. It is worth noting that the tree for set Z is larger compared to the one for F_X. This can likely be attributed to the presence of a greater number of observations and fewer features within set Z.

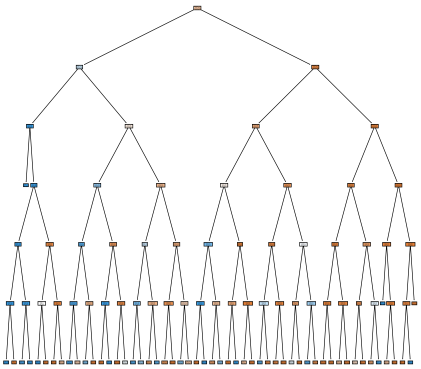


Figure 7.7: A tuned tree in the Random Forest classifier for Z

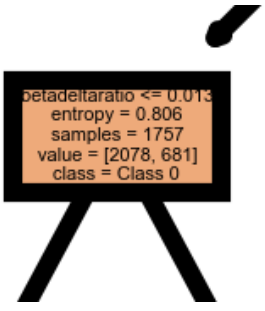


Figure 7.8: A split in the tuned tree in the Random Forest classifier for Z

7.3.1. Quality

For the default RF models trained on normal, undersampled and oversampled F_X and Z sets, we compare the scores on the test sets in Table 7.6. In our research, we prioritized the performance metrics sensitivity and F-score, followed by accuracy. The sensitivity score from the table tells us how well the model can classify observations from the class with label 1. A score of 0 unfortunately means the model is not capable of classifying the observations correctly. The accuracy score for undersampled F_X is emphasized even though it is lower than the accuracy score of the normal F_X set. This is due to the dataset being undersampled, an accuracy score of 0.5 serves as the 'baseline' accuracy for a simplistic classifier. By referring to a simplistic classifier, we imply that the classifier assigns all observations to one particular class. A score of 0.667 indicates it improves the simplistic classifier. Whereas the baseline accuracy score for the normal set is 0.786. The oversampled dataset of F_X are even below their baseline accuracy. The classifier for the oversampled set also has a sensitivity of 0, indicating that the classifier fails to correctly classify any observation from class 0. One notable advantage of the oversampled F_X set is the high OOB score. According to Breiman, the OOB score serves as a reliable estimate of the error for the test set which shares the same size as the training set [36]. An explanation for the higher OOB score can be that the OOB samples of the oversampled dataset contain more observation than the ones of the undersampled and normal F_X set. The predicted versus true labels of all F_X test set observations can be found in Appendix F, Figure F.3.

Test set	Default RF						
Reduced F_X	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.667	1	0.333	1	0.5	0.667	0.417
Normal	0.786	1	0	0	0	0.5	0.8
Oversampled	0.455	0.909	0	0	0	0.454	0.958
EEG Z	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.667	0.667	0.667	0.667	0.667	0.667	0.997
Normal	0.786	1	0	0	0	0.5	0.990
Oversampled	0.5	1	0	0	0	0.5	0.999
PCA on EEG Z_{PCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.5	0.667	**0.333**	0.5	**0.4**	0.5	**0.951**
Normal	0.786	1	0	0	0	0.5	0.947
Oversampled	0.5	1	0	0	0	0.5	**0.978**

Table 7.6: Scores performance metrics for reduced dataset (F_X), EEG (Z) and PCA on EEG (Z_{PCA}): default RF (in bold higher accuracy, sensitivity, F-score and OOB scores are emphasized)

Undersampling the EEG dataset has a positive effect in the case of a default RF. Visualizing the predicted versus true labels, this statement is verified, see Figure 7.9. The model is able to correctly classify two observations (`EEG_09_06` and `EEG_33_01`) from class 1, namely we see a sensitivity score of 0.667. However, the classifier also fails to correctly classify an observation (`EEG_41_03`) from class 0.

Figure 7.9: Predicted versus true class labels of the undersampled EEG test set ($\mathcal{Z}$): default RF

The predicted versus true labels of the other EEG and EEG PCA test set observations as determined by a default RF model can be found in Appendix F, Figures F.4 and F.5.

The test scores for the tuned RF models are presented in Table 7.7, together with the mean of the three validation scores. The mean of the three validation scores decide on a best configuration for the RF model. The CV mean scores for the EEG datasets are measured over all epochs, while the test set scores use the majority vote of all classified epochs.

Tuned RF	Mean of CV validation sets (in best setting)						
Reduced F_X	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.833	0.833	0.833	0.833	0.883	0.75	
Normal	0.8	0.958	0.167	0.333	0.222	0.771	
Oversampled	0.938	0.875	1	0.889	0.941	0.990	
	Separate test set						
Reduced F_X	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.5	1	0	0	0	0.5	0.417
Normal	0.786	1	0	0	0	0.5	0.8
Oversampled	0.455	0.909	0	0	0	0.455	0.917

Tuned RF	Mean of CV validation sets (in best setting)						
EEG Z	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.756	0.559	0.850	0.817	0.830	0.688	
Normal	0.766	0.894	0.519	0.725	0.598	0.761	
Oversampled	0.922	0.785	0.999	0.895	0.943	0.989	
	Separate test set						
EEG Z	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.667	0.667	0.667	0.667	0.667	0.667	0.971
Normal	0.786	1	0	0	0	0.5	0.938
Oversampled	0.818	1	0.636	1	0.778	0.818	0.978

Tuned RF	Mean of CV validation sets (in best setting)						
PCA on EEG Z_{PCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	
Undersampled	0.651	0.06	0.940	0.671	0.783	0.607	
Normal	0.802	0.907	0.601	0.779	0.678	0.790	
Oversampled	0.894	0.794	0.952	0.890	0.919	0.952	
	Separate test set						
PCA on EEG Z_{PCA}	Acc	Spec	Sens	Pre	F-score	ROC-AUC	OOB score
Undersampled	0.333	0.333	0.333	0.333	0.333	0.333	0.799
Normal	0.786	1	0	0	0	0.5	0.889
Oversampled	0.5	1	0	0	0	0.5	0.933

Table 7.7: Scores performance metrics for reduced dataset (F_X), EEG (Z) and PCA on EEG (Z_{PCA}): tuned RFs (in bold higher accuracy, sensitivity and F-score scores are emphasized)

First, we notice that only the tuned RF model for the oversampled EEG set Z performs better than the default configuration. Remarkably, the model for the undersampled F_X set even performs slightly worse compared to performance in the default setting. In both the default and tuned setting, the PCA datasets have less favorable test scores than the original EEG sets.

In Table 7.7, the mean of validation sets of the oversampled EEG set Z exhibit high scores. It seems that the model is correctly classifying the EEG epochs with label 1, even more than the ones with label 0: sensitivity is higher than specificity. Luckily, the test scores are also high, indicating that the model is able to generalize on unseen data. However, we observe an alteration in both the specificity and sensitivity scores. On the test set, the tuned model is better in classifying observations with label 0 than label 1. However, sensitivity for the oversampled EEG test set (Z) is still favorable compared to other datasets.
We notice that all models for the normal datasets misclassify all observations with class label 1, leading to its baseline accuracy score and 0 values for sensitivity and precision. So whether F_X, Z or Z_{PCA} is used as input data, the scores suggest under- or oversampling is advised. The undersampled F_X also attains its baseline performance scores. Figure 7.10 shows the predicted and true class labels for the

oversampled F_X test set. The lower specificity and precision can be attributed to the misclassification of an observation with class label 0. The high scores in the mean of CV validation sets of the oversampled F_X is a sign of overfitting. The oversampled set contain multiple replicates of observations with label 1. The training set predominantly learns from the patterns within these observations, thereby making it more challenging to generalize to unseen data.

Figure 7.10: Predicted versus true labels of oversampled F_X test set

This outcome is again suboptimal and, in fact, quite alarming. However, this finding is not surprising. The scores indicate that the RF model has a hard time classifying observations from the minority class with label 1. Based on the table, the under- and oversampled EEG sets Z display the most promising results.

To check that the grid search was performed well, we check a larger range per parameter, keeping the other parameters in the best setting, to see if the largest mean of the validation sets was chosen. Trying out a more detailed grid search for `max_depth` shows us we can reach multiple maximum `mean_test_F_score`'s of 0.222 for the normal dataset, see Figure 7.11, but after testing the parameters we conclude that they do not increase our final test score.

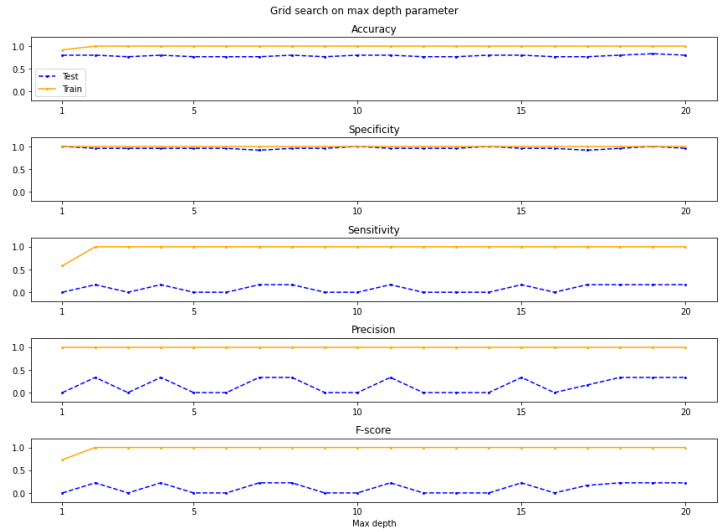


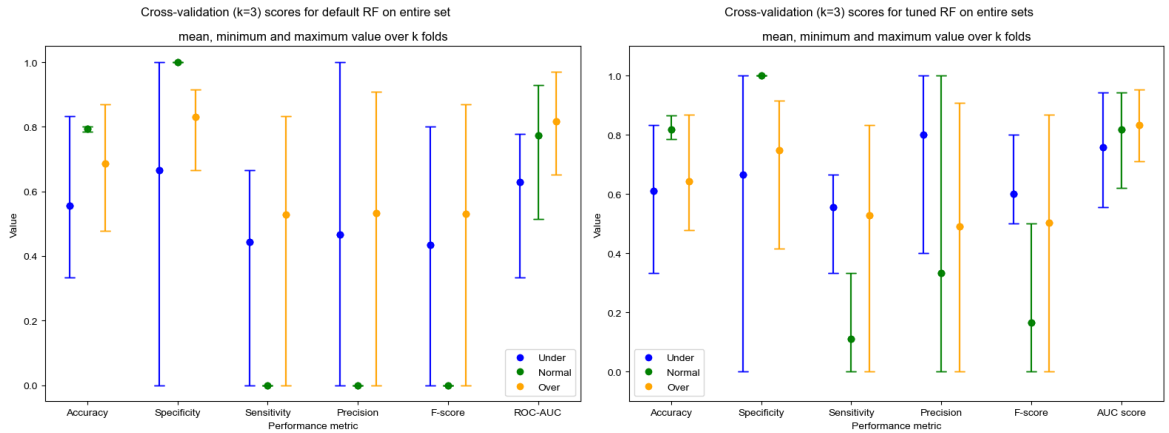
Figure 7.11: Grid search on parameter max depth for normal F_X dataset: blue line represents the mean scores for the validation sets and yellow line for the training sets.

Using $\sqrt{p}$ instead of `None` also gives us the same results as from the table. Looking at the grid search on `max_depth` for the undersampled dataset F_X, we see that a value of 10 gives us a higher `mean_test_F_score`. However, trying this new max depth with the same settings, does not give us a higher F-score for the test set. In Appendix, the plots for more datasets and their grid on the tuning parameters can be found to confirm the best parameter settings.

7.3.2. Robustness

Cross-validation

We performed 3-fold cross-validation on the entire set of under- normal, and oversampled F_X, see the scores in Figure 7.12a). In a 3-fold CV the sets are split into 3 sets of which 2 are used for training and 1 is used for validation. The splits are stratified which means each fold contain the same amount of observations with class labels 1.



(a)) On default RF

(b)) On tuned RF

Figure 7.12: 3-fold cross-validation on sets F_X. The dot represents the mean of the three validation scores, the upperbound is the maximum score reached and the lowerbound is the minimum score.

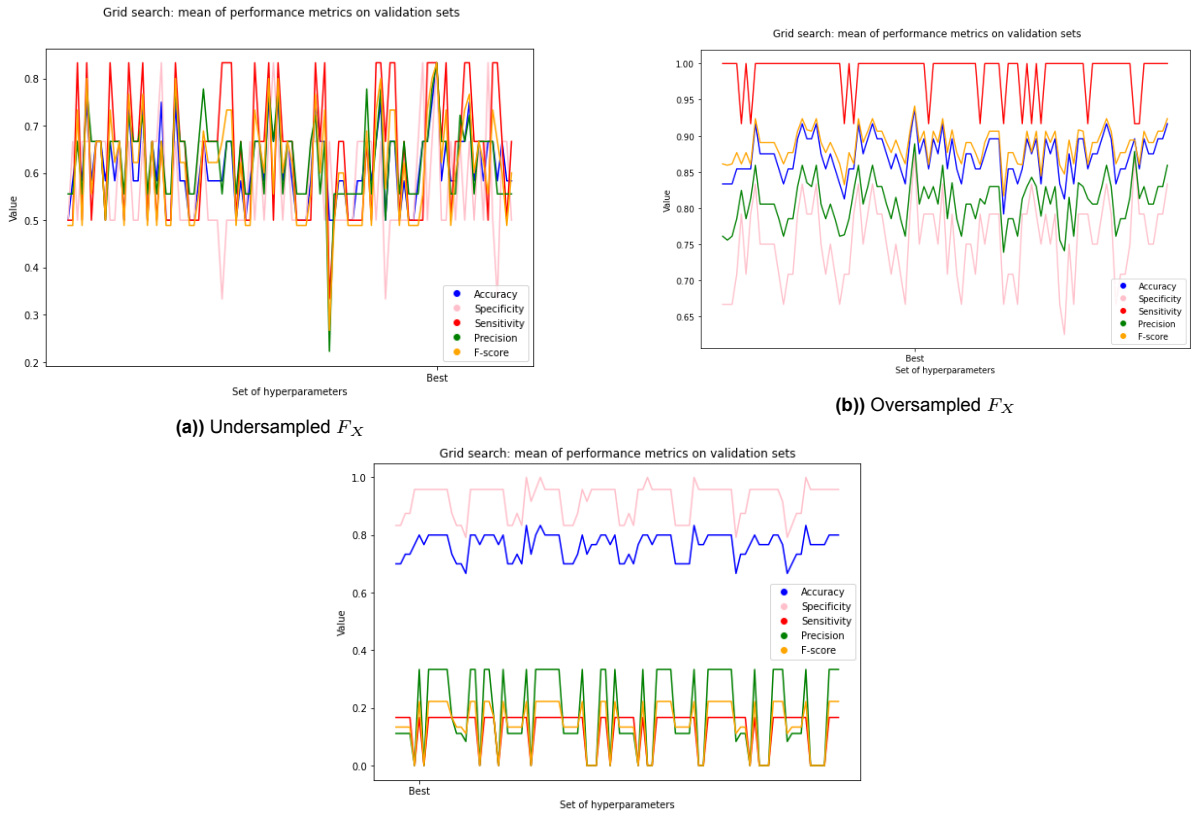
The normal F_X correctly classifies observations with class label 0 in all three folds, as the specificity score is constantly 1.

Comparing the default and tuned RF models, we see that the biggest improvement is for the normal F_X set, enhancing its sensitivity, precision and F-score. We also notice that in most cases, the interval

of the scores become smaller for the tuned RF model. This suggests that the model is better able to centralize the scores of the three folds.

Parameter settings

During the tuning phase of the RF classifier, we tried different settings of the tuning parameters via grid search. For the F_X sets, the following scores of the performance metrics are observed during the grid search, Figure 7.13.



(a)) Undersampled F_X

(b)) Oversampled F_X

(c)) Normal F_X

Figure 7.13: Grid search on tuning parameters RF for undersampled, normal and oversampled F_X

The grid search of the normal F_X set shows that the specificity, precision and F-scores never reach a value of 0.4 or higher. The oversampled set of F_X consistently exhibits the higher scores among the three sets with a sensitivity score of almost constantly 1. The more disappointing are the low test scores of the oversampled F_X set. The grid searches for the EEG Z and PCA Z_{PCA} sets can be found in Appendix F. There is pattern visable, where the set consistently performs worse but also that specificity and sensitivity behave in the opposite way. The figures of these datasets reveal a noteworthy characteristic: they exhibit greater stability compared to the figures of F_X. This phenomenon can be attributed to the larger number of observations present in the datasets. The reason why low test scores are still observed in Table 7.7 is the majority voting process after the grid search. This approach, which enables the classification of patients, further reduces the dataset to 44 observations. Another noteworthy characteristics are the sensitivity scores of the EEG and EEG PCA datasets compared to the specificity scores. This observation suggests that most of the class 1 epochs are correctly classified by the undersampled and oversampled sets, while the class 0 epochs seem more difficult to classify (due to lower specificity). In Figure **??** of Appendix F, an imbalance arises in the number of EEG epochs. While there is a balance between patients with label 0 and 1, the unequal distribution of EEG epochs among patients introduces an new form of imbalance and asks for further research. The high mean scores are in contrast to the obtained test scores. Furthermore, relying on the majority vote across epochs to classify a patient has its limitations. This approach fails to account for the possibility

that critical information may be concentrated in only a few epochs, while the remaining epochs can be regarded as noise.

7.3.3. Interpretability

The scores provide limited insights into the performance of the models. Our interest lies in determining the features that drive these models. In Chapter 3, we have seen that certain features are highly correlated with the class label. Furthermore, we showed that features significantly differ in their mean or variance between the classes. At the outset of this chapter, PCA assigned the most weight to the features `spectralentropy`, `BSI`, `Hjorthmobility`, `regularity` and `PLI`, which are coincidentally also the features with high correlation with the target label. Correlation with the target label does not necessarily indicate high values in the PCA approach.
During the evaluation of the RF models, we anticipate on the previously mentioned features, as well as features `coherencealpha`, `betadeltaratio`, `occipitalalphadeltaratio`, `coherencetheta`, `alphadeltaratio`, `deltapower` and `sampleentropy`.

Mean Decrease Impurity

For the MDI feature importance, we compare the undersampled set F_X and EEG Z for the tuned RF. The x-axes of Figures 7.14 and 7.15 display a different range making it hard to compare the feature importance of the two sets. This makes us aware of the disadvantage that the feature dimension of F_X is four times as large as the dimension of Z. However, the feature importance plot may be used as a tool for future feature selection. Furthermore, it is interesting to observe the same phenomenon as observed in the correlation matrices. The features related to the connectivity domain become important when taking the mean, minimum or maximum. This is also noticed comparing the feature importance figures for the normal and oversampled F_X and S sets, Figure F.7 in Appendix F.

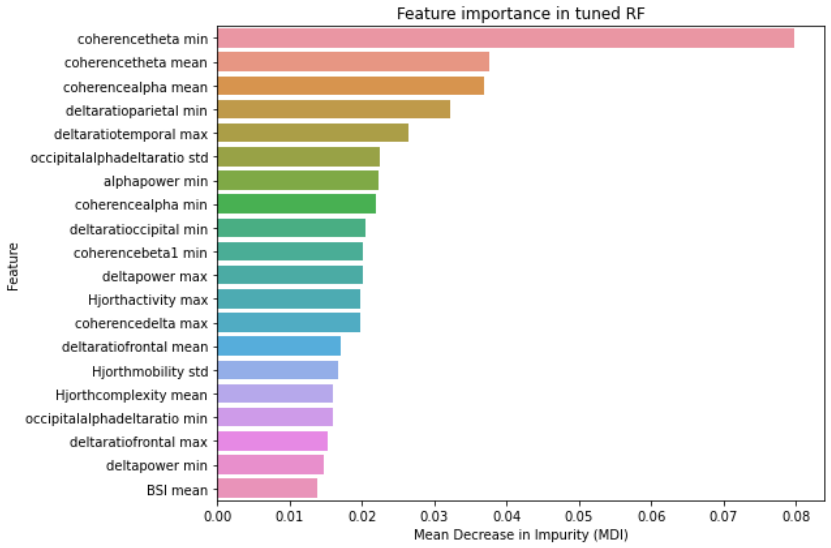


Figure 7.14: Feature importance based on Mean Decrease Impurity: Undersampled F_X

The feature that was mostly correlated with class label was `sampleentropy std` which is not included in the feature importance plot of undersampled F_X. The feature however is considered more important for the normal and oversampled sets. For the normal and oversampled EEG set Z, the tuned RF assigns `sampleentropy` the most MDI. This means that a split made on `sampleentropy` on average ensures the most pure splits. Furthermore, in all three cases for Z, `betapower` and `occipital`-related features support the splitting process by creating more pure nodes according to the MDI score.
Another notable observation in the cases of F_X sets is that the features that exhibited statistically significant differences between class labels are not well represented in the feature importance plots. Only the features `sampleentropy std`, `coherencetheta min` and `deltapower max` are in the top 15 features over the three F_X sets. Because of the large number of features (116), analyzing the MDI for each individual feature becomes challenging.

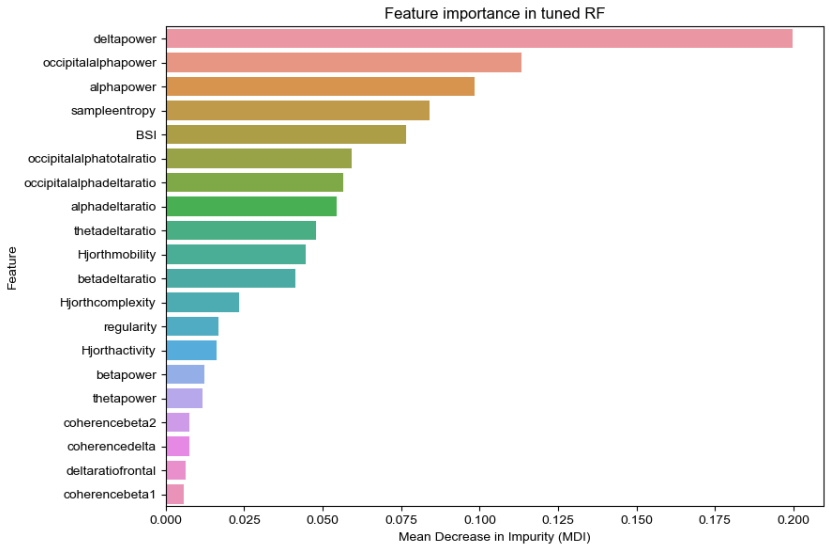


Figure 7.15: Feature importance based on Mean Decrease Impurity: Undersampled Z

SHAP

In the model development phase, we have seen that the undersampled and oversampled dataset of Z had the most promising results. In both sets and the normal EEG set Z, patient `EEG_13_02` was misclassified as class 0, while the patient has label 1. A big advantage of SHAP algorithm is that the SHAP values can tell us why a specific observation is misclassified. The values are namely calculated per observation. However, due to the majority vote applied to all EEG epochs, it becomes difficult to determine the specific features that contribute to this misclassification. It is unclear how the clustering affects the SHAP algorithm. In Figure 7.16, the SHAP values of a randomly chosen EEG epoch of the normal EEG set Z are displayed. The epoch is classified as class 0. This is primarily accomplished by feature `sampleentropy`.

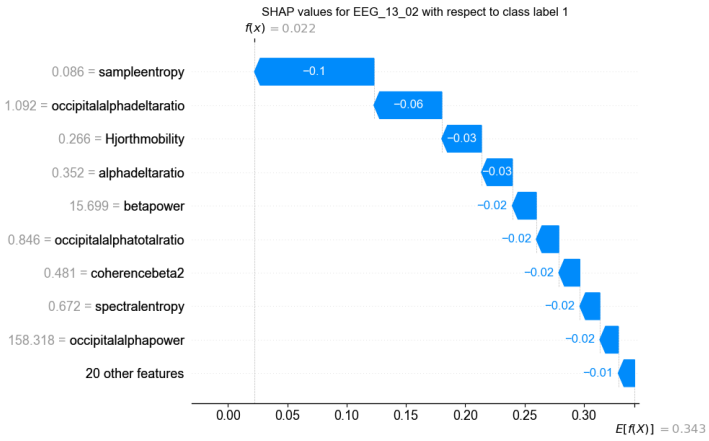


Figure 7.16: SHAP values of a random epoch

Looking at the mean SHAP values for the normal EEG Z set, we notice prominent importance of the same feature, `sampleentropy` with a SHAP value almost three times the second highest SHAP value.

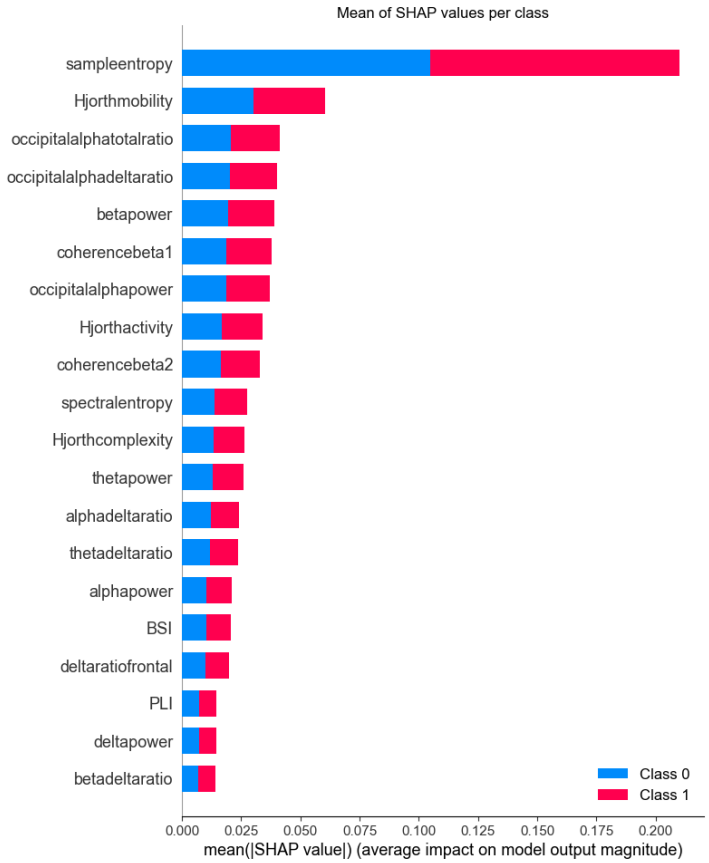
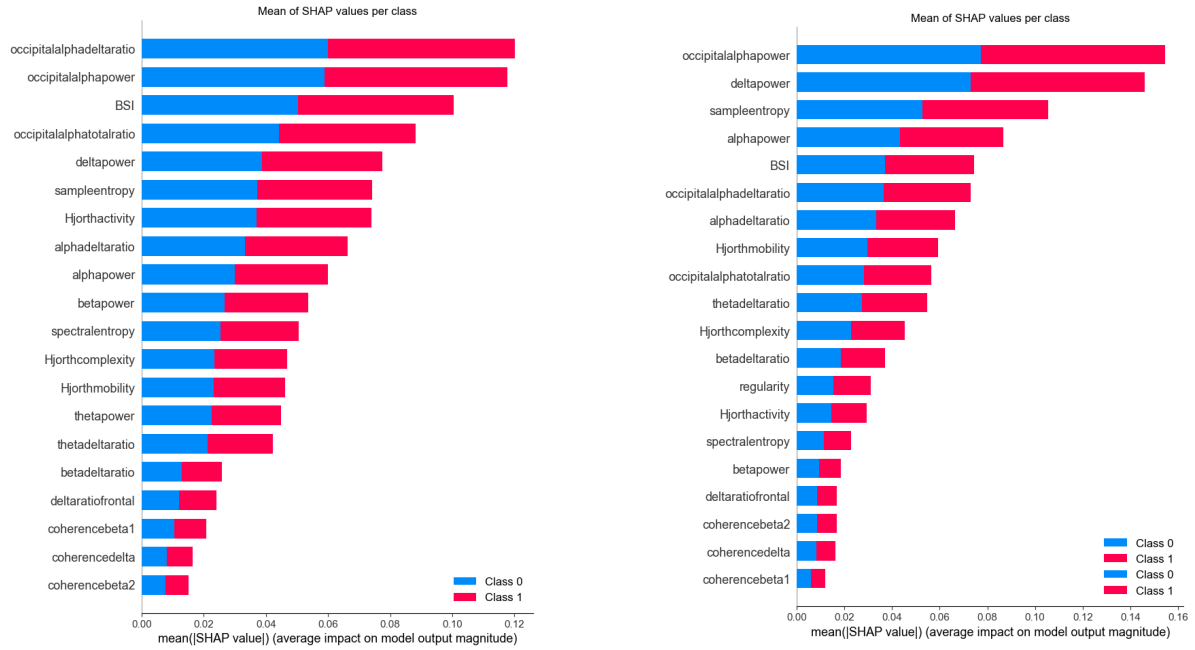


Figure 7.17: Mean of SHAP values of normal EEG $\mathcal{Z}$: Tuned RF

We are more interested in the reasoning of the RF models for under- and oversampled sets. Figure 7.18b) shows the mean of SHAP values for the undersampled EEG set $\mathcal{Z}$. Initially, it is observed that both the SHAP and MDI methods prioritize the same five features, implying similar performance. In the model development section, we examined the scores for a default RF on all sets. Despite the suboptimal scores, we quickly examine the most important features for the default and tuned model of the undersampled EEG set. This analysis can help us determine whether the SHAP values rely on the type of model used.



(a)) Default RF

(b)) Tuned RF

Figure 7.18: Mean of SHAP values of undersampled EEG Z

Figure 7.18 shows that different models prioritize other features. Remarkably, the models have a feature overlap of 5 out of the first 6 features, even though their order differs. While feature `occipitalalphadeltaratio` is most important in the default setting, it become slightly less important in the tuned RF model.
Both the undersampled EEG and normal EEG set demonstrate overlap in important features, such as the `occipitalalpha-` features. However, looking at the mean values, the model for the normal EEG set assigns the second most important feature a mean value of ± 0.06, whereas in case of the undersampled EEG set, there are approximately 9 features that have a higher importance than 0.06. One possible explanation for the lower performance of the normal EEG set could be attributed to its heavy reliance on `sampleentropy`.

In the model development phase, we have seen that the RF models for the F_X sets demonstrate inadequate performance on all metrics. The models reached scores at or even below their baseline levels. The RF classifier for the oversampled F_X set performed well in the cross-validation sets, but did not generalize well on the test set. In Figure 7.19, we see that the mean of SHAP values is greatest for feature `coherencebeta1 min`, followed by `occipitalalphadeltaratio min` and `Hjortmobility std`. Furthermore, these features resemble the most important features according to MDI, Figure **??**. The two figures show the same top seven feature in different order.

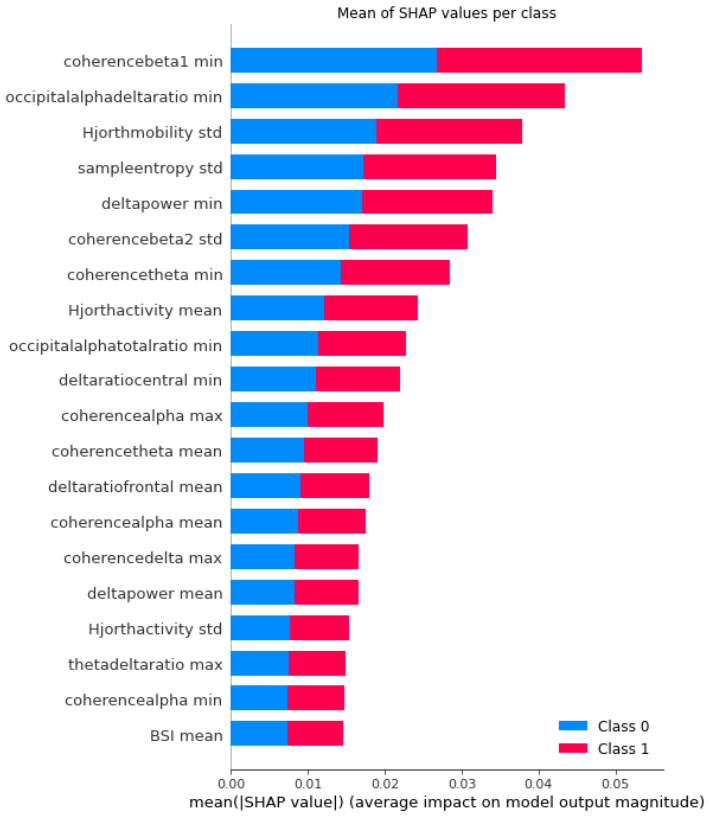


Figure 7.19: Mean of SHAP values of oversampled F_X: tuned RF

Lastly, for the oversampled set F_X, we present the threshold for the value of feature `coherencebeta1 min` in Figure 7.20. Values below 0.42, push the observation to class 1, whereas a value greater than 0.42 help the observation to class 0.

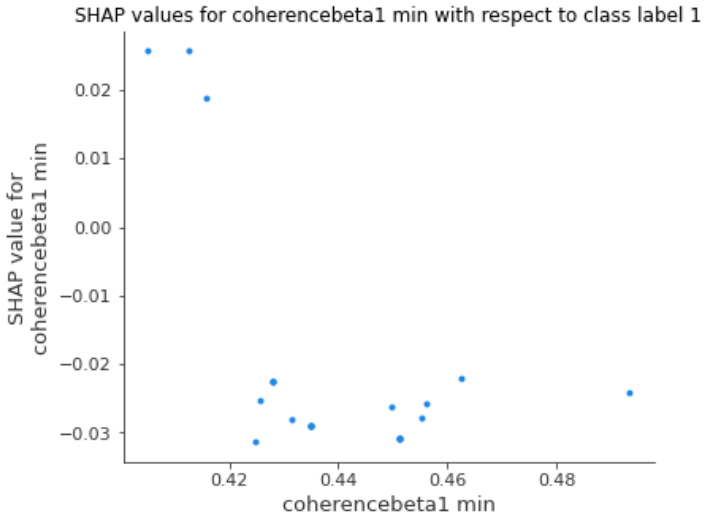
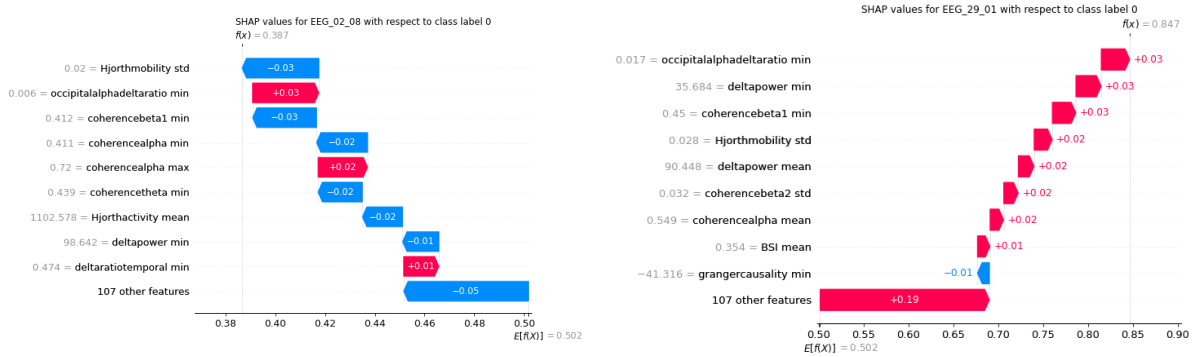


Figure 7.20: A threshold for coherencebeta1 min and its SHAP values for oversampled F_X. *On the x-axis values for feature coherencebeta1 min are displayed with corresponding SHAP values on the y-axis. The figure is created with respect to class* 1*, meaning that positive SHAP values increase the probability of being classified to class* 1.

In Table 7.7, the model for the oversampled F_X set misclassified an observation with true class label 0, due to the lower accuracy and specificity score. It appears in Figure **??** in Appendix F that `EEG_02_08` was misclassified as class label 1. The model correctly assigned `EEG_29_01` to class 0. In Figure 7.21,

both SHAP values for the EEG codes are plotted according to class label 0.



(a)) EEG 02 08

(b)) EEG 29 01

Figure 7.21: SHAP values for oversampled F_X: tuned RF

Observation `EEG_29_01` receives probability $f(x) = 0.847$ so that the observation is classified to class 0. On the other hand, we see that observation `EEG_02_08` get assigned a probability of $f(x) = 0.387$. It is remarkable that `coherencebeta1 min` pushes the classification of `EEG_02_08` to label 1, where the same feature points in opposite direction for `EEG_29_01`. Figure 7.20 shows that the threshold of `coherencebeta1 min` is around 0.42. On the y-axis of Figure 7.21, the observed values for the features are displayed. Evidently, this threshold guarantees that both observation receive contrasting SHAP values for `coherencebeta1 min`. All other 107 features push `EEG_29_01` in the right direction of class 1 with a value of 0.19, while in the case of `EEG_02_08` the other 107 features account for only -0.05. Other features considered important are `Hjorthmobility std`, `occipitaldeltaratio min` and `deltapower min`. The presence of these features in Figure 7.21 for observations `EEG_02_08` and `EEG_29_01` is not a mere coincidence, since the same features appeared in Figure 7.19.

The F_X set are summarized features of the values in the EEG dataset. The SHAP values tell us that from the EEG important feature `sampleentropy`, its standard deviation support the RF classifier in classification, as seen in Figure 7.19. Additionally, it is worth noting that features that are not considered important in the EEG data setting, gain importance when averaged or maximized, and vice versa. Both MDI and the SHAP values confirm this statement. For example, `spectralentropy` seemed important in the EEG setting, but vanishes in the F_X setting. Furthermore, the features linked to `coherence` are promimently present in the F_X feature importance measures, but they have low relevance in the EEG setting.

8

Conclusion

8.1. Conclusion

During the initial phase of diagnosis, patients with anti-NDMA-receptor encephalitis (anti-NMDARE) often experience severe symptoms that significantly impact their quality of life. Anti-NDMARE is an autoimmune disorder that affects the brain. Electroencephalography (EEG) plays a crucial role in the diagnosis and treatment process. An EEG recording measures the electrical activity of the brain by attaching electrodes to the patient's head. The ability to identify which EEG patterns relate to a positive or negative prognosis of anti-NMDARE is crucial. By distinguishing which patients have a favorable or unfavorable long-term (12 months) outcome, the intensity of the treatment can be adjusted accordingly. Currently, the classification of favorable or unfavorable outcome is performed manually, a process that is labor-intensive and provides a rough evaluation. EEG recordings are classified manually, which is labor-intensive and open to broad interpretation. Also, the factors that contribute to the severity of the disease remain unclear. If diagnosis, prognosis and treatment can be better understood, this could lead to an increased quality of life for patients with anti-NMDARE. The aim of this thesis was to analyse the EEG data for the anti-NMDARE disease and to predict which patients exhibit positive recovery after 12 months of standard treatment.

Initially, we made the decision to construct a Random Forest (RF) classifier for predicting the outcome after 12 months of anti-NDMARE based on available electroencephalography (EEG) features. While we have 44 patients with 29 features and 500 to 1500 recordings, the EEG dataset is complex and we have differences within one patient and between the observations of patients. The RF classifier can handle non-linear input data with high dimensions and is considered an accurately and interpretable classification model. By valuing interpretability of the prediction, we defined one of our goals to determine which features improve the classification model.

A prominent challenge encountered early in this research was the class imbalance. Only 9 patients in the dataset were classified with an unfavorable outcome (label $= 1$), whereas 35 patients had a favorable outcome after 12 months (label $= 0$). The issue was addressed by under- and oversampling the dataset.
In the exploratory analysis, we noticed features that were highly correlated with the class labels (`spectralentropy`, `PLI` and `BSI`) and ones which were not (`coherencetheta`, `thetapower` and `coherencealpha`). The presumption was that these features should play an important role in the classification model. Statistical significance tests did not provide us much information for the EEG features. This was partly due to the fact that the EEG data can be considered as clustered data, as there are repeated measurements per patient (=cluster). The cluster-based structure was confirmed by performing an F-test and with the use of an Intracluster Correlation Coefficient (ICC) which showed that some features had large variance between the means of all patients compared to the total variance within patients. Again `spectralentropy` and `BSI` came out as important features in this analysis. Furthermore, `sampleentropy`, `Hjorthmobility` and `occipitalalphapower` indicated a clustering effect in the data. Identifying the clustered structure of the EEG dataset led to defining three ways to handle at the data:

ignoring clustering, reducing the clustering to independent observations and explicitly accounting for clustering. The first two options were explored throughout this research. Reducing clustering led to a dataset noted as F_X which consists of the mean, standard deviation, minimum and maximum of all EEG features per patient. Subsequently, we identified features, that were not correlated with the class label for the EEG case, but were of statistical significance in the reduced F_X set.

The RF is a boosting algorithm that aggregates the predictions of multiple Decision Trees (DT). By bootstrap aggregation, the RF method reduces the risk over overfitting. This is due to the fact that not all trees are trained on the same subset of the training data. Furthermore, RF include feature randomness ensuring the DTs are decorrelated. Features are chosen randomly at each split, ensuring not only important features can be chosen at a split in the DT. The Gini index and entropy are impurity measures used for measuring the quality of a split and identifying the best feature for splitting a node. However, they do not take class imbalance into account. The parameters of the RF model were tuned, aiming to find an accurate classification model for the anti-NMDARE data. Besides the RF model, we used both the Mean Decrease Impurity and SHAP algorithm for the model's interpretability.

In the simulation study, we have seen that the EEG set Z_{sim} (undersampled, normal or oversampled) perform better than the $F_{X\text{sim}}$ sets. This suggests that the presence of more information contained in the EEG sets leads to improved performance, while acknowledging the ignorance of the independence assumption. In the case of the simulation sets, under- or oversampling did not yield the desired effect, The normal sets exhibited similar of even superior performance compared to the the under- and oversampled sets. However, PCA did not have the desired effect on the scores. This could possibly be attributed to the class dependent features that have low correlation with the rest of the features. SHAP values indicated correctly that the feature dependent on class label was considered most important. Furthermore, the threshold for `feature 2 mean` defined by the SHAP values, would able to split the training observations indicating an accurate threshold. We recognize that the simulation data was defined transparently and understandably. In real life case, there is hardly ever one feature that is able to classify all instances. Moreover, the classification method would be excessive, as it is possible to separate the classes using only one feature.

Classifying the true observed EEG data, the tuned RFs for the under- and oversampled EEG Z succeeded in correctly classifying observations from class 1. All normal sets assigned all observations to label 0, reaching a baseline for the accuracy of 0.786. This baseline represents the scores of a simplistic classifier that classifies all observation to the majority class. All three models for the F_X sets were at or even below their baseline levels. Based on this observation, for both the real data and the simulated case, it is recommended to use the complete EEG sets Z as input for the RF classifier. Also, on the real EEG dataset, PCA does not perform better than the normal EEG sets.
Analyzing the robustness of our model with cross-validation, we have seen that the oversampled F_X set reaches high scores in all 3 folds, and is preferred over the undersampled and normal F_X set. An interesting finding from the EEG dataset was that the undersampled set exhibited higher mean sensitivity and F-score, but at the cost of reduced mean in specificity. This might be partly due to its new class imbalance of EEG epochs which warrants further attention.

Even though we see that the scores for the performance metrics are not yet optimal, we have a good understanding of the dataset and by performing several tests, together with PCA and feature importance methods, we identified the most important features for determining the class labels. Concluding remarks from feature importance methods MDI and SHAP are that features related to the connectivity domain become important when taking the mean, minimum or maximum in F_X. These features include `coherencebeta1 min`, `coherencebeta2 std` and `coherencetheta min`. Other recurrent features were `occipitalalphadeltaration min`, `Hjorthmobility std`, `sampleentropy std` and `deltapower min`. In the EEG setting, MDI emphasized the importance of the features `deltapower`, `sampleentropy` and `occipital`-related features. These features remain important in the F_X set. Remarkably, `spectralentropy` did not receive the importance assumed in the beginning of this research. Both MDI and SHAP prioritize the same features, however SHAP can identify how certain features contribute to the prediction of a observation, making the prediction and the RF model highly interpretable.

The challenges for the RF classifier in the case of anti-NDMARE lies in the class imbalance and are associated with classifying the minority class. Under- and oversampling the dataset led to the succession of correctly classifying observations from the minority class. However, a part from its explainable characters, the classifier's ability to classify both classes needs to be improved.

8.2. Future research

Several recommendations to address the challenges encountered in this research are

- The results from this research suggest the complete EEG sets Z perform better than the reduced F_X sets. Too much noise in the F_X dataset might have led to the suboptimal scores for class labels 1. The set contains 116 features compared to 44 independent observations. A suggestion for further research is exploring various feature selection methods for the F_X and EEG dataset given the identification of important features in this research. Different combinations of these features, taking into account their correlation coefficients, should be explored as input for the classifier. In this thesis, some of the sets are tested but did not prove any improvements in test scores.
- To improve the scores for the performance metrics, other classification methods can be explored and tested such as the Support Vector Classifier (SVC) or logistic regression. Because the RF classifier has higher interpretability, we did not choose these methods initially. By using the feature importance methods from the RF classifier, a possibility is to incorporate these features into less interpretable classifiers, keeping in mind that this might lead to biased RF models.
- A straightforward yet highly effective solution for class imbalance is to collect more data. The classifier's limited ability to classify observation from the minority class can be traced back to the insufficient amount of data available. Determining the optimal amount of data required for improved performance is challenging. Continuous testing is necessary to assess the ideal data quantity. Additionally, apart from over- and undersampling, exploring alternative class balancing techniques can be advantageous. Furthermore, when there is more data available, the F_X sets will have more favorable dimensions for the classifier. If there are more observations than feature, the feature reduction method PCA can also be applied on this set. However, it should be noted that in our research, PCA did not improve performance.
- This thesis focused on the applications of EEG data and on highlighting the important features extracted from the EEG signals. The addition of other clinically relevant features such as age, gender or MRI findings, not directly associated with EEG data, could prove advantageous.
- In addition to the previous point, we must also consider the ethical challenges of fair training ML models. We do not want our model to be unintentionally biased against certain groups of people. While we acknowledge that the disease predominantly affects young women and children, it should not result in men always being misclassified. The subject of ethics and the potential for biased training in ML methods is a broad topic that requires careful study and consideration.
- Another recommended approach or consideration is labeling all EEG epochs and not keeping all labels constant over every epoch per patient. This objective would be challenging, since there is no strong association with an EEG epoch and the disease outcome. One potential approach could involve labeling epochs that are considered significant in assessing the severe of the disease. However, biases should be avoided, as Machine Learning (ML) models then learn from the information provided by clinicians. After acknowledging the challenges and limitations of this task, the curiosity for the potential impact of this on the model's performance remains. Besides anti-NMDARE, this research can for example also explore the classification of epileptic seizures.
- An evident next step is to leverage all the information captured in the EEG data while also acknowledging the assumption of independence in ML models. In this thesis, RF models that can handle clustering were introduced. These solutions appear promising when the objective is to utilize the extracted features from the EEG signals. Due to their complex characteristics and challenging implementation, the methods were categorized for further research. For future research, it is important to validate the assumptions of the RF models, as certain models require each observation to have its own class label.
- For raw EEG signals, a Convolution Neural Network (CNN) classifier in combination with the explainable Artificial Intelligence (AI) algorithm GradCAM was considered a promising solution

for the anti-NMDARE classification problem. This method requires less data preprocessing and can identify important fragments in the EEG signal, which does not compromise interpretability. However, the perception of a CNN as a black-box might render it a less appealing method to explore.

- Finally, a recommendation is to explore alternative methods discussed in the aforementioned points by generating a more realistic simulated dataset. Conducting a simulation study guarantees a better understanding of the model's performance. In this thesis, the simulated dataset unfortunately fell short of meeting the expectations in terms of resemblance to the true dataset.
- In order to boost the interdisciplinarity of the project, there is a personal preference to incorporate diverse perspectives. Although, it may be challenging for a mathematician to understand signal analysis in EEG data or comprehend the EEG signals themselves, the model could potentially gain valuable insights by understanding the interaction among various features. This interdisciplinary approach would contribute to a more comprehensive understanding of the anti-NDMARE classification problem.

It is important to continue research on this topic while it could be a possibility to expand our knowledge on this rare disease, improving patients' quality of life. Additionally, other uncommon and rare diseases with imbalanced outcomes of treatment can then be analyzed by the methodology.

References

[1] AV Sonderen, S Arends, DLJ Tavy, et al. “Predictive value of electroencephalography in anti-NMDA receptor encephalitis”. In: *J Neurol Neurosurg Psychiatry* 89 (Oct. 2018), pp. 1101–1106. DOI: https://doi.org/10.1136/jnnp-2018-318376.

[2] Brin Freund and Eva K. Ritzl. *A review of EEG in anti-NMDA receptor encephalitis*. July 2019. DOI: 10.1016/j.jneuroim.2019.03.010.

[3] Lui F Samanta D. “Anti-NMDA Receptor Encephalitis”. In: *StatPearls [Internet]* (2022). URL: https://www.ncbi.nlm.nih.gov/books/NBK551672/.

[4] M. Gavaret, A. Iftimovici, and E. Pruvost-Robieux. “EEG: Current relevance and promising quantitative analyses”. In: *Revue Neurologique* (Apr. 2023). ISSN: 00353787. DOI: 10.1016/j.neurol.2022.12.008.

[5] Amir Masoud Rahmani, Efat Yousefpoor, Mohammad Sadegh Yousefpoor, et al. *Machine learning (Ml) in medicine: Review, applications, and challenges*. Nov. 2021. DOI: 10.3390/math9222970.

[6] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. “Deep learning for electroencephalogram (EEG) classification tasks: a review”. In: *Journal of Neural Engineering* 16.3 (June 2019), p. 031001. ISSN: 1741-2560. DOI: 10.1088/1741-2552/ab0ab5.

[7] Hema Sekhar Reddy Rajula, Giuseppe Verlato, Mirko Manchia, et al. “Comparison of conventional statistical methods with machine learning in medicine: Diagnosis, drug development, and treatment”. In: *Medicina (Lithuania)* 56.9 (Sept. 2020), pp. 1–10. ISSN: 16489144. DOI: 10.3390/medicina56090455.

[8] Luc Rubinger, Aaron Gazendam, Seper Ekhtiari, et al. “Machine learning and artificial intelligence in research and healthcare”. In: *Injury* (2022). ISSN: 18790267. DOI: 10.1016/j.injury.2022.01.046.

[9] Hui Wen Loh, Chui Ping Ooi, Silvia Seoni, et al. *Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011–2022)*. Nov. 2022. DOI: 10.1016/j.cmpb.2022.107161.

[10] Roberto Confalonieri, Ludovik Coba, Benedikt Wagner, et al. “A historical perspective of explainable Artificial Intelligence”. In: *WIREs Data Mining and Knowledge Discovery* 11.1 (2021), e1391. DOI: https://doi.org/10.1002/widm.1391. URL: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1391.

[11] Po Hsuan Cameron Chen, Yun Liu, and Lily Peng. *How to develop machine learning models for healthcare*. May 2019. DOI: 10.1038/s41563-019-0345-0.

[12] Gareth James, Daniela Witten, Trevor Hastie, et al. *An Introduction to Statistical Learning with Applications in R Second Edition*. 2021.

[13] Leo Breiman. “Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)”. In: *Statistical Science* 16.3 (Aug. 2001). ISSN: 0883-4237. DOI: 10.1214/ss/1009213726.

[14] Rich Caruana. “An Empirical Comparison of Supervised Learning Algorithms”. In: *ICML ’06: Proceedings of the 23rd international conference on Machine learning* (2006), pp. 161–168. DOI: https://doi.org/10.1145/1143844.1143865.

[15] M. Swathy and K. Saruladha. “A comparative study of classification and prediction of Cardio-Vascular Diseases (CVD) using Machine Learning and Deep Learning techniques”. In: *ICT Express* 8.1 (Mar. 2022), pp. 109–116. ISSN: 24059595. DOI: 10.1016/j.icte.2021.08.021.

[16] Xiaolong Zhai, Zhanhong Zhou, and Chung Tin. “Semi-supervised learning for ECG classification without patient-specific labeled data”. In: *Expert Systems with Applications* 158 (Nov. 2020). ISSN: 09574174. DOI: 10.1016/j.eswa.2020.113411.

[17] Amir Masoud Rahmani, Efat Yousefpoor, Mohammad Sadegh Yousefpoor, et al. *Machine learning (Ml) in medicine: Review, applications, and challenges*. Nov. 2021. DOI: 10.3390/math9222970.

[18] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*. May 2015. DOI: 10.1038/nature14539.

[19] Daniel Westreich, Justin Lessler, and Michele Jonsson Funk. *Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression*. 2010. DOI: 10.1016/j.jclinepi.2009.11.020.

[20] Yousef Methkal Abd Algani, Mahyudin Ritonga, B. Kiran Bala, et al. “Machine learning in health condition check-up: An approach using Breiman’s random forest algorithm”. In: *Measurement: Sensors* 23 (Oct. 2022). ISSN: 26659174. DOI: 10.1016/j.measen.2022.100406.

[21] Rich Caruana and Alexandru Niculescu-Mizil. *Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria*. Tech. rep. 2004.

[22] Rachael Hagan, Charles J. Gillan, and Fiona Mallett. “Comparison of machine learning methods for the classification of cardiovascular disease”. In: *Informatics in Medicine Unlocked* 24 (Jan. 2021). ISSN: 23529148. DOI: 10.1016/j.imu.2021.100606.

[23] Xiaolong Zhai and Chung Tin. “Automated ECG Classification Using Dual Heartbeat Coupling Based on Convolutional Neural Network”. In: *IEEE Access* 6 (May 2018), pp. 27465–27472. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2833841.

[24] Kevin Fauvel, Tao Lin, Véronique Masson, et al. “Xcm: An explainable convolutional neural network for multivariate time series classification”. In: *Mathematics* 9.23 (Dec. 2021). ISSN: 22277390. DOI: 10.3390/math9233137.

[25] Microgen. *Female Patient in a Neurology Lab doing EEG Scan*. [Online; accessed on June 27, 2023]. URL: https://canva.com/.

[26] Rungruedee Malasri. *Brain wave patterns on EEG*. [Online; accessed on June 27, 2023]. URL: https://canva.com/.

[27] The SciPy community. *SciPy skewness*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skew.html.

[28] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[29] Jaime Lynn Speiser, Bethany J. Wolf, Dongjun Chung, et al. “BiMM forest: A random forest method for modeling clustered and longitudinal binary outcomes”. In: *Chemometrics and Intelligent Laboratory Systems* 185 (Feb. 2019), pp. 122–134. ISSN: 18733239. DOI: 10.1016/j.chemolab.2019.01.002.

[30] Jianchang Hu and Silke Szymczak. “A review on longitudinal data analysis with random forest”. In: *Briefings in Bioinformatics* (Jan. 2023). ISSN: 1467-5463. DOI: 10.1093/bib/bbad002.

[31] David Liljequist, Britt Elfving, and Kirsti Skavberg Roaldsen. “Intraclass correlation – A discussion and demonstration of basic features”. In: *PLoS ONE* 14.7 (July 2019). ISSN: 19326203. DOI: 10.1371/journal.pone.0219854.

[32] H. Shou, A. Eloyan, S. Lee, et al. “Quantifying the reliability of image replication studies: The image intraclass correlation coefficient (I2C2)”. In: *Cognitive, Affective and Behavioral Neuroscience* 13.4 (Dec. 2013), pp. 714–724. ISSN: 15307026. DOI: 10.3758/s13415-013-0196-0.

[33] Jessalyn K. Holodinsky, Peter C. Austin, and Tyler S. Williamson. “An introduction to clustered data and multilevel analyses”. In: *Family Practice* 37.5 (2020), pp. 719–722. ISSN: 14602229. DOI: 10.1093/FAMPRA/CMAA017.

[34] Sally Galbraith, James A. Daniel, and Bryce Vissel. “A study of clustered data and approaches to its analysis”. In: *Journal of Neuroscience* 30.32 (Aug. 2010), pp. 10601–10608. ISSN: 02706474. DOI: 10.1523/JNEUROSCI.0362-10.2010.

[35] Terry K. Koo and Mae Y. Li. “A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research”. In: *Journal of Chiropractic Medicine* 15.2 (2016), pp. 155–163. ISSN: 1556-3707. DOI: https://doi.org/10.1016/j.jcm.2016.02.012. URL: https://www.sciencedirect.com/science/article/pii/S1556370716000158.

[36] Leo Breiman. *Random Forests*. Tech. rep. 2001, pp. 5–32.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Tech. rep.

[38] Peter Calhoun, Richard A. Levine, and Juanjuan Fan. “Repeated measures random forests (RMRF): Identifying factors associated with nocturnal hypoglycemia”. In: *Biometrics* 77.1 (Mar. 2021), pp. 343–351. ISSN: 15410420. DOI: 10.1111/biom.13284.

[39] Weikuan Jia, Meili Sun, Jian Lian, et al. “Feature dimensionality reduction: a review”. In: *Complex and Intelligent Systems* 8.3 (June 2022), pp. 2663–2693. ISSN: 21986053. DOI: 10.1007/s40747-021-00637-x.

[40] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[42] scikit-learn. *scikit-learn: cross-validation*. https://scikit-learn.org/stable/modules/cross_validation.html.

[43] Gérard Biau. *Consistency of Random Forests and Other Averaging Classifiers Luc Devroye Gábor Lugosi*. Tech. rep. 2008, pp. 2015–2033. URL: http://www.stat.berkeley.edu/users/breiman/RandomForests.

[44] Gérard Biau. *Analysis of a Random Forests Model*. Tech. rep. 2012, pp. 1063–1095.

[45] Wei Gao and Zhi-Hua Zhou. *Towards Convergence Rate Analysis of Random Forests for Classification*. Tech. rep.

[46] Pakize Yiğit, Abdulbari Bener, and Seda Karabulut. “Comparison of machine learning classification techniques to predict implantation success in an IVF treatment cycle”. In: *Reproductive BioMedicine Online* 45.5 (Nov. 2022), pp. 923–934. ISSN: 14726491. DOI: 10.1016/j.rbmo.2022.06.022.

[47] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. “Correlation and variable importance in random forests”. In: (Oct. 2013). DOI: 10.1007/s11222-016-9646-1. URL: http://arxiv.org/abs/1310.5726%20http://dx.doi.org/10.1007/s11222-016-9646-1.

[48] Codecademy Team. *Feature Importance*. https://www.codecademy.com/article/fe-feature-importance-final.

[49] Gilles Louppe. “Understanding Random Forests: From Theory to Practice”. In: (July 2014). URL: http://arxiv.org/abs/1407.7502.

[50] Mukund Sundararajan and Amir Najmi. “The many Shapley values for model explanation”. In: (Aug. 2019). URL: http://arxiv.org/abs/1908.08474.

[51] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[52] Maryna Ievdokimova. *EEG wave in human brain*. [Online; accessed on June 27, 2023]. URL: https://canva.com/.

List of Figures

List of Tables

A

Advantages and Disadvantages of Classification Methods

.

Algorithm	Advantages	Disadvantages
Naive Bayes	simple implementation, high computational speed, high learning speed, high classification speed, managing overfitting, managing noisy data, and managing missing values	assuming independence of features, lack of ability to manage features with high correlation, low accuracy
Decision Tree (DT)	simple understanding, high computational speed, high learning speed, high classification speed, managing missing values	the complexity of designing large trees, lack of ability to manage overfitting, low ability to manage noisy data, low ability to manage data with high correlation, mediuma accuracy
Articificial Neural Network (ANN)	high flexibility, high accuracy, high classification speed, ability to manage data with high correlation, suitable for nonlinear and complex databases	difficult implementation, low learning speeds, inability to manage noisy data, lack of ability to manage overfitting
Ensemble learning system	high accuracy, ability to manage missing values, ability to manage noisy data, ability to manage overfitting, ability to manage data with high correlation, high classification speed, high stability	difficult implementation, low learning speed, high computational complexity
Random Forest (RF)	ability to manage noisy data, high classification speed, suitable for large and heterogeneous databases	difficult understanding by humans, difficult implementation, medium accuracy, low learning speed, low ability to manage missing values, low ability to manage overfitting, low ability to manage data with high correlation
Deep Learning (DL)	suitable for large and high dimensional databases, high accuracy, high classification speed, ability to manage noisy data, ability to manage data with high correlation	difficult implementation, low learning speed, inability to manage overfitting, low ability to manage missing values
Support Vector Machine (SVM)	ability to manage data with linear separability and nonlinear separability, high accuracy, high classification speed, ability to manage data with high correlation	assuming linear separability for dataset, low ability to manage overfitting, low learning speed, low ability in managing missing values, low ability to manage noisy data
k-Nearest Neighbor (k-NN)	simple algorithm, stable performance, high learning speed, ability to manage overfitting	high computational overhead, sensitivity to local data structures, medium accuracy, low classifcation speed, low ability in managing missing values, inability to manage noisy data, inability to manage data with high correlation

Table A.1: Supervised learning methods and their advantages and disadvantages by [17]

B

Description of Extracted EEG Features

B.1. Table with descriptions

	Feature	Definition
Frequentie	deltapower	absolute power per $[0.4 - 4Hz]$
	thetapower	absolute power per $[4 - 8Hz]$
	alphapower	absolute power per $[8 - 12Hz]$
	betapower	absolute power per $[12 - 25Hz]$
	thetadeltaratio	$\frac{\text{absolute power of theta band}}{\text{absolute power of delta band}}$
	alphadeltaratio	$\frac{\text{absolute power of alpha band}}{\text{absolute power of delta band}}$
	betadeltaratio	$\frac{\text{absolute power of beta band}}{\text{absolute power of delta band}}$
	deltaratiofrontal	$\frac{\text{absolute power of delta band in frontal region}}{\text{total power of delta band}}$
	deltaratiotemporal	$\frac{\text{absolute power of delta band in temporal region}}{\text{total power of delta band}}$
	deltaratiocentral	$\frac{\text{absolute power of delta band in central region}}{\text{total power of delta band}}$
	deltaratioparietal	$\frac{\text{absolute power of delta band in parietal region}}{\text{total power of delta band}}$
	deltaratiooccipital	$\frac{\text{absolute power of delta band in occipital region}}{\text{total power of delta band}}$
	occipitalalphapower	absolute power of alpha band on occipitale electrodes
	occipitalalphadeltaratio	$\frac{\text{absolute power of alpha band}}{\text{absolute power of delta band}}$ (both on occipitale electrodes)
	occipitalalphatotalratio	$\frac{\text{absolute power of alpha band}}{\text{total power}}$ (both on occipitale electrodes)
	Brain Symmetry Index (BSI)	$\frac{\text{left hemisphere frequency}}{\text{right hemisphere frequency}}$
Complexity	spectralentropy	similarity on the frequency spectrum
	sampleentropy	similarity of the amplitude
	regularity	normalised variance of the amplitudes
	Hjorthactivity	variance of a time series
	Hjorthmobility	mean frequency of a time series
	Hjorthcomplexity	change in the signal compared to a pure sine wave
Connectivity	Phase-lag index (PLI)	measure of the extent to which phase synchronization occurs in signals, measured in different brain regions
	coherencedelta	coherence in [0.5-4Hz] domain
	coherencetheta	coherence in [4-8Hz] domain
	coherencealpha	coherence in [8-12Hz] domain
	coherencebeta1	coherence in [12-18Hz] domain
	coherencebeta2	coherence in [18-25Hz] domain
	grangercausality	outcome of a statistical test to what extent a time series predicts another

Table B.1: Extracted features from EEG signals

B.2. Table with summary statistics per EEG feature

EEG feature	Mean	Std	Min	Max
deltapower	220.22	285.49	2.72	3401.31
thetapower	22.25	94.53	0.48	2950.51
alphapower	8.31	12.58	0.25	281.64
betapower	7.30	7.67	0.39	181.19
thetadeltaratio	0.16	0.21	0	3.12
alphadeltaratio	0.16	0.34	0	6.51
betadeltaratio	0.14	0.25	0	3.96
deltaratiofrontal	1.27	0.33	0.14	3.52
deltaratiotemporal	0.98	0.33	0.10	3.71
deltaratiocentral	0.77	0.36	0.05	4.16
deltaratioparietal	0.81	0.31	0.06	4.37
deltaratiooccipital	0.78	0.30	0.06	3.86
occipitalalphapower	16.91	30.88	0.31	539.68
occipitalalphadeltaratio	0.40	1.15	0	33.76
occipitalalphatotalratio	0.16	0.37	0	7.14
BSI	0.31	0.13	0.05	0.77
spectralentropy	0.66	0.06	0.53	0.86
sampleentropy	0.03	0.02	0	0.14
regularity	0.72	0.06	0.35	0.93
Hjorthactivity	397.85	496.93	18.63	11606.54
Hjorthmobility	0.24	0.15	0.03	1
Hjorthcomplexity	1.04	0.20	0.40	1.50
PLI	0.11	0.04	0.03	0.34
coherencedelta	0.58	0.06	0.41	0.88
coherencetheta	0.56	0.05	0.41	0.86
coherencealpha	0.54	0.05	0.39	0.80
coherencebeta1	0.53	0.05	0.39	0.82
coherencebeta2	0.51	0.04	0.39	0.77
grangercausality	-25.74	5.94	-46.70	-0.16

Table B.2: Descriptive analysis of EEG features: mean, std, min and max

C

SHAP values

The SHAP values for classification use the predicted probability for the classes. In the case of the RF classifier, the `.predict.proba()` property is utilized. This function computes the class means of predicted probabilities over all trees in the forest. The SHAP values are computed per class, so we continue the explanation for one class.

The predicted probability for a class is denoted by $\hat{f}(x)$, where $\hat{f}$ is our RF model and x an observation. Observation x consists of multiple features $j \in \{1, \ldots, p\}$. SHAP value $\phi_j(\hat{f})$ is the contribution of fetaure j on the predicted outcome. Summing over all features, we get

$$\sum_{j=1}^{p} \phi_j(x_j) = \hat{f}(x) - \mathbb{E}(\hat{f}(X))$$

This equation can be easily derived from a linear prediction model. For one observation the linear prediction model is

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

Each β_j is the weight corresponding to feature j. The contribution of feature j on the prediction $\hat{f}(x)$ is [51]

$$\phi_j(x_j) = \beta_j x_j - \mathbb{E}(\beta_j x_j) = \beta_j x_j - \beta_j \mathbb{E}(x_j)$$

Summing over all features, we get

$$\begin{aligned}
\sum_{i=1}^{p} \phi_j(\hat{f}) &= \sum_{j=1}^{p} (\beta_j x_j - \mathbb{E}(\beta_j x_j)) \\
&= (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} \mathbb{E}(\beta_j x_j)) \\
&= \hat{f}(x) - \mathbb{E}(\hat{f}(X))
\end{aligned}$$

Note that $\mathbb{E}(\hat{f}(X))$ is the expected value of $\hat{f}$ for an RV X, where X follows the distribution of the observations. We do not know the true distributions of the features. We can try to estimate them by using the observations x from our dataset.

Since our model is not linear, we need another equation for Shapley values. The computation of Shapley value uses a function $v(S)$, see Equation 5.6. This set function $v(S)$ represents the prediction for feature values in the set S which is calculated by marginalizing over the features that are not included in S, i.e.

$$v(S) = \int \hat{f}(x_1, \ldots, x_p) d\mathbb{P}_{x \notin S} - \mathbb{E}(\hat{f}(X))$$

As an example, we pick a non-linear model that uses features $x_1, \ldots, x_4$. Let S be a subset consisting of the features x_1 and x_3. To evaluate the prediction of set S, we calculate

$$v(S) = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{f}(x_1, X_2, x_3, X_4) d\mathbb{P}_{X_2} d\mathbb{P}_{X_4} - \mathbb{E}_X(\hat{f}(X))$$

The first term is integrating over all possible values of X_2 and X_4 according to their probability measure $\mathbb{P}$ in the prediction model. Intuitively, the double integral represents a way of aggregating the prediction model $f(X)$ by averaging it over all possible values of X_2 and X_4. Subtracting the expected value of the prediction model for all X (in our example $X = (X_1, X_2, X_3, X_4)$), denoted by $\mathbb{E}_X(\hat{f}_X(X))$, determines the effect of subset S on the prediction model.

Some properties that are satisfied by the Shapley values are

1. **Efficiency** For each observation, the Shapley values must add up to the difference between the prediction of x and the expected values for $\hat{f}(X)$:

   $$\sum_{j=1}^{p} \phi_j = \hat{f}(x) - \mathbb{E}_X(\hat{f}(X))$$

2. **Symmetry** If two features contribute equally to all possible subsets, their Shapley values should be the same. So if for all $S \subseteq \{1, \ldots, p\} \setminus \{j, k\}$

   $$v(S \cup \{j\}) = v(s \cup \{k\})$$

   then

   $$\phi_j = \phi_k$$

   This principle ensures that the contribution of features j and k are equal when they have the same value across all subsets in which they do not participate.
3. **Dummy** If a feature is added to a subset S and it does not change the predicted value, it's Shapley value should be 0. So if for all $S \subseteq \{1, \ldots, p\}$

   $$v(S \cup \{j\}) = v(S)$$

   then

   $$\phi_j = 0$$

4. **Additivity** For a prediction model that combines multiple predictions, the Shapley values are defined by

   $$\phi_j^{(1)} + \phi_j^{(2)}$$

   This property guarantees that for a Random Forest classifier the Shapley values of all Decision Trees can be averaged to obtain a Shapley value for the RF.

D

Definition of Simulated Data in Python

```
#%% define function to simulate data per patient

# Initiate data per patient

k = 29 #(no. features)

def SimulateFeatures(y, j):
    i = np.random.randint(500, 1500+1)
    X = pd.DataFrame(index = ['EEG_'+str(j) for t in np.arange(1,i+1,1)],
                     columns = ['feature ' + str(l) for l in np.arange(1,k+1,1)])

    MS = np.zeros((2,k))

    MS[0,0:5] = np.random.uniform(0,2,5)
    #MS[0,4] = 2*MS[0,2]
    MS[1,0:5] = np.random.uniform(0,1,5)
    #MS[1,3] = 4*MS[1,0]

    MS[0,5:10] = np.random.uniform(4,5,5)
    #MS[0,6] = 0.5*MS[0,8]
    MS[1,5:10] = np.random.uniform(0,2,5)
    #MS[1,6] = 0.5*MS[0,8]

    MS[0,10:15] = np.random.uniform(20,30,5)
    MS[1,10:15] = np.random.uniform(0,4,5)

    MS[0,15:20] = np.random.uniform(2,20,5)
    MS[1,15:20] = np.random.uniform(0,2,5)

    MS[0,20:25] = np.random.uniform(2,10,5)
    MS[0,25:29] = np.random.uniform(1,4,4)

    f1 = np.random.normal(MS[0,0], MS[1,0], size = (i,))
    if y == 1:
        f2 = np.random.normal(MS[0,1], MS[1,1], size = (i,))
    else:
        f2 = 0.5*np.random.normal(MS[0,1], MS[1,1], size = (i,))+3
    f3 = np.random.normal(MS[0,2], MS[1,2], size = (i,))
    f4 = np.random.normal(MS[0,3], MS[1,3], size = (i,))
    f5 = np.random.normal(MS[0,4], MS[1,4], size = (i,))
    f6 = np.random.normal(MS[0,5], MS[1,5], size = (i,))
    f7 = np.random.normal(MS[0,6], MS[1,6], size = (i,))
    if y == 1:
        f8 = np.random.normal(MS[0,7], MS[1,7], size = (i,))
    else:
        f8 = 3*np.random.normal(MS[0,5], MS[1,5], size = (i,))-4
    f9 = np.random.normal(MS[0,8], MS[1,8], size = (i,))
    f10 = np.random.normal(MS[0,9], MS[1,9], size = (i,))

    f11 = np.random.normal(MS[0,10], MS[1,10], size = (i,))
```

```
    f12 = np.random.normal(MS[0,11], MS[1,11], size = (i,))
    f13 = np.random.normal(MS[0,12], MS[1,12], size = (i,))
    if y == 1:
        f14 = np.random.normal(MS[0,13], MS[1,13], size = (i,))
    else:
        f14 = np.random.normal(MS[0,12], MS[1,12], size = (i,))-5
    f15 = np.random.normal(MS[0,14], MS[1,14], size = (i,))
    #f16 = np.random.normal(MS[0,15], MS[1,15], size = (i,))
    f16 = f15/7
    f17 = np.random.normal(MS[0,16], MS[1,16], size = (i,))
    f18 = np.random.normal(MS[0,17], MS[1,17], size = (i,))
    f19 = np.random.normal(MS[0,18], MS[1,18], size = (i,))
    if y == 1:
        f20 = np.random.normal(MS[0,18], MS[1,18], size = (i,))*0.2
    else:
        f20 = np.random.normal(MS[0,19], MS[1,19], size = (i,))

    f21 = np.random.exponential(MS[0,20], size = (i,))
    f22 = np.random.exponential(MS[0,21], size = (i,))
    f23 = np.random.exponential(MS[0,22], size = (i,))
    if y == 1:
        f24 = 2*np.random.exponential(MS[0,22], size = (i,))
    else:
        f24 = np.random.exponential(MS[0,23], size = (i,))
    if y == 1:
        f25 = np.random.exponential(MS[0,24], size = (i,))
    else:
        f25 = np.random.exponential(MS[0,25], size = (i,))
    f26 = np.random.exponential(MS[0,25], size = (i,))
    #f27 = np.random.exponential(MS[0,27], size = (i,))
    f27 = 0.5*f26
    f28 = np.random.exponential(MS[0,27], size = (i,))
    f29 = np.random.exponential(MS[0,28], size = (i,))

    #rang = np.arange(1, i+1,1)

    #X['Epoch'] = rang
    X['feature 1'] = f1
    X['feature 2'] = f2
    X['feature 3'] = f3
    X['feature 4'] = f4
    X['feature 5'] = f5
    X['feature 6'] = f6
    X['feature 7'] = f7
    X['feature 8'] = f8
    X['feature 9'] = f9
    X['feature 10'] = f10
    X['feature 11'] = f11
    X['feature 12'] = f12
    X['feature 13'] = f13
    X['feature 14'] = f14
    X['feature 15'] = f15
    X['feature 16'] = f16
    X['feature 17'] = f17
    X['feature 18'] = f18
    X['feature 19'] = f19
    X['feature 20'] = f20
    X['feature 21'] = f21
    X['feature 22'] = f22
    X['feature 23'] = f23
    X['feature 24'] = f24
    X['feature 25'] = f25
    X['feature 26'] = f26
    X['feature 27'] = f27
    X['feature 28'] = f28
    X['feature 29'] = f29

    return X

#%% Repeat this for every patient and simulate their labels
no_pat = 40
```

```
def simulate_EEG_data(no_pat):
    patient_list = list()
    labels = np.zeros(no_pat,)

    for p in range(0,no_pat):
        labels[p] = np.random.binomial(1, 0.2)
        X = SimulateFeatures(labels[p], p+1)
        patient_list.append(X)

    y = pd.DataFrame(labels.astype(int),
                     index = ['EEG_'+str(i) for i in np.arange(1, no_pat+1,1)],
                 columns = ['Target label'])
    EEG_features = pd.concat(patient_list, axis = 0)
    EEG_features['Target label'] = y
    y_EEG = pd.DataFrame(columns = ['Target label'],
                         data = EEG_features[['Target label']].values,
                         index = EEG_features.index)
    EEG_features = EEG_features.drop(['Target label'], axis = 1)

    return EEG_features, y, y_EEG

EEG_features, y, y_EEG = simulate_EEG_data(44)
```

E

Simulation Results

E.1. Principal Component Analysis

Features with larger variances (see density functions) appear to have larger PCA weights. Even though the features are scaled. The scaling does take the variance into account. The scatterplots confirm correlation between certain features and so why they got assigned larger weights.

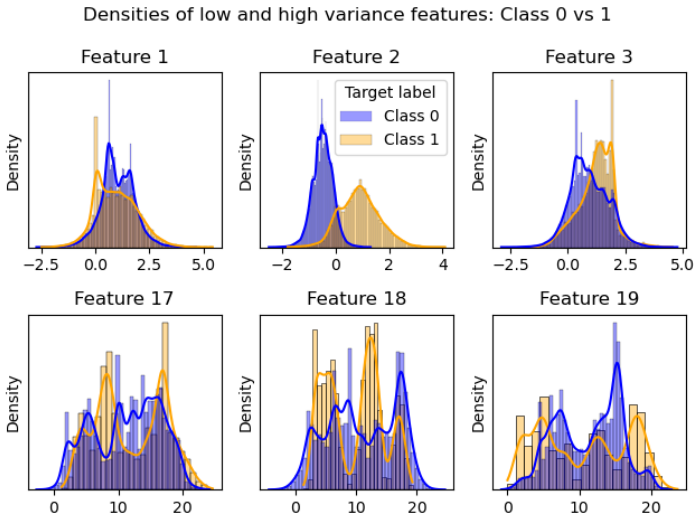


Figure E.1: Density of high and low covariance features

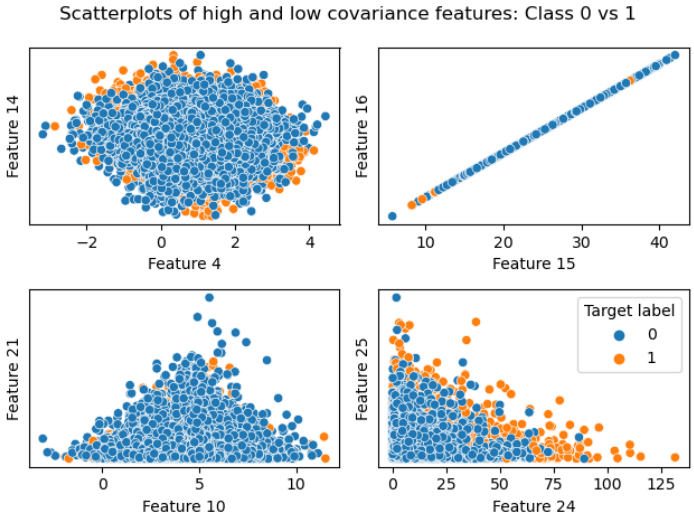


Figure E.2: Scatterplots for low and highly correlated features

E.2. Effect under- and oversampling

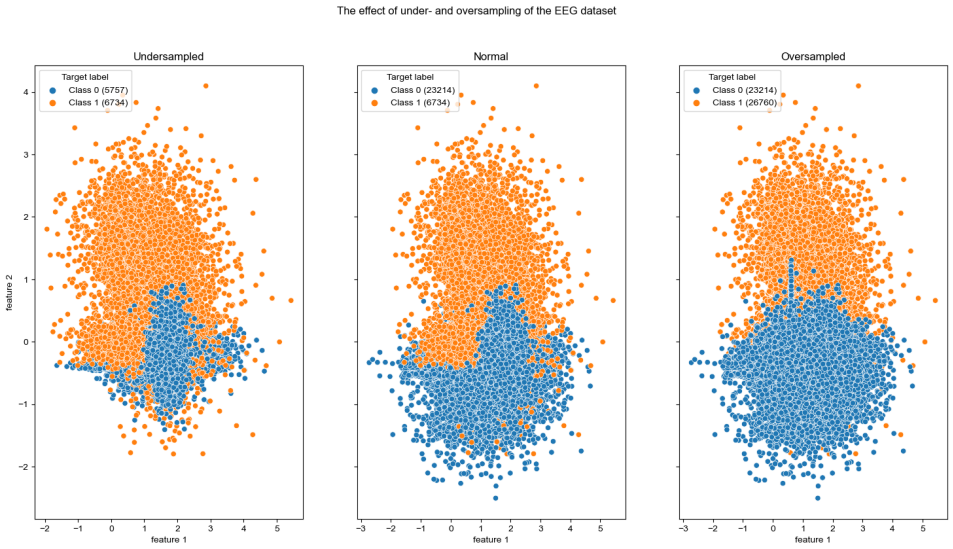


Figure E.3: The effect of under- and oversampling compared to the normal simulated EEG dataset Z_{sim} (in brackets the no. EEG observations per class)

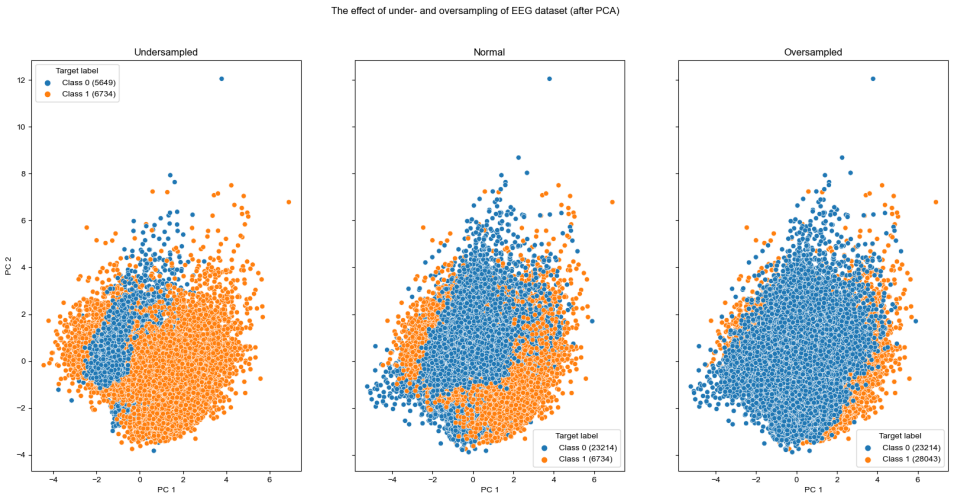
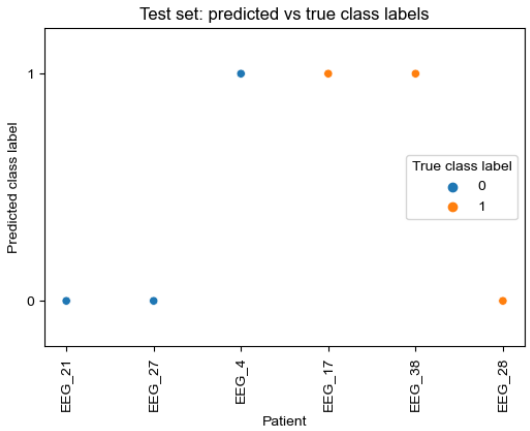


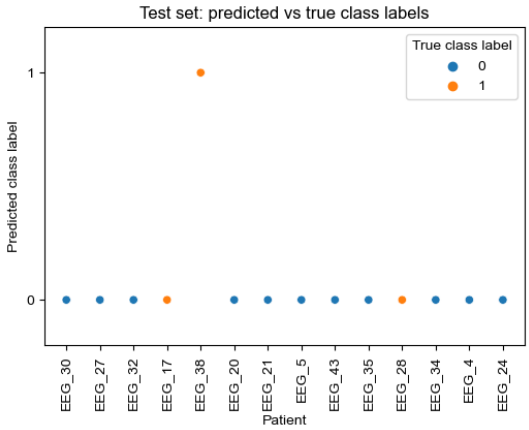
Figure E.4: The effect of under- and oversampling compared to the normal simulated EEG dataset after PCA Z_{simPCA}(in brackets the no. EEG observations per class)

E.3. Predicted versus true class labels

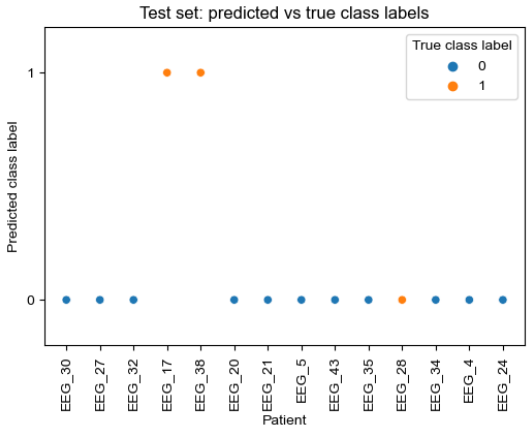
(a)) Undersampled $F_{X\text{sim}}$: default RF



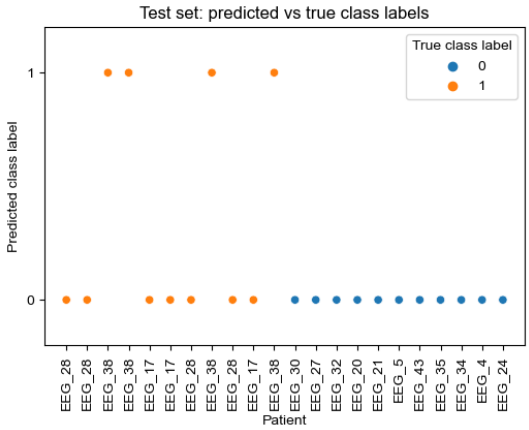
(b)) Undersampled $F_{X\text{sim}}$: optimized RF



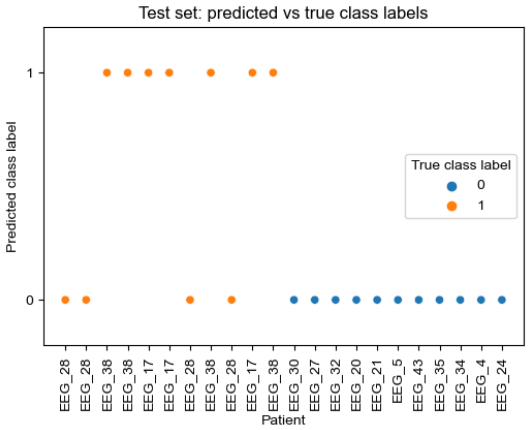
(c)) Normal $F_{X\text{sim}}$: default RF



(d)) Normal $F_{X\text{sim}}$: optimized RF



(e)) Oversampled $F_{X\text{sim}}$: default RF

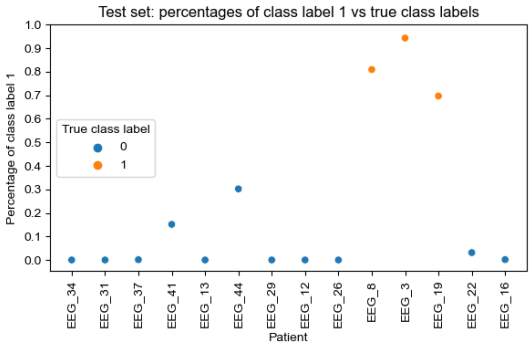


(f)) Oversampled $F_{X\text{sim}}$: optimized RF

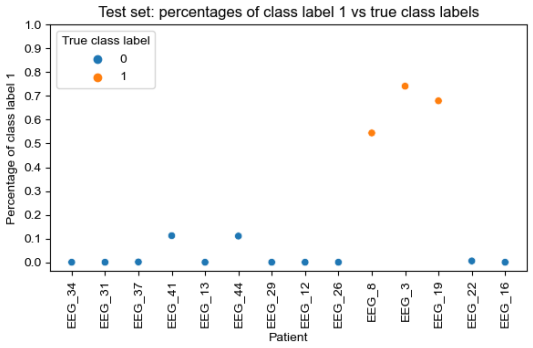
Figure E.5: Predicted and true class labels: default RF and optimized RFs

(a)) Undersampled Z_{sim}: default RF

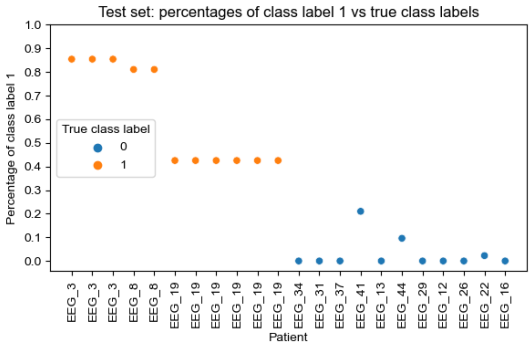
(b)) Undersampled Z_{sim}: optimized RF



(c)) Normal Z_{sim}: default RF



(d)) Normal Z_{sim}: optimized RF



(e)) Oversampled Z_{sim}: default RF

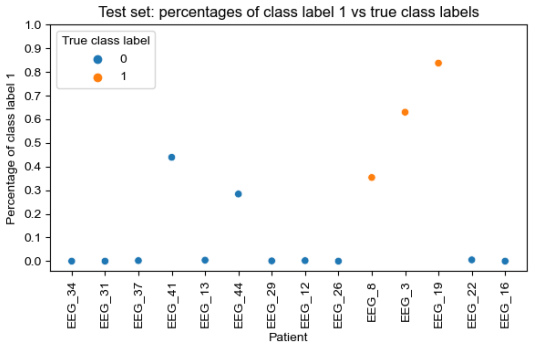
(f)) Oversampled Z_{sim}: optimized RF

Figure E.6: Predicted and true class labels: default RF and optimized RFs

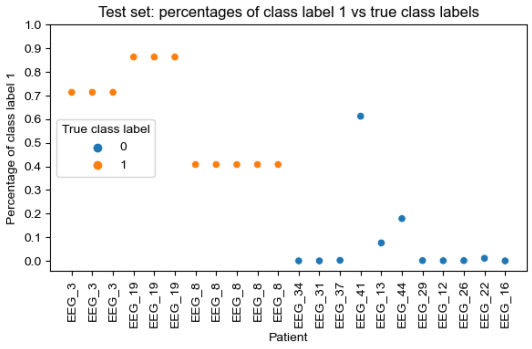
(a)) Undersampled Z_{simPCA}: default RF

(b)) Undersampled Z_{simPCA}: tuned RF

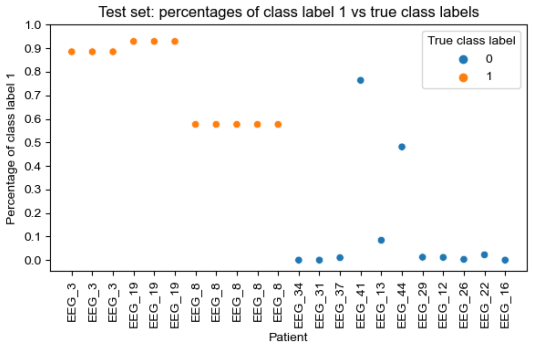
(c)) Normal Z_{simPCA}: default RF



(d)) Normal Z_{simPCA}: tuned RF



(e)) Oversampled Z_{simPCA}: default RF



(f)) Oversampled Z_{simPCA}: tuned RF

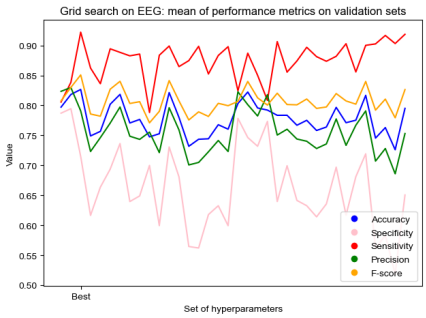
Figure E.7: Predicted and true class labels: default RF and optimized RFs

E.4. Best settings for parameters

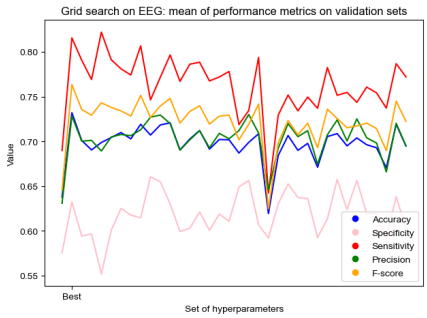
Dataset	`max_depth`	`no_estimators`	`max_features`	splitting rule
Normal F_X	3	100	`None`	Gini
Under F_X	3	100	0.2	entropy
Over F_X	3	100	`None`	Gini
Normal EEG Z	6	10	$\sqrt{p}$	Gini
Under EEG Z	3	20	$\sqrt{p}$	Gini
Over EEG Z	9	50	0.2	entropy
Normal EEG PCA	9	10	$\sqrt{p}$	Gini
Under EEG PCA	9	50	$\sqrt{p}$	entropy
Over EEG PCA	9	50	0.2	Gini

Table E.1: Best parameters from GridSearch per dataset

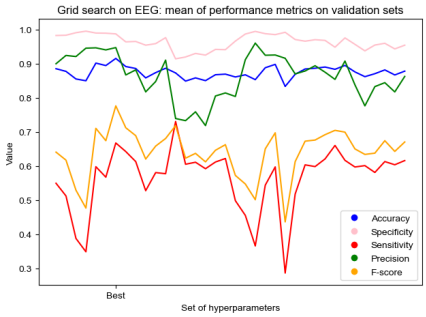
E.5. Grid search: different parameter settings for Z_{sim} and Z_{simPCA}



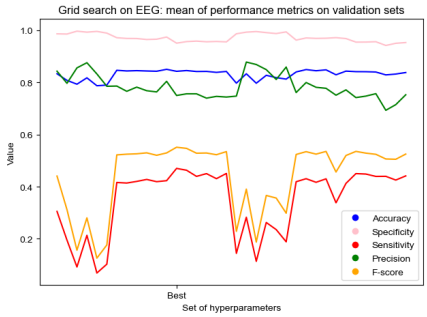
(a)) Undersampled Z_{sim}



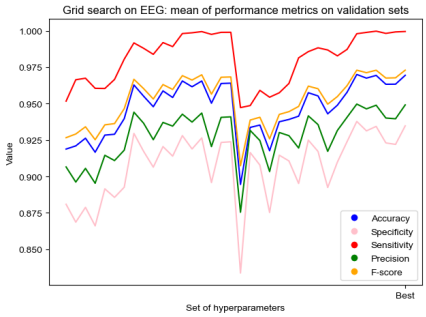
(b)) Undersampled Z_{simPCA}



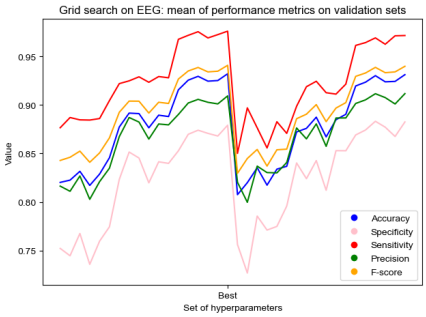
(c)) Normal Z_{sim}



(d)) Normal Z_{simPCA}



(e)) Over Z_{sim}



(f)) Over Z_{simPCA}

Figure E.8: Grid search for both EEG sets, Z_{sim} and Z_{simPCA}. *On the x-axis the ticks are a set of tuning parameters. The mean score of the validation sets of a specific tuning parameters is displayed on the y-axis.*

F

Results

F.1. Effect under- and oversampling

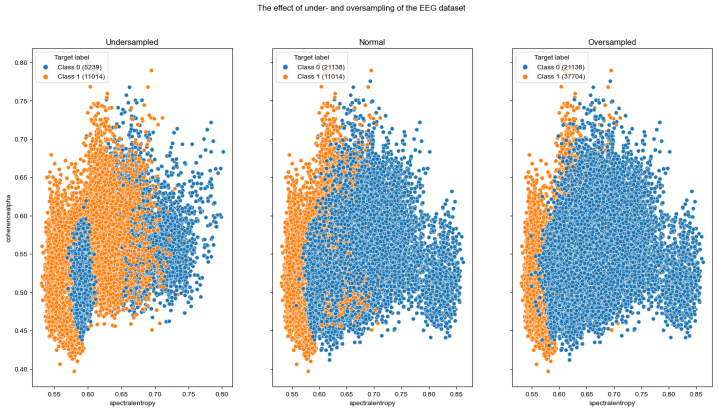


Figure F.1: The effect of under- and oversampling compared to the normal simulated EEG dataset Z (in brackets the no. EEG observations per class)

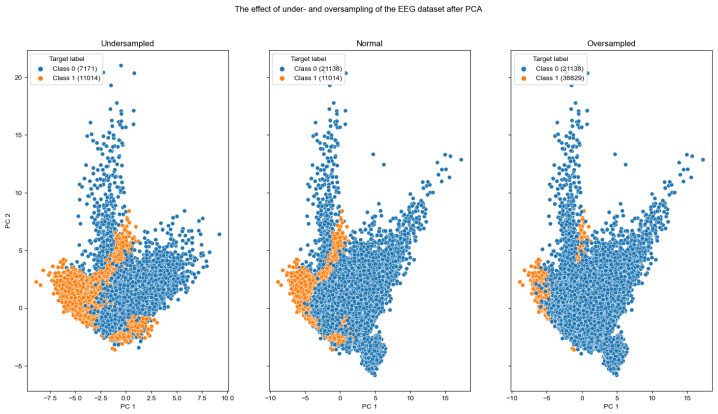
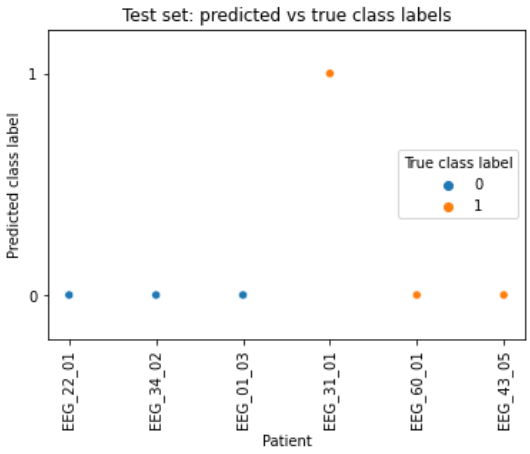


Figure F.2: The effect of under- and oversampling compared to the normal simulated EEG dataset after PCA Z_{PCA}(in brackets the no. EEG observations per class)

F.2. Predicted versus true class labels



(a)) Undersampled F_X: default RF

(b)) Undersampled F_X: tuned RF

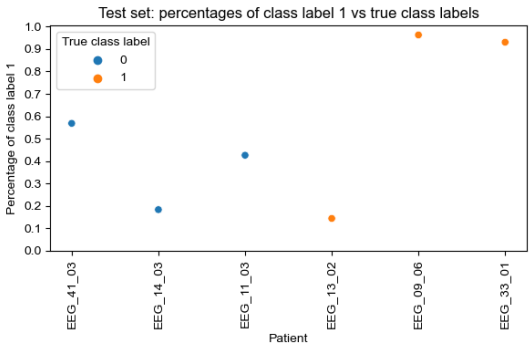
(c)) Normal F_X: default RF

(d)) Normal F_X: tuned RF

(e)) Oversampled F_X: default RF

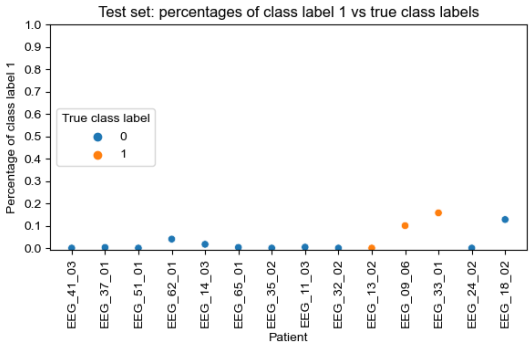
(f)) Oversampled F_X: tuned RF

Figure F.3: Predicted versus true labels: default and tuned RF



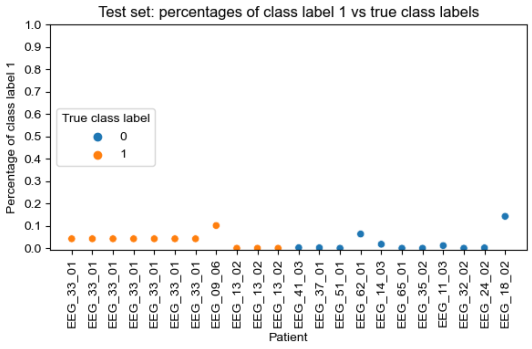
(a)) Undersampled Z: default RF

(b)) Undersampled Z: tuned RF

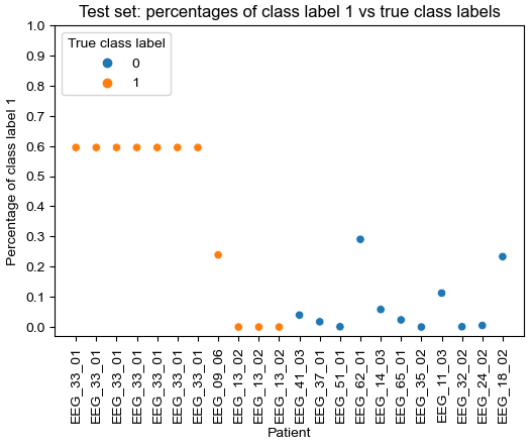


(c)) Normal Z: default RF

(d)) Normal Z: tuned RF



(e)) Oversampled Z: default RF

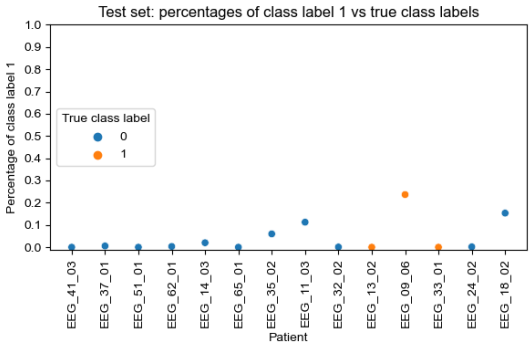


(f)) Oversampled Z: tuned RF

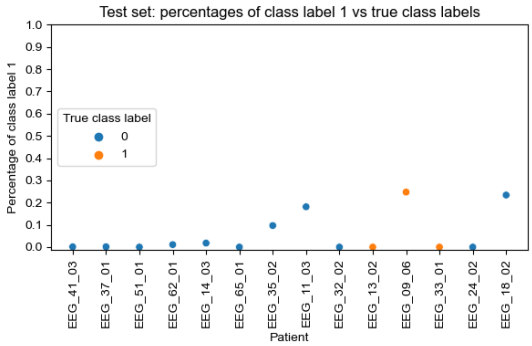
Figure F.4: Predicted versus true labels: default and tuned RF

(a)) Undersampled Z_{PCA}: default RF

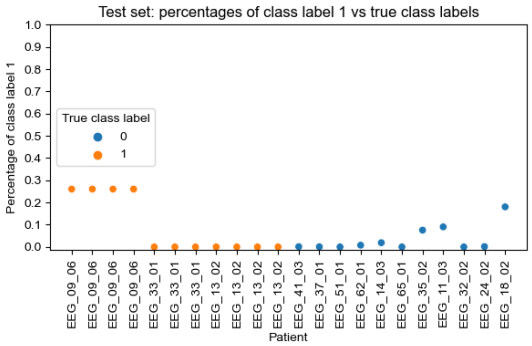
(b)) Undersampled Z_{PCA}: tuned RF



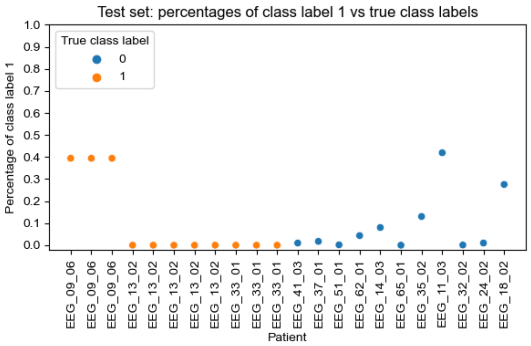
(c)) Normal Z_{PCA}: default RF



(d)) Normal Z_{PCA}: tuned RF



(e)) Oversampled Z_{PCA}: default RF



(f)) Oversampled Z_{PCA}: tuned RF

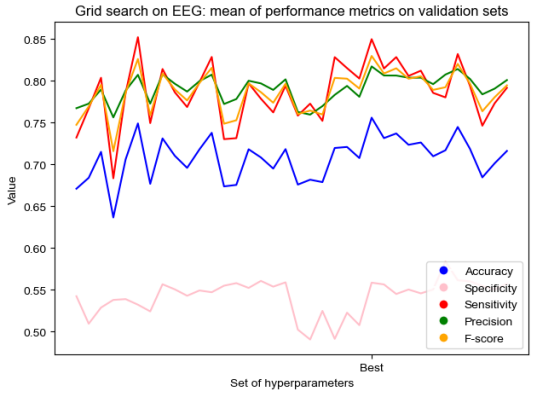
Figure F.5: Predicted versus true labels: default and tuned RF

F.3. Best settings for parameters

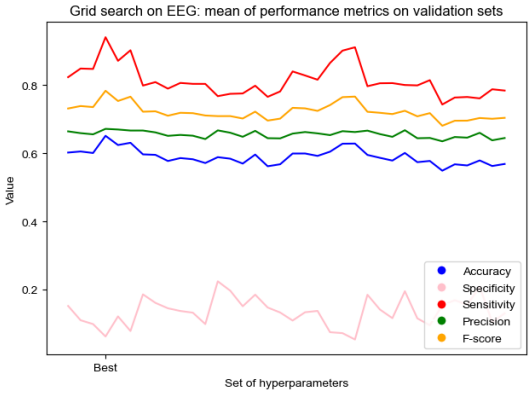
Dataset	`max_depth`	`n_estimators`	`max_features`	splitting rule
Under F_X	9	250	$\sqrt{p}$	entropy
Normal F_X	3	150	$\sqrt{p}$	Gini
Over F_X	`None`	150	$\sqrt{p}$	Gini
Normal EEG Z	6	20	0.2	entropy
Under EEG Z	3	10	0.2	Gini
Over EEG Z	9	50	$\sqrt{p}$	Gini
Normal EEG PCA	6	50	$\sqrt{p}$	entropy
Under EEG PCA	3	10	0.2	Gini
Over EEG PCA	9	50	$\sqrt{p}$	entropy

Table F.1: Best parameters from GridSearch per dataset

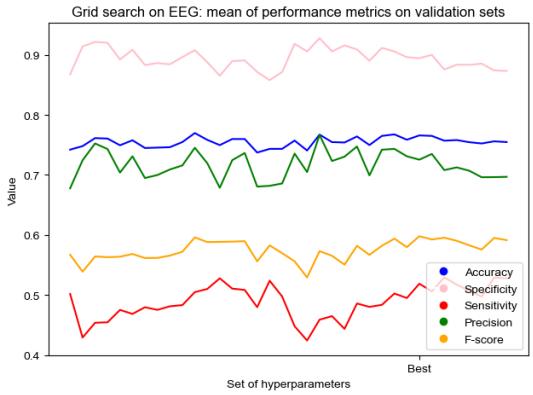
F.4. Grid search: different parameter settings for Z and Z_{PCA}



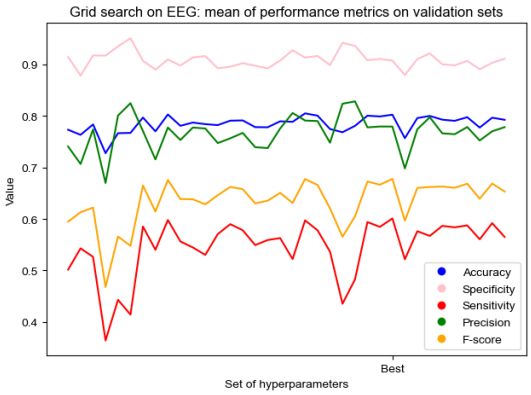
(a)) Undersampled Z



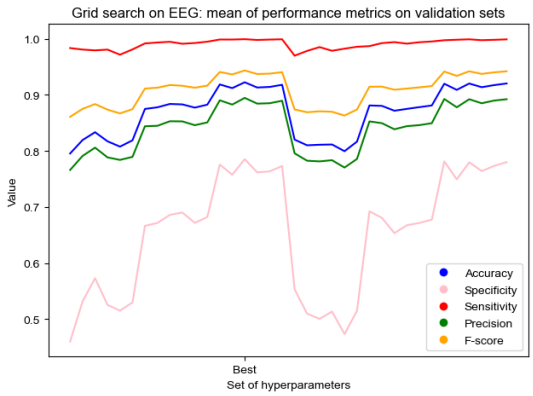
(b)) Undersampled Z_{PCA}



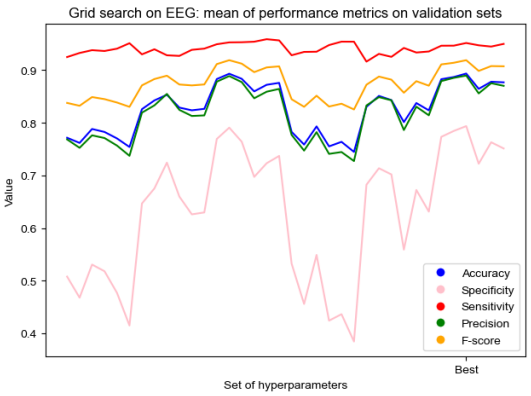
(c)) Normal Z



(d)) Normal Z_{PCA}



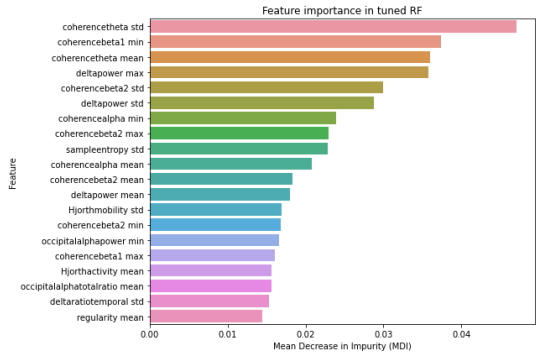
(e)) Oversampled Z



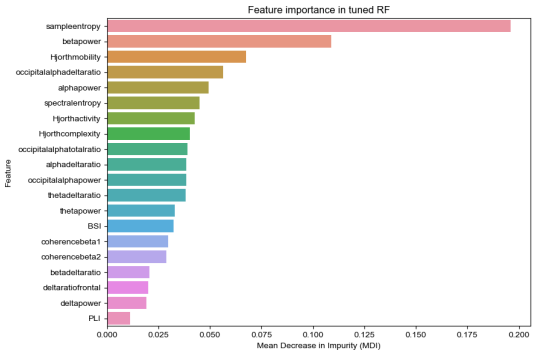
(f)) Oversampled Z_{PCA}

Figure F.6: Grid search on tuning parameters RF for undersampled, normal and oversampled Z and Z_{sim}

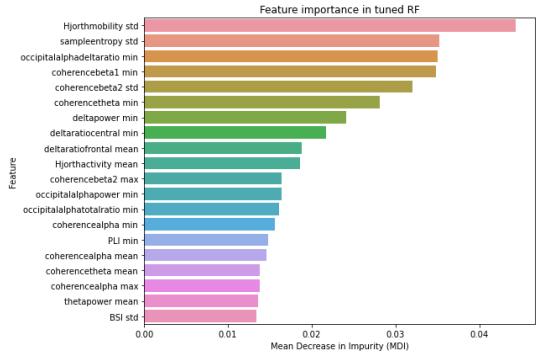
F.5. Feature importance: MDI



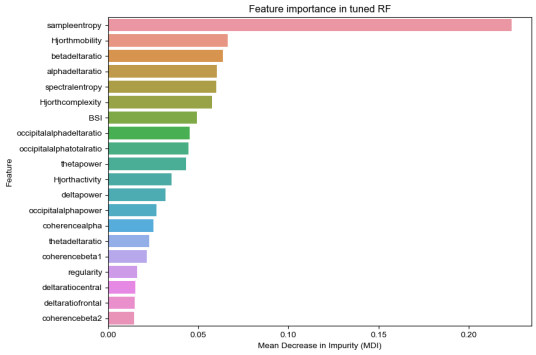
(a)) Normal F_X



(b)) Normal Z



(c)) Oversampled F_X



(d)) Oversampled Z

Figure F.7: Feature importance based on Mean Decrease Impurity: tuned RF