UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Computer Science Curriculum

Madis Masso

Empirical Comparison of Machine Learning Algorithms Based on EEG Data

Bachelor's Thesis (6 ECTS)

Supervisors:
Ilya Kuzovkin, MSc
Kristjan Korjus, MSc

Tartu 2016

Empirical Comparison of Machine Learning Algorithms Based on EEG Data

Abstract:

The aim of this work is to compare different machine learning algorithms in an attempt to find the best one for classifying EEG data. In order to achieve this, the data from ten subjects were classified by ten machine learning algorithms. The algorithms were compared in three ways: Firstly, they were compared by using three performance metrics, secondly, by using clustergrams and lastly, by using corralation matrices. The results from the comparison show that the without parameter optimization, logistic regression model is the most efficient algorithm for classifying EEG data. However, with parameter optimization, random forest is the most efficient algorithm for classifying EEG data.

Keywords: Machine learning algorithms, electroencephalography, statistical analysis, comparison.

Kogemuslik masinõppealgoritmide võrdlus EEG andmete põhjal

Lühikokkuvõte:

Selle töö eesmärgiks on võrrelda erinevaid masinõppealgoritme ning üritada leida nende hulgast parim EEG andmete klassifitseerimise jaoks. Selle saavutamiseks klassifitseeriti 10 inimese andmeid 10 masinõppealgoritmi poolt. Algoritme võrreldi kolmel viisil: esiteks võrreldi neid kolme erineva jõudlust iseloomustava näitaja alusel, teiseks kasutati klasteranalüüsi meetodeid ja dendrogramme ning viimaks kasutati selleks korrelatsioonimaatrikseid. Saadud võrdluse tulemused näitavad, et optimeerimata parameetrite korral on logistilise regressiooni mudel kõige efektiivsem algoritm EEG andmete klassifitseerimisel. Optimeeritud parameetrite korral on kõige efektiivsemaks algoritmiks juhumets.

Võtmesõnad: Masinõppealgoritmid, elektroentsefalograafia, statistiline analüüs, võrdlus.

Table of Contents

Introduction

The aim of this thesis is to compare different machine learning algorithms in an attempt to find the best one for classifying electroencephalography (EEG) data.

In order to achieve this goal, 10 modern machine learning algorithms were chosen for the comparison, such as: AdaBoost [1, 2, 3], k-Nearest Neighbours [4, 5], C4.5 decision tree [6], logistic regression model [7, 8, 9], multilayer perceptron network [10, 11, 12], naive Bayes [13, 14, 15], random forest [16, 17, 18], radial basis function network [19, 20], minimal cost-complexity pruning [21, 22] and sequential minimal optimization algorithm for training a support vector classifier [23, 24, 25, 26, 27].

Although many of these machine learning algorithms have already been reported to give great results, it is still difficult to say which of these algorithms is the best one for classifying EEG data due to differences in performance metrics [28], datasets [29], chosen algorithms [30] and/or the number of classes in each data set.

This paper attempts to find the best machine learning algorithm for classifying EEG data by following these simple rules for comparison:

1. All of the datasets are based on motor imagery and have two classes.
2. All of the datasets are evaluated single-trial - this means no averaging is done across multiple trials.
3. Three performance metrics are used for comparison: classification accuracy, time taken to build the model and F-score.
4. Classification results for each machine learning algorithm and dataset are evaluated using 10-fold cross-validation.

The paper is organized as follows: Chapter 1 presents the reader with the theoretical background information, Chapter 2 describes the datasets and the classifiers, the results are provided in Chapter 3 and the conclusions are given in Chapter 4.

Chapter 1

Theoretical Background

The goal of this chapter is to provide some background information and to explain to the reader what is electroencephalography and machine learning, how are they used and what are they used for.

1.1 Electroencephalography (EEG)

Electroencephalography, a method for recording the electrical activity of the brain, was invented by a German neurologist named Hans Berger in 1929 [32].

The device, which measures voltage fluctuations resulting from ionic current flows within the neurons of the brain, is called an electroencephalograph (see Figure 1.1). An electroencephalogram, on the other hand, is the graph which shows the bioelectrical activity of the brain.

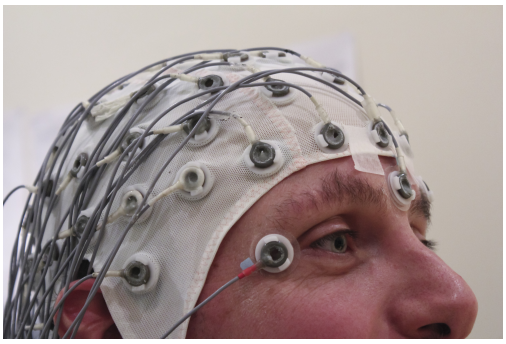

Figure 1.1: An EEG cap with 32 electrodes [31].

An example of an EEG recording can be seen on Figure 1.2. Due to the similar nature of all these terms, the abbreviation EEG may stand for electroencephalography, electroencephalograph and also electroencephalogram.

The recording of the brain's spontaneous electrical activity is done by placing multiple electrodes on the scalp.

Since EEG has a very high *temporal resolution* (precision of a measurement with respect to time) on the order of milliseconds rather than seconds, it is commonly recorded at *sampling rates* between 250 and 2000 Hz in clinical and research settings. For instance, a sampling rate of 1000 Hz would mean that the number of samples of EEG data being recorded per second is 1000 [33].

EEG is most often used to diagnose epilepsy [34], sleep disorders [35], coma [36], encephalopathies [37] and brain death [38]. It is also used extensively in neuroscience [39], cognitive science [40], cognitive psychology [41], neurolinguistics [42] and psychophysiological research [43].

EEG, the invention, has been described "as one of the most surprising, remarkable, and momentous developments in the history of clinical neurology" [44].

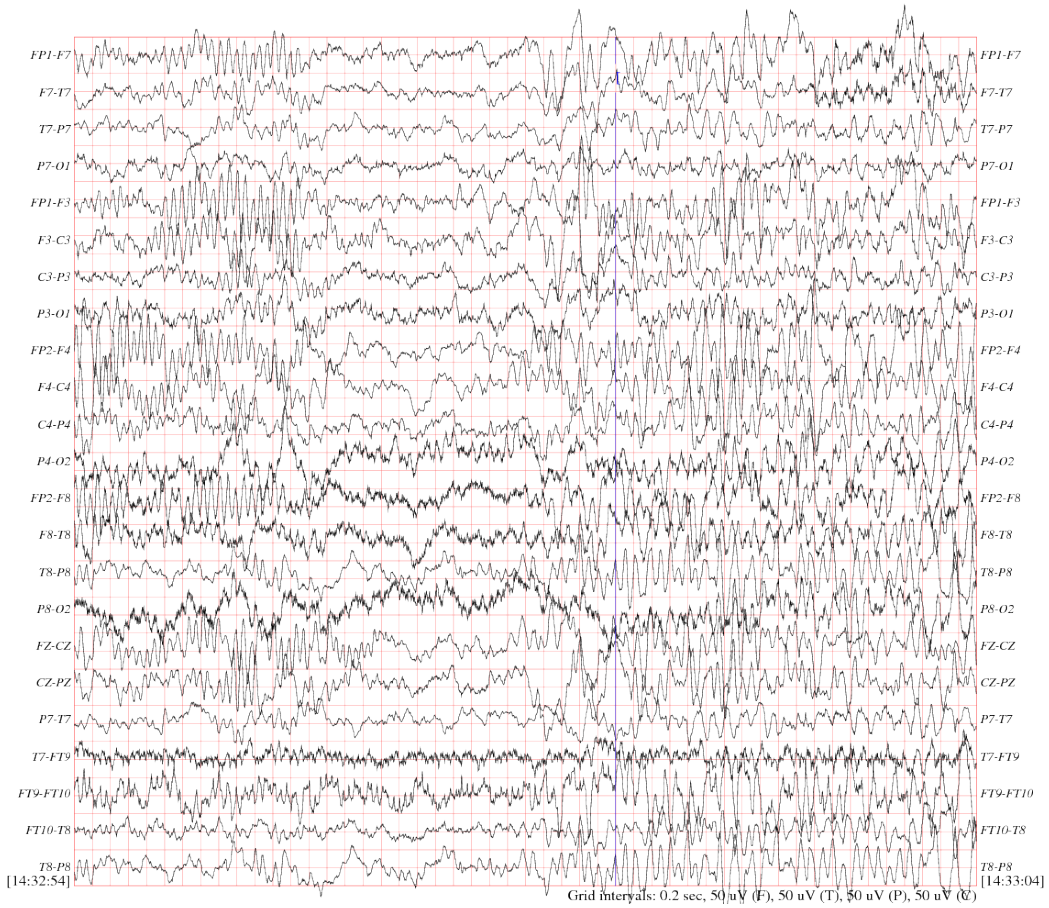

Figure 1.2: Example EEG showing the electrical activity of the brain of a pediatric subject suffering from epileptic seizures [45].

1.2 Machine Learning

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. It concentrates on algorithms that can learn from data by generalizing from experience [46].

Generalization in this context is the ability of a machine learning algorithm to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases. This is the main difference between machine learning algorithms and algorithms, which simply follow a predefined set of instructions to solve problems [47].

They can be used to tackle numerous types of tasks such as predicting user preferences for movie recommendations [48], foreseeing economic movements for example the next financial crisis [49] or even revealing previously unknown influences between famous painters [50].

1.2.1 Supervised Learning

Supervised learning is a subcategory of machine learning, where the learning algorithm is presented with labeled training data (data that has class values), which consists of example inputs and their desired outputs, and the goal is to learn a general rule that maps inputs to outputs [51].

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine what kind of data will be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and

corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the feature representation of the data. The accuracy of the supervised learning algorithm depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, but should contain enough information to accurately predict the output.

4. Determine what type of learning algorithm to use. For example, the engineer may choose to use support vector machines or decision trees.

5. Run the learning algorithm on the training set. Most supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset of the training set called validation set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set and the validation set.

A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems [52].

1.2.2 Unsupervised Learning

Unsupervised learning is a subcategory of machine learning, where the learning algorithm is presented with unlabeled training data, leaving it on its own to find the structure in its input. Unsupervised learning can be a goal in itself — for example, it can be used to discover hidden patterns in data [51].

Since the examples given to the learning algorithm are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning. Unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data [53].

1.3 Optimization

In mathematics and computer science, optimization is the selection of a best element (with regard to some criteria) from some set of available alternatives [54].

In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function.

Optimization is very closely tied to machine learning due to the fact that most machine learning problems reduce to optimization problems. By using optimization techniques on the paramaters of each machine learning algorithm, a mathematically optimal solution for a (classification) problem can be found.

In subsection 2.2.1, this thesis makes use of SMAC (Sequential Model-based Algorithm Configuration [55]) and AutoWEKA (a Program for Combined Selection and Hyperparameter Optimization of Classification Algorithms [56]) for the purpose of optimizing the aforementioned machine learning algorithms.

Chapter 2

Materials and Methods

This chapter describes the datasets chosen for this thesis, explains how the data was preprocessed and gives a brief overview for each of the chosen machine learning algorithms.

2.1 The Datasets

In order to follow all of the rules that were described in the introduction of this thesis, the following three datasets were chosen from the BCI Competitions:

1. BCI Competition III Dataset IVa [57, 58]
2. BCI Competition III Dataset IVb [58, 59]
3. BCI Competition IV Dataset 1 [60, 61]

All of the datasets had a sampling rate of 1000Hz and contained the following information:

- matrix containing the data of continuous EEG signals. For example, the original dimensions of the matrix from Dataset 2: BCI Competition III were 2102599 (milliseconds) x 118 (electrodes).
- table containing information about the timings of *cues* (visual prompt given to a subject, which means he/she has to perform an activity, for example, to move their left hand).

- table containing information about the structure of the data.

The final dataset consists of 10 individual datasets, 1 for each subject. The first 5 datasets/subjects are obtained from Dataset 1: BCI Competition III Dataset IVa, 1 dataset/subject from Dataset 2: BCI Competition III Dataset IVb and 4 datasets/subjects from Dataset 3: BCI Competition IV Dataset 1.

2.1.1 Dataset 1: BCI Competition III Dataset IVa

Dataset 1 contains the continuous signals of 118 EEG channels and markers that indicate the time points of 280 cues for each of the 5 subjects (aa, al, av, aw, ay). For some markers no target class information was provided (value `NaN`) for competition purpose. These markers were not used as a part of this thesis.

The dataset is well balanced - after the preprocessing process the total number of samples is 1680. Out of those 1680 samples, 846 have a class value of 0 and 834 a class value of 1, which means the ratio between the two classes is almost 50 : 50 (50.3571 : 49.6429).

The dataset was recorded from five healthy subjects and contains only the data from the 4 first sessions. Subjects sat in a comfortable chair with arms resting on armrests. Visual cues indicated for 3.5 s which of the following 2 motor imageries the subject should perform: (R) right hand, (F) right foot. The presentation of target cues were intermitted by periods of random length, 1.75 to 2.25 s, in which the subject could relax (see Figure 2.1 for an illustration of the experimental setup).

There were two types of visual stimulation: (1) where targets were indicated by letters appearing behind a fixation cross, and (2) where a randomly moving object indicated targets (inducing target-uncorrelated eye movements).

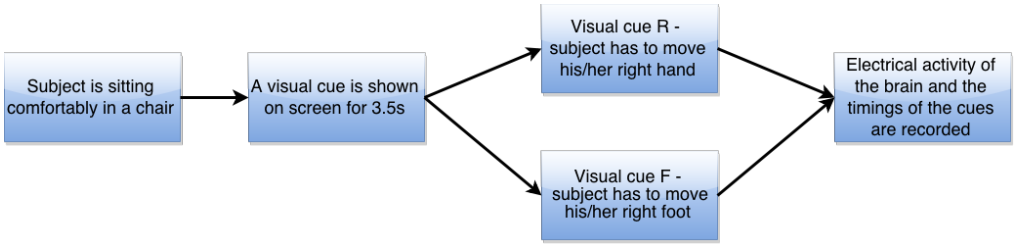

Figure 2.1: Graph illustrating the process of recording Dataset 1.

2.1.2 Dataset 2: BCI Competition III Dataset IVb

Dataset 2 contains the continuous signals of 118 EEG channels and, for the training data, markers that indicate the time points of 210 cues and the corresponding target classes. Only cues for the classes left and foot are provided for the competition.

The dataset is well balanced - after the preprocessing process the total number of samples is 627. Out of those 627 samples, 315 have a class value of 0 and 312 a class value of 1, which means the ratio between the two classes is almost 50 : 50 (50.2392 : 49.7608).

The dataset was recorded from one healthy subject and contains only data from the 7 first sessions. He sat in a comfortable chair with arms resting on armrests. The first 3 sessions are given with labels as training set. Visual cues (letter presentation) indicated for 3.5 seconds which of the following 2 motor imageries the subject should perform: (L) left hand, (F) right foot. The presentation of target cues were intermitted by periods of random length, 1.75 to 2.25 seconds, in which the subject could relax.

Because the continous EEG signals of sessions 4 to 7 were given without any cue information (neither target class nor timing) as a test set, these sessions were not used in this thesis.

2.1.3 Dataset 3: BCI Competition IV Dataset 1

Dataset 3 contains the continuous signals of 59 EEG channels and, for the calibration data, markers that indicate the time points of cue presentation and the corresponding target classes.

The dataset was recorded from four healthy subjects. For each subject two out of three classes of motor imagery were selected (the classes being as follows: left hand, right hand and foot. The side being used was chosen by the subject).

The dataset is well balanced - after the preprocessing process the total number of samples is 2388. Out of those 2388 samples, 1188 have a class value of 0 and 1200 a class value of 1, which means the ratio between the two classes is almost 50 : 50 (49.7487 : 50.2513).

The first two runs were used as calibration data. Arrows pointing left, right, or down were presented as visual cues on a computer screen. Cues were displayed for a period of 4 s during which the subject was instructed to perform the cued motor imagery task. These periods were interleaved with 2 s of blank screen and 2 s with a fixation cross shown in the center of the screen. The fixation cross was superimposed on the cues, i.e. it was shown for 6 s. These data sets are provided with complete marker information.

The following four runs were used as evaluation data. Due to the fact that these runs did not contain any markers or cue information, they were not used in this thesis.

2.2 Data Preprocessing

The preprocessing of each dataset consisted of the following steps:

1. The `.mat` file that contains the dataset is loaded into Matlab environment.
2. In order to use the original microvolt (µV) values of the EEG recording, the data is converted from datatype `INT16` into `DOUBLE`.
3. 16 EEG channels are chosen for use (the rest of the channels were removed to reduce the size of the datasets):

   `AF3, AF4, F3, F4, FC5, FCz, FC6, C5, C6, CP5, CPz, CP6, P5, P6, O1` and `O2` (illustrated in Figure 2.2) These specific EEG channels were chosen in an attempt to cover the maximum area of the subjects' scalp (and therefore achieve the greatest variance possible in EEG signals).
4. Since this thesis only focuses on supervised machine learning algorithms which require labeled samples, unlabeled samples are removed.
5. Each sample is divided into three equally sized subsamples.
6. The `detrend` function is used on each subsample. This subtracts the mean or a best-fit line (in the least-squares sense) from the data. This is done in order to emphasise short-term changes that otherwise might have gone unnoticed.

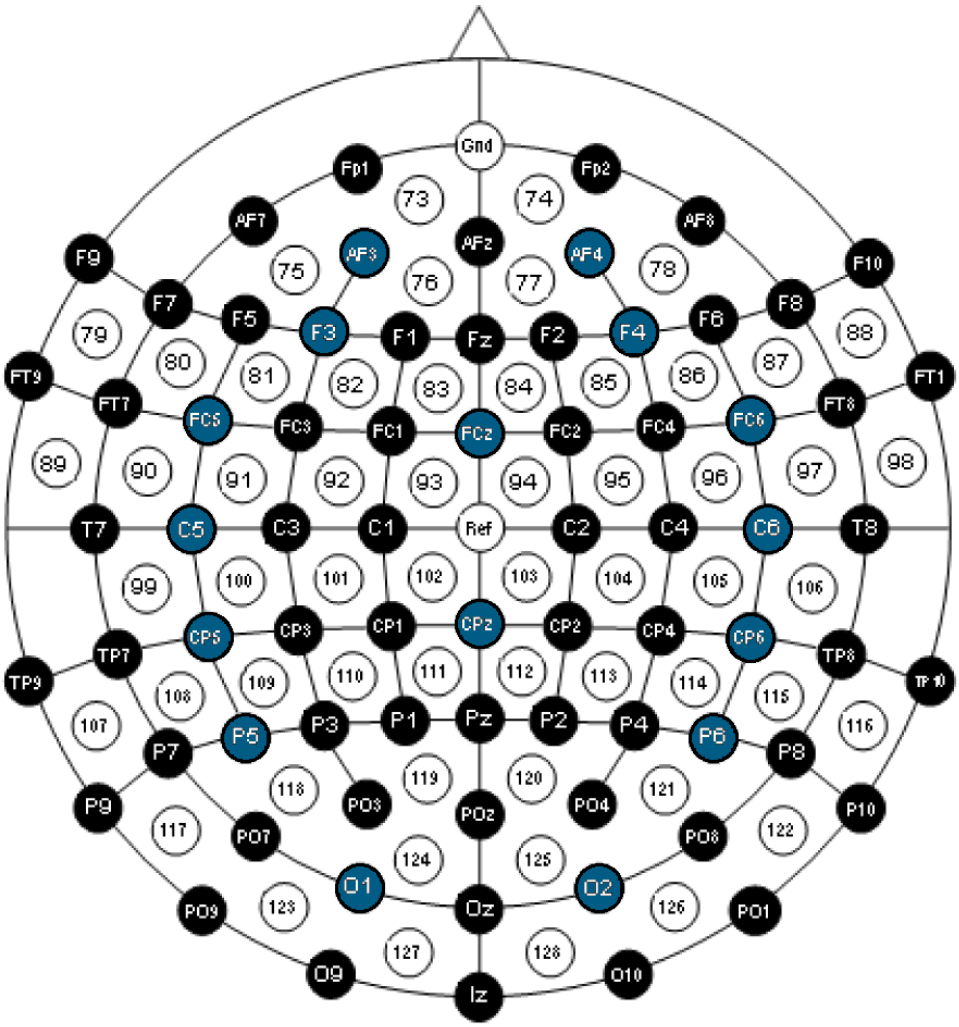

Figure 2.2: Layout of the 16 EEG channels that were chosen (in blue).

7. Fourier transform (FT) algorithm is used on each subsample (see subsection 2.2.1)

8. Each subsample is divided into the following 7 frequency bands:

   (a) 1 - 3 Hz (Delta)

   (b) 4 - 7 Hz (Theta)

   (c) 8 - 15 Hz (Alpha)

   (d) 16 - 31 Hz (Beta)

   (e) 32 - 50 Hz (Low-Gamma)

   (f) 51 - 80 Hz (Medium-Gamma)

   (g) 81 - 120 Hz (High-Gamma)

9. Mean values are found for each frequency band.

10. The data is shuffled and saved.

11. The data is converted from `.mat` to `.arff` in order to use the machine learning classifiers that are implenented in Weka 3 (data mining software) [62] and AutoWEKA.

2.2.1 Fourier Transform

Because of the fact that the voltage fluctuations in the brain (also known as brain waves) are periodical, they can be decomposed into *frequency components* that make them up. This is done by using Fourier transform, a method created by Joseph Fourier in 1822, who showed that almost all functions could be written as an infinite sum of *harmonics* [63].

This thesis uses Fast Fourier Transform (FFT) to compute the Discrete Fourier Transform (DFT) in order to convert a signal from time domain (EEG recording) to the frequency domain. As a result of this transformation the *frequency components* of the EEG signal are found.

The Discrete Fourier Transform (DFT) is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, ..., N-1,$$

where N is the amount of values which are recorded in the predefined amount of time; x_n is the value which is being recorded when time is equal to n; k is the current frequency; X_k is the energy of the current frequency k.

An example of a signal before and after Fourier Transform can be seen in figures 2.3 and 2.4 correspondingly.

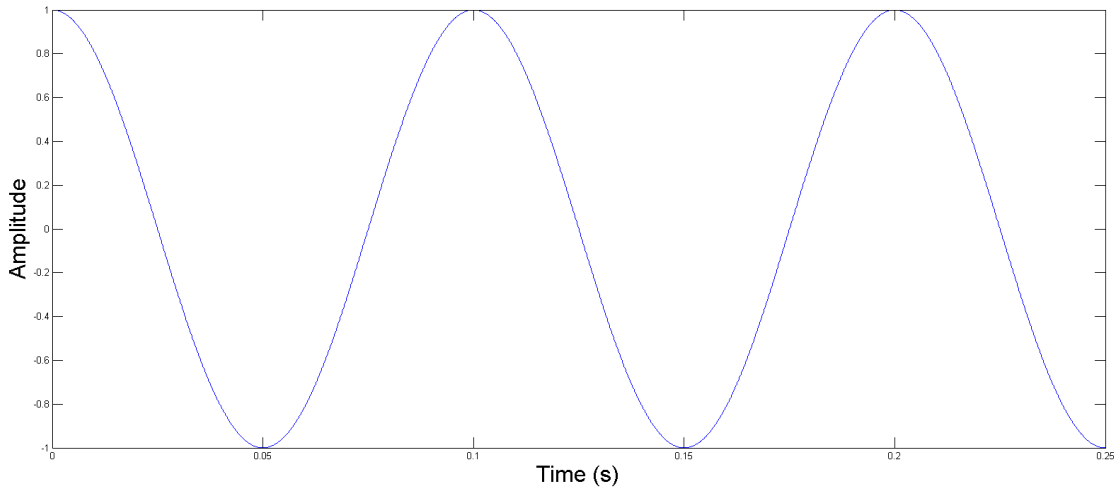

Figure 2.3: An example of a signal of frequency 10 Hz before Fourier Transform.

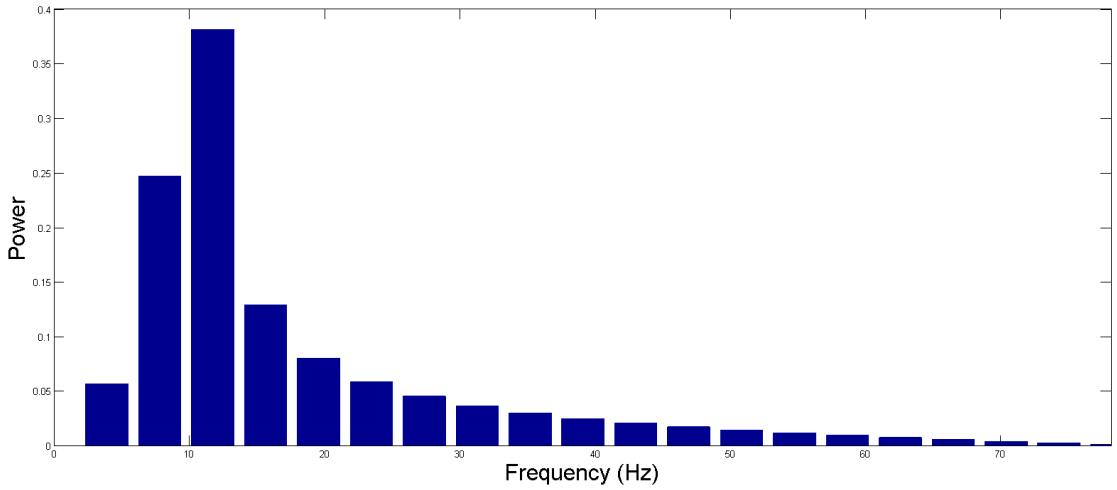

Figure 2.4: An example of a signal of frequency 10 Hz after Fourier Transform.

2.3 Classification

In machine learning and statistics, the goal of classification is to use object's characteristics to identify which class (or group) it belongs to. In supervised learning, this is achieved by the use of a classifier - an algorithm which has to be trained on labeled training examples to be able to distinguish new unlabeled examples between a fixed set of classes [64].

This thesis focuses on the classification problem of EEG data from BCI competitions.

In order to achieve this, from each dataset, subject and sample, features are extracted, which are then used to train the classifiers. Then a feature vector is formed for each of the classifiers. These feature vectors are then used as the representation of the corresponding sample. In the testing phase, each trained classifier predicts the class label according to the extracted feature vector which was obtained from the corresponding test sample.

AdaBoost, k-Nearest Neighbours, C4.5 decision tree, logistic regression, multilayer perceptron network, Naive Bayes, random forest, radial basis function network, minimal cost-complexity pruning and support vector classifiers were trained on 5 subjects from Dataset 1, 1 subject from Dataset 2 and 4 subjects from Dataset 3.

In the following subsections, all of the previously chosen classifiers are briefly described and reviewed.

2.3.1 AdaBoost Classifier

AdaBoost (AB)[1], short for "Adaptive Boosting", is a machine learning *meta-algorithm* formulated by Yoav Freund and Robert Schapire [1, 2].

It can be used in conjunction with many other types of learning algorithms to improve the overall result. The outputs of the other learning algorithms are combined into a weighted sum that represents the final output of the boosted classifier.

AdaBoost is adaptive in the sense that subsequent weak learners (a classifier which is only slightly correlated with the true classification, but can still label examples better than random guessing) are tweaked in favor of those instances misclassified by previous classifiers. It is sensitive to noisy data and outliers. In some problems, however, it can be less susceptible than other learning algorithms to the overfitting problem, which occurs when a statistical model describes random error or noise instead of the underlying relationship.

The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (i.e., their error rate is smaller than 0.5 for binary classification) and they succeed each on a different subset of samples, the final model can be proven to converge to a strong learner [2].

[1] AdaBoostM1

This thesis uses AdaBoost in conjunction with the decision stump classifier, which is a machine learning model consisting of a one-level decision tree. It is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its leaves). A decision stump makes a prediction based on the value of just a single input feature [3].

AdaBoost can also be used in conjunction with many other classifiers, such as RandomForest, RandomTree or C4.5 decision tree.

2.3.2 k-Nearest Neighbours Classifier

The k-Nearest Neighbors algorithm (KNN)[2] is a method used for classification [4].

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small and arbitrarily chosen) [5].

An example of k-NN classification can be seen in Figure 2.5.

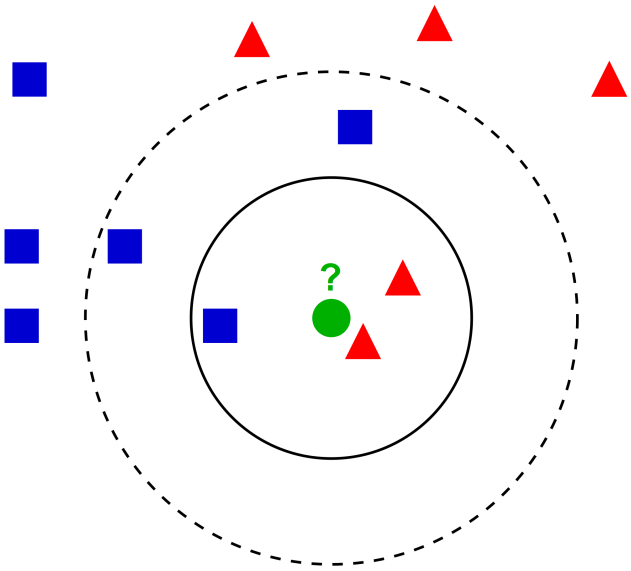

Figure 2.5: Example of k-NN classification. If k = 1, the green circle or the current test sample is classified as the class of its nearest neighbor. If k = 3 (solid line circle), it is classified as a red triangle because there are more triangles than squares inside the inner circle. If k = 5 (dashed line circle), it is classified as a blue square (3 squares vs. 2 triangles inside the outer circle) [65].

[2] IBk

2.3.3 C4.5 Decision Tree

C4.5 (C45)[3] is an algorithm used to generate a decision tree developed by Ross Quinlan. It's an extension of Quinlan's earlier ID3 (Iterative Dichotomiser 3) algorithm [66].

C4.5 builds decision trees from a set of training data, using the concept of information entropy. The training data is a set $S = s_1, s_2, ...$ of already classified samples. Each sample s_i consists of a p-dimensional vector $(x_{1,i}, x_{2,i}, ..., x_{p,i})$, where the x_j represent attributes or features of the sample, as well as the class in which s_i falls.

General algorithm for building a decision tree:

1. Check for base cases.
   (a) If all of the samples in the list belong to the same class, C4.5 creates a leaf node for the decision tree saying to choose that class.
   (b) If none of the features provide any information gain, C4.5 creates a decision node higher up the tree using the expected value of the class.
   (c) If an instance of previously-unseen class is encountered, C4.5 creates a decision node higher up the tree using the expected value.
2. Find the normalized information gain ratio from splitting on each of the attributes.
3. Create a decision node that splits on the attribute which has the highest normalized information gain (a measurement, which shows the amount of information gained by doing the split using that particular feature).
4. Recur on the sublists obtained by splitting on the attribute that has the highest normalized information gain.
5. Add the nodes found in step 4 as children of the decision node that was created in step 3 [6].

[3] J48

2.3.4 Logistic Regression

Logistic regression (LR)[4] is a model that was developed by statistician D. R. Cox in 1958 [7]. It models the relationship between a dependent and one or more independent variables, making it possible to look at the fit of the model as well as at the significance of the relationships which are being modelled.

These relationships between variables are measured by estimating probabilities. Logistic regression is used widely in many fields, including the medical and social sciences [9].

For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed using logistic regression [8]. The technique can also be used in engineering, especially for predicting the probability of failure of a given process, system or product [67, 68].

2.3.5 Multilayer Perceptron Network

A Multilayer Perceptron Network (MPN)[5] is an artificial neural network model that maps sets of input data onto a set of appropriate outputs. Multilayer Perceptron Network utilizes a supervised learning technique called backpropagation for training the neural network model [10].

It consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a non-linear activation function. This means that the output cannot be reproduced from a linear combination of inputs. Without a non-linear activation function in the network, a neural network would behave just like a single perceptron (because no matter how many layers the neural network had, summing the layers would only give us another linear function).

A multilayer perceptron is a modification of the standard linear perceptron, it can distinguish data that is not linearly separable [11]. Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result [12, 10].

[4] Logistic
[5] MultilayerPerceptron

2.3.6 Naive Bayes Classifier

Naive Bayes (NB)[6] classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features [13].

For example, a fruit may be considered to be a tangerine if it is orange, round, and about 5 cm in diameter. A Naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is a tangerine, regardless of any possible correlations between the color, roundness and diameter features.

The usage of Naive Bayes classifiers has proven to be a popular method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate) with word frequencies as the features.

With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines [14]. It has also found application in automatic medical diagnosis [15].

An example of classification with a Naive Bayes classifier can be seen in Figure 2.6.

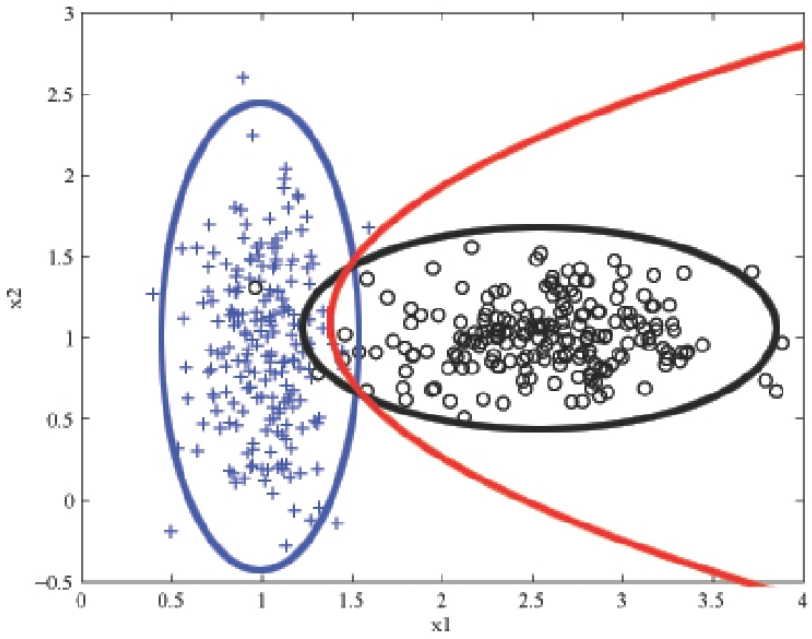

Figure 2.6: 2D binary classification with Naive Bayes. A density contour is drawn for the Gaussian model of each class and the decision boundary is shown in red. [69].

[6]NaiveBayes

2.3.7 Random Forest

Random forests (RF)[7] are an ensemble learning method that was developed by Leo Breiman and Adele Cutler for classification. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees [16].

Random forests try to correct for decision trees' habit of overfitting to their training set. The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho [17] and Amit and Geman [18] in order to construct a collection of decision trees with controlled variance.

2.3.8 Radial Basis Function Network

A radial basis function network (RFN)[8] is an artificial neural network which was first formulated in a 1988 paper by Broomhead and Lowe [19].

It uses radial basis functions (real-valued functions whose values depend only on the distance from the origin) as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters.

Radial basis function networks have many uses, including function approximation, time series prediction, classification, and system control [19, 20].

2.3.9 Minimal Cost-Complexity Pruning

Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and can improve predictive accuracy by the reduction of overfitting [21].

Minimal cost-complexity pruning (MCP)[9] generates a series of trees $T_0, ..., T_m$ where T_0 is the initial tree and T_m is the root alone. At step i the tree is created by removing a subtree from tree $i-1$ and replacing it with a leaf node with value

[7] RandomForest
[8] RBFNetwork
[9] SimpleCart

chosen as in the tree building algorithm (see the general algorithm for building a decision tree at subsection 2.3.3). The subtree that is removed is chosen as follows:

1. Define the error rate of tree T over data set S as $err(T, S)$.
2. The subtree that minimizes $\frac{err(prune(T,t),S)-err(T,S)}{|leaves(T)|-|leaves(prune(T,t))|}$ is chosen for removal.
3. The function $prune(T, t)$ defines the tree gotten by pruning the subtrees t from the tree T.
4. Once the series of trees has been created, the best tree is chosen by generalized accuracy as measured by a training set or cross-validation [22].

2.3.10 Sequential Minimal Optimization Algorithm for Training a Support Vector Classifier

Support vector machines (SVM)[10], are supervised learning models that analyze data and recognize patterns, used for classification and regression analysis [23, 24, 25].

Given a set of training examples, each marked as belonging to one of two categories, a SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible (see Figure 2.7). New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

There exist several specialized algorithms for quickly solving the quadratic programming optimization problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more-manageable chunks.

One of these specialized algorithms is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that may be solved analytically, eliminating the need for a numerical optimization algorithm [26, 27].

[10] SMO

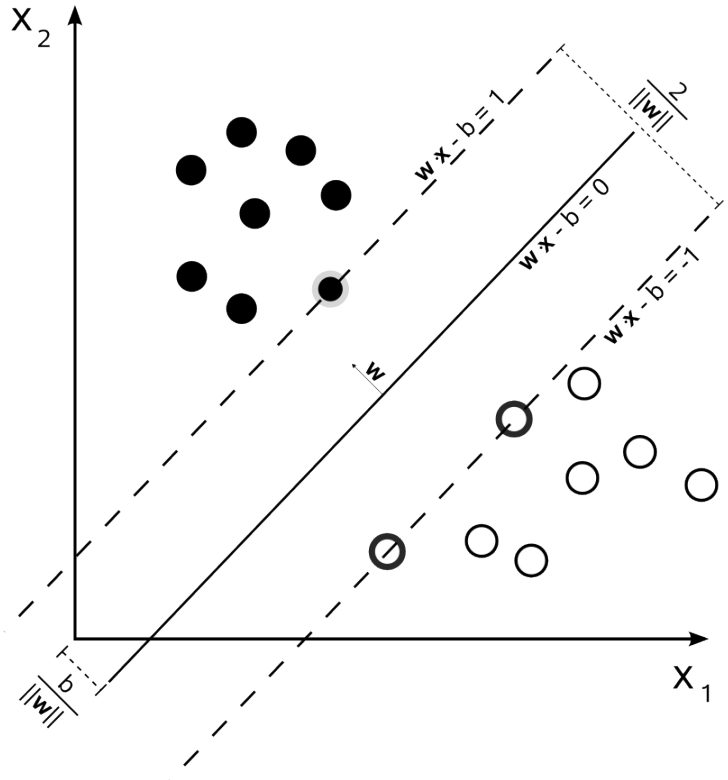

Figure 2.7: Example of classification by using the SMO algorithm. The figure shows an SVM trained with samples from two classes (black circles and white circles). Samples on the margin (dotted line) are called the support vectors [70].

2.4 Performance Metrics Used for the Comparison of the Classifiers

This section explains the three performance metrics used to compare the classifiers of the machine learning algorithms/methods: classification accuracy (CA), time taken to build a model (Time) and F-score.

2.4.1 Classification Accuracy

By defining class labels of the binary (two-class) prediction problem as positive and negative, each classifier has the following four possible outcomes:

- True positive (TP): The number of positive samples correctly predicted.
- True negative (TN): The number of negative samples correctly predicted.

- False positive (FP): The number of negative samples incorrectly predicted as positive samples.
- False negative (FN): The number of positive samples incorrectly predicted as negative samples.

Classification accuracy (CA) is defined as the percentage of the number of samples classifed correctly in the test set over the total samples and it is calculated by [71]:

$$CA = \frac{TP + TN}{TP + TN + FP + FN} * 100 \tag{2.1}$$

2.4.2 Time Taken to Build a Model

For each subject and classifier, the time taken to build a model was recorded. The runtime experiments were conducted on a PC with Intel(R) Core(TM) i7 CPU 860 @ 2.80 GHz, 12 GB 1333Mhz RAM.

2.4.3 F-score

F-score also known as F-Measure or F1-score is a combined performance metric, which considers both the precision p and the recall r of the test to compute a score: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned [72, 73].

It is calculated by:

$$F_1 = \frac{2 * p * r}{p + r}, \tag{2.2}$$

where

$$p = \frac{TP}{TP + FP}; \quad r = \frac{TP}{TP + FN} \tag{2.3}$$

Chapter 3

Results

3.1 Comparison by Using Performance Metrics

Over all of the BCI Competition Datasets, only the data from 10 subjects was used in this thesis (5 subjects from the Dataset 1: BCI Competition III Dataset IVa, 1 subject from the Dataset 2: BCI Competition III Dataset IVb and 4 subjects (1a, 1b, 1f, 1g) from the Dataset 3: BCI Competition IV Dataset 1).

The data from these subjects was tested with 10 supervised learning algorithms using all of the features for each subject. All of the classifiers were trained by using the default values - this means none of the classifiers were specifically optimized for this type of two-classed EEG data.

The results of the classifiers in terms of three metrics including CA (classification accuracy), Time (time taken to build a model) and F-score (or F1-score) for all of the subjects can be found in Figure 3.1, Figure 3.2 and Figure 3.3 respectively.

S1 to S10 represent the subjects from all of the datasets: S1 to S5 are the subjects from Dataset 1, S6 the subject from Dataset 2 and S7 to S10 the subjects from Dataset 3.

Acronyms explained: AB - AdaBoost; KNN - k-Nearest Neighbours; C45 - C4.5 decision tree; LR - logistic regression; MPN - multilayer perceptron network; NB - naive Bayes; RF - random forest; RFN - radial basis function network; MCP - minimal cost-complexity pruning; SVM - sequential minimal optimization algorithm for training a support vector classifier.

The averages of the metrics are given in Figure 3.4, Figure 3.5 and Figure 3.6.

In case of the Subject 2 from the Dataset 1 (S2), the logistic regression (LR) classifier provided the best CA and F-score performance which are 82.14% and 0.821, respectively. The best time performance was obtained from the k-Nearest Neighbours classifier (KNN), which managed to build a model in 0.00s for each of the subjects.

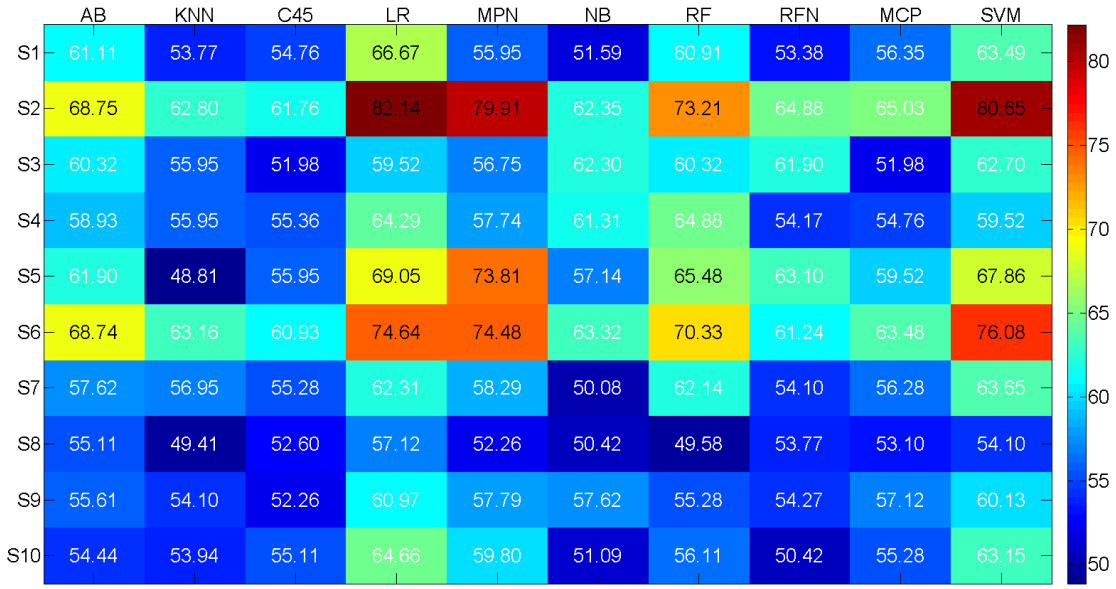

Figure 3.1: Classification accuracy for all subjects.

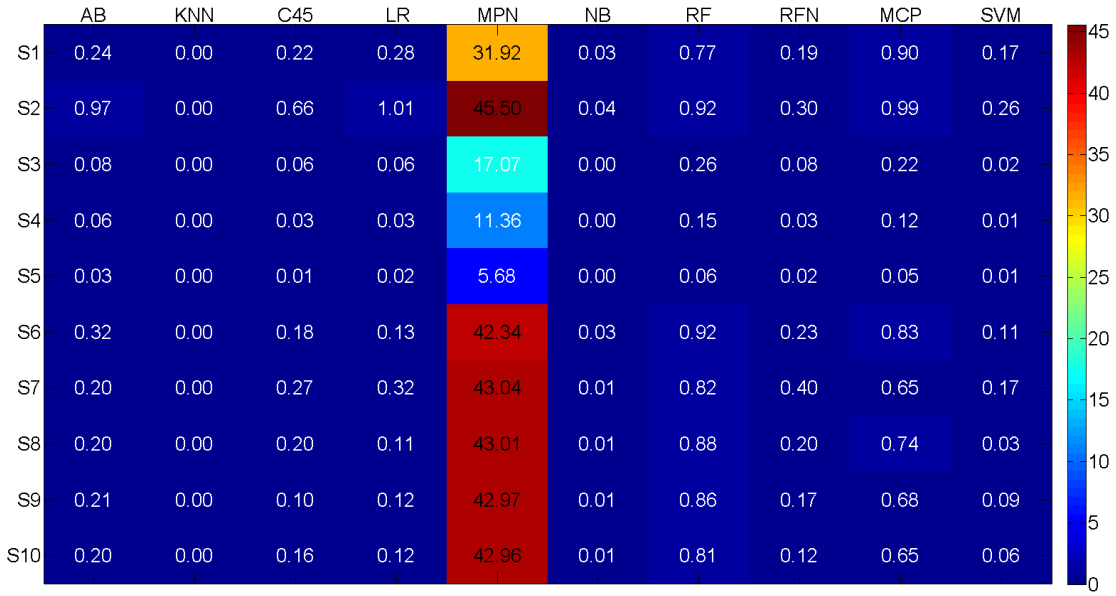

Figure 3.2: Time taken to build model for all subjects.

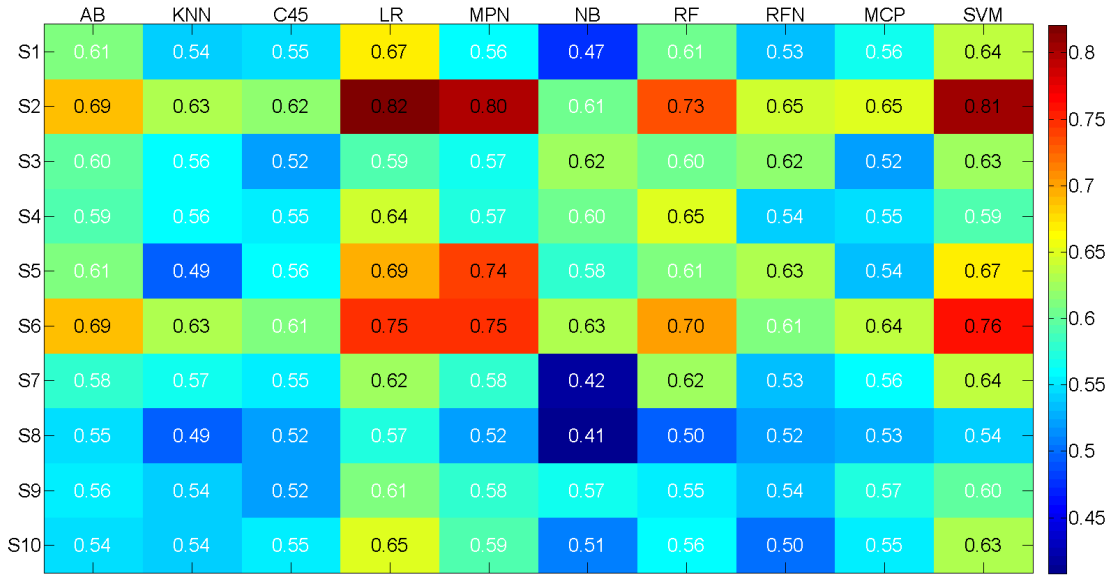

Figure 3.3: F-score for all subjects.

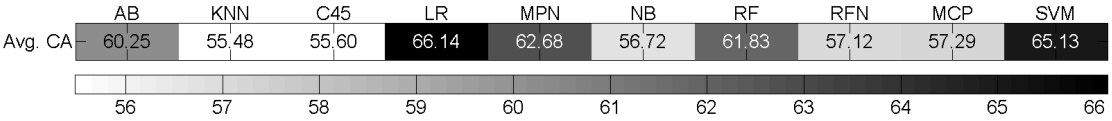

Figure 3.4: Average classification accuracy over all subjects.

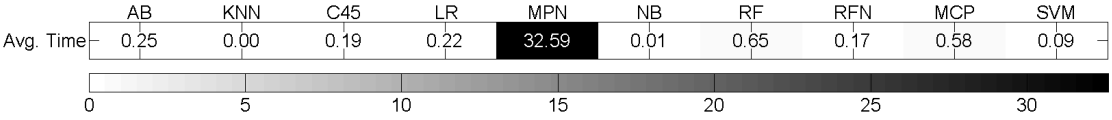

Figure 3.5: Average time taken to build model over all subjects.

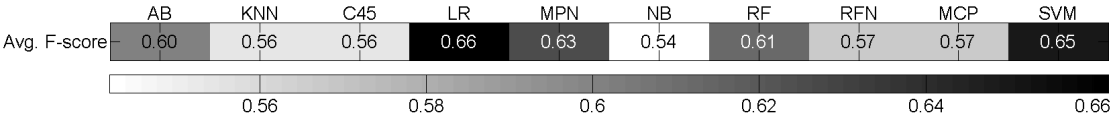

Figure 3.6: Average F-score over all subjects.

According to the datasets chosen for this thesis, the best classifier for classifying EEG data without parameter optimization is the logistic regression model (LR), followed by sequential minimal optimization algorithm for training a support vector (SVM) and multilayer perceptron network (MPN), with average values for CA, Time and F-score being 66.14%, 0.22s, 0.661; 65.13%, 0.09s, 0.650 and 62.68%, 32.59s, 0.625, respectively.

3.2 Visual Comparison by Using Clustergrams

A visual comparison of the classifiers was done by using clustergrams.

Clustergram is an object which contains the analysis data based on given input data (prediction values from each classifier for each subject) and visualizes it by displaying a dendrogram and a heat map.

It uses *hierarchical clustering* with *Euclidean distance metric* and average linkage to generate the hierarchical tree.

It clusters along the rows of the data, which results in clustered columns in the matrix of the input data [74].

This comparison was done for the following reasons:

1. To visualize the classification predictions created by different classifiers.
2. To use the created dendrogram in order to see if the predictions generated by the classifiers are highly correlated or not — if the classifiers are highly correlated then the clusters would be nearer the bottom of the dendrogram and vice versa.

Since the prediction values from each classifier can only have four possible outcomes (TP, TN, FP, FN) (see subsection 2.4.1), each prediction value in the heat map was colored as follows:

- True Positive (TP) & True Negative (TN)
  - Color: Green
  - Heat map numerical value: -1
- False Positive (FP) & False Negative (FN)
  - Color: Red
  - Heat map numerical value: 1

An example clustergram for Subject 2 from Dataset 1: BCI Competition III Dataset IVa is shown in Figure 3.7.

In the provided example, it is possible to see multilayer perceptron network (MPN) and support vector classifier (SVM) being highly correlated due to the fact that the clusters are near the bottom of the dendrogram. This can also be seen upon visual inspection - for instance, near the midpoint of the dendrogram, both of the classifiers have made similar False Positive (FP) and False Negative (FN) predictions.

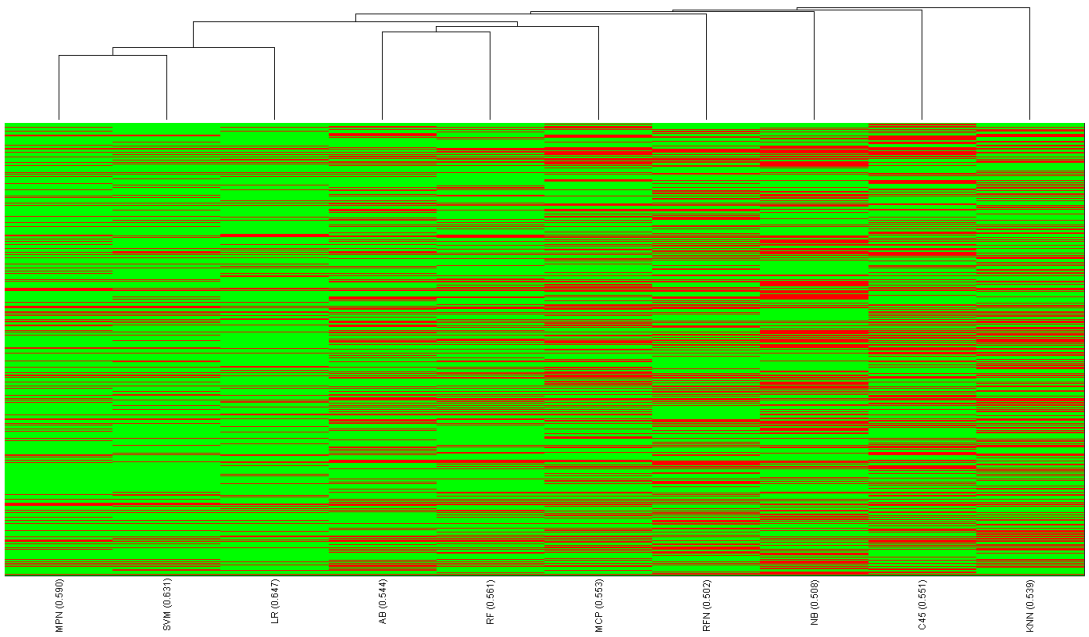

Figure 3.7: Example clustergram for the Subject 2 from the Dataset 1. The number in the brackets is the F-score of the classifier.

The clustergrams for all of the subjects can be found in Appendix A.

3.3 Comparison by Using Correlation Matrices

A comparison of the classifiers was done by using linear or rank correlation. Matrices for each subject from the datasets were created which contained the pairwise linear correlation coefficient between each pair of classifiers. The correlation matrices are symmetric because the correlation between X_i and X_j is the same as the correlation between X_j and X_i (where X_j and X_i indicate performance of algorithms j and i).

There are several correlation coefficients, often denoted ρ or r, measuring the degree of correlation. This thesis used the most common of these, the Pearson

correlation coefficient, which is sensitive only to a linear relationship between two variables. This comparison was done in order to get the numerical correlation coefficient values between each pair of classifiers.

An example correlation matrix containing the average correlation over all subjects is shown in Figure 3.8. In the provided example, it is possible to see logistic regression (LR) and multilayer perceptron network (MPN) both having high linear correlation with support vector classifier (SVM). It is also possible to see that k-Nearest Neighbours (KNN) has the lowest linear correlation with logistic regression (LR).

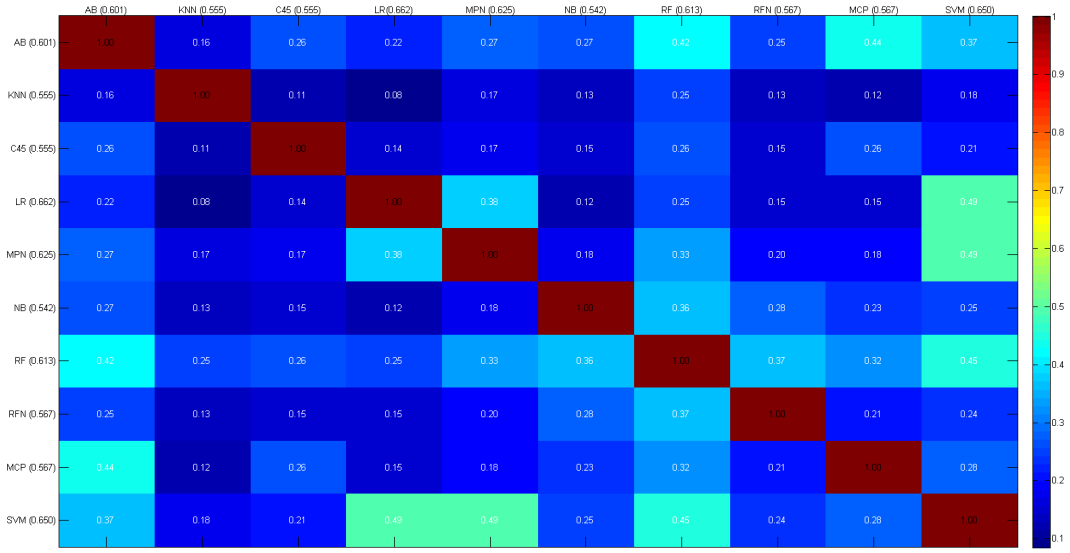

Figure 3.8: Example average correlation matrix over all of the subjects. The number in the brackets is the F-score of the classifier.

The correlation matrices for all of the subjects can be found in Appendix B.

3.4 Comparison after Parameter Optimization

In order to fully actualize the importance of parameter optimization in machine learning algorithms, SMAC and AutoWEKA were used.

All of the datasets and machine learning algorithms were optimized by using 10-fold cross-validation when generating their corresponding instances.

Due to the limitations of AutoWEKA, the only result metric that could be used was Classification Accuracy (CA). There were also issues with some of the ma-

chine learning algorithms, therefore it was not possible to obtain the optimization results for AdaBoost (AB), radial basis function network (RFN) and minimal cost-complexity pruning (MCP).

The following AutoWEKA configuration parameters were used for each dataset and machine learning algorithm:

- Optimization Timeout (hours): 1
- Training Memory Limit (MB): 2048
- Training Run Timeout (minutes): 60
- Use Attribute Selection: Yes
- Attribute Selection Timeout (minutes): 60

As soon as one of the timeouts was reached, AutoWEKA stopped the optimization process and used the parameters that gave the best results at that particular point in time.

The results obtained by AutoWEKA after parameter optimization for Classification Accuracy (CA) can be seen in figure 3.9.

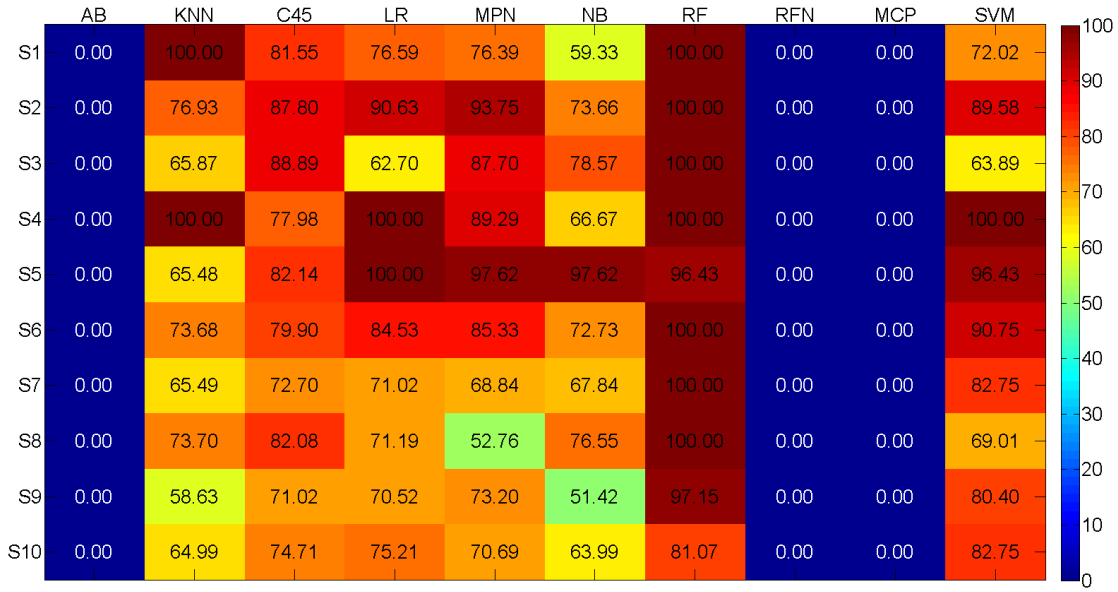

Figure 3.9: Classification accuracy for all subjects after parameter optimization.

The average results for Classification Accuracy (CA) with and without Parameter Optimization (PO) can be seen in figure 3.10.

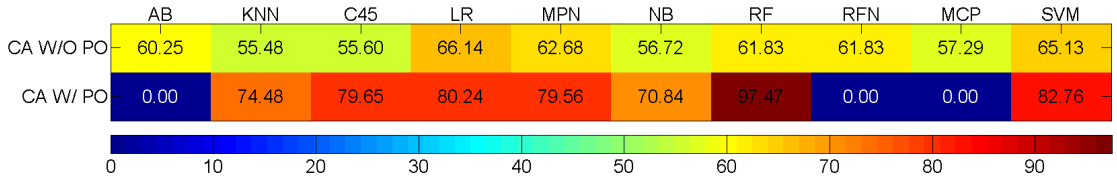

Figure 3.10: Average classification accuracy for all subjects with and without parameter optimization.

The optimized parameter values for all of the subjects can be found in Appendix C. More detailed information about the optimized parameters can be found at:

`https://github.com/madism/ECMLA/tree/master/Parameters`

Chapter 4

Conclusion

This thesis was created in order to find the best machine learning algorithm for classifying EEG data by means of an empirical comparison. In order to achieve this, three datasets, which were created for various brain-computer interface (BCI) competitions, were used. A total of 10 subjects from the datasets were used.

These datasets were preprocessed and then classified by the following machine learning algorithms/methods: AdaBoost, k-Nearest Neighbours, C4.5 decision tree, logistic regression model, multilayer perceptron network, naive Bayes, random forest, radial basis function network, minimal cost-complexity pruning and support vector classifier. The classification results were evaluated using 10-fold cross-validation.

The visual comparison which was done by using clustergrams showed that some of the predictions generated by the classifiers were highly correlated. This was confirmed by the comparison done by using correlation matrices, which showed that there was a moderately strong (value of the correlation being in the range of 0.4 to 0.6) correlation between the following classifiers:

- AdaBoost and random forest;
- AdaBoost and minimal cost-complexity pruning;
- Logistic regression and SVM;
- Multilayer Perceptron Network and SVM;
- Random forest and SVM.

The comparison that was done by using correlation matrices showed that the highest correlation coefficient was in Subject 2 from Dataset 1 between sequential minimal optimization algorithm for training a support vector classifier (SVM) and Multilayer Perceptron Network (MPN), which had a correlation value of 0.77, therefore indicating a strong correlation between the two classifiers.

The comparison that was done by using three performance metrics (classification accuracy, time taken to build a model and F-score) showed that according to our datasets, the best classifier for classifying EEG data without parameter optimization is the logistic regression model (LM), followed by sequential minimal optimization algorithm for training a support vector (SVM) and Multilayer Perceptron Network (MPN).

The comparison that was done by using SMAC and AutoWEKA showed the importance of parameter optimization. All of the machine learning algorithms which were able to be optimized benefited greatly from the optimization. Average Classification Accuracy (CA) was improved by the amount of 14.10% in the case of the logistic regression and up to 24.05% in the case of the C4.5 decision tree model.

The results from the comparison by using optimized parameters showed that according to our datasets, the best classifier for classifying EEG data with parameter optimization is random forest (RF), followed by sequential minimal optimization algorithm for training a support vector (SVM) and logistic regression model (LM).

Bibliography

[1] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Thirteenth International Conference on Machine Learning*, (San Francisco), pp. 148–156, Morgan Kaufmann, 1996.

[2] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[3] W. Iba and P. Langley, "Induction of one-level decision trees," in *Proceedings of the ninth international conference on machine learning*, pp. 233–240, 1992.

[4] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[5] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[6] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[7] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[8] K. Søreide, A. J. Krüger, A. L. Vårdal, C. L. Ellingsen, E. Søreide, and H. M. Lossius, "Epidemiology and contemporary patterns of trauma deaths: changing place, similar pace, older face," *World journal of surgery*, vol. 31, no. 11, pp. 2092–2103, 2007.

[9] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1-2, pp. 167–179, 1967.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., DTIC Document, 1985.

[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[12] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," tech. rep., DTIC Document, 1961.

[13] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Eleventh Conference on Uncertainty in Artificial Intelligence*, (San Mateo), pp. 338–345, Morgan Kaufmann, 1995.

[14] J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, *et al.*, "Tackling the poor assumptions of naive Bayes text classifiers," in *ICML*, vol. 3, pp. 616–623, Washington DC), 2003.

[15] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, IBM New York, 2001.

[16] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 278–282, IEEE, 1995.

[18] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[19] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," tech. rep., DTIC Document, 1988.

[20] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural networks*, vol. 14, no. 4, pp. 439–458, 2001.

[21] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, California: Wadsworth International Group, 1984.

[22] J. Pearl, "Heuristics: intelligent search strategies for computer problem solving," 1984.

[23] J. Platt *et al.*, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods — Support Vector Learning*, vol. 3, 1999.

[24] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[25] T. Hastie, R. Tibshirani, *et al.*, "Classification by pairwise coupling," *The annals of statistics*, vol. 26, no. 2, pp. 451–471, 1998.

[26] J. C. Platt *et al.*, "Using analytic QP and sparseness to speed training of support vector machines," *Advances in neural information processing systems*, pp. 557–563, 1999.

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, pp. 161–168, ACM, 2006.

[29] B. Karlık and Ş. B. Hayta, "Comparison Machine Learning Algorithms for Recognition of Epileptic Seizures in EEG,"

[30] P. Sajda, A. Gerson, K. Muller, B. Blankertz, and L. Parra, "A data analysis competition to evaluate machine learning algorithms for use in brain-computer interfaces," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 11, no. 2, pp. 184–185, 2003.

[31] C. Hope, "A cap holds electrodes in place while recording an EEG," 2012.

[32] M. Tudor, L. Tudor, and K. I. Tudor, "Hans Berger (1873-1941)–the history of electroencephalography," *Acta medica Croatica: casopis Hravatske akademije medicinskih znanosti*, vol. 59, no. 4, pp. 307–313, 2004.

[33] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila, and O. V. Lounasmaa, "Magnetoencephalography—theory, instrumentation, and applications

to noninvasive studies of the working human brain," *Reviews of modern Physics*, vol. 65, no. 2, p. 413, 1993.

[34] J. Claassen, S. Mayer, R. Kowalski, R. Emerson, and L. Hirsch, "Detection of electrographic seizures with continuous EEG monitoring in critically ill patients," *Neurology*, vol. 62, no. 10, pp. 1743–1748, 2004.

[35] D. J. Kupfer and M. E. Thase, "The use of the sleep laboratory in the diagnosis of affective disorders," *Psychiatric Clinics of North America*, 1983.

[36] F. Plum and J. B. Posner, *The diagnosis of stupor and coma*, vol. 19. Oxford University Press, 1982.

[37] P. Ferenci, A. Lockwood, K. Mullen, R. Tarter, K. Weissenborn, and A. T. Blei, "Hepatic encephalopathy—definition, nomenclature, diagnosis, and quantification: final report of the working party at the 11th World Congresses of Gastroenterology, Vienna, 1998," *Hepatology*, vol. 35, no. 3, pp. 716–721, 2002.

[38] E. F. Wijdicks, "Determining brain death in adults," *Neurology*, vol. 45, no. 5, pp. 1003–1011, 1995.

[39] C. Miniussi and G. Thut, "Combining TMS and EEG offers new prospects in cognitive neuroscience," *Brain topography*, vol. 22, no. 4, pp. 249–256, 2010.

[40] W. J. Ray and H. W. Cole, "EEG alpha activity reflects attentional demands, and beta activity reflects emotional and cognitive processes," *Science*, vol. 228, no. 4700, pp. 750–752, 1985.

[41] N. Braisby and A. Gellatly, *Cognitive psychology*. Oxford University Press, 2012.

[42] H. M. Müller, "Neurolinguistic findings on the language lexicon: The special role of proper names," *Chinese Journal of Physiology*, vol. 53, no. 6, pp. 351–358, 2010.

[43] A. Kales, A. Jacobson, M. J. Paulson, J. D. Kales, and R. D. Walter, "Somnambulism: Psychophysiological correlates: I. All-night EEG studies," *Archives of General Psychiatry*, vol. 14, no. 6, pp. 586–594, 1966.

[44] D. Millet, "The Origins of EEG," in *7th Annual Meeting of the International Society for the History of the Neurosciences (ISHN)*, 2002.

[45] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[46] R. Kohavi and F. Provost, "Glossary of terms," *Machine Learning*, vol. 30, no. 2-3, pp. 271–274, 1998.

[47] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[48] Y. Koren, "The Bellkor solution to the Netflix grand prize," *Netflix prize documentation*, vol. 81, 2009.

[49] S. Patterson, "Letting the machines decide," *Wall Street Journal (Online)*, 2010.

[50] B. Saleh, K. Abe, R. S. Arora, and A. Elgammal, "Toward automated discovery of artistic influence," *Multimedia Tools and Applications*, pp. 1–27, 2014.

[51] P. R. Norvig and S. A. Intelligence, "A modern approach," 2002.

[52] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[53] A. B. Tucker, *Computer science handbook*. CRC press, 2004.

[54] G. Dantzig, "The nature of mathematical programming," *Mathematical Programming Glossary*, 2010.

[55] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, pp. 507–523, Springer, 2011.

[56] C. Thornton and F. Hutter and H. H. Hoos and K. Leyton-Brown, "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms," in *Proc. of KDD-2013*, pp. 847–855, 2013.

[57] I. D. A. G. Fraunhofer FIRST, "Data set IVa <motor imagery, small training sets>," 2004.

[58] G. Dornhege, B. Blankertz, G. Curio, and K. Muller, "Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 993–1002, 2004.

[59] I. D. A. G. Fraunhofer FIRST, "Data set IVb <motor imagery, uncued classifier application>," 2004.

[60] B. B. group and F. FIRST, "Data sets 1 <motor imagery, uncued classifier application>," 2007.

[61] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, and G. Curio, "The non-invasive Berlin brain–computer interface: fast acquisition of effective performance in untrained subjects," *NeuroImage*, vol. 37, no. 2, pp. 539–550, 2007.

[62] N. Z. University of Waikato, "Weka 3: Data Mining Software in Java," 2015.

[63] J. Fourier, *Theorie analytique de la chaleur, par M. Fourier.* Chez Firmin Didot, père et fils, 1822.

[64] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[65] A. Ajanki, "Example of k-nearest neighbour classification — Wikipedia, The Free Encyclopedia," 2007.

[66] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[67] M. Strano and B. M. Colosimo, "Logistic regression analysis for experimental determination of forming limit diagrams," *International Journal of Machine Tools and Manufacture*, vol. 46, no. 6, pp. 673–682, 2006.

[68] S. K. Palei and S. K. Das, "Logistic regression model for prediction of roof fall risks in bord and pillar workings in coal mines: an approach," *Safety science*, vol. 47, no. 1, pp. 88–96, 2009.

[69] PmWiki, "CIS 520: Machine Learning: Fall 09 - 2D binary classification with Naive Bayes," 2009.

[70] Cyc, "Graphic showing the maximum separating hyperplane and the margin — Wikipedia, The Free Encyclopedia," 2008.

[71] C. E. Metz, "Basic principles of ROC analysis," in *Seminars in nuclear medicine*, vol. 8, pp. 283–298, Elsevier, 1978.

[72] C. J. van Rijsbergen, "A theoretical basis for the use of co-occurrence data in information retrieval," *Journal of documentation*, vol. 33, no. 2, pp. 106–119, 1977.

[73] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," 2011.

[74] MathWorks Documentation, "clustergram - Compute hierarchical clustering, display dendrogram and heat map, and create clustergram object," 2015.

Internet URLs were valid on January 7, 2016.

Appendices

Appendix A: Clustergrams

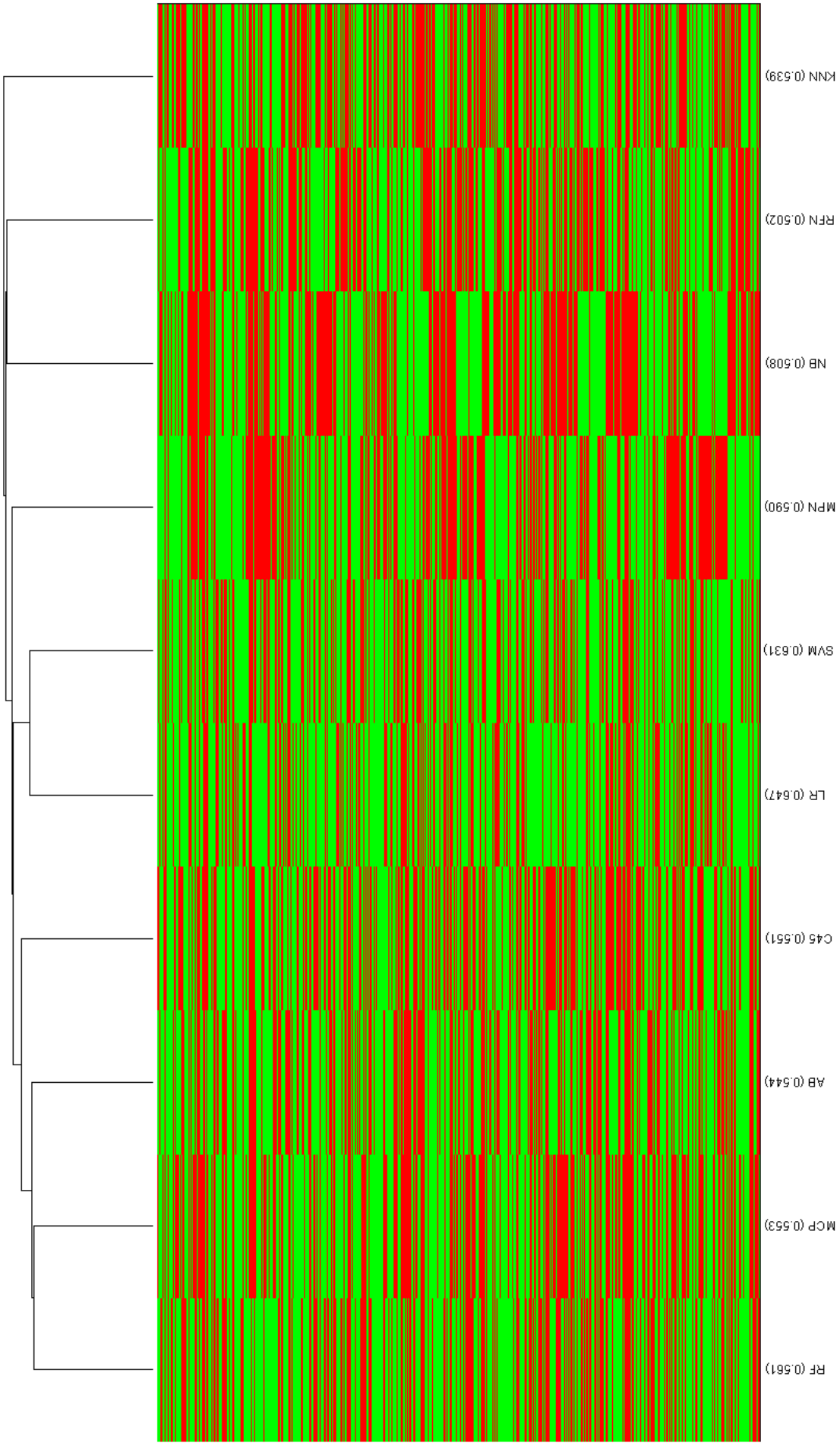

Figure 4.1: Clustergram for Subject 1 from Dataset 1.

Figure 4.2: Clustergram for Subject 2 from Dataset 1.

Figure 4.3: Clustergram for Subject 3 from Dataset 1.

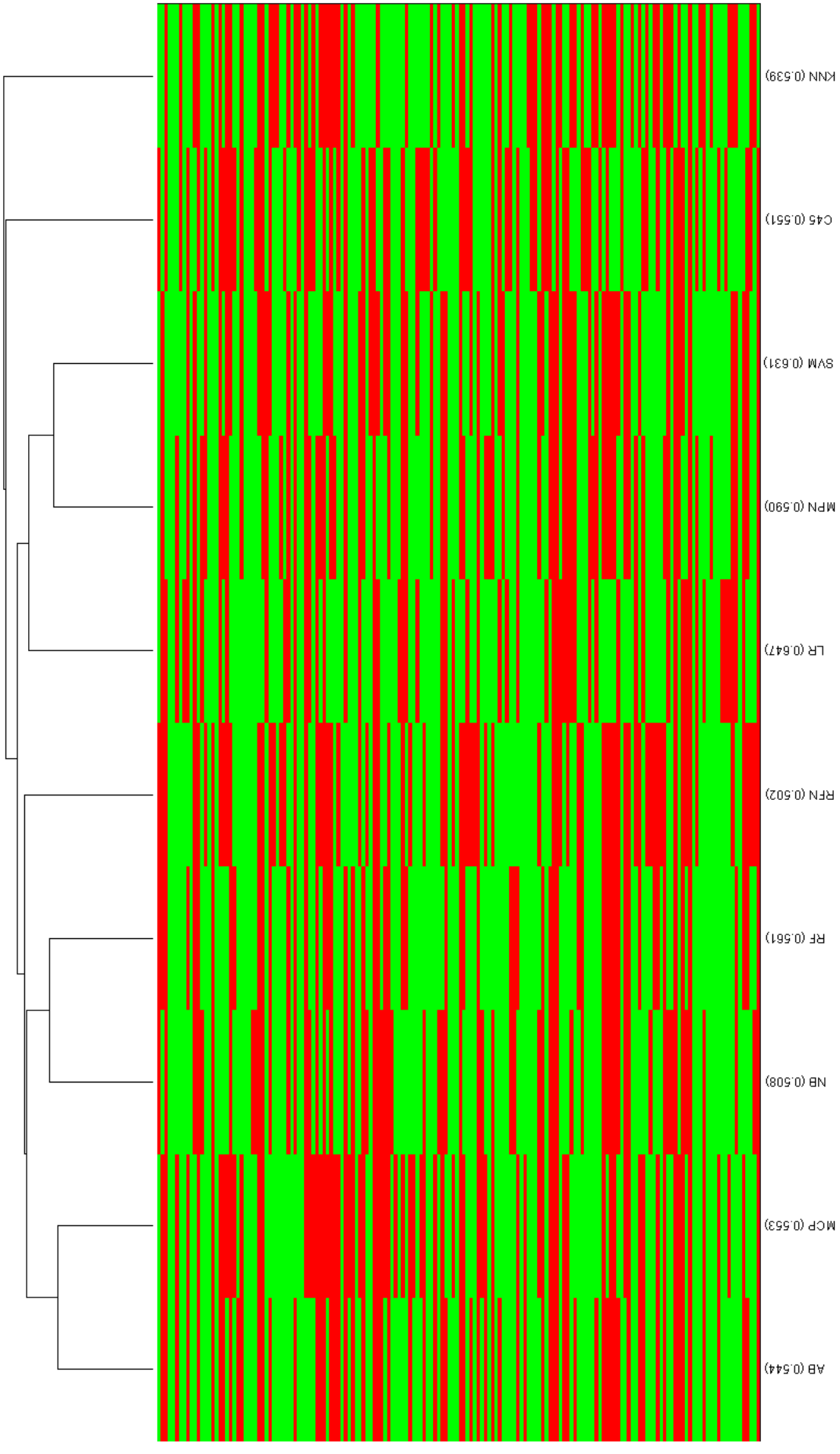

Figure 4.4: Clustergram for Subject 4 from Dataset 1.

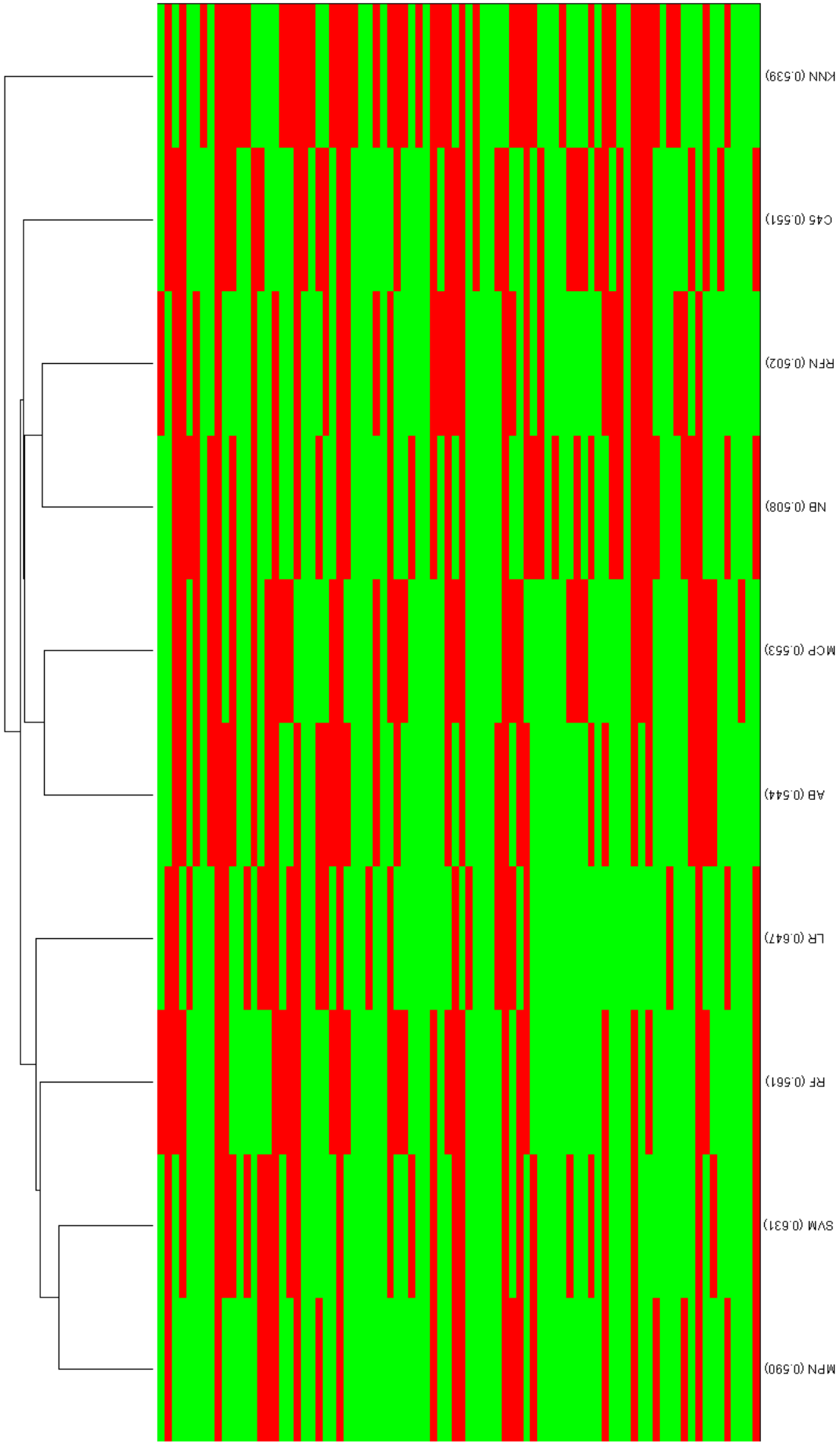

Figure 4.5: Clustergram for Subject 5 from Dataset 1.

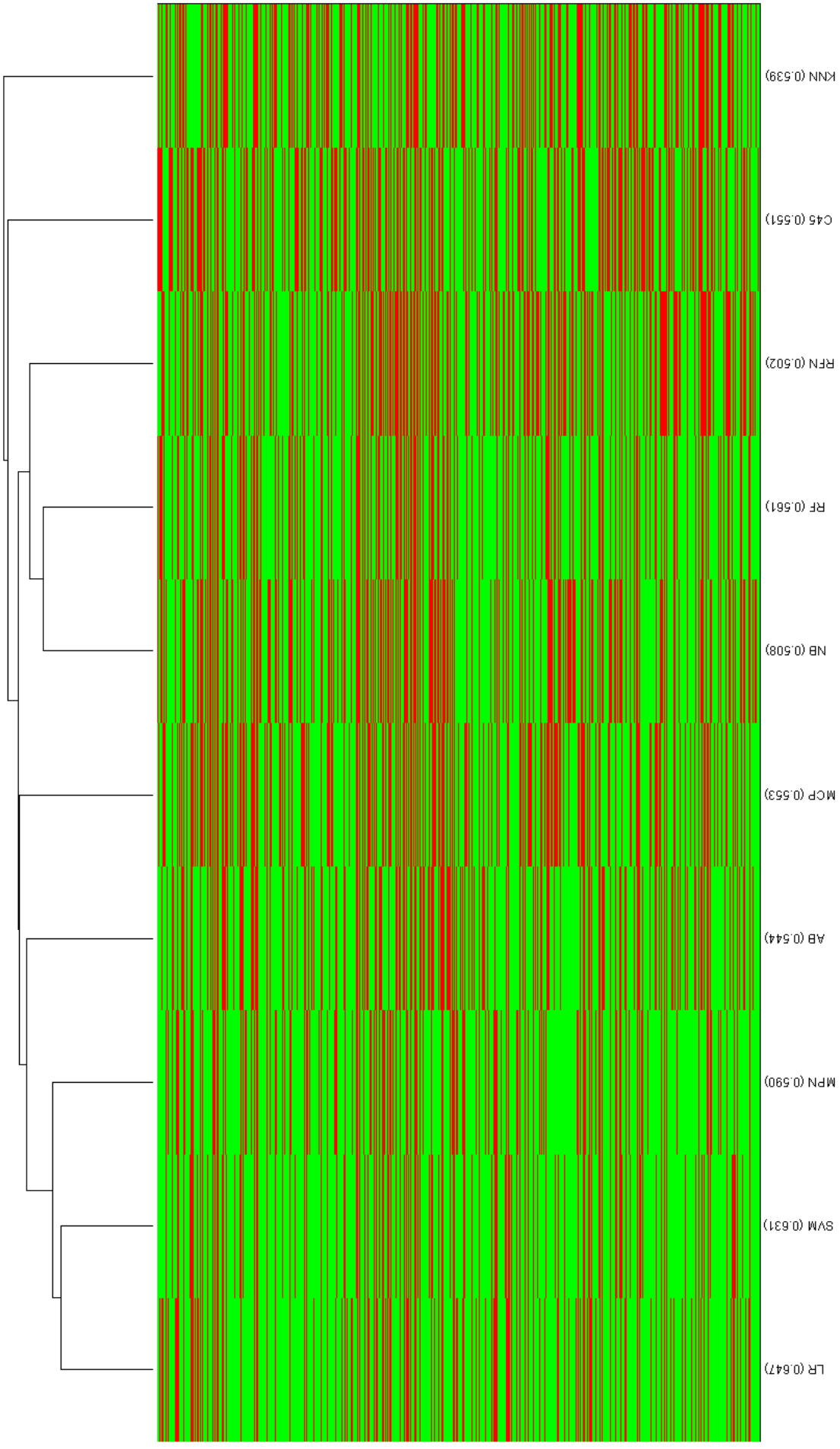

Figure 4.6: Clustergram for Subject 1 from Dataset 2.

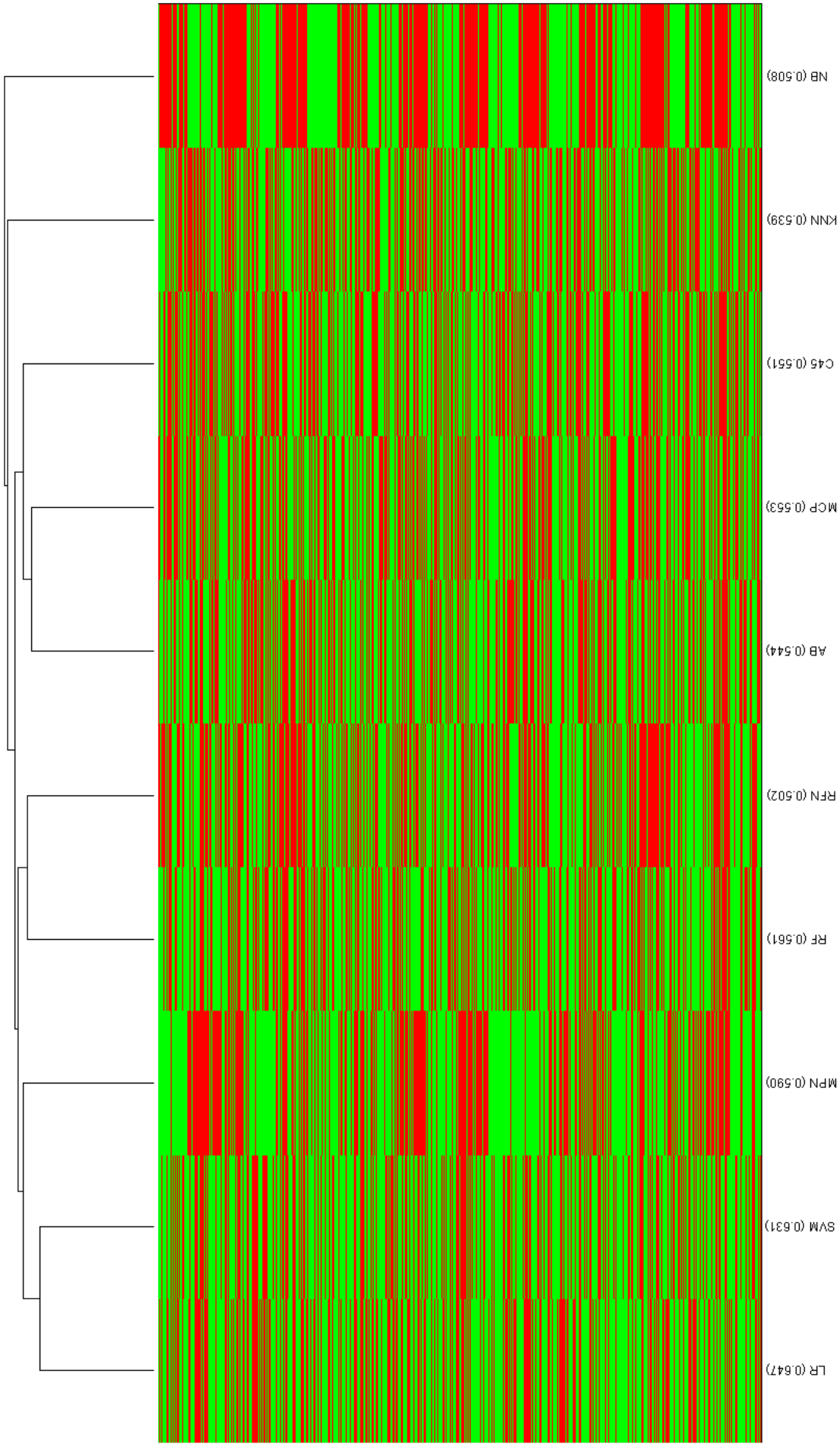

Figure 4.7: Clustergram for Subject 1/1a from Dataset 3.

Figure 4.8: Clustergram for Subject 2/1b from Dataset 3.

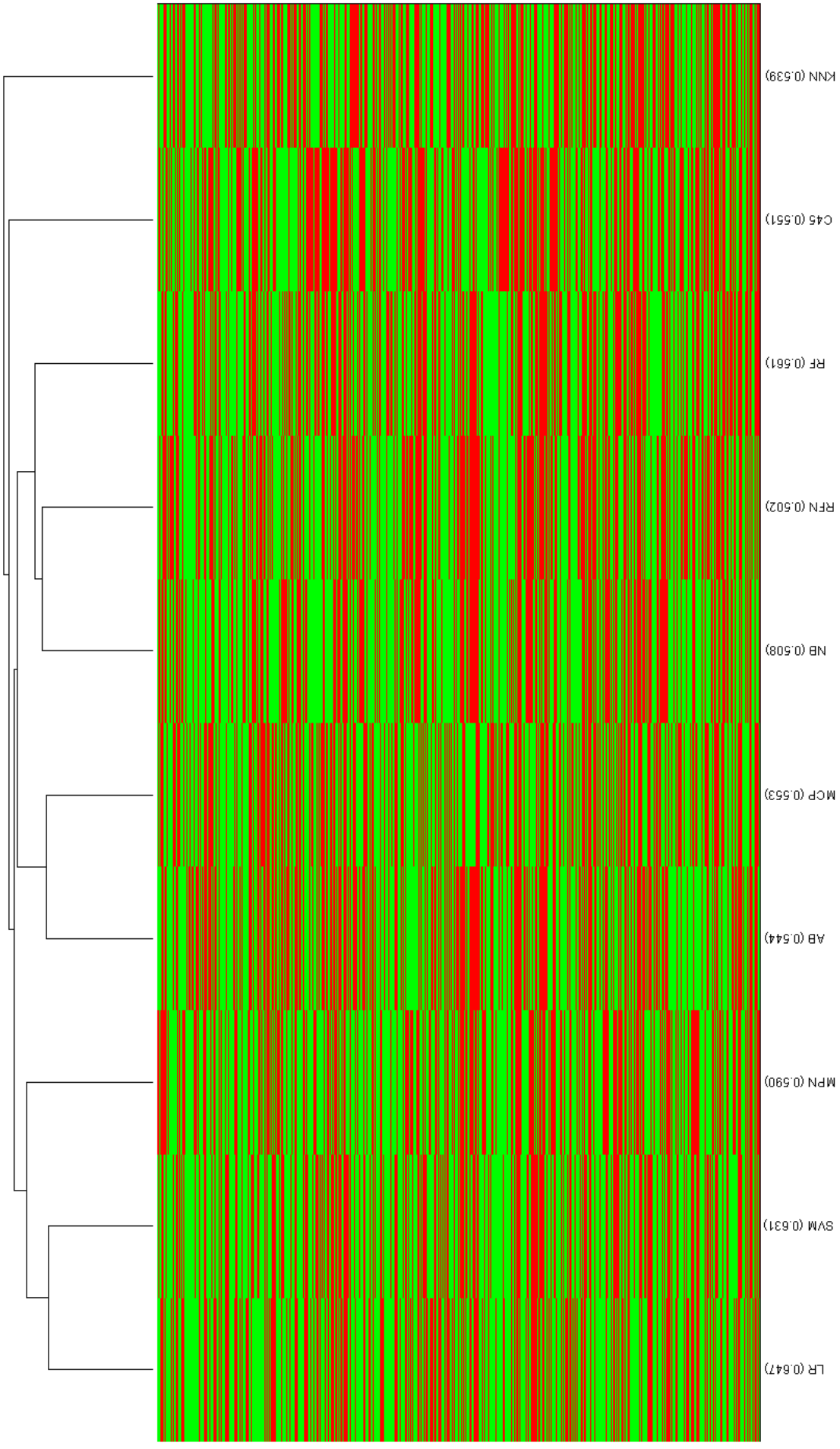

Figure 4.9: Clustergram for Subject 3/1f from Dataset 3.

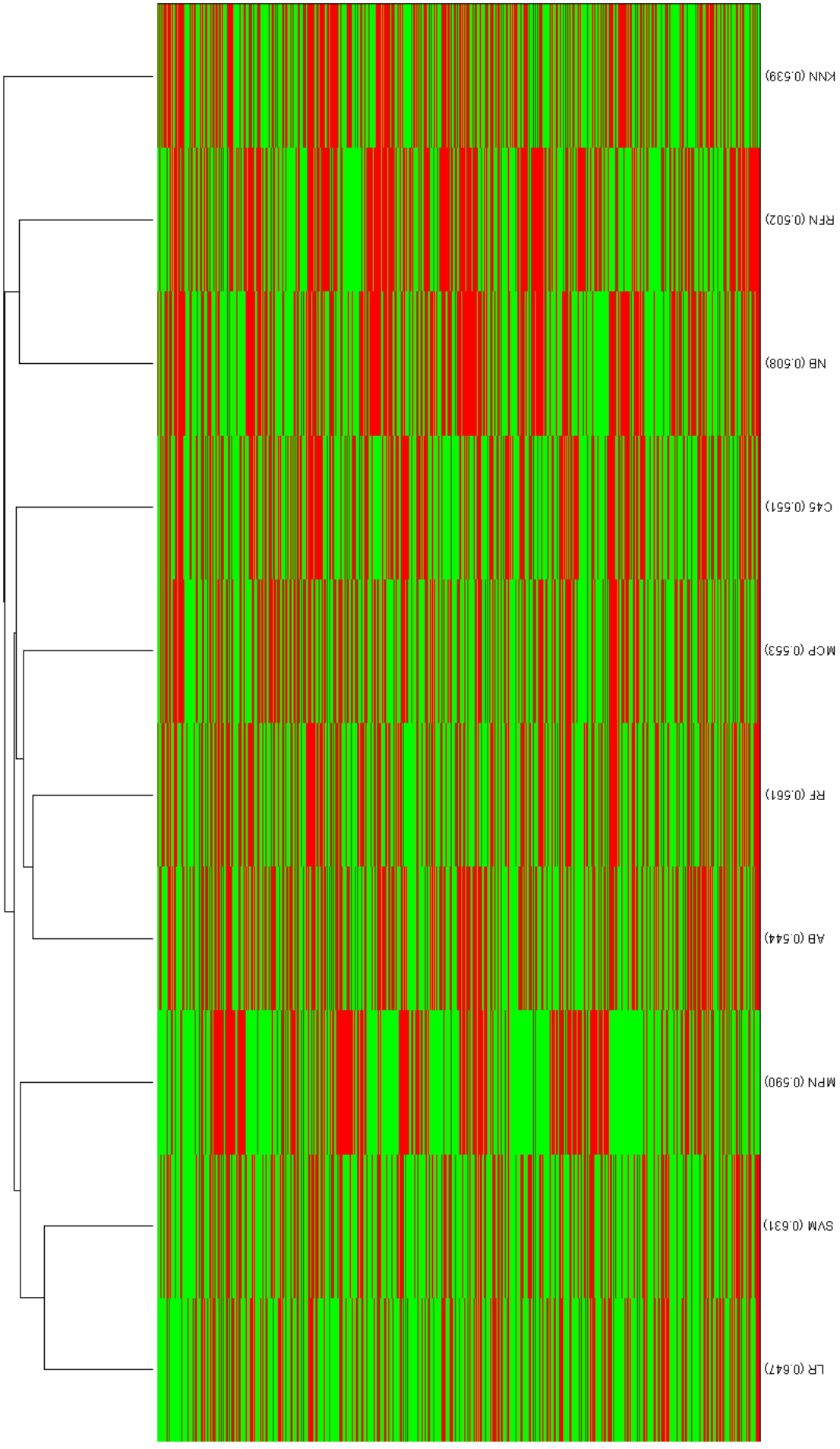

Figure 4.10: Clustergram for Subject 4/1g from Dataset 3.

Appendix B: Correlation Matrices

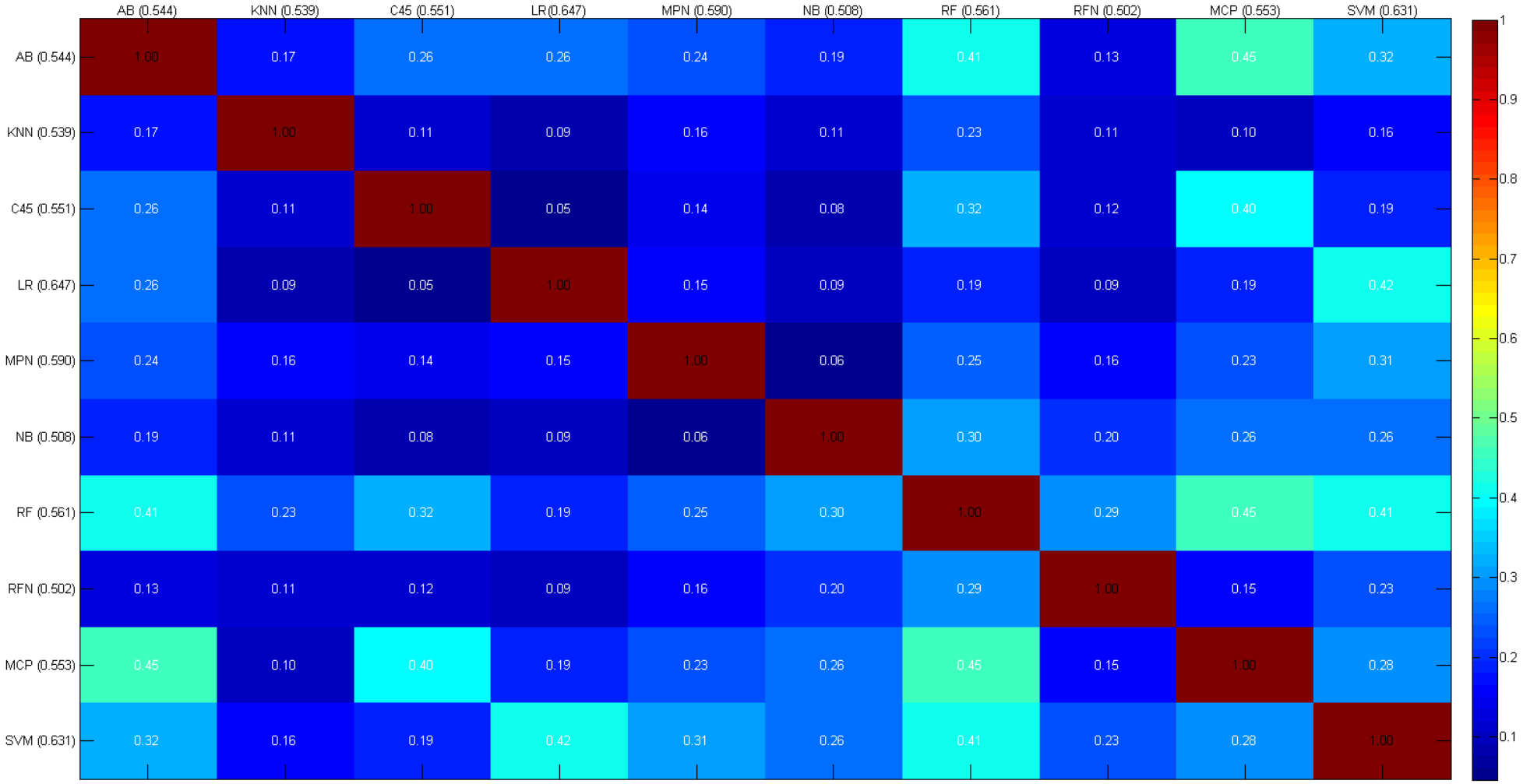

Figure 4.11: Pearson correlation matrix for Subject 1 from Dataset 1.

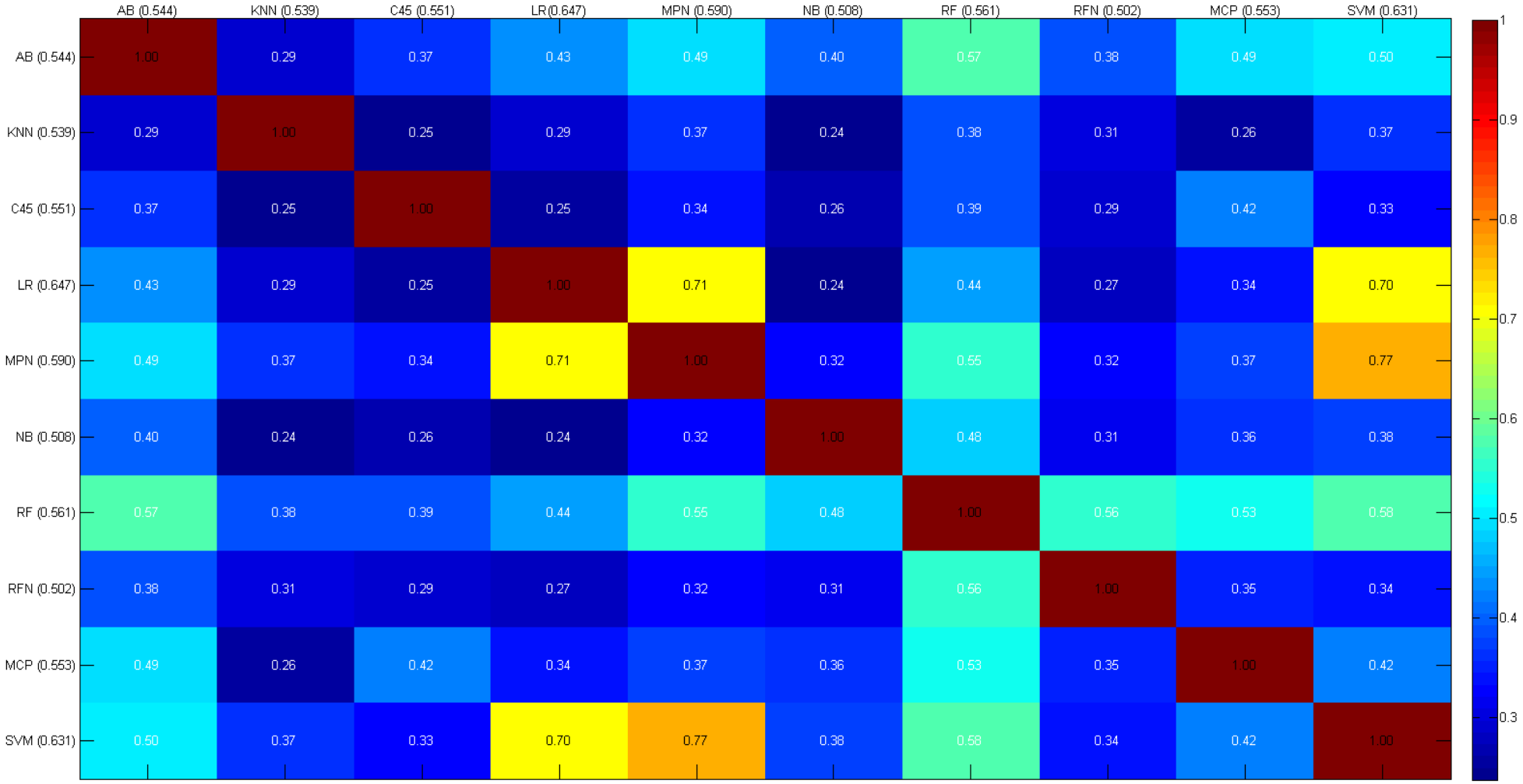

Figure 4.12: Pearson correlation matrix for Subject 2 from Dataset 1.

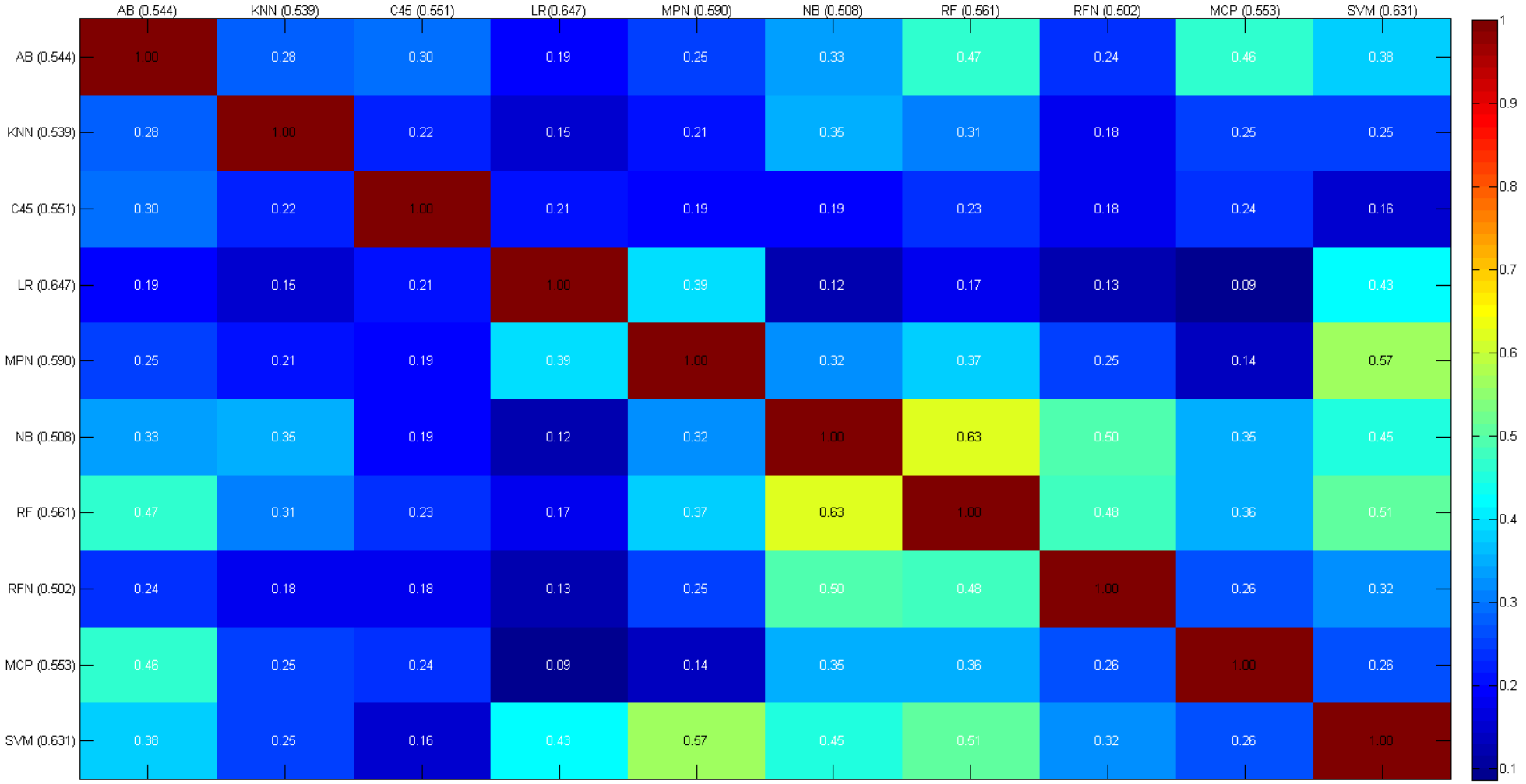

Figure 4.13: Pearson correlation matrix for Subject 3 from Dataset 1.

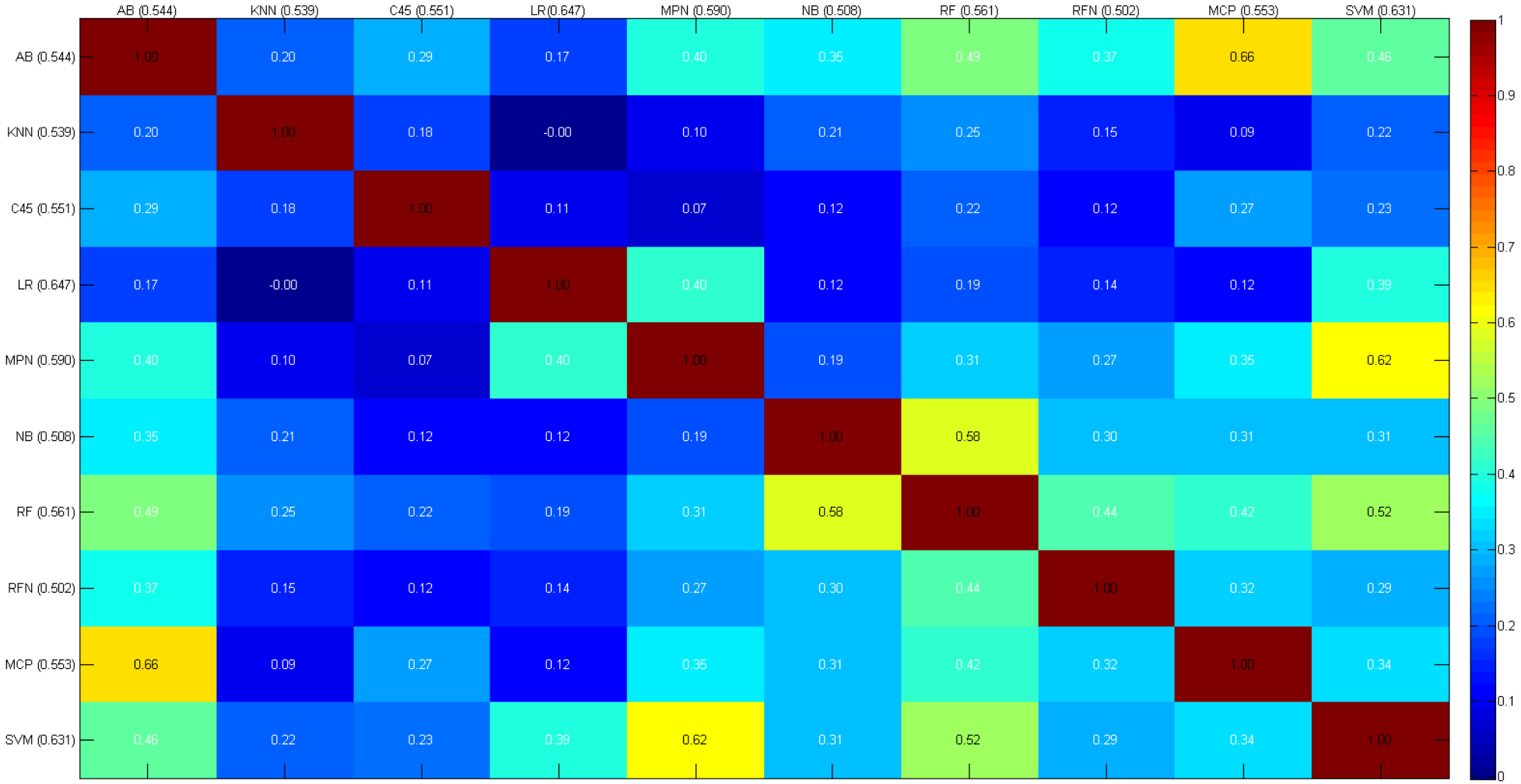

Figure 4.14: Pearson correlation matrix for Subject 4 from Dataset 1.

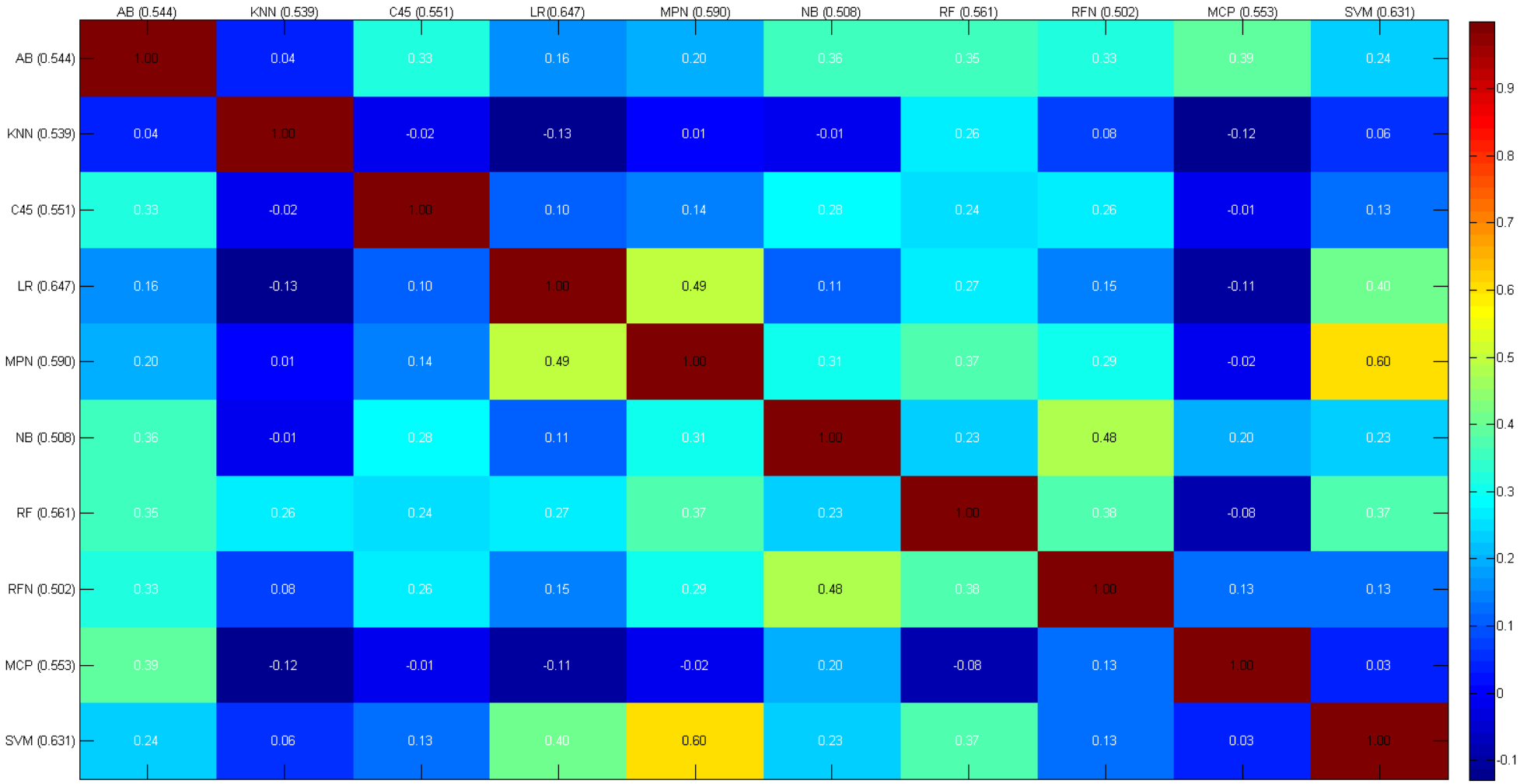

Figure 4.15: Pearson correlation matrix for Subject 5 from Dataset 1.

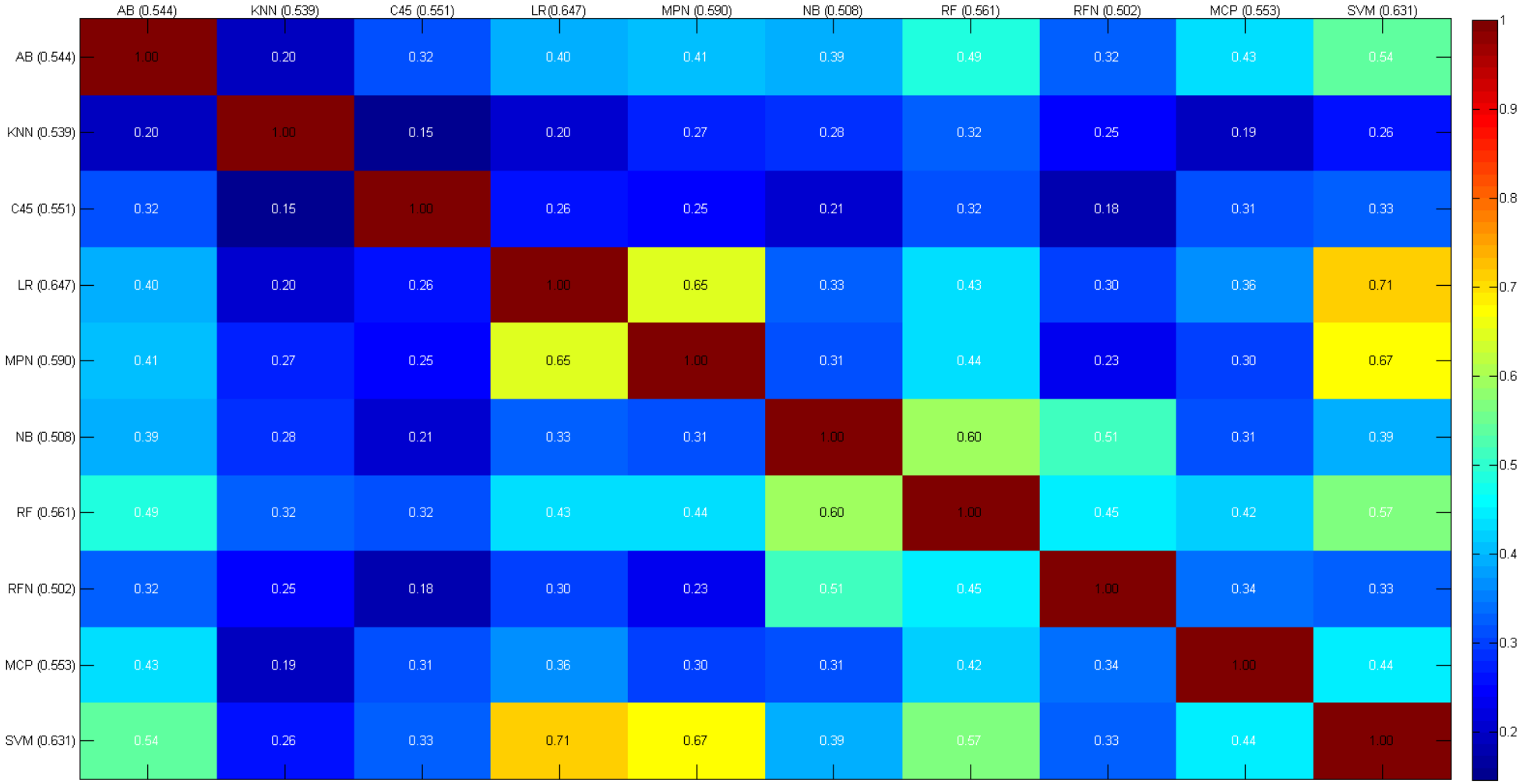

Figure 4.16: Pearson correlation matrix for Subject 1 from Dataset 2.

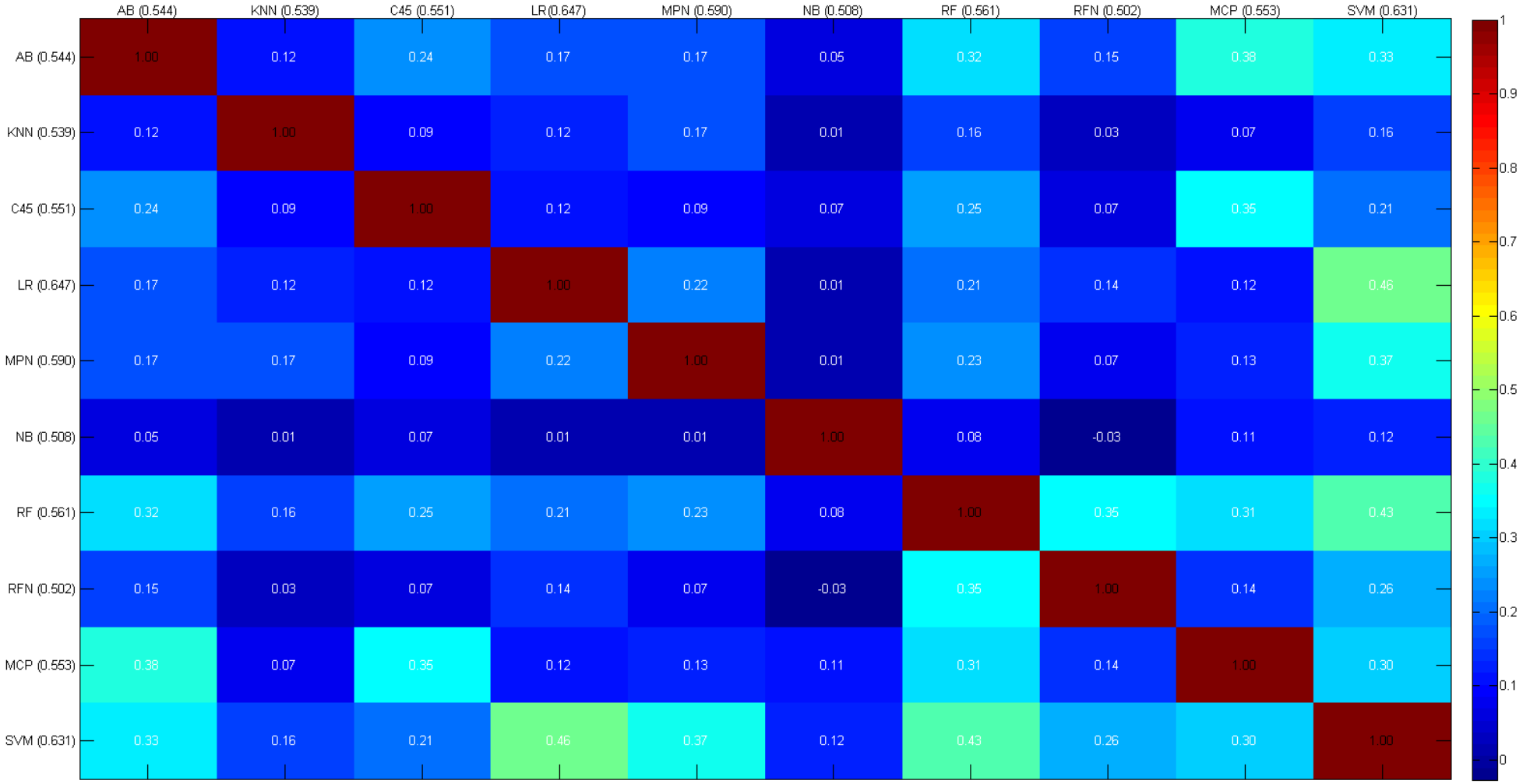

Figure 4.17: Pearson correlation matrix for Subject 1 from Dataset 3.

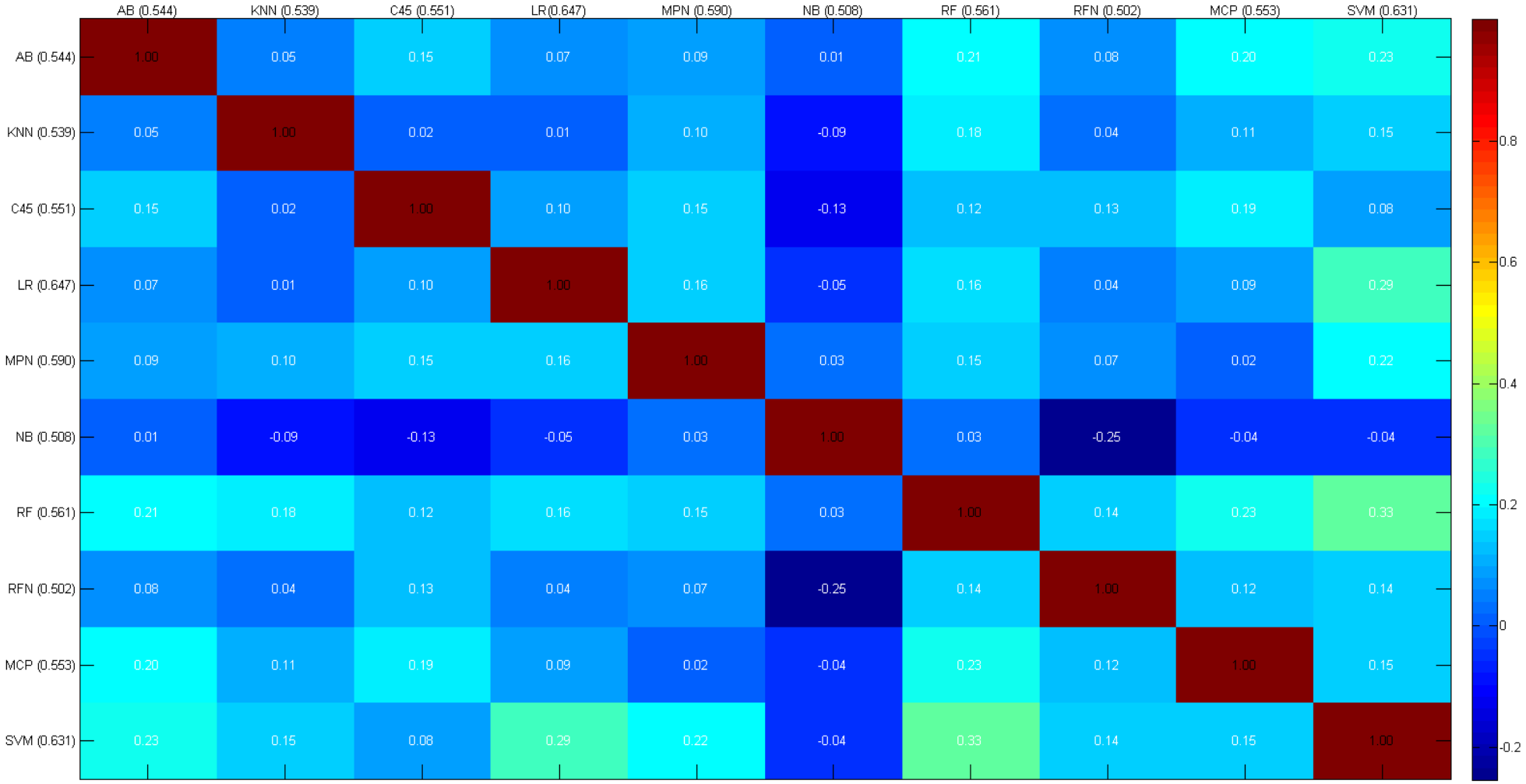

Figure 4.18: Pearson correlation matrix for Subject 2 from Dataset 3.

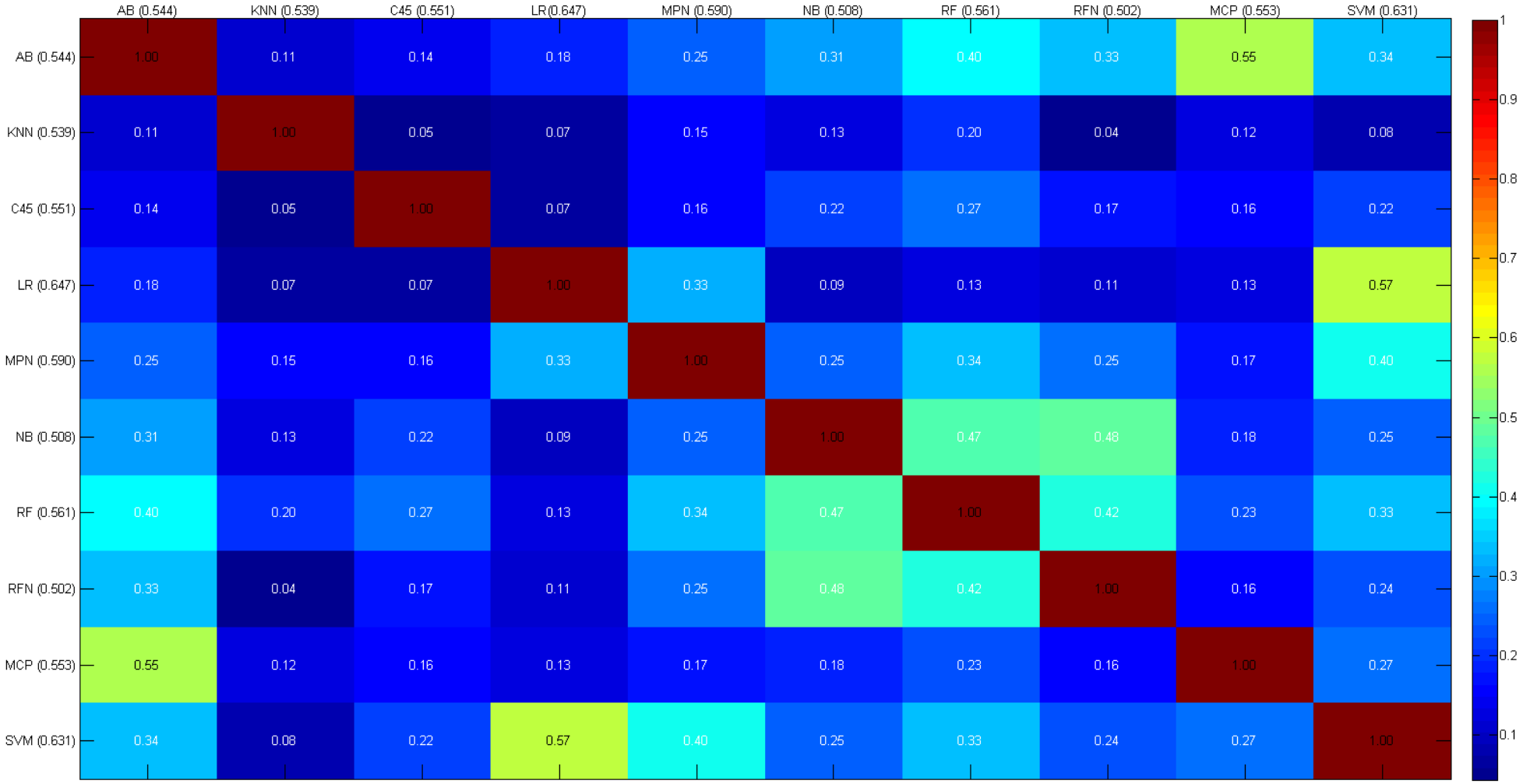

Figure 4.19: Pearson correlation matrix for Subject 3 from Dataset 3.

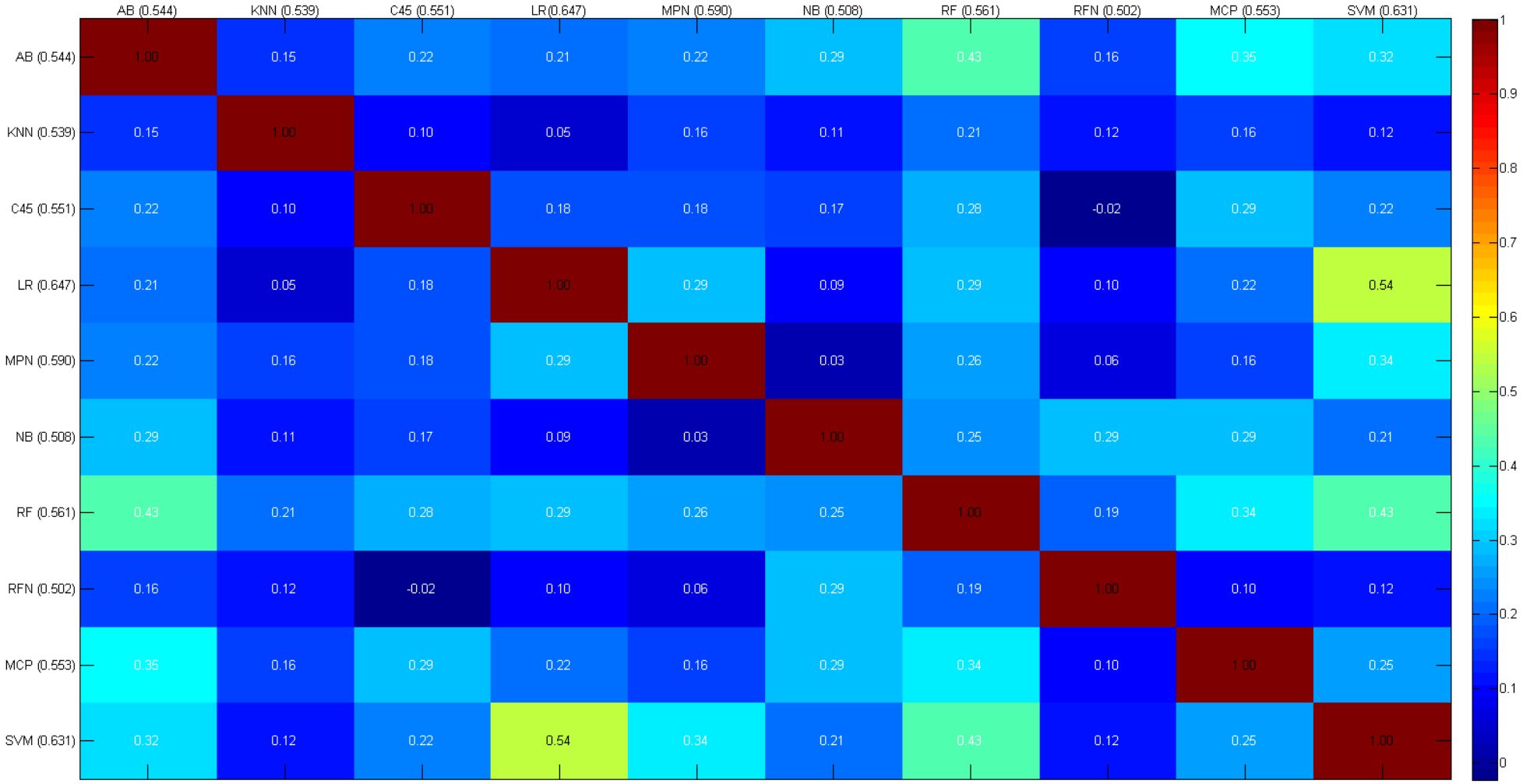

Figure 4.20: Pearson correlation matrix for Subject 4 from Dataset 3.

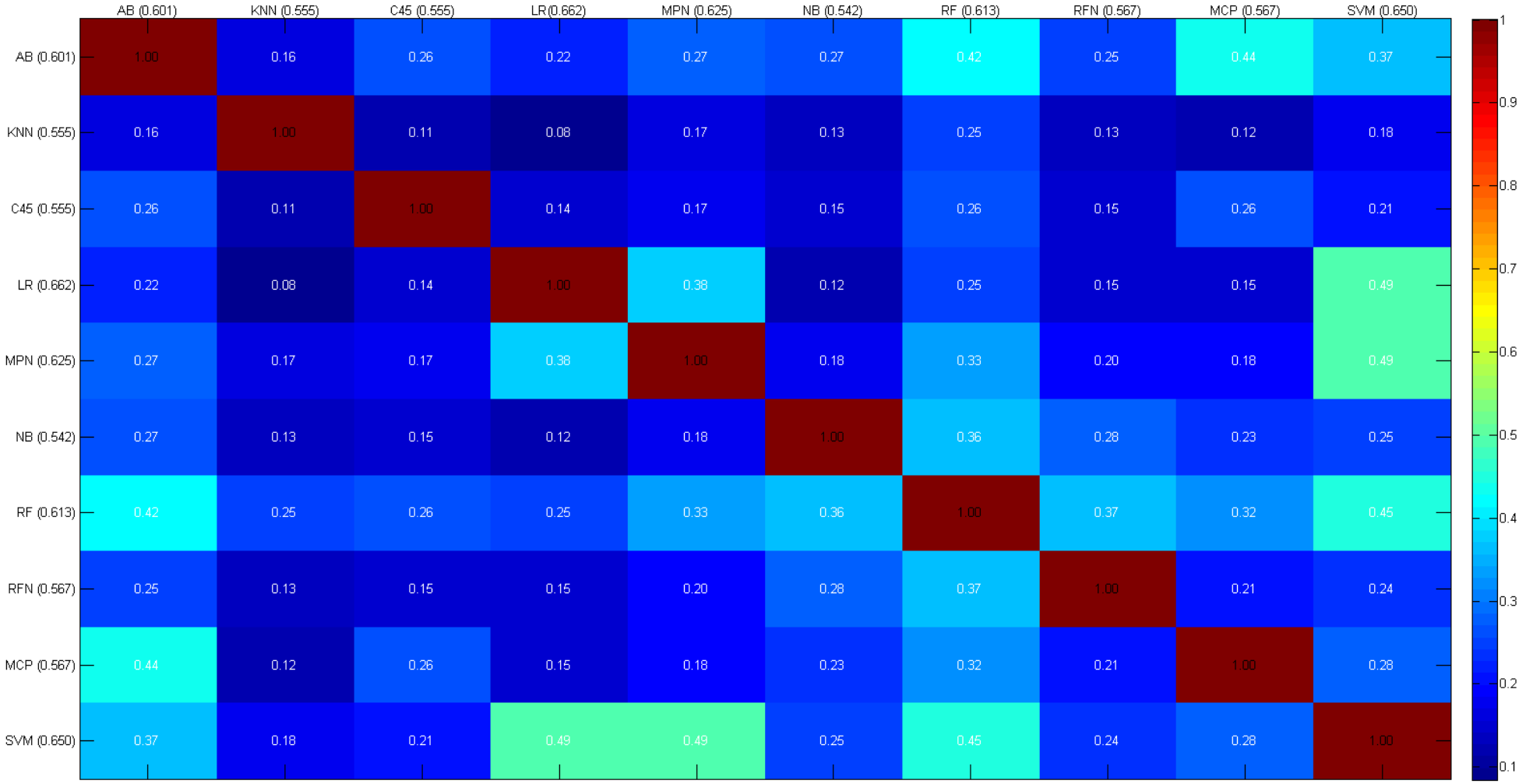

Figure 4.21: Average Pearson correlation matrix over all subjects.

Appendix C: Optimized Parameter Values

k-Nearest Neighbours Subject 1: KNN = 24; lookupCacheSize = 2; searchTermination = 7 attribute_eval = SymmetricalUncertAttributeEval; attribute_search = BestFirst;

k-Nearest Neighbours Subject 2: KNN = 21; searchTermination = 15 attribute_eval = GainRatioAttributeEval; attribute_search = GreedyStepwise;

k-Nearest Neighbours Subject 3: KNN = 51; lookupCacheSize = 0; searchTermination = 6 attribute_eval = ReliefFAttributeEval; numNeighbours = 3; weightByDistance = False; sigma = 1; attribute_search = BestFirst;

k-Nearest Neighbours Subject 4: KNN = 1; searchTermination = 124 attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

k-Nearest Neighbours Subject 5: KNN = 20; searchTermination = 15; attribute_eval = SymmetricalUncertAttributeEval; attribute_search = GreedyStepwise;

k-Nearest Neighbours Subject 6: KNN = 25; searchTermination = 203; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

k-Nearest Neighbours Subject 7: KNN = 13; lookupCacheSize = 0; searchTermination = 5; attribute_eval = ReliefFAttributeEval; numNeighbours = 10; weightByDistance = False; sigma = 7; attribute_search = BestFirst;

k-Nearest Neighbours Subject 8: KNN = 16; attribute_search = NONE;

k-Nearest Neighbours Subject 9: KNN = 52; lookupCacheSize = 0; searchTermination = 4; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

k-Nearest Neighbours Subject 10: KNN = 16; lookupCacheSize = 0; searchTermination = 6; attribute_eval = CorrelationAttributeEval; attribute_search = BestFirst;

C4.5 Decision Tree Subject 1: minNumObj = 13; confidenceFactor = 0.0159; lookupCacheSize = 0; searchTermination = 5; attribute_eval = SymmetricalUncertAttributeEval; attribute_search = BestFirst;

C4.5 Decision Tree Subject 2: minNumObj = 6; confidenceFactor = 0.1929;

lookupCacheSize = 1; searchTermination = 7; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

C4.5 Decision Tree Subject 3: minNumObj = 8; confidenceFactor = 0.2566; threshold = 14.6656; attribute_eval = GainRatioAttributeEval; attribute_search = GreedyStepwise;

C4.5 Decision Tree Subject 4: minNumObj = 10; confidenceFactor = 0.7056; searchTermination = 21; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

C4.5 Decision Tree Subject 5: minNumObj = 11; confidenceFactor = 0.3397; centerData = True; maximumAttributeNames = 1; varianceCovered = 0.9280; lookupCacheSize = 0; searchTermination = 10; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

C4.5 Decision Tree Subject 6: minNumObj = 4; confidenceFactor = 0.2527; lookupCacheSize = 1; searchTermination = 8; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

C4.5 Decision Tree Subject 7: minNumObj = 43; confidenceFactor = 0.3747; folds = 2; minimumBucketSize = 1; threshold = 1.6254; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

C4.5 Decision Tree Subject 8: minNumObj = 14; confidenceFactor = 0.3500; folds = 14; minimumBucketSize = 2; threshold = 9.2158; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

C4.5 Decision Tree Subject 9: minNumObj = 30; confidenceFactor = 0.0974; centerData = True; maximumAttributeNames = -; varianceCovered = 0.9858; lookupCacheSize = 2; searchTermination = 5; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

C4.5 Decision Tree Subject 10: minNumObj = 21; confidenceFactor = 0.0363; folds = 10; minimumBucketSize = 5; threshold = 5.5871; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Logistic Regression Model Subject 1: ridge = 5.2227; attribute_search = NONE;

Logistic Regression Model Subject 2: ridge = 0.3669; numToSelect = 38; attribute_eval = CorrelationAttributeEval; attribute_search = GreedyStepwise;

Logistic Regression Model Subject 3: ridge = 2.8541; lookupCacheSize = 2; searchTermination = 10; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

Logistic Regression Model Subject 4: ridge = 8.3614E-7; numNeighbours = 44; weightByDistance = False; sigma = 1; threshold = 3.0904; attribute_eval = ReliefFAttributeEval; attribute_search = GreedyStepwise;

Logistic Regression Model Subject 5: ridge = 0.0246; folds = 8; minimumBucketSize = 2; threshold = 2.8797; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Logistic Regression Model Subject 6: ridge = 2.0988; threshold = 17.7142; attribute_eval = GainRatioAttributeEval; attribute_search = GreedyStepwise;

Logistic Regression Model Subject 7: ridge = 9.9170; attribute_search = NONE;

Logistic Regression Model Subject 8: ridge = 0.7175; numNeighbours = 15; weightByDistance = False; sigma = 4; lookupCacheSize = 2; searchTermination = 10; attribute_eval = ReliefFAttributeEval; attribute_search = BestFirst;

Logistic Regression Model Subject 9: ridge = 3.6800; folds = 11; minimumBucketSize = 8; threshold = 0.2781; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Logistic Regression Model Subject 10: ridge = 0.7175; numNeighbours = 15; weightByDistance = False; sigma = 4; lookupCacheSize = 2; searchTermination = 10; attribute_eval = ReliefFAttributeEval; attribute_search = BestFirst;

Multilayer Perceptron Network Subject 1: learningRate = 0.2969; momentum = 0.8217; hiddenLayers = o; folds = 13; minimumBucketSize = 51; lookupCacheSize = 2; searchTermination = 4; attribute_eval = OneRAttributeEval; attribute_search = BestFirst;

Multilayer Perceptron Network Subject 2: learningRate = 0.6786; momentum = 0.1688; hiddenLayers = t; attribute_search = NONE;

Multilayer Perceptron Network Subject 3: learningRate = 0.3319; momentum = 0.7439; hiddenLayers = t; threshold = 12.1851; attribute_eval = GainRatioAttributeEval; attribute_search = GreedyStepwise;

Multilayer Perceptron Network Subject 4: learningRate = 0.5675; momentum = 0.1054; hiddenLayers = t; lookupCacheSize = 2; searchTermination = 8; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

Multilayer Perceptron Network Subject 5: learningRate = 0.2185; momentum = 0.5771; hiddenLayers = a; folds = 6; minimumBucketSize = 49; threshold = 9.8556; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Multilayer Perceptron Network Subject 6: learningRate = 0.2126; momentum = 0.3269; hiddenLayers = i; folds = 2; minimumBucketSize = 11; threshold = 9.1441; attribute_eval = OneRAttributeEval; attribute_search = GreedyStepwise;

Multilayer Perceptron Network Subject 7: learningRate = 0.4645; momentum = 0.1866; hiddenLayers = o; folds = 2; minimumBucketSize = 1; threshold = 3.5890; attribute_eval = OneRAttributeEval; attribute_search = GreedyStepwise;

Multilayer Perceptron Network Subject 8: learningRate = 0.7832; momentum = 0.1490; hiddenLayers = t; attribute_search = NONE;

Multilayer Perceptron Network Subject 9: learningRate = 0.6786; momentum = 0.1688; hiddenLayers = t; attribute_search = NONE;

Multilayer Perceptron Network Subject 10: learningRate = 0.5633; momentum = 0.7133; hiddenLayers = a; attribute_search = NONE;

Naive Bayes Subject 1: searchTermination = 81; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

Naive Bayes Subject 2: searchTermination = 570; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

Naive Bayes Subject 3: searchTermination = 570; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

Naive Bayes Subject 4: threshold = 0.2939; attribute_eval = CorrelationAttributeEval; attribute_search = Ranker;

Naive Bayes Subject 5: centerData = True; maximumAttributeNames = 272; varianceCovered = 0.9964; lookupCacheSize = 0; searchTermination = 2; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

Naive Bayes Subject 6: searchTermination = 161; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

Naive Bayes Subject 7: centerData = False; maximumAttributeNames = -1; varianceCovered = 0.9598; lookupCacheSize = 0; searchTermination = 3; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

Naive Bayes Subject 8: centerData = True; maximumAttributeNames = 45; varianceCovered = 0.9972; lookupCacheSize = 2; searchTermination = 3; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

Naive Bayes Subject 9: centerData = True; maximumAttributeNames = 442; varianceCovered = 0.5276; lookupCacheSize = 1; searchTermination = 6; attribute_eval = PrincipalComponents; attribute_search = BestFirst;

Naive Bayes Subject 10: centerData = True; maximumAttributeNames = 180; varianceCovered = 0.9632; threshold = 4.5383; attribute_eval = PrincipalComponents; attribute_search = GreedyStepwise;

Random Forest Subject 1: numTrees = 57; features_HIDDEN = False; numFeatures = 0; depth_HIDDEN = False; maxDepth = 0; folds = 4; minimumBucketSize = 10; threshold = 5.4648; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Random Forest Subject 2: numTrees = 131; features_HIDDEN = False; numFeatures = 0; depth_HIDDEN = False; maxDepth = 0; lookupCacheSize = 2;

searchTermination = 3; attribute_eval = InfoGainAttributeEval; attribute_search = BestFirst;

Random Forest Subject 3: numTrees = 127; features_HIDDEN = False; numFeatures = 0; depth_HIDDEN = True; maxDepth = 16; folds = 13; minimumBucketSize = 14; threshold = 3.5659; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Random Forest Subject 4: numTrees = 128; features_HIDDEN = True; numFeatures = 8; depth_HIDDEN = True; maxDepth = 9; lookupCacheSize = 2; searchTermination = 4; attribute_eval = CfsSubsetEval; attribute_search = BestFirst;

Random Forest Subject 5: numTrees = 8; features_HIDDEN = True; numFeatures = 5; depth_HIDDEN = False; maxDepth = 0; threshold = 17.3189; attribute_eval = InfoGainAttributeEval; attribute_search = GreedyStepwise;

Random Forest Subject 6: numTrees = 51; features_HIDDEN = True; numFeatures = 31; depth_HIDDEN = False; maxDepth = 0; numNeighbours = 15; sigma = 2; lookupCacheSize = 2; searchTermination = 3; attribute_eval = ReliefFAttributeEval; attribute_search = BestFirst;

Random Forest Subject 7: numTrees = 189; features_HIDDEN = True; numFeatures = 5; depth_HIDDEN = False; maxDepth = 0; attribute_search = NONE;

Random Forest Subject 8: numTrees = 158; features_HIDDEN = True; numFeatures = 8; depth_HIDDEN = True; maxDepth = 10; attribute_search = NONE;

Random Forest Subject 9: numTrees = 8; features_HIDDEN = False; numFeatures = 0; depth_HIDDEN = False; maxDepth = 0; numNeighbours = 2; sigma = 1; searchTermination = 420; attribute_eval = ReliefFAttributeEval; attribute_search = GreedyStepwise;

Random Forest Subject 10: numTrees = 174; features_HIDDEN = False; numFeatures = 0; depth_HIDDEN = True; maxDepth = 4; lookupCacheSize = 0; searchTermination = 3; attribute_eval = InfoGainAttributeEval; attribute_search = BestFirst;

Support Vector Classifier Subject 1: c = 1.1213; filterType = 1; kernel = RBFKernel; gamma = 0.0027; numNeighbours = 61; sigma = 2; lookupCacheSize = 1; searchTermination = 2; attribute_eval = ReliefFAttributeEval; attribute_search = BestFirst;

Support Vector Classifier Subject 2: c = 0.5199; filterType = 0; kernel = PolyKernel; exponent = 1.3900; attribute_search = NONE;

Support Vector Classifier Subject 3: c = 1.2166; filterType = 2; kernel = PolyKernel; exponent = 3.4430; searchTermination = 54; attribute_eval = CfsSubsetEval; attribute_search = GreedyStepwise;

Support Vector Classifier Subject 4: c = 0.5196; filterType = 0; kernel = PolyKernel; exponent = 2.1511; threshold = 10.3495; attribute_eval = SymmetricalUncertAttributeEval; attribute_search = GreedyStepwise;

Support Vector Classifier Subject 5: c = 0.5710; filterType = 2; kernel = PolyKernel; exponent = 1.1682; folds = 9; minimumBucketSize = 11; threshold = 17.0519; attribute_eval = OneRAttributeEval; attribute_search = GreedyStepwise;

Support Vector Classifier Subject 6: c = 1.1873; filterType = 1; kernel = RBFKernel; gamma = 0.0059; attribute_search = NONE;

Support Vector Classifier Subject 7: c = 1.2884; filterType = 0; kernel = RBFKernel; gamma = 0.8826; folds = 11; minimumBucketSize = 4; threshold = 2.8913; attribute_eval = OneRAttributeEval; attribute_search = Ranker;

Support Vector Classifier Subject 8: c = 0.6114; filterType = 0; kernel = PolyKernel; exponent = 1.9743; attribute_search = NONE;

Support Vector Classifier Subject 9: c = 1.4620; filterType = 0; kernel = Puk; sigma = 4.2011; omega = 0.6600; attribute_search = NONE;

Support Vector Classifier Subject 10: c = 1.0273; filterType = 0; kernel = PolyKernel; exponent = 2.2284; attribute_search = NONE;

Appendix D: Scripts and Other Materials

The Matlab scripts and other materials that were created in the making of this thesis can be found in an open source online repository at:

`https://github.com/madism/ECMLA`

It contains the following items:

- `README.md` - contains the guidelines for running the scripts and the changelog.
- `Dataset_name_converter.m` - script for preprocessing the specified dataset before conversion with `mat2arffmod.m`.
- `Dataset_name_converted.mat` - the dataset after using `Dataset_name_converter.m`.
- `Dataset_name_converted.arff` - the dataset after using `mat2arffmod.m`.
- `Datasets` - folder which contains the datasets for each subject.
- `Graphs` - folder which contains the plotted clustergrams, correlation matrixes and heatmaps.
- `Parameters` - folder which contains more detailed information about the optimized parameters for each machine learning algorithm.
- `Results` - folder which contains the cross validation results for each dataset and files created by Weka 3 which are needed for plotting the results.
- `Scripts` - folder which contains the Matlab scripts created for this thesis.
- `mat2arffmod.m` - script which converts the each corresponding dataset from `.mat` format to `.arff` for Weka 3.
- `draw_heatmap.m` - script for drawing a heatmap for for each performance metric.
- `draw_signal.m` - script for drawing an example signal before and after Fourier Transform.
- `loadARFF.m` - script for loading data from a Weka .arff file into a Java Weka Instances object for use by Weka classes.

- `plotresults.m` - script for plotting a clustergram and a correlation matrix.
- `results_tables.mat` - tables containing the results obtained from Weka 3 and AutoWEKA for each algorithm.
- `weka2matlab.m` - script for converting Weka data, stored in a Java Weka Instance object, to a Matlab data type.
- `wekaPathCheck.m` - script for checking whether Weka has been added to system path which is needed for other scripts to work.

Licence

Non-exclusive license to reproduce and make this thesis public

I, Madis Masso (18.02.1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   (a) reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   (b) make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "Empirical Comparison of Machine Learning Algorithms Based on EEG Data", supervised by Ilya Kuzovkin and Kristjan Korjus,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu

January 21, 2016