Using Bayesian Optimization to Effectively Tune Random Forest and XGBoost Hyperparameters for Early Alzheimer's Disease Diagnosis

Louise Bloch[1,2](✉) and Christoph M. Friedrich[1,2]

[1] Department of Computer Science, University of Applied Sciences and Arts Dortmund, Emil-Figge-Str. 42, 44227 Dortmund, Germany
{louise.bloch,christoph.friedrich}@fh-dortmund.de

[2] Institute for Medical Informatics, Biometry and Epidemiology (IMIBE), University Hospital Essen, Essen, Germany

Abstract. Many research articles used Machine Learning (ML) for early detection of Alzheimer's Disease (AD) especially based on Magnetic Resonance Imaging (MRI). Most ML algorithms depend on a large number of hyperparameters. Those hyperparameters have a strong influence on the model performance and thus choosing good hyperparameters is important in ML. In this article, Bayesian Optimization (BO) was used to time-efficiently find good hyperparameters for Random Forest (RF) and eXtreme Gradient Boosting (XGBoost) models, which are based on four and seven hyperparameters and promise good classification results. Those models are applied to distinguish if mild cognitive impaired (MCI) subjects from the Alzheimer's disease neuroimaging initiative (ADNI) dataset will prospectively convert to AD. The results showed comparable cross-validation (CV) classification accuracies for models trained using BO and grid-search, whereas BO has been less time-consuming. The initial combinations for BO were set using Latin Hypercube Design (LHD) and via Random Initialization (RI). Furthermore, many models trained using BO achieved better classification results for the independent test dataset than the model based on the grid-search. The best model achieved an accuracy of 73.43% for the independent test dataset. This model was an XGBoost model trained with BO and RI.

Keywords: Bayesian optimization · Computer-aided diagnosis · Early Alzheimer's Disease diagnosis · eXtreme Gradient Boosting · Random Forests

1 Introduction

Alzheimer's Disease (AD) is a neurodegenerative disease [2] and the most frequent cause of dementia. The early identification of subjects at risk to develop AD is important to recruit and monitor subjects for therapy studies, as there currently is no causal therapy [2]. Subjects with Mild Cognitive Impairment (MCI)

J. Ye et al. (Eds.): MobiHealth 2020, LNICST 362, pp. 285–299, 2021.
https://doi.org/10.1007/978-3-030-70569-5_18

have a higher risk to develop AD [8] than cognitively normal (CN) controls. Thus, the prediction of future conversion to AD is important for MCI subjects. There have been many articles that used Machine Learning (ML) to improve the identification of those subjects. Most of them use models with a large number of hyperparameters.

Finding good hyperparameters is one of the key problems in ML. Hyperparameter tuning can improve model performance and prevent overfitting. For real-world problems, like the prediction of AD, good hyperparameters often depend on the data [34, p. 305]. Thus, parameter tuning is a complex and time-consuming task. One possibility to find good hyperparameters are optimization methods, which have the advantage to time-efficiently find robust parameters.

1.1 Prior Work

Many articles used ML models with a large number of hyperparameters to predict different AD stages. Some approaches used the default hyperparameters [4,20] to reduce the complexity of this problem. However, neither good performance nor high generalizability can be guaranteed. Other articles used grid-search [5,19,27], which is time-consuming for models with many hyperparameters. Some articles had no documentation about the hyperparameters at all. Only a few approaches used methods for time-efficient and stable hyperparameter optimization. In [18], Bayesian Optimization (BO) with Random Initialization (RI) was used to predict MCI conversion within three years. The hyperparameters of different ML models like Support Vector Machines (SVMs) [12] and Random Forests (RFs) [6] were tuned for 353 subjects with stable MCI (sMCI) and 193 with progressive MCI (pMCI) from the Alzheimer's Disease Neuroimaging Initiative (ADNI) [29] cohort. The feature set included sociodemographic and clinical characteristics, neuropsychological tests, and the baseline (BL) MCI type. The final ensemble model achieved an Area Under the Receiver Operating Characteristic (AUROC) of 0.88.

BO with RI has been also used in [28] to optimize the hyperparameters of a Deep Neural Network (DNN). Two datasets, which are available online (https://github.com/ChihyunPark/DNN_for_ADprediction. Last accessed 8 Aug 2020), were used. The first one included large-scale gene expressions from 257 CN and 439 AD subjects and the second one contained Deoxyribonucleic Acid (DNA) methylation data of 68 CN and 74 AD subjects. The final model achieved an accuracy of 82.3% for the test dataset.

[32] used BO with RI to tune the parameters of a radial-basis SVM and classifies 17 subjects with Subjective Cognitive Impairment (SCI) vs. 53 MCI vs. 50 AD. The dataset was not publicly available. The feature set included neuropsychological tests and the results of a reaction test. Accuracies of 80.6% and 65.0% were reached for MCI vs. AD and SCI vs. MCI vs. AD classification.

This article aims to efficiently tune the parameters of RFs and eXtreme Gradient Boosting (XGBoost) models for early AD diagnosis. In addition to the previous articles, a Latin Hypercube Design (LHD) [25] was used to initialize the BO. The results of this method were compared to a RI, a grid-search and

Table 1. ADNI demographics at BL. p-values are calculated using Mann-Whitney-U-test for continuous variables and χ^2-test for frequency variables.

	sMCI	pMCI	Σ	p-value
n	401	319	720	
Age in years (mean $\pm$ sd)	73.2 $\pm$ 7.5	74.0 $\pm$ 7.1	73.5 $\pm$ 7.3	0.1156
Gender (proportion of males)	59.6%	59.9%	59.7%	1.000
MMSE (mean $\pm$ sd)	27.8 $\pm$ 1.8	27.0 $\pm$ 1.7	27.4 $\pm$ 1.8	<0.0001
CDR (mean $\pm$ sd)	0.5 $\pm$ 0.0	0.5 $\pm$ 0.0	0.5 $\pm$ 0.0	0.2634
ApoEϵ4 (count of ApoEϵ4 alleles): 0	56.9%	34.2%	46.8%	<0.0001
ApoEϵ4: 1	33.9%	49.5%	40.8%	
ApoEϵ4: 2	9.2%	16.3%	12.4%	
Time to final diagnosis in months (mean $\pm$ sd)	47.1 $\pm$ 32.4	30.6 $\pm$ 24.7	39.8 $\pm$ 30.4	<0.0001

the default parameters. Section 2 presents the dataset and methods. The ML workflow is described in Sect. 3. The experimental results are demonstrated in Section 4 and finally discussed in Sect. 5.

2 Materials and Methods

2.1 Dataset

Data used in the preparation of this article were obtained from the ADNI [29] cohort. 720 subjects of the study phases ADNI-1 (354 subjects), ADNIGO (92 subjects) and ADNI-2 (274 subjects) were selected. All subjects had a BL diagnosis of MCI and were classified as sMCI if all subsequent diagnoses correspond to MCI and as pMCI if they converted to AD at any visit and AD was the diagnosis for all subsequent visits. Subjects who reverted to CN or MCI were excluded from this study. The demographic data are summarized in Table 1. The time between the BL and the final diagnosis ranged between 4.7 and 156.2 months.

For each subject, one fully preprocessed [21] BL 1.5 T or 3 T T1-weighted Magnetization-Prepared Rapid Gradient-Echo (MP-RAGE) Magnetic Resonance Imaging (MRI) scan was selected. FreeSurfer v6.0 [15] extracted volumes of 34 cortical Regions of Interest (ROIs) per hemisphere, defined in Desikan-Killiany atlas [13], 34 subcortical ROIs [16], and the estimated Total Intracranial Volume (eTIV). The resulting 103 MRI features were normalized by eTIV [33].

Two different datasets were used for model training. *Dataset 1* included 106 features - MRI-features, age, gender and count of Apolipoprotein E ϵ4 (ApoEϵ4) alleles. *Dataset 2* added Mini-Mental State Examination (MMSE), a logical long-term (LDEL) and short-term memory test (LIMM) resulting in 109 features. Clinical Dementia Rating (CDR) was excluded due to small variance.

2.2 eXtreme Gradient Boosting

Boosting algorithms assume that the iterative combination of multiple weak classifiers leads to a strong classifier. Gradient boosting [17] meets this assumption by training the first classifier to learn the independent variable and the subsequent classifiers to learn the gradients of the previous classifiers. The gradients $l(y_i, \hat{y}_i^{(t-1)})$ are defined as the deviation between the additive classification $\hat{y}_i^{(t-1)}$ of the previous iteration $(t-1)$ and the correct classification y_i of observation i. The loss function $L^{(t)}$ at iteration t using n observations corresponds to Eq. 1. Here, f_t represents the weak classifier at iteration t and $\Omega(f_t)$ is a regularization term which controls the complexity of the classifier.

$$L^{(t)} = \sum_{i=1}^{n} l\big(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\big) + \Omega(f_t) \tag{1}$$

The additive combination of all weak classifiers f_k determines the final classification $\hat{y}_i$ for observation i, as can be seen in Eq. 2.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{2}$$

eXtreme Gradient Boosting (XGBoost) [10] is an open-source software library and an implementation of gradient boosting with a high focus on scalability, parallelization and distributed execution. XGBoost with Classification and regression trees (CARTs) [7] as weak classifier depends on seven hyperparameters, summarized in Table 2. *nrounds* (n) determines the number of iterations in the training process. The learning rate *eta* (η) controls the influence of each weak classifier on the final model and thus prevents overfitting. The hyperparameter *gamma* (γ) determines the minimum loss reduction required to specialize leaf nodes. High values lead to preserving models. *max_depth* (d_{max}) specifies the maximum depth of a tree. Deep models are more complex and prone to overfitting. The parameter *min_child_weights* (w_{min}) sets the minimum number of weighted observations in a child node. High values for *min_child_weights* achieve more conservative models. *subsample* (s) sets the ratio of training instances randomly selected in each iteration. Small values prevent overfitting. *colsample_bytree* (c) is the ratio of randomly subsampled features in each iteration. Small values lead to robust models, but values near zero lead to poor results.

2.3 Random Forest

Random Forests (RFs) [6] are based on multiple CARTs and the majority voting is used to robustly predict an unknown observation. Table 3 summarizes the hyperparameters for the RF. n_{tree} sets the number of trees in the RF. Training only a few trees often leads to less accurate results. For each tree, a bootstrap sample [14] of the dataset is generated and for each split, a subset of *mtry* (m_{try}) features are randomly chosen to train the models. The higher *mtry*, the higher

Table 2. XGBoost parameters and intervals. The grid is a grid-search of length five.

Name	Minimum	Maximum	Grid
nrounds (n)	1	500	{1.00, 125.00, 250.00, 375.00, 500.00}
eta (η)	0	1	{0.00, 0.25, 0.50, 0.75, 1.00}
gamma (γ)	0	20	{0.00, 5.00, 10.00, 15.00, 20.00}
max_depth (d_{max})	1	20	{1.00, 5.00, 10.00, 15.00, 20.00}
min_child_weights (w_{min})	1	30	{1.00, 8.25, 15.50, 22.75, 30.00}
subsample (s)	0	1	{0.00, 0.25, 0.50, 0.75, 1.00}
colsample_bytree (c)	0	1	{0.00, 0.25, 0.50, 0.75, 1.00}

Table 3. RF parameters and intervals. The grid refers to a grid-search of length five.

Name	Minimum	Maximum	Grid
mtry (for *dataset 1*) (m_{try})	2	109	{2, 28, 55, 82, 109}
mtry (for *dataset 2*) (m_{try})	2	112	{2, 29, 57, 84, 112}
ntree (n_{tree})	250	1250	{250, 500, 750, 1000, 1250}
nodesize (s_{min})	1	20	{1, 5, 10, 15, 20}
maxnodes (nd_{max})	50	100	{50, 62, 75, 87, 100}

is the risk of overfitting. *nodesize* (s_{min}) sets the minimum size of terminal nodes for each tree. The smaller *nodesize*, the less robust the trained models are. Hyperparameter *maxnodes* (nd_{max}) specifies the maximum number of terminal nodes for each tree. Trees with many terminal nodes tend to overfit the dataset.

2.4 Latin Hypercube Design

Latin hypercube design (LHD) [25] is a method to generate a nearly random sample based on a multi-dimensional distribution. The objective is to select p samples from a q-dimensional space. To generate an LHD each dimension is split into p equidistant intervals. One value is randomly selected per interval, resulting in p parameters for each dimension. The parameters of the individual dimensions are randomly merged to p samples with q dimensions. LHD ensures complete coverage of the range for each variable.

2.5 Bayesian Optimization

Bayesian Optimization (BO) [26] is a global optimization method for black-box functions. In this research, hyperparameter tuning has been considered as the optimization of a black-box function. The model performance was maximized dependently on the hyperparameters. First, a set of initial parameter combinations were arranged. In this article, LHD and RI were used for this purpose. The models were evaluated for each combination to estimate their performance. A Gaussian Process (GP) was fitted to model the relationship between parameter combinations and model performances. This GP was optimized to find the next promising parameter combination. The optimization considered exploration and exploitation using an acquisition function, which depends on the expected model performance $\hat{\mu}_{\Theta}$ and the covariance $\hat{\Sigma}_{\Theta}$ at parameter combination Θ. The covariance $\hat{\Sigma}_{\Theta}$ was smaller the closer previously examined parameter combinations were. In this work, the upper confidence bounds (UCB) [1,22], given in Eq. 3, was used as the acquisition function. The parameter κ determines, the proportion between exploitation and exploration. For higher values of κ, exploration is preferred, whereas lower values favour exploitation.

$$UCB(\Theta) = \hat{\mu}_{\Theta} + \kappa \cdot \hat{\Sigma}_{\Theta} \tag{3}$$

The performance of the ML model was evaluated using the new parameter combination and added to the GP model. This process was repeated until previously determined criteria, e.g. a maximum number of iterations, were met.

3 Machine Learning Workflow

In this article, an ML workflow was implemented, using the programming language R v3.5.3 [30], to distinguish sMCI and pMCI subjects. Figure 1 gives an overview of the workflow. Subject selection and image processing are described in Sect. 2.1. The subjects of each diagnosis group were randomly split into a training and an independent test dataset. The test dataset contained 20% of the original dataset and the remaining 80% were used to train the model and tune the hyperparameters. LHD, implemented using the R package SPOT v2.0.3 [3], and RI were used to generate ten initial parameter combinations for the BO. After training the initial BO model, promising parameter combinations were successively determined and evaluated. 25 parameter combinations were generated by BO to tune the hyperparameters of RF and XGBoost models. BO, XGBoost and RF were implemented using the R packages rBayesianOptimization v1.1.0 [35], xgboost v0.82.1 [11] and randomForest v4.6-14 [24]. 10×10-fold Cross-Validation (CV) [31] was used as a resampling strategy and was implemented using the R package caret v6.0-82 [23], by splitting the training dataset into ten distinct folds. Ten iterations were performed with a different fold used as validation dataset in each iteration and the remaining nine folds were used to train the model. This procedure was repeated ten times, whereas the data was shuffled and stratified in each repetition. CV-accuracy was used as the metric for BO. The best parameters were selected to train the final model. The preprocessing, nested in the tuning workflow, included centering, scaling and median imputation. Synthetic Minority Over-sampling Technique (SMOTE) [9] compensated class imbalances during the parameter tuning. The final model was evaluated for the independent test dataset.

As a comparison, a grid-search was implemented using the R package caret v6.0-82 [23]. The grid contained the cartesian product with five values per parameter, which results in $5^7 = 78125$ XGBoost and $5^4 = 525$ RF combinations.

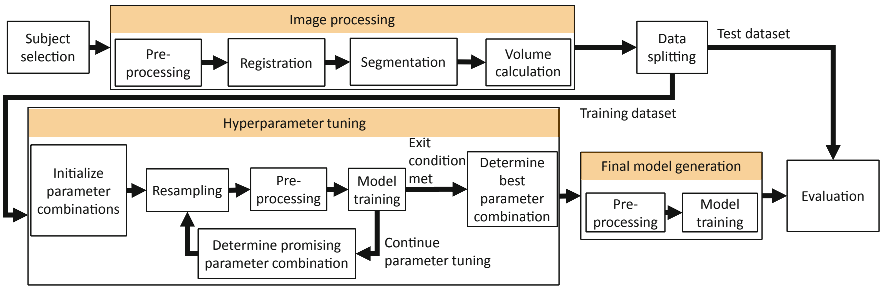

Fig. 1. Machine learning workflow.

4 Results

In the experiments, BO has been applied to optimize four hyperparameters of an RF classifier (Sect. 4.1) and seven hyperparameters of an XGBoost model (Sect. 4.2). This optimization has been demonstrated on two previously described AD datasets. *Dataset 1* included volumetric MRI features, demographics, and ApoEϵ4, and *dataset 2* added cognitive test results to *dataset 1*.

The experiments, that used BO, included ten initial parameter combinations and 25 combinations during optimization, resulting in 35 evaluations. The grid-search models used five values per hyperparameter m, resulting in 5^m and thus $5^4 = 625$ RF and $5^7 = 78125$ XGBoost grid combinations. Thus, the number of grid-search evaluations increased exponentially with the number of hyperparameters, while the number of evaluations was constant for BO. BO was applied using five different values for the parameter κ ($\kappa \in \{0.5, 1.0, 2.0, 5.0, 10.0\}$).

4.1 Bayesian Optimization for Random Forest Classifiers

Dataset 1. Table 4 summarizes the RF results for the different hyperparameter tuning methods on *dataset 1*. The best CV-results were 69.76% and achieved using BO with RI and $\kappa = 2.0$. The grid-search model performed 0.30% worse than the best model. The default parameter model reached the worst accuracy of 68.20%. The LHD BO models obtained CV-accuracies between those values. BO with RI outperformed the LHD initialization for this dataset and CV-results.

The best accuracy for the independent test dataset was 67.83%, achieved by the grid-search and the BO model with LHD initialization and $\kappa = 2.0$. All LHD BO models except the model with $\kappa = 2.0$ selected the same hyperparameters and thus obtained equal results. The worst accuracy for the independent test set was 62.94%, reached by the BO model with RI and $\kappa = 5.0$. The performances for the independent test dataset differed by 4.89% between the tuning methods.

The boxplots in Fig. 2 show the relations between grid-search parameters and the mean CV-accuracies. The best performances for the hyperparameter *mtry* were obtained for a value of 28. Consistently, all BO models selected *mtry* values between 25 and 35. Increasing values of *ntree* led to better model performances.

Table 4. Classification results and RF hyperparameters achieved for *dataset 1*. Comparison of default parameters, grid-search and BO with RI and LHD initialization for parameter tuning. The best results are highlighted in bold.

Hyperparameter optimization	m_{try}	n_{tree}	s_{min}	nd_{max}	CV-accuracy (mean ± sd) in %	Test accuracy in %
Default parameters	10	500	1	max	68.20 ± 6.47	65.73
Grid-search	28	1000	15	50	69.46 ± 6.53	**67.83**
BO RI $\kappa = 0.5$	25	1107	8	50	69.26 ± 6.30	66.43
BO RI $\kappa = 1.0$	30	615	8	50	69.28 ± 6.34	67.13
BO RI $\kappa = 2.0$	35	464	1	56	**69.76 ± 6.35**	65.73
BO RI $\kappa = 5.0$	30	1216	1	50	69.28 ± 6.54	62.94
BO RI $\kappa = 10.0$	33	647	3	81	69.10 ± 6.48	67.13
BO LHD $\kappa \in \{0.5, 1.0, 5.0, 10.0\}$	27	808	16	96	68.79 ± 5.99	65.73
BO LHD $\kappa = 2.0$	29	1250	3	68	68.96 ± 6.16	**67.83**

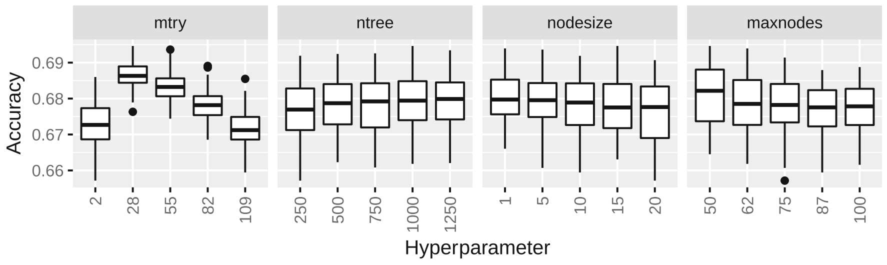

Fig. 2. Boxplots summarizing the mean CV-accuracies for RF grid-search hyperparameters and *dataset 1*.

A slight decrease was detected on the mean CV-accuracy for increasing values of *nodesize*. The hyperparameter *maxnodes* obtained better results for a value of 50 and the performance decreased for higher values. The observations were mainly reflected in the BO model with LHD initialization and $\kappa = 2.0$ and all RI models except for $\kappa = 10.0$. The other models selected deviating parameters.

Dataset 2. *Dataset 2* supplemented cognitive test results to *dataset 1*. However, the experiments executed on the datasets do not differ and the results are summarized in Table 5. CV-results of the RF models trained on *dataset 2* were between 70.01%, achieved by the default parameter model and 70.68% and thus differ by 0.67%. The BO model with LHD initialization and $\kappa = 2.0$ obtained the best results. The CV-results on *dataset 2* outperformed the results for *dataset 1*. The grid-search model reached a CV-accuracy of 70.61%.

The results for the independent test dataset were similar to the CV-results. The worst result of 67.13% for the independent test dataset was achieved for

Table 5. Classification results and RF hyperparameters achieved for *dataset 2*. Comparison of default parameters, grid-search and BO with RI and LHD initialization for parameter tuning. The best results are highlighted in bold.

Hyperparameter optimization	m_{try}	n_{tree}	s_{min}	nd_{max}	CV-accuracy (mean ± sd) in %	Test accuracy in %
Default parameters	10	500	1	max	70.01 ± 6.41	69.23
Grid-search	29	1000	10	100	70.61 ± 6.53	69.93
BO RI $\kappa = 0.5$	11	959	9	60	70.52 ± 5.93	69.23
BO RI $\kappa = 1.0$	21	1250	16	72	70.61 ± 5.78	70.63
BO RI $\kappa = 2.0$	13	756	5	74	70.42 ± 6.10	69.23
BO RI $\kappa = 5.0$	18	1250	1	95	70.42 ± 5.91	69.23
BO RI $\kappa = 10.0$	39	1250	1	100	70.19 ± 6.13	**71.33**
BO LHD $\kappa = 0.5$	13	1250	16	92	70.48 ± 5.92	69.23
BO LHD $\kappa = 1.0$	16	757	20	62	70.23 ± 5.80	68.53
BO LHD $\kappa = 2.0$	20	1250	11	85	**70.68 ± 6.01**	67.13
BO LHD $\kappa = 5.0$	16	808	16	96	70.21 ± 6.32	67.83
BO LHD $\kappa = 10.0$	23	1250	1	82	70.49 ± 6.02	**71.33**

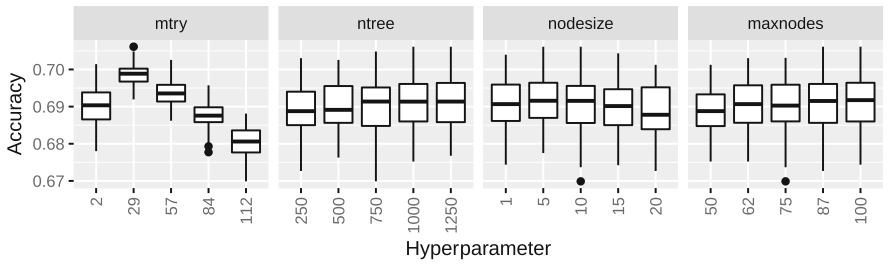

Fig. 3. Boxplots summarizing the mean CV-accuracies for RF grid-search hyperparameters and *dataset 2*.

the BO model with LHD initialization and $\kappa = 2.0$. This model achieved the best CV-results. The best accuracy for the independent test dataset of 71.33% has been reached for the BO models with $\kappa = 10.0$ and both LHD and RI. The grid-search model reached an accuracy of 69.93%.

The boxplots in Fig. 3 illustrate the mean CV-accuracies depending on the grid-search hyperparameters. The hyperparameters *mtry* and *ntree* showed similar relations as those observed for *dataset 1*. All BOs, except the model with RI and $\kappa = 10.0$, selected values between 11 and 23 for the *mtry* parameter. BO preferred high values for *ntree*. *nodesize* showed slightly worse results for a value of 20 in the boxplots, however, BO with LHD initialization and $\kappa = 1.0$ selected this value. *maxnodes* showed slightly increasing results for higher values. All BO models selected values higher than 62.

Table 6. Classification results and XGBoost hyperparameters for *dataset 1*. Using default parameters, grid-search and BO with RI and LHD initialization for parameter tuning. CV-accuracies are given as mean ± sd. The best results are highlighted in bold.

Hyperparameter optimization	n	d_{max}	η	γ	c	w_{min}	s	CV-accuracy in %	Test accuracy in %
Default parameters	100	6	0.300	0.000	1.000	1.000	1.000	65.29 ± 6.81	68.53
Grid-search	250	20	0.250	10.000	0.750	1.000	1.000	66.64 ± 6.46	60.84
BO RI $\kappa = 0.5$	490	7	0.020	10.263	0.190	2.580	0.924	66.26 ± 5.56	64.34
BO RI $\kappa = 1.0$	484	16	0.186	7.477	0.071	2.847	0.363	64.55 ± 5.67	62.24
BO RI $\kappa = 2.0$	483	14	0.095	19.642	0.283	17.813	0.200	66.44 ± 5.68	65.03
BO RI $\kappa = 5.0$	110	20	0.163	1.526	0.525	1.000	0.612	66.09 ± 5.69	71.33
BO RI $\kappa = 10.0$	500	20	0.200	0.000	0.439	1.000	0.935	65.57 ± 5.60	**73.43**
BO LHD $\kappa = 0.5$	149	6	0.010	11.364	0.781	8.062	0.817	66.59 ± 5.24	60.84
BO LHD $\kappa = 1.0$	452	2	0.085	20.000	1.000	1.000	1.000	**66.75 ± 4.97**	63.64
BO LHD $\kappa = 2.0$	426	1	0.120	0.371	0.349	17.080	0.746	65.14 ± 5.80	67.83
BO LHD $\kappa = 5.0$	50	1	0.171	20.000	0.994	19.490	0.654	66.54 ± 5.05	60.14
BO LHD $\kappa = 10.0$	193	1	0.045	0.000	1.000	29.822	1.000	66.66 ± 5.38	64.34

4.2 Bayesian Optimization for XGBoost Classifiers

Dataset 1. Table 6 summarizes the XGBoost results for *dataset 1*. Seven hyperparameters were tuned in these experiments. All models achieved similar CV-accuracies. The best CV-accuracy was 66.75% for the BO with LHD initialization and $\kappa = 1.0$. The worst CV-accuracy of 64.55% was reached by the same model but RI. The grid-search CV-accuracy was 66.64%.

The best result of 73.43% for the independent test dataset was achieved using BO with RI and $\kappa = 10.0$. The accuracy for the independent test set exceeds the mean CV-accuracy of this model which was 65.57%. The XGBoost results for the independent test dataset differed more than the RF results. The grid-search model achieved a worse performance of 60.84% for the independent test dataset. The BO model with $\kappa = 5.0$ reached the worst accuracy of 60.14%.

The boxplots in Fig. 4 summarize the relations between the grid-search hyperparameters and the mean CV-accuracies. All 28125 observations with a value of 0.00 for *eta* or *subsample* were excluded because a learning rate of 0.00 means that there is no learning effect and a subsampling of 0.00% led to a model training without any subjects. All excluded results achieved mean CV-accuracies less than 45.00%, which would distort the interpretability of the figure. All boxplots had a large number of outliers below the box. Small values for *eta* were associated with better results. Consistently, all BO models selected values between 0.010 and 0.200. The minimum value of 0.000 for hyperparameters *gamma* and *colsample_bytree* and 1 for *nrounds*, showed stronger variations in the results than the remaining values. Using only one boosting iteration led to worse results. Small differences were detected between using 125 and 500 boosting iterations. The BO selected values between 50 and 500 for this parameter.

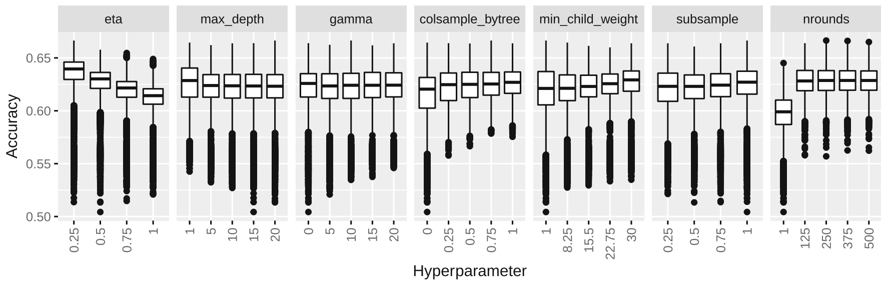

Fig. 4. Boxplots summarizing the mean CV-accuracies for XGBoost grid-search hyperparameters and *dataset 1*. All combinations with *eta* or *subsample* = 0.00 were excluded.

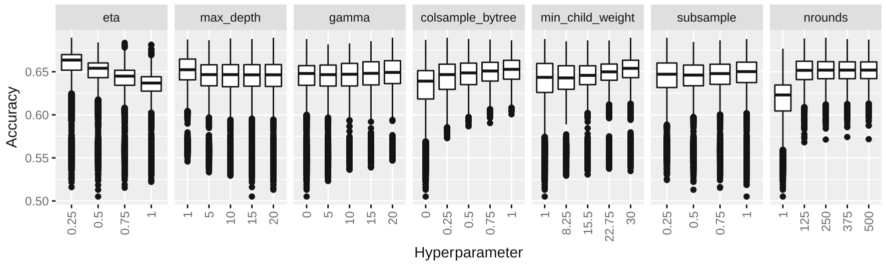

Fig. 5. Boxplots summarizing the mean CV-accuracies for XGBoost grid-search hyperparameters and *dataset 2*. All combinations with *eta* or *subsample*=0.00 were excluded.

Dataset 2. Table 7 shows the results achieved by training an XGBoost model with *dataset 2*. The achieved CV-accuracies exceed the results of *dataset 1*. The best CV-accuracy was 69.14% ± 5.48%, reached by the BO model with RI and $\kappa = 1.0$. The grid-search model achieved a CV-accuracy of 68.95% ± 6.25% and the default parameter model a CV-accuracy of 68.08% ± 6.61%.

The results for the independent test dataset were between 65.03%, for the BO with RI and $\kappa = 2.0$ and 71.33%, for the default parameter model and the BO model with LHD initialiation and $\kappa = 5.0$. The grid-search model achieved an accuracy of 69.93% for the independent test dataset.

The boxplots in Fig. 5 show the effects of the grid-search hyperparameters on the mean CV-results. Consistently with Fig. 4, all examinations with an *eta* or *subsample* value of 0.00 were excluded, as they represent random models. For the parameter *eta*, small values performed better than large ones. This observation is consistent with the BO parameter selection. The hyperparameters *gamma*, *colsample_bytree* and *min_child_weight* showed slightly better results the higher the parameter values. BO showed no clear focus for these parameters. However, all BO models with RI, except the model with $\kappa = 5.0$ selected values higher than 0.929 for *colsample_bytree*. The BO and the boxplots show, that using only one boosting iteration has a negative effect on the mean CV-accuracy.

Table 7. Classification results and XGBoost hyperparameters for *dataset 2*. Using default parameters, grid-search and BO with RI and LHD initialization for parameter tuning. CV-accuracies are given as mean ± sd. The best results are highlighted in bold.

Hyperparameter optimization	n	d_{max}	η	γ	c	w_{min}	s	CV-accuracy in %	Test accuracy in %
Default parameters	100	6	0.300	0.000	1.000	1.000	1.000	68.08 ± 6.61	**71.33**
Grid-search	250	20	0.250	20.000	0.250	30.000	0.250	68.95 ± 6.25	69.93
BO RI κ = 0.5	127	8	0.143	20.000	1.000	20.442	0.326	68.30 ± 5.42	67.83
BO RI κ = 1.0	359	14	0.146	3.037	1.000	30.000	0.174	**69.14 ± 5.48**	69.93
BO RI κ = 2.0	481	10	0.102	14.519	0.961	13.497	0.321	68.68 ± 5.53	65.03
BO RI κ = 5.0	418	3	0.134	14.849	0.280	23.008	0.316	68.20 ± 5.12	69.23
BO RI κ = 10.0	452	20	0.138	19.556	0.929	25.000	0.834	67.63 ± 5.55	65.73
BO LHD κ = 0.5	357	11	0.077	12.699	0.298	28.618	0.480	68.14 ± 5.32	70.63
BO LHD κ = 1.0	323	4	0.026	0.000	0.372	8.693	0.847	68.08 ± 6.04	69.23
BO LHD κ = 2.0	259	19	0.088	9.922	0.696	19.255	0.595	68.21 ± 5.92	67.83
BO LHD κ = 5.0	447	5	0.104	6.379	0.718	1.938	0.441	67.56 ± 5.95	**71.33**
BO LHD κ = 10.0	300	14	0.002	7.291	0.475	10.290	0.683	68.18 ± 6.03	66.43

5 Conclusions

In this article, BO has been used to time-efficiently find hyperparameters for MCI-conversion prediction based on MRI volumetrics, demographics, ApoEϵ4 features and cognitive test results. As a comparison, a time-consuming grid-search has been implemented. The XGBoost and RF models were evaluated using 10 × 10-fold-CV, a robust resampling method, and an additional evaluation for an independent test dataset. The outcomes showed that BO was able to find parameters which can keep up with the time-efficient grid-search and is thus most interesting for models with many hyperparameters. Some tendencies for good hyperparameter choices which were detected considering the grid-search models can be also recognized for the BO parameter selection. Thus, BO offered a trade-off between the time-efficiency and robust, reproducible models.

The approach was applied for two different AD datasets of the ADNI cohort. *Dataset 1* included MRI volumetric, demographic and ApoEϵ4 features and *dataset 2* additionally included BL cognitive test results. The results of the RF models showed better accuracies for models trained on *dataset 2*. The best result for the independent test dataset was achieved for *dataset 1* and an XGBoost model. The outcomes showed promising results for the models trained using BO for hyperparameter optimization. For both datasets and both ML techniques, the best CV-accuracies were achieved using BO. This observation could also be confirmed for the independent test dataset except for the RF models trained on *dataset 1*. In this case, BO and grid-search achieved the same accuracy. Comparing CV-accuracies of the XGBoost and RF models, better results were achieved by the RF models. The results for the independent test dataset showed a different observation because two XGBoost models achieved outstanding results. No major differences were detected between randomly initialized BO and BO with LHD initialization. Some of the model errors for pMCI subjects can be traced back to a large distance in time between BL diagnosis and conversion diagnosis. For these subjects, a classifier depending on longitudinal input data might be more expedient. Future studies should validate the results for different AD cohorts. Both classifiers in this article were tree-based models. Thus, it should be investigated in future research, how BO and LHD initialization works for different ML models. The use of alternative optimization methods such as Sequential Parameter Optimization (SPO) [3] might be another promising research approach.

Acknowledgment. Data used in the preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database. We thank the patients and ADNI for the availability of the data.

The work of Louise Bloch was partially funded by a PhD grant from the University of Applied Sciences and Arts Dortmund, Germany.

References

1. Agrawal, R.: Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. Adv. Appl. Prob. **27**(4), 1054–1078 (1995). https://doi.org/10.2307/1427934
2. Alzheimer's Association: 2020 Alzheimer's Disease facts and figures. Alzheimer's Dement. **16**(3), 391–460 (2020). https://doi.org/10.1002/alz.12068
3. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 773–780 (2005). https://doi.org/10.1109/cec.2005.1554761
4. Benussi, A., et al.: Classification accuracy of transcranial magnetic stimulation for the diagnosis of neurodegenerative dementias. Ann. Neurol. **87**(3), 394–404 (2020). https://doi.org/10.1002/ana.25677
5. Bloch, L., Friedrich, C.M.: Classification of Alzheimer's disease using volumetric features of multiple MRI scans. In: Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 2396–2401, July 2019. https://doi.org/10.1109/EMBC.2019.8857188
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
7. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees, 1st edn. CRC Press, Boca Raton (1984). https://doi.org/10.1201/9781315139470
8. Burns, A., Iliffe, S.: Alzheimer's disease. BMJ **338** (2009). https://doi.org/10.1136/bmj.b158
9. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**(1), 321–357 (2002). https://doi.org/10.1613/jair.953
10. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. New York, August 2016. https://doi.org/10.1145/2939672.2939785
11. Chen, T., et al.: XGBoost: eXtreme Gradient Boosting. R package v0.82.1 (2019). https://CRAN.R-project.org/package=xgboost. Accessed 5 Aug 2020

12. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
13. Desikan, R.S., et al.: An automated labeling system for subdividing the human cerebral cortex on MRI scans into GYRAL based regions of interest. NeuroImage **31**(3), 968–980, August 2006. https://doi.org/10.1016/j.neuroimage.2006.01.021
14. Efron, B., Tibshirani, R.: Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Stat. Sci. **1**(1), 54–75 (1986). https://doi.org/10.1214/ss/1177013815
15. Fischl, B.: FreeSurfer. NeuroImage **62**(2), 774–781 (2012). https://doi.org/10.1016/j.neuroimage.2012.01.021
16. Fischl, B., et al.: Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. Neuron **33**(3), 341–355 (2002). https://doi.org/10.1016/S0896-6273(02)00569-X
17. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001). https://doi.org/10.1214/aos/1013203451
18. Grassi, M., et al.: Alzheimer's disease neuroimaging initiative: a novel ensemble-based machine learning algorithm to predict the conversion from mild cognitive impairment to Alzheimer's disease using socio-demographic characteristics, clinical information, and neuropsychological measures. Front. Neurol. **10**, 756 (2019). https://doi.org/10.3389/fneur.2019.00756
19. Gupta, Y., Lama, R.K., Kwon, G.R., Alzheimer's disease neuroimaging initiative: prediction and classification of alzheimer's disease based on combined features from Apolipoprotein-E genotype, cerebrospinal fluid, MR, and FDG-PET imaging biomarkers. Front. Comput. Neurosci. **13**, 72 (2019). https://doi.org/10.3389/fncom.2019.00072
20. Hon, M., Khan, N.M.: Towards Alzheimer's disease classification through transfer learning. In: Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1166–1169, November 2017. https://doi.org/10.1109/BIBM.2017.8217822
21. Jack Jr., et al.: Magnetic resonance imaging in Alzheimer's disease neuroimaging initiative 2. Alzheimer's Dement. **11**(7), 740–756 (2015). https://doi.org/10.1016/j.jalz.2015.05.002
22. Katehakis, M.N., Robbins, H.: Sequential choice from several populations. Proc. Nat. Acad. Sci. **92**(19), 8584–8585 (1995). https://doi.org/10.1073/pnas.92.19.8584
23. Kuhn, M.: Caret: Classification and Regression Training. R package v6.0-82 (2019). https://CRAN.R-project.org/package=caret. Accessed 5 Aug 2020
24. Liaw, A., Wiener, M.: Classification and regression by random forest. R News vol. 2, no. 3, pp. 18–22 (2002). https://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf. Accessed 12 Aug 2020
25. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979). https://doi.org/10.2307/1268522
26. Močkus, J.: On Bayesian methods for seeking the extremum. In: Marchuk, G.I. (ed.) Optimization Techniques 1974. LNCS, vol. 27, pp. 400–404. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07165-2_55
27. Oh, K., Chung, Y.C., Kim, K., Kim, W.S., Oh, I.S.: Classification and visualization of Alzheimer's disease using volumetric convolutional neural network and transfer learning. Sci. Rep. **9** (2019). https://doi.org/10.1038/s41598-019-54548-6

28. Park, C., Ha, J., Park, S.: Prediction of Alzheimer's disease based on deep neural network by integrating gene expression and DNA methylation dataset. Expert Syst. Appl. **140**, 112873 (2020). https://doi.org/10.1016/j.eswa.2019.112873
29. Petersen, R.C., et al.: Alzheimer's disease neuroimaging initiative (ADNI). Neurology **74**(3), 201–209 (2010). https://doi.org/10.1212/WNL.0b013e3181cb3e25
30. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2019). https://www.R-project.org/. Accessed 5 Aug 2020
31. Refaeilzadeh, P., Tang, L., Liu, H.: Cross-validation. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 532–538, Springer, US, Boston, MA (2009). https://doi.org/10.1007/978-0-387-39940-9_565
32. Wallert, J., Westman, E., Ulinder, J., Annerstedt, M., Terzis, B., Ekman, U.: Differentiating patients at the memory clinic with simple reaction time variables: a predictive modeling approach using support vector machines and Bayesian optimization. Front. Aging Neurosci. **10**, 144 (2018). https://doi.org/10.3389/fnagi.2018.00144
33. Westman, E., Aguilar, C., Muehlboeck, J.S., Simmons, A.: Regional magnetic resonance imaging measures for multivariate analysis in alzheimer's disease and mild cognitive impairment. Brain Topogr. **26**(1), 9–23 (2012). https://doi.org/10.1007/s10548-012-0246-x
34. Witten, I.H., Frank, E., Hall, M.A. (eds.): Data mining: practical machine learning tools and techniques. In: The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, 3rd edn. (2011). https://doi.org/10.1016/B978-0-12-374856-0.00023-7
35. Yan, Y.: rBayesianOptimization: Bayesian Optimization of Hyperparameters. R package v1.1.0 (2016). https://CRAN.R-project.org/package=rBayesianOptimization. Accessed 5 Aug 2020