since 1961

BALTICA Volume 37 Number 2 December 2024

Baltica

An International Journal on Geosciences of the circum-Baltic States

https://baltica.gamtc.lt/en

Supporting Online Material for

HYDRAULIC CONDUCTIVITY DETERMINATION OF LITHUANIAN SOILS USING MACHINE LEARNING

V. Samalavičius, E. K.-M. Vanhala, I. Lekstutytė, S. Gadeikienė, S. Gadeikis, G. Žaržojus

Published online 9 December 2024, Baltica vol. 37 no. 2

DOI: https://doi.org/10.5200/baltica.2024.2.5

PDF file includes text

Hyperparameters used for machine learning algorithms' boosting

The Random Forest Regressor (RFR) was tested with several hyperparameters to evaluate its performance. The `n_estimators` parameter, which defines the number of trees in the forest, was set to [50, 100, 200]. This variation helps in understanding the effect of the number of trees on the model's accuracy and overfitting tendency. The `max_depth` parameter, which determines the maximum depth of each tree, was tested with values [None, 10, 20, 30]. This helps in controlling the complexity of the model and preventing overfitting. Additionally, the `min_samples_split` parameter, indicating the minimum number of samples required to split an internal node, was tested with values [2, 5, 10]. The `min_samples_leaf` parameter, which specifies the minimum number of samples required to be at a leaf node, was also varied with values [1, 2, 4] to ensure a balanced trade-off between bias and variance.

For the Gradient Boosting Regressor (GBR), a comprehensive set of hyperparameters was tested. The `n_estimators` parameter, representing the number of boosting stages, was evaluated with [50, 100, 200] stages. This helps in understanding the impact of the number of boosting iterations on the model's performance. The `learning_rate` parameter, which controls the contribution of each tree, was set to [0.01, 0.05, 0.1] to assess its effect on convergence and accuracy. The `max_depth` of the individual regression estimators was tested with values [3, 5, 7, 9] to balance model complexity and overfitting. The `min_samples_split` and `min_samples_leaf` parameters were tested with values [2, 5, 10], and [1, 2, 4], respectively, to fine-tune the model's sensitivity to data variation. The `subsample` parameter, defining the fraction of samples used for fitting the base learners, was set to [0.6, 0.8, 1.0] to study its effect on the model's robustness and variance.

The K-Nearest Neighbors Regressor (KNR) was evaluated using a range of hyperparameters to determine its optimal configuration. The `n_neighbors` parameter, which specifies the number of neighbors to consider, was varied from [1 to 15] to analyze its impact on model precision and sensitivity. The `algorithm` parameter, which dictates the algorithm used to compute the nearest neighbors, was tested with ['auto', 'ball_tree', 'kd_tree', 'brute'] to compare their computational efficiency and accuracy. The `leaf_size` parameter, affecting the leaf size passed to tree-based algorithms, was set to [10, 20, 30, 40, 50]. This helps in balancing the speed and accuracy of the model. The `p` parameter, which defines the power parameter for the Minkowski metric, was tested with values [1, 2] to understand its influence on distance calculation.

The Multi-Layer Perceptron Regressor (MLPR) was fine-tuned using various hyperparameters to enhance its performance. The `hidden_layer_sizes` parameter, specifying the number of neurons in the hidden layers, was set to [(50,), (100,), (50, 50), (100, 50), (100, 100)]. This allows the model to learn different levels of abstraction in the data. The `activation` function for the hidden layers was tested with ['identity', 'logistic', 'tanh', 'relu'] to evaluate their effect on non-linearity and learning capability. The `solver` parameter, which determines the algorithm for weight optimization, was tested with ['lbfgs', 'sgd', 'adam'] to compare their convergence speed and reliability. The `alpha` parameter, representing the L2 penalty term for regularization, was varied with values [0.0001, 0.001, 0.01] to prevent overfitting and improve generalization.

For the Elastic Net (EN) algorithm, hyperparameter tuning was performed to optimize its performance. The `alpha` parameter, which controls the regularization strength, was tested with values [0.1, 0.5, 1.0, 2.0, 5.0, 10.0]. This helps in balancing the trade-off between bias and variance. The `l1_ratio` parameter, defining the mix ratio between l1 and l2 penalties, was varied with values [0.1, 0.3, 0.5, 0.7, 0.9, 1.0] to understand its effect on sparsity and regularization. The `max_iter` parameter, indicating the maximum number of iterations for optimization, was set to [1000, 2000, 3000, 5000] to ensure sufficient convergence. The `tol` parameter, representing the tolerance for optimization, was tested with values [1e-4, 1e-3, 1e-2] to achieve the desired precision. The `selection` parameter, which determines if a random coefficient is updated every iteration, was tested with ['cyclic', 'random'] to compare their impact on optimization speed and performance.

The Huber Regressor (HR) was fine-tuned using a set of hyperparameters to enhance its robustness. The `epsilon` parameter, which determines the threshold for considering samples as outliers, was tested with values [1.0, 1.5, 2.0]. This helps in balancing the sensitivity to outliers and model robustness. The `alpha` parameter, representing the regularization strength, was varied with values [0.0001, 0.001, 0.01] to control overfitting. The `max_iter` parameter, indicating the maximum number of iterations for optimization, was set to [100, 200, 300] to ensure adequate convergence. The `tol` parameter, defining the tolerance for optimization, was tested with values [1e-4, 1e-3, 1e-2] to achieve the desired level of accuracy.