Social Science and Humanities Journal
Everant Journal

ISSN: 2456-2653
Volume 09 Issue 01 January 2025

https://sshjournal.com/
Impact Factor: 2024: 6.576
2023: 5.731

DOI: https://doi.org/10.18535/sshj.v9i01.1461

Assessing the performance of different variations of ensembled tree models in chlorophyll concentration prediction

Nelson[a,1] | Jimmy Tjen[b,2] | Genrawan Hoendarto[c,3]

[1,2,3]Deparment of Informatics, Universitas Widya Dharma Pontianak, Indonesia, 78117
[a]nelsonnelson41314@proton.me, [b]genrawan@widyadharma.ac.id, [c]jimmy.tjen@mathmods.eu

***Received* 09-12-2024**
***Revised* 10-12-2024**
***Accepted* 31-12-2024**
***Published* 03-01-2025**

Abstract:

Severe microalgae blooming is detrimental towards human life and the aquatic ecosystem in which it is blooming uncontrollably in. Chlorophyll concentration is a common parameter used to predict microalgal bloom. In this study, a variety of ensembled tree models which consists of random forest, gradient boosting frameworks, specifically XGBoost and LightGBM, and extra trees were implemented to predict amounts of chlorophyll concentration that can be found in running water. A comparison was also made between the models to find which performs the best in prediction and computational time. The comparison was conducted by comparing the NRMSE of each model and the average computing time. Each of the model's hyperparameters has been tuned with the help of random search, as a method for hyper-parameter optimization. The results were as such: random forest took 16.95 ms to compute and the result of the NRMSE was 0.75, XGBoost took 8.28 ms to compute and the result of the NRMSE was 0.71, LightGBM took 2.81 ms to compute and the result of the NRMSE was 0.63, and extra trees took 17.15 ms to compute and the result of the NRMSE was 0.72. The comparison showed that both of the gradient boosting based frameworks performed better compared to both random forest and extra trees. Specifically, LightGBM performed the best in terms of both predictive performance and computational time. The results of this study serves as a purpose to find a faster alternative with similar or better accuracy compared to random forest as a baseline in predicting chlorophyll concentration.

Keywords: Ensembled Tree, Chlorophyll Concentration, Performance analysis, Prediction

1. Introduction:

Microalgae are a natural part of aquatic ecosystems. Though microalgae have shown to provide a lot of benefit to humans and the environment, severe microalgae blooming has shown to be detrimental towards humans and the environment [1]. As such, constant monitoring of microalgae proliferation done reliably and in real-time is important to help mitigate or even prevent damages that might happen to the local environment/economy of which the blooming is occuring. According to a study done previously, one of the parameters that can be used to reliably predict algal bloom is by investigating the Chlorophyll-a (Chl-a) concentration in waterbodies [2]. Another study also supports this notion stating that Chl-a is still regarded as a typical primary proxy for microalgae bloom [3].

This study focuses on predicting the concentration of chlorophyll that may be found in rivers and estuaries. Due to the non-linearity relationship between Chl-a concentration and its environmental factors, models such as Artificial Neural Network (ANN) and Support Vector Machine (SVM) has been used to predict chlorophyll concentration [2]. As reported by Yongeun Park, it has been found that though the accuracy of training prediction were almost identical, SVM displayed a higher accuracy in the validation step compared to ANN. Another model, Random Forest (RF), has also been used to predict chlorophyll concentration [4]. In said study, RF was compared to Support Vector Regression (SVR), and it was found that Random Forest (RF) performed better.

Though the model RF works really well in predicting chlorophyll concentration prediction performance-wise, due to the nature of how RF works, RF may be less effective in predicting in real-time due to the sheer number of trees generated and how each data has to be computed for each individual decision trees. Having a faster model in predicting with similar levels of accuracy may help in detecting potential microalgae bloom earlier. Though, because RF has proven to be reliable, as such this study will try to explore other models related to ensembling Regression Trees in hopes of finding a model with faster prediction time and better accuracy compared to RF. The specific models that will be compared with RF will be Extremely Randomized Trees (Extra Trees) and gradient boosting based frameworks. Specifically, Extreme Gradient Boosting (XGBoost) and Light Gradient Boosting Machine (LightGBM).

In this study, a comparison of how well the models perform will be done through measuring the prediction performance and computing time needed of each of the models. The parameters that were used in this study are the following: orthophosphate, ammonia, nitrate, nitrite, water temperature, pH, dissolved oxygen, and the chlorophyll concentration itself. The water quality dataset has been obtained through the Department for Environment, Food & Rural Affairs' publicly available dataset.

2. Experimental Setup

2.1 Dataset and Variables

The water quality dataset obtained to conduct this study is a publicly available dataset through the Department for Environment, Food & Rural Affairs' website [5]. All of the data that were used in this study are mostly for monitoring purposes, and the type of water body that were used in this study are all made up of rivers (or running surface water) and estuaries. The water bodies that were used in this study come from different parts of England. Due to the lack of data in some individual months, rows with missing data were not taken into account. Another important thing to note is that the data that was accumulated contains natural outliers that were kept. The data used in this study amounts to 639 rows of data. The data used in this study are taken from completely random times between the range of 10th January 2022 up to 21st June 2024.

The features used in this study are all numerical features and contain no categorical feature. The features chosen in this study follow similarly to studies done in the past to predict chlorophyll concentration. The features that were used in this study are the following: orthophosphate, ammonia, nitrate, nitrite, water temperature, pH, and dissolved oxygen. The statistics of each of the water quality parameters are shown in Table 1.

Table 1. Statistics of water quality in the dataset

Parameter	Mean	Max	Min	Standard Deviation
Orthophosphate (mg/l)	0.146516	1.9	0.01	0.191571
Ammonia (mg/l)	0.088047	1.7	0.03	0.177738
Nitrate (mg/l)	8.657063	33.6	0.194	4.062073
Nitrite (mg/l)	0.047285	0.63	0.004	0.069705
Water Temperature (celcius)	10.613255	22.3	0.1	4.521982
pH	7.880689	8.97	6.69	0.263007
Dissolved Oxygen (mg/l)	9.850282	15.8	1.82	2.085699
Chlorophyll Concentration (μg/l)	5.548732	230	0.5	14.457147

2.2 Assessing Model Performance

The method of comparing the performance of the models used in this study are as per the following, the Root Mean Square Error (RMSE), and the Normalized Root Mean Square Error (NRMSE), which could be expressed as the following equations:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}},$$

$$NRMSE = \frac{RMSE}{\sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}{n}}}, \qquad (1)$$

where y is the observed value, $\hat{y}$ is the predicted value, $\bar{y}$ is the mean value of the observed value, and n is the number of samples. This variation of NRMSE was used instead of the commonly used, where the RMSE is divided by the mean of the observed data. This is to take into account how the mean is easily skewed by extreme values, because in this study extreme values were kept. Therefore instead of using the mean, in this study, the standard deviation of the observed data will be used instead. Lower values of NRMSE indicate less difference between the observed values and the predicted values.

The method of comparing the time it takes for each model to predict used in this study is by using the average amount of time elapsed for each model to predict, which could be expressed as the following equations:

$$Elapsed\ Time = Time_{end} - Time_{start},$$

$$Mean\ of\ Elapsed\ Time = \frac{Elapsed\ Time}{Amount\ of\ Iterations}. \qquad (2)$$

The amount of iterations used in this study is 10000 iterations. It is to be noted that, the number 10000 was chosen arbitrarily, though it was done with the aim of generating a large amount of samples. In this study, the time it takes to train the model will not be taken into account, because the focus of this study is to find a better model, compared to Random Forest as a baseline, when trying to predict in real-time.

3. Methodology

3.1 Classification and Regression Tree

Classification and Regression Tree (CART) is a machine learning model which is made up of classifiers acting as the partition [6]. CART consists of the root node, internal node, and leaf node. Both the root node and internal node acts as the classifier, while the leaf node is the outcome. The depth of the tree is the length from the root node to the leaf node with the largest amount of depth.

3.2 Random Forest

The Random Forest model is a machine learning model that combines a collection of unpruned trees to perform classification and regression. In a random forest model, each individual CART in a random forest relies on values of a random vector sampled independently from the training set, usually through bagging, to perform classifications and predictions. Leaf node splits are done by choosing the best split over randomly chosen subset of features. When used in regression problems, the results from each regression tree is averaged to reach the final prediction. Random Forest also has resistance towards overfitting due to the law of large numbers [7].

3.3 Gradient Boosting framework

Gradient Boosting is a machine learning algorithm that constructs additive approximations iteratively, which is expressed by the following equation:

$$F_n(x) = F_{n-1}(x) + \rho_n h_n(x). \qquad (3)$$

In this study, there are two libraries that were used that implements the gradient boosting framework, namely XGBoost and LightGBM. XGBoost is a framework which implements gradient boosting. Generally, XGBoost uses a greedy algorithm to find splits. XGBoost has the ability to parallelize split finding by storing data in blocks, which are in-memory units. By doing this, it allows different blocks to be distributed, thus allowing parallelization, which in turn makes learning faster. XGBoost implements regularization similar to Regularized Greedy Forest which is expressed by the following equation:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k),$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|, \qquad (4)$$

where $l(\hat{y}_i, y_i)$ represents the loss function that measures the difference between the prediction and the target, and $\Omega(f)$ which represents the parameter for regularization [8].

LightGBM is another framework that utilizes the gradient boosting framework. The difference between LightGBM and the alternatives proposed in this study lies in the split finding algorithm, tree growth strategy, and data sampling. LightGBM uses a histogram-based algorithm to find splits. The histogram-based algorithm buckets continuous feature values into discrete bins, and then the algorithm uses those bins to construct feature histograms during training [9]. The histogram-based algorithm later finds the best split points based on the feature histograms. LightGBM uses a leaf-wise tree growth strategy. Essentially, it chooses the leaf with max delta loss to grow. Furthermore, LightGBM introduces a novel way of data sampling which is known as GOSS (Gradient-based One-side Sampling). Essentially the training set retains all instances with large gradients and performs random sampling on instances with small gradients. To compensate for the change in data distribution, when computing the information again, GOSS introduces a constant multiplier, which is expressed by Eq. (5), for the data instances with small gradients.

$$\frac{1-a}{b}. \qquad (5)$$

where a represents the large gradient instances and b represents the small gradient instances.

3.4 Extremely Randomized Trees

The Extra Trees model is a machine learning model that works similarly to Random Forest, where the model combines a collection of decision trees to perform classification and regression. However, the difference between random forest and extra trees lies in the process of how the trees are made. In random forest, CARTs are created through bootstrapping the training set and node splitting is done greedily over a random selection of subset of features, whereas in extra trees, CARTs are created using the whole training set and node splits are done in an entirely random manner. Similar to Random Forest, in regression problems, the results from each regression tree is averaged to reach the final prediction [10].

3.5 Hyper-parameter optimization using Random Search

Properly configuring the hyper-parameters of each model will influence how the model performs. Hyper-parameters are essentially parameters which control the learning process of the models. Specifically in tree-based models, these are the components which need to be optimized: the maximum tree depth, function to measure the quality of splits, split selection method, number of considered features, minimum number of data points to split a decision node, maximum number of leaf nodes, and the minimum number of the total weight. Ensembled tree models have the same hyper-parameters as decision tree models, with additional hyper-parameters to consider such as: the number of decision trees, the minimum loss reduction for a split, L_1 and L_2 regularization on leaf nodes, and learning rate [11].

Though, strictly speaking, manually tuning each parameter for each model is highly inefficient and time-consuming. Therefore in this study, hyper-parameter optimization will be used to help alleviate most of the effort needed to tune the hyper-parameters. Hyper-parameter optimization are algorithms developed to identify good value for hyper-parameters [12]. In this study, the algorithm that has been chosen to perform the optimization is Random Search. Essentially, Random Search randomly selects a pre-defined number of samples between the upper and lower bound in the search space as candidate hyper-parameter values, and then the algorithm will train using the random combinations until the defined number of iterations [11].

3.6 Implementation

In this segment, the method of implementing the study will be discussed. The dataset was split into an 80-to-20 train-to-test ratio. After splitting the dataset, hyper-parameter optimization was performed before the process of training using the model itself. After finding the appropriate hyper-parameters, the training process begins, which is then followed-up with the testing process, where the time it takes for each model to predict will also be recorded. After that, the results of the testing process will be compared with the observed data to assess the performance of the models. The process could be expressed through the following:

Algorithm 1: Implementation

```
Input: X,y,paramgrid
Output: nrmse, mean_time
Process:
$X_{train}$,$y_{train}$,$X_{test}$,$y_{test}$=train_test_split(X,y,test_size=0.2)
param=randomized_search(paramgrid)
mdl=model(param)
for i=0:10000
start=time()
mdl.fit($X_{train}$,$y_{train}$)
total_time=total_time+time()-start
end
mean_time=total_time/10000
print(mean_time)
$y_{pred}$=mdl.predict($X_{test}$)
rmse=root_mean_squared_error($y_{test}$,$y_{predict}$)
nrmse=rmse/standard_deviation(y)
print(nrmse)
```

4. Results and discussion:

Table 2 shows the hyper-parameters used in this study. It is important to note that there are some hyper-parameters that are not shared among the models, or there are cases where the default setting is better. In such cases, the hyper-parameters for the model will instead be noted as '-' in the table. Furthermore, hyper-parameters that are similar but has different names across the different models will be substituted using a description of what the hyperparameter does. Another important detail to point out is that the parameters random_state and n_jobs do not contribute to the model's precision. The parameter random_state is only there for reproducibility, and n_jobs is used to utilize all of the available cpu cores, or in other words parallelization.

Table 2. The hyper-parameters of each models used in this study

Hyper-parameter	Random Forest	XGBoost	LightGBM	Extra Trees
random_state	7	7	7	7
n_jobs	-1	-1	-1	-1
Number of trees	45	40	70	45
Minimum number of samples required to split an internal node	2	-	-	10
Maximum number of features when splitting	2	-	-	2
Maximum depth of tree	10	12	9	10
Minimum loss reduction required to make a further partition on a leaf node	-	3	15	-
L1 Regularization	-	-	1	-
L2 Regularization	-	27.005	0.55	-
Booster	-	'dart'	'dart'	-
Step size shrinkage	-	0.15	0.5	-
Maximum number of bins to bucket features	-	-	30	-

As mentioned before, the bulk of the work done to tune the hyper-parameters of each models were mostly done with the help of hyper-parameter optimization, though some manual tuning was also done to improve upon the results of the hyper-parameter optimization even further. Figure 1 shows the results of the models compared to the observed value, while Table 3 presents the RMSE and NRMSE of the predictive models used in this study.

Table 3. The performance results of the models and the time needed for each of the models to make the predictions

Metric	Random Forest	XGBoost	LightGBM	Extra Trees
Time (ms)	16.95	8.28	2.81	17.15
RMSE	10.89	10.21	9.09	10.39
NRMSE	0.75	0.71	0.63	0.72

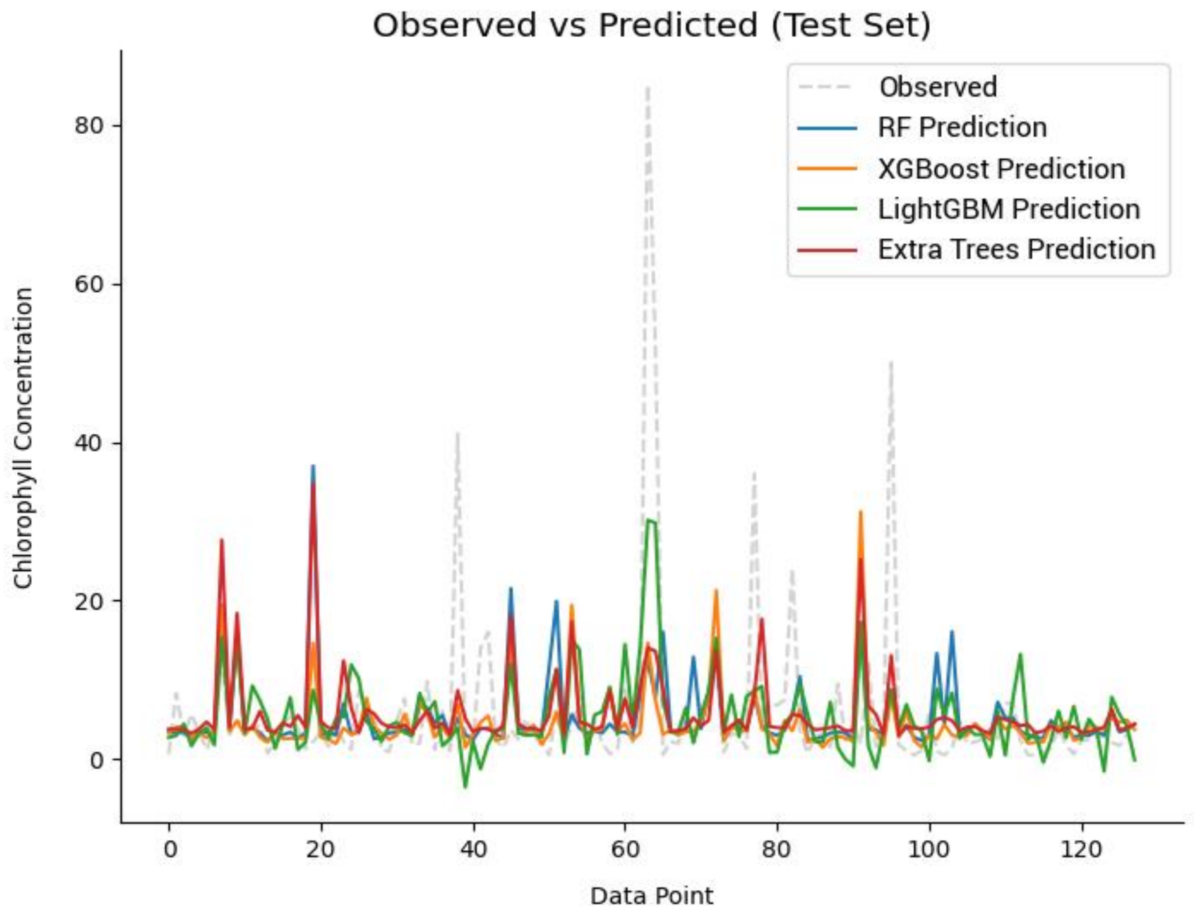

Figure 1. Results of the Predictive Model and the Observed Value

Utilizing the metrics that has been proposed before, it is to be noted that the predicting performance of the models are mostly similar. LightGBM has been found to perform the best out of the proposed models in terms of both predictive performance. Both random forest and extra trees did worse than two of the gradient-boosting models in terms of prediction performance.

The performance of both gradient boosting libraries compared to random forest and extra trees in terms of prediction performance can be attributed to how the models deal with prediction errors. In both random forest and extra trees, both prevent prediction errors by controlling the tree's complexity and how it selects its features, in other words by controlling the amount of depth a tree, the number of nodes of each tree, and the amount of features sampled when splitting. On the other hand, while both gradient boosting-based models is also able to control the tree's complexity as well, both of the gradient boosting-based models have regularization, which affects the loss function of the model.

Subsequently, LightGBM and XGBoost also performed better in computational time. The complexity of the models affect the computational time to predict. XGBoost and LightGBM both have less leaf nodes in average compared to both random forest and extra trees. Though as shown in Table 4, LightGBM has significantly less leaf nodes and tree depth compared to XGBoost. Besides that, LightGBM also grows its trees leaf-wise compared to the other models, which grows the trees depth-wise. In other words, this alters the final structure of the trees of LightGBM. Though, it is to be noted that random forest did better than extra trees despite having more leaf nodes. Referring to Table 2, in this study the minimum number of samples required to split an internal node in extra trees is higher than that of random forest, thus altering the final structure of the trees of extra trees.

Table 4. The average amount of leaf nodes and tree depth of each models

Attribute	Random Forest	XGBoost	LightGBM	Extra Trees
Leaf nodes	78.73	42.33	11.46	50.38
Tree depth	10	11.38	7.91	10

5. Conclusion:

The main focus of this study is to compare several well-known ensembled tree models to find a better performing model compared to Random Forest as the baseline in the context of predicting cholorophyll concentration content that can be found in rivers and estuaries. In this study, the models were compared in terms of prediction performance and computational time. NRMSE was used as a metric to compare the prediction performance of the models, and the average computational time over an arbitrary amount of iterations was used to compare the prediction speed of the models. In this study, the results of all of the models were mostly similar. Though, the model that was found to perform the best in terms of prediction performance and computational time is LightGBM.

References:

[1] G. M. Hallegraeff, D. M. Anderson, A. D. Cembella and H. O. Enevoldsen, Manual on harmful marine microalgae, Unesco, 2004.

[2] Y. Park, K. H. Cho, J. Park, S. M. Cha and J. H. Kim, "Development of early-warning protocol for predicting chlorophyll-a concentration using machine learning models in freshwater and estuarine reservoirs, Korea," Science of the Total Environment, vol. 502, pp. 31-41, 2015.

[3] Q. V. Ly, X. C. Nguyen, N. C. Le, T.-D. Truong, T.-H. T. Hoang, T. J. Park, T. Maqbool, J. Pyo, K. H. Cho, K.-S. Lee and others, "Application of Machine Learning for eutrophication analysis and algal bloom prediction in an urban river: A 10-year study of the Han River, South Korea," Science of The Total Environment, vol. 797, p. 149040, 2021.

[4] X. Li, J. Sha and Z.-L. Wang, "Application of feature selection and regression models for chlorophyll-a prediction in a shallow lake," Environmental Science and Pollution Research, vol. 25, pp. 19488-19498, 2018.

[5] "Department for Environment Food & Rural Affairs," [Online]. Available: https://environment.data.gov.uk/water-quality/view/download/new. [Accessed 29 July 2024].

[6] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, Classification and Regression Trees, Taylor & Francis, 1984

[7] L. Breiman, "Random forests," Machine learning, vol. 45, pp. 5-32, 2001.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," Advances in neural information processing systems, vol. 30, 2017.

[10] P. Geurts, D. Ernst and L. Wehenkel, "Extremely randomized trees," Machine learning, vol. 63, pp. 3-42, 2006.

[11] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," Neurocomputing, vol. 415, pp. 295-316, 2020.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," Journal of machine learning research, vol. 13, no. 2, pp. 281-305, 2012.