Song Popularity Prediction with nonlinear hyperparameter optimization

Qiuyi Zhang, Zishuai Liu
Mckelvey School of Engineering, Washington University in St. Louis
May 5, 2023

Abstract

It is not only the choice of algorithm that affects the performance of machine learning, but also the combination of hyperparameters. The right combination of hyperparameters can greatly improve the performance of the model. The traditional hyperparameter optimization methods are grid search and random search, but both of them have their own limitations. A nonlinear hyperparameter optimization method, Bayesian optimization algorithm, has been proposed to solve the runtime problem and model robustness. Our experiments are based on realistic data--Spotify song database, and we compare the advantages and disadvantages of three hyperparameter optimization methods by performing hyperparameter optimization on different models separately, and finally get the model and hyperparameter combination with the best prediction performance. The obtained model will be able to solve the song popularity prediction problem.

Keywords: Non-linear Optimization, Hyperparameter Tuning, Bayesian Optimization, LightGBM, Random Search, Grid Search, AdaBoost

1. Introduction

For digital music platforms such as Spotify, predicting user preferences and song popularity would be helpful in building a recommendation system.

However, accurately predicting the song popularity will be a complicated problem. Predicting the popularity of a song by its audio mode, acousticness, danceability, energy and other features requires building accurate machine learning models. In particular, hyperparameter optimization is crucial to improve the model performance.

In this paper, we choose to use several machine learning models, including LightGBM, kNN, SVM, random forest, and AdaBoost, to predict song popularity and compare the performance of grid search, random search, and Bayesian optimization performing hyperparameter optimization on different models to finally obtain a model with the best prediction performance.

We will perform data processing and feature engineering on the song feature dataset and perform hyperparameter optimization and model evaluation. Our objective is to compare the performance of different hyperparameter optimization techniques on different machine learning models and finally determine the best hyperparameters combination for model performance. By building predictive models with good accuracy, we can improve the recommendation accuracy of digital music platforms and enhance user experience. Through this experiment, we obtained the model with the best overall performance and efficiency, i.e., the Bayesian optimized LightGBM model.

Section 2 reviews the relevant theory and practice, Section 3 briefly explains the machine learning models and hyperparameter optimization techniques, Section 4 gives the experimental results, Section 5 compares the experimental results and gives suggestions for the best model, Section 6 talks about future work, and the last section is the contribution of the team members.

2. Related theory and practice

In this section, we review the development of commonly used machine learning models and hyperparameter optimization algorithms in recent years.

Support vector machines (SVM), k-nearest neighbors (kNN), LightGBM, random forest, and AdaBoost are commonly used machine learning algorithms. SVM performs the classification on data by finding decision boundaries, and the hyperparameters can be adjusted to optimize the performance effectively (Hearst et al., 1998); KNN uses geometric distance to classify neighboring sample points and is one of the most commonly used classification algorithms (Peterson, 2009). LightGBM is an improvement of GBDT, and the model has a significant increase in training speed with little impact on accuracy (Ke et al., 2017). Random Forest and AdaBoost are both ensembled learning techniques that improve prediction accuracy by combining multiple decision trees (Wyner et al., 2017).

In order to create an effective machine learning model, it is not enough to choose the right model algorithm, but hyperparameter optimization must also be performed. Different hyperparameters may have a significant impact on the performance of the model; therefore, it is important to find the best combination of hyperparameters and models when solving machine learning problems. Grid search, Random Search (Bergstra & Bengio, 2012), and Bayesian Optimization (Eggensperger et al., 2013) are all common hyperparameter optimization techniques.

The grid search exhaustively evaluates all hyperparameter combinations in a fixed domain. Random search randomly selects hyperparameter combinations in the search space and compares the performance of different models. Bayesian optimization determines the next hyperparameter value based on the results of previously tested hyperparameter values, reducing many unnecessary evaluations and enabling the detection of the best hyperparameter combination in fewer iterations.

Several studies have compared the performance of different hyperparameter optimization methods on various machine learning models. For example, Bergstra and Bengio (2012) performed hyperparameters optimization on the neural network using grid search and random search and found that random search had better performance. They argued that random search has more uniform sampling and therefore can have a larger search space. In addition, grid search wastes time on unimportant hyperparameters dimensions, while random search is more efficient.

Yang and Shami (2020) compare various algorithms and practices for machine learning hyperparameter optimization and propose that Bayesian optimization can be used to optimize conditional hyperparameters. In addition, Elgeldawi et al. (2021) compared the performance of hyperparameter optimization algorithms such as random search, grid search and Bayesian optimization on models including SVM, random forest, etc. and found that SVM obtained the highest accuracy on the selected dataset after hyperparameter tuning using Bayesian optimization.

3. Technical details

3.1 Models

A. LightGBM

Light Gradient Boosting Machine (LightGBM) is a gradient boosting framework, which train decision trees leaf-wise to predict the gradient of the loss function (Brownlee, 2020). During this process, LightGBM chooses the leaf by determine if it could cause the largest decrease in loss (Manish, 2020). It included two state-of-the-art techniques: GOSS and EFB, which made the algorithm effective and well-suited for handling large datasets faster.

B. KNN

The k-nearest neighbors algorithm (KNN) is a non-parametric supervised learning algorithm (Fix & Hodges, 1951). The basic idea behind it is to classify an unlabeled test point by assigning the most frequent label among the k training samples which is the nearest. The output is the average of values of the nearest neighbors.

C. SVM

Support Vector Machines (SVMs) are a kind of supervised learning algorithm which can be used to solve not only classification problems but also regression tasks (Joachims, 1998). The main idea is to find supporting vectors to clarify the hyperplane that could classify the data into several categories maximally. In this paper, we use SVM with radial basis function (RBF) kernel.

The idea behind RBF kernel function is to calculate the similarity score between points in a high-dimensional space. Similarity score can be defined from the distance between two points. The function is shown below (Jean-Philippe et al., 2004):

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

||x – x'|| is defined as the squared Euclidean distance between the two feature vectors x and x'. σ is a free parameter.

D. Random Forest

Random forest is to build a large number of decision trees, each of which is trained on a different subset of the train dataset and a random subset of the features (Wyner et al., 2017). The basic idea is to randomly select subsets of features and train in the decision trees to combine such weak learners to obtain a stronger learner.

For prediction problems, the algorithm aggregates the prediction of all the individual decision trees to make final prediction results. It could be the mean or average prediction of the decision trees.

E. AdaBoost

Adaptive Boosting (AdaBoost) is a classification meta-algorithm that combines base weak learners and train them into a strong classifier. These weak classifiers, known as decision stumps, are usually simple and fast in the training process. The output from them would be combined to form a weighted result that represents the final output. Basic idea is to give a higher weight of samples which were misclassified in next iteration sequentially (Wyner et al., 2017). The boosted classifier could be presented as follow:

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

Here, $f_t(x)$is the weak classifier. x is the input. $f_t(x)$ gets the input x and returns the result which indicates the category of the input.

3.2 Optimization

A. Bayesian Algorithm

Bayesian Optimization is a nonlinear optimization method that uses Bayes Theorem to direct the search in order to find the minimum or maximum of an objective function (Garnett, 2023). It is an approach that is most useful for objective functions that are complex, noisy, and/or expensive to evaluate.

The idea behind Bayesian optimization is to model the objective function (e.g., accuracy, loss) as a probabilistic surrogate function, using a probabilistic model called a Gaussian process (Dewancker, 2016). The GP model captures the relationship between the hyperparameters and the objective function and can be used to predict the objective function value for new sets of hyperparameters.

B. Grid Search Algorithm

The main idea behind grid search is to find the combination of hyperparameters over a predefined grid to achieve the best model performance. It trains the model for each combination of hyperparameter in the specified hyperparameter grid and selects the best on the train set. The main problem could be curse of dimensionality.

C. Random Search Algorithm

The key point of random search is to sample hyperparameter values at random from a specified distribution without wasting too much time on those that are unlikely to help improve the model performance. It could be more efficient than grid search in searching a broad space of hyperparameters without being constrained by a specified grid. However, to get the best combination option, it would require more iterations.

4. Experimental

4.1 Data processing

In this work, prediction models with optimizers are trained on Spotify popular song database. Spotify is one of the largest music providers, with more than 489 million monthly active users. It provides over 100 million songs and 5 million podcasts from different singers and musical companies. It's critical for such a competitive and promising music industry to predict the song popularity.

There are 14 features in our dataset. First, we detect and discard outliers. Then we label the target value_song popularity as 0 or 1(score>66.5 as 1), so this becomes a classification problem. We perform one-hot encoding on categoric features including "audio_mode","key" and "time_signature", and perform standardization on data. After data processing and feature engineering, we obtain 30 features.

4.2 Experiment

Firstly, we define the search space of hyperparameters and initialize models. Then, construct the optimizer using different hyperparameter optimization methods and fit the model on the training set separately. We train the model with the best combination of hyperparameters and then evaluate the model's performance. The best combination of hyperparameters obtained by different hyperparameter optimization methods with different models is as follows:

A. LightGBM

Table I. Comparative analysis of the best hyperparameters of LightGBM.

	Bayes	**Grid**	**Random**
learning_rate	0.0540	0.1	0.6136
max_depth	22	20	12
num_leaves	49	50	16
n_estimators	1000	1000	975
min_child_samples	11	1	31

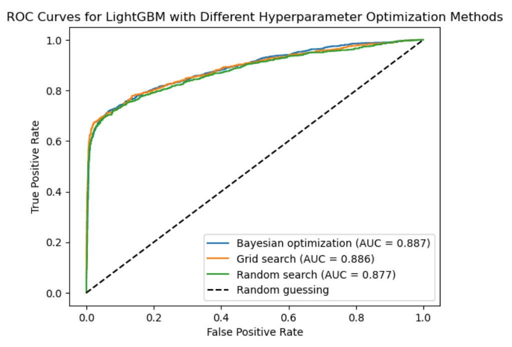

Fig. I. ROC Curves for LightGBM.

The best hyperparameters of LightGBM obtained by different optimization methods are given in Table I. We can see that Bayesian Optimization found the lowest learning rate of 0.054 and highest max depth of 22 among the three methods. The number of leaves and min child samples are in between the values found by grid search and random search.

The performance of LightGBM with different best hyperparameter combinations on the test set is given in Table II. The results showed that Bayesian optimization achieved the highest precision and AUC score. Grid search achieved the highest accuracy of 0.8801 and Bayesian optimization had an accuracy of 0.8779 . Random search had the lowest accuracy of 0.8609. Considering both model performance and time, Bayesian optimization was the most efficient method, with hyperparameter optimization taking only 111.2 seconds, while grid search had the highest computational cost, taking 1364.76 seconds. Random search was the fastest method, but the model accuracy and AUC score did not perform well.

Overall, we can consider both Bayesian optimization and grid search as effective methods for LightGBM hyperparameter optimization, while grid search has the best accuracy but can be time consuming, and Bayesian optimization is the best choice considering time and performance. Random search is the most time-efficient and can be chosen when too high accuracy is not needed but time is limited.

We also perform cross-validation using the model with different best hyperparameter combinations and draw the ROC curve in Figure I. We can see that the hyperparameters obtained from Bayesian optimization method performed the best in terms of the ROC AUC score for LightGBM.

B. KNN

Table III. Comparative analysis of the best hyperparameters of KNN.

	Bayes	**Grid**	**Random**
n_neighbors	50	50	50
p	1	1	1
weights	distance	distance	distance
algorithm	kd_tree	ball_tree	ball_tree

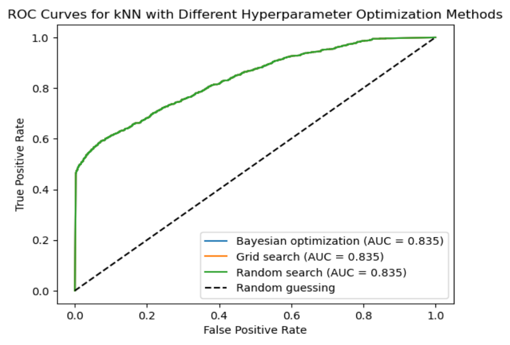

Fig. II. ROC Curves for KNN.

From the best hyperparameters of KNN given in table II, we can see that the three hyperparameter optimization method find the same hyperparameters combination, so the model performances given in table II and Figure II are also the same. However, Bayesian optimization and random search are much more time-efficient than grid search.

The results show that for simple models and search spaces, it is sufficient to obtain good performance using Bayesian optimization and random search.

C. SVM

Table IV. Comparative analysis of the best hyperparameters of SVM.

	Bayes	**Grid**	**Random**
C	5.1878	10	5.2190
gamma	1.0	1	0.8726

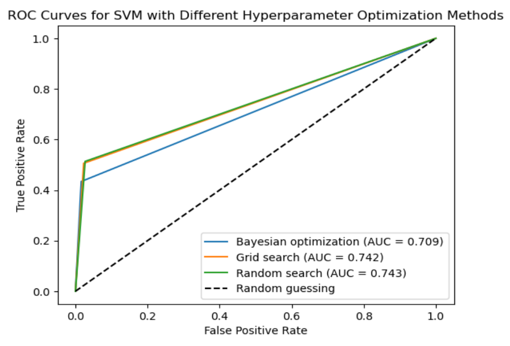

Fig. III. ROC Curves for SVM.

Table III shows the best combination of hyperparameters of SVM (RBF kernel). We can see that Bayesian optimization gets a similar C value with random search and a similar gamma value with grid search. Even the precision of SVM with Bayesian optimization is 0.9180, larger than which of grid search and random search, SVM with Bayesian optimization doesn't perform well in our dataset. From Figure III, we can see Bayesian optimization get the worst ROC AUC score among three optimization techniques. Thus, SVM with Bayesian optimization is not effective and efficient in this situation. SVM models don't perform well in recall, F1score with any of the three optimization methods. We can conclude that SVM may not be suitable for this problem.

D. Random Forest

Table V. Comparative analysis of the best hyperparameters of Random Forest.

	Bayes	Grid	Random
max_depth	23	None	20
max_features	0.506073130985518	0.1	1.0
min_samples_leaf	3	1	3
min_samples_split	5	2	7
n_estimators	282	100	74

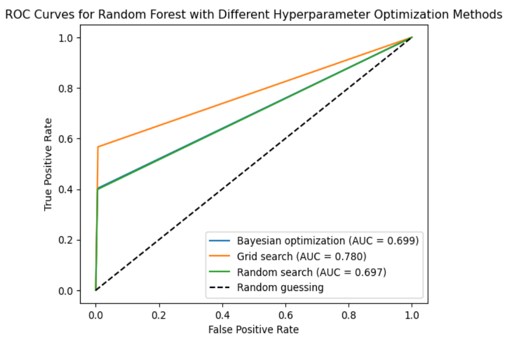

Fig. IV. ROC Curves for Random Forest.

From Table V, we can see the best combination of hyperparameters of Random Forest obtained by optimizations above. The Bayesian optimizer got the max depth of 23, which is none in grid search and 20 in random search. The values of max features and min samples split are between grid search and random search. Besides, Bayesian optimization found the highest value of n estimators of 282.

Table II shows the performance of Random Forest with different best hyperparameters on the testing set. The results showed that the model optimized by grid search had the best accuracy of 0.8684. However, the improvement of accuracy, compared with which from models optimized by Bayesian optimization or random search, was elevated to a small extent. We should focus on the tradeoff between accuracy and time. Random Forest with grid search obtained the longest processing time of 971.06 seconds, compared with 109.22 seconds from Bayesian optimization and 61.73 seconds from random search. Combining accuracy and time from different optimizers, we concluded that Random Forest with Bayesian optimization had the best performance on the testing dataset. However, the precision, recall, F1 score, and ROC AUC score performed well in grid search. From Figure IV, we can see ROC AUC score for Random Forest with these optimizations. Grid search performed best in ROC AUC score, and that from Bayesian optimization had a limited difference with it.

In conclusion, Bayesian optimization and grid search perform best in optimization of hyperparameters of Random Forest in our dataset. The tradeoff between them could make us select effective one to optimize Random Forest to meet different requirements in real situations.

E. AdaBoost:

Table VI. Comparative analysis of the best hyperparameters of AdaBoost.

	Bayes	Grid	Random
learning_rate	0.1034	0.1	0.1

n_estimators	100	100	75

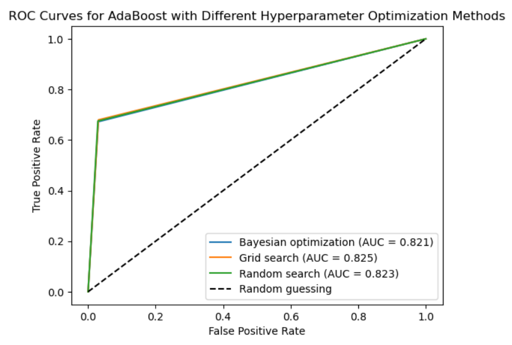

Fig. V. ROC Curves for AdaBoost.

Table VI shows the best combination of hyperparameters of AdaBoost with three optimizations. We can see from the table that the values of learning rate are almost similar. And n estimators of Bayes and grid search are both 100.

From Figure V, we find that the AUC ROC values are quite similar in different optimizers. Table II shows that evaluation metrics among three models with Bayesian optimization, grid search, random search are closer to each other. Thus, different optimizations on AdaBoost don't make any differences on the performance of models. However, grid search is not time efficient compared with the other two optimizations.

Table II. Comparative analysis using various performance metrics.

	Optimization Method	**Accuracy**	**Precision**	**Recall**	**F1 score**	**ROC AUC score**	**Time(sec)**
LGBM	Bayesian	0.8779	0.8734	0.6814	0.7656	0.8203	111.20
	Grid	0.8801	0.8728	0.6907	0.7711	0.8245	1364.76
	Random	0.8609	0.7940	0.7082	0.7487	0.8161	49.47
KNN	Bayesian	0.8285	0.7738	0.5845	0.6660	0.7569	575.75
	Grid	0.8285	0.7738	0.5845	0.6660	0.7569	1952.92
	Random	0.8285	0.7738	0.5845	0.6660	0.7569	299.78
SVM(RBF)	Bayesian	0.8231	0.9180	0.4340	0.5893	0.7090	2254.24
	Grid	0.8393	0.9013	0.5060	0.6481	0.7415	2213.02
	Random	0.8387	0.8882	0.5134	0. 6507	0.7433	610.58
RandomForest	Bayesian	0.8214	0.9689	0.4026	0.5688	0.6986	109.22
	Grid	0.8684	0.9715	0.5669	0. 7160	0.7800	971.06
	Random	0.8201	0.9664	0.3989	0. 5647	0.6966	61.73
AdaBoost	Bayesian	0.8828	0.9021	0.6722	0.7704	0.8210	1046.91
	Grid	0.8847	0.9020	0.6796	0.7751	0.8245	1390.04
	Random	0.8844	0.9058	0.6750	0.7735	0.8230	121.62

5. Discussions, observations, and comparisons

In this study, we performed hyperparameter optimization on five commonly used machine learning algorithms (i.e., SVM, k-NN, LightGBM, Random Forest, and AdaBoost) using Bayesian optimization, grid search, and random search to predict the popularity of songs on

Spotify. The results show that all three optimization methods significantly improve the performance of the algorithms, with Bayesian optimization performing best on most models in aggregate.

The best performing algorithm overall was LightGBM. Using Bayesian optimization, the model obtained the highest precision on test data and the highest AUC score through cross-validation, indicating the best robustness of the optimized model. Also, the average time for hyperparameter optimization and model training is almost the lowest among several models. Random search is the most time-efficient method of LightGBM, Bayesian optimization is close to it, and grid search is the most time-consuming method.

AdaBoost performs second best overall, with a maximum accuracy of 0.8847 and an AUC score of 0.8245, achieved using Grid search. For Adaboost, random search is also the most time efficient. However, collectively, the three optimization methods perform very similar and obtain the similar hyperparameters combinations on AdaBoost, probably because the search space is set too small, and further comparisons can be made in the future by increasing the search space.

Random forest has the highest accuracy of 0.8684 and ROC AUC score of 0.7800, implemented using grid search, but this method consumes a lot of time for hyperparameter optimization and is not cost-effective enough. Bayesian optimization and random search are both relatively time efficient for optimizing random forests, while Bayesian optimization has slightly better model performance.

K-NN has the poor performance with the ROC AUC score of 0.7569, while the best parameters and model performance of KNN obtained by the three optimization algorithms are the same. Random search is the most time efficient method for k-NN, while grid search is the most time consuming.

The obtained SVM (RBF) models using all three optimization methods have the lowest overall performance, especially the recall and AUC, indicating that SVM (RBF) may not be suitable for our dataset and cases. We can try SVM with other kernels or different models later.

In summary, we can conclude that hyperparameter optimization can significantly improve the performance of machine learning algorithms in the task of predicting the popularity of songs on Spotify. Bayesian optimization is relatively better than the other two methods, indicating that it is a powerful nonlinear optimization technique for hyperparameter tuning in machine learning. Regarding the model selection, LightGBM is the most efficient algorithm for our dataset with relatively good overall performance and the most time efficient. Therefore, LightGBM can be chosen to build the song popularity prediction model and optimized using Bayesian optimization.

6. Future works

Bayesian optimization is most useful and effective for objective functions that are complex, noisy and expensive to evaluate. As we conclude from the results from models above, the improvement of models on the dataset is not that significant compared with grid search and random search. In the later work, we will concentrate on extending the search space and improving the prediction model. For example, for the search space in SVM, we could extend the range of ‘C’ and ‘gamma’ hyperparameters to get a larger space for optimizers. Besides, we would also plan to try more state-of-the-art prediction models or construct neural networks to train on our dataset.

Besides, we are also interested in exploring more hyperparameter optimizations in this situation. For example, Nayak et al. (2021) has proposed an advanced ensemble LightGBM on hand gesture recognition with improved memetic firefly algorithm. As a swarm-based algorithm, firefly algorithm (FA) is a more effective technique in dealing with multimodal constraints. We find it inspiring and plan to apply it on prediction of song popularity. In the future works, we would construct the ensemble LightGBM based on Spotify dataset and optimize it with advanced firefly algorithm. We’re also enlightened by the heuristic method of building the perturbation operator in hyperparameter tuning in FA introduced in the paper.

7. Contributions of team members

Experiment:

Qiuyi Zhang: Train LightGBM, KNN with Bayesian optimization, grid search and random search. Compare and analyze the results and make selections.

Zishuai Liu: Train SVM (RBF kernel), Random Forest, AdaBoost with Bayesian optimization, grid search and random search. Compare and analyze the results.

Paper:

Qiuyi Zhang: Abstract, Introduction, Related Theory, Experiment Result Analysis and Discussion.

Zishuai Liu: Technique details, Experiment Result Analysis and Future works.

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281-305.

Brownlee, J. (2020, March 31). Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost.

Dewancker, I., McCourt, M., & Clark, S. (2016). Bayesian optimization for machine learning: A practical guidebook. arXiv preprint arXiv:1612.04858.

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., & Leyton-Brown, K. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In BayesOpt Work (pp. 1-5).

Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. Informatics, 8(4), 79. https://doi.org/10.3390/informatics8040079

Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas. Archived (PDF) from the original on September 26, 2020.

Garnett, R. (2023). Bayesian Optimization. Cambridge University Press. ISBN 978-1-108-42578-0.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications, 13(4), 18-28.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

Manish, Mehta; Rakesh, Agrawal; Jorma, Rissanen (Nov 24, 2020). "SLIQ: A fast scalable classifier for data mining". International Conference on Extending Database Technology: 18–32. CiteSeerX 10.1.1.89.7734.

Nayak, J., Naik, B., Dash, P. B., Souri, A., & Shanmuganathan, V. (2021). Hyper-parameter tuned light gradient boosting machine using memetic firefly algorithm for hand gesture recognition. Applied Soft Computing, 107478. doi:10.1016/j.asoc.2021.107478

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830.

Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1), 148-175.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. arXiv preprint arXiv:1206.2944.

Vert, J. P., Tsuda, K., & Schölkopf, B. (2004). A primer on kernel methods. Kernel Methods in Computational Biology.

Wyner, A. J., Olson, M., Bleich, J., & Mease, D. (2017). Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers. Journal of Machine Learning Research, 18(48), 1-33.

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316. https://doi.org/10.1016/j.neucom.2020.07.061