African Journal of
Management Engineering
and Technology

Afr. J. Manag. Engin. Technol., 2023, Vol. 1, N° 2, 149-160

An Approach to Optimizing Machine Learning Models for the Diagnosis of COVID-19 via Hyperparameter Optimization

Hamida S. [1, 2]*, El-Gannour O. [3, 4], Cherradi B. [3, 5]

[1] *2IACS Laboratory, ENSET of Mohammedia, Hassan II University of Casablanca, Mohammedia, Morocco.*
[2] *GENIUS Laboratory, SupMTI of Rabat Rabat, Morocco.*
[3] *EEIS Laboratory, ENSET of Mohammedia, Hassan II University of Casablanca Mohammedia, Morocco.*
[4] *LIASSE Laboratory, ENSA of Fez, Sidi Mohamed Ben Abdellah University, Fez, Morocco.*
[5] *CRMEF Casablanca-Settat, provincial section of El Jadida, 24000, El Jadida, Morocco.*
**Corresponding author, Email address:* hamida@enset-media.ac.ma

Received *11 Nov 2023,*
Revised *21 Dec 2023,*
Accepted *23 Dec 2023*
Citation*: Hamida S., El-Gannour O., Cherradi B. (2023) An Approach to Optimizing Machine Learning Models for the Diagnosis of COVID-19 via Hyperparameter Optimization, Afr. J. Manag. Engg. Technol., 1(2), 149-160*

Abstract: The process of picking appropriate hyper-parameters for classification or prediction algorithms is a tough endeavor in the field of modeling. This selection is essential for the capacity for generalization and the performance of classifiers. Over the course of two tests, this article examines and evaluates the performance of five different Machine Learning (ML) algorithms: Support Vector Machine (SVM), AdaBoost, RandomForest, XGBoost, and DecisionTree. When it comes to training and testing, the first experiment makes use of the default settings, while the second experiment makes use of the GridSearch function to locate the most effective configurations. The tests make use of a dataset that was gathered from Albert Einstein Hospital in Sao Paulo, Brazil, and the dataset contains anonymous individuals that either have or do not have COVID-19. Usage of evaluation metrics includes things like accuracy, precision, recall, area under the curve (AUC), and F1-score. According to the findings, improving hyper-parameters results in an 18% improvement in recall.

Keywords: *Optimization; Hyper-parameters; Coronavirus; Machine learning; Model*

1. Introduction

As the coronavirus disease (COVID-19) posed a worldwide risk to lives, businesses, and travel, reports on the virus are in a constant state of flux. On January 30, 2020, and March 11, 2020, respectively, the World Health Organization (WHO) classified COVID-19 as a public health emergency of international concern and a pandemic. Assembling solutions to this health crisis requires immediate action. Hours are required to analyze a nasopharyngeal specimen submitted to a laboratory for virologic or PCR tests that detect the virus (Qiu *et al.*, 2016).

Despite this, many nations continue to lack adequate COVID-19 testing capabilities, which necessitates the development of additional early detection tools to prevent the virus's spread (Laghmati *et al.*, 2020; Moujahid *et al.*, 2020; Terrada, *et al.*, 2020; Touzani *et al.*, 2020; Toubi *et al.*, 2024). Understanding the virus's transmission and implementing evidence-based measures to limit the pandemic are facilitated by early diagnosis. A WHO report from February 2020 posited that the utilization of big data analysis and Artificial Intelligence (AI) might prove indispensable in the fight against the COVID-19 pandemic (Terrada, *et al.,* 2020). As a subfield of AI, Machine Learning (ML) is concerned with the design, analysis, and implementation of automated learning methods for machines. Significantly utilized in the medical sphere, ML has demonstrated its worth in predicting positive cases of numerous diseases (Song *et al.,* 2019). A machine learning model is a mathematical construct whose parameters require data-driven tuning. Users select hyper-parameters, which are not amenable to direct examination, through testing and intuition prior to the commencement of the learning process. These parameters have a substantial influence on the efficacy of the model, including aspects like complexity and learning speed. Finding the optimal combination of the numerous hyper-parameters that models may contain is an independent research problem (Feurer & Hutter, 2019).

The objective of this research is to improve the efficacy of machine learning classification models through the optimization of algorithm parameters. Furthermore, the objective is to construct an early detection model that utilizes ML techniques to distinguish between healthy individuals and COVID-19 cases by analyzing viral and blood data. The primary objective is to compare the performance of classification and prediction algorithms by employing the GridSearch function, which delineates the parameters to be evaluated throughout the training phase. Parameter grids are dictionaries in which the values represent ranges of values to be verified and the keys represent the names of the parameters. Five ML algorithms are trained and evaluated using a COVID-19 dataset comprising blood and viral test results, representing various categories (Guyon *et al.,* 2019).

The structure of the remainder of this paper is as follows: Section 2 offers a concise synopsis of pertinent supplementary literature. In Section 3, the dataset structure and the machine learning algorithms that were examined are detailed. The stages and components of the proposed prediction methodology are delineated in Section 4. The experimental outcomes and a discussion of performance evaluation metrics are provided in Section 5. Section 6 provides a concluding remark and outlines potential future developments.

2. Literature survey

In recent years, there has been a surge in research focused on developing predictive systems for chronic and cancerous diseases. The goal of these studies is to enhance performance and create robust prediction models. Parameter selection is a critical aspect of modeling as it significantly impacts the precision performance metric of the forecast model (Guyon *et al.,* 2019). In the field of programming, numerous functions are available for exhaustively searching hyperparameter values. These functions are often executed on computation clusters due to their computational expense. The GridSearch function, proposed by (James Bergstra, 2012), is one of the simplest and most widely used methods.

In a study by (Fayed & Atiya, 2019), an approach was introduced to expedite the grid search function for selecting optimal parameters for Support Vector Machine (SVM) classifiers. This approach aimed to avoid retraining each SVM model from scratch by leveraging support vectors obtained during the training process of an SVM with a smaller value of the C parameter, rather than starting anew with the entire dataset.

Another study by (Kaur *et al.,* 2020) proposed a clinical decision-making system to aid in diagnosing patients with Parkinson's disease. The study focused on optimizing the GridSearch function and developing an optimized deep learning model for early prediction of Parkinson's disease. Fine-tuning the parameters of the deep learning model yielded an overall test accuracy of 89.23% and an average classification accuracy of 91.69%.

In a different research endeavor, (Victoria & Maragatham, 2020) applied hyperparameter optimization of the Bayesian algorithm on the CIFAR-10 dataset to enhance model performance. Bayesian optimization efficiently obtained optimized values for all hyperparameters, resulting in time savings and improved performance. The results demonstrated a 6.2% reduction in error on the graphics processing unit during validation.

3. Tools and materials

This section contains an account of the dataset and the classifier methods that were implemented during the experiments. Acquisition of Datasets.

3.1 Dataset

The utilized database comprises anonymized patient information obtained from Albert Einstein Hospital (AEH) visits. This dataset comprises two distinct categories: patients who are healthy (Negative cases) and those who are COVID-19-infected (Positive cases). Conversely, this dataset comprises 111 features and 5644 instances, albeit with a significant number of absent values. It was imperative to conduct a preprocessing investigation by eliminating the null values in order to accomplish this. We then eliminate features with a data lacuna greater than eighty percent.

3.2 ML algorithms

3.2.1 Support Vector Machine

It consists of every algorithm required to tackle problems involving classification analysis and regression. SVM Classification is predicated on whether an object in an N-dimensional space is situated in one of two classifications. By constructing a hyperplane of dimension N-1, the support vector method ensures that every object is contained within one of the two groups (Lameski et al., 2015). This is illustrated in **Figure 1** as follows: Instances of two distinct classes—Positive and Negative—are present, and they are linearly divisible.

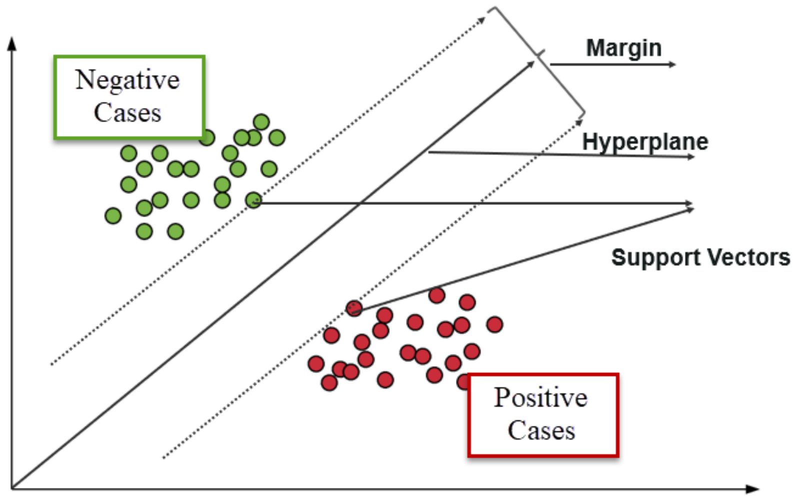

Figure 1. The SVM classification of binary class data (Positive, Negative).

This method not only divides the instances but also produces a hyperplane that is positioned at the greatest distance possible from the case that is closest to each class. SVM and its variants aid in the resolution of complex problems. The selection of the regularization kernel parameters influences the efficacy of the SVM classifier (Liu *et al.*, 2006; Syarif *et al.*, 2016).

3.2.2 Decision Tree Algorithm

The decision tree method is typically applied to binary classification problems. By organizing the gathered attributes into a graphical tree consisting of nodes and branches, the aim is to construct a set of options. The DT algorithm constructs the classification model in the form of a tree structure by utilizing if-then principles (Kotsiantis, 2013). Subsequently, the data is decomposed into more manageable structures, which may be linked together in an incremental decision tree fashion. Using training data in a sequential fashion, rules are acquired one at a time. When a rule is acquired knowledge of, the tuples that encompass the rule are removed. The procedure remains ongoing on the training set until it reaches the endpoint (Shaikhina *et al.*, 2019). The model's ultimate architecture resembles a tree composed of nodes and limbs. **Eqn. 1** provides the probability P(T) of estimating that an observation j is in node X.

$$P(T) = \sum_{j \in X} w_j \tag{1}$$

3.2.3 Random Forest Algorithm

Random Forest (RF) is an algorithm that effectively addresses prediction problems. This is, in fact, an instance in which bagging is implemented in decision trees. Bagging methods operate on the principle of averaging the forecasts of multiple independent models in order to diminish variance and forecast error. In order to develop an RF model, a number of bootstrap examples are chosen (Ramadhan *et al.*, 2017). A considerable quantity of decision trees are employed, with each tree being trained using a distinct subset of the training set. The determination at a node of each tree construction is contingent upon a subset of variables that is selected at random. Subsequently, we employ each of the generated decision trees to generate the forecast, selecting the predicted variable of type factor by majority vote (S. Wang *et al.*, 2019). In **Figure 2**, the construction architecture of an RF model is depicted.

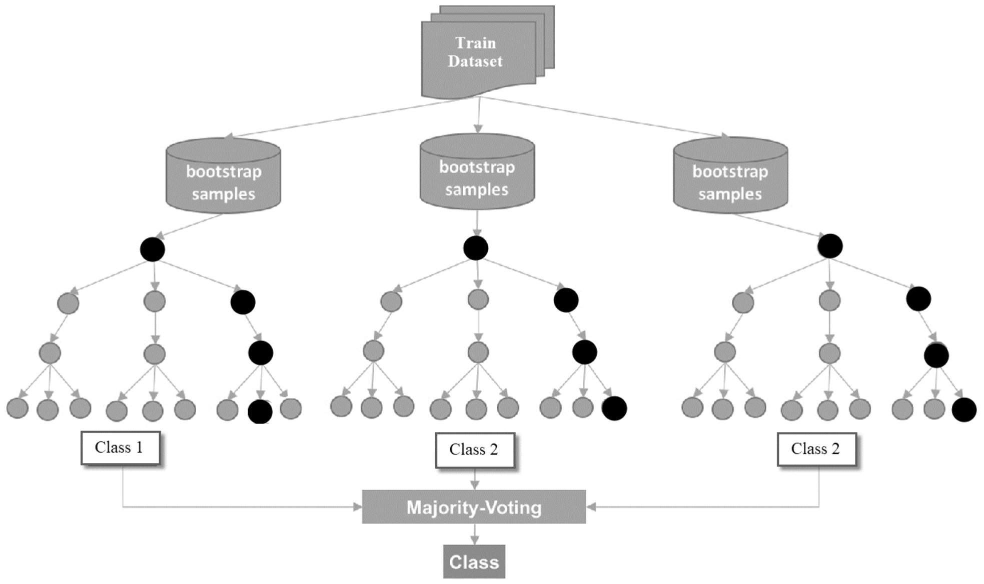

Figure 2. A schematic representation of the Random Forest algorithm in operation.

3.2.4. Adaboost Algorithm

AdaBoost, or Adaptive Boosting, is a well-known algorithmic technique within the domain of machine learning. This is the most prevalent derivative of the Boosting method, which is designed to improve the efficacy of the learning algorithm. AdaBoost utilizes error rate reduction to effectively upgrade a "weak" classifier to a "strong" form. At each iteration, the AdaBoost algorithm invokes a learning algorithm that trains the instances for classification. Subsequently, a novel probability distribution is established for the learning instances (R. Wang, 2012). In light of the algorithm's previous iteration's outcomes, the weight assigned to incorrectly classified instances is increased. By integrating feeble data with a weighted vote, AdaBoost ultimately generates a robust classifier.

Algorithm 1 show the AdaBoost algorithm statements.

Algorithm 1. AdaBoost algorithm statements

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$.
    (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

The AdaBoost algorithm works by iteratively training a series of weak classifiers on weighted versions of the training data and combining their predictions to create a strong classifier. The process begins by assigning equal weights to all training examples. In each iteration, a weak classifier is trained on the current weighted training data to minimize the weighted error rate, where the weights are adjusted to focus more on the examples that were misclassified by previous classifiers. After training, the weak classifier's performance is evaluated based on its ability to correctly classify the weighted training data. The classifier with the lowest weighted error rate is given more influence in the final ensemble. Subsequent weak classifiers are trained using updated weights, placing more emphasis on examples that were misclassified by previous classifiers. The final strong classifier is a weighted sum of the individual weak classifiers, where classifiers with higher performance contribute more to the final prediction. This iterative process allows AdaBoost to progressively improve its performance by focusing on difficult-to-classify examples, leading to a strong classifier that can effectively handle complex decision boundaries and achieve high accuracy on diverse datasets.

Indeed, AdaBoost is, in fact, attempting to determine the optimal classifier by combining a limited learning data set (J. Wang *et al.*, 2011). By designating distinct weights to the learning instances each iteration and subsequently combining the results, the Boosting algorithm is based on the principle of reusing a classifier multiple times in order to identify a single "strong/very precise" classifier.

3.2.5. XGBoost algorithm

The XGBoost algorithm is an absolute standout in competitions for machine learning. It is an open-source implementation of the gradient boost tree algorithm that has been optimized. Gradient Boosting is, in fact, an algorithm for supervised learning. The underlying principle is to enhance prediction accuracy by aggregating the outcomes of a reduced-complexity and less robust collection of models.

The concept is straightforward: the algorithm will combine multiple models into a single output rather than calculating each individually. Operation of the algorithm is sequential. In contrast to the random forest, for instance. Although this will result in a delayed execution, it will enable the algorithm to enhance its capitalization in comparison to prior iterations. He commences by constructing and assessing an initial model. Based on the assessment of the initial model, the weighting of each individual will be determined by the accuracy of the prediction (Chen & Guestrin, 2016). A multitude of modifiable and adjustable hyperparameters are incorporated into XGBoost in order to enhance performance.

4. Methodology

4.1 Global Overview

This paper aims to enhance the performance of classification models using the GridSearch function. **Figure 3** illustrates the main stages of building the prediction system for patients infected with COVID-19.

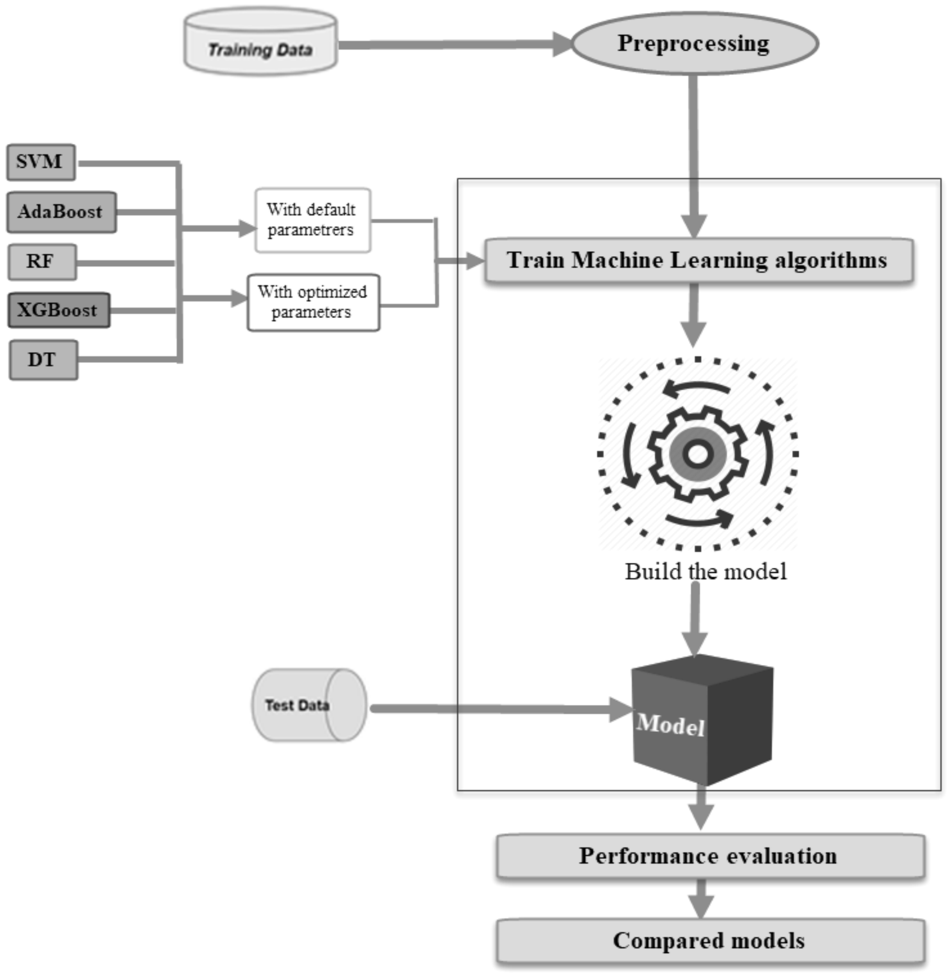

Figure 3. Illustration of the proposed methodology steps.

The training and testing ratio used is 80% and 20%, respectively. Two experiments were conducted: in the first experiment, models were trained and tested with their default hyper-parameters, and performance metrics were calculated along with a confusion matrix. In the second experiment, hyper-parameters were examined, and grid parameters were created for each algorithm. This grid contained a range of values to find the optimal hyper-parameter value. The performance of the optimized models was then evaluated and compared with the results from the first experiment.

4.2 *Pre-processing*

The main purpose of this step is to remove missing values and unnecessary features while retaining significant and correlated features with the outcome (Xue *et al.*, 2016). Preprocessing operations included encoding, removing unnecessary data, and feature engineering. **Table 1** shows the final architecture of the dataset used.

Table 1. The AEH Dataset Architecture.

Instances	Positive : 1	558
	Negative : 0	702
Relevant Features	Input	17
	Output	2
Train set (80%)	Positive cases	436
	Negative cases	572
Test set (20%)	Positive cases	122
	Negative cases	130

4.3 *Parameter Selection with GridSearch*

Optimizing hyper-parameters involves adjusting a parameter to minimize the cost function of the model. This process applies to any parameter of a machine-learning model. Most machine learning models require at least one variable defined before learning (Jiménez *et al.*, 2007). The GridSearch function is used to define the optimal parameters of a model, allowing rapid iteration over possible combinations of values for any hyper-parameter value (Bao & Liu, 2006).

4.4 *Performance Evaluation Methods*

It is essential to evaluate the performance of a classifier in order to determine its precision and effectiveness. We utilized two primary techniques in this investigation: the Holdout method and cross-validation. The Holdout method entails dividing the dataset into two distinct sets: a training set, comprising approximately 80% of the data, which is utilized to train the model; and a test set, comprising the remaining 20% of the data, which is employed to assess the predictive performance of the model. This methodology aids in evaluating the model's ability to generalize to new, unseen data (H. Wang & Zheng, 2013).

In order to enhance the reliability of our assessment, we employed the cross-validation technique. The methodology entails partitioning the dataset into k subsets, commonly denoted as folds, training the model on k-1 of these folds, and subsequently evaluating its performance on the remaining fold. K iterations of this procedure are performed, using each fold as the test set once. We obtain a more accurate estimate of the model's performance and reduce the risk of overfitting, which occurs when a model performs well on the training data but inadequately on unseen data, by averaging the results of these iterations (Dalianis, 2018). Furthermore, we computed a number of evaluation metrics to appraise the effectiveness of our classifiers. The metrics encompassed in this set are as follows: accuracy, which quantifies the classifier's predictions in terms of their overall correctness; precision, which determines the ratio of true positive predictions to total positive predictions; recall, which is also referred to as sensitivity; the Receiver Operating Characteristic (ROC) curve, which depicts the compromise between the true positive rate and the false positive rate; and sensitivity. The combination of these metrics yields a thorough assessment of the classifier's efficacy and precision in distinguishing between positive and negative instances.

5. Results and Discussion

Using the Holdout and Cross-validation approach, we will analyze the performance of the five supervised machine learning algorithms that are included in this part. Through the use of two separate studies, we evaluated the quality of the model using a variety of metrics: As part of the first experiment, the default model hyper-parameters are maintained, and the AEL dataset is utilized with preprocessing. Using the same dataset and performing the same preprocessing, we conduct the second experiment, which involves developing a GridSearch of the parameters for each model in order to determine the optimal combination of parameters.

5.1. Experiment using algorithms with default parameters

The performance results of the algorithms with default hyper-parameters are shown in this portion of the article of the same name. The five algorithms are trained on a total of 1008 examples, and then they are tested on 252 examples. For the purpose of drawing the confusion matrix for each classification, we have first determined the components of the binary classification. The values of the examples that are presented in Table 3 are those in which the prediction is positive and the actual value is positive (also known as a True Positive). Examples of situations in which the prediction is positive but the actual value is negative (also known as a false positive). Examples of situations in which the prediction is negative and the actual value is also negative (also known as a True Negative). In addition, the instances in which the prediction is negative but the actual number is positive (also known as a misleading negative).

In light of these findings, it is clear that the XGBoost, RF, and DT models were able to produce a classification that was superior to that of the other models. For the purpose of providing a clear illustration of these performances, we have computed and given in **Table 2** a number of percentage performance indicators.

Table 2. Performance Evaluation Metric Of The Five Algorithms With Default Hyper-Parameters.

Metrics	**SVM**	**AdaBoost**	**RF**	**XGBoost**	**DT**
Accuracy (%)	86.90	84.52	94.44	94.44	90.08
Precision (%)	98.90	91.92	97.37	96.55	89.43
Recall (%)	73.77	74.59	90.98	91.80	90.16
ROC	0.86	0.84	0.94	0.94	0.90
F1-score	0.84	0.82	0.94	0.94	0.89

The three algorithms that make up the decision trees, namely XGBoost, RF, and DT, collectively have an accuracy of 94%, which is something that we can observe very fast. It is not possible for the SVM and the AdaBoost to go above 86%. The SVM model, on the other hand, outperforms all other models in terms of precision, reaching 98.90%. When it comes to the other metrics that are still in use, such as recall, ROC, and F1, we have observed that the trees algorithm family consistently demonstrates the highest level of performance.

5.2. Experiment using algorithms with optimized parameters

In the second experiment, we trained the same algorithms on the same dataset; however, this time we used the GridSearch function to introduce appropriate hyper-parameters. This was the only difference between the two experiments. Different classifications were obtained by us in comparison to those obtained in Experiment 1. Through the use of this table, we are able to observe that the classification of the five models has been enhanced in comparison to the previously conducted experiment. The updated values of the evaluation measures for performance models are included in **Table 3**.

Table 3. Performance Evaluation Metric of The Five Algorithms with Default Hyper-Parameters.

Models	SVM	AdaBoost	RF	XGBoost	DT
Accuracy (%)	94.05	93.25	94.84	94.44	90.87
Precision (%)	99.08	93.39	99.10	96.55	91.60
Recall (%)	88.52	92.62	90.16	91.80	89.34
ROC	0.93	0.93	0.94	0.95	0.90
F1-score	0.93	0.93	0.94	0.94	0.90

We are able to detect those certain models have undergone a modest improvement in performance when we compare these performances with the results that are mentioned in Table 4. Within the context of the recall measure, the SVM and AdaBoost models, on the other hand, exhibited a notable improvement. The percentage of improvement that each model was able to achieve in comparison to experiment 1 is presented in **Table 4**.

Table 4. Percentage Improvement Rate of The Five Models Compared to Experiment 1.

Metrics	SVM	AdaBoost	RF	XGBoost	DT
Accuracy (%)	**+7.15**	+8.73	+0.4	0	+0.79
Precision (%)	+0.18	+1.47	+1.73	0	+2.17
Recall (%)	**+14.7**	**+18.03**	-0.82	0	-0.82
ROC	+7.38	+9.01	+0.38	+0.8	+0.75
F1-score	+9	+1.35	+0.35	0	+0.66

The XGBoost model maintained the same levels of performance as the first trial, with the exception of a marginal enhancement of 0.8% on the ROC metrics. Both the RF and the DT showed a little improvement in all parameters, with the exception of the recall, which showed a reduction of 2.82 percent. To the contrary, the performance of both the Support Vector Machine (SVM) and the AdaBoost models has been significantly improved. The recall of AdaBoost attained an extreme augmentation of 18.03%, which was followed by the recall of SVM, which reached 14.7% on the same criteria.

5.3. *Experiment using algorithms with optimized parameters*

SVM, AdaBoost, RF, XGBoost, and DT are the five machine-learning algorithms that were utilized in this study project. The purpose of these experiments was to enhance the prediction of patients who were infected with COVID-19. An open dataset that contained information on blood tests performed on about 5,644 people was investigated by us. A number of pre-processing processes were carried out by us in order to extract the essential characteristics and minimize the occurrence of missing information. Finally, we were able to acquire a dataset that included 1260 different occurrences. Both of the tests were conducted with the same separation ratio of the dataset, which was 80% for training and 20% for testing. The algorithms were trained on this ratio. The second experiment involves the utilization of a GridSearch for each algorithm in order to determine the hyper-parameters that are ideal. The bulk of the models have shown an improvement in their performance, as we discovered from the part on the testing data. In terms of the recall measure, the AdaBoost model saw a significant improvement. The support vector machine (SVM) also demonstrates a considerable improvement across all criteria. The fact that the two classifications that were produced by the SVM and RF models both have a FN that is equal to one is a very significant note that we have derived from the confusion matrices. When it comes to identifying patients who are genuinely infected with COVID-19, these two models have an extremely high level of sensitivity.

Conclusion

The optimization of the hyper-parameter is a significant factor in each algorithm, as stated in the previous sentence. This has the potential to have an impact on the effectiveness of the model learning. The performance of a model was shown to be improved and increased through the utilization of a grid search function, as that was demonstrated in this paper. By using this grid search, the optimal configuration of an algorithm can be determined. This study has the potential to assist future researchers in selecting the most appropriate hyper-parameters for their model in order to enhance it by utilizing one of the GridSearch functions. We are going to design an architecture for an optimized convolutional neural network model that is based on Transfer Learning techniques in the work that we will be doing in the future. X-ray or CT images will be used to address the challenges associated with the detection of patients who are infected with COVID-19.

Disclosure statement: *Conflict of Interest:* The authors declare that there are no conflicts of interest.
Compliance with Ethical Standards: This article does not contain any studies involving human or animal subjects.

References

Bao, Y., & Liu, Z. (2006). A Fast Grid Search Method in Support Vector Regression Forecasting Time Series. In E. Corchado, H. Yin, V. Botti, & C. Fyfe (Eds.), *Intelligent Data Engineering and Automated Learning – IDEAL* 2006 (Vol. 4224, pp. 504–511). Springer Berlin Heidelberg. https://doi.org/10.1007/11875581_61

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Dalianis, H. (2018). Evaluation Metrics and Evaluation. In H. Dalianis, Clinical Text Mining (pp. 45–53). Springer International Publishing. https://doi.org/10.1007/978-3-319-78503-5_6

Fayed, H. A., & Atiya, A. F. (2019). Speed up grid-search for parameter selection of support vector machines. *Applied Soft Computing*, 80, 202–210. https://doi.org/10.1016/j.asoc.2019.03.037

Feurer, M., & Hutter, F. (2019). Hyperparameter Optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated Machine Learning* (pp. 3–33). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_1

Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., Statnikov, A., Tu, W.-W., & Viegas, E. (2019). Analysis of the AutoML Challenge Series 2015–2018. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), Automated Machine Learning (pp. 177–219). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_10

James Bergstra. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13, 281-305.

Jiménez, Á. B., Lázaro, J. L., & Dorronsoro, J. R. (2007). Finding Optimal Model Parameters by Discrete Grid Search. In E. Corchado, J. M. Corchado, & A. Abraham (Eds.), *Innovations in Hybrid Intelligent Systems* (Vol. 44, pp. 120–127). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74972-1_17

Kaur, S., Aggarwal, H., & Rani, R. (2020). Hyper-parameter optimization of deep learning model for prediction of Parkinson’s disease. *Machine Vision and Applications*, 31(5), 32. https://doi.org/10.1007/s00138-020-01078-1

Kotsiantis, S. B. (2013). Decision trees: A recent overview. Artificial Intelligence Review, 39(4), 261–283. https://doi.org/10.1007/s10462-011-9272-4

Laghmati, S., Cherradi, B., Tmiri, A., Daanouni, O., & Hamida, S. (2020). Classification of Patients with Breast Cancer using Neighbourhood Component Analysis and Supervised Machine Learning Techniques. 2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet), Marrakech, Morocco, Morocco. https://doi.org/10.1109/CommNet49926.2020.9199633

Lameski, P., Zdravevski, E., Mingov, R., & Kulakov, A. (2015). SVM Parameter Tuning with Grid Search and Its Impact on Reduction of Model Over-fitting. In Y. Yao, Q. Hu, H. Yu, & J. W. Grzymala-Busse (Eds.), *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing* (Vol. 9437, pp. 464–474). Springer International Publishing. https://doi.org/10.1007/978-3-319-25783-9_41

Liu, R., Liu, E., Yang, J., Li, M., & Wang, F. (2006). Optimizing the Hyper-parameters for SVM by Combining Evolution Strategies with a Grid Search. In D.-S. Huang, K. Li, & G. W. Irwin (Eds.), *Intelligent Control and Automation* (Vol. 344, pp. 712–721). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-37256-1_87

Moujahid, H., Cherradi, B., Bahatti, L., Terrada, O., & Hamida, S. (2020). Convolutional Neural Network Based Classification of Patients with Pneumonia using X-ray Lung Images. *Advances in Science, Technology and Engineering Systems Journal*, 167–175.

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1), 67. https://doi.org/10.1186/s13634-016-0355-x

Ramadhan, M. M., Sitanggang, I. S., Nasution, F. R., & Ghifari, A. (2017). Parameter Tuning in Random Forest Based on Grid Search Method for Gender Classification Based on Voice Frequency. *DEStech Transactions on Computer Science and Engineering, cece*. doi.org/10.12783/dtcse/cece2017/14611

Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R., & Khovanova, N. (2019). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. *Biomedical Signal Processing and Control*, 52, 456–462. https://doi.org/10.1016/j.bspc.2017.01.012

Song, H., Triguero, I., & Özcan, E. (2019). A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence*, 8(2), 143–165. https://doi.org/10.1007/s13748-019-00185-z

Syarif, I., Prugel-Bennett, A., & Wills, G. (2016). SVM Parameter Optimization using Grid Search and Genetic Algorithm to Improve Classification Performance. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14(4), 1502. https://doi.org/10.12928/telkomnika.v14i4.3956

Terrada, O., Cherradi, B., Hamida, S., Raihani, A., Moujahid, H., & Bouattane, O. (2020). Prediction of Patients with Heart Disease using Artificial Neural Network and Adaptive Boosting techniques. 2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet). https://doi.org/10.1109/CommNet49926.2020.9199620

Terrada, O., Hamida, S., Cherradi, B., Raihani, A., & Bouattane, O. (2020). Supervised Machine Learning Based Medical Diagnosis Support System for Prediction of Patients with Heart Disease. *Advances in Science, Technology and Engineering Systems Journal*, 269–277.

Toubi Y., Hakmaoui Y., EL Ajlaoui R., Abrigach F., Zahri D., Radi S., Rakib E. M., Lgaz H., Hammouti B. (2024) Unexpected Efficient One-Pot Synthesis, DFT Calculations, and Docking study of new 4-hydroxy-2H-chromen-2-one Derivatives predicted to target SARS-CoV-2 spike protein., *J. Molecular Structure*, 2024, 136789, doi.org/10.1016/j.molstruc.2023.136789

Touzani R., Hammouti B., Almalki F.A., Ben Hadda T. (2020) Coronavirus, Covid19, Covid-19 and SARS-Cov-2: A Global Pandemic, A Short Review, *J. Mater. Environ. Sci.*, 11(4), 736-750

Victoria, A. H., & Maragatham, G. (2020). Automatic tuning of hyperparameters using Bayesian optimization. Evolving Systems. https://doi.org/10.1007/s12530-020-09345-2

Wang, H., & Zheng, H. (2013). Model Validation, Machine Learning. In W. Dubitzky, O. Wolkenhauer, K.-H. Cho, & H. Yokota (Eds.), *Encyclopedia of Systems Biology* (pp. 1406–1407). Springer New York. https://doi.org/10.1007/978-1-4419-9863-7_233

Wang, J., Gao, L., Zhang, H., & Xu, J. (2011). Adaboost with SVM-Based Classifier for the Classification of Brain Motor Imagery Tasks. In C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction. Users Diversity (Vol. 6766, pp. 629–634). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-21663-3_68

Wang, R. (2012). AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review. *Physics Procedia*, 25, 800–807. https://doi.org/10.1016/j.phpro.2012.03.160

Wang, S., Ren, W., Zhang, Y., & Liang, F. (2019). Random Forest Classifier for Distributed Multi-plant Order Allocation. In G. Q. Huang, C.-F. Chien, & R. Dou (Eds.), *Proceeding of the 24th International Conference on Industrial Engineering and Engineering Management* 2018 (pp. 123–132). Springer Singapore. https://doi.org/10.1007/978-981-13-3402-3_14

Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2016). A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4), 606–626. https://doi.org/10.1109/TEVC.2015.2504420