Journal of Computer and Communications, 2024, 12, 160-183
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Particle Swarm Optimization-Based Hyperparameters Tuning of Machine Learning Models for Big COVID-19 Data Analysis

Hend S. Salem, Mohamed A. Mead, Ghada S. El-Taweel

Department of Computer Sciences, Suez Canal University, Ismailia, Egypt
Email: hend_shaaban@ci.suez.edu.eg, mohamedmead@ci.suez.edu.eg, ghada@ci.suez.edu.eg

How to cite this paper: Salem, H.S., Mead, M.A. and. El-Taweel, G.S (2024) Particle Swarm Optimization-Based Hyperparameters Tuning of Machine Learning Models for Big COVID-19 Data Analysis. *Journal of Computer and Communications*, **12**, 160-183.
https://doi.org/10.4236/jcc.2024.123010

Received: February 22, 2024
Accepted: March 24, 2024
Published: March 27, 2024

Abstract

Analyzing big data, especially medical data, helps to provide good health care to patients and face the risks of death. The COVID-19 pandemic has had a significant impact on public health worldwide, emphasizing the need for effective risk prediction models. Machine learning (ML) techniques have shown promise in analyzing complex data patterns and predicting disease outcomes. The accuracy of these techniques is greatly affected by changing their parameters. Hyperparameter optimization plays a crucial role in improving model performance. In this work, the Particle Swarm Optimization (PSO) algorithm was used to effectively search the hyperparameter space and improve the predictive power of the machine learning models by identifying the optimal hyperparameters that can provide the highest accuracy. A dataset with a variety of clinical and epidemiological characteristics linked to COVID-19 cases was used in this study. Various machine learning models, including Random Forests, Decision Trees, Support Vector Machines, and Neural Networks, were utilized to capture the complex relationships present in the data. To evaluate the predictive performance of the models, the accuracy metric was employed. The experimental findings showed that the suggested method of estimating COVID-19 risk is effective. When compared to baseline models, the optimized machine learning models performed better and produced better results.

Keywords

Big COVID-19 Data, Machine Learning, Hyperparameter Optimization, Particle Swarm Optimization, Computational Intelligence

1. Introduction

Big data analysis, especially in the medical field, helps manage mortality risks

and give patients the right care. The COVID-19 pandemic has highlighted the critical need for effective risk prediction models to enhance public health strategies worldwide. As a respiratory infection caused by the SARS-CoV-2 virus, COVID-19 has significantly impacted global health systems, necessitating advanced approaches for risk assessment and management [1]. In response to this challenge, machine learning (ML) techniques have emerged as valuable tools for analyzing complex medical data and predicting disease outcomes with a high degree of accuracy.

This study focuses on addressing this challenge by leveraging the Particle Swarm Optimization (PSO) algorithm to effectively search the hyperparameter space and enhance the predictive performance of ML models. The main objective of this research is to develop and evaluate optimized ML models for predicting COVID-19 risk by identifying the optimal hyperparameters that maximize accuracy.

To achieve this objective, a diverse dataset encompassing clinical and epidemiological characteristics linked to COVID-19 cases serves as the foundation for model development and evaluation. Various ML algorithms, including Random Forests, Decision Trees, Support Vector Machines, and Neural Networks, are employed to capture the intricate relationships within the data and facilitate accurate risk prediction.

The significance of this research lies in its potential to advance the field of COVID-19 risk prediction through the application of state-of-the-art ML techniques. By optimizing hyperparameters using the PSO algorithm, this study aims to enhance the accuracy and reliability of ML models for predicting COVID-19 risk, ultimately contributing to more effective public health interventions and patient care strategies.

ML is a branch of artificial intelligence (AI) with an emphasis on developing systems that can learn and improve on their own without being explicitly programmed [2].

Some uses of AI and machine learning technologies to battle the COVID-19 pandemic are:

• Early detection: AI and machine learning algorithms have been used to develop tools for early detection of COVID-19, based on symptoms, medical history, and other factors [3].

• Diagnosis: Machine and deep learning algorithms have been used to analyze chest CT scans and X-ray images to accurately diagnose COVID-19 with a high degree of sensitivity and specificity [4].

• Epidemiological modeling: Machine learning algorithms have been used to develop epidemiological models to predict the infection and spread of COVID-19 [5].

• Contact tracing: AI and machine learning algorithms have been used to develop contact tracing tools that can quickly identify and isolate people who have been in close contact with COVID-19 patients [6].

The effectiveness of a model can be significantly impacted by choosing the right hyperparameters. Underfitting can occur when a hyperparameter is set too low, and overfitting can occur when a hyperparameter is set too high. Finding the ideal hyperparameter values that strike a balance between model complexity and performance is crucial.

Amidst the vast array of ML methodologies, the selection and optimization of hyperparameters play a pivotal role in enhancing model performance. Hyperparameter optimization, the process of fine-tuning configuration settings that are not learned during training, significantly impacts the predictive power and generalization ability of ML models [7].

However, achieving optimal hyperparameter settings presents a formidable challenge, particularly in the context of COVID-19 risk prediction. This study focuses on addressing this challenge by leveraging the Particle Swarm Optimization (PSO) algorithm to effectively search the hyperparameter space and enhance the predictive performance of ML models. The main objective of this research is to develop and evaluate optimized ML models for predicting COVID-19 risk by identifying the optimal hyperparameters that maximize accuracy.

To achieve this objective, a diverse dataset encompassing clinical and epidemiological characteristics linked to COVID-19 cases serves as the foundation for model development and evaluation. Various ML algorithms, including Random Forests, Decision Trees, Support Vector Machines, and Neural Networks, are employed to capture the intricate relationships within the data and facilitate accurate risk prediction.

Numerous benefits come with the hyperparameters optimization process [8], such as improved model generalization, faster convergence, increased robustness, reproducibility, and cost-cutting. Various methods are employed for optimising hyperparameters, including Grid Search [9], Random Search [10], Bayesian Optimization [11], Genetic Algorithm [9], Gradient-Based Optimization [12], Swarm intelligence [13] [14], and Ensemble-based Methods [15].

The significance of this research lies in its potential to advance the field of COVID-19 risk prediction through the application of state-of-the-art ML techniques. By optimizing hyperparameters using the PSO algorithm, this study aims to enhance the accuracy and reliability of ML models for predicting COVID-19 risk, ultimately contributing to more effective public health interventions and patient care strategies.

In general, the dataset and particular problem being used determine which hyperparameter optimization method is best. In order to increase the likelihood of finding the optimal hyperparameters, it is frequently advised to utilize a variety of approaches. In addition, it is imperative to employ sufficient validation procedures to guarantee that the model does not overfit to the validation data while optimizing hyperparameters.

The remainder of this paper is organized as follows. Section 0 provides an overview of related work on machine and deep learning models applied to COVID-19 prediction. Section 0 presents the materials and methods, providing

a detailed description of the research approach and the dataset. Section 0.0.7 describes the experimental setup and evaluation method as well as presents the findings obtained from the study and offers an in-depth analysis and interpretation of the results. Finally, section 14 concludes the paper with a summary of our findings and directions for future research.

2. Literature Review

Research on machine and deep learning for COVID-19 has rapidly expanded since the pandemic began, covering diagnosis, prognosis, and treatment planning. These methods have been used to differentiate COVID-19 from other respiratory diseases using clinical data and imaging.

The study in [16] employed a deep learning algorithm to analyze chest CT scans for precise COVID-19 diagnosis and forecast patient outcomes with machine learning. Authors in [17] used machine learning to develop a predictive model for COVID-19 mortality based on patient demographic data, medical history, and symptoms. The research in [18] utilized a deep learning model to automatically detect abnormalities in chest CT scans of COVID-19 patients and evaluated its performance against radiology residents. In [4], a deep learning approach using X-ray images to detect COVID-19 achieved notable accuracy and sensitivity. The paper [19] introduced a model for predicting drug-target interactions (DTIs) using the structural properties of proteins and drugs. In [20], a diverse set of machine learning algorithms was used to assess a combined dataset for predicting COVID-19 based on symptoms. Authors in [21] introduced DeepCOVID-XR, an AI algorithm for detecting COVID-19 on chest radiographs with high accuracy. The study in [22] proposed COVID-Net CT, a deep convolutional neural network for detecting COVID-19 cases from chest CT images, achieving remarkable accuracy. In [5], authors developed supervised machine learning models to forecast COVID-19 infection cases in Mexico, achieving high accuracy without optimization. The study represented in [23] a deep learning method for predicting epidemic risk using recurrent neural networks and deep reinforcement learning was proposed. In [24] presented a method for forecasting COVID-19 cases, deaths, and recoveries using LR, SVM, ES, and LASSO, with ES showing the most accurate performance. The paper [25] integrated PSO with various machine learning techniques for predicting landslide susceptibility, achieving high accuracy. The study in [26] focused on predicting heart disease using machine learning techniques, employing PSO for optimizing model parameters. The paper [27] explored using PSO to optimize parameters of deep learning models, highlighting its effectiveness over grid search. The study in [28] addressed the challenge of setting hyperparameters for CNNs by proposing a DPSO approach, achieving significant speedup. The research [29] presented a framework for optimizing SVM hyperparameters using PSO, with results indicating improved effectiveness and efficiency. The research in [30] introduced SAFE-PSO for automatically optimizing hyperparameters and archi-

tecture of NNs, demonstrating high effectiveness and efficiency.

In summary, Research on machine and deep learning for COVID-19 management has surged, showing potential in diagnosis, prognosis, and treatment planning by accurately identifying cases and predicting outcomes. Deep learning excels in analyzing medical imaging like chest CT scans, while machine learning forecasts disease progression using demographic and clinical data. Despite the advancements in utilizing these techniques, there is a notable absence of optimization approaches in many studies, which could potentially enhance the accuracy and efficiency of predictive models. Furthermore, beyond COVID-19 research, machine learning and optimization techniques have been applied in various domains.

However, the optimization of hyperparameters and model architectures remains a challenging and computationally expensive task in many deep learning applications. Addressing this challenge in the proposed study could significantly improve the performance and efficiency of machine learning models across diverse domains, leading to more accurate predictions and better-informed decision-making processes.

3. Materials and Methods

The overall prediction framework is depicted in **Figure 1**. The prediction procedure is divided into five stages: Covid-19 data collecting, preprocessing, prediction, performance evaluation, and visualisation of outcomes. The data collection phase involves gathering the Novel COVID-2019 dataset, which is then utilized to assess the machine learning models. Preprocessing phase involves changing, purifying, updating, and preparing the dataset for machine learning model training. Prediction phase includes the prediction of the COVID-19 risk, multiple PSO-based refined machine learning models were used on the training dataset. During the performance evaluation phase, the results are scrutinized and assessed to ascertain their effectiveness. Result visualisation phase entails converting the prediction outcomes into a graphical representation. **Figure 2** shows the detailed steps of the prediction process.

3.1. Dataset

In this study, the "COVID-19 Dataset" is obtained from Kaggle [31]. This dataset includes 21 distinct features and 1,048,576 distinct cases. The dataset's features columns are described in **Table 1**. The features' data types include date and numeric.

3.2. Data Preprocessing

Three tasks had been performed to preprocess the COVID-19 Dataset:

• Label-encoding: One Hot Encoding was used to encode each column with a numeric value (0 or 1) except columns "MEDICAL_UNIT", "AGE", and "CLASIFFICATION_FINAL".

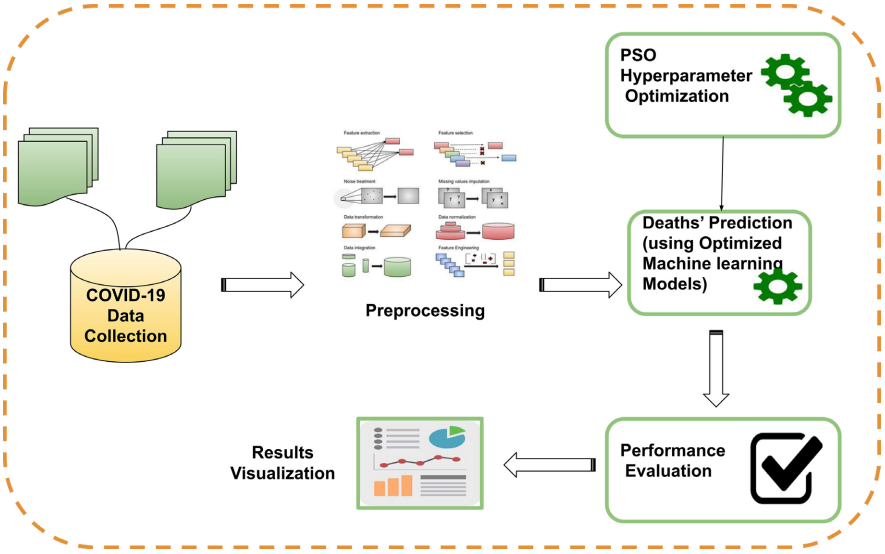


Figure 1. Schematic architectural diagram of the proposed COVID-19 risk prediction framework.

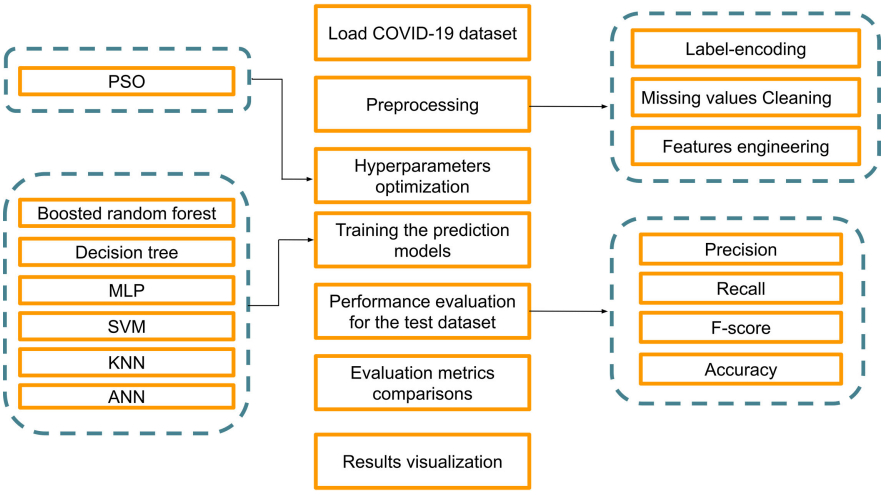


Figure 2. Block diagram of COVID-19 prediction process.

Table 1. COVID-19 dataset description.

Feature	Type	Description
USMER	Numerical	the patient treated medical units of 1st, 2nd or 3rd level
MEDICAL_UNIT	Numerical	type of institution that provided the care
SEX	Numerical	1 - female. 2 - male
PATIENT_TYPE	Numerical	type of care the patient received
DATE_DIED	Date	The date of death, or 9999-99-99 otherwise
INTUBED	Numerical	whether the patient was connected to the ventilator
PNEUMONIA	Numerical	whether the patient already have air sacs inflammation or not
AGE	Numerical	Age of the patient.

Continued

PREGNANT	Numerical	whether the patient is pregnant or not
DIABETES	Numerical	whether the patient has diabetes or not
COPD	Numerical	has Chronic obstructive pulmonary disease or not
ASTHMA	Numerical	whether the patient has asthma or not
INMSUPR	Numerical	whether the patient is immunosuppressed or not
HIPERTENSION	Numerical	whether the patient has hypertension or not
OTHER_DISEASE	Numerical	whether the patient has other disease or not
CARDIOVASCULAR	Numerical	whether the patient has heart or blood vessels related disease
OBESITY	Numerical	whether the patient is obese or not
RENAL_CHRONIC	Numerical	whether the patient has chronic renal disease or not
TOBACCO	Numerical	whether the patient is a tobacco user
CLASIFFICATION_FINAL	Numerical	Covid test results.
ICU	Numerical	whether the patient had been intensive care unit

• Cleaning the missing values: the rows containing values 97 and 99 were missing, so the data was cleaned to preserve rows that contain 1 and 2 values only.

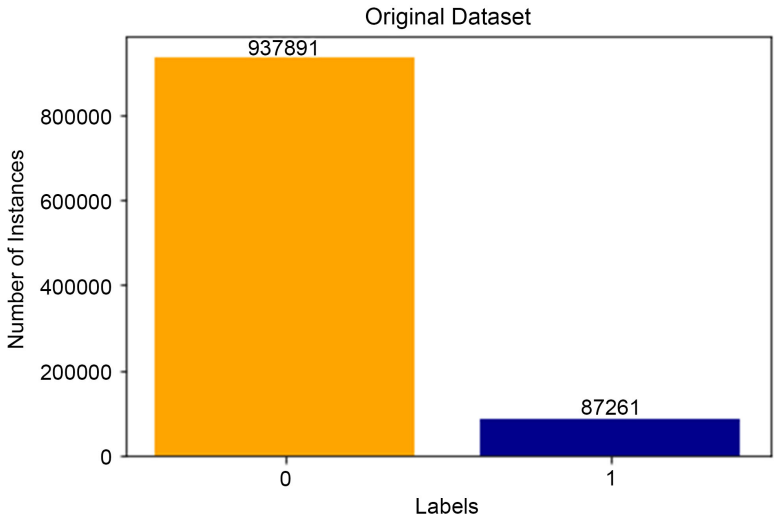
• Feature engineering: columns "DATE_DIED", "INTUBED", and "ICU" were summed together resulting a label column "AT_RISK" that expresses whether the patient is at risk from COVID-19 or not.

• Handling imbalancing classes by using Synthetic Minority Over-sampling Technique (SMOTE): SMOTE is a statistical technique used to increase the number of cases in a dataset in a balanced way [32]. It works by creating synthetic samples from the minority class instead of creating copies. This is achieved by randomly selecting a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are then created by choosing one of the k-nearest neighbors and forming a linear combination with the original point. This method helps in balancing the dataset by augmenting the minority class, which improves the performance of the classification algorithms. Unlike simple oversampling techniques that replicate existing samples, SMOTE generates new instances that are plausible and within the feature space of the minority class. This leads to a more diverse and representative sample of the minority class, reducing the likelihood of overfitting that is common with simple oversampling. By synthesizing new examples, SMOTE can improve the decision boundary derived by the classifier, making it more robust and accurate in distinguishing between the classes. **Figure 3** portraits the affect of SMOTE on the imbalancing dataset. In the "Original Dataset" chart in **Figure 3(a)**, we see a significant imbalance between the two classes, with the majority class (label 0) having 937,891 instances compared to 87,261 instances in the minority class (la-

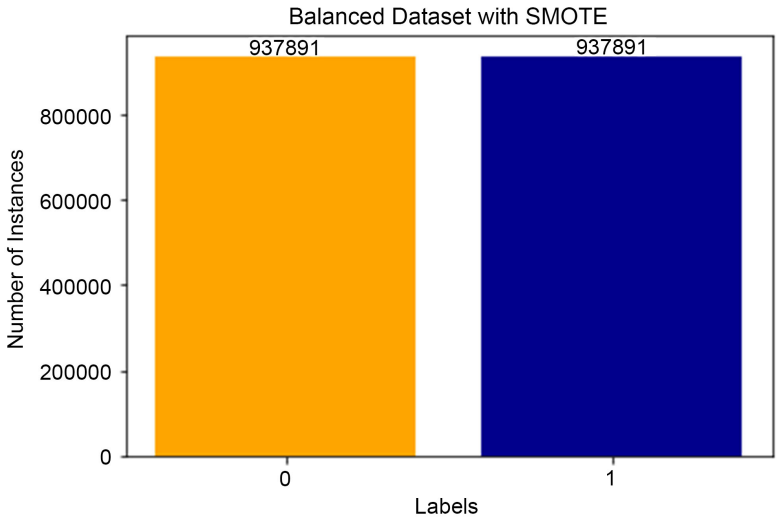
bel 1). This kind of imbalance can skew the performance of machine learning models, often leading to a bias towards the majority class. The balanced dataset with SMOTE chart in **Figure 3(b)** shows the class distribution after applying SMOTE. Both classes are now equal with 937,891 instances each. By synthesizing new samples in the minority class, SMOTE has created a balanced dataset, which can potentially improve the performance of a classifier. With this balance, a model can better learn the characteristics of both classes, leading to improved generalization and a more accurate prediction on unseen data.

3.3. Hyperparameters of Machine Learning Models

Selecting and configuring the machine and deep learning algorithms is a challenging task in the prediction process. The performance of numerous models is dependent on picking the appropriate model configurations [33]. Consequently,



(a)



(b)

Figure 3. Balancing the COVID-19 dataset by SMOTE.

it is critical to successfully define the model's ideal hyperparameters prior to model learning. Hyperparameters are the parameters of the machine learning algorithms that shape the structure of the machine learning models and they are set by the user before the learning phase [34]. Certain hyperparameters, like the learning rate for neural network training or the SVM model's penalty parameter, are used to configure the model, while others, like the neural network's activation function and the SVM model's kernel type, are used to choose the algorithm that minimizes the cost function. The hyperparameters can be categorized as discrete, continuous, or categorical hyperparameters [35]. **Table 2** provides a list of general hyperparameters that are used to build machine learning models. The process of determining the ideal combination of hyperparameter settings that maximises the overall model's performance and accuracy is known as hyperparameter tuning [36]. Because of its direct influence on the accuracy of the analysis and prediction tasks, the hyperparameter tuning process is regarded as a game changer for any machine learning system.

Table 2. Hyperparameters for machine learning models.

Machine Learning Models	Hyperparameter
Boosted Random Forest	n estimators max depth criterion min samples split min samples leaf max features
Decision Tree	criterion max depth min samples split min samples leaf max features
SVM	C kernel
KNN	N neighbors
MLP ANN	number of hidden layers, loss, optimizer, activation, learning rate, dropout rate, epochs, batch size, early stop patience

3.3.1. Particle Swarm Optimization (PSO)

The PSO algorithm is a kind of evolutionary algorithm (EA) that uses natural phenomena as inspiration to find optimal solutions to optimization problems. The PSO works by dispersing a large number of interactive particles, or agents, throughout the search space of the problem. Each particle expresses a candidate solution for the problem. Each particle moves in the search space by tracking its position and velocity, which are updated for each state to keep track of the optimal particles [37]. The main steps of the PSO algorithm are explained in Algorithm 1.

Algorithm 1: The Standard algorithm of Particle Swarm optimization (PSO)

Input:
Number of Particles (N)
Maximum Iterations
Inertia Weight (ω)
Result:
Global Best ($gbest$)
for $i \leftarrow 1$ **to** N_p **do**
Initialize the particle's position $P_i(0) = U(LB, UB)$, where LB and UB represent the lower and upper bounds of the search space;
Initialize $pbest(i, 0) \leftarrow P_i(0)$;
Initialize $gbest$ to the minimal value of the swarm as $gbest \leftarrow \arg\min(f[P_i(0)])$;
Initialize velocity $V_i \sim U(-|UB - LB|, |UB - LB|)$;
repeat
foreach *particle* $i \sim 1$ **to** N_p **do**
Pick random numbers: $r_1, r_2 \sim U(0, 1)$.;
Update velocity $V_i(t+1) = \omega \mathrm{V}_i(t) + c_1 r_1(pbest(i,t) - P_i(t)) + c_1 r_1(gbest(t) - P_i(t))$;
Update position $P_i(t+1) = P_i(t) + V_i(t+1)$;
if $f[P_i(t)] < f[pbest(i,t)]$ **then**
Set best position as $pbest(i,t) \leftarrow P_i(t)$;
if $f[P_i(t)] < f[gbest(t)]$ **then** $gbest(t) \leftarrow P_i(t)$;
Move to next iteration $t \leftarrow t + 1$;
until *termination criteria is met*;
Output: $gbest(t)$ that holds the best found solution

The algorithm takes several inputs, including the number of particles, maximum iterations, inertia weight, learning factors, and hyperparameter boundaries. Each particle in the PSO algorithm represents a potential solution, with its position corresponding to a set of hyperparameters for a specific machine learning model. The algorithm initializes each particle with a random position within the specified hyperparameter boundaries and assigns a random initial velocity. It then evaluates the performance of each particle's hyperparameters using a predefined evaluation function. Throughout the optimization process, particles update their positions based on their velocities, personal best positions, and global best position found by the entire swarm. The velocity update equation includes terms representing the particle's personal best position, the global best position, and the inertia weight, which controls the impact of the particle's previous velocity on its movement. The PSO algorithm iteratively updates the positions of particles over a specified number of iterations. At each iteration, particles compare their current performance with their personal best performance and update

their personal best positions accordingly. Additionally, the algorithm tracks the global best position found by any particle in the swarm and updates it whenever a particle discovers a better solution. After the specified number of iterations, the algorithm returns the global best set of hyperparameters, which represents the optimized configuration for the machine learning model.

3.3.2. Hyperparameters Optimization of Machine Learning Models Using PSO

In this study, the PSO algorithm is utilized to optimize the hyperparameters of the machine learning models. Recent studies have demonstrated that swarm intelligence, particularly PSO, is capable of achieving highly effective results in hyperparameter optimization tasks, particularly when dealing with large-scale datasets. The findings of this study also provide evidence of the efficacy of PSO in optimizing hyperparameters for improved performance. The procedure of hyperparameter optimization process by using the PSO is mentioned in Algorithm 2.

Algorithm 2: PSO for Machine Learning Hyperparameter Optimization.

Input:
Number of Particles (N)
Maximum Iterations ($MaxIter$)
Inertia Weight (ω)
Learning Factors (c_1, c_2)
Hyperparameter Boundaries ($ParamLimits$)

Result:
Global Best set of Hyperparameters of a given ML Model ($BestGlobal$)

for $i \leftarrow 1$ **to** N **do**
  $particle[i].pos \leftarrow$ random_pos_within(ParamLimits)
  $particle[i].vel \leftarrow$ random_init_velocity()
  $particle[i].pbest \leftarrow particle[i].pos$
  $particle[i]$.pbest_score $\leftarrow evaluate(particle[i].pos)$
$BestGlobal \leftarrow$ Best scoring particle among $particle[i].best$ $BestGlobalScore \leftarrow$ Score of $BestGlobal$
for $iter = 1$ ***to*** $MaxIter$ **do**
  foreach *particle* **do**
    foreach *dimension* d **do**
      $rand1, rand2 \leftarrow$ Uniform random numbers in $[0, 1]$
      $personal \leftarrow c_1 \times rand1 \times (particle.best[d] - particle.pos[d])$
      $social \leftarrow c_2 \times rand2 \times (BestGlobal[d] - particle.pos[dimension])$
      $particle.vel[d] \leftarrow \omega \times particle.vel[d] + personal + social$
      $particle.pos \leftarrow$ Update position with $particle.vel$
      Enforce bounds on $particle.pos$ within $ParamLimits$
      current_score $\leftarrow evaluate(particle.pos)$
      if $current\_score > particle.best\_score$ **then**
        $particle.best \leftarrow particle.pos$
        $particle$.best_score $\leftarrow$ current_score
      if $current\_score > BestGlobalScore$ **then**
        $BestGlobal \leftarrow particle.pos$
        $BestGlobalScore \leftarrow$ current_score
return $BestGlobal$

When it comes to machine learning, the selection and optimization of objective functions are pivotal in enhancing the accuracy and efficiency of the models. These functions act as critical navigators for the algorithm, steering it towards

minimizing or maximizing a designated metric, thus ensuring an optimal fit of the model. The significance of choosing an appropriate objective function cannot be overstated, as it fundamentally influences the learning trajectory and the eventual performance of the machine learning models. This research employs supervised learning techniques, wherein models are meticulously trained on a dataset annotated with correct answers. Predominantly, the loss function emerges as the paramount objective function in this context. It measures the variance between the model's predictions and the actual observations, with the primary aim being the reduction of this variance. This objective is pursued through the implementation of the Particle Swarm Optimization (PSO) algorithm. Detailed in the subsequent sections are the explorations of objective functions and the critical tuning of hyperparameters across six distinct machine learning models.

3.4. COVID-19 Risk Prediction

For figuring out COVID-19 risk, six distinct machine learning models were employed: Boosted Random Forest, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), and Artificial Neural Networks. These models were optimized using PSO and trained using a COVID-19 dataset containing various attributes and clinical factors.

3.4.1. K-Nearest Neighbor (KNN)

For each test sample in KNN, the predicted class is determined by identifying the majority class among its k-nearest neighbors within the training set. A class y is denoted by:

$$y = \arg\max_{c_j} \sum_{x_i \in N_k(x)} I\left(y_i = c_j\right), \tag{1}$$

$$i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, m$$

where $I = 1$ when $y_i = c_j$, otherwise $I = 0$; $N_k(x)$ is the field involving the k-nearest neighbors of x. The hyperparameters for K-nearest neighbors (KNN) algorithm are ***n* neighbors** which determines the value of k, the **weights** assigned to each neighbor; weights can be "uniform" or "distance", and the **metric** that specifies the distance metric used to calculate the distances between data points; commonly used distance are "euclidean", "manhattan", and "chebyshev".

3.4.2. Support Vector Machine (SVM)

The objective function of SVM classifier can be expressed as

$$\arg\max_{w} \left\{ \frac{1}{n} \sum_{i=1}^{n} \max\left\{0, 1 - y_i f\left(x_i\right)\right\} + C\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \right\} \tag{2}$$

where $\boldsymbol{w}$ is a normalization vector, C is penalty parameter, the kernel function $f(x)$. **C** one of the hyperparameters that need to be tuned which plays a pivotal role by setting the penalty for incorrect classification of data points. Equally important is the **kernel** hyperparameter, which specifies the kernel function to be used, options including "linear", "polynomial", "sigmoid", or "radial basis func-

tion (RBF)". Furthermore, **gamma** is relevant for specific kernel functions (RBF, polynomial, and sigmoid), influencing the decision boundary's curvature. Lastly, **degree** is a parameter exclusive to the polynomial kernel, dictating the complexity of the model. Each of these hyperparameters directly affects the model's ability to learn and generalize from the data, underscoring the necessity of their careful adjustment.

3.4.3. Boosted Random Forest

Random Forest is an ensemble learning method that combines several decision trees to make predictions. Unlike certain other algorithms, Random Forest's training process does not directly optimize a specific objective function. Rather, it builds a variety of decision trees and aggregates their predictions using a mix of methods, such as bootstrapping and random feature selection.

The following are the RF hyperparameters that require tuning: **criterion**, which defines the function used to assess the quality of a split in each decision tree, **gini impurity**, which quantifies the likelihood of incorrectly classifying a randomly selected element, **entropy**, which quantifies the expected amount of information gained by splitting on a feature, **n estimators**, which establish how many decision trees to include in the RF, **max depth**, which determines the maximum depth allowed for each decision tree in the RF, **min samples split**, which sets the minimum number of samples required to split an internal node, **min samples leaf**, which sets the minimum number of samples required to be at a leaf node, **max features**, that controls the number of features randomly selected for each decision tree.

3.4.4. Artificial Neural Network (ANN)

The kind of objective functions being used determines the performance of the ANN classifier. For various ANN classifier types, there are a number of shared objective functions.

• Cross-Entropy Loss:

Let y_i be the true class label for the i-th sample (binary or one-hot encoded) p_i be the predicted class probability for the i-th sample, then the cross-entropy loss is given by:

$$L_{\mathrm{CE}} = -\sum_i y_i \log\left(p_i\right) \tag{3}$$

• Mean Squared Error (MSE) Loss:

Let y_i be the true class label (0 or 1) for the i-th sample p_i be the predicted class probability (between 0 and 1) for the i-th sample, then the mean squared error loss is given by:

$$L_{\mathrm{MSE}} = \frac{1}{N}\sum_i \left(y_i - p_i\right)^2 \tag{4}$$

• Hinge Loss:

Let y_i be the true class label (−1 or 1) for the i-th sample f_i be the predicted output (before applying the sign function) for the i-th sample, then the hinge loss is given by:

$$L_{\text{hinge}} = \frac{1}{N}\sum_{i}\max\left(0,1-y_i \cdot f_i\right) \tag{5}$$

3.4.5. Multilayer Perceptron Neural Network (MLP)

The objective function of an MLP classifier using cross-entropy loss is given by:

$$L_{\text{CE}} = -\frac{1}{N}\sum_{i}\sum_{j} y_{ij}\log\left(p_{ij}\right) \tag{6}$$

where y_i is the true class label (binary or one-hot encoded) for the i-th sample and p_i be the predicted class probabilities (output) for the i-th sample. The hyperparamters that need to be tune in MLP are: **hidden_layer_sizes** that determines the quantity of neurons present in each hidden layer of the MLP, **activation** which decides the activation function applied in the hidden layers, **solver** that designates the optimization algorithm utilized for training the MLP, **alpha** that commonly referred to as weight decay, which serves the purpose of mitigating overfitting, **learning_rate** establishes the learning rate employed for weight updates during training, **batch_size** determines the quantity of samples utilized in each training batch, and **max_iter** that sets the maximum number of iterations (epochs) for training the MLP.

3.5. Performance Evaluation

The following evaluation metrics were employed in the assessment procedure to ascertain the machine learning models' level of effectiveness.

• **Accuracy:** is the ratio of the total correct predictions (TP + TN + FP + FN) by the predictor or classifier to the total data points (TP + TN) of a dataset. Equation 7 is used to calculate the accuracy metric.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{7}$$

• Recall (sensitivity): evaluates the model's capability to accurately recognize positive instances among all the actual positive instances, providing a measure of the correctly predicted true positives.

The recall metric is calculated by using Equation (8) as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{8}$$

• Precision (positive predictive value): measures the proportion of true positives out of all instances predicted as positive by the model, emphasizing the accuracy of positive predictions. The precision metric is calculated by using Equation (9):

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{9}$$

• F1 score: is a composite metric that integrates precision and recall into a unified value, offering a balanced evaluation of a model's performance by considering both false positives and false negatives. The F1 score is calculated by using Equation (10):

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}} \tag{10}$$

4. Results and Discussion

The proposed methodology was validated using Boosted Random Forest, Decision Tree, SVM, MLP, KNN, and ANN. The hyperparameters of each classifier were adjusted using the PSO method. In the experimental configuration, a laptop running Ubuntu 20.04.1 LTS, equipped with an Intel® CoreTM i5-8250U CPU running at 1.60 GHz and 8 GB of RAM, was used to run the classifiers code by Python 3. Numerous packages and libraries, such as Numpy, Pandas, SciPy, Datetime, Scikit Learn, sklearn, Pyswarm, and Matplotlib, were required by the code. In these implementations, the dataset was split into 80% training and 20% testing and validation. For figuring out COVID-19 risk, the PSO algorithm was used to optimise the parameters of these machine learning models. The models were trained on a COVID-19 dataset. Table 3 summarizes the hyperparameters that were tuned using PSO in this study. Regarding the choice of hyperparameters and their ranges, these are typically determined based on domain knowledge, previous research findings, and experimentation. For example, the maximum depth, maximum features, and number of estimators for Boosted Random Forest are chosen to balance model complexity and performance. Similarly, the choice of hyperparameters for other models such as Decision Tree, SVM, KNN, MLP, and ANN is guided by their respective characteristics and requirements. The ranges for these hyperparameters are selected to cover a wide range of potential values while ensuring that they are meaningful and relevant for the specific machine learning model being optimized.

Table 3. A summary of the machine learning models' hyperparameter values used in the experimental work.

Model	Hyperparameter	Value
Boosted Random Forest	max depth	9
	max features	7
	n estimator	67
Decision Tree	max depth	7
	min samples split	7
SVM	C	79.61
	gamma	0.155
KNN	K	32
	P	1
MLP	hidden layers	2
	number of neurons	2
	tanh	1
	learning rate	0.0776
ANN	hidden layer sizes	(10, 25)
	alpha	0.00029
	learning rate	0.00226

Figures 4-6 demonstrate confusion matrices of optimized models for KNN, SVM, and MLP classifiers, respectively.

To evaluate the performance of the PSO-optimized models, they were compared to baseline models that were trained without optimization. The criterion for this comparison was accuracy. Findings from this evaluation illuminated a notable enhancement in performance, with the PSO-optimized models surpassing their baseline counterparts in terms of accuracy. This outcome underscores the value of PSO in refining machine learning models to achieve superior accuracy.

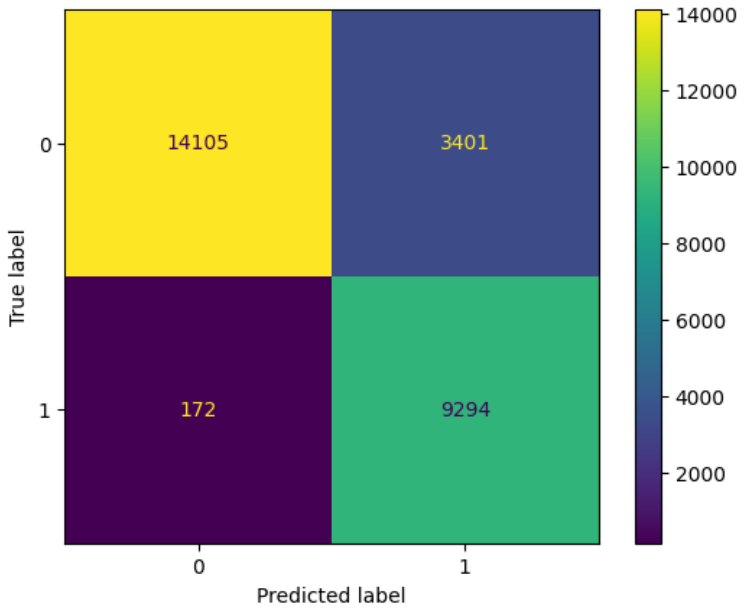


Figure 4. Confusion matrix of optimized KNN.

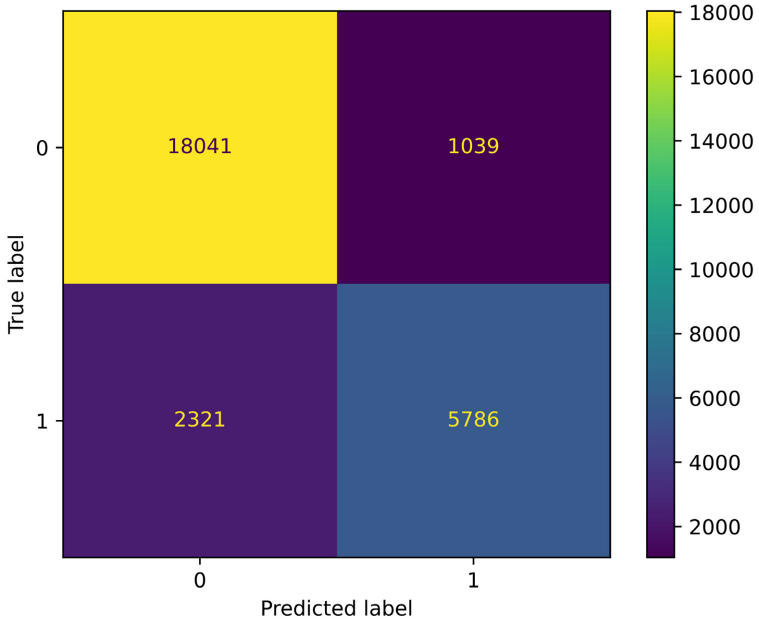


Figure 5. Confusion matrix of optimized SVM.

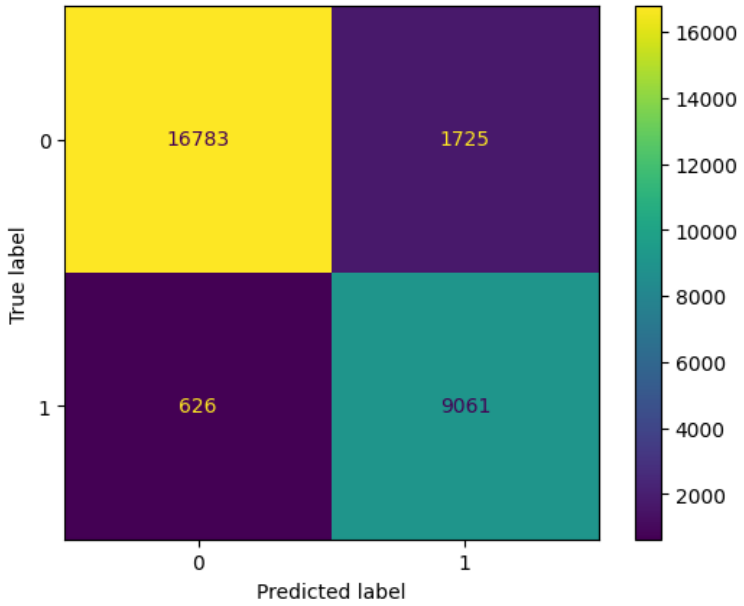


Figure 6. Confusion matrix of optimized MLP.

The optimized models' accuracy varied between 88.26% and 92.20%, whereas the baseline models' accuracy was between 86% and 89.52%. The accuracy of the PSO-optimized models compared to the baseline models was plotted in bar charts to illustrate these findings. According to the charts, the optimized versions of all the models were more accurate than the baseline versions (**Figure 7**, **Figure 8**).

As an illustration, in **Figure 9**, the comparison chart of Decision Tree models demonstrated that the PSO-optimized model attained an accuracy level of 92.20%, whereas the baseline model reached an accuracy of 89.52%.

Together, these results suggest that the PSO algorithm effectively adjusts parameters of machine learning models to improve their ability to predict COVID-19 risk. These findings hold important implications for shaping accurate and effective public health interventions and policies amidst the global endeavor to combat the COVID-19 pandemic (**Figures 10-12**).

An additional metric, the Area Under Curve (AUC), was calculated to provide further validation of the model's performance beyond the measured accuracy score. The AUC serves as a complementary measure, affirming the model's high-quality classification capabilities, particularly following the optimization of hyperparameters. **Figures 13-15** shows Receiver Operating Characteristic (ROC) Curve and the AUC metric value for KNN, SVM and MLP optimized models respectively, reinforcing robust classification performance.

These findings collectively imply that the PSO algorithm is capable of efficiently fine-tuning machine learning model parameters to enhance the models' performance to forecast COVID-19 risk. These findings have significant ramifications for formulating precise and successful public health interventions and policies in the course of the ongoing international effort to combat the COVID-19 pandemic.

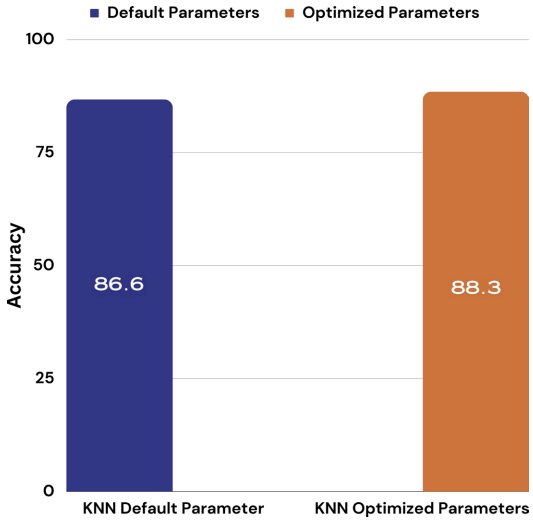


Figure 7. Accuracy of optimized KNN and its baseline counterpart.

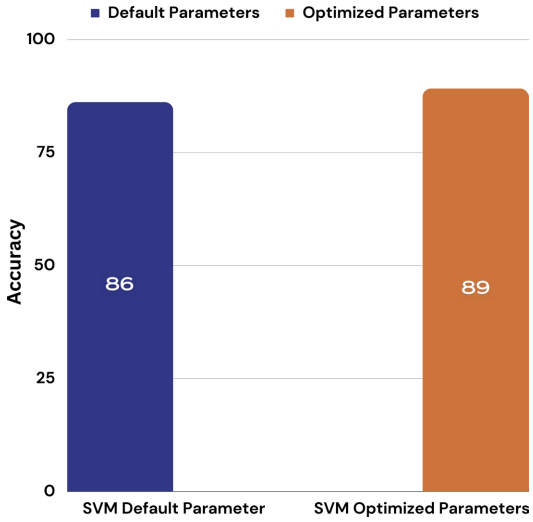


Figure 8. Accuracy of optimized SVM and its baseline counterpart.

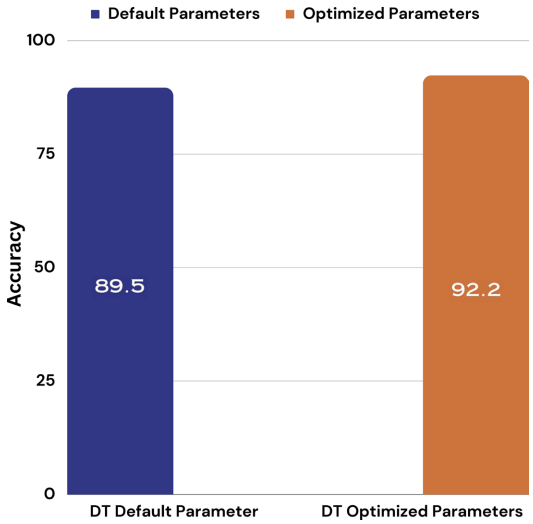


Figure 9. Accuracy of optimized Decision Tree and its baseline counterpart.

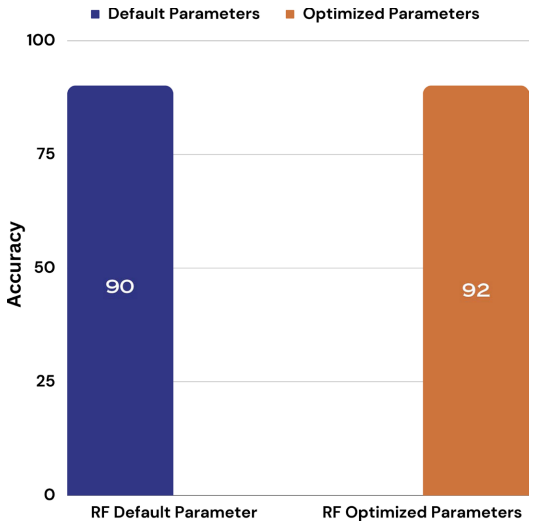


Figure 10. Accuracy of optimized Random Forest and its baseline counterpart.

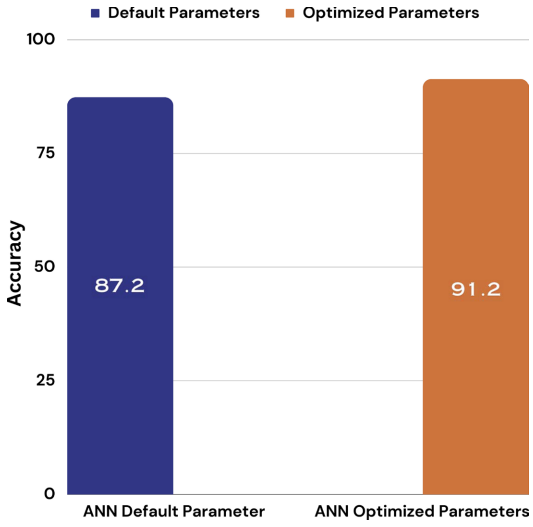


Figure 11. Accuracy of optimized ANN and its baseline counterpart.

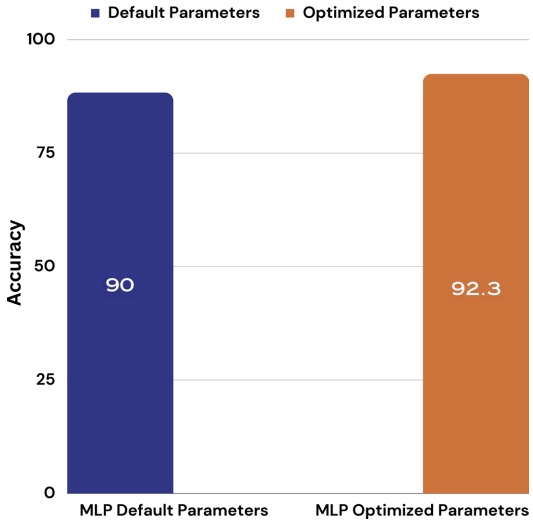


Figure 12. Accuracy of optimized MLP and its baseline counterpart.

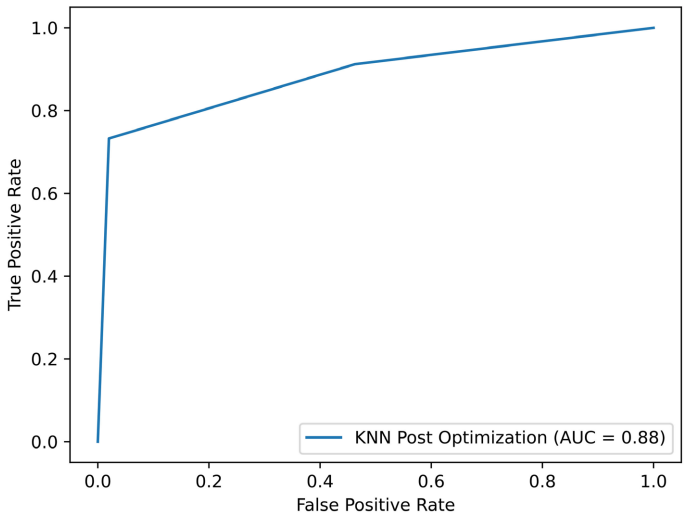


Figure 13. Area Under Curve optimized KNN and its baseline counterpart.

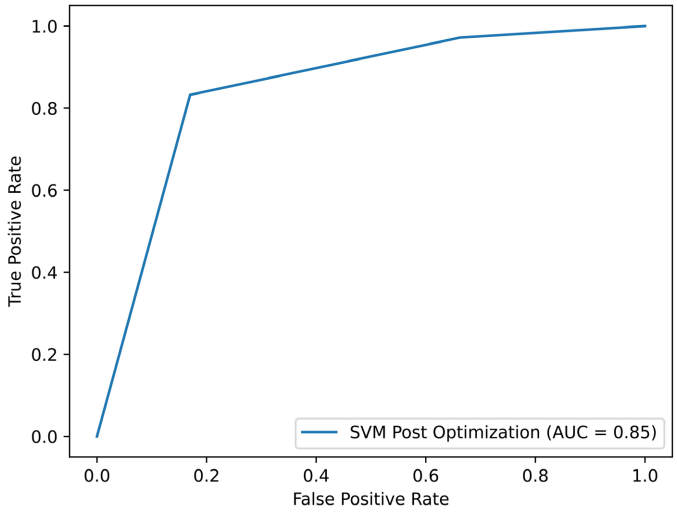


Figure 14. Accuracy of optimized SVM and its baseline counterpart.

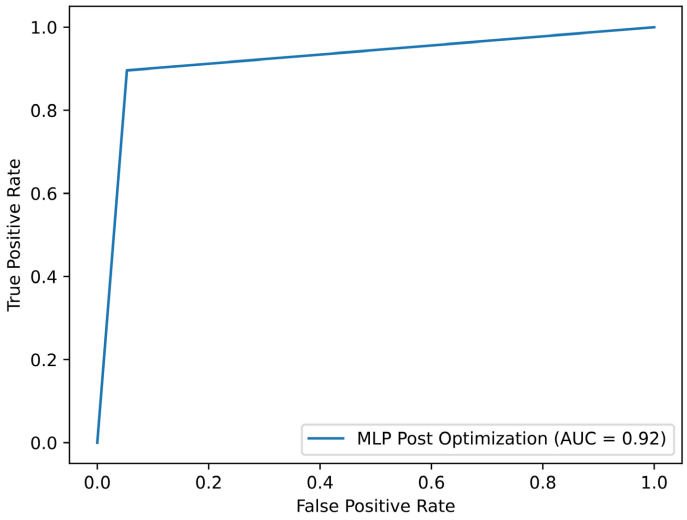


Figure 15. Accuracy of optimized MLP and its baseline counterpart.

5. Conclusion and Future Work

In conclusion, the application of the PSO algorithm for optimizing the hyperparameters of six machine learning models-Random Forest, Decision Tree, SVM, KNN, MLP, and Artificial Neural Networks-demonstrated enhanced accuracy in analyzing and predicting COVID-19 risk, surpassing that of baseline models. This underscores the effectiveness of the PSO algorithm in refining machine learning models for the prognostication of COVID-19 risk, offering potential utility in shaping public health policies and interventions. The study's findings illuminate the promise of leveraging optimized machine learning models for precise COVID-19 risk prediction, contributing valuable perspectives to the global efforts aimed at mitigating the pandemic. However, it is imperative to acknowledge a limitation stemming from the constrained computational resources, which restricted the number of trials for the PSO algorithm. This limitation highlights the need for further investigation with expanded computational capacity to explore the full spectrum of PSO outcomes and potentially unveil more profound insights. Future endeavors in this domain may include extending the scope of variables, such as incorporating healthcare system capacity, to augment model accuracy. This expansion necessitates comprehensive data collection and preprocessing, along with the meticulous selection of pertinent variables. Additionally, validating the models with independent datasets across diverse geographic regions and populations will be crucial to affirm their generalizability and robustness.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Xiang, Y., Jia, Y., Chen, L., Guo, L., Shu, B. and Long, E. (2021) Covid-19 Epidemic Prediction and the Impact of Public Health Interventions: A Review of Covid-19 Epidemic Models. *Infectious Disease Modelling*, **6**, 324-342. https://doi.org/10.1016/j.idm.2021.01.001

[2] Mahesh, B. (2020) Machine Learning Algorithms—A Review. *International Journal of Science and Research*, **9**, 381-386.

[3] Alyasseri, Z.A.A., Al-Betar, M.A., Doush, I.A., Awadallah, M.A., Abasi, A.K., Makhadmeh, S.N., Alomari, O.A., Abdulkareem, K.H., Adam, A., Damasevicius, R., *et al.* (2022) Review on Covid-19 Diagnosis Models Based on Machine Learning and Deep Learning Approaches. *Expert Systems*, **39**, E12759. https://doi.org/10.1111/exsy.12759

[4] Jain, G., Mittal, D., Thakur, D. and Mittal, M.K. (2020) A Deep Learning Approach to Detect Covid-19 Coronavirus with X-Ray Images. *Biocybernetics and Biomedical Engineering*, **40**, 1391-1405. https://doi.org/10.1016/j.bbe.2020.08.008

[5] Muhammad, L., Algehyne, E.A., Usman, S.S., Ahmad, A., Chakraborty, C. and Mohammed, I.A. (2021) Supervised Machine Learning Models for Prediction of Covid-19 Infection Using Epidemiology Dataset. *SN Computer Science*, **2**, Article No.

11. https://doi.org/10.1007/s42979-020-00394-7

[6] Jian, S.-W., Cheng, H.-Y., Huang, X.-T. and Liu, D.-P. (2020) Contact Tracing with Digital Assistance in Taiwan's Covid-19 Outbreak Response. *International Journal of Infectious Diseases*, **101**, 348-352. https://doi.org/10.1016/j.ijid.2020.09.1483

[7] Hertel, L., Collado, J., Sadowski, P., Ott, J. and Baldi, P. (2020) Sherpa: Robust Hyperparameter Optimization for Machine Learning. *SoftwareX*, **12**, Article ID: 100591. https://doi.org/10.1016/j.softx.2020.100591

[8] Khalid, R. and Javaid, N. (2020) A Survey on Hyperparameters Optimization Algorithms of Forecasting Models in Smart Grid. *Sustainable Cities and Society*, **61**, Article ID: 102275. https://doi.org/10.1016/j.scs.2020.102275

[9] Alibrahim, H. and Ludwig, S.A. (2021) Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization. 2021 *IEEE Congress on Evolutionary Computation* (*CEC*), Kraków, 28 June-1 July 2021, 1551-1559. https://doi.org/10.1109/CEC45853.2021.9504761

[10] Li, L. and Talwalkar, A. (2020) Random Search and Reproducibility for Neural Architecture Search. *Proceedings of the* 35*th Uncertainty in Artificial Intelligence Conference*, Vol. 115, 367-377.

[11] Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z. and Guyon, I. (2021) Bayesian Optimization Is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. *NeurIPS* 2020 *Competition and Demonstration Track*, Vancouver, 6-12 December 2020, 3-26.

[12] Bakhteev, O.Y. and Strijov, V.V. (2020) Comprehensive Analysis of Gradient-Based Hyperparameter Optimization Algorithms. *Annals of Operations Research*, **289**, 51-65. https://doi.org/10.1007/s10479-019-03286-z

[13] Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I. and Tuba, M. (2020) Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics. *Algorithms*, **13**, Article No. 67. https://doi.org/10.3390/a13030067

[14] Byla, E. and Pang, W. (2020) Deepswarm: Optimising Convolutional Neural Networks Using Swarm Intelligence. *The* 19*th UK Workshop on Computational Intelligence*, Portsmouth, 4-6 September 2019, 119-130. https://doi.org/10.1007/978-3-030-29933-0_10

[15] Ouyang, B., Song, Y., Li, Y., Sant, G. and Bauchy, M. (2021) Ebod: An Ensemble-Based Outlier Detection Algorithm for Noisy Datasets. *Knowledge-Based Systems*, **231**, Article ID: 107400. https://doi.org/10.1016/j.knosys.2021.107400

[16] Jin, C., Chen, W., Cao, Y., Xu, Z., Tan, Z., Zhang, X., Deng, L., Zheng, C., Zhou, J., Shi, H., *et al.* (2020) Development and Evaluation of an Artificial Intelligence System for Covid-19 Diagnosis. *Nature Communications*, **11**, Article No. 5088. https://doi.org/10.1038/s41467-020-18685-1

[17] Estiri, H., Strasser, Z.H., Klann, J.G., Naseri, P., Wagholikar, K.B. and Murphy, S.N. (2021) Predicting Covid-19 Mortality with Electronic Medical Records. *NPJ Digital Medicine*, **4**, Article No. 15. https://doi.org/10.1038/s41746-021-00383-x

[18] Ni, Q., Sun, Z.Y., Qi, L., Chen, W., Yang, Y., Wang, L., Zhang, X., Yang, L., Fang, Y., Xing, Z., *et al.* (2020) A Deep Learning Approach to Characterize 2019 Coronavirus Disease (Covid-19) Pneumonia in Chest CT Images. *European Radiology*, **30**, 6517-6527. https://doi.org/10.1007/s00330-020-07044-9

[19] El-Behery, H., Attia, A.-F., El-Fishawy, N. and Torkey, H. (2021) Efficient Machine Learning Model for Predicting Drug-Target Interactions with Case Study for Co-

vid-19. *Computational Biology and Chemistry*, **93**, Article ID: 107536. https://doi.org/10.1016/j.compbiolchem.2021.107536

[20] Buvana, M. and Muthumayil, K. (2021) Prediction of Covid-19 Patient Using Supervised Machine Learning Algorithm. *Sains Malaysiana*, **50**, 2479-2497. https://doi.org/10.17576/jsm-2021-5008-28

[21] Wehbe, R.M., Sheng, J., Dutta, S., Chai, S., Dravid, A., Barutcu, S., Wu, Y., Cantrell, D.R., Xiao, N., Allen, B.D., *et al.* (2021) Deepcovid-Xr: An Artificial Intelligence Algorithm to Detect Covid-19 on Chest Radiographs Trained and Tested on a Large Us Clinical Data Set. *Radiology*, **299**, E167-E176. https://doi.org/10.1148/radiol.2020203511

[22] Wang, L., Lin, Z.Q. and Wong, A. (2020) Covid-Net: A Tailored Deep Convolutional Neural Network Design for Detection of Covid-19 Cases from Chest X-Ray Images. *Scientific Reports*, **10**, Article No. 19549. https://doi.org/10.1038/s41598-020-76550-z

[23] Kumar, R.L., Khan, F., Din, S., Band, S.S., Mosavi, A. and Ibeke, E. (2021) Recurrent Neural Network and Reinforcement Learning Model for Covid-19 Prediction. *Frontiers in Public Health*, **9**, Article ID: 744100. https://doi.org/10.3389/fpubh.2021.744100

[24] Rustam, F., Reshi, A.A., Mehmood, A., Ullah, S., On, B.-W., Aslam, W. and Choi, G.S. (2020) Covid-19 Future Forecasting Using Supervised Machine Learning Models. *IEEE Access*, **8**, 101489-101499. https://doi.org/10.1109/ACCESS.2020.2997311

[25] Saha, S., Saha, A., Roy, B., Sarkar, R., Bhardwaj, D. and Kundu, B. (2022) Integrating the Particle Swarm Optimization (Pso) with Machine Learning Methods for Improving the Accuracy of the Landslide Susceptibility Model. *Earth Science Informatics*, **15**, 2637-2662. https://doi.org/10.1007/s12145-022-00878-5

[26] Khourdifi, Y. and Baha, M. (2019) Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization. *International Journal of Intelligent Engineering & Systems*, **12**, 242-252. https://doi.org/10.22266/ijies2019.0228.24

[27] Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A. and Benhaddou, D. (2017) Parameters Optimization of Deep Learning Models Using Particle Swarm Optimization. 2017 13*th International Wireless Communications and Mobile Computing Conference* (*IWCMC*), Valencia, 26-30 June 2017, 1285-1290. https://doi.org/10.1109/IWCMC.2017.7986470

[28] Guo, Y., Li, J.-Y. and Zhan, Z.-H. (2020) Efficient Hyperparameter Optimization for Convolution Neural Networks in Deep Learning: A Distributed Particle Swarm Optimization Approach. *Cybernetics and Systems*, **52**, 36-57. https://doi.org/10.1080/01969722.2020.1827797

[29] Kalita, D.J., Singh, V.P. and Kumar, V. (2020) SVM Hyper-Parameters Optimization Using Multi-Pso for Intrusion Detection. In: Shukla, R.K., *et al.*, Eds., *Social Networking and Computational Intelligence*, Springer, Berlin, 227-241. https://doi.org/10.1007/978-981-15-2071-6_19

[30] Wang, Y.-Q., Li, J.-Y., Chen, C.-H., Zhang, J. and Zhan, Z.-H. (2023) Scale Adaptive Fitness Evaluation-Based Particle Swarm Optimisation for Hyperparameter and Architecture Optimisation in Neural Networks and Deep Learning. *CAAI Transactions on Intelligence Technology*, **8**, 849-862. https://doi.org/10.1049/cit2.12106

[31] Rajkumar, S. (2020) The Dataset Novel Corona Virus 2019 Dataset. https://www.kaggle.com/datasets/meirnizri/covid19-dataset

[32] Chawla, N., Bowyer, K., Hall, L.O. and Kegelmeyer, W.P. (2002) Smote: Synthetic

Minority Over-Sampling Technique.
https://api.semanticscholar.org/corpusid:1554582
https://doi.org/10.1613/jair.953

[33] Tsai, C.-W., Hsia, C.-H., Yang, S.-J., Liu, S.-J. and Fang, Z.-Y. (2020) Optimizing Hyperparameters of Deep Learning in Predicting Bus Passengers Based on Simulated Annealing. *Applied Soft Computing*, **88**, Article ID: 106068.
https://doi.org/10.1016/j.asoc.2020.106068

[34] Yang, L. and 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

[35] DeCastro-Garcia, N., Munoz Castaneda, A.L., Escudero Garcia, D. and Carriegos, M.V. (2019) Effect of the Sampling of A Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm. *Complexity*, **2019**, Article ID: 6278908. https://doi.org/10.1155/2019/6278908

[36] Elshawi, R., Maher, M. and Sakr, S. (2019) Automated Machine Learning: State-of-the-Art and Open Challenges.

[37] Wang, D., Tan, D. and Liu, L. (2018) Particle Swarm Optimization Algorithm: An Overview. *Soft Computing*, **22**, 387-408. https://doi.org/10.1007/s00500-016-2474-6