Dissertation on

Whale Optimization-based Machine Learning Algorithm for Mortality Prediction in Sepsis Patients

Thesis submitted towards partial fulfilment
of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by

SRIJITA MUKHOPADHYAY

EXAMINATION ROLL NO. M4CWE24003
UNIVERSITY REGISTRATION NO. 163772 of 2022-23

Under the guidance of
Dr. SASWATI MUKHERJEE

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India
2024

M.Tech in IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Whale Optimization-based Machine Learning Algorithm for Mortality Prediction in Sepsis Patients"** is a bona fide work carried out by **SRIJITA MUKHOPADHYAY** under our supervision and guidance for partial fulfilment of the requirements for the degree of **Master of Technology in IT (Courseware Engineering)** in **School of Education Technology**, during the academic session 2023-2024.

- -

Dr. SASWATI MUKHERJEE
SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

- -

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

- -

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

M.Tech in IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL**

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approves the thesis only for the purpose for which it has been submitted.

Committee of final examination
for evaluation of the Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME	:	SRIJITA MUKHOPADHYAY
EXAMINATION ROLL NUMBER	:	M4CWE24003
REGISTRATION NUMBER	:	163772 of 2022-23
THESIS TITLE	:	WHALE OPTIMIZATION-BASED MACHINE LEARNING ALGORITHM FOR MORTALITY PREDICTION IN SEPSIS PATIENTS

SIGNATURE: DATE:

Acknowledgment

I feel fortunate while presenting this dissertation at the **School of Education Technology, Jadavpur University, Kolkata**, in the partial fulfilment of the requirement for the degree of **Master of Technology in IT (Courseware Engineering).**

I hereby take this opportunity to show my gratitude to my mentor, **Dr Saswati Mukherjee,** who has guided and helped me with all possible suggestions, support, aspiring advice, and constructive criticism, along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Prof. Dr. Matangini Chattopadhyay**, **Director of the School of Education Technology,** for her timely encouragement, support, and advice. I would also like to thank **Mr Joydeep Mukherjee** for his constant support during my entire course of work. My thanks and appreciation go to my classmates from M. Tech in Information Technology (Courseware Engineering) and Master in Multimedia Development.

I wish to thank all the departmental support staff and everyone else who has contributed to this dissertation.

Finally, I would like to express my special gratitude to my parents, who have invariably sacrificed and supported me and helped me achieve this height.

Date:
Place: Kolkata

Srijita Mukhopadhyay
Examination Roll Number: M4CWE24003
M. Tech in IT (Courseware Engineering)
School of Education Technology
Jadavpur University
Kolkata:700032

Contents

Executive Summary

Sepsis is a life-threatening medical emergency caused by the body's extreme response to an infection, leading to widespread inflammation and severe organ damage. Infections, triggering sepsis, typically originate from the gastrointestinal tract, lungs, skin, or urinary tract. The risk of death from sepsis is as high as 30%, increasing to 50% for severe sepsis and up to 80% for septic shock. An extensive review of the computerised techniques for the prediction of mortality of sepsis patients has been performed to associate the work done so far in this field. While performing the studies, it became evident that since computerised analysis of sepsis mortality prediction is a relatively emerging field, related literature and research work have been limited. There is also a lack of a standardised public database of sepsis patients.

This dissertation report aims to develop a classification model for predicting the mortality of sepsis patients using a machine learning algorithm. LightGBM is considered as the prediction model whose parameters are tuned by the whale optimization algorithm. The proposed prediction method helps to take prompt decisions on whether a sepsis patient admitted to the Intensive Care Unit will survive or not.

In this approach, the samples of sepsis patients are collected from the AMRI hospital of Kolkata. The dataset is annotated by a domain expert which helps in the pre-processing of the data. A machine learning model is utilised to predict the mortality of sepsis patients. The experimental results demonstrate that the proposed predictive model has accurately classified the patients based on mortality.

1. Introduction

1.1 Overview

Sepsis, as per the Sepsis-3 criteria, denotes a dysregulated systemic inflammatory response syndrome triggered by infection, leading to severe organ dysfunction and an increased risk of mortality. Organ dysfunction can be identified as an acute change in total Sequential Organ Failure Assessment (SOFA) score ≥2 points consequent to the infection. Patients afflicted with sepsis face a markedly elevated probability of death [1]. In the United States, approximately 10% of patients admitted to the Intensive Care Unit (ICU) are diagnosed with sepsis, and a significant proportion, roughly 25% of ICU beds are occupied by sepsis patients. Considering the elevated mortality rate among sepsis patients in the ICU, it becomes imperative to ascertain the risk of in-hospital mortality as early as possible [2]. Due to the heightened mortality rates among sepsis patients in the ICU, it is crucial to promptly identify those at risk of in-hospital death. Early and precise detection of sepsis patients with a high likelihood of in-hospital mortality enables ICU physicians to make informed clinical decisions, potentially enhancing patient outcomes [3].

In 2017, there were an estimated 48.9 million cases of sepsis globally, resulting in approximately 11 million sepsis-related deaths. Developed countries reported an incidence of severe sepsis at 4.00 cases per 1,000 population, while in China, the incidence of sepsis in intensive care units stood at 20.6%. The cost of treating sepsis in the United States surpassed $20 billion in 2009, constituting 5.2% of total clinical expenses in U.S. hospitals, which poses a significant financial burden on patients and healthcare systems [4]. In the United States, the mortality rate for severe sepsis in intensive care units ranges from 21.0% to 36.7%, while the hospital mortality rate varies from 22.0% to 44.5%, depending on factors such as patient population and the definition of severe sepsis [5].

In recent studies, novel machine learning techniques have showcased enhanced predictive capabilities compared to traditional prediction methods [6], thereby garnering significant research attention due to the vast accumulation of big data and advancements in data storage techniques [7]. Notably, various innovative and pragmatic machine-learning approaches have been proposed, exhibiting promising prediction performance in medicine, including developing machine learning-based models for ICU mortality prediction [8].

1.2 Problem Statement

Whale optimization-based machine learning algorithm for mortality prediction in sepsis patients.

1.3 Objective

The objectives are:

a) To develop a whale optimization-based machine learning model for mortality prediction of sepsis patients in the ICU.
b) To identify and select the most relevant features contributing to the patient’s mortality.
c) To optimize the hyperparameters of the machine learning model using whale optimization.
d) To identify the features contributing to the patient’s mortality using statistical analysis.

2. Background Concepts

New models have been developed to better predict the likelihood of in-hospital death among ICU patients with sepsis, as existing severity scores are inadequate. Machine learning can process complex, non-linear data to extract information and provide insights that assist clinical decision-making. This unique technique has been applied in various medical fields to create robust risk models and to improve prediction accuracy [8-10].

2.1 Whale Optimization Algorithm (WOA)

The hunting behaviours of humpback whales are based on the idea of creating a bubble net to encircle and trap their prey. There are three main ways that humpback whales hunt: by encircling their prey, searching for prey, and using the bubble-net attacking mechanism.

Encircling the prey: Humpback whales often work together in groups to encircle their prey, creating a barrier or circle around it to prevent escape. Similarly, in WOA, candidate solutions are iteratively adjusted and refined to encircle the optimal solution.

$$\vec{D} = |\vec{C} \, . \, \vec{X}^*(\text{t}) - \vec{X}(\text{t}) \qquad (1)$$

$$\vec{X}(\text{t} + 1) = \vec{X}^*(\text{t}) - \vec{A}.\vec{D} \qquad (2)$$

$\vec{X}^*$(t) denotes the best search agent of the swarm at iteration t, $\vec{X}$(t) represents a candidate search agent of the swarm, $\vec{D}$ denotes the distance between two agents as shown in Equations (1) and (2).

$$\vec{A} = 2. \, \vec{a} \, . \, \vec{r} - \vec{a} \qquad (3)$$

$$\vec{C} \ = \ 2 \, . \vec{r} \qquad (4)$$

where $\vec{a}$ is a vector of 2 that is linearly decreased to 0 over iterations, and $\vec{r}$ is a vector of randomly generated numbers in [0,1] based on Gaussian distribution as given in Equations (3) and (4).

Searching for prey: Before encircling the prey, humpback whales search for it in the surrounding environment. They use various sensory cues to detect the presence of prey. In WOA, this searching behaviour is reflected in the exploration phase of the algorithm, where candidate solutions explore the solution space to find regions with potentially better solutions as shown in Equations (5) and (6).

$$\vec{D} = |\overrightarrow{C} \, . \, \vec{X}\text{rand(t)} - \vec{X}(\text{t})| \qquad (5)$$

$$\vec{X}(t + 1) = \vec{X}rand(t) - \vec{A} . \vec{D} \qquad (6)$$

Bubble-net attacking mechanism: The bubble-net technique involves humpback whales blowing bubbles in a circular pattern underwater to create a barrier, driving the prey towards the surface where they can easily catch it. This mechanism is analogous to the exploitation phase of WOA, where candidate solutions converge towards the optimal solution identified during the exploration phase.

$$\vec{D}' = |\vec{X}^*(t) - \vec{X}(t)| \qquad (7)$$

$$\vec{X}(t + 1) = \vec{D}'. e^{bl} . \cos(2\pi l) + \vec{X}^*(t) \qquad (8)$$

where b is a constant for deciding the scale of spiral, and l is a random number uniformly distributed in the interval [-1,1] as shown in Equations (7) and (8).

Algorithm

```
01: Initialise the swarm $\vec{X}_i$ ( i = 1, 2, …, n)
02: Evaluate the fitness value of all incumbents
03: $\overrightarrow{X*} = \overrightarrow{X*(t)}$(the best incumbent)
04: **While** the termination criterion is not met **do**
05:     **For** each incumbent **do**
06:         Update a,$\vec{A}$,$\vec{C}$, l and p
07:         **If** (p <0.5)
08:             **If** (|$\vec{A}$| < 1)
09:                 $\vec{D} = |\vec{C} . \vec{X}^*(t) - \vec{X}(t); \vec{X}(t + 1) = \vec{X}^*(t) - \vec{A}.\vec{D}$
10:             **Else**
11:                 $\vec{D} = |\overrightarrow{C} . \vec{X}rand(t) - \vec{X}(t)|; \vec{X}(t + 1) = \vec{X}rand(t) - \vec{A} . \vec{D}$
12:         **Else**
13:             $\vec{D}' = |\vec{X}^*(t) - \vec{X}(t)|; \vec{X}(t + 1) = \vec{D}'. e^{bl} . \cos(2\pi l) + \vec{X}^*(t)$
14:     **End For**
15:     Revise the incumbent if it exceeds the boundary
16:     Calculate the fitness value for all incumbents
17:     Update the best incumbent $\overrightarrow{X*(t)}$if there is any promotion
18: **End While**
19: **return** $\overrightarrow{X*(t)}$
```

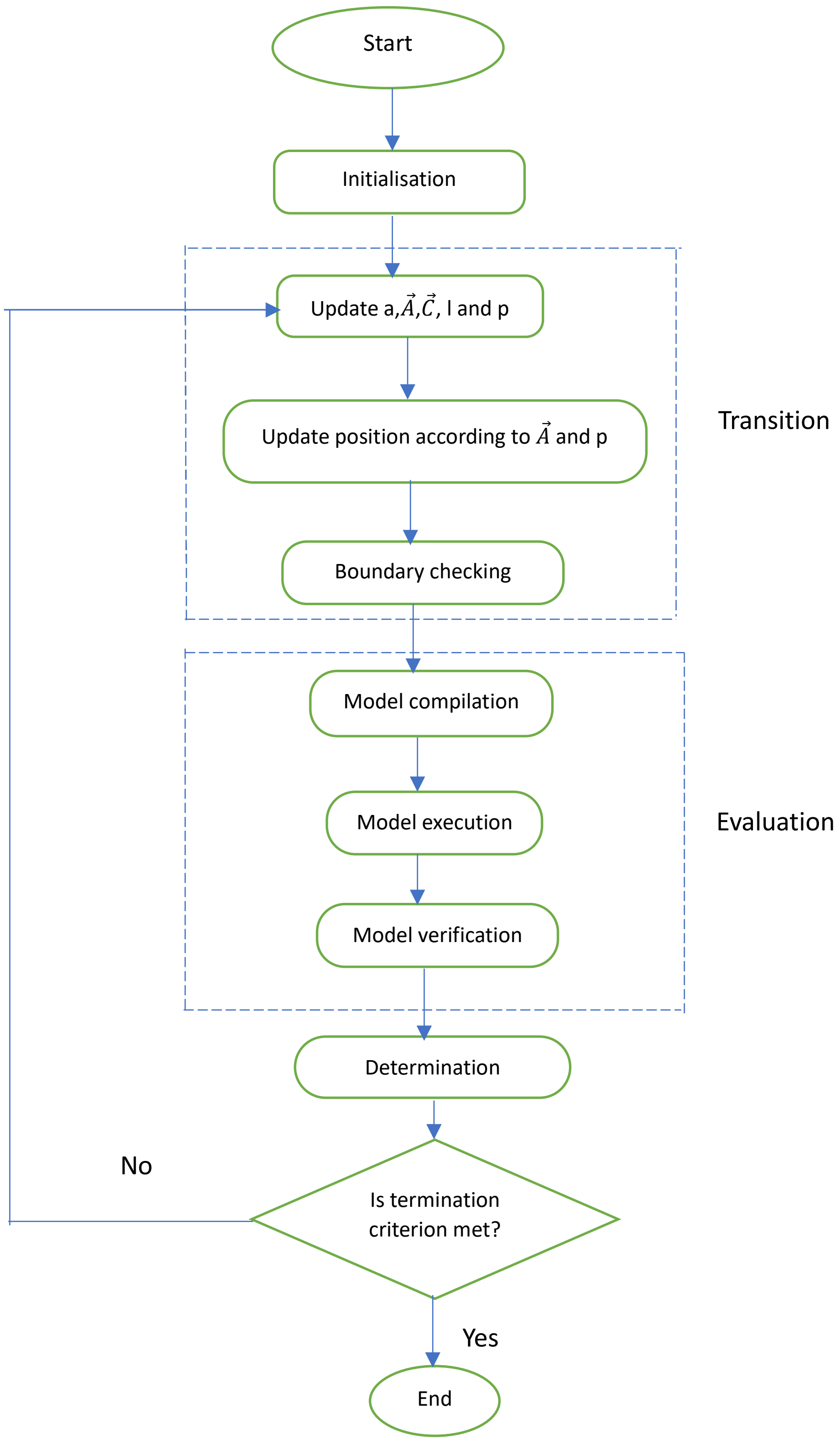


Figure 1: Flowchart of WOA for hyperparameter optimization in LightGBM [11]

2.2 WOA for Hyperparameter Optimization

Initialisation

A collection of hyperparameters is used to construct each search agent randomly based on a uniform distribution. These configurations can be applied to the decision-making process of machine learning algorithms. Every search agent can be considered a WOA solution, with each sub-solution representing a hyperparameter. After encoding is complete, WOA will select the best search agent (X*) from the newly generated search agents by assessing each swarm member in the context of the intended training model. Figure 1 shows the flowchart of the WOA for LightGBM hyperparameter optimization.

Transition

WOA creates this process. At the start of each cycle, WOA randomly creates distinct coefficients for each search agent. This process helps prevent the early convergence issue by diversifying the moving path. The new position is then adjusted in accordance with vector $\vec{A}$, and probability p. Search agents have more opportunities to refer to the prey as each iteration progresses. More precisely, there is an eventual increase in the probability of choosing the optimal search agent. However, as a result of parameter $\vec{a}$ being reduced from 2 to 0, there is no longer any possibility that $\vec{A}$ will increase to more than 1 during the second half of the total number of iterations. As a result, at the later stage, the prey-searching mechanism (i.e. Eqs. (5) and (6)) will be completely removed. Out-of-order solutions cannot exist throughout the search process; once the new position is established, the boundary requirements must be met if the search agent exceeds the allotted amount.

Evaluation

Evaluation involves assessing the outcomes using a machine learning model. Every search agent has a hyperparameter configuration associated with it. These hyperparameter values, verified by testing current WOA candidates, comprise the solutions.

Determination
It uses feedback from the training model to determine performance. The best search agent is replaced by improved configurations, updating positions until termination requirements are achieved [11].

2.3 Gradient Boosting Machine Model

Gradient Boost Algorithm (GBM) is a machine learning technique influenced by learning theory. It employs a series of decision trees and constructs trees in order, reducing the residual errors of earlier trees and fixing previous issues. To get the most homogeneous responses to the predictors, the GBM framework algorithm partitions the input variables to minimise the loss function. GBM constantly shows better prediction accuracy than other machine learning models, such as Random Forests and support vector networks. Their effectiveness is due to their repetitive feature, in which each repetition expands based on the knowledge of the preceding learner. GBM models handle a variety of data formats with minimal effort for data preprocessing.

Weak Learners
Gradient boosting builds an ensemble of weak learners, typically decision trees with limited depth. These weak learners are simple models that perform slightly better than random guessing.

Residuals
Initially, the model makes a prediction on the training data. The difference between the actual outcomes and the predicted outcomes is known as the residuals.

Iterative Fitting
In each iteration, a new weak learner is trained to predict the residuals (errors) from the previous model. By focusing on these residuals, the new learner aims to correct the errors made by the previous models.

Model Update
The predictions from the new weak learner are added to the overall model's prediction. This process of updating the model by adding new learners is repeated for a predefined number of iterations or until the performance improvement occurs.

Weighted Sum of Learners
The final model is a weighted sum of all the weak learners, where each learner corrects the errors of the preceding ones. This aggregation of models results in a robust prediction system [12].

3. Literature Survey

Su et al. [13] presented a study on three machine learning models for predicting sepsis patient mortality in the ICU. The performance of three machine learning models, namely logistic regression, random forest, and XGBoost, were compared. The Least Absolute Shrinkage and Selection Operator (LASSO) was applied for feature selection. The training and test sets were divided randomly by 70% and 30%. The best performance was observed for random forest, which resulted in an AUC of 0.77.

Guo et al. [14] performed a retrospective study on the prediction of 30-day mortality on sepsis using Artificial Neural Network (ANN). Three layers comprised the fundamental architecture of the Artificial Neural Network: the input layer, the hidden layer, and the output layer. The input layer comprised 12 nodes, the hidden layer comprised 6 nodes, and the output layer had 2 nodes. An oversampling algorithm was applied to deal with the imbalance between training and validation sets. The model achieved an AUC of 0.873 on the training and 0.811 on the validation set.

Hou et al. [15] established a 30-day mortality prediction model based on real-world data for patients with sepsis 3.0. Three predictive models namely eXtreme Gradient Boosting (XGBoost), conventional logistic regression, and SAPS-II score prediction models, were used. The performance was compared using AUCs receiver operating curve and decision curve. The XGBoost model achieved the best result with an AUC of 0.857.

Peng et al. [16] developed and validated 9 different machine-learning models to predict 30-day mortality in patients with encephalopathy associated with sepsis. In this approach, the top 15 features were selected using the Recursive Feature Elimination (RFE) method. Then 9 models were employed, namely artificial neural network (NNET), Naïve Bayes (NB), Logistic Regression (LR), Gradient Boosting Machine (GBM), Adaptation boosting (AdaBoost), Random Forest (RF), bagged trees (BT), XGBoost and CatBoost. For internal validation, a bootstrap resampling technique was utilised with 100 iterations. The NNET, LR, and AdaBoost models performed well in calibration, as seen by their high prediction accuracy with p-values of 0.831, 0.119, and 0.129, respectively. Adapting boosting got an AUC of 0.834 in the test set.

Taylor et al. [17] compared three machine learning models, namely the Random Forest model, the Classification and Regression Tree (CART) model, and the logistic regression model, to predict in-hospital mortality of sepsis patients in the emergency department (ED). All models predicted in-hospital mortality with an AUC greater than 0.69. The random forest model achieved an AUC of 0.86.

In the field of sepsis research, Zhang et al. [18] utilised the LASSO method to develop a tool for predicting the mortality risk of sepsis patients using the Medical Information Mart for Intensive Care (MIMIC) III dataset. Their findings demonstrated that the LASSO-based prediction model outperformed the SOFA score regarding discrimination by achieving an AUC of 0.772.

Zamani et al. [19] employed four swarm intelligence algorithms, namely Particle Swarm Optimization (PSO), Imperialist Competitive Algorithm (ICA), Firefly Algorithm (FA), and Invasive Weed Optimization (IWO), for breast cancer diagnosis. The datasets were divided into two sets, a training set and a test set, of 70% and 30%, respectively, at random. The performance of each algorithm was assessed using the multi-layer perceptron (MLP) network. The Kolmogorov theorem was used in the MLP network to calculate the number of hidden nodes, and the number of inputs is set with the number of features. For all the algorithms, the maximum iteration was fixed at 15, and the initial population size was set to 30. It was observed that the Firefly Algorithm showed the best performance with an accuracy of 98.54%.

In diagnosing type II diabetes [20], the feature selection was done using Particle Swarm Optimization. Meta-heuristic algorithms, Grid Search, and a Genetic algorithm optimized the hyperparameters. The machine learning models Decision Tree, KNN, Multi-layer Perceptron (MLP), and Support Vector Machine (SVM) were employed to calculate the results. GA-SVM achieved the best result with an AUC of 0.93.

In [21], the Whale Optimization Algorithm (WOA) and SVM were used in 20 images to develop an intelligent lung tumour diagnosis system. The WOA in this study helped to select the best feature subsets. Then, the accuracies of SVM with kernels, namely linear, Polynomial, and Radial Based Function (RBF), were compared. RBF SVM provided the best accuracy of 95%.

4. Proposed Approach

The work aims to create a predictive model for assessing the mortality of sepsis patients in the ICU. LightGBM, a machine learning model is used to predict the survival or mortality of sepsis patients and fine-tune the hyperparameters using the whale optimization algorithm.

The dataset of sepsis patients collected from the primary care hospital in Kolkata has been annotated by a domain expert while maintaining confidentiality and anonymity. To improve the performance of machine learning models, data augmentation has been performed on the dataset. Subsequently, feature selection using Analysis of Variance (ANOVA) has been carried out on this augmented dataset. The selected features are then used for model training. Before training the model, hyperparameter tuning is conducted using a whale optimization algorithm, which involves using an objective function to evaluate the performance of the model with different sets of hyperparameters. The search agents (whales) are initially initialised with random values within the specified hyperparameter ranges. Each agent's performance is evaluated using an objective function that trains a LightGBM model with the agent's hyperparameters and provides classification scores and performance measures. The agent with the highest accuracy is identified as the best agent.

4.1 Dataset Description

The dataset used in this study is collected from a primary care hospital in Kolkata. To protect patient confidentiality, all patient identities are removed. The database consists of 503 patient records with 105 features, including 30 clinical features, one target feature, and 104 independent features. Of the 503 patients, three-quarters are non-survivors, with 178 belonging to the survivor category.

4.2 Data Pre-processing

The following steps are performed to pre-process the data.

Handling of Missing Values

It is very common to encounter many missing values in a medical dataset, and this dataset is no exception. In this case, the fields with missing values are set to 0 because the domain expert had no information about them. Setting these values to 0 helps eliminate biases that could have occurred if we had considered 1 or 2 instead. This could have affected the model analysis due to increased Type I and II errors.

Handling of Categorical Features

In machine learning, when training and validating a model, categorical features need to be converted to numerical features. This is achieved through a process called one-hot encoding, which transforms the categorical features into binary numerical variables. In the dataset, out of 104 independent columns, 76 are categorical, which are replaced with 91 one-hot encoded columns. With one-hot encoding, each categorical feature is represented in different binary columns based on the number of categories, making them easier for machine learning algorithms to interpret.

Data Augmentation

Due to the small size of the dataset, the number of patient records is increased after removing the missing values and converting the categorical data. To address this, Gretel.ai, a synthetic data-generating tool, creates 5000 synthetic data points based on the original 503 patients' data. The clinical features in the augmented dataset align with the reference range of the original clinical values. It is important to note that machine learning models may not yield accurate results when working with small sample datasets.

Removing the Outliers

After data augmentation, the dataset contains some outliers, which are removed by deleting those records. After removing the outliers, the total number of datapoints is 3650 with 195 features.

Feature Selection

40 features are selected out of 195 using statistical analysis tests, such as ANOVA, to select the most relevant features that directly contribute to the final output. The features, along with the p-values, are shown in Table 1.

Table 1: Significance analysis of clinical features using ANOVA test

Feature	p-value
1. Cumulative_fluid_1_0	7.788e^(-97)
2. Cumulative_fluid_2_0	5.023e^(-78)
3. Loading_dose_1_0	3.24e^(-76)
4. Nutrition_started_1_0	4.381e^(-73)
5. Systolic_BP_lesser_than_90_2_0	1.019e^(-71)
6. Rel_Support_2_0	1.235e^(-71)
7. Was_sedation_scale_2_0	2.232e^(-70)
8. Infection__PrimaryReason_ICU_Admission__1_0	5.828e^(-69)
9. Blood_transfusion_2_0	2.335e^(-68)
10. Pain_scale_monitored_2_0	4.991e^(-66)
11. DVT_prophylaxis_1_0	5.227e^(-65)
12. encephalopathy_2_0	1.212e^(-64)
13. CI_sedation_2_0	7.427e^(-64)
14. Any_Other_Culture_Sent_1_0	1.297e^(-63)
15. Rel_adjustment_1_0	1.462e^(-63)
16. Hemodymic_monitoring_1_0	1.729e^(-59)
17. vasopressor_1_0	2.042e^(-59)
18. Clinical_Feature_hypoperfusion_2_0	9.597e^(-58)
19. Nutrition_started_2_0	9.630e^(-58)
20. Descalation_of_antibiotic_2_0	6.022e^(-57)
21. Was_sedation_scale_1_0	1.217e^(-56)
22. Thiamine_2_0	1.257e^(-56)
23. Discharge_medical_advice_2_0	6.116e^(-56)
24. Albumin_infusion__3hr_2_0	1.276e^(-55)
25. Loading_dose_2_0	2.205e^(-54)
26. septic_shock_1_0	1.979e^(-52)
27. Systolic_BP_lesser_than_90_1_0	5.110e^(-52)
28. encephalopathy_1_0	9.922e^(-52)
29. Blood_transfusion_1_0	1.555e^(-51)
30. Infection__PrimaryReason_ICU_Admission__2_0	1.721e^(-51)
31. Pain_scale_monitored_1_0	2.577e^(-51)
32. Inf_organism_cultured_1_0	9.209e^(-51)
33. Rel_Support_1_0	2.195e^(-50)
34. Any_Other_Culture_Sent_2_0	1.274e^(-48)

Feature	p-value
35. Corticosteroids_2_0	5.071e^(-48)
36. Rel_adjustment_2_0	1.805e^(-47)
37. Clinical_Feature_hypoperfusion_1_0	4.488e^(-46)
38. vasopressor_2_0	3.789e^(-43)
39. Descalation_of_antibiotic_1_0	4.119e^(-43)
40. DVT_prophylaxis_2_0	4.657e^(-43)

4.3 Methodology

Figure 2 shows the framework of the classification method. After increasing the number of samples in the dataset, the pre-processing steps are performed.

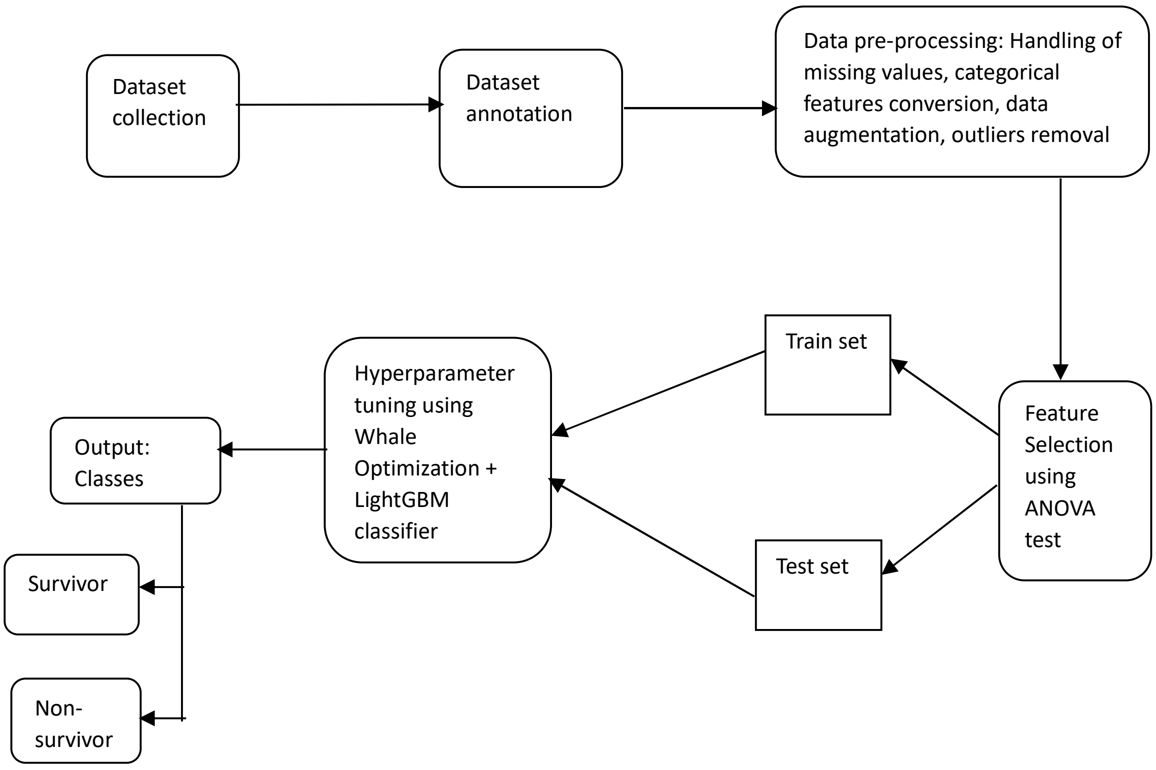

Dataset collection
Dataset annotation
Data pre-processing: Handling of missing values, categorical features conversion, data augmentation, outliers removal
Feature Selection using ANOVA test
Train set
Test set
Hyperparameter tuning using Whale Optimization + LightGBM classifier
Output: Classes
Survivor
Non-survivor

Figure 2: Framework of the classification method

The top 40 features are extracted using ANOVA. After the features are identified through ANOVA, the whole dataset is split into a training set and a testing set.

A meta-heuristic optimization algorithm is applied to adjust the hyperparameters of the LightGBM model for improved model performance. The meta-heuristic optimization technique, namely the whale optimization algorithm, is applied to obtain the optimized hyperparameters of LightGBM, which are then fed into the model for training. Each set of hyperparameters contains 8 hyperparameters: 1) number of leaves, 2) bagging frequency, 3) maximum depth, 4) min_child_samples, 5) feature fraction, 6) bagging fraction, 7) colsample_bytree, 8) subsample.

The fitness function used in the Whale Optimization algorithm is optimized to get the best results. The fitness function takes the LightGBM model as the input which gives the classification accuracy, sensitivity(recall), specificity, and hyperparameters. Specific intervals for each parameter define the range of hyperparameter exploration, enabling a comprehensive analysis of the parameter space while limiting the search to feasible areas. The whale optimization algorithm uses 24 search agents, a space with 8 dimensions, and a maximum number of iterations of 10. The search space's lower bound is 5, and its upper bound is 10.

5. Results and Analysis

In this work, Python (version 3.10.9) is used as the run-time environment with the required built-in libraries, such as numpy and pandas. Lightgbm, a Python library, is considered for the LightGBM model implementation. Sklearn built-in packages are considered for the evaluation metrics.

In a medical dataset of critical diseases, the ratio of the presence or absence of such diseases in an individual varies significantly. In this case, accurately predicting the death or survival of sepsis patients admitted to the ICU is crucial and must not be misrepresented. True positives indicate correctly identified sepsis patients who will die, contributing to high sensitivity (recall), crucial for timely interventions and treatment. Conversely, a greater number of false positives decreases precision and increases the risk of patients to remain untreated. False negatives indicate missed positive cases, reducing sensitivity and suggesting unnecessary treatment to survivors. High true negatives contribute to high specificity, ensuring the correct identification of surviving patients. Thus, false negatives and false positives should be low, while true positives and true negatives must be as high as possible. Sensitivity (True Positive Rate) measures the proportion of actual positive cases(survivors) that are correctly identified by the model. In mathematical terms, sensitivity can be represented as shown in Equation (9):

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \qquad (9)$$

where TP represents the number of true positives, and FN denotes the number of false negatives.

The dataset used in the proposed approach consists of sepsis patient data with an imbalanced mortality ratio of 2:1. There are 2279 individuals in the deceased category and 1371 individuals in the survivor category. In this context, it's important to note that classification accuracy may not provide the desired results due to the imbalanced ratio. Experimental results demonstrate that with a mortality ratio 2:1, the accuracy obtained is 72.8%, while the sensitivity is 84.9%. The model correctly predicted the 72.8% survivor class, but the rest of the 27.2% were considered dead as survivors. Thus, sensitivity has been chosen as the preferred metric due to its ability to provide effective results. Sensitivity is higher in this case because it measures the proportion of actual positives that are correctly identified, which is crucial in medical diagnoses whereas missing a positive case (false negative) can have severe consequences. Therefore, ensuring a high sensitivity helps in accurately identifying patients with sepsis, thereby reducing the risk of undetected positive cases. Specificity measures the model’s ability to correctly identify all negative instances. In mathematical terms, specificity can be represented as shown in Equation (10):

$$\text{Specificity} = \frac{TN}{TN + FP} \qquad (10)$$

where TN denotes true negative, and FP denotes false positive. In this case, specificity is 51.3%.

Additionally, the F1 score is an essential metric in evaluating model performance in imbalanced datasets, particularly in the healthcare domain. The F1 score harmonizes precision and recall, offering a balanced assessment of the classifier's accuracy by considering both false positives and false negatives, which are critical considerations in medical diagnosis. It is calculated as the harmonic mean of precision and recall. Recall represents sensitivity. Precision represents the proportion of positive predictions that are actually correct. Precision and F1 score can be calculated using the expressions shown in Equations (11) and (12):

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (11)$$

$$\text{F1} = \frac{2 \times Precision \times Recall}{Precision + Recall} \qquad (12)$$

where TP represents true positives and FP represents false positives. In this case, the precision is 75.5%, and the F1-score is 0.799. The confusion matrix of the test dataset is shown in Figure 3.

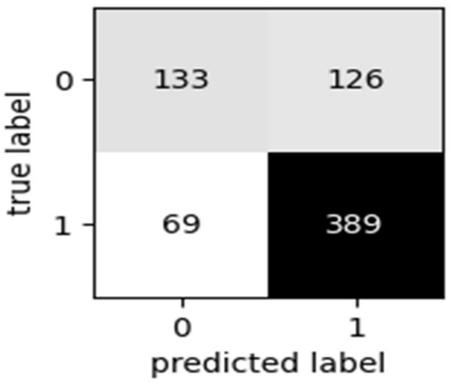


Figure 3: Confusion Matrix

This confusion matrix is used to evaluate the performance of the binary classification model. It compares the actual labels (true labels) with the predicted labels produced by the model. 133 patients who survived are accurately classified as such (true negatives), while 126 survivors are incorrectly predicted as deceased (false positives). Additionally, the model fails to identify 69 patients who ultimately succumbed to sepsis (false negatives) but successfully identifies 389 individuals who unfortunately passed away (true positives).

Table 2: Hyperparameters selected for whale optimization and their optimized values

Hyperparameter	Value
num_leaves	61
bagging_freq	6
max_depth	5
min_child_samples	56
feature_fraction	0.902952284118516
bagging_fraction	0.23112898351107802
colsample_bytree	0.6402379665917124
subsample	0.6700526236007138

The hyperparameter optimization process for LightGBM model yields the best set of hyperparameters as shown in Table 2. The num_leaves is set to 61, within the search space of [40, 100], allowing the model to capture complex interactions within the data; bagging_freq is optimized at 6, within the search space of [1, 10], enhancing robustness by reducing overfitting; max_depth is set to 5, within the search space of [3, 8], striking a balance between capturing intricate patterns and avoiding overfitting; min_child_samples is set to 56, within the search space of [45, 60], serving as a regularization parameter to prevent splitting nodes with very few samples; feature_fraction and colsample_bytree are set to 0.902952284118516 and 0.6402379665917124 respectively, within the search space of [0.1, 1], ensuring diverse subsets of features and reducing overfitting; bagging_fraction and subsample are set to 0.23112898351107802 and 0.6700526236007138 respectively, within the search space of [0.1, 1], effectively using subsampling to enhance model robustness and generalization performance. These hyperparameters, tuned within specified search spaces, balance model complexity, computational efficiency, and performance, significantly improving the model's robustness and predictive accuracy.

The following parameters of the machine learning model are considered for training. The loss function used is the binary cross-entropy loss, which is a standard for binary classification tasks and is calculated internally by using the Equation (13):

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(p_i) + (1 - y_i) log(1 - p_i)] \qquad (13)$$

where y_i represents the true binary label, p_i represents the predicted probability for the positive class, and N is the number of samples. A learning rate of 0.1 is used. GBDT is chosen as the boosting type. The model is trained for 100 boosting iterations. LightGBM doesn't have a default batch size parameter, as the entire dataset is generally used in each boosting iteration.

6. Conclusion and Future Scope

The proposed approach predicts the mortality of sepsis patients by tuning the hyperparameters of LightGBM by using a meta-heuristic algorithm to improve the performance of the gradient boosting model. Performance of the LightGBM model is measured in the sepsis patient dataset which is augmented and pre-processed. Sets of 8 hyperparameters of the LightGBM model are tuned to improve the model performance. Each agent's performance is evaluated using an objective function that trains a LightGBM model with the agent's hyperparameters and returns performance metrics (accuracy, sensitivity, specificity, confusion matrix). As it is an imbalanced dataset with an imbalance ratio of 2:1, accuracy is not the correct metric for performance evaluation as it may consider some dead patients as survivors. For performance evaluation sensitivity is preferred. Sensitivity (True Positive Rate) measures the proportion of actual positive cases (survivors) that are correctly identified by the model. This approach achieved an accuracy of 72.8% with a sensitivity of 84.9%. and specificity of 51.3%. Additionally, the F1-score is calculated as it harmonizes precision and recall, offering a balanced assessment of a classifier's accuracy by considering both false positives and false negatives. The calculated F1-score is 0.799.

In the future, this work can be extended further to improve the evaluation metrics of the model by applying other meta-heuristic and hybrid meta-heuristic algorithms to auto tune the hyperparameters.

Reference

[1] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J. D. Chiche, J. C. Marshall, C. M. Coopersmith, M. M. Levy, “The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3*)*”, *Special Communication,* vol. 315, pp. 801-810, 2016.

[2] G. Kong, K. Lin, Y. Hu, “Using machine learning methods to predict in-hospital mortality of sepsis patients in the ICU”, *BMC Medical Informatics and Decision Making*, vol. 20, 2020.

[3] D. A. Ojeda, V. Iglesias, F. Bobillo, L. Rico, F. Gandia, A. M. Loma, C. Nieto, R. Diego, E. Ramos, “Early natural killer cell counts in blood predict mortality in severe sepsis”, *Critical Care*, vol. 15, 2011.

[4] C. Bao, F. Deng, S. Zhao, “Machine-learning models for prediction of sepsis patients mortality”, *Medicina Intensiva*, vol. 47, issue 6, pp. 315-325, 2023.

[5] W. F. Fang, I. S. Douglas, Y. M. Chen, C. Y. Lin, H. C. Kao, Y. T. Fang, C. H. Huang, Y. T. Chang, K. T. Huang, Y. H. Wang, C. C. Wang, M. C. Lin, “Development and validation of immune dysfunction score to predict 28-day mortality of sepsis patients”, *PLOS ONE*, vol. 12, issue 10, 2017.

[6] Y. S. Kwon, M. S. Baek, “Development and Validation of a Quick Sepsis-Related Organ Failure Assessment-Based Machine-Learning Model for Mortality Prediction in Patients with Suspected Infection in the Emergency Department”, *Journal of Clinical Medicine*, vol. 9, issue 3, 2020.

[7] M. Mahdi, S. Al-Janabi, “A novel software to improve healthcare base on predictive analytics and mobile services for cloud data centers”, *Springer*, vol. 81, pp. 320-339, 2020.

[8] S. V. Pouchke, Z. Zhang, M. Schmitz, M. Vukicevic, M. V. Laenen, L. A. Celi, C. D. Deyne, “Scalable predictive analysis in critically ill patients using a visual open data analysis platform”, *PLOS ONE*, vol. 11, 2016.

[9] H. Zhou, L. Liu, Q. Zhao, X. Jin, Z. Peng, W. Wang, L. Huang, Y. Xie, H. Xu, L. Tao, X. Xiao, W. Nie, F. Liu, L. Li, Q. Yuan, “Machine learning for the prediction of all-cause mortality in patients with sepsis-associated acute kidney injury during hospitalization”, *Frontiers in Immunology*, vol. 14, issue 10, 2023.

[10] Z. Zheng, S. Yao, J. Zheng, X. Gong, “Development and validation of a novel blending machine learning model for hospital mortality prediction in ICU patients with Sepsis”, *BioData Mining*, vol. 14, issue 40, 2021.

[11] Z-H. Zhuang, M-C Chiang, “The Whale Optimization Algorithm for Hyperparameter Optimization in Network-Wide Traffic Speed Prediction” in *ACM International Conference on Intelligent Computing and its emerging Applications*, GangWon, Republic of Korea, 2020.

[12] Z. Zhang, Y. Zhao, A. Canes, D. Steinberg, O. Lyashevska, “Predictive analytics with gradient boosting in clinical medicine”, *Annals of Translational Medicine,* vol. 7, issue 7, 2019.

[13] L. Su, Z. Xu, F. Chang, Y. Ma, S. Liu, H. Jiang, H. Wang, D. Li, H. Chen, X. Zhou, N. Hong, W Zhu, Y. Long, “Early Prediction of Mortality, Severity, and Length of Stay in the Intensive-Care Unit of Sepsis patients Based on Sepsis 3.0 by Machine Learning Models”, *Frontiers in Medicine*, vol. 8, 2021.

[14] Y. Su, C. Guo, S. Zhou, C. Li, N. Ding, “Early predicting 30-day mortality in sepsis in MIMIC-III by an artificial neural networks model”, *European Journal of Medical Research*, vol. 27, issue 294, 2022.

[15] N. Hou, M. Li, L. He, B. Xie, L. Wang, R. Zhang, Y. Yu, X. Sun, Z. Pan, K. Wang, “Predicting 30-days mortality for MIMIC-III patients with sepsis-3: a machine learning approach using XGboost”, *Journal of Translational Medicine,* vol. 18, 2020.

[16] L. Peng, C. Peng, F. Yang, J. Wang, W. Zuo, C. Cheng, Z. Mao, Z. Jin, W. Li, “Machine learning approach for the prediction of 30-day mortality in patients with sepsis-associated encephalopathy”, *BMC Medical Research Methodology,* vol. 22, issue 183, 2022.

[17] R. A. Taylor, J. R. Pare, A. K. Venkatesh, H. Mowafi, E. R. Melnick, W. Fleischman, M. K. Hall, “Prediction of In-hospital Mortality in Emergency Department Patients With Sepsis: A Local Big Data–Driven, Machine Learning Approach”, *Academic Emergency Medicine*, vol.23, issue 3, pp. 269-278, 2016.

[18] Z. Zhang and Y. Hong, “Development of a novel score for the prediction of hospital mortality in patients with severe sepsis: the use of electronic healthcare records with LASSO regression”, *Oncotarget*, vol. 8, issue 30, pp. 49637-49645, 2017.

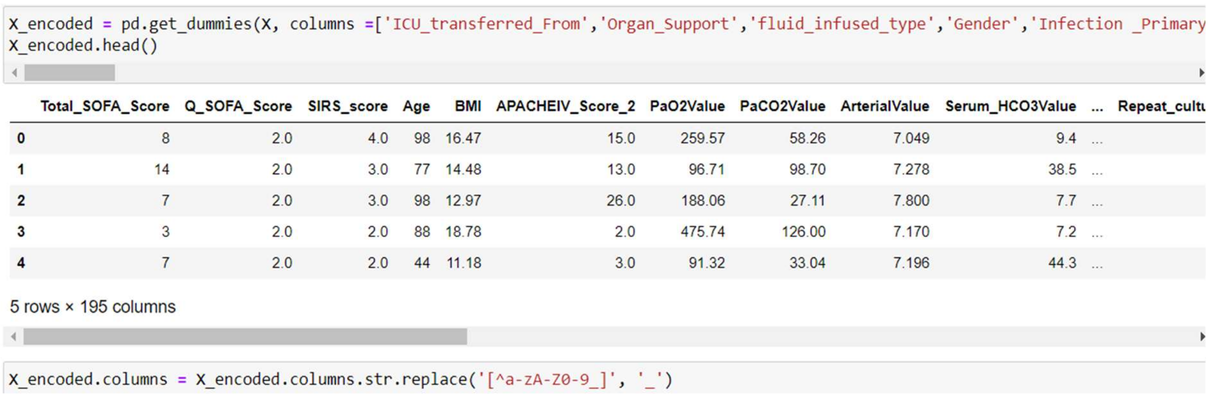
[19] H. Zamani, M. H. N. Shahraki, “Swarm Intelligence Approach for Breast Cancer Diagnosis”, *International Journal of Computer Applications,* vol. 151, issue 1, pp. 40-44, 2016.

[20] F. Navazi, Y. Yuan, N. Archer, “An examination of the hybrid meta-heuristic machine learning algorithms for early diagnosis of type II diabetes using big data feature selection”, *Healthcare Analytics,* vol. 4, 2023.

[21] S. Vijh, D. Gaur, S. Kumar, “An intelligent lung tumour diagnosis system using whale optimization algorithm and support vector machine”, *International Journal of System Assurance Engineering and Management,* vol. 11, issue 2, pp. 374-384, 2020.

Appendix

Sepsis patient mortality prediction model using Whale Optimization Algorithm and LightGBM.

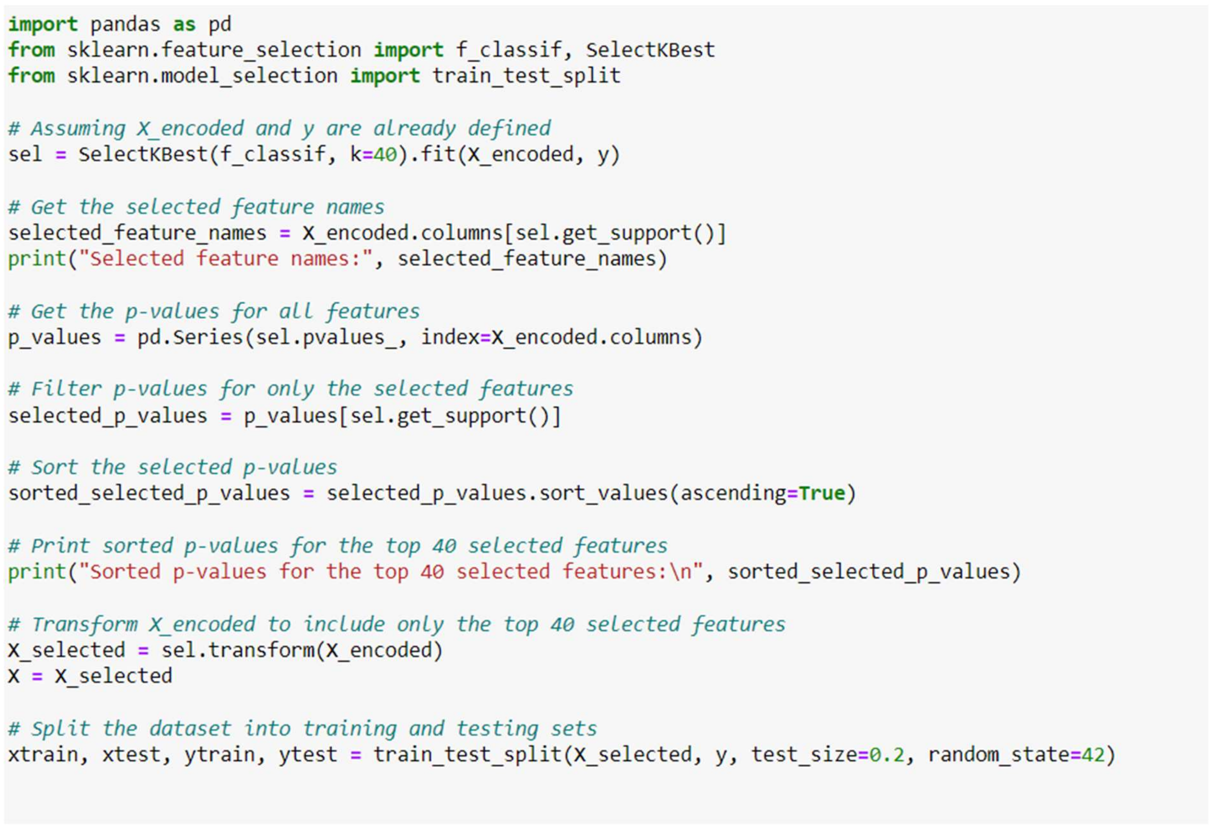


```
X_encoded = pd.get_dummies(X, columns =['ICU_transferred_From','Organ_Support','fluid_infused_type','Gender','Infection _Primary
X_encoded.head()
```

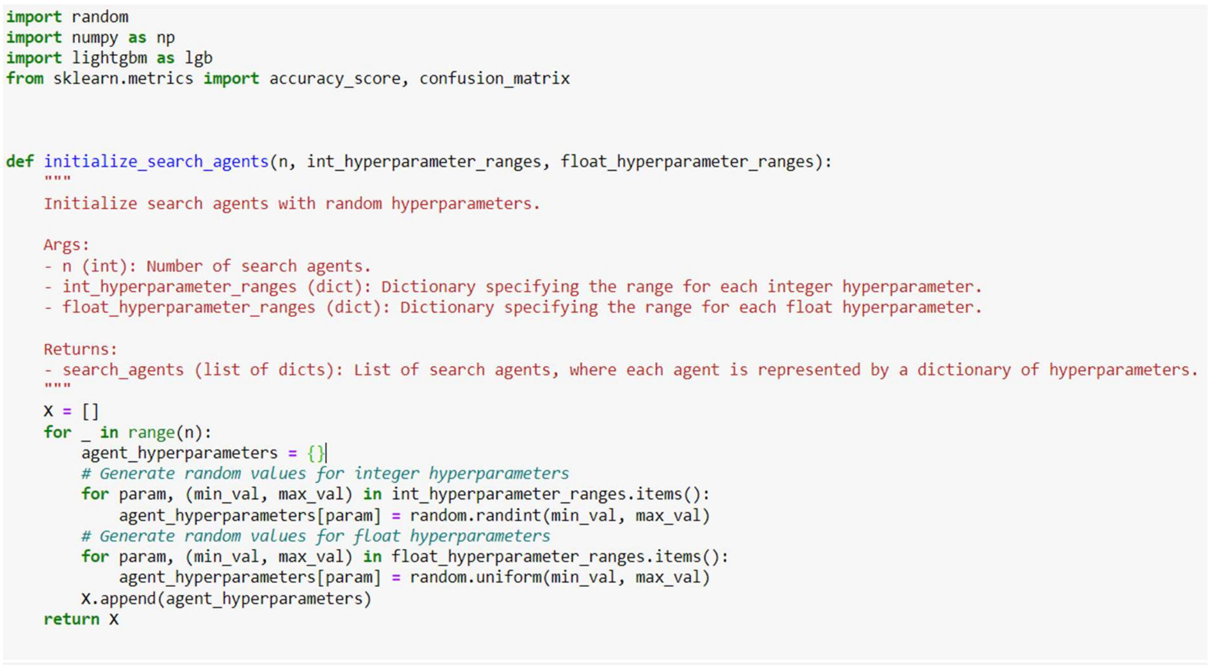
	Total_SOFA_Score	Q_SOFA_Score	SIRS_score	Age	BMI	APACHEIV_Score_2	PaO2Value	PaCO2Value	ArterialValue	Serum_HCO3Value	...	Repeat_cultu
0	8	2.0	4.0	98	16.47	15.0	259.57	58.26	7.049	9.4	...	
1	14	2.0	3.0	77	14.48	13.0	96.71	98.70	7.278	38.5	...	
2	7	2.0	3.0	98	12.97	26.0	188.06	27.11	7.800	7.7	...	
3	3	2.0	2.0	88	18.78	2.0	475.74	126.00	7.170	7.2	...	
4	7	2.0	2.0	44	11.18	3.0	91.32	33.04	7.196	44.3	...	

5 rows × 195 columns

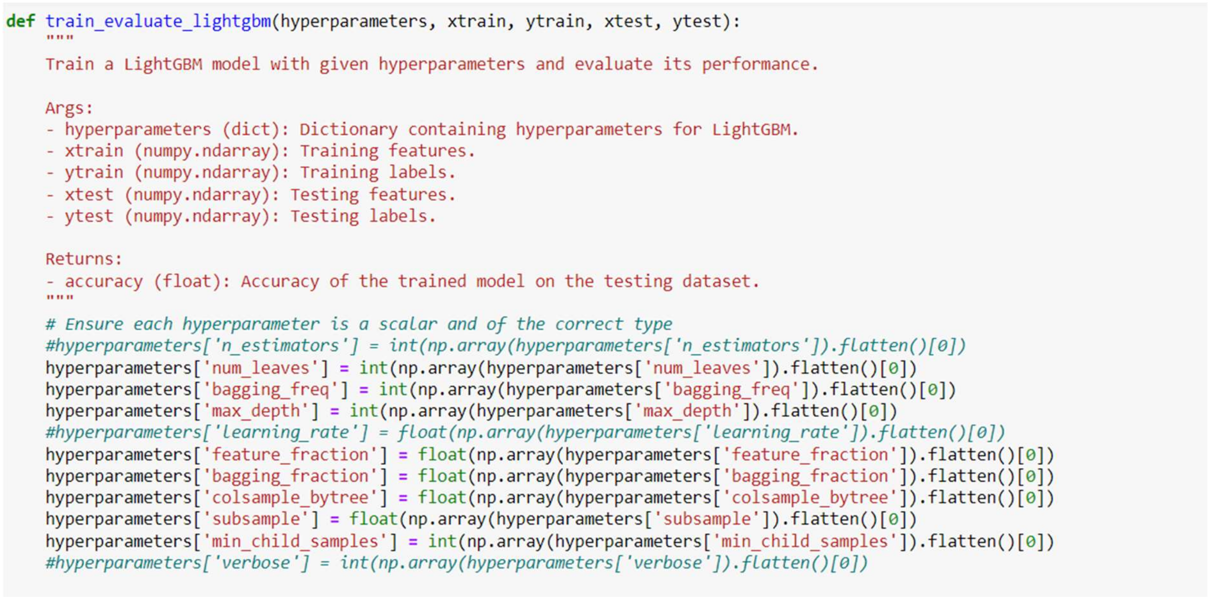
```
X_encoded.columns = X_encoded.columns.str.replace('[^a-zA-Z0-9_]', '_')
```



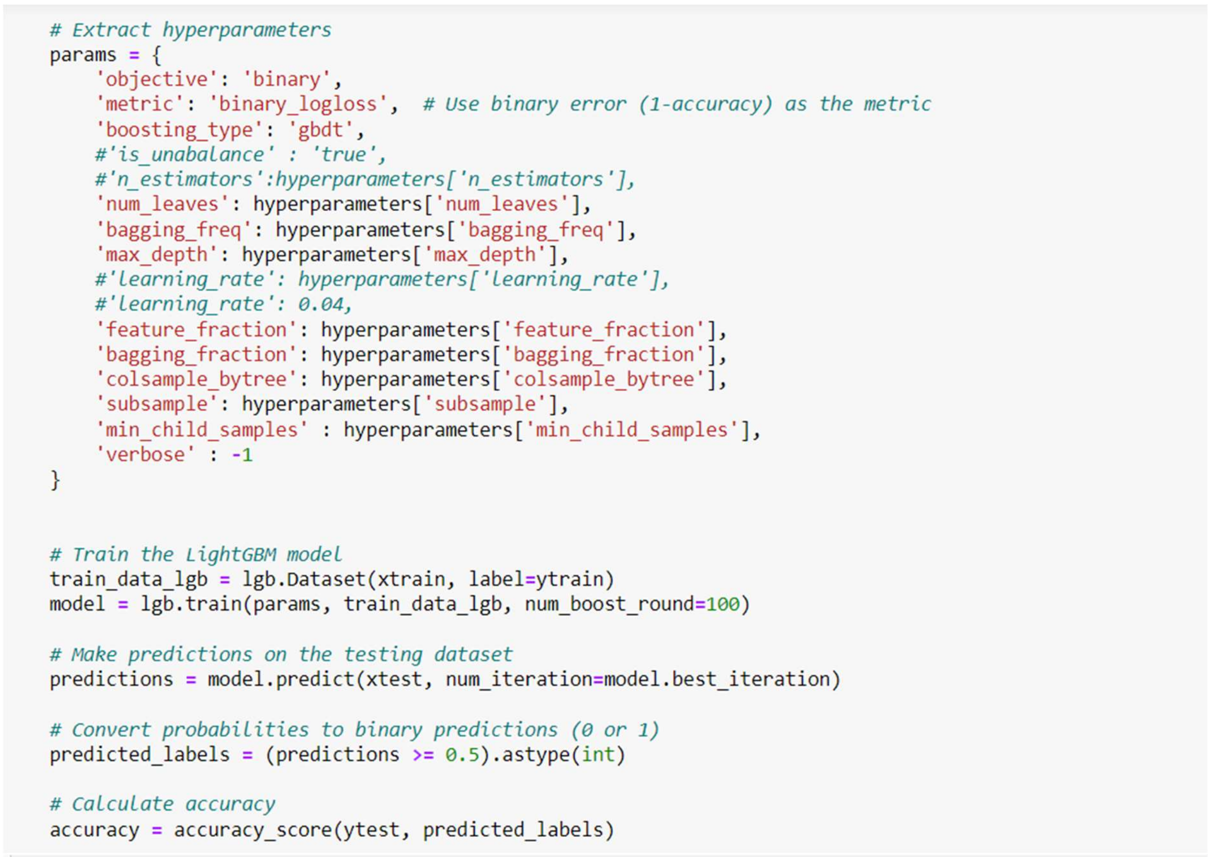
```
import pandas as pd
from sklearn.feature_selection import f_classif, SelectKBest
from sklearn.model_selection import train_test_split

# Assuming X_encoded and y are already defined
sel = SelectKBest(f_classif, k=40).fit(X_encoded, y)

# Get the selected feature names
selected_feature_names = X_encoded.columns[sel.get_support()]
print("Selected feature names:", selected_feature_names)

# Get the p-values for all features
p_values = pd.Series(sel.pvalues_, index=X_encoded.columns)

# Filter p-values for only the selected features
selected_p_values = p_values[sel.get_support()]

# Sort the selected p-values
sorted_selected_p_values = selected_p_values.sort_values(ascending=True)

# Print sorted p-values for the top 40 selected features
print("Sorted p-values for the top 40 selected features:\n", sorted_selected_p_values)

# Transform X_encoded to include only the top 40 selected features
X_selected = sel.transform(X_encoded)
X = X_selected

# Split the dataset into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(X_selected, y, test_size=0.2, random_state=42)
```



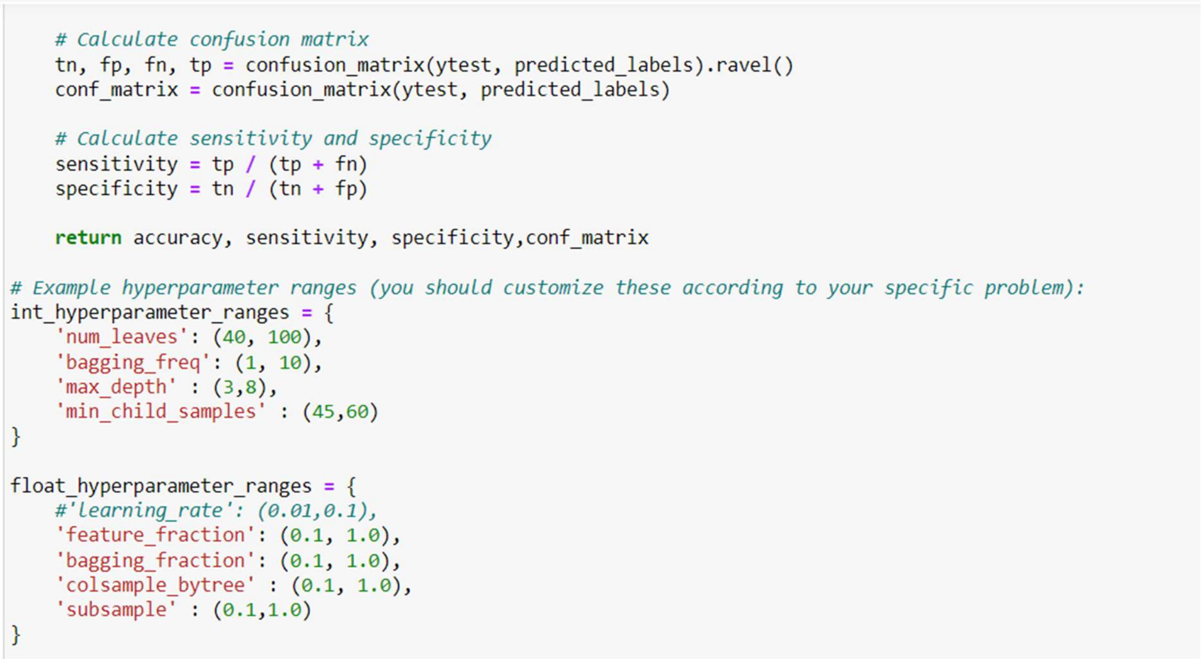
```python
import random
import numpy as np
import lightgbm as lgb
from sklearn.metrics import accuracy_score, confusion_matrix



def initialize_search_agents(n, int_hyperparameter_ranges, float_hyperparameter_ranges):
    """
    Initialize search agents with random hyperparameters.

    Args:
    - n (int): Number of search agents.
    - int_hyperparameter_ranges (dict): Dictionary specifying the range for each integer hyperparameter.
    - float_hyperparameter_ranges (dict): Dictionary specifying the range for each float hyperparameter.

    Returns:
    - search_agents (list of dicts): List of search agents, where each agent is represented by a dictionary of hyperparameters.
    """
    X = []
    for _ in range(n):
        agent_hyperparameters = {}
        # Generate random values for integer hyperparameters
        for param, (min_val, max_val) in int_hyperparameter_ranges.items():
            agent_hyperparameters[param] = random.randint(min_val, max_val)
        # Generate random values for float hyperparameters
        for param, (min_val, max_val) in float_hyperparameter_ranges.items():
            agent_hyperparameters[param] = random.uniform(min_val, max_val)
        X.append(agent_hyperparameters)
    return X
```



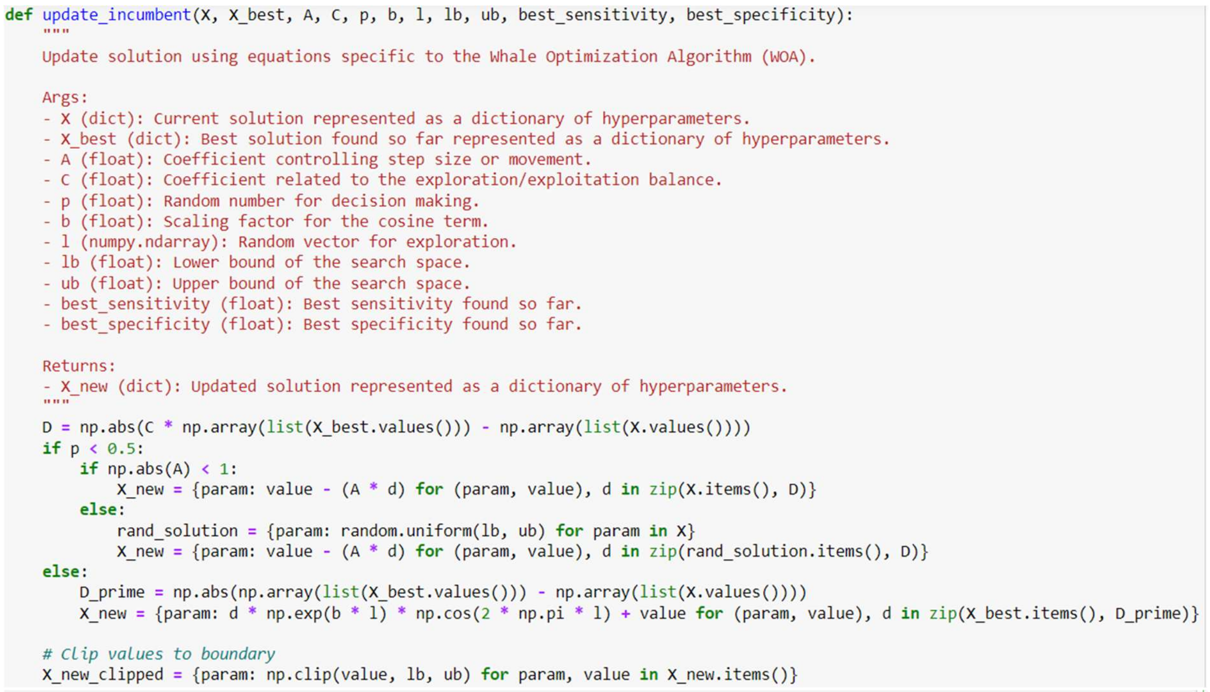
```python
def train_evaluate_lightgbm(hyperparameters, xtrain, ytrain, xtest, ytest):
    """
    Train a LightGBM model with given hyperparameters and evaluate its performance.

    Args:
    - hyperparameters (dict): Dictionary containing hyperparameters for LightGBM.
    - xtrain (numpy.ndarray): Training features.
    - ytrain (numpy.ndarray): Training labels.
    - xtest (numpy.ndarray): Testing features.
    - ytest (numpy.ndarray): Testing labels.

    Returns:
    - accuracy (float): Accuracy of the trained model on the testing dataset.
    """
    # Ensure each hyperparameter is a scalar and of the correct type
    #hyperparameters['n_estimators'] = int(np.array(hyperparameters['n_estimators']).flatten()[0])
    hyperparameters['num_leaves'] = int(np.array(hyperparameters['num_leaves']).flatten()[0])
    hyperparameters['bagging_freq'] = int(np.array(hyperparameters['bagging_freq']).flatten()[0])
    hyperparameters['max_depth'] = int(np.array(hyperparameters['max_depth']).flatten()[0])
    #hyperparameters['learning_rate'] = float(np.array(hyperparameters['learning_rate']).flatten()[0])
    hyperparameters['feature_fraction'] = float(np.array(hyperparameters['feature_fraction']).flatten()[0])
    hyperparameters['bagging_fraction'] = float(np.array(hyperparameters['bagging_fraction']).flatten()[0])
    hyperparameters['colsample_bytree'] = float(np.array(hyperparameters['colsample_bytree']).flatten()[0])
    hyperparameters['subsample'] = float(np.array(hyperparameters['subsample']).flatten()[0])
    hyperparameters['min_child_samples'] = int(np.array(hyperparameters['min_child_samples']).flatten()[0])
    #hyperparameters['verbose'] = int(np.array(hyperparameters['verbose']).flatten()[0])
```



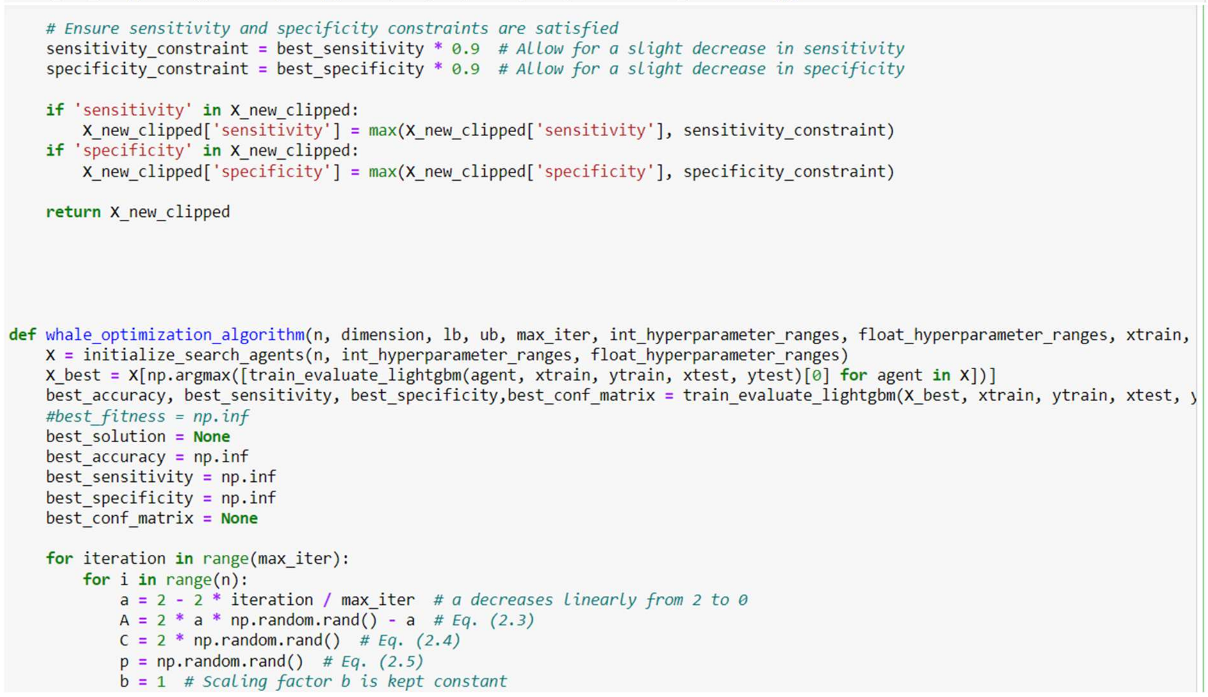
```
    # Extract hyperparameters
    params = {
        'objective': 'binary',
        'metric': 'binary_logloss',  # Use binary error (1-accuracy) as the metric
        'boosting_type': 'gbdt',
        #'is_unabalance' : 'true',
        #'n_estimators':hyperparameters['n_estimators'],
        'num_leaves': hyperparameters['num_leaves'],
        'bagging_freq': hyperparameters['bagging_freq'],
        'max_depth': hyperparameters['max_depth'],
        #'learning_rate': hyperparameters['learning_rate'],
        #'learning_rate': 0.04,
        'feature_fraction': hyperparameters['feature_fraction'],
        'bagging_fraction': hyperparameters['bagging_fraction'],
        'colsample_bytree': hyperparameters['colsample_bytree'],
        'subsample': hyperparameters['subsample'],
        'min_child_samples' : hyperparameters['min_child_samples'],
        'verbose' : -1
    }


    # Train the LightGBM model
    train_data_lgb = lgb.Dataset(xtrain, label=ytrain)
    model = lgb.train(params, train_data_lgb, num_boost_round=100)

    # Make predictions on the testing dataset
    predictions = model.predict(xtest, num_iteration=model.best_iteration)

    # Convert probabilities to binary predictions (0 or 1)
    predicted_labels = (predictions >= 0.5).astype(int)

    # Calculate accuracy
    accuracy = accuracy_score(ytest, predicted_labels)
```



```
    # Calculate confusion matrix
    tn, fp, fn, tp = confusion_matrix(ytest, predicted_labels).ravel()
    conf_matrix = confusion_matrix(ytest, predicted_labels)

    # Calculate sensitivity and specificity
    sensitivity = tp / (tp + fn)
    specificity = tn / (tn + fp)

    return accuracy, sensitivity, specificity,conf_matrix

# Example hyperparameter ranges (you should customize these according to your specific problem):
int_hyperparameter_ranges = {
    'num_leaves': (40, 100),
    'bagging_freq': (1, 10),
    'max_depth' : (3,8),
    'min_child_samples' : (45,60)
}

float_hyperparameter_ranges = {
    #'learning_rate': (0.01,0.1),
    'feature_fraction': (0.1, 1.0),
    'bagging_fraction': (0.1, 1.0),
    'colsample_bytree' : (0.1, 1.0),
    'subsample' : (0.1,1.0)
}
```



```python
def update_incumbent(X, X_best, A, C, p, b, l, lb, ub, best_sensitivity, best_specificity):
    """
    Update solution using equations specific to the Whale Optimization Algorithm (WOA).

    Args:
    - X (dict): Current solution represented as a dictionary of hyperparameters.
    - X_best (dict): Best solution found so far represented as a dictionary of hyperparameters.
    - A (float): Coefficient controlling step size or movement.
    - C (float): Coefficient related to the exploration/exploitation balance.
    - p (float): Random number for decision making.
    - b (float): Scaling factor for the cosine term.
    - l (numpy.ndarray): Random vector for exploration.
    - lb (float): Lower bound of the search space.
    - ub (float): Upper bound of the search space.
    - best_sensitivity (float): Best sensitivity found so far.
    - best_specificity (float): Best specificity found so far.

    Returns:
    - X_new (dict): Updated solution represented as a dictionary of hyperparameters.
    """
    D = np.abs(C * np.array(list(X_best.values())) - np.array(list(X.values())))
    if p < 0.5:
        if np.abs(A) < 1:
            X_new = {param: value - (A * d) for (param, value), d in zip(X.items(), D)}
        else:
            rand_solution = {param: random.uniform(lb, ub) for param in X}
            X_new = {param: value - (A * d) for (param, value), d in zip(rand_solution.items(), D)}
    else:
        D_prime = np.abs(np.array(list(X_best.values())) - np.array(list(X.values())))
        X_new = {param: d * np.exp(b * l) * np.cos(2 * np.pi * l) + value for (param, value), d in zip(X_best.items(), D_prime)}

    # Clip values to boundary
    X_new_clipped = {param: np.clip(value, lb, ub) for param, value in X_new.items()}
```



```python
    # Ensure sensitivity and specificity constraints are satisfied
    sensitivity_constraint = best_sensitivity * 0.9  # Allow for a slight decrease in sensitivity
    specificity_constraint = best_specificity * 0.9  # Allow for a slight decrease in specificity

    if 'sensitivity' in X_new_clipped:
        X_new_clipped['sensitivity'] = max(X_new_clipped['sensitivity'], sensitivity_constraint)
    if 'specificity' in X_new_clipped:
        X_new_clipped['specificity'] = max(X_new_clipped['specificity'], specificity_constraint)

    return X_new_clipped




def whale_optimization_algorithm(n, dimension, lb, ub, max_iter, int_hyperparameter_ranges, float_hyperparameter_ranges, xtrain,
    X = initialize_search_agents(n, int_hyperparameter_ranges, float_hyperparameter_ranges)
    X_best = X[np.argmax([train_evaluate_lightgbm(agent, xtrain, ytrain, xtest, ytest)[0] for agent in X])]
    best_accuracy, best_sensitivity, best_specificity,best_conf_matrix = train_evaluate_lightgbm(X_best, xtrain, ytrain, xtest, y
    #best_fitness = np.inf
    best_solution = None
    best_accuracy = np.inf
    best_sensitivity = np.inf
    best_specificity = np.inf
    best_conf_matrix = None

    for iteration in range(max_iter):
        for i in range(n):
            a = 2 - 2 * iteration / max_iter  # a decreases linearly from 2 to 0
            A = 2 * a * np.random.rand() - a  # Eq. (2.3)
            C = 2 * np.random.rand()  # Eq. (2.4)
            p = np.random.rand()  # Eq. (2.5)
            b = 1  # Scaling factor b is kept constant
            l = np.random.uniform(-1, 1, dimension)  # Eq. (2.6)

            X[i] = update_incumbent(X[i], X_best, A, C, p, b, l, lb, ub, best_sensitivity, best_specificity)

        for i in range(len(X)):
            for param in X[i]:
                X[i][param] = np.clip(X[i][param], lb, ub)
            if 'bagging_fraction' in X[i]:
                X[i]['bagging_fraction'] = np.clip(X[i]['bagging_fraction'], 0.1, 1.0)
            if 'feature_fraction' in X[i]:
                X[i]['feature_fraction'] = np.clip(X[i]['feature_fraction'], 0.1, 1.0)


        fitness = [train_evaluate_lightgbm(agent, xtrain, ytrain, xtest, ytest) for agent in X]
        '''
        max_fitness_index = np.argmax([f[0] for f in fitness])
        max_accuracy, max_conf_matrix = X_best

        if max_accuracy < best_accuracy:
            best_accuracy = max_accuracy
            best_solution = X_best
            #best_solution = X[max_fitness_index]
            best_sensitivity = fitness[max_fitness_index][1]
            best_specificity = fitness[max_fitness_index][2]
            best_conf_matrix = max_conf_matrix
        '''
        max_fitness_index = np.argmax([f[0] for f in fitness])
        max_fitness = fitness[max_fitness_index][0]

        if max_fitness < best_accuracy:
            best_accuracy = max_fitness
            best_solution = X_best
            #best_solution = X[max_fitness_index]
            best_sensitivity = fitness[max_fitness_index][1]
```

```python
            best_sensitivity = fitness[max_fitness_index][1]
            best_specificity = fitness[max_fitness_index][2]
            best_conf_matrix = fitness[max_fitness_index][3]

    return best_solution, best_accuracy, best_sensitivity, best_specificity,best_conf_matrix


'''
ı = 24  # Number of search agents
dimension = 8  # Dimensionality of the search space
lb = 100  # Lower bound of the search space
ıb = 500  # Upper bound of the search space
nax_iter = 20  # Maximum number of iterations
'''

ı = 24  # Number of search agents
dimension = 8  # Dimensionality of the search space
lb = 5  # Lower bound of the search space
ıb = 10  # Upper bound of the search space
nax_iter = 10  # Maximum number of iterations

oest_solution, best_accuracy, best_sensitivity, best_specificity, best_conf_matrix = whale_optimization_algorithm(n, dimension, l
orint("Best solution:", best_solution)
orint("Best accuracy:", best_accuracy)
orint("Best sensitivity:", best_sensitivity)
orint("Best specificity:", best_specificity)
orint("Best confusion matrix:",best_conf_matrix)
```