Received 12 June 2023, accepted 13 July 2023, date of publication 17 July 2023, date of current version 27 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3296147

RESEARCH ARTICLE

Predicting ICU Mortality Based on Generative Adversarial Nets and Ensemble Methods

MINGYI WEI[1,2], ZHEJUN HUANG[1,2], DIPING YUAN[3], AND LILI YANG[1,2]
[1]Shenzhen Key Laboratory of Safety and Security for Next Generation of Industrial Internet, Southern University of Science and Technology, Shenzhen 518055, China
[2]Department of Statistics and Data Science, Southern University of Science and Technology, Shenzhen 518055, China
[3]Technology and Information Center, Shenzhen Urban Public Safety Technology Institute, Shenzhen 518100, China

Corresponding author: Lili Yang (yangll@sustech.edu.cn)

This work was supported in part by the Shenzhen Science and Technology Program under Grant ZDSYS20210623092007023 and Grant JCYJ20200109141218676, in part by the Shenzhen Scientific Research Funding under Grant K22627501, and in part by the National Key Research and Development Program of China under Grant 2019YFC0810705.

ABSTRACT The intensive care unit (ICU) typically admits patients who require urgent medical intervention. Predicting ICU mortality is crucial for identifying those who are at higher risk. Traditional statistical methods, such as logistic regression, have been widely used for ICU survival prediction. However, these methods often have limitations in capturing complex nonlinear relationships between the clinical features. A prediction model based on ensemble learning was proposed for the ICU mortality prediction problem: MTX-stacking model. Firstly, the imbalanced data was processed based on the modified generative adversarial network method. This approach is more explanatory and more effective than traditional data generation methods. Secondly, XGBoost was optimized by tree-structured parzen estimator and stacking structure to prevent overfitting. The proposed MTX-stacking model was evaluated using 131,051 patients from MIT's GOSSIS initiative. The results indicate that MTX-stacking outperforms the state-of-the-art approaches in terms of area under the receiver operator characteristic (ROC) curve (91.2% and 90.9%). These findings demonstrate the ability and efficiency of the MTX-stacking model to predict ICU mortality.

INDEX TERMS ICU survival prediction, ensemble learning, generative adversarial nets, Bayesian optimization.

I. INTRODUCTION

During the ongoing COVID-19 pandemic, the rapid assessment of patients' overall health status assumes paramount importance, as healthcare providers worldwide grapple with the management of overwhelmed hospitals inundated with critically ill individuals. Amongst these patients, those with the most severe conditions requiring life-sustaining therapies or close monitoring are admitted to the intensive care unit (ICU). The ICU serves as a pivotal component in enhancing emergency care effectiveness and further reducing mortality rates, given its concentration of state-of-the-art monitoring technologies and emergency facilities [1].

Regrettably, the availability of verified medical histories for admitted patients in the ICU is often inadequate. Unconscious or unresponsive patients are unable to provide information regarding chronic conditions such as heart disease or diabetes. Moreover, medical records may take considerable time to be transferred, particularly for patients originating from different healthcare providers or systems. Early identification of individuals at high risk of mortality enables clinicians to make informed decisions regarding patient care. For instance, patients identified as high-risk can benefit from more aggressive interventions like mechanical ventilation or renal replacement therapy. Conversely, patients deemed low-risk can receive appropriate treatment without undergoing unnecessary tests or procedures.

Considering the substantial mortality rate observed in ICU patients, various prognostic scoring systems have been developed to accurately predict individual mortality [2], [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

Traditional scoring systems such as the Assessment and Chronic Health Evaluation (APACHE) II and III, Simplified Acute Physiology Score (SAPS) II and III, Mortality Probability Model (MPM) II and III, and Sequential Organ Failure Assessment (SOFA) have exhibited moderate efficacy in predicting individual patient mortality. However, the limited accuracy of these scoring systems hinders their universal applicability to all ICU patients, as documented in the literature [4]. Consequently, there has been an increasing utilization of machine learning (ML) techniques, particularly in the realm of ICU mortality prediction [5]. The purpose of this study was to propose an accurate and medically intuitive ICU mortality prediction model based on GAN and stacking structure. The presence of class imbalance in the dataset, coupled with the limitations of previous methods for handling class imbalance that lack interpretability. Therefore, GAN is selected to address the issue of class imbalance and theoretical evidence is provided to support this choice. Ensemble learning can improve the performance of classifiers to some extent. Therefore, after applying GAN for data preprocessing, this paper selects the Bayesian optimization-based XGBoost classifier as the first-layer model and combines it using the stacking framework to obtain the final MTX-stacking model. We compared MTX-stacking model with conventional scoring systems and ML algorithms. Furthermore, we also explained the purposed model using SHAP.

II. RELATED WORK

Predicting ICU mortality involves using clinical data and other relevant factors to estimate the likelihood that a patient admitted to the intensive care unit (ICU) will die while in the ICU. This information is important for physicians and other healthcare providers to help guide treatment decisions and provide appropriate care to the patient.

Recently, common ICU mortality prediction methods are mainly divided into three categories: statistical methods, machine learning methods and ensemble methods.

As for statistical learning methods, early prediction methods used logistic regression: Chen et al. [6] used multiple logistic regression to predict mortality of 134 patients with sepsis. Thao et al. [7] employed univariate logistic regression and multivariate analysis to predict mortality of 123 patients with septic shock. Ros et al. [8] used stepwise logistic regression to predict ICU mortality. Also, many prognostic scoring systems are frequently used to predict mortality using well-known generalized linear model approaches. Three of these scoring methods are widely used, namely the Acute Physiology and Chronic Health Evaluation (APACHE) scoring system [9], the Simplified Acute Physiology Score (SAPS) [3], and the Probabilistic Mortality Model (MPM) [10]. For the most recent versions of these systems, the worst physiological value for the first 24 hours after the patient's admission to the ICU is used to build logistic regression (LR) models, while other data are not used, which results in a loss of information. In addition, linear models are inaccurate in predicting patient status.

As technology is considerably researched and developed, and the amount of clinical data on ICU increases, ever-increasing researchers tend to use machine learning methods to predict mortality. Machine learning methods belong to nonlinear modeling approaches: artificial neural networks [11], [12], support vector machines [13], [14], decision trees [15], naive Bayesian models [16], [17]. Aczon et al. [18] and Alves et al. [19] employed RNN and CNN-LSTM respectively to predict ICU mortality. More complex models such as incremental information networks [20] have been explored in the last decades to characterize patients. All these approaches attempt to calculate risk scores for patients using global predictive models extracted from all available training data, however, the behavior of various patients is highly individualized. In the last decades, different patient-specific models have been developed for decision support [21] or adaptive monitoring in critical care [22], [23]. Recently, personalized models in the context of medical applications have been investigated, where KASABOV [24] proposed a personalized modeling approach using evolutionary optimization techniques, which was used in some particular fields, such as the design of customized drugs. Furthermore, just-in-time learning (JITL) and principal component analysis (PCA), referred to as learned PCA (L-PCA) [25], were combined to build an online personality model to monitor the patient's status, where JITL collects the most relevant samples for adaptive modeling of complex physiological processes and PCA was used for personalized modeling. Recently, another novel personalized modeling approach was proposed, named JITL-ELM, which integrates JITL and extreme learning machine (ELM). JITL borrows the diagnostic idea of ''similar symptoms characterizing similar outcomes'' to build patient-specific models by finding the most relevant samples, aiming to improve accuracy. However, these machine learning methods have some drawbacks. One of the disadvantages is that their prediction accuracy can be unstable, which means that the model's performance may fluctuate in different scenarios. Additionally, they can be difficult to interpret, making it challenging to understand how the model makes decisions.

The last category is ensemble methods. Final classification is achieved by building and combining multiple learners. Liu et al. [26] employed XGBoost for icu mortality prediction. Ye et al. [27] employed GBDT to predict icu survival rate. Guo et al. [28] proposed a dynamic ensemble learning algorithm based on $K-$means to predict mortality. An improved patient-specific stacking ensemble model [29] was proposed by combining KNN, MLP, DT and LR to predict mortality for MIMIC III dataset. A stacked ensemble model [30] was proposed by combining logistic regression, random forest, XGBoost and so on. The above ensemble learning methods have some drawbacks: firstly, the models are too complex and involve multiple machine learning methods, requiring training of multiple base models. Secondly, the final model is difficult to interpret and cannot avoid the ''black box'' phenomenon.

In general, the purpose of this study is to compensate for the shortcomings of the above three types of models: lack of information, low accuracy, poor model stability, and poor explanatory performance.

III. MODIFIED-GAN METHOD FOR PROCESSING DATA IMBALANCE

To improve the performance of a predictive model for ICU patients, it is necessary to prepare a new strategy. In this research, MTX-stacking model is proposed based on modified GAN, XGBoost and stacking structure. MTX-stacking model uses only XGBoost as the base model and then employs logistic regression as the final output. This has two advantages. On the one hand, it avoids over-fitting and improves the model's stability. On the other hand, it is time-efficient and reduces the time required to train machine learning-based models. Furthermore, the model's feature importance is determined using statistical methods, providing some explanation for the model. The flowchart of MTX-stacking is shown in Figure 1.

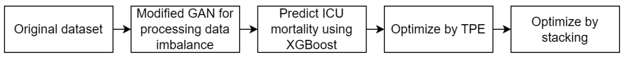
Original dataset
Modified GAN for processing data imbalance
Predict ICU mortality using XGBoost
Optimize by TPE
Optimize by stacking

FIGURE 1. The structure of MTX-stacking model.

The traditional method for processing data imbalance changes the number of samples by oversampling or undersampling. Repeated sampling can easily lead to overfitting; artificially generated sample often changes the distribution of data. The classical SMOTE [31] algorithm randomly synthesizes a new sample from the connection between the sample from minority class and its adjacent samples. In high-dimensional data, the spatial distance between samples grows exponentially, resulting in a pronounced deviation of the samples generated by the SMOTE algorithm from the distribution of real samples.

To address the problems of SMOTE, this research will use GAN [32] as a tool to deal with data imbalance. GAN is often used for image generation and have great effect. GAN is modified to process data imbalance as below: firstly, generate data for only the minority class, thus achieving data balance. Secondly, after generating data, a classifier layer is constructed to classify categorical variables.

A. DESCRIPTION OF ORIGINAL GAN

In contrast to the SMOTE algorithm, which functions as a generative model by generating new samples based on the distance between samples from the minority class, GAN adopts a different approach by learning the distribution of samples through noise. This innovative methodology overcomes the challenges of class overlap and high-dimensionality that are prevalent in the SMOTE algorithm, allowing for the more effective generation of new samples. GAN was first introduced by Ian J. Goodfellow in 2014. The structure of GAN is divided into two parts: a generation network G and a discriminative network D. The whole process of GAN can be shown in Figure 2.

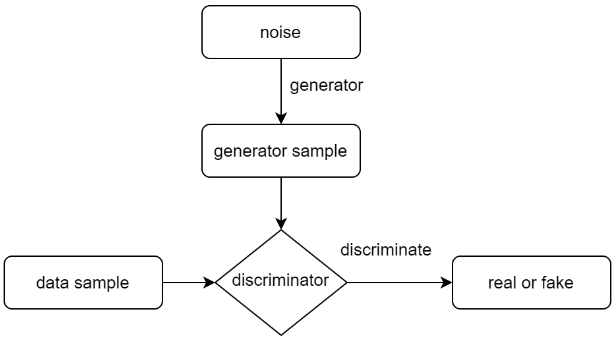

FIGURE 2. The structure of original generative adversarial net.

B. MODIFIED-GAN

GAN was used to generate data (mainly picture data) of different categories. In order to solve the problem of data imbalance, GAN was modified as below: assuming that the number of majority class's samples is n_1, the number of minority class's samples is n_2, the minority class's data is x. It is necessary to generate $n_1 - n_2$ minority class's samples to make the data balance. Assuming that the number of features is p, the shape of "generative data" is $(n_1 - n_2) * p$.

So firstly, assuming the initial distribution of the "noise" z is p_z. The aim is to use G to transfer z to a dataset $G(z)$ with the shape of $(n_1 - n_2) * p$. There is a question: how to measure the distance between generative data and real data? Discriminative network D will be introduced. It's aim is to discriminate whether the data is real (1) or generated (0). Given G:

- D wants to give x (x is taken from the real sample) the label of "1", which is equal to maximize $E_{x \sim P_{\text{data}}(x)}[\log D(x)]$.
- D also wants to give the generative data a label of "0", which helps G to generate better "real" data, which is equal to maximize $E_{z \sim P_z}[\log(1 - D(G(z)))]$.

So all in all, the discriminative nerwork $D's$ aim is to maximize:

$$V(D, G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim P_z}[\log(1 - D(G(z)))]. \tag{1}$$

The ideal state of the data augmentation method is to extract all samples from samples of the minority class, learn the overall distribution of these samples, and subsequently sample this learned distribution to generate a comprehensive set of "generative samples". However, such an approach is difficult to implement in practice, as it is not only time-consuming, but also difficult to learn the distribution of most samples. Therefore, Modified-GAN starts from noise to "simulate" the distribution of samples from the minority class and learns the distribution of real data through iterations. The final goal of GAN method was proved in [32]: use noise

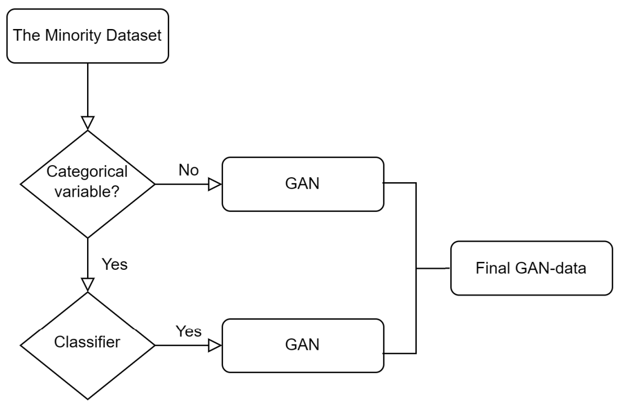

FIGURE 3. **Flowchart of ICU patients-based GAN.**

to learn the distribution of the real data and generate samples of the minority class.

Assuming the parameter of G is θ_G, the parameter of D is θ_D, the first generated data is called G_0, the learning rate is γ. D_0^* can be calculated. θ_{G_1} can be updated first by:

$$\theta_{G_1} \leftarrow \theta_{G_0} - \gamma \frac{\partial V(G_0, D_0^*)}{\partial \theta_{G_0}}, \tag{2}$$

and then the best D_1^* will be found and update θ_{G_2} by:

$$\theta_{G_2} \leftarrow \theta_{G_1} - \gamma \frac{\partial V(G_1, D_1^*)}{\partial \theta_{G_1}}, \tag{3}$$

the result of the nth update can be expressed as:

$$\theta_{G_n} \leftarrow \theta_{G_{n-1}} - \gamma \frac{\partial V(G_{n-1}, D_{n-1}^*)}{\partial \theta_{G_{n-1}}}. \tag{4}$$

After multiple iterations, the generated data will become more and more like real data. But there exists a problem to be solved: how to deal with binary features?

In the data set used, there are some binary variables, such as leukemia, solid_tumor_with_metastasis. If we just use the model above, these variables will become unrealistic in generated data. So the GAN is further modified as below: a classifier layer is constructed to classify the predicted results of these categorical variables after GAN. When the predicted value is greater than 0.5, the output is 1; otherwise, the output is 0. So the final structure of ICU patients-based GAN can be shown in Figure 3:

In all, G is a generative model, similar to shady merchants who make antique fakes and try to generate realistic antiques, while D is a discriminative model that tries to separate real antiques from fake ones.

Initially, G has poor generative effect and the generated fakes are easily discriminated by D; then G improves the generative effect based on the discriminative result of D, and D discriminates again. After several cycles, G's generation effect will approach the real antique step by step, and D's discriminating ability becomes more and more stronger. Such a dynamic game eventually makes G generate forgeries that D can no longer distinguish from real antiques.

The subtlety of this method is to introduce the idea of game theory into deep learning, where Modified-GAN uses the generative network G and the discriminative network D to improve each other. G aims to learn the distribution of real data, and D tries to discriminate whether the input data comes from real data or generators. In order to win the game, the generative and discriminative networks continuously optimize themselves, and the performance of G and D is continuously improved in the process of the game, finally reaching the Nash equilibrium, i.e., the generative network G is able to generate samples that the discriminative network D cannot distinguish between true and false, i.e., the generative network G has learned the distribution of real data at this time.

IV. PREDICTION OF ICU MORTALITY

A. XGBoost

Based on CART tree, extreme gradient boosting (XGBoost) was proposed in 2006 [33]. XGBoost is an extension of the gradient boosting algorithm and uses a more advanced regularization technique called L1 and L2 regularization to prevent overfitting. It also incorporates other features such as handling missing values and built-in cross-validation, which makes it a popular choice for competitions and industrial applications. The general idea of XGBoost is to keep adding trees, with each new tree being designed to fit the residuals of the last prediction. The final score (prediction) for each sample equals to the sum of the scores of this sample on each CART tree.

XGBoost uses a number of hyperparameters that can be tuned to optimize its performance, such as the learning rate, the number of trees, the maximum depth of each tree and the regularization parameters.

The whole process of XGBoost can be described in six steps:

Step 1: XGBoost considers the regularization term to prevent overfitting. The objective function consists of two parts: the first part is the sum of loss functions for classification errors, and the second part is regularization term, which helps to restrict the number of leaves and leaf scores.

$$\begin{aligned} L(\phi) &= \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \\ &= \sum_i l(y_i, \hat{y}_i) + \gamma T + \frac{1}{2}\lambda \|w\|^2 . \end{aligned} \tag{5}$$

Step 2: The newly created tree is utilized to match the residual of the previous prediction. After generating t trees, the prediction score of the i-th sample can be written as:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x). \tag{6}$$

The objective function (t-th tree) can thus be rewritten as:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f_t). \tag{7}$$

Step 3: Now, since we need to convert the original objective function to the Euclidean domain function to use the conventional optimization technique, we have to perform Tylor series expansion on loss function in equation 7:

$$L^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t),$$
$$g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial^2 \hat{y}^{(t-1)}}. \tag{8}$$

Step 4: To make the objective function easier to understand, the constant term was left out. The previous round's tree-building loss function has no bearing on our current tree-building efforts. After removing the constant term, we will get the below equation.

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) \\ &= \sum_{i=1}^{n}[g_i w_q(x_i) + \frac{1}{2}h_i w_q^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \\ &= \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T. \end{aligned} \tag{9}$$

Step 5: Take the derivative for the *j*th score of the *t*th tree:

$$\begin{aligned} &\frac{\partial obj^{(t)}}{\partial w_j} = 0 \\ &\Rightarrow w_j^* = -\frac{G_j}{H_j + \lambda} \\ &\Rightarrow \hat{L}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T. \end{aligned} \tag{10}$$

Step 6: Generate all decision trees after the 5 steps.

B. IMPROVED XGBoost BY TREE-STRUCTURED PARZEN ESTIMATOR (TPE-XGBoost)

The optimal expression of the objective function is derived above, that is, the optimal scores of the leaf nodes are known, but the structure of the whole tree is unknown, how to build this tree next?

Commonly used methods for hyperparameter selection include:

- Grid Search [34]: Grid search is a popular method for hyperparameter selection that involves defining a grid of possible hyperparameter values and evaluating the performance of the model for each combination of hyperparameters. Grid search is simple to implement and can be used with any model, but it can be computationally expensive when the number of hyperparameters and their possible values are large.

TABLE 1. The main hyperparameters of XGBoost.

Hyperparameter	Description	Default
colsample_bytree	Control the percentage of columns randomly sampled per tree (each column is a feature).	1
gamma	When a node is split, the node is split only if the value of the loss function drops after the split. gamma specifies the minimum drop in the loss function required to split the node. The larger the value of this parameter, the more conservative the algorithm is.	0
learning_rate (η)	The step size of each iteration in the update is important. Too big is not running accurate, too small is running slow.	0.3
max_depth	The maximum depth of the tree, this value is used to control overfitting.	6
min_child_weight	It determines the sum of the minimum leaf node sample weights.	1
lambda (λ)	L2 regularization term for the weights, this parameter is used to control the regularization part of XGBoost.	1
alpha	L1 canonical coefficients of leaf node weights.	0

- Random Search [35]: Random search is a method that randomly samples hyperparameters from their respective distributions, instead of exhaustively searching through a predefined set of hyperparameters as in grid search. Random search is computationally more efficient than grid search and can sometimes lead to better performance by exploring a larger portion of the hyperparameter space.

Although they are used frequently, the time consumed is enormous and the accuracy is not guaranted. In this study, Tree-structured Parzen estimator (TPE) was used to find the optimal hyperparameter and the optimal XGBoost model was constructed.

The following table lists the main hyperparameters of XGBoost:

Assuming that the set of hyperparameters is m, the minimization loss function $L(m)$ is transformed into:

$$I(m) = max(L^* - L(m), 0). \tag{11}$$

And the expectation of $I(m)$ can be calculated:

$$\begin{aligned} EI(m) &= \int_{-\infty}^{L^*}(L^* - L)p(L|m)dL \\ &= \int_{-\infty}^{L^*}(L^* - L)\frac{p(m|L)p(L)}{p(m)}dL, \end{aligned} \tag{12}$$

where a threshold value of 0.15 is generally set ($P(L < L^*) = 0.15$) and cut into two pieces.

$$p(m|L) = \begin{cases} l(m) & L < L^*, \\ g(m) & c \geq L^*. \end{cases} \tag{13}$$

Divide the historical data into two parts by L^*, taking the part $L < L^*$ as an example.

Suppose there exist n values $(m_1, m_2, \ldots, m_n)$, then the probability density is estimated as:

$$\hat{l}(m) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{m - m_i}{h}), \tag{14}$$

where:

$$K(\frac{m - m_i}{h}) = \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}(\frac{m - m_i}{h})^2), \tag{15}$$

and

$$h = (\frac{4\hat{\sigma}^5}{3n})^{\frac{1}{5}} \approx 1.06\hat{\sigma} n^{-\frac{1}{5}}. \tag{16}$$

Then $p(L)$ can be calculated using beyesian equation:

$$\begin{aligned} p(m) &= \int_R p(m|L)p(L)dL \\ &= \int_{-\infty}^{L^*} p(m|L)p(L)dL + \int_{L^*}^{\infty} p(m|L)p(L)dL \\ &= \nu l(m) + (1 - \nu)g(m). \end{aligned} \tag{17}$$

Substitute this into the expression for the expectation:

$$\begin{aligned} EI(m) &= \frac{\nu c^* - l(m) \int_{-\infty}^{c^*} cp(c)dc}{\nu l(m) + (1 - \nu)g(m)} \\ &\propto (\nu + \frac{g(m)}{l(m)}(1 - \nu))^{-1} \end{aligned} \tag{18}$$

So in order to make the loss function small enough, i.e., the expectation of $I(m)$ is large enough, one needs to maximize $\frac{l(m)}{g(m)}$. For each tree, a node is sampled 100 times under the $l(m)$ distribution using the Parzen estimator. The set of hyperparameters obtained from each sampling can be obtained separately, and then multiplied to obtain the joint probability to calculate $l(m)$ and $g(m)$. When $l(m)/g(m)$ reaches its maximum, m is the optimal hyperparameter in this iteration.

In all, the TPE algorithm works by constructing two separate probability distributions: one to model the hyperparameters that lead to good performance (the "good" distribution), and another to model the hyperparameters that lead to poor performance (the "bad" distribution).

To construct the two distributions, TPE uses a tree-based search algorithm that partitions the hyperparameter space into regions based on their performance. At each iteration, the algorithm evaluates a fixed number of randomly chosen hyperparameter settings and then updates the probability distributions based on the performance of these settings.

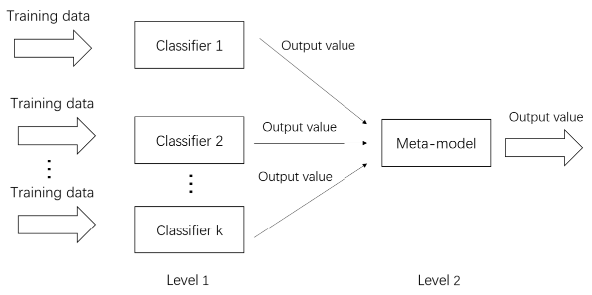

FIGURE 4. The flow chart of the original stacking structure.

The key advantage of TPE is its ability to focus the search on promising regions of the hyperparameter space, rather than exhaustively exploring the entire space. This makes it much more efficient than grid search or random search for high-dimensional problems, where the search space is too large to explore exhaustively. So TPE was applied on XGBoost to work more efficiently.

C. OPTIMIZE BY STACKING-NET

The above text explains the improved model of XGBoost using TPE and we get the proposed TPE-XGBoost model. Generally, setting the number of iterations to 50 can yield models that perform well on the validation set. However, in order to further alleviate overfitting and obtain even better models, this thesis employs the stacking method for ensemble learning.

Stacking is an ensemble learning method used in machine learning to combine multiple classification or regression models to improve their predictive performance. In stacking, the predictions made by several base models are combined using a higher-level model, called the meta-model.

The basic idea of stacking is to train several diverse base models on the same dataset and use their predictions as input to a meta-model, which predicts the final output. This helps to reduce the individual weaknesses of each base model and improves the overall performance. The flowchart of original-stacking is shown in Figure 4:

There are two phases in stacking: training and prediction. In the training phase, the base models are trained on the training data, and their predictions are stored. In the prediction phase, the meta-model is trained on the stored predictions from the training data and used to predict the final output on the test data.

The essence of a stacking model is a hierarchical structure. In this study, we consider using a two-level stacking approach. Suppose we have k base models, Model1_1 through Model1_k, and a secondary model, Model2. The general process is as follows:

Step 1: The base model Model1_1 is trained on the training set.

$$\begin{pmatrix} \vdots \\ X_{\text{train}} \\ \vdots \end{pmatrix} \overset{\text{Model1\_1 Train}}{\Longrightarrow} \begin{pmatrix} \vdots \\ Y_{\text{True}} \\ \vdots \end{pmatrix} \tag{19}$$

After training, the model Model1_1 predicts the label for both the "train" and "test" datasets, resulting in the predicted labels P_1 and T_1, respectively.

$$\begin{pmatrix} \vdots \\ X_{\text{train}} \\ \vdots \end{pmatrix} \overset{\text{Model1\_1 Predict}}{\Longrightarrow} \begin{pmatrix} \vdots \\ P_1 \\ \vdots \end{pmatrix} \tag{20}$$

$$\begin{pmatrix} \vdots \\ X_{\text{test}} \\ \vdots \end{pmatrix} \overset{\text{Model1\_1 Predict}}{\Longrightarrow} \begin{pmatrix} \vdots \\ T_1 \\ \vdots \end{pmatrix} \tag{21}$$

Step 2: For the other base models, Model1_2 through Model1_k, the same training and prediction steps are repeated. The prediction for Model1_i is as follows:

$$\begin{pmatrix} \vdots \\ X_{\text{train}} \\ \vdots \end{pmatrix} \overset{\text{Model1\_i Predict}}{\Longrightarrow} \begin{pmatrix} \vdots \\ P_{\text{i}} \\ \vdots \end{pmatrix} \tag{22}$$

$$\begin{pmatrix} \vdots \\ X_{\text{test}} \\ \vdots \end{pmatrix} \overset{\text{Model1\_i Predict}}{\Longrightarrow} \begin{pmatrix} \vdots \\ T_{\text{i}} \\ \vdots \end{pmatrix} \tag{23}$$

Step 3: The predicted labels $P_1, P_2, \ldots, P_k, T_1, T_2, \ldots, T_k$ are combined to form a new training set and testing set, denoted as Train_2 and Test_2, respectively.

$$\overbrace{\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ P_1 & P_2 & \ldots & P_k \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}}^{\text{Train\_2}} \textit{and} \overbrace{\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ T_1 & T_2 & \ldots & T_k \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}}^{\text{Test\_2}} \tag{24}$$

Step 4: The meta-model, Model2, is then trained on the true labels of the training set using Train_2 as the feature set. The trained Model2 is then used to predict the labels for Test_2, resulting in the final predicted label column for the testing set.

$$\overbrace{\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ P_1 & P_2 & \ldots & P_k \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}}^{\text{Train\_2}} \overset{\text{Model2 Train}}{\Longrightarrow} \begin{pmatrix} \vdots \\ Y_{\text{True}} \\ \vdots \end{pmatrix} \tag{25}$$

$$\overbrace{\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ T_1 & T_2 & \ldots & T_k \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}}^{\text{Test\_2}} \overset{\text{Model2 Predict}}{\Longrightarrow} \begin{pmatrix} \vdots \\ Y_{\text{Predict}} \\ \vdots \end{pmatrix} \tag{26}$$

Y_{Predict} represents the final predicted output of the stacking model. This is the basic and original idea of a two-level stacking model: adding another layer of model on top of the predictions made by different base classifiers, then retraining the model to obtain the final prediction.

Stacking is essentially a straightforward approach, but it can sometimes pose problems when the training and testing sets are not consistent. The problem lies in using the labels trained by the initial model and then retraining with the true labels, which undoubtedly leads to some degree of overfitting on the training set. This may result in reduced generalization ability or effectiveness of the model on the testing set. Therefore, the current problem is how to reduce the overfitting caused by retraining. Here, we generally have two methods: select a simple linear model as the meta-model or use K-fold cross-validation. Here, 5-fold cross-validation is applied to the stacking model.

After 5-fold cross-validation, the final stacking model was applied to the test set to obtain the final results. Compared with other models, stacking is more stable. It can integrate the performance of different base classifiers and use a highly fitted meta-model for prediction. It is particularly useful when the problem is complex and the available data is limited or noisy. We call the final model combined with Modified-GAN, XGBoost with TPE and stacking: MTX-stacking. The prediction for specific data will be explained in the next section.

V. EXPERIMENTS

The global GOSSIS consortium at MIT has developed a valuable resource for assessing illness severity. They have compiled multiple databases to create a comprehensive dataset containing 131,051 ICU admissions in a year. This dataset includes patients' demographic information, test results, chronic health conditions, APACHE scores and other metrics. In total, the dataset contains 185 features.

Predicting mortality is the main target variable. However, the training set is imbalanced, with only 7,915 deaths out of a total 91,713 admissions. Despite this phenomenon, the dataset offers a rich and diverse set of features, making it a valuable resource for machine learning researchers and practitioners in the healthcare field.

Many statistical analysis methods are used to impute missing values, including hot-deck and regression imputation [36]. In this research, features with more than 50% of missing data are entirely removed. For features of continuous type, the median is chosen to fill in; for features of subtype, the plural is chosen to fill in. Finally, a data set without missing values was obtained. The final data set has 91713 samples and 80 features.

The characteristic of each feature is different, such as property, dimension, and magnitude, which makes it impossible to analyze the characteristics directly. Also, as big data grows, the time required also increases. In order to effectively alleviate the gradient explosion problem brought by big data, unitization of features is performed in this research. All the features were rescaled using the min-max normalization technique (Equation 27), where X' is the scaled value, and X is the original value.

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)}. \tag{27}$$

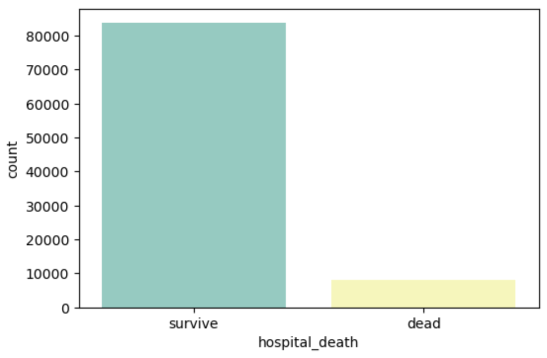

FIGURE 5. **The distribution of hospital_death.**

It normalizes the data using the minimum and maximum of the original data, and maps the original data into the area of [0,1].

The dataset used in this study is a common imbalance data. As is shown in the Figure 5, in this data set, the number of people who survived within 24 h was much larger than the number of people who died. So it is necessary to use method to balance the dataset so that the model has a good generalization capability.

First, data on patients who died within 24 h (data from minority class) were selected. These data were used to train the Modified-GAN model. The Modified-GAN structure consists of a generator and a discriminator. Both the generator and the discriminator are three-layer fully connected networks.

Suppose $z \sim N(0, 1)$ and $z \in (0, 1)$. The Modified-GAN approach involves introducing random noise into the generator, which is subsequently repeated multiple times to generate the requisite number of samples of the minority class while also performing feature rounding (classification) for the 0-1 type features in the generated data. This initial data generation process represents the outcome of the 0th iteration. Subsequently, the Modified-GAN discriminator is leveraged to determine the degree of proximity between the generated and actual data. Through iterative procedures, the distribution of the generated data gradually converges with that of the real data, ultimately resulting in synthetic samples that are indistinguishable from the actual data.

Modified-GAN is conducted for 500 rounds. Two features are selected to visualize when the iterations are 0, 100, 200, 300, 400 and 500. The original data was random, so the generative data's distribution is quite far from the real data (Picture (a) in Figure 6). After 400 rounds, the generative data is very close to real data. During 400-500 iterations, the generated data tends to stabilize, meaning that the generated data no longer differs much from the real data. So finally 500 iterations are chosen to be the strategy which helps balancing the dataset.

If the model is trained directly with the training set, it will easily cause overfitting. Therefore, the training set after processing data imbalance is sliced again: 85% of the original training set is used as the training set and 15% of the original training set is used as the validation set (to adjust the hyperparameters). The test set was divided into two equal parts: the public and private sets, as shown in Figure 7. Both of these sets are completely new and unseen by the model, making them valuable resources for evaluating the model's performance.

The evaluation metric used for assessing the model's performance is the area under the receiver operating characteristic curve (AUC) [37]. The AUC is calculated by comparing the anticipated mortality results to the values of reality. This is a commonly used metric in healthcare-related machine learning tasks and provides a comprehensive assessment of the model's performance.

The XGBoost model is constructed on the training set and the hyperparameters are adjusted using TPE in the validation set. TPE helps to improve XGBoost model. A total of 50 iterations were performed in this thesis, and the following table and figure shows the results of the iterations. To evaluate the data augmentation performance of the Modified-GAN in comparison to the traditional SMOTE algorithm, both methods are applied to the same training set to create new

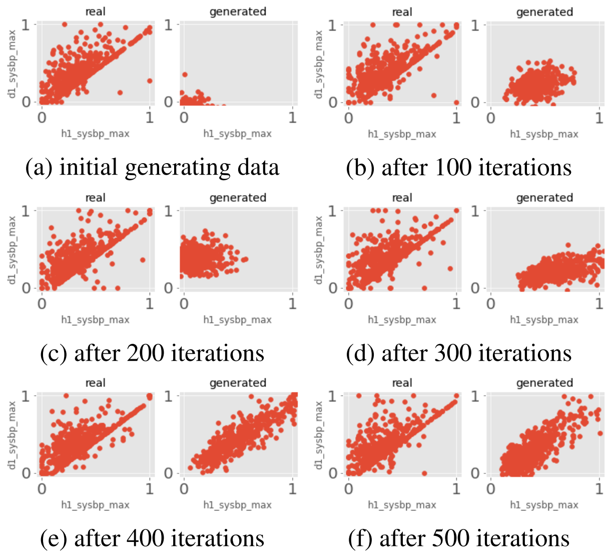

FIGURE 6. **The visualization of Modified-GAN.**

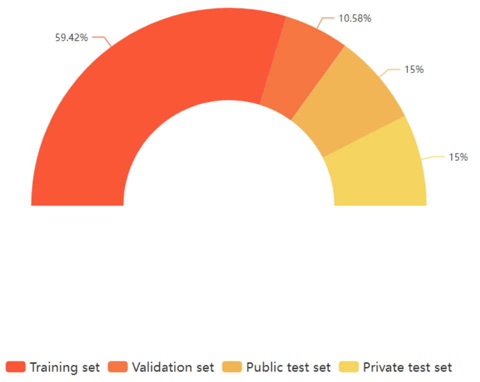

FIGURE 7. **Proportion of training, validation, test set.**

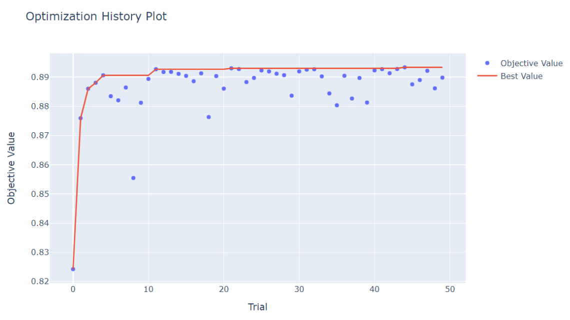

FIGURE 8. Training model based on Modified-GAN's data: improve XGBoost using Tree-structured Parzen estimator.

TABLE 2. The AUC value of best model using initial data set or processed data set.

Data set	Private score	Public score
initial data set	0.832	0.841
SMOTE-processed data set	0.872	0.878
Modified-GAN-processed data set	**0.894**	**0.896**

training sets for each method, and the parameters are adjusted using the Xgboost+Tree-structured Parzen estimator method.

According to Figure 6, after 500 iterations, the initial Modified-GAN's training data can be expressed by Z. Z is divided in final training data and validation data. Furthermore, XGBoost with TPE is constructed using the above data. After 50 iterations, the best AUC value is 0.8933.

According to Figure 8, the red lineplot means the best value (AUC) and it's obvious that the value almost stops increasing when it rises to 0.89, which means that after 50 rounds, the model has reached a relatively stable level, indicating the usefulness of Modified-GAN in data pre-processing.

Finally, the optimal model is applied to the ''black-box'' prediction set by tuning the parameters on the original data and the enhanced data of the above two methods (SMOTE and Modified-GAN), and the final results are shown in the Table 2.

The findings reveal that the approach to handling data imbalance is a critical factor in improving prediction accuracy. The initial accuracy levels of 0.832 and 0.841, respectively, without any data processing, increase significantly when the XGBoost prediction model is applied after processing the imbalanced data through either SMOTE or Modified-GAN. Notably, the Modified-GAN method yields a more substantial improvement in model accuracy, thereby underscoring its superiority in generating stable and realistic data. Consequently, we select the Modified-GAN method for handling data imbalance.

After tuning with TPE, the five best performing base models on the validation set are selected. According to Figure 4, in this research, the five models are used as the base models of the first layer $Classifier1, Classifier2, \ldots, Classifier5$, and for each base classifier, the training set is cross-validated in five folds. After cross-validation, the predictions on the training set are combined and used as the output of the first layer. And the output of these five models are merged to generate a 5-column prediction output O_{train_1}. This output will be fed to the second layer model (logistic regression with default parameters) in order to alleviate the overfitting situation to some extent. The logistic regression is used to train O_{train_1} and output the prediction (classification) O_{train_2} of the stacking model on the training set, which is the training process of the stacking model. After training, the whole stacking-model is applied to the prediction data set to classify.

TABLE 3. Comparison of MTX-stacking model and some classical methods on ICU patients' dataset.

Scoring System	Private AUC Score	Public AUC Score
APACHE IV	0.868	0.871
Deep Learning	0.895	0.896
H20 AutoML	0.900	0.902
Only TPE-XGBoost+stacking	0.837	0.846
SMOTE+TPE-XGBoost+stacking	0.879	0.882
Our method: MTX-stacking model	**0.909**	**0.912**

To verify the advantage of this model, we compare the MTX-stacking model with several classical or state-of-the-art models.

Our final method, the MTX-stacking model, outperforms all other techniques considered in this study. Other top-performing techniques include H2O AutoML, which utilizes an ensemble learning approach based on several models, achieving AUC scores of 0.900 and 0.902 on the private and public sets, respectively. Additionally, the DEEP Learning technique, which employs a feedforward neural network, obtained an AUC score of 0.895 and 0.896. Finally, the APACHE IV, a well-known scoring system based on logistic regression, achieved an AUC score of 0.868 and 0.871.

Also, we conducted a comparative analysis of MTX-stacking model with several stacking models that utilized various data preprocessing techniques. The results indicate that data preprocessing is a critical step in improving the AUC score. We observed that without data preprocessing, the AUC score was comparatively lower. However, after applying the SMOTE method, the AUC score improved significantly, achieving a score of 0.879 and 0.882. Nonetheless, the AUC score of these models still could not outperform Deep Learning and H20 AutoML. As a result, we employed Modified-GAN as a data preprocessing method to process TPE+XGBoost with stacking net, ultimately leading to the development of the MTX-stacking model, which achieved a better prediction performance.

To sum up, the MTX-stacking model proposed in this research achieved an AUC of 0.912 on the public dataset and 0.909 on the private dataset for predicting ICU mortality. The model outperforms classical methods, demonstrating its better prediction performance on this dataset and addressing the issue of low prediction accuracy to some extent. Moreover, as explained earlier, the stacking framework used in this model can alleviate overfitting. Therefore, this model can be used for predicting ICU mortality and contribute to mitigating the shortage of medical resources.

The SHAP method is applied to the MIT's GOSSIS dataset and the MTX-stacking model. For feature j,

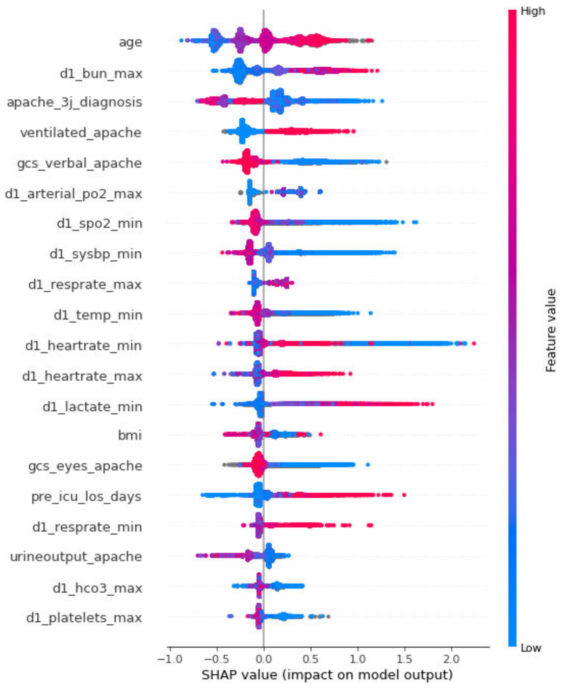

FIGURE 9. **The top twenty shapley values on the MTX-stacking model. The dots on this graph represent the impact of these features on the model predictions.**

its value:

$$\Phi_j = \sum\nolimits_{S\subseteq[\{x_1,x_2,\ldots,x_k\}\setminus\{x_j\}]}\eta * (f_x(S \cup \{x_j\}) - f_x(S)), \quad (28)$$

$$\eta = \frac{|S|!(k-|S|-1)!}{k!}, \quad (29)$$

where $\{x_1, x_2, \ldots, x_k\}$ represents the set of all input features, k is the number of features, $\{x_1, x_2, \ldots, x_k\} \setminus \{x_j\}$ represents the set of all possible input features excluding $x_j, f_x(S)$ is the prediction of feature subset S.

The weighting variable η can be understood as:

- Denominator $k!$: there are $k!$ combinations of k features in any ordering.
- Molecular $|S|!(k - |S| - 1)!$: after determining the subset S, k features in a particular ordering have $|S|!(k - |S| - 1)!$ kinds of combination cases. When the subset S is determined, the set of features should be $\{x_1, \ldots, x_{|S|}, x_j, x_{|S|+2}, \ldots, x_p\}$ and the subset $\{x_1, \ldots, x_{|S|}\}$ itself has $|S|!$ of sequential combinations to be followed immediately by the feature j,then the remaining features $\{x_{|S|+2}, \ldots, x_p\}$ have $(k - |S| - 1)!$ combinations, then after determining the subset S there are $|S|!(k - |S| - 1)!$ combinations of cases.
- So η is the percentage of feature combination cases of subset S, and the sum of the percentage of feature combination cases of all possible subsets S is equal to 1.

Figure 9 displays the top twenty features with the highest shapley values on the MTX-stacking model. Shapley values provide important insights into how different features contribute to the model's predictions and can be used to identify the most important features for accurate predictions.

The features displayed in Figure 9 are sorted from most important to least important, with the most important feature at the top. The "shapley value (impact on model)" is the horizontal location that indicates whether the feature has a positive or negative impact on the model's prediction. The color of each point represents the value of the corresponding feature for the given observation, with red indicating high values and blue indicating low values. By examining the plot, we can gain insights into which features are most important for the model's predictions and how they contribute to the overall performance of the model.

It is vividly shown that the most significant feature is the age. The output is similar to the real-life situation: the older the patient (red dots), the more likely the patient is to die within 24h (positive shapley value), while the younger patient (blue dots) has a relatively lower hospital_death (negative shapley value). The second-rank feature with high significance is "d1_bun_max", which means the highest blood urea nitrogen concentration of the patient in their serum or plasma during the first 24 hours of their unit stay. It is obvious that blood urea nitrogen concentration increases the predicted mortality risks (when the feature value is high, the shapley value is positive), which means that the blood urea nitrogen concentration of the patient is also a feature worth focusing on. If this feature is quite high, the patient may be hard to survive according to the MIT's GOSSIS dataset.

The "ventilated_apache" is also important, which is a binary variable indicating whether the patient is invasively ventilated at the time of the highest scoring arterial blood gas using the oxygenation scoring algorithm, including any mode of positive pressure ventilation delivered through a circuit attached to an endo-tracheal tube or tracheostomy. When the patient is mechanically ventilated (ventilated_apache=1, the red dots), the shapley value is positive, which means that all patients' mortality rates rise as a result of ventilation. Some Glasgow Coma Scale features (including "gcs_verbal_apache" and "gcs_eyes_apache") are considered as important factors in the model: these features indicates a person's level of consciousness using verbal and eyes. A low score (blue dots) is associated with positive shapley value, which indicates those with little consciousness will be more possible to be close to death.

Finally, some features associated with patients' vital data are contributing to the model. The "d1_spo2_min" and "d1_sysbp_min" indicate the patient's lowest peripheral oxygen saturation during the first 24 hours of their unit stay and the patient's lowest systolic blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured respectively. They all get positive shapley values when they are not too high, indicating that a patient with low oxygen and blood pressure has higher predicted risk.

VI. CONCLUSION

This research endeavors to construct an ensemble learning model for predicting the survival of intensive care unit (ICU) patients within a 24-hour timeframe, utilizing MIT's GOSSIS dataset as a case study.

Firstly, considering the issue of data imbalance, we employed the classical SMOTE method to address the original dataset. Additionally, we introduced a novel approach called Modified-GAN to tackle data imbalance and demonstrated the convergence of the Modified-GAN method. Secondly, following the data preprocessing stage, we applied the XGBoost boosting method for classification purposes. However, due to the modest prediction accuracy of the original XGBoost model, we incorporated the tree-structured Parzen estimator (TPE) to optimize the hyperparameters of XGBoost. Through a comprehensive evaluation of various iterations, we identified the TPE-XGBoost model that exhibited the most favorable performance on the validation set. Subsequently, the predictions of TPE-XGBoost were compared on the original data, SMOTE-preprocessed data, and Modified-GAN-preprocessed data in the final black-box test set. The results demonstrated that Modified-GAN + TPE-XGBoost yielded the most promising outcome, affirming the effectiveness of the proposed Modified-GAN method in addressing data imbalance. Thirdly, to further mitigate overfitting and enhance the model's performance, we employed the stacking method, where the top five XGBoost models after TPE were employed as base learners, with logistic regression serving as the meta-model to generate the final output. This comprehensive model, referred to as the MTX-stacking model, achieved a high area under the curve (AUC) of 0.909 and 0.912 on the private and public test sets, respectively. Moreover, we conducted a comparative analysis with commonly used or state-of-the-art methods, including APACHE IV, DEEP LEARNING, and H2O AUTOML, revealing the superior performance of our final model.

Finally, to provide interpretability to the MTX-stacking model, we employed the SHAP (Shapley Additive exPlanations) method and observed that the generated explanations aligned with clinical expectations, enhancing the model's transparency and trustworthiness.

Although the results obtained in this research are more powerful than other compared model, there exists some potential prospects:

(1) Our aim was to construct the most accurate model by AUC, but there existed some other factors which could result in the final death of patients. Moreover, the selection of variables in this research was simple. We did not consider the correlation between variables.

(2) The Modified-GAN method provides an idea of data augmentation, but there is still room for improvement within it: the commonly used 500 iterations are set as the gate value in this research, and perhaps an optimization algorithm can be used to determine the optimal number of iterations to achieve optimal data augmentation.

REFERENCES

[1] Y. Ding, Y. Wang, and D. Zhou, "Mortality prediction for ICU patients combining just-in-time learning and extreme learning machine," *Neurocomputing*, vol. 281, pp. 12–19, Mar. 2018.

[2] W. A. Knaus, E. A. Draper, D. P. Wagner, and J. E. Zimmerman, "APACHE II: A severity of disease classification system," *Crit. Care Med.*, vol. 13, no. 10, pp. 818–829, Oct. 1985.

[3] J. R. Le Gall, "A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study," *J. Amer. Med. Assoc.*, vol. 270, no. 24, pp. 2957–2963, Dec. 1993.

[4] D. Sun, H. Ding, C. Zhao, Y. Li, J. Wang, J. Yan, and D. W. Wang, "Value of SOFA, APACHE IV and SAPS II scoring systems in predicting short-term mortality in patients with acute myocarditis," *Oncotarget*, vol. 8, no. 38, pp. 63073–63083, Sep. 2017.

[5] R. A. Taylor, J. R. Pare, A. K. Venkatesh, H. Mowafi, E. R. Melnick, W. Fleischman, and M. K. Hall, "Prediction of in-hospital mortality in emergency department patients with sepsis: A local big data-driven, machine learning approach," *Academic Emergency Med.*, vol. 23, no. 3, pp. 269–278, Mar. 2016.

[6] K. Chen, C. Gao, Q. Zhou, W. Li, and Z. Lin, "Predictors of in-hospital mortality for sepsis patients in intensive care units," *Int. J. Clin. Exp. Med.*, vol. 9, no. 2, pp. 34–4029, 2016.

[7] P. T. N. Thao, T. T. Tra, N. T. Son, and K. Wada, "Reduction in the IL-6 level at 24 H after admission to the intensive care unit is a survival predictor for Vietnamese patients with sepsis and septic shock: A prospective study," *BMC Emergency Med.*, vol. 18, no. 1, pp. 1–7, Dec. 2018.

[8] M. M. Ros, H. J. van der Zaag-Loonen, J. G. M. Hofhuis, and P. E. Spronk, "Survival prediction in severely ill patients study—The prediction of survival in critically ill patients by ICU physicians," *Crit. Care Explor.*, vol. 3, no. 1, p. e0317, 2021.

[9] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, "Acute physiology and chronic health evaluation (APACHE) IV: Hospital mortality assessment for today's critically ill patients," *Crit. Care Med.*, vol. 34, no. 5, pp. 1297–1310, May 2006.

[10] S. Lemeshow, "Mortality probability models (MPM II) based on an international cohort of intensive care unit patients," *JAMA: J. Amer. Med. Assoc.*, vol. 270, no. 20, pp. 2478–2486, Nov. 1993.

[11] A. J. Hussain, P. Fergus, H. Al-Askar, D. Al-Jumeily, and F. Jager, "Dynamic neural network architecture inspired by the immune algorithm to predict preterm deliveries in pregnant women," *Neurocomputing*, vol. 151, pp. 963–974, Mar. 2015.

[12] K.-J. Kim and S.-B. Cho, "Prediction of colon cancer using an evolutionary neural network," *Neurocomputing*, vol. 61, pp. 361–379, Oct. 2004.

[13] P. Chen, L. Yuan, Y. He, and S. Luo, "An improved SVM classifier based on double chains quantum genetic algorithm and its application in analogue circuit diagnosis," *Neurocomputing*, vol. 211, pp. 202–211, Oct. 2016.

[14] A. T. Azar and S. A. El-Said, "Performance analysis of support vector machines classifiers in breast cancer mammography recognition," *Neural Comput. Appl.*, vol. 24, no. 5, pp. 1163–1177, Apr. 2014.

[15] A. T. Azar and S. M. El-Metwally, "Decision tree classifiers for automated medical diagnosis," *Neural Comput. Appl.*, vol. 23, nos. 7–8, pp. 2387–2403, Dec. 2013.

[16] O.-P. Ryynänen, E. J. Soini, A. Lindqvist, M. Kilpeläinen, and T. Laitinen, "Bayesian predictors of very poor health related quality of life and mortality in patients with COPD," *BMC Med. Informat. Decis. Making*, vol. 13, no. 1, pp. 1–10, Dec. 2013.

[17] Z. Cui, Y. Wang, X. Gao, J. Li, and Y. Zheng, "Multispectral image classification based on improved weighted MRF Bayesian," *Neurocomputing*, vol. 212, pp. 75–87, Nov. 2016.

[18] M. Aczon, D. Ledbetter, L. Ho, A. Gunny, A. Flynn, J. Williams, and R. Wetzel, "Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks," 2017, *arXiv:1701.06675*.

[19] T. Alves, A. Laender, A. Veloso, and N. Ziviani, "Dynamic prediction of ICU mortality risk using domain adaptation," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1328–1336.

[20] M. Last, O. Tosas, T. Gallo Cassarino, Z. Kozlakidis, and J. Edgeworth, "Evolving classification of intensive care patients from event data," *Artif. Intell. Med.*, vol. 69, pp. 22–32, May 2016.

[21] J. G. Klann, P. Szolovits, S. M. Downs, and G. Schadow, "Decision support from local data: Creating adaptive order menus from past clinician behavior," *J. Biomed. Informat.*, vol. 48, pp. 84–93, Apr. 2014.

[22] Y. Zhang and P. Szolovits, "Patient-specific learning in real time for adaptive monitoring in critical care," *J. Biomed. Informat.*, vol. 41, no. 3, pp. 452–460, Jun. 2008.

[23] C. G. Enright and M. G. Madden, "Modelling and monitoring the individual patient in real time," in *Foundations of Biomedical Knowledge Representation: Methods and Applications*, 2015, pp. 107–136.

[24] N. Kasabov and Y. Hu, “Integrated optimisation method for personalised modelling and case studies for medical decision support,” *Int. J. Funct. Informat. Personalised Med.*, vol. 3, no. 3, pp. 236–256, 2010.
[25] X. Li and Y. Wang, “Adaptive online monitoring for ICU patients by combining just-in-time learning and principal component analysis,” *J. Clin. Monitor. Comput.*, vol. 30, no. 6, pp. 807–820, Dec. 2016.
[26] M. Liu, C. Guo, and S. Guo, “An explainable knowledge distillation method with XGBoost for ICU mortality prediction,” *Comput. Biol. Med.*, vol. 152, Jan. 2023, Art. no. 106466.
[27] Z. Ye, S. An, Y. Gao, E. Xie, X. Zhao, Z. Guo, Y. Li, N. Shen, J. Ren, and J. Zheng, “The prediction of in-hospital mortality in chronic kidney disease patients with coronary artery disease using machine learning models,” *Eur. J. Med. Res.*, vol. 28, no. 1, pp. 1–13, Jan. 2023.
[28] C. Guo, M. Liu, and M. Lu, “A dynamic ensemble learning algorithm based on K-means for ICU mortality prediction,” *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107166.
[29] N. El-Rashidy, S. El-Sappagh, T. Abuhmed, S. Abdelrazek, and H. M. El-Bakry, “Intensive care unit mortality prediction: An improved patient-specific stacking ensemble model,” *IEEE Access*, vol. 8, pp. 133541–133564, 2020.
[30] N. Ren, X. Zhao, and X. Zhang, “Mortality prediction in ICU using a stacked ensemble model,” *Comput. Math. Methods Med.*, vol. 2022, Nov. 2022, Art. no. 3938492.
[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[32] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014, *arXiv:1406.2661*.
[33] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[34] Y. Bao and Z. Liu, “A fast grid search method in support vector regression forecasting time series,” in *Proc. Intell. Data Eng. Automated Learn. 7th Int. Conf.* Burgos, Spain: Springer, Sep. 2006, pp. 504–511.
[35] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 2, pp. 1–25, 2012.
[36] C. Curley, R. M. Krause, R. Feiock, and C. V. Hawkins, “Dealing with missing data: A comparative exploration of approaches using the integrated city sustainability database,” *Urban Affairs Rev.*, vol. 55, no. 2, pp. 591–615, Mar. 2019.
[37] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997.

MINGYI WEI received the B.S. degree in data science from the Zhejiang University of Finance and Economics. He is currently pursuing the master’s degree in mathematics with the Department of Statistics and Data Science, Southern University of Science and Technology. His research interests include processing data imbalance and categorical data analysis.

ZHEJUN HUANG received the Ph.D. degree in mathematics from the University of New South Wales, Australia, in 2017. He was a Postdoctoral Researcher with the University of New South Wales and the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He is currently a Research Assistant Professor with the Department of Statistics and Data Science, Southern University of Science and Technology. He has published over ten papers in international high-level journals and conferences. His research interests include public safety and emergency management, disaster risk analysis, disaster response decision making, disaster risk reduction and management, the decision and control of autonomous vehicles, and reaction diffusion models.

DIPING YUAN is currently a professor and a doctoral supervisor. He is also a specially appointed Professor with the China University of Mining and Technology, where he also serves as the Dean of the Shenzhen Research Institute. He was the former Deputy Director of the Key Laboratory of Firefighting Rescue Technology, Ministry of Public Security. He has led one project in the National Key Research and Development Program: Research and Application Demonstration of Public Safety Risk Prevention and Control System in Emerging Megacities in Southern China and one project in the Guangdong Province Key Research and Development Program: Key Technologies and Equipment for Monitoring, Early Warning, and Firefighting and Rescue in Super High-Rise Building Fires and Application Demonstration. His research interests include big data applications in urban public safety and emergency management, safety risk monitoring and early warning, emergency intelligent decision-making and simulation, and smart firefighting.

He is a selected member of the National Talent Engineering Project, a young and middle-aged expert with outstanding contributions at the national level, a national leading talent in Shenzhen, and a national advanced individual in public security science and technology. He enjoys special allowances from the State Council and the Ministry of Public Security. He is also the Vice President of Shenzhen Emergency Management Association and the Director of the Information Committee, a Standing Committee Member of the Community Safety Committee of the Chinese Emergency Management Society, and a member of the Firefighting Rescue Technology Professional Committee and the Academic Committee of the Chinese Fire Protection Association.

LILI YANG received the master’s and Ph.D. degrees in computer science from Loughborough University, U.K.

She was a Professor with the School of Business and Economics, Loughborough University. She is currently with the Department of Statistics and Data Science, Southern University of Science and Technology, China. Her former institution, Loughborough University, has consistently ranked among the top ten in the U.K. for many years. During her time in the U.K., she led her team to complete a large number of research projects, receiving project funding totaling over three million pounds in just a few years. She has published many articles in top-tier international journals, and she has long been engaged in research on model building and emergency systems. Combining her years of teaching and research experience in management science with her computer background, she has made fruitful achievements in the field of emergency management, gaining high international reputation, and often being invited to give keynote speeches at international conferences around the world. She is a rare research expert in the field of information systems and emergency management in the U.K., who has made outstanding contributions. She was invited by senior officials of the British Cabinet to give a report on her research achievements to the government in London. The U.K.’s largest media organization, the BBC, also invited her to speak about her research achievements. Her research has had a profound impact on the global research community.

Dr. Yang is a fellow of the British Computer Society and a registered IT professional.