Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

An optimized XGBoost based diagnostic system for effective prediction of heart disease

Kartik Budholiya *, Shailendra Kumar Shrivastava, Vivek Sharma

Computer Science & Engineering, Samrat Ashok Technological Institute, Vidisha, Madhya Pradesh, India

ARTICLE INFO

Article history:
Received 5 March 2020
Revised 24 September 2020
Accepted 17 October 2020
Available online xxxx

Keywords:
XGBoost
Bayesian Optimization
Categorical feature encoding
Heart Disease
Prediction

ABSTRACT

Researchers have created several expert systems over the years to predict heart disease early and assist cardiologists to enhance the diagnosis process. We present a diagnostic system in this paper that utilizes an optimized XGBoost (Extreme Gradient Boosting) classifier to predict heart disease. Proper hyper-parameter tuning is essential for any classifier's successful application. To optimize the hyper-parameters of XGBoost, we used Bayesian optimization, which is a very efficient method for hyper-parameter optimization. We also used One-Hot (OH) encoding technique to encode categorical features in the dataset to improve prediction accuracy. The efficacy of the proposed model is evaluated on Cleveland heart disease dataset and compared it with Random Forest (RF) and Extra Tree (ET) classifiers. Five different evaluation metrics: accuracy, sensitivity, specificity, F1-score, and AUC (area under the curve) of ROC charts were used for performance evaluation. The experimental results showed its validity and efficacy in the prediction of heart disease. In addition, proposed model displays better performance compared to the previously suggested models. Moreover, our proposed method reaches the high prediction accuracy of 91.8%. Our results indicate that the proposed method could be used reliably to predict heart disease in the clinic.

© 2020 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Healthcare organizations (hospitals, medical centers) face a major challenge in providing quality services at affordable costs. Quality service includes medical evaluation and effective treatments being delivered correctly. An expert system based on machine learning can reduce the medical test's associated costs, and it also enhances the process of diagnosis. In the previous studies, researchers have developed various diagnostic systems for the prediction of heart disease based on different techniques (Samuel et al., 2017, 2013; Alizadehsani et al., 2012; Arabasadi et al., 2017; Polat et al., 2007; Das et al., 2009; Anooj, 2012; Babaoglu et al., 2010; Olaniyi et al., 2015; Abushariah et al., 2014; Manogaran et al., 2018; Özşen and Güneş, 2009; Ali et al., 2019).

Motivated by the development of various diagnostic systems to lower heart disease diagnostic barriers and improve predictive accuracy, we are trying to develop a diagnostic system based on XGBoost (Extreme Gradient Boosting) Classifier. XGBoost Algorithm is the advanced implementation of gradient boosting algorithm and has been successfully applied to some studies (Xia et al., 2017; Zieba et al., 2016). It is capable of handing regularization and overfitting-underfitting issues. It evaluates its efficacy in classification problem using accuracy and AUC of ROC chart for a set of hyper-parameters values given by the user. The efficacy of a classifier derived from this model depends heavily on the number of parameters to be modified by the user; these are commonly referred to as hyper-parameters and their values can significantly affect a classifier's efficiency. The proper adjustment of a machine learning algorithm's hyper-parameters requires knowledge of the algorithm, practice, and usually check and error. However, this task can be presented as an optimization problem in order to obtain the best potential solution systematically and effectively, given an appropriate objective function capturing the classifier's predictive performance in terms of hyper-parameter configurations. Several approaches, such as Manual, Grid Search (GS), Random Search (RS) (BergstraandY and Bengio, 2012; Mantovani et al., 2015) and Bayesian Optimization (Snoek et al., 2012), have been effective in

* Corresponding author at: Shakti Bhawan, Sai Enclave, Vidisha, Madhya Pradesh 464001, India.
E-mail address: kartikbudholiya@outlook.com (K. Budholiya).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.10.013
1319-1578/© 2020 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Please cite this article as: K. Budholiya, Shailendra Kumar Shrivastava and V. Sharma, An optimized XGBoost based diagnostic system for effective prediction of heart disease, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2020.10.013

solving the problem of hyper-parameter tuning. Manual tuning of hyper-parameters by the user is very ineffective and unpromising approach, if there is substantial number hyper-parameters values to be taken into consideration. Grid and Random search require long run times because they waste time evaluating unpromising areas of the search space. These methods hardly rely on any information learned by the model during previous optimizations. Bayesian Optimization (Wu et al., 2019) on the other hand constantly learns from previous optimizations to find the best-optimized parameter list and also requires fewer samples to learn or derive the best values. Bayesian optimization captured the attention of scientists in machine learning as an effective tool for hyper-parameter tuning, particularly for complex models (Bergstra et al., 2011; Thornton et al., 2013; Mockus, 1994; Lizotte et al., 2007; Brochu et al., 1012; González et al., 1505). Experiments were carried out on Cleveland heart disease dataset taken from the University of California, Irvine (UCI) online machine learning, and data mining repository. Using this dataset, we propose an Optimized XGBoost based model for effective prediction of heart disease.

This paper's key contributions are as follows:

We designed a diagnostic system for predicting Heart Disease, which includes categorical features encoding and Heart Disease prediction.

We proposed an Optimized XGBoost based Heart Disease prediction model in which One-Hot (OH) encoding technique to encode categorical features is used at data pre-processing step and Bayesian Optimization technique is used for XGBoost hyper-parameter tuning to improve the prediction results.

We compared the final proposed model with other machine learning models. The results demonstrate that the proposed model shows better performance.

The rest of the paper is organized as follows. In Section 2, we discuss the details of the material and methods that are used in this paper. In Section 3, we describe the methodology, and the experimental results and discussion are shown in Section 4. Finally the conclusion is drawn in Section 5.

2. Background

This section provides a brief description of the dataset, previous methods and relevant concepts used in this paper.

2.1. Heart disease dataset description

For our study, the Cleveland heart disease dataset obtained from the University of California, Irvine (UCI) online machine learning, and data mining repository (Cleveland, 0000). The dataset contains 303 instances of subject records, but 6 of these contained missing class values. The dataset contains 76 variables per subject, but past studies indicated that 13 features are effective in detecting Heart Disease. The dataset has both categorical and numerical features; we list them in Table 1. The purpose of the dataset is to predict the presence or absence of heart disease given the results of various medical tests carried out on a subject. The "num" variable in the dataset shows the presence or absence of heart disease in the subject. "num" variable has values from 0 (no presence) to 4. Previous studies on Cleveland dataset have attempted to distinguish presence (values 1, 2, 3, 4) of heart disease from absence (value 0) of heart disease.

2.2. Previous studies

Cleveland heart disease dataset was used in the prediction of heart disease by various classification models, and they have reported high prediction accuracy in the last ten years. Das et al. (2009) developed a technique for diagnosing heart disease using SAS base software In the middle of the proposed system is a neural network (multi-layer feedforward neural network) ensemble process. This ensemble-based approach creates new models from several predecessor models by integrating the posterior probabilities or the expected values. Their proposed model achieved classification accuracy of 89.01%. Anooj (2012) presented a weighted Fuzzy Clinical Decision Support System (CDSS) for the diagnosis of heart disease. Their proposed clinical decision support system has two phases. First, to obtain the weighted fuzzy rules, they used the mining method, attribute selection and attribute weighting method. Then, according to the weighted fuzzy rules and selected attributes, the fuzzy system is designed. An experiment conducted using the Cleveland heart disease dataset on this proposed system has a predictive accuracy of 62.35%. Samuel et al. (2017) used the Fuzzy analytical hierarchy process (Fuzzy AHP) technique to measure global weights of attributes based on their individual contribution. Then the global weights representing the attribute contributions were applied to train an ANN classifier to predict heart disease. Their proposed hybrid method (ANN and Fuzzy AHP) was implemented on MATLAB. Ali et al. (2019) introduced a diagnostic system which stacks two support vector machine (SVM) models to predict the heart disease. The first SVM model is L1 regularized and has linear kernel. And the second SVM model is L2 regularized. They proposed a hybrid grid search algorithm (HGSA) which can optimize the two models simultaneously. Their proposed model has 92.22% accuracy. They used Python packages for simulating the experiments. It is clear from (Ali et al., 2019) that we need to address regularization and overfitting-underfitting problem. Although, previous methods provided substantial improvement in the results, however, there still exists some techniques which remain unexplored: (i) none of the aforementioned approaches explored tree-based machine learning algorithms like XGBoost, which has a built-in parameters for regularization and handing overfitting-underfitting; (ii) the existing approaches did not use categorical feature encoding techniques to encode categorical features in the heart disease dataset; (iii) until now, these previous methods did not used Bayesian optimization as a hyper-parameter optimization technique for optimizing machine learning models, which is very efficient compared to exhaustive search strategy.

Table 1
Features of Heart Disease dataset.

Feature No.	Feature Description	Type
1	Age	Numerical
2	Sex	Categorical
3	Chest pain type	Categorical
4	Resting blood pressure	Numerical
5	Serum Cholesterol	Numerical
6	Fasting blood sugar	Categorical
7	Resting electrocardiograph results	Categorical
8	Maximum heart rate achieved	Numerical
9	Exercise-induced angina	Categorical
10	St-depression	Numerical
11	St-slope	Categorical
12	Number of major vessels	Categorical
13	Thalassemia	Categorical
14	num (target variable)	Categorical

2.3. Xgboost (Extreme gradient Boosting)

XGBoost (Chen and Guestrin, 2016)(Extreme Gradient Boosting) and Gradient Boosting (Friedman, 2001) (GB) are both ensemble tree methods that use the gradient descent architecture to boost weak learners. XGBoost, however, strengthens the basic GB

architecture through system optimization and algorithmic improvements. Initially carried out by Chen and Guestrin (2016) and continued by other developers. XGboost is a package that belongs to the Distributed Machine Learning Community (DMLC). GB is a stage-wise additive modeling (Friedman, Oct. 2001). First, a weak classifier is fit to the data. It fits one more weak classifier to improve the performance of current model, without making change in the previous classifier, and this process continues. Each new classifier has to consider where the previous classifiers were not performing well. General Boosting algorithm flow is shown in Fig. 1. First, we estimate y1 by fitting the data to a decision tree, and the second tree is fitted based on the residual from the previous step which is y-y1 and this process continues. The algorithm error can be decreased efficiently by analogy.

The GB and XGBoost algorithm are summarized as follows after carefully reading the work in Chen and Guestrin (2016), Friedman (2001) and Xia et al. (2017). Assume we've got a dataset D.

$$\mathrm{D} = \mathrm{x}\ ;\ \mathrm{y},\ |\mathrm{D}| = n, x \in R^m, y \in R$$

n is the number of samples, m the number of features and x and y denote the features and the target variable of the dataset. Our heart disease dataset contains n = 303 observations and m = 13 features. In GB for dataset D, k trees predicted scores sum is the prediction result which is calculated by a function called K additive function, as shown in Eq. (1):

$$\hat{y}_i = \sum_{k=1}^{k} f_k(x_i), f_k \in F \tag{1}$$

where, $\hat{y}_i$ represent the prediction of the i-th instance at the k-th boost, x_i represent the i-th instance sample of the training dataset. The value of the k-th tree is $f_k(x_i)$ and all the values of the decision trees are represented by the function F . GB minimizes the loss function L_k which is defined in Eq. (2).

$$L_k = \sum_{i=1}^{n} L(\hat{y}_i, y_i) \tag{2}$$

Since both GB and XGBoost are decision tree based algorithm, multiple tree-related hyper-parameters, including subsample and max_depth, are used to reduce the overfitting issue and to improve the model performance. In addition, the learning_rate manages the trees weighting which are added to the model and it is also used to lower the model's rate of adaptation to the training data. XGBoost also defines these hyper-parameters and their descriptions can be found in Table 2. XGBoost objective function has a regularization concept that helps to choose predictive functions and control the model's complexity. Putting the loss function and regularization term together, we get the objective function of the XGBoost. The predictive power of the model is controlled by loss function and the simplicity of the model is controlled by regularization term. We can define objective function of the XGBoost as shown in equation (3)

Table 2
XGBoost Classifier parameters.

Parameter	Default	Description
learning_rate	0.3	Shrink the weights on each step
n_estimators	100	Number of trees to fit.
objective	binary: logistic	logistic regression for binary classification
booster	gbtree	Select the model for each iteration
nthread	max	Input the system core number
min_child_weight	1	Minimum sum of weights
max_depth	6	Maximum depth of a tree.
gamma	0	The minimum loss reduction needed for splitting
subsample	1	Control the sample's proportion
colsample_bytree	1	Column's fraction of random samples
reg_lambda	1	L2 regularization term on weights
reg_alpha	0	L1 regularization term on weights

$$\mathrm{Obj} = \sum_{i=1}^{n} L(\hat{y}_i, y_i) + \sum_{i=1}^{k} \mathrm{R}(f_i) \tag{3}$$

where L represents the loss function which determines the compatibility of the model on train data; predicted label is denoted by $\hat{y}_i$ and y_i denotes the actual label . R(f) is responsible for penalizing the complexity of the functions of the training tree. It also handles the overfitting problem. To define the complexity, first, we need to define the function of tree $f(x)$ as

$$f(x) = w_{q(x)}, w \in R^T, q : R^m \to \{1, 2, \cdots, T\} \tag{4}$$

Here w represent the leaves scores vector, q represent a mapping function which maps data instances to the corresponding leaf, and the number of leaves is represented by T. The formula to penalize model's complexity is shown in Eq. (5):

$$\mathrm{R}(f) = \gamma T + \alpha(||w||) + \frac{1}{2}\lambda\left(||w||^2\right) \tag{5}$$

where λ and γ are the hyper-parameters or constant coefficients, each leaf value is represented by γ, and the total number of leaves in the tree is represented by T. $||w||^2$ denotes the L2-norm of the weight of the leaf controls by λ term and $||w||$ denotes the L1-norm of the weight of the leaf controls by α term. L2 regularization (controlled by the reg_lambda term) encourages the weights to be small, whereas L1 regularization (controlled by the reg_alpha term) encourages sparsity. The hyper-parameter γ (gamma) for further partition determines the minimum loss reduction. Similar to γ, a hyper-parameter w_{mc} (min_child_weight) controls the depth of tree and a large w_{mc} can make the model more conservative in the

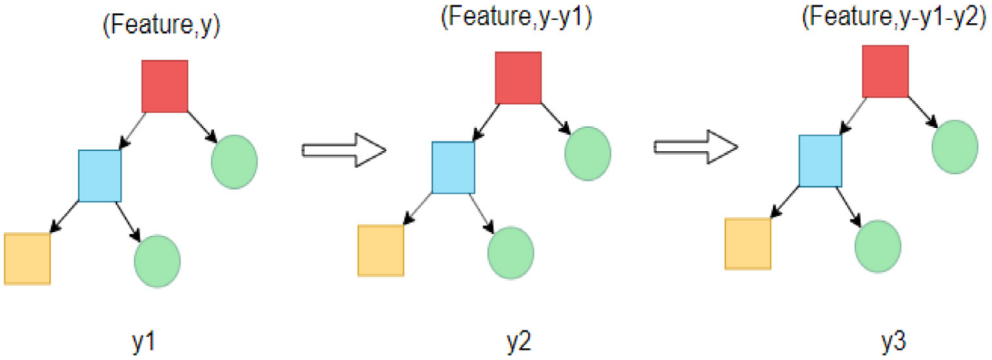

Fig. 1. Flow of gradient boosting algorithm.

process of splitting. The definitions of the hyper-parameters described above can be found in Table 2. Like other supervised learning models, we have an objective function and we need to optimize it. XGBoost uses gradient descent to optimize the objective function. Our model is an additive model in which it includes a tree in the model each time the result of the prediction is equal to the combined result of the previous tree and the new tree. So, at the t-th step, among these equations, the objective at each step is calculated by Eq. (6) and a f_t is selected to minimize the objective function, which is to reduce the error between the predicted outcome and the actual outcome after adding f_t.

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + R(f_t) + constant \tag{6}$$

Since we do not have a derivative for every objective function, we calculate the second-order Taylor approximation as shown in Eq. (7).

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + R(f_t) + constant \tag{7}$$

where g_i is (8) and h_i is (9).

$$\begin{aligned} g_i &= \partial_{\hat{y}_i^{(t-1)}} L(y_i, \hat{y}_i^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} L(y_i, \hat{y}_i^{(t-1)}) \end{aligned} \tag{9}$$

By removing the constant terms and adding regularization term from Eq. (5), we get Eq. (10),

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i)] + \gamma T + \alpha \sum_{j=1}^{T} \omega_j + \tfrac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2 \tag{10}$$

Eq. (10) shows the objective function at the t^{th} step.

Compared to GB, the column subsampling (Zieba et al., 2016) is another technique used in XGBoost to further avoid overfitting. It is demonstrated that the use of column subsampling is even more effective in preventing overfitting than conventional row subsampling (Bergstra and Bengio, 2012). Row subsampling of the data is done by hyper-parameter "subsample" and its description can be found in Table 2. The definition of the "colsample_bytree" hyper-parameter can be found in Table 2. The tree structure is established by calculating the leaf scores, regularization, and objective function at each level since it is not possible to calculate all combinations of trees at the same time. This tree structure will be reused in subsequent iterations, which will significantly reduce the computational complexity. Furthermore, the gain of each feature is calculated in the node splitting process. It finds the best splitting point recursively until it reaches to the maximum depth. Then it prunes out the nodes in a bottom-up order which has a negative gain. That's how XGBoost goes deep into trees and classify the data. XGBoost has many hyper-parameters (XGBoost, 0000), and these hyper-parameters can be used to perform some tasks as desired by the model. We list parameters that were used to get the results in Table 2.

We calculated the results based on the described hyper-parameters in Table 2. If parameters are not set, default values are chosen by XGBoost, although the parameters can be specified according to the desired model.

2.4. Hyper-parameter optimisation

Hyper-parameter optimization in machine learning is to find the best hyper-parameters for a given machine learning algorithm which gives the best performance when evaluated on a validation set. Hyper-parameter optimization is expressed in the form of the equation as:

$$x^* = \arg\min_{x \in X} f(x) \tag{11}$$

Here f(x) represents an objective score to minimize that is evaluated in the validation set; x^* is a set of hyper-parameters that yields the lowest score value, and x can take any value in the X domain. In simple terms, we want to find the model hyper-parameters that give the best score on the validation set metric. The issue with hyper-parameter optimization is that it is extremely costly to evaluate the objective function to find the score. We have to train a model on the training data, make predictions on the validation data, and then measure the validation metric every time we try different hyper-parameters. With a large number of hyper-parameters and models such as ensembles or deep neural networks that can take days to train, this process cannot be accomplished manually. Four common methods of hyper-parameter optimization are listed below:

- Manual Search
- Grid search
- Random search
- Bayesian optimization

Manual search is intractable with a large number of hyper-parameters. Grid and random search are better than manual search because they can run the train-predict-evaluate cycle automatically in a loop on a pre-decided grid of model hyper-parameters. However, even these methods are somewhat inefficient because they do not takes into account past evaluations when selecting the next hyper-parameters to evaluate**.** As a result, many times they spend a significant amount of time evaluating the wrong set of hyper-parameters.

Bayesian optimization, on the other hand, takes into account past evaluations when selecting the hyper-parameter set to evaluate next. By selecting its hyper-parameter combinations in an informed way, it enables itself to concentrate on those areas of the search space that it believes will bring the most promising validation scores. Typically, this method needs less iteration to get to the optimum set of hyper-parameter values. And after carefully reading (Wu et al., 2019) we can safely assume that Bayesian optimization can be applied to widely used machine learning models and it also reduces the runtime significantly compare to the manual search.

2.5. Bayesian optimisation

XGBoost has several hyper-parameters and tuning these hyper-parameters can be very complicated as selecting hyper-parameters significantly affects the performance of the model. Therefore, careful tuning of these hyper-parameters is important. Grid search (GS) has been applied for the hyper-parameter tuning of models in the previous studies (Ali et al., 2019) which have less number of hyper-parameters in their models but it would be infeasible for our XGBoost model because there is a substantial number of hyper-parameters included in our model. Bayesian optimization is an effective way to optimize objective functions globally which are costly to evaluate (Mockus, 1994; Jones et al., 0000). In this subsection, Bayesian hyper-parameter optimization technique is introduced which was used to optimize XGBoost model. Most of the machine learning optimization problems are black-box optimization problems where $f(\mathbf{x})$ is a black-box function. We have no analytical expression for f, nor do we know its derivatives. This is the area in which Bayesian optimization techniques are most useful. The model that this technique uses to approximate the objective function is called surrogate model. Gaussian processes (GPs) (Rasmussen, 2004; Williams and Rasmussen, 2006) are a common surrogate model for Bayesian optimization. The unknown objective

function space is modeled using a Gaussian process (GP). A GP is a collection of infinite numbers of random variables that have a joint Gaussian distribution for any finite number of combinations. GPs provide a way to specify prior distributions over smooth function space which is defined by its mean and covariance function. Matérn kernel 2.5 that is also used in the proposed model is a popular choice of covariance function. Also, an acquisition function is used by Bayesian optimization, which directs sampling to areas where there is a possibility of improvement over current best observation exists. The acquisition function's role is to direct us towards reaching the optimum objective function. Acquisition functions are defined in such a way that a high acquisition function value corresponds to a high objective function value. To get the next evaluation point, it optimizes the acquisition function. Expected improvement (EI), maximum probability of improvement (MPI) and upper bound confidence (UCB) are common acquisition functions. In the following, the expected improvement (EI) will be used in the proposed model which is the most widely used acquisition function. Suppose f is the objective function and its sampling point is $\mathbf{x}_t$:

$$\mathbf{x}_t = \text{argmax}_{\mathbf{x}} u(\mathbf{x} \mid \mathcal{D}_{1:t-1}) \tag{12}$$

where u is the acquisition function which is expected improvement (EI) in our case and $\mathcal{D}_{1:t-1} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{t-1}, y_{t-1})$, contain $t-1$ samples that were drawn from f so far.

2.5.1. The generic Bayesian optimization algorithm

For $t = 1, 2, 3 \cdots$ repeat:

- By optimizing the acquisition function over the GP, find the next sampling point: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} u(\mathbf{x} \mid \mathcal{D}_{1:t-1})$
- By evaluating the objective function f, obtain a possibly noisy sample$y_t = f(\mathbf{x}_t) + \epsilon_t$
- Add new sample $(\mathbf{x}_t, y_t)$ to the previous samples $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.

2.6. Mixed data of categorical and numerical variables

The dataset (Chen and Guestrin, 2016) used in this paper is a collection of categorical and numerical variables as shown in Table 1. If we want to apply classification algorithms to data with both categorical and numerical variables, we must either convert numerical variables into categorical variables or convert categorical variables into numerical variables. Heart disease dataset contain 8 categorical features. We encoded categorical features in this paper. We used One-Hot (OH) encoding technique for converting categorical variables into numerical variables.

2.6.1. One-Hot (OH) encoding

One-Hot encoding, sometimes referred to as "dummy coding," is a frequently used technique for turning a categorical variable into a numerical variable. In One-hot encoding, a new feature is created for each category, i.e. for each category level, we create a binary feature. An example of One-Hot encoding is shown in the Fig. 2. However, in the case of high cardinality, this technique results in a considerable number of features.

In the case of missing values, One-hot-encoding maintain each row containing missing values (NaN) as a separate feature column as shown in Fig. 3.

3. Proposed methodology

3.1. Preprocessing dataset

Heart disease dataset is split into train and test datasets in terms of 80% and 20% respectively. We use train dataset for model training and optimization. In the dataset 6 subjects have missing values, 4 values of 'number of major vessels' and 2 values of 'thalassemia'. Since both are categorical features, categorical feature encoding technique (OH encoding) will handle missing values in the categorical features encoding step as shown in Fig. 4. We converted "num" variable of dataset to a target variable named "diagnosis" which has two classes, presence, and absence of heart disease. This becomes the problem of binary classification and target variable class labels are 0 and 1, 1 for having heart disease and 0 for not having heart disease.

3.2. Xgboost training and Hyper-parameter optimization

The proposed diagnostic system is shown in Fig. 5. After preprocessing of train and test datasets, we applied XGBoost classifier for the binary classification of the target variable with Bayesian Optimization on training data. Bayesian Optimization is used to tune the hyper-parameters. There is no need to adjust all the hyper-parameters in Table 2. In this paper, we selected nine parameters for tuning; they are learning_rate, n_estimators, min_child_weight and max_depth, subsample, cosample_bytree, gamma, reg_lambda, reg_alpha. The learning_rate makes the model more stable and robust, and the min_child_weight, max_depth, subsample, colsample_bytree, gamma is used to control over-fitting. Meanwhile, regularization parameters reg_lambda and reg_alpha penalize models as they become more complex and reduce them to simple models. While we selected these nine hyper-parameters to be optimized by the Bayesian Optimization, we set several other important hyper-parameters in Table 1 to default in which binary:logistic is the objective hyper-parameter, which is intended for classification processing in the XGBoost algorithm. The optimization task is aimed at the objective function which is XGBoost algorithm with the different sets of hyper-parameters and evaluation indicator which is the mean AUC score of 20-fold stratified cross-validation of training data, which vary depending on the objective function selected. Fig. 5 shows the cross-validation of training data and evaluation of mean AUC score for a set of hyper parameter of XGBoost. Bayesian Optimization tries to maximize the mean AUC score with each iteration. After the given number of iteration is completed, it selects the model with the highest mean AUC score for prediction on holdout test data. We use different evaluation metrics measure the performance of the selected optimized XGBoost model on hold out test data.

Chest_pain_type
typical angina
atypical angina
non anginal pain
asymptomatic

→

Chest_pain_type_0	Chest_pain_type_1	Chest_pain_type_2	Chest_pain_type_3
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Fig. 2. One-hot encoding.

thalassemia
normal
fixed defect
reversable defect
NaN

thalassemia_0	thalassemia_1	thalassemia_2	NaN
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Fig. 3. One-hot encoding missing value treatment.

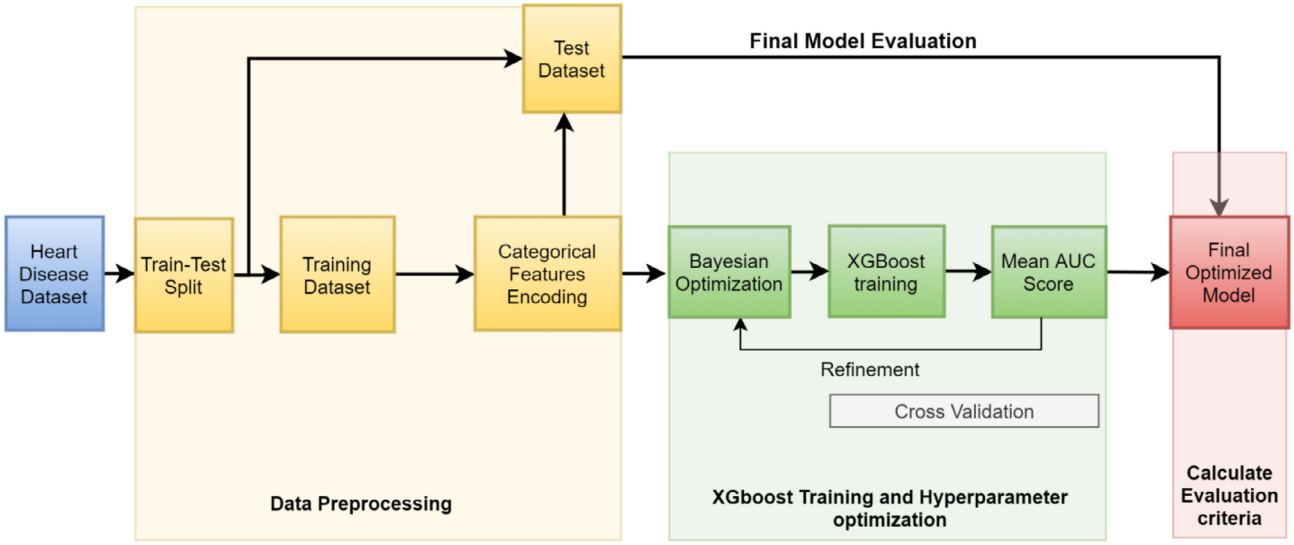

Fig. 4. Block diagram of the proposed method.

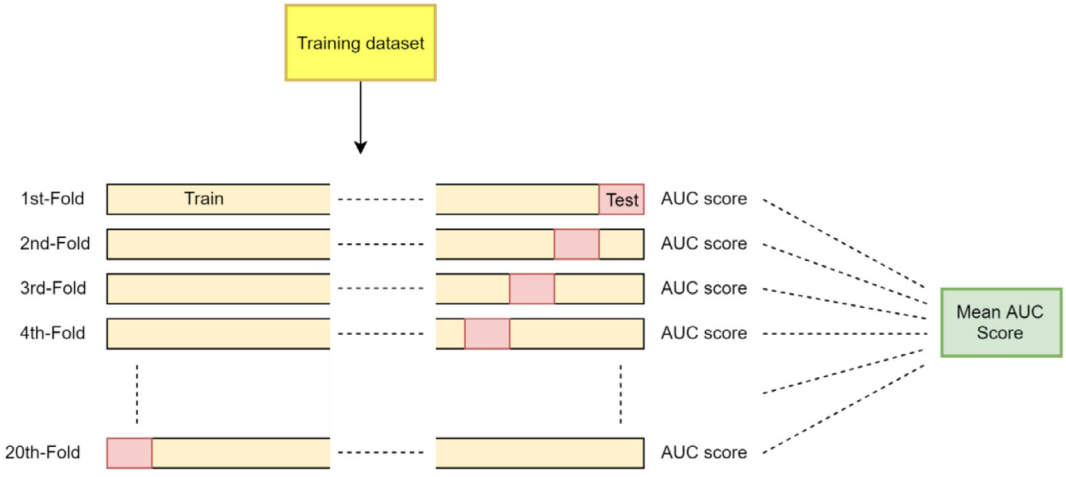

Fig. 5. 20-fold Cross-validation of XGBoost model for a set of hyper-parameters proposed by Bayesian optimization.

3.3. Evaluation metrics

We evaluate the efficacy of the proposed method using evaluation metrics such as accuracy, sensitivity, specificity, F1-score, and area under the curve (AUC) of ROC charts. Accuracy is the percentage of total subjects classified correctly. Sensitivity is the proportion of those who do have the disease who test positive. Specificity is the proportion of those who do not have the disease who test negative. Recall is same as sensitivity. Precision is the number of subjects correctly identified as positive out of total subjects identified as positive. F1-Score is a harmonic mean of precision and recall. These are

$$Accuracy = \frac{\Sigma TP + \Sigma TN}{\Sigma TP + \Sigma TN + \Sigma FP + \Sigma FN} \tag{13}$$

$$sensitivity = Recall = \frac{\Sigma TP}{\Sigma TP + \Sigma FN} \tag{14}$$

$$specificity = \frac{\Sigma TN}{\Sigma TN + \Sigma FP} \tag{15}$$

$$Precision = \frac{\Sigma TP}{\Sigma TP + \Sigma FP} \tag{16}$$

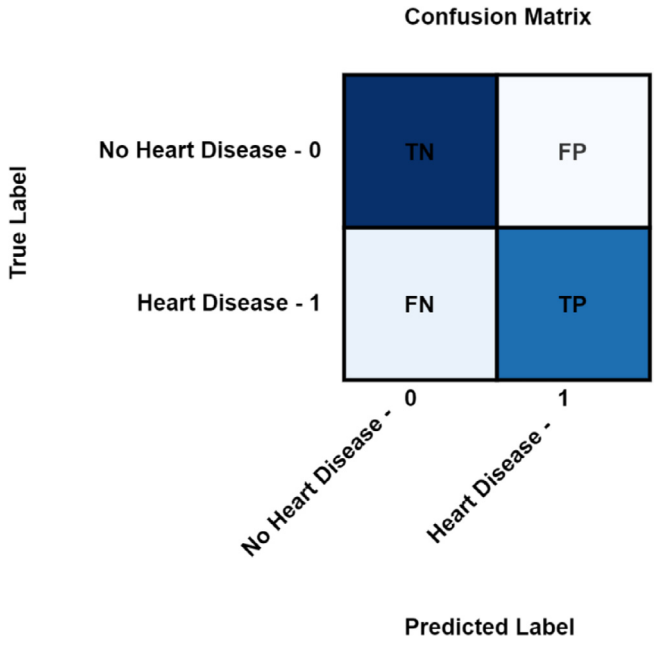

Fig. 6. Confusion Matrix for binary classification.

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{17}$$

where *TP* and *FP* denotes the number of correctly and wrongly classified subject having heart disease, respectively. Similarly, *TN* and *FN* denotes the number of correctly and wrongly classified subject not having heart disease, respectively.

ROC curve (Receiver operating characteristic curve) plots True Positive Rate (TPR) versus False Positive Rate (FPR) at different classification thresholds.

$$TPR = \frac{\Sigma TP}{\Sigma TP + \Sigma FN} \tag{18}$$

$$FPR = \frac{\Sigma FP}{\Sigma TN + \Sigma FP} \tag{19}$$

AUC stands for "Area under the ROC Curve". The AUC = 1 value is a perfect anomaly classifier, whereas the AUC = 0.5 value means the model's output is no better than a random guess.

Confusion matrix shown in the Fig. 6 is used to describe the binary classification prediction results. It contains the summary of prediction results of all instances of dataset used for testing.

4. Results and discussion

We performed two types of experiments in this section to evaluate the efficacy of the proposed model. In the first experiment, XGBoost Classifier with Bayesian optimization is used for the prediction of Heart Disease and we evaluate performance with different performance metrics. Finally, to compare the performance of the proposed model with other machine learning models, the second experiment is performed. We performed all computations on Google Colab which Operates under Ubuntu 64 bits and it is composed of a single core hyper threaded Intel Xeon processor@2.3 GHz and 13 GB of RAM. Moreover, Python programming open source packages were used to simulate the experiments.

4.1. Experimental results

We use the XGBoost machine learning algorithm as a classifier for training and testing in this paper. First, the hyper-parameters of XGBoost algorithm were optimized by the Bayesian Optimization algorithm and then using those optimized hyper-parameters performance analysis is done. In the Hyper-parameter optimization stage, the Bayesian Optimization algorithm is applying the different combinations of parameters of XGBoost and tried to maximize the mean AUC score on 20-fold stratified cross-validation. By Bayesian Optimization algorithm iterations, we obtain an optimal set of parameters. We set the model to stop after 20 iterations because there is no major improvement is expected. OH encoding was used to encode categorical features and dataset was created with OH encoded categorical features. We show the processed hyper-parameters in the hyper-parameter optimization stage in Tables 3. Iteration 15th in Table 3 has the optimal parameters of the proposed model. We show the performance evaluation of

Table 4
Performance of Proposed model.

Data	Accuracy	specificity	sensitivity	F1-score
Train	0.8595	0.9007	0.8108	0.8411
Test	0.9180	0.9696	0.8571	0.9056

Table 3
Simulation results of XGBoost Hyper-parameter optimization using Bayesian optimization on the dataset with OH encoded categorical features.

iteration	target	learning_rate	n_estimators	max_depth	min_child_weight	colsample_bytree	subsample	gamma	reg_alpha	reg_lambda
0	0.5	0.071257	950	6	0	0.832238	0.966996	1.865695	47.76631	31.26424
1	0.5	0.354186	984	9	2	0.927449	0.672767	4.579862	20.02506	42.46383
2	0.5	0.193667	865	1	0	0.594093	0.987199	0.831917	67.96005	16.75764
3	0.851032	0.444497	873	4	0	0.802402	0.996949	0.049939	13.96628	33.5512
4	0.5	0.122879	867	8	3	0.777473	0.668733	2.820285	29.24229	82.55795
5	0.5	0.412998	830	0	1	0.829181	0.960191	2.840075	56.77296	73.91137
6	0.5	0.33898	934	4	3	0.580988	0.880301	4.817821	32.36081	35.26305
7	0.5	0.113141	940	0	1	0.700866	0.559809	3.540582	7.197965	23.00471
8	0.5	0.247004	951	8	3	0.586746	0.503253	2.571438	47.65168	95.1622
9	0.5	0.357743	831	2	1	0.595679	0.856677	2.506682	38.1841	18.97874
10	0.780575	0.33927	874	3	0	0.599618	0.516853	2.580285	12.20429	31.65795
11	0.5	0.5	1000	10	5	1	1	0	100	100
12	0.861667	0.5	800	10	0	1	1	0	0	75.13478
13	0.845794	0.5	850	10	5	1	1	0	0	44.55726
14	0.5	0.5	1000	0	0	1	1	0	0	100
15	0.870794	0.309467	801	3	3	0.658399	0.642807	0.761624	1.135854	97.30065
16	0.854802	0.5	879	10	0	1	1	0	0	41.64379
17	0.5	0.5	800	10	5	1	1	0	100	100
18	0.5	0.5	1000	0	5	1	1	0	100	0
19	0.5	0.5	829	0	0	1	1	0	0	62.66708

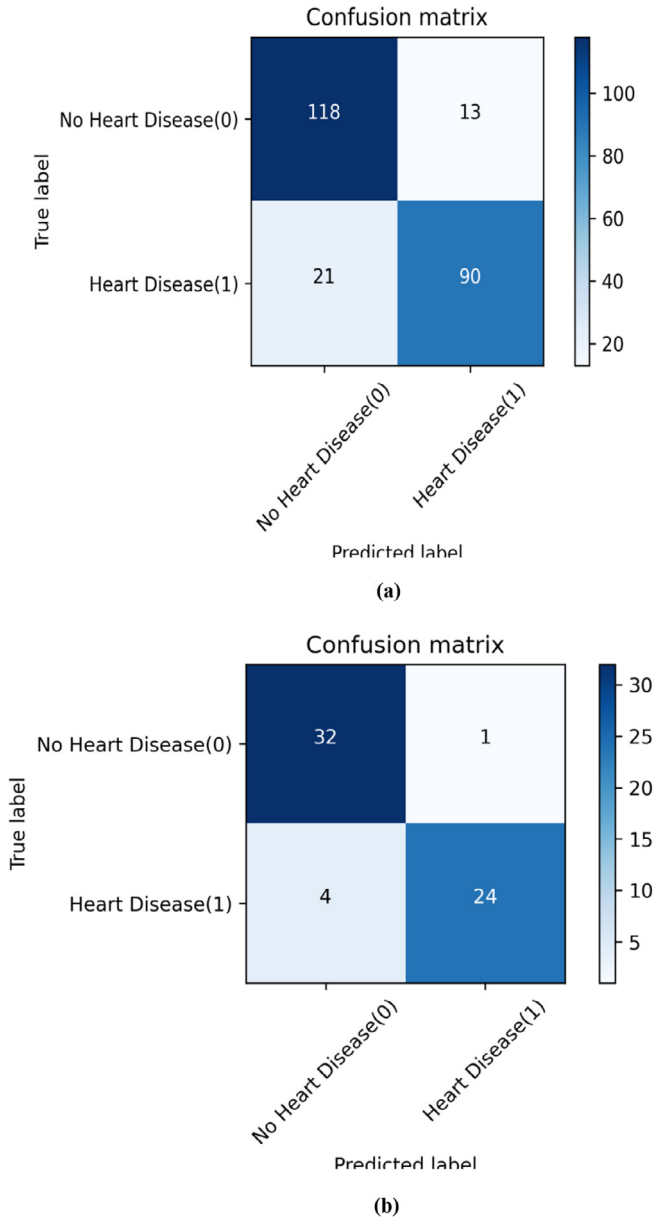

Fig. 7. (a) Confusion Matrix of Train data prediction result of proposed model. (b) Confusion Matrix of Test data prediction result of proposed model.

proposed model in Tables 4. Confusion matrix contain the summary of prediction results on train data and test data of proposed model is shown in Fig. 7(a) and (b) respectively.

ROC chart is another evaluation metric which is used to further verify the performance of the proposed model. From Fig. 6, it can be seen that we got AUC = 0.9134 for the optimized XGBoost model on a dataset with OH encoded categorical features.

Therefore, it can be seen that performance of the proposed model seems better from all evaluation aspects. So, we consider the proposed model for further comparative analysis.

4.2. Comparative analysis with other Machine learning models

4.2.1. Comparison with other tree-based models

We compared the proposed model experimental results with Random Forest (RF) and Extra Tree (ET) classifiers to verify the efficacy of the proposed model. We also trained these models on the dataset with OH encoded categorical feature and Bayesian Optimization is used to optimize the hyper-parameters of these two models. For both models, the processed hyper-parameters in the hyper-parameter optimization stage are shown in Tables 5 and 6. Iteration 13th in Table 5 and Iteration 5th in Table 6 have the optimal parameters of the RF model and ET model respectively The comparative analysis is conducted based on accuracy of classification. Table 7 shows the results of these models. From Table 7 we can conclude that performance of the proposed model is better than RF and ET models.

We also compared the performance of the proposed model with other models based on the AUC of ROC charts to further verify the efficacy of the proposed model. The proposed model, RF and ET model ROC charts are shown in Fig. 8, Fig. 9(a) and (b), respectively.

It is clear from the figures that the ROC curve AUC is 0.9285 for the proposed model, the RF model ROC curve AUC is 0.8804 and for the ET model ROC curve AUC is 0.8831. Therefore, the performance of the proposed model is better than the RF and ET models based on two evaluation metrics used which are accuracy and AUC of the ROC curve. This validates the efficacy of the proposed model.

4.2.2. Comparison with previous methods

In this section, a comparative analysis is done with previously proposed methods based on accuracy of classification. We show the comparison of these methods with classification accuracies in Table 8. In Ali et al. (2019) accuracy is recorded high this might be because they used a stacking technique that stacks two SVM models and they also used grid search method on both models simultaneously which can take considerable resources and time but their result is accurate and cannot be improved further. On

Table 5
Simulation results of RF hyper-parameter optimization using Bayesian optimization on the dataset with OH encoded categorical features.

iteration	target	n_estimators	max_depth	min_samples_split	max_features
0	0.832341	15	1	5	0.659276
1	0.866746	16	3	4	0.345061
2	0.869802	10	3	4	0.887463
3	0.846627	11	3	4	0.740719
4	0.880357	17	2	8	0.112378
5	0.861786	20	4	10	0.999
6	0.781329	1	4	10	0.1
7	0.788373	1	4	2	0.1
8	0.876706	13	4	10	0.1
9	0.874365	19	3	6	0.144512
10	0.876508	10	4	2	0.1
11	0.871706	19	1	9	0.146712
12	0.868651	16	3	7	0.139933
13	0.882262	20	4	2	0.1
14	0.854286	8	3	9	0.317397

Table 6
Simulation results of ET hyper-parameter optimization using Bayesian optimization on the dataset with OH encoded categorical features.

iteration	target	n_estimators	max_depth	min_samples_leaf	max_features
0	0.814306	2	2	1	4.937048
1	0.851647	6	1	4	9.370585
2	0.859702	9	1	1	7.906862
3	0.826825	6	1	1	8.924275
4	0.87246	10	1	1	8.709617
5	0.893532	20	3	5	1
6	0.874762	20	3	5	10
7	0.844008	12	1	5	1
8	0.891944	20	3	1	1
9	0.889901	12	3	5	8.383702
10	0.860595	17	1	1	7.536591
11	0.692361	1	3	5	1
12	0.854663	13	1	5	10
13	0.882381	14	3	1	6.825332
14	0.828829	1	3	5	10

Table 7
Performance comparison with other models.

Model	Accuracy
Random Forest	0.8852
Extra Tree	0.8852
Proposed model	0.9180

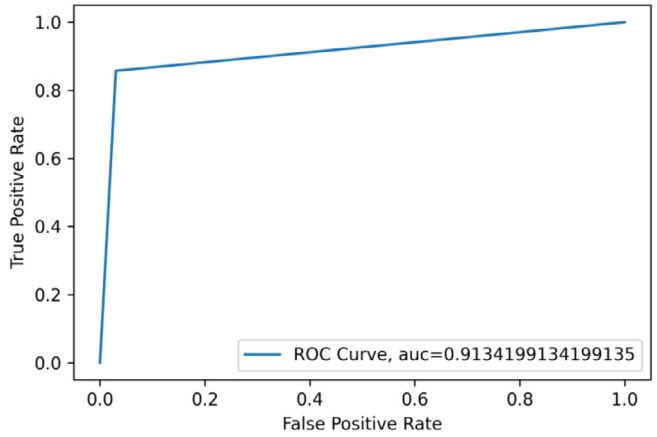

Fig. 8. ROC chart of proposed model.

the other hand, there is a chance that our model accuracy can still be improved if we also use grid search. From Table 8, we can conclude that the proposed model is better than some of the other models.

Table 8
Classification accuracies of the proposed model and previous models in the literature that used the heart disease dataset.

Study (year)	Method	Accuracy%
Das et al. (2009)	Neural networks ensembles	89.01
Anooj (2012)	Weighted fuzzy rules	62.35
Samuel et al. (2017)	ANN and fuzzy AHP	91.10
Ali et al. (2019)	Optimized Stacked SVM	92.22
Proposed model	Optimized XGBoost on a dataset with OH encoded categorical features	91.80

5. Conclusion

We proposed a diagnostic system for heart disease diagnosis in this paper. Our proposed method for the diagnostic system uses One-Hot encoding to encode categorical features of the dataset and optimized XGBoost for classification. We evaluated the proposed method using five different evaluation metrics, namely accuracy, sensitivity, specificity, F1-score, and Area under the curve (AUC) of ROC charts. We observed that our tree-based ensemble proposed method performs better than the other three previously proposed methods in terms of accuracy. The proposed method used Bayesian optimization as a hyper-parameter optimization technique which is proved to be a very efficient technique to get the best hyper-parameters. In addition, we compared the quality of the proposed method with two other tree-based ensemble machine learning methods. Our model outperforms both models by 3.28% in terms of accuracy. Based on experimental results, we can conclude that the suggested diagnostic method would improve the quality of decision-making during the diagnosis of the heart disease process. Future Research should test for other

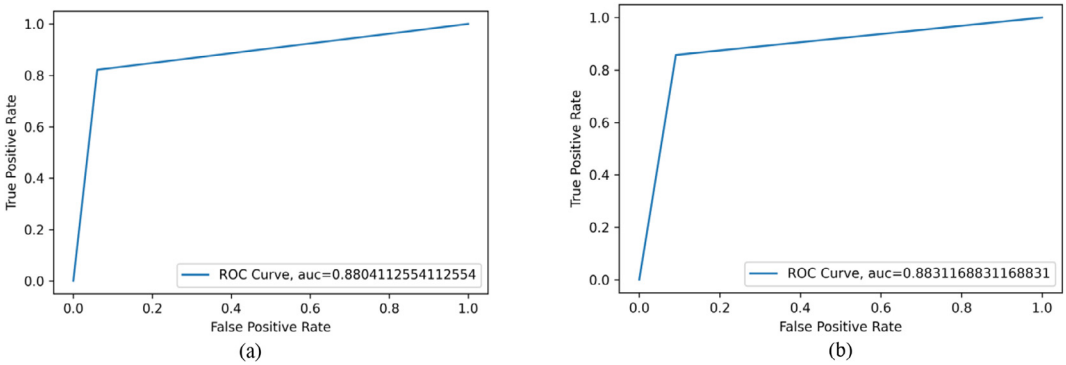

Fig. 9. ROC charts of RF and ET models.

similar tasks or other related data sets to evaluate its ability to produce similar accuracy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Samuel, O.W., Asogbon, G.M., Sangaiah, A.K., Fang, P., Li, G., 2017. An integrated decision support system based on ANN and fuzzy_AHP for heart failure risk prediction. Expert Syst. Appl. 68, 163–172.

Alizadehsani, R., Hosseini, M.J., Sani, Z.A., Ghandeharioun, A., Boghrati, R., 2012. 'Diagnosis of coronary artery disease using cost-sensitive algorithms. In: Proc. IEEE 12th Int. Conf. Data Mining Workshops (ICDMW), Dec. 2012, pp. 9–16.

Arabasadi, Z., Alizadehsani, R., Roshanzamir, M., Moosaei, H., Yarifard, A.A., Apr. 2017. 'Computer aided decision making for heart disease detection using hybrid neural network-genetic algorithm'. Comput. Methods Programs Biomed. 141, 19–26.

Polat, K., Şahan, S., Güneş, S., 2007. 'Automatic detection of heart disease using an artificial immune recognition system(AIRS) with fuzzy resource allocation mechanism and k-nn (nearest neighbour) based weighting preprocessing'. Expert Syst. Appl. 32 (2), 625–631.

Das, R., Turkoglu, I., Sengur, A., 2009. 'Effective diagnosis of heart disease through neural networks ensembles'. Expert Syst. Appl. 36 (4), 7675–7680.

Samuel, O.W., Omisore, M.O., Ojokoh, B.A., 2013. 'A Web based decision support system driven by fuzzy logic for the diagnosis of typhoid fever'. Expert Syst. Appl. 40 (10), 4164–4171.

Anooj, P.K., 2012. 'Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules'. J. King Saud Univ. Comput. Inf. Sci. 24 (1), 27–40.

Babaoglu, I., Findik, O., Ülker, E., 2010. 'A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine'. Expert Syst. Appl. 37 (4), 3177–3183.

Olaniyi, E.O., Oyedotun, O.K., Adnan, K., 2015. Heart diseases diagnosis using neural networks arbitration. Int. J. Intell. Syst. Appl. 7(12), 72.

Abushariah, M.A., Alqudah, A.A., Adwan, O.Y., Yousef, R.M., 2014. Automatic heart disease diagnosis system based on artificial neural network (ANN) and adaptive neuro-fuzzy inference systems (ANFIS) approaches. J. Softw. Eng. Appl. 7(12), 1055.

Manogaran, G., Varatharajan, R., Priyan, M.K., 2018. Hybrid recommendation system for heart disease diagnosis based on multiple kernel learning with adaptive neuro-fuzzy inference system. Multimedia Tools Appl. 77 (4), 4379–4399.

Özşen, S., Güneş, S., 2009. Attribute weighting via genetic algorithms for attribute weighted artificial immune system (AWAIS) and its application to heart disease and liver disorders problems. Expert Syst. Appl. 36 (1), 386–392.

Ali, L., Rahman, A., Khan, A., Zhou, M., Javeed, A., Ali Khan, J., 2019. An optimized stacked support vector machines based expert system for the effective prediction of heart failure. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2909969.

Bergstraand, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13 (1), 281–305.

Mantovani, R.G., Rossi, A.L.D., Vanschoren, J., de Bischl, A.C.P.L.F., Effectiveness of random search in SVM hyper-parameter tuning. In: Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2015, pp. 1–8.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, 2012, pp. 2951–2959.

Bergstra, J., Bardenet, R., Kégl, B., Bengio, Y., 2011. Implementations of algorithms for hyper-parameter optimization. In: NIPS Workshop on Bayesian optimization, p. 29.

Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2013. Auto-weka: Combined selection and hyper-parameter optimization of classification algorithms. In: In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 847–855.

Mockus, J., 1994. Application of Bayesian approach to numerical methods of global and stochastic optimization. J. Global Optim. 4 (4), 347–365.

Lizotte, D.J., Wang, T., Bowling, M.H., Schuurmans, D., 2007. Automatic gait optimization with gaussian process regression. In: IJCAI, vol. 7, pp. 944–949.

Brochu, E., Cora, V.M., De Freitas, N., 2010. tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.

González, J., Longworth, J., James, D.C., Lawrence, N.D., 2015. Bayesian optimization for synthetic gene design. arXiv preprint arXiv:1505.01627.

Cleveland heart disease database from an online machine learning and data mining repository of the University of California, Irvine (UCI). https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 785–794.

Friedman, J.H., Oct. 2001. Greedy function approximation: agradient boosting machine. Ann. Statist. 29 (5), 1189–1232.

Xia, Y., Liu, C., Li, Y., Liu, N., 2017. A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. Expert Syst. Appl. 78, 225–241.

Zieba, M., Tomczak, S.K., Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. Expert Syst. Appl. 58, 93–101.

Bergstra, J., Bengio, Y., 2012. andom search for hyper-parameter optimization. J. Mach. Learning Res. 13, 281–305.

XGBoost Parameters—Xgboost 0.7 Documentation. Available online: http://xgboost.readthedocs.io/en/ latest/parameter.html

Mockus, J., 1994. Application of Bayesian approach to numerical mssssssethods of global and stochastic optimization. J. Global Optim. 4 (4), 347–365.

Donald R. Jones, Matthias Schonlau, William J. Welch, xxxx. Efficient Global Optimization of Expensive Black-Box Functions.

Rasmussen, C.E., 2004. Gaussian processes in machine learning. Lect. Notes Comput. Sci. 63–71. https://doi.org/10.1007/978-3-540-28650-9_4.

Williams, C.K., Rasmussen, C.E., 2006. Gaussian Processes for Machine Learning. the MIT Press 2 (3), 4.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., Deng, S.-H., 2019. Hyper-parameter optimization for machine learning models based on bayesian optimization. J. Electron. Sci. Technol. 17(1).