Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2023, Article ID 5542049, 10 pages
https://doi.org/10.1155/2023/5542049

Research Article

Performance Augmentation of Base Classifiers Using Adaptive Boosting Framework for Medical Datasets

Durr e Nayab,[1] **Rehan Ullah Khan**,[2] **and Ali Mustafa Qamar**[3]

[1]*Department of Computer Systems Engineering, University of Engineering and Technology, Peshawar 25120, Pakistan*
[2]*Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia*
[3]*Department of Computer Science, College of Computer, Qassim University, Buraydah, Saudi Arabia*

Correspondence should be addressed to Ali Mustafa Qamar; al.khan@qu.edu.sa

Received 4 August 2023; Revised 28 November 2023; Accepted 7 December 2023; Published 22 December 2023

Academic Editor: Kalapraveen Bagadi

This paper investigates the performance enhancement of base classifiers within the AdaBoost framework applied to medical datasets. Adaptive boosting (AdaBoost), being an instance of boosting, combines other classifiers to enhance their performance. We conducted a comprehensive experiment to assess the efficacy of twelve base classifiers with the AdaBoost framework, namely, Bayes network, decision stump, ZeroR, decision tree, Naïve Bayes, J-48, voted perceptron, random forest, bagging, random tree, stacking, and AdaBoost itself. The experiments are carried out on five datasets from the medical domain based on various types of cancers, i.e., global cancer map (GCM), lymphoma-I, lymphoma-II, leukaemia, and embryonal tumours. The evaluation focuses on the accuracy, precision, and efficiency of the base classifiers in the AdaBoost framework. The results show that the performance of Naïve Bayes, Bayes network, and voted perceptron is highly improved compared to the rest of the base classifiers, attaining accuracies as high as 94.74%, 97.78%, and 97.78%, respectively. The results also show that in most cases, the base classifiers perform better with AdaBoost compared to their performance, i.e., for voted perceptron, the accuracy is improved up to 13.34%.For bagging, it is improved by up to 7%. This research aims to identify such base classifiers with optimal boosting capabilities within the AdaBoost framework for medical datasets. The significance of these results is that they provide insight into the performance of the base classifiers when used in the boosting framework to enhance the classification performance of classifiers in scenarios where individual classifiers do not perform up to the mark.

1. Introduction

Boosting in machine learning (ML) is achieved with the ability of the ML technique to boost the functionality of other classifiers when combined [1]. Boosting is a very effective technique for solving bi-class classification problems [2]. The boosting technique enhances the functioning and improves the correctness of any given learning algorithm by adding new modules. This procedure forms a new classifier, an ensemble of both classifiers with improved accuracy on a given training set [2]. For this reason, the name Boosting is assigned to such techniques as they boost the performance of other classifiers.

An example of a boosting technique, namely, adaptive boosting (AdaBoost), has a mechanism for training the data set based on allocating weights. Uniform weights are assigned to the datasets, and the probability of data selection is based on these weights. The training set, once classified accurately for one classifier, reduces the chance of the training set being utilized in the successive classifier [3]. Therefore, the selection of the training set is based on a classifier trained on it and the assigned weights. AdaBoost trains the classifier on beneficial, informative, and complicated patterns by iteratively running its algorithm. After each iteration, the training error is calculated, and the weights are allocated for the classifier.

Schapire and Freund purported the first solid algorithm for Adaboost, which is by AdaBoost's methodology [4, 5]. Viola and Jones proposed an updated version of the AdaBoost technique by taking the weak classifiers with weak features [6, 7]. Therefore, the Viola and Jones version of the

AdaBoost technique is a repetitive process combining multiple weak classifiers that approximate the base classifiers [8]. The AdaBoost classifier is mathematically expressed in the following equation:

$$C(x) = \text{argmax}_k \sum_{m=1}^{M} \left(\alpha^{(m)}. \text{II}\left(T^m(x) = k\right)\right), \quad (1)$$

where $C(x)$ represents a linear classifier, i.e., a linear sequence of all the constituent classifiers; $\alpha^{(m)}$ and II $(T^{(m)}(x))$ represent the arguments for the base classifiers. AdaBoost has a huge margin and generalization capability, making its performance better than other boosting techniques. There are some limitations with the AdaBoost technique, such as considerations taken for each constituent classifier and the range that influences the performance simplification of the constituent classifiers [9]. AdaBoost has an accuracy and diversity calamity, which means attaining higher accuracy with two-component classifiers increases the chance for them to disagree. Maintaining a balanced trade-off between accuracy and diversity is required to accomplish an acceptable generalization for performance.

The rest of the paper is organized as follows: Section 2 provides details of the related materials, experimental setup, and methods. Section 3 gives details about the experimental setup. Similarly, Section 4 provides an insight into the results of the proposed algorithms and their comparisons with recent algorithms. Lastly, Section 5 concludes the paper and provides insight on the future enhancements.

2. Materials and Methods

As mentioned earlier, we conducted 60 experiments on five medical datasets with about 12 base classifiers. These results are prepared to analyze the base classifiers with the AdaBoost framework based on their percentage accuracy, error, precision, F-measure, and recall. Initially, the datasets are extracted and preprocessed to be employed by the machine learning algorithms. The data is cleaned by removing redundant information and empty cells, replacing missing records, and normalizing the data to be presented in a uniform format. Essential features are extracted from the data based on evaluating the ML techniques used to train the algorithms. The performance of the ML techniques is evaluated, and the most efficient ML techniques are selected for the allocation of suitable base classifiers in a given scenario. This process ultimately selects the optimal base classifier for a given problem. The proposed methodology is depicted in Figure 1 as a block diagram.

This diagram provides an overview of our proposed algorithm for the performance evaluation of base classifiers in the AdaBoost Framework for given medical datasets. The algorithms are monitored to attain the best accuracy. The model attaining the best accuracy is then deployed to classify the tumors for best performance in a given scenario. The details for base classifiers and the setups used for these base classifiers are given in Section 2.1.

2.1. Base Classifiers. The performance of AdaBoost with the base classifiers is evaluated in this research with twelve base classifiers in combination with AdaBoost in almost sixty experiments. The base classifiers are chosen from almost all the major categories of classifiers, such as Bayes, Functions, Rules, Networks, Trees, and Meta Functions. In this way, all the major classifier groups have been evaluated, and therefore, these experiments comprehensively illustrate the role of base classifiers in the AdaBoost framework. The details of the base classifiers are provided in the following sections.

2.1.1. Naïve Bayes. Naive Bayes (NB) classification is a supervised learning technique used as a statistical method for classification. NB has good performance in classification and pattern recognition [10]. NB acquired its name from the well-known Thomas Bayes theorem that categorizes the training set by opting for the class with the closest relation to the dataset. Bayes proposed that there is no connection between the existence and nonexistence of input features with each other [11]. All the properties are supposed to contribute independently and equally to the output probability. NB uses a maximum likelihood method for parameter estimation [12]. This is called an NB assumption that is mathematically illustrated in the following equations:

$$v = \max_{cj=C} P\left(c_j \mid x_1, x_2, \ldots x_n\right), \quad (2)$$

$$= \max_{cj=C} \frac{P\left(x_1, x_2, \ldots x_n \mid c_j\right) P\left(v_j\right)}{P\left(x_1, x_2, \ldots x_n\right)}, \quad (3)$$

$$= \max_{cj=C} P\left(x_1, x_2, \ldots x_n \mid c_j\right) P\left(c_j\right), \quad (4)$$

$$P\left(x_1, x_2, \ldots x_n \mid c_j\right) = \prod_i P\left(x_i \mid v_j\right), \quad (5)$$

$$f_{nb}(E) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{n} \frac{p(x_i \mid C=+)}{p(x_i \mid C=-)}, \quad (6)$$

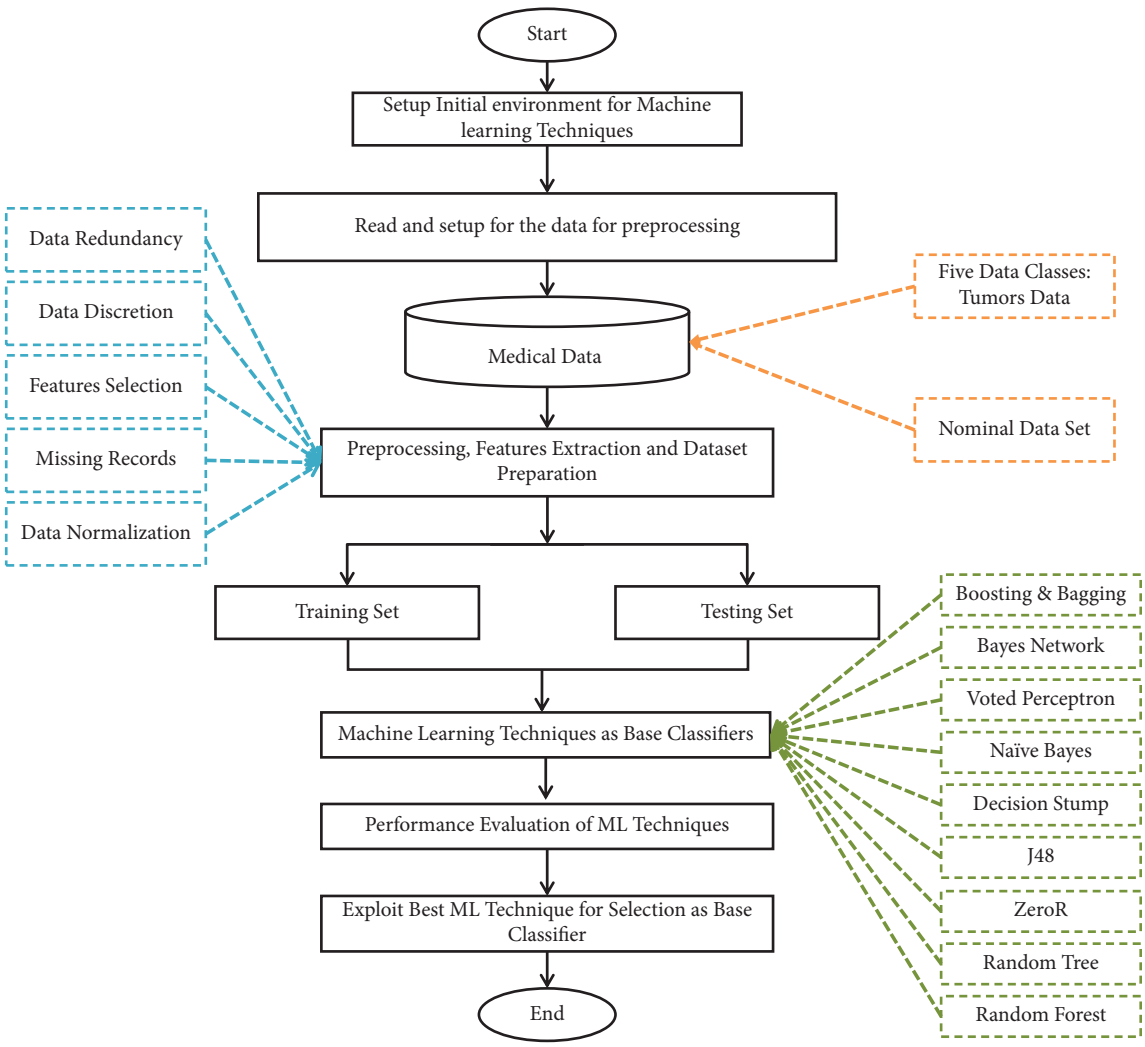

Figure 1: Block diagram for implementation of the proposed techniques.

where the classification value of the classes is given by C, P is the probability, E is the expectation, function f_{nb} (E) is the NB classifier, and $x_1, x_2, x_3, x_4, \ldots, x_n$ are attribute values. Each attribute node has only one parent, i.e., the primary parent node C in NB. All attributes do not necessarily depend on other attributes in NB [13]. Hence, knowing class variables is enough in NB to conduct the classification procedure. Moreover, all the attributes are statistically independent and equally essential in NB. NB classifier requires only moderate training data for mean, variance, and other classification parameter approximation [14]. Depending upon the characteristics of the probability model in supervised learning settings, the NB classifier with the most significant value leads the hypothesis.

2.1.2. Voted Perceptron. Voted Perceptron (VP) is a linear classifier used under the supervised classification scenario. Frank Rosenblatt proposed the VP classification technique in 1957 [15]. VP works with the online learning technique by processing the training set individually and making predictions based on linear predictor functions. The weights are set as zero and initially act as parameter vectors. The VP algorithm stores the parameter vectors by passing over the training set [16]. The errors, for example, are handled with modifications in the parameter vectors on the fly. Hence, the technique for VP uses $f(x)$ to map a single-valued input vector x to output y. The mathematical expression for the binary classifier VP is given in the following equation:

$$f(x) = \begin{cases} 1, & w.x + b > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

VP accumulates more information in the training phase, and improved predictions are generated with highly structured information on the test data. The VP algorithm uses the batch training mechanism for learning purposes, and it runs iteratively over the training set while waiting for it to locate a prediction vector [17, 18]. The prediction vector makes accurate learning on all the training sets and is used to estimate the labels on the training set.

2.1.3. Bayes Network. Bayes network (BN) is a simple network structure comprising nodes and edges, which was proposed in 1988 by Pearl [19]. The network assumes that every attribute, i.e., the leaf, is independent of every other attribute in the given state of the classifier [20]. Random variables represent the BN nodes, including variables, unknown parameters, observable quantities, or hypotheses. Disconnected nodes characterize the short-term independent variables, and the edges represent conditional dependencies. Additional edges are formed between the attributes in BN to grab correlations. All possible edge combinations that form the whole network must be searched by BN. The combined probability distribution of random variables is used for training, as shown mathematically in the following equation:

$$P_B(X_1, X_2, \ldots . X_n) = \prod_{i=1}^{n} P_B(X_i \mid \pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi_i}. \quad (8)$$

BN takes its shape in various kinds of acyclic networks depending upon the problem state for efficiently searching the whole network space. BN comprises the two-stage learning process of natural division, giving it a dual nature [21]. The first stage learns a network structure, and the second learns probability tables. BN has an influence diagram structure that represents and resolves the decision problems [22, 23]. BN forms sequences of variables known as dynamic BN in speech signals or protein sequence applications.

2.1.4. Decision Stump. A decision stump (DS) is a single-layered decision tree (DT) that makes it comparatively easier to build. Instances in DS are classified by assembling them with feature values [24, 25]. DS has a finite number of splits on the attributes, so only one attribute is necessary for its network. In DS classifiers, an instance feature to be classified is represented by a node, and a node value represents the corresponding branch [26]. The learning model of DS is based on a single internal root node. The architecture of the network is such that the root is instantaneously connected to the terminal nodes that make its leaves. The leaves further expand in the network and form a vast tree-like structure. Hence, it makes decisions based on a single input feature, also known as one-rules [27].

DS is often used as a module (called "weak learners") in ML ensemble methods, such as bagging and boosting [27]. DS may be constructed with a leaf for each potential trivial feature, one for fitting with some chosen group and another for the rest of the categories. Stated scenarios are analogous to binary features and may be considered a different category if a feature is unavailable. Threshold levels are used for features to classify them into two different leaves if they are above or below the threshold value and multiple leaves with multiple threshold levels.

2.1.5. Random Tree. A random tree (RT) classifier was introduced by Cutler and Breiman to address regression and classification problems [28]. RT has a network of tree predictors, also known as an ensemble and decision splits, which efficiently model the data on attributes. The RT algorithm requires the input feature space, and each tree classifies inputs in the forest. Ultimately, the class label that makes the most votes is produced, and decisions are taken based on the weights assigned to the nodes.

RT is a very efficient decision algorithm that outperforms in terms of accuracy [29]. In the case of noisy data sets, this classifier is observed to have inadequate performance. If a significant portion of data is misplaced, RT approximates missing data with its inbuilt technique and retains accuracy.

2.1.6. Boosting and Bagging. Breiman developed the bagging technique as a procedure to enhance the performance of classification under ML methods [30]. The bagging title is attained from "bootstrap aggregating" terminology, as bagging is a clustering technique. Bagging generates individual classifiers for its clusters from each classifier based on a random distribution of the training dataset [31]. To create a classifier training set, the data is randomly taken with the replacement of examples, and the resultant data is equal in size to that of the original training set. Significant variations are reflected in the model from small changes in training data, which means this classifier has an unstable predictor. Bagging amalgamates multiple hypotheses with huge errors and generates a classifier with reduced errors in the training set.

Boosting is a collection of methods that primarily aims to produce and combine a series of classifiers [32]. It combines hypotheses generated by related learning methods that invoke various distributions of the training set. Boosting attains improvement in recognition for unstable classifiers and smoothing over discontinuities by similar. A boosting classifier is comparatively more prolific and efficient and has a trouble-free group learning approach [33]. Bagging and boosting have methods trained on dissimilar data established with Bootstrap, which resamples the original data. Bagging and boosting algorithms combine base classifiers whose outputs are assessed to determine the ultimate output.

2.1.7. Random Forest. Random forest (RF) is one of the finest classification techniques for massive data [34]. A large number of inputs do not affect the input variable when the RF technique is used. Compared to other techniques, it deals with huge amounts of data without much adverse effect on accuracy. The RF technique efficiently approximates the missing data, and because of this, it preserves accuracy even with a large amount of missing data. RF solves the issues with unbalanced data and balances the error in the class population [35]. This classifier has great potential to resolve the problems associated with vague data sets, which helps deal with unsupervised clustering, outlier detection, and data views. The RF method calculates the proximities between pairs of clusters that aid in locating the outliers in clustering. The RF technique also proposes an experimental method for detecting variable interactions in data [35].

2.1.8. ZeroR. ZeroR is based on a rule that works with a straightforward classification method on the target and prediction of the majority class. ZeroR ignores all predictors

and relies on its decisions based on the majority of occurrences of an instance [36]. The predictability power of ZeroR is negligible, but it has a significant role as a standard baseline for other classifiers. ZeroR maintains a frequency table for the target class and selects it based on the majority frequency. ZeroR uses the most common class values for classification after identifying them. ZeroR is often employed as a baseline for other ML algorithms to evaluate their results since it returns a value for each instance.

2.1.9. J-48. J-48 is based on Quinlan's C4.5 DT algorithm, which is the most frequently used. The J-48 technique has a much more common approach than DT, which divides the data into small subsets based on a decision criterion [37]. Leaves in J-48 represent similar subclasses, and the potential data gains are associated with the attributes based on the test. Instances are categorized in the same class or leaf they are associated with, and each attribute's potential data is tested on attributes that provide the gain on data. Eventually, the selection parameter is used to select the best-suited attribute [38]. There are some limitations to the J-48 algorithm, such as empty branches, over-fitting, and insignificant branch problems, which must be resolved and handled well when working with J-48. Some solutions are proposed to these problems, such as adding RT and Kendall's rank correlation (KRC) to J-48, besides many others that target the mentioned issues with J-48 and improve the overall performance [37, 38].

3. Experimental Setup

The experimental setup for this research is based on twelve base classifiers in combination with AdaBoost and five datasets. These data sets are mostly taken from medical problems such as various types of cancer. The datasets have many attributes and instances; as in medical problems, a large amount of information is required for making decisions. A brief description of the data sets is given in the following sections.

3.1. Data Collection and Preprocessing. The data is then preprocessed to prepare the training and testing data sets, including significant steps such as removing redundant data, data discretion and features construction, features selection, retrieval of missing records, separation of the testing and training sets, and data normalization. In the data preprocessing module, the data is analyzed for redundant and missing data and divided into training and testing sets.

3.1.1. Global Cancer Map. The global cancer map (GCM) is a dataset for multiclass cancer (MCC) diagnosis and is assessed using the technique of tumour gene expression signatures (TGES) [39]. There are about 16,064 attributes, each of which has 144 instances. The classifiers verify the output for fourteen classes: breast, colorectal, prostate, uterus-adeno, lung, renal, lymphoma, melanoma, mesothelioma, pancreas, bladder, leukaemia, central nervous system (CNS), and ovary. The decision is taken to classify the datasets into any of the aforementioned classes. Figure 2 shows the output distribution among these classes for a randomly chosen attribute.

3.1.2. Lymphoma-I. Lymphoma is a cancer that attacks the lymphocytes, a constituent part of the immune system. Typically, lymphoma is an undetectable solid tumour of lymphoid cells that affects the body's immune system. Detecting lymphoma tumours is challenging and requires special methods such as gene expression profiling (GEP) to identify these tumours. The lymphoma-I dataset used in this research has two classes, i.e., anterior cruciate ligament (ACL) and granule cell layer (GCL), which are classified with medical diagnosis [39]. By GEP, distinct types of diffused large blood-cell lymphoma (DLBCL) are diagnosed. Numerous attributes are utilized to evaluate the results, essential for detecting lymphoma. There are 4,027 attributes, each of which has 45 instances. The output distribution between these classes for a randomly selected attribute is shown in Figure 3.

3.1.3. Lymphoma-II. In the lymphoma-II data set, the lymphoma is monitored for nine classes instead of two, i.e., NIL, DLBCL, ABB, GCB, RAT, RBB, FL, TCL, and CLL [39]. These nine classes are attributed to the GEP analysis. The GEP technique is used to classify distinct types of DLBCL. The lymphoma-II data set is employed to monitor various attributes for testing the results, which is essential for detecting lymphoma. There are 4,027 attributes, each of which has 96 instances. The data has been taken for its medical diagnosis by classifying it into nine classes; the output distribution for a randomly selected attribute is shown in Figure 4.

3.1.4. Leukaemia. Leukaemia is a bone marrow or blood cancer, while in general, it is also attributed to a wide range of diseases. Leukaemia is identified by an abnormal rise in immature white blood cells (WBCs), also known as blasts. Leukaemia-II is used to monito the molecular classification of cancer (MCC), which includes methods such as class discovery and prediction by GEP [39]. There are 7,130 attributes, each of which has 38 instances. The data has been taken for its medical diagnosis by classifying it into two classes, i.e., acute lymphoblastic leukaemia (ALL) and acute-myeloid leukaemia (AML). The output distribution between these classes for a randomly selected attribute is shown in Figure 5.

3.1.5. Embryonal Tumours. Embryonal tumour (ET) data is taken from the results of CNS for ET. It is monitored for the results of the GEP for the prediction of CNS ET. This data set has 7,130 attributes, each of which has 60 instances. The data is classified into positive results, denoted by one, and negative results, denoted by 0. ET data is taken for medical diagnosis based on genes [39]. The output distribution between these classes for a randomly selected attribute is shown in Figure 6.

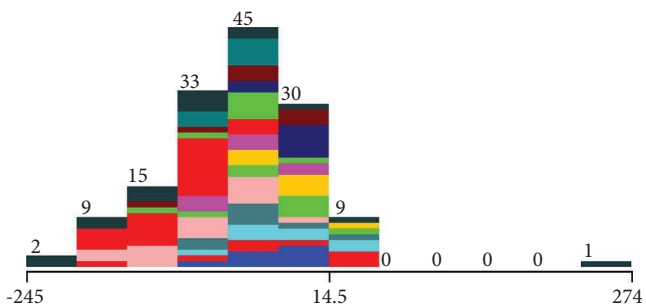

Figure 2: Data distribution for GCM.

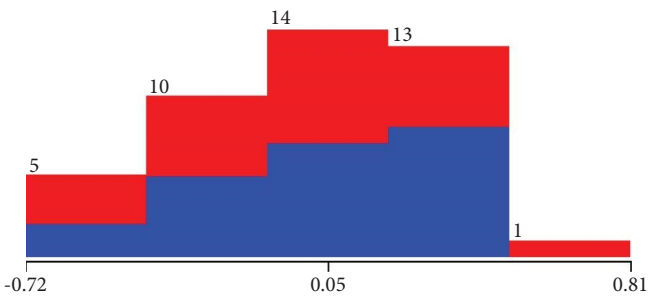

Figure 3: Data distribution for lymphoma-I with two classes.

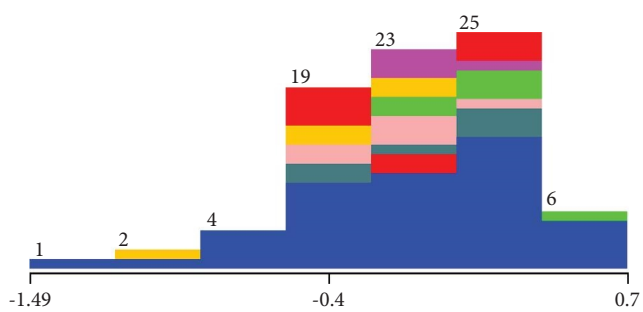

Figure 4: Data distribution for lymphoma-II with nine classes.

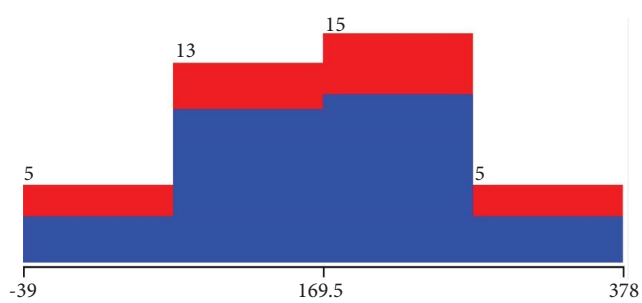

Figure 5: Data distribution for leukemia.

3.2. Environment Setup for Machine Learning Classifiers. Certain parameters are considered as input features and are varied for different scenarios. In order to validate the model, 10-fold cross-validation is utilized for all machine learning methods. To increase efficiency, highly correlated input attributes are chosen as the data. Additionally, feature selection methods are employed to eliminate attributes that are not relevant to the output variable. To minimize the complexity of the model, a ridge regularization technique is applied, which prevents any coefficient from reaching an excessive value by reducing the sum of the squares of the learned coefficients. The distance between the data points is calculated using Euclidean distance, as the data is on the same scale. The data is searched and stored using a linear NN search method, and no windowing is required. The nearest neighbour is located through a linear search mechanism.

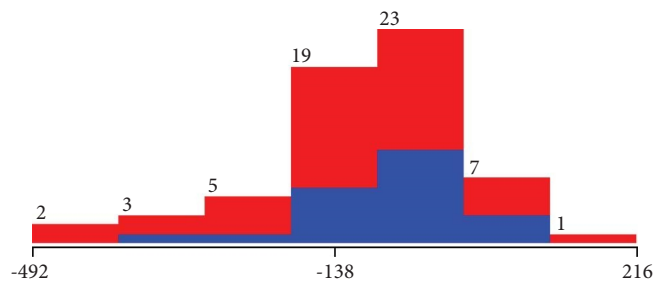

Figure 6: Data distribution for embryonal tumors.

A learning rate of 0.001 is established, and the regularization parameter is adjusted based on the number of epochs. The regularisation parameter is reduced as the number of epochs increases to 1,012. Seventy percent of the total data is used for training, and thirty percent is used for testing. During the training phase, in the beginning, RMSE generally decreased as the number of nodes in the hidden layer increased, and then RMSE began to increase when the model started to over-fit. The early stopping criterion is used to avoid over-fitting. Various internal parameters are chosen by the trial and error method. The excess use of input variables usually has a negative influence because it decreases the processing speed and affects the redundancy contained in the different variables.

3.3. Performance Evaluation Metrics. The results for our research are collected from 60 experiments conducted on five medical datasets with about twelve base classifiers, as mentioned in earlier sections. Based on their percentage accuracy, error, precision, F-measure, and recall, these results are prepared to analyze the base classifiers with the AdaBoost framework. The percentage accuracy of the base classifier is displayed in terms of correctly classified instances against incorrectly classified instances for all twelve base classifiers. The precision of the base classifiers has been computed from the ratio of the examples that are truly classified as class x to the total number of examples. The values for recall are calculated from the proportion of examples classified as class x to the actual total in class x. Similarly, the F-measure is computed from the measure of precision and recall. The mathematical expressions of these performance evaluators are provided in the following equations:

Table 1: Results for accuracy and percentage error of all the classifiers in AdaBoost framework for five medical (cancer) datasets.

Classifiers	Datasets Instances Attributes	Leukaemia 38 7130	Lymphoma-I 45 4027	Lymphoma-II 96 4027	GCM 144 16064	Data set C 60 7130
Naïve Bayes	Correctly classified	**94.74%**	**91.11%**	75.00%	16.67%	60.00%
	Without AdaBoost	**94.74%**	**91.11%**	75.00%	16.67%	61.67%
	Incorrectly classified	5.26%	8.89%	25.00%	83.33%	40.00%
	Without AdaBoost	5.26%	8.89%	25.00%	83.33%	38.33%
Decision stump	Correctly classified	**89.47%**	**86.67%**	51.04%	16.67%	63.33%
	Without AdaBoost	**89.47%**	**82.22%**	51.04%	16.67%	68.33%
	Incorrectly classified	**10.53%**	**13.33%**	48.96%	83.33%	36.67%
	Without AdaBoost	**10.53%**	**17.77%**	48.96%	83.33%	31.68%
Voted perceptron	Correctly classified	**78.95%**	**97.78%**	xx	16.67%	58.33%
	Without AdaBoost	**73.68%**	**84.44%**	xx	16.67%	65.00%
	Incorrectly classified	**21.05%**	2.22%	xx	83.33%	41.67%
	Without AdaBoost	**26.31%**	15.55%	xx	83.33%	35.00%
Stacking	Correctly classified	71.05%	51.11%	47.92%	16.67%	65%
	Without AdaBoost	71.05%	44.44%	47.92%	16.67%	65%
	Incorrectly classified	28.95%	48.89%	52.08%	83.33%	35%
	Without AdaBoost	28.95%	55.56%	52.08%	83.33%	35%
Bagging	Correctly classified	**92.11%**	**93.33%**	**86.46%**	xx	61.67%
	Without AdaBoost	**84.3%**	**86.67%**	**70.83%**	xx	66.00%
	Incorrectly classified	**7.89%**	**6.67%**	**13.54%**	xx	38.33%
	Without AdaBoost	**15.78%**	**13.33%**	**29.17%**	xx	33.00%
J-48	Correctly classified	84.21%	82.22%	**86.46%**	xx	56.67%
	Without AdaBoost	84.21%	77.78%	**81.25%**	xx	58.00%
	Incorrectly classified	15.79%	17.78%	13.54%	xx	43.33%
	Without AdaBoost	15.79%	22.22%	18.75%	xx	42.00%
Random tree	Correctly classified	81.57%	64.44%	**66.67%**	38.19%	**68.33%**
	Without AdaBoost	86.84%	68.89%	**58.33%**	40%	**63.33%**
	Incorrectly classified	18.42%	35.56%	**33.33%**	61.81%	**31.67%**
	Without AdaBoost	13.15%	31.11%	**41.66%**	60%	**36.68%**
Random forest	Correctly classified	79.41%	91.11%	78.13%	**52.08%**	65.00%
	Without AdaBoost	88.23%	91.11%	82.29%	**50%**	66.67%
	Incorrectly classified	20.58%	8.89%	21.88%	47.92%	35.00%
	Without AdaBoost	11.76	8.89%	17.7%	50%	33.33%
Bayes network	Correctly classified	**94.74%**	**97.78%**	90.62%	16.67%	62.34%
	Without AdaBoost	**94.74%**	**97.78%**	90.83%	16.67%	68%
	Incorrectly classified	6.66%	2.22%	9.375%	83.33%	37.36%
	Without AdaBoost	6.66%	2.22%	8.16%	83.33%	31%
AdaBoost	Correctly classified	89.47%	86.67%	51.04%	16.67%	63.33%
	Without AdaBoost	89.47%	86.67%	51.04%	16.67%	63.33%
	Incorrectly classified	10.53%	13.33%	48.96%	83.33%	36.67%
	Without AdaBoost	10.53%	13.33%	48.96%	83.33%	36.67%
ZeroR	Correctly classified	71.05%	44.44%	47.92%	16.67%	65%
	Without AdaBoost	71.05%	44.44%	47.92%	16.67%	65%
	Incorrectly classified	28.95%	55.56%	52.08%	83.33%	35%
	Without AdaBoost	28.95%	55.56%	52.08%	83.33%	35%
Input mapped classifier	Correctly classified	71.05%	44.44%	47.92%	16.67%	65%
	Without AdaBoost	71.05%	44.44%	47.92%	16.67%	65%
	Incorrectly classified	28.95%	55.56%	52.08%	83.33%	35%
	Without AdaBoost	28.95%	55.56%	52.08%	83.33%	35%

*The crossed (xx) cells show that the results could not be generated for the specific classifier because of the limitation of the framework or the data set. Hence, the evaluation of these classifiers' results has been carried out manually to check if any better results could be gathered for comparison. Bold values indicate the improved values.

Table 2: Results for precision, F-measure, and recall error of all the classifiers in AdaBoost framework for various data sets.

Classifiers	Data sets	Leukaemia	Lymphoma-I	Lymphoma-II	GCM	Data set C
	Instances	38	45	96	144	60
	Attributes	7130	4027	4027	16064	7130
Naïve Bayes	Recall	**0.947**	0.911	0.75	0.167	0.6
	F-measure	**0.946**	0.911	0.692	0.048	0.56
	Precision	**0.951**	0.914	0.683	0.028	0.552
Voted perceptron	Recall	0.789	**0.978**	xx	0.167	0.6
	F-measure	0.799	**0.978**	xx	0.048	0.58
	Precision	0.847	**0.979**	xx	0.028	0.583
Stacking	Recall	0.711	0.511	0.479	0.167	0.65
	F-measure	0.59	0.511	0.31	0.048	0.512
	Precision	0.505	0.511	0.23	0.028	0.423
Adaboost	Recall	0.895	0.867	0.51	0.167	0.583
	F-measure	0.895	0.867	0.445	0.094	0.52
	Precision	0.895	0.87	0.403	0.066	0.5
Bagging	Recall	0.921	0.933	**0.865**	xx	0.633
	F-measure	0.92	0.933	**0.84**	xx	0.629
	Precision	0.92	0.934	0.836	xx	0.626
J48	Recall	0.842	0.822	0.835	xx	0.567
	F-measure	0.842	0.822	0.835	xx	0.546
	Precision	0.842	0.825	**0.864**	xx	0.536
Random tree	Recall	0.789	0.644	0.667	0.382	**0.683**
	F-measure	0.789	0.643	0.649	0.373	**0.685**
	Precision	0.789	0.645	0.648	0.37	**0.687**
Random forest	Recall	0.791	0.778	0.781	**0.521**	0.65
	F-measure	0.761	0.776	0.743	**0.509**	0.643
	Precision	0.791	0.788	0.75	**0.53**	0.639
Bayes network	Recall	0.933	**0.978**	xx	0.167	0.633
	F-measure	0.933	**0.978**	xx	0.048	0.629
	Precision	0.934	**0.979**	xx	0.028	0.626
Decision stump	Recall	0.895	0.867	0.51	0.167	0.633
	F-measure	0.895	0.867	0.445	0.094	0.629
	Precision	0.895	0.87	0.403	0.066	0.626
Zero-R	Recall	0.711	0.444	0.479	0.167	0.65
	F-measure	0.59	0.429	0.31	0.048	0.512
	Precision	0.505	0.434	0.23	0.028	0.423
Input mapped classifier	Recall	0.711	0.444	0.479	0.167	0.567
	F-measure	0.59	0.429	0.31	0.048	0.571
	Precision	0.505	0.434	0.23	0.028	0.576

*The crossed (xx) cells show that the results could not be generated for the specific classifier because of the limitation of the framework or the data set. Hence, the evaluation of these classifiers' results has been carried out manually to check if any better results could be gathered for comparison. The results in bold indicate the best results for different datasets.

$$\textbf{accuracy} = \frac{\text{correctly identified instances}}{\text{incorrectly classified instances}}, \tag{9}$$

$$\textbf{precision} = \frac{\text{truly classified class X examples}}{\text{total classified class X examples}}, \tag{10}$$

$$\textbf{recall} = \frac{\text{truly classified class X examples}}{\text{total class X examples}}, \tag{11}$$

$$\textbf{f-measure} = \frac{2 * (\text{precision} * \text{recall})}{(\text{precision} + \text{recall})}. \tag{12}$$

4. Results and Discussions

The results are shown in Table 1, where the rows represent various classification algorithms and the percentage of correctly and incorrectly classified examples, while the columns represent data sets. The results show that the performance of Naïve Bayes, Bayes network, voted perceptron, and bagging as base classifiers in AdaBoost is better than the rest. These base classifiers outperformed in AdaBoost, attaining accuracies of **94.74%, 97.78%, 97.78%,** and **93.33%**, respectively, while their accuracies are lower, i.e., **84.44%**, for voted perceptron and **86.67%** for bagging

technique. The results also show that in most cases, the base classifiers perform better with AdaBoost compared to their individual performance, i.e., for Voted Perceptron, the accuracy is improved up to **13.34%,** and for bagging, it is improved up to **7%.** Table 2 shows the precision, recall, and F-measure of the base classifiers in AdaBoost. Table 2 highlights the best precision values, i.e., **97.9%** achieved by VP and BN. The finest recall values are highlighted in Table 2, which are **97.8%** achieved by VP and BN. The best values for F- Measure are highlighted in Table 2, i.e., **97.8%** achieved by VP and BN. The precision, recall, and F-measure for each base classifier are evaluated from comparisons between each base classifier so that their role in the AdaBoost framework is examined.

Using attributes to analyse the behaviour of base classifiers plays a prognostic role in identifying the best combination of base classifiers with AdaBoost. Hence, this performance evaluation provides an analytical model for choosing a base classifier in a given problem domain, making it easy to select an AdaBoost environment with a base classifier. As a future enhancement of proposed research, these observations can be used for some applications of AdaBoost, such as Viola-Jones object detection, and improve its performance with suitable base classifiers.

5. Conclusions and Future Work

Adaptive Boosting (AdaBoost), being an instance of boosting, combines other classifiers to enhance their performance. This boosting functionality of AdaBoost is highlighted in this work by monitoring the performance of several base classifiers with AdaBoost. Almost 60 experiments are carried out to observe the responses of twelve base classifiers on five significant medical data sets. The results of these experiments show that the AdaBoost framework attains better results with some base classifiers (Naïve Bayes, Bayes network, and voted perceptron) than other classifiers (J48, bagging, decision stump, random forest, and random tree). The reason is that base classifiers have a unique role in the AdaBoost classification. This research aims to track the unique role of the base classifiers in the AdaBoost framework and identify the classifiers with the best performance for the given medical dataset. The performance of the base classifiers is monitored in terms of their accuracy, precision, recall, and F-measure. The results show that the performance of Naïve Bayes, Bayes network, voted perceptron, and bagging as in AdaBoost is better than the rest of the base classifiers. These base classifiers outperformed AdaBoost, attaining accuracies of **94.74%, 97.78%, 97.78%,** and **93.33%,** respectively, while their individual accuracies are lower, i.e., **84.44%** for voted perceptron and **86.67%** for bagging technique. The results also show that in most cases, the base classifiers perform much better with AdaBoost compared to their individual performance, i.e., for Voted Perceptron, the accuracy is improved up to **13.34%,** and for bagging, it is improved up to **7%.** One of the limitations of this research is the application of the proposed algorithm to a dataset that belongs to a similar category, i.e., cancer data. Hence, as a future extension of this work, these experiments will be applied to other types of medical datasets, i.e., brain tumours, skin cancers, ECGs, medical imaging, clinical trials, Oasis, and CT datasets, and enhanced by taking some other applications of AdaBoost such as Viola-Jones object detection to improve its performance with the base classifiers.

Data Availability

The dataset could be made available on request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Shan, P. Yang, X. Chen, and W. Gao, "AdaBoost Gabor Fisher classifier for face recognition," in *Proceedings of the Second International Workshop on Analysis and Modelling of Faces and Gestures*, pp. 278–291, Beijing, China, October 2005.

[2] D. Oosterlinck, D. F. Benoit, and P. Baecke, "From one-class to two-class classification by incorporating expert knowledge: novelty detection in human behaviour," *European Journal of Operational Research*, vol. 282, no. 3, pp. 1011–1024, 2020.

[3] G. Hu, C. Yin, M. Wan, Y. Zhang, and Y. Fang, "Recognition of diseased Pinus trees in UAV images using deep learning and AdaBoost classifier," *Biosystems Engineering*, vol. 194, no. 1, pp. 138–151, 2020.

[4] A. Shahraki, M. Abbasi, and O. Haugen, "Boosting algorithms for network intrusion detection: a comparative evaluation of real AdaBoost, gentle AdaBoost and modest AdaBoost," *Engineering Applications of Artificial Intelligence*, vol. 94, no. 1, Article ID 103770, 2020.

[5] K. W. Walker, "Exploring adaptive boosting (AdaBoost) as a platform for the predictive modeling of tangible collection usage," *The Journal of Academic Librarianship*, vol. 47, no. 6, Article ID 102450, 2021.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple," *Proceedings of CVPR*, vol. 1, no. 1, pp. 511–518, 2001.

[8] J. Zhu, A. Arbor, and T. Hastie, "Multi-class AdaBoost," *Machine Learning*, vol. 4, no. 1, pp. 0–20, 2006.

[9] X. Li, L. Wang, and E. Sung, "AdaBoost with SVM-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, 2008.

[10] Z. Nisar, J. Zahoor, R. U. Khan, and R. J. Qureshi, "Palmprint recognition: a naive bayesian approach," *World Applied Sciences Journal*, vol. 31, no. 11, pp. 771–775, 2014.

[11] N. Salmi and Z. Rustam, "Naïve Bayes classifier models for predicting the colon cancer," *IOP Conference Series: Materials Science and Engineering*, vol. 546, no. 5, Article ID 052068, 2019.

[12] H. Zhang, "The optimality of Naïve Bayes," *Machine Learning*, vol. 3, no. 1, pp. 440–450, 2004.

[13] B. G. Marcot and T. D. Penman, "Advances in Bayesian network modelling: integration of modelling technologies,"

Environmental Modelling & Software, vol. 111, no. 1, pp. 386–393, 2019.

[14] F. J. Yang, "An implementation of Naïve Bayes classifier," in *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 301–306, Las Vegas, NV, USA, December 2018.

[15] M. Collens and F. Park, "Ranking algorithm for named–entity extraction: boosting and the Voted Perceptron," in *Proceedings of the 40th Annual Meeting for Computational Linguistics*, pp. 489–496, Philadelphia, PA, USA, July 2002.

[16] Y. Freund and R. E. Schapire, "Large margin classification using the Perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.

[17] T. Saba, "Automated lung nodule detection and classification based on multiple classifiers voting," *Microscopy Research and Technique*, vol. 82, no. 9, pp. 1601–1609, 2019.

[18] I. Martišius, S. Kestutis, and R. Damaševičius, "Real-time training of voted perceptron for classification of EEG data," *International Journal of Artificial Intelligence*, vol. 10, no. 13, pp. 41–50, 2013.

[19] B. Gal, F. Ruggeri, F. Faltin, and R. Kenett, "Bayesian networks," *Neural Networks*, vol. 5, pp. 70–76, 2007.

[20] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2/3, pp. 131–163, 1997.

[21] P. Larranaga, H. Karshenas, C. Bielza, and R. Santana, "A review on evolutionary algorithms in Bayesian network learning and inference tasks," *Information Sciences*, vol. 233, no. 1, pp. 109–125, 2013.

[22] L. Shi, Q. Duan, P. Dong, L. Xi, and X. Ma, "Signal prediction based on boosting and decision stump," *International Journal of Computational Science and Engineering*, vol. 16, no. 2, pp. 117–122, 2018.

[23] M. Shah, M. Marchand, and J. Corbeil, "Feature selection with conjunctions of Decision Stumps and learning from microarray data," *Machine Learning*, vol. 9, no. 1, pp. 1–15, 2010.

[24] S. B. kotsiants, D. Kanellopoulos, and P. E. Pintelas, "Local additive regression of decision stumps," *Neural Networks*, vol. 3, no. 1, pp. 148–157, 2006.

[25] W. Iba, M. Kaufmann, and P. Langley, "Induction of one-level decision trees," in *Proceedings of the Ninth International Machine Learning Conference*, pp. 233–240, Aberdeen, Scotland, February 1992.

[26] L. Peter, "Building decision trees from decision stumps," *Forum American Bar Association*, vol. 2, no. 2, pp. 1–7, 2010.

[27] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013.

[28] R. Rastogi and K. Shim, "PUBLIC: a decision tree classifier that integrates building and pruning," *Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 315–344, 2000.

[29] G. Jagannathan, P. Krishnan, and R. N. Wright, "A practical differentially private random decision tree classifier," in *Proceedings of the IEEE International Conference on Data Mining Workshops, ICDMW'09*, pp. 114–121, Miami, FL, USA, December 2009.

[30] K. Machova, F. Barcak, and P. Bednar, "A Bagging method using Decision Tress in the role of base classifiers," *Cybernetics*, vol. 3, no. 2, pp. 121–132, 2006.

[31] S. E. Roshan and S. Asadi, "Improvement of Bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization," *Engineering Applications of Artificial Intelligence*, vol. 87, no. 2, Article ID 103319, 2020.

[32] S. M. Basha, D. S. Rajput, and V. Vandhan, "Impact of gradient ascent and boosting algorithm in classification," *International Journal of Intelligent Engineering and Systems (IJIES)*, vol. 11, no. 1, pp. 41–49, 2018.

[33] D. Nayab, M. H. Zafar, and A. Altalbe, "Prediction of scenarios for routing in MANETs based on expanding ring search and random early detection parameters using machine learning techniques," *IEEE Access*, vol. 9, no. 1, pp. 47033–47047, 2021.

[34] L. Devroye, "Consistency of Random Forests and other averaging classifiers," *Journal of Machine Learning Research*, vol. 9, no. 3, pp. 2039–2057, 2008.

[35] J. Ali, R. U. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[36] S. M. Narangale and N. Sm, "Preprocessor agent approach to knowledge discovery using Zero-R algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 12, pp. 82–84, 2011.

[37] S. Aljawarneh, M. B. Yassein, and M. Aljundi, "An enhanced J48 classification algorithm for the anomaly intrusion detection systems," *Cluster Computing*, vol. 22, no. 5, pp. 10549–10565, 2019.

[38] R. Joshi and M. Alehegn, "Analysis and prediction of diabetes diseases using machine learning algorithm: ensemble approach," *International Research Journal of Engineering and Technology*, vol. 4, no. 10, pp. 426–435, 2017.

[39] X. Zhang, *Cancer Gene Expression Datasets*, Zenodo, Genève, Switzerland, 2015.