Vol. 10(37), Jul. 2020, PP. 4707-4715

Contents list available at IJMEC

International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)

Journal Homepage: www.aeuso.org

PISSN: 2411-6173 - EISSN: 2305-0543

Comparing the Function of Different Classifiers on MIMIC-III Dataset: A Case Study

Saber Ziaei and Mohsen Morshedi

Electrical and computer engineering department, Babol Noshirvani University of Technology Babol,

***Corresponding Author's E-mail:** saber.ziaei@gmail.com*

Abstract

MIMIC-III (Medical Information Mart for Intensive Care III) is a large, freely-available database comprising of de-identified health-related data associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. This database includes information such as demographics, vital sign measurements made at the bedside, laboratory test results, procedures, medications, caregiver notes, imaging reports, and mortality and classification of these data is an important problem. In this paper, we used different types of classification models like Random Forest, KNN, Logistic Regression, Etc. Moreover, we have reached F1 score of 93.5% in Gradient Boosting Classifier with the base learner of Random Forest.

Keywords: *MIMIC, Health, Classification, Boosting, KNN.*

1. Introduction

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modelling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing, etc. There are two types of learners in classification as lazy learners and eager learners.

A. Lazy learners

Lazy learners simply store the training data and wait until testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting (e.g. K-nearest neighbours).

B. Eager learners

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space.

Article History:
Received Date: Apr. 14, 2020
Accepted Date: Jul. 01, 2020
Available Online: Jul. 20, 2020

Due to the model construction, eager learners take a long time to train and less time to predict. (e.g. Decision Tree, Naïve Bayes). Below are the classifier models that we used for the classification of the MIMIC III dataset.

1.1. Decision Tree

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision-making knowledge from the supplied data. The decision tree can be generated from training sets.

1.2. Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as a bagging ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models**.** The low correlation between models is the key.

1.3. AdaBoost

AdaBoost, short for "Adaptive Boosting", is the first practical boosting algorithm proposed by Freund and Schapire in 1996. It focuses on classification problems and aims to convert a set of weak classifiers into a strong one. In this paper, we applied the decision tree and random forest as weak classifiers.

1.4. Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. In this paper, we applied the decision tree and random forest as weak prediction models.

1.5. K-Nearest Neighbours

K-nearest neighbour is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g. distance functions). A case is classified by a majority vote of its neighbours, with the case being assigned to the class most common amongst its K-nearest neighbours measured by a distance function.

1.6. MIMIC- III dataset

MIMIC-III ('Medical Information Mart for Intensive Care') is a large, single-center database comprising information relating to patients admitted to critical care units at a large tertiary care hospital. Data includes vital signs, medications, laboratory measurements, observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and more. The database supports applications including academic and industrial research, quality improvement initiatives, and higher education coursework.

2. Related Works

The boosting technique was introduced in 1994 by Schapire [1], The actual idea in boosting is that the understanding of the weak and strong classifiers are equivalent, which means that strong learning classification will be achieved by merging the weak learners. The training tuples are trained with weights in the boosting method. The tuples are then learned in a repetitive manner using the boosting classifier. During each cycle of repetition, the weights of the tuples are recorded. The tuples that have been correctly classified are given lower weights but those that have been given wrong classifications are given higher. This is because as more weight is given to the tuple, more focus is put on it. The performance of this boosting method is evaluated by how correctly the weights are assigned to the tuples. This boosting method can be improved as it relies on assumptions so that at the end it transforms the large margins of error to the smaller margins.

A. Random Forest

Random Forest constructs multiple numbers of decision trees and combines all the decision trees to bring maximum accuracy and more constant prediction. Random Forest technique implements by considering the arbitrary feature subset selection for the splitting of a node [2]. Instead of finding for the optimum probable saturation points as in common decision trees, for each and every feature, it uses arbitrary thresholds to make the decision trees utmost random. This mechanism does well, as the summation of more decision trees decreases the noise influence which tends to give more accurate outcomes, whereas a single decision tree can prone to noise effect. The disadvantage is that created subsets of different decision trees may tend to overlap and also it is tough to interpret. The main disadvantage is taking the maximum number of trees creates ineptness in the technique and making to work slow and not suitable for real-time forecasts.

B. Random Forest:

AdaBoost (adaptive boosting), brought up by Yoav and Robert (1997), is a machine learning algorithm known for its distinctive and exceptional performance on classification and regression. The algorithm continually trains weak classifiers for T iterations and during each round, it forces the next classifier to focus more on the incorrectly classified samples. The intuition of the algorithm is to alter the distribution over the original data, laying more emphasis on the harder samples to classify, which leads to higher accuracies of the weak classifiers in the next iterations on these parts. The final hypothesis combines previous weak hypotheses by summing their probabilistic predictions and proves to be successful in real-world problems. Concretely, every sample possesses a weight, which will be ordinarily set to $1/N$ if there are N training samples in total. In the beginning, a weak classifier is called and m samples are selected randomly for training. Suppose c samples are wrongly classified. In the second iteration, these c samples will remain while m-c samples are to be selected from N-m samples in the original dataset at random. Simultaneously, the weights of the c samples are increased while those

of the correctly classified ones will be decreased. Identical procedures are reduplicated until specified iterations [3]. The summary of the AdaBoost algorithm is shown as follows:

Input a weak learning algorithm WeakLearn and a dataset with *N* samples: (x_1, y_1), (x_2, y_2)... (x_N, y_N) where x_i is the sample (feature vector) and y_i is the corresponding label. Set the number of iterations (T).

- Initialize the weights $w_i^1 = 1/N$ for all i
- For t = 1, 2, 3... T

1) Normalize the weights:

$$p^t = \frac{w^t}{\sum_{i=1}^{N} \omega_i^t} \quad (1)$$

2) Call WeakLearn with the normalized weights p^t and obtain a hypothesis $h_t : X \rightarrow [0, 1]$.

3) Compute the error of h_t:

$$\varepsilon_t = \sum_{i=1}^{N} p_i^t \, |h_t(x_i) - y_i| \quad (2)$$

4) Set βt=εt/(1 −εt).

5) Update the weights vector:

$$\omega_i^{t+1} = \omega_i^t \beta_t^{1-|h_t(x_i)-y_i|} \quad (3)$$

After T iterations, output the final strong classifier:

$$h_f(x) = \begin{cases} 1 & if \ \sum_{t=1}^{T} (\log 1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log(\frac{1}{\beta_t}) \\ 0 & otherwise \end{cases} \quad (4)$$

C. Gradient Boosting

Among the many machine learning techniques, gradient boosting is a particularly attractive approach. It was first introduced in 1999 by Stanford University Professor Jerome H. Friedman (4) and has been further developed and refined in several open-source packages such as Scikit-Learn, and R. Major software developers such as SAS and IBM have developed their own implementations and Friedman's original gradient boosting machine (with refinements) has been part of the Salford Systems SPM product since 2001. As of early 2019, Google scholar reports about 10,000 citations in the scientific literature to Friedman's two papers introducing gradient boosting and applications that can be found in almost all fields of scientific investigation. Let's look at a simple example that reveals the main idea of formal gradient boosting. Traditionally, modelers would try to find a function that can accurately describe the

data. However, the function can only be an approximation of the data distribution and there must be errors:

$$y_i = F_1(X_i) + error_{1i} \quad [1]$$

where y_i is the outcome variable and X_i is a vector of predictors.

Suppose that the $F_1(X_i)$ function is a weak learner and the relationship between X and y is not fully described. In this situation, the error or residual is not white noise but has some correlation with y. We may want to train a second model on the error term:

$$y_i - F_1(X_i) = error_{1i} = h_1(X_i) + error_{2i} \quad [2]$$

The updated model will look something like:

$$F_2(X_i) = F_1(X_i) + h_1(X_i) \quad [3]$$

After performing this procedure iteratively we finally can fit a model at the M^{th} step:

$$F_M(X_i) = F_{M-1}(X_i) + h_{M-1}(X_i) \quad [4]$$

Gradient boosting allows one to plug in any class of weak learners $h_m(X_i)$ to improve predictive accuracy. In other words, the $h_m(X_i)$ is a weak learner that can take any functional form such as a GLM, a neural network or a decision tree. Although there is no requirement for $h_M(X_i)$ to be a specific function, it is usually a tree-based learner in practice.

The residual or error can be expressed as the loss function $L[Y, F_M(X)]$ in machine learning terminology and our objective is to build a model that minimizes this loss. More formally, the loss function is defined as the difference between the predicted $[F_M(X)]$ and the observed value (Y). The loss function can be the squared error for an ordinary least square's regression model. For loss functions that cannot be resolved with simple algebra, gradient descent is commonly employed to estimate the optimal parameters.

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function (6). Conventionally, the gradient is calculated with respect to the model parameter. In gradient boosting, the gradient of the loss function is calculated with respect to the predicted value. That is, one takes the derivative of the loss function with respect to the predicted value. The gradient can be considered as pseudo-residual. More formally, gradient boosting proceeds with the following steps (7):
(I) Initialize the model by solving the following equation:

$$F0(x)=\arg\min_{\gamma}\sum_{i=1}^{n} L(yi, \gamma); \quad \text{then we get: } F0(x)=\sum_{i=1}^{n} yin \quad [5]$$

(II) For m =1 to M
Compute the gradient with respect to the predicted value,

$$rim=-\{\partial L[yi,Fm-1(xi)]\partial Fm-1(xi)\}[6]$$

where *i* is the index for observations. Each observation will have one gradient value. The lowercase *m* is the number of boosting iterations with m ∈ [1, M].

(III) Fit the weak learner h_m (x) to the residuals by:

Computing the γ_m to solve the optimization problem:

$$\gamma m=\arg\min\gamma\sum ni=1\ L[yi,Fm-1(xi)+\gamma\cdot Fm(xi)][7]$$

By solving this equation we can get:

$$\gamma m=\sum ni=1\ hm(xi)\cdot[yi-Fm-1(xi)]\sum ni=1\ hm(xi)2[8]$$

(IV) Update the F_m (x) = F_{m-1} (x) + $\gamma_m \cdot h_m$ (x)

D. Decision Tree

Inductive inference is the process of moving from concrete examples to general models, where the goal is to learn how to classify objects by analyzing a set of instances (already solved cases) whose classes are known. Instances are typically represented as attribute-value vectors. Learning input consists of a set of such vectors, each belonging to a known class, and the output consists of a mapping from attribute values to classes. This mapping should accurately classify both the given instances and other unseen instances.

A decision tree [Quinlan, 1993] is a formalism for expressing such mappings and consists of tests or attribute nodes linked to two or more sub-trees and leaves or decision nodes labeled with a class which means the decision. A test node computes some outcome based on the attribute values of an instance, where each possible outcome is associated with one of the subtrees. An instance is classified by starting at the root node of the tree. If this node is a test, the outcome for the instance is determined and the process continues using the appropriate subtree. When a leaf is eventually encountered, its label gives the predicted class of the instance.

3. Results

In this section, we summarize the results of studies performed on the MIMIC III dataset.

A. Comparing different classification models:

First, we minimize MIMIC III dataset and continued in the minor scope. We have 10000 test data from the dataset with the minimum features. Absolutely MIMIC III dataset has 4 columns, ROW_ID, SUBJECT_ID, ITEM_ID, and VALUENUM which are data section in dataset moreover there is a column named WARNING which is dataset label. Actually, column WARNING considered as a binary number that shows an alarm sound. We start our work with 6 classifiers:

- Logistic Regression
- Random Forest Classifier
- Gradient Boosting Classifier
- AdaBoost Classifier
- KNN

Also, we evaluated algorithms base of 4 Criterion:

- Accuracy
- Precision
- Recall
- ROC
- F1 Score

With respect to the details that we mentioned above, we implemented 6 algorithms:

- Linear Regression
- Random Forest (Base Algorithm)
- Gradient Boosting (Random Forest Learner)
- AdaBoost (Random Forest Learner)
- AdaBoost (Decision Tree Learner)
- KNN

In table1 we present our results. Certainly, F1-Score is the main criterion and we comprise algorithm results base on it. According to these outcomes, we have the best performance in boosting algorithms. As shown in table 1 F1-Score for baseline algorithm (Random Forest) is 0.84 while F1-Score for Gradient Boosting and AdaBoost is 0.94 and 0.93 which these results are great in all algorithm that we had.

Table 1: implementation results

Classifier	Recall	Precision	Accuracy	F1 Score
Linear Regression	0.51	0.51	0.53	**0.49**
Random Forest(Baseline Algorithm)	0.94	0.93	0.93	**0.84**
Gradient Boosting (Random Forest Learner)	0.93	0.94	0.94	**0.94**
AdaBoost (Random Forest Learner)	0.93	0.93	0.93	**0.93**
AdaBoost (Decision Tree Learner)	0.89	0.91	0.90	**0.90**
KNN	0.61	0.58	0.58	**0.60**

As mentioned, ROC is one of the main metrics that we measured. In figure 1 we can see the results of algorithms based on ROC.

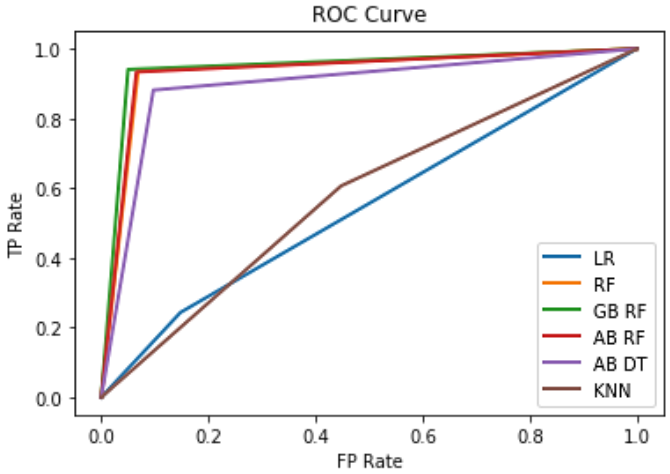

Figure 1: ROC based results

Extraordinarily, we considered two rates for ROC measurement, TP-rate, and FP-rate. Based on ROC Curve we can find out gradient boosting has the best performance.

CONCLUSION

In this article, we stated information about MIMIC III and why it is one of the important datasets in the health field. Then we introduced algorithms that can be used effectively. We talked about their details, formulated them. Next, we implemented all of them and measured their results base on a variety of metrics. Finally, according to outcomes we found that boosting algorithms have the best consequent rather baseline algorithms. In the future, we can use boosting algorithms to achieve great outcomes in the fields of health.

References

[1] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sep. 16, 1997.

[6] (2015) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Shell. (2015) IEEEtran webpage on CTAN. [Online]. Available: http://www.ctan.org/pkg/ieeetran

[8] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9] "PDCA12-70 datasheet," Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.