Enhancing the prediction of student performance based on the machine learning XGBoost algorithm

Amal Asselman, Mohamed Khaldi & Souhaib Aammou

To cite this article: Amal Asselman, Mohamed Khaldi & Souhaib Aammou (2023) Enhancing the prediction of student performance based on the machine learning XGBoost algorithm, Interactive Learning Environments, 31:6, 3360-3379, DOI: 10.1080/10494820.2021.1928235

To link to this article: https://doi.org/10.1080/10494820.2021.1928235

INTERACTIVE LEARNING ENVIRONMENTS
2023, VOL. 31, NO. 6, 3360–3379
https://doi.org/10.1080/10494820.2021.1928235

Enhancing the prediction of student performance based on the machine learning XGBoost algorithm

Amal Asselman, Mohamed Khaldi and Souhaib Aammou

LIROSA laboratory, Faculty of Sciences, Abdelmalek Essaadi University, Tetouan, Morocco

ABSTRACT

Performance Factors Analysis (PFA) is considered one of the most important Knowledge Tracing (KT) approaches used for constructing adaptive educational hypermedia systems. It has shown a high prediction accuracy against many other KT approaches. While, the desire to estimate more accurately the student level leads researchers to enhance PFA by inventing several advanced extensions. However, most of the proposed extensions have exclusively been improved in a pedagogical sense, as the improvements have mostly been limited to the analysis of students' behaviour during their learning process. In contrast, Machine Learning provides many powerful methods that could be efficient to enhance, in the technical sense, the prediction of student performance. Our goal is to focus on the exploitation of Ensemble Learning methods as an extremely effective Machine Learning paradigm used to create many advanced solutions in several fields. In this sense, we propose a new PFA approach based on different models (Random Forest, AdaBoost, and XGBoost) in order to increase the predictive accuracy of student performance. Our models have been evaluated on three different datasets. The experimental results show that the scalable XGBoost has outperformed the other evaluated models and substantially improved the performance prediction compared to the original PFA algorithm.

ARTICLE HISTORY
Received 17 March 2021
Accepted 3 May 2021

KEYWORDS
Performance factors analysis; knowledge tracing; prediction accuracy; machine learning; ensemble learning; XGBoost

Introduction

Constructing an effective adaptive education system requires the development of a more accurate student model that can collect data about students in a more automated way. Therefore, Knowledge Tracing (KT) is one of the most effective method used for estimating the cognitive level of the learner ((Corbett & Anderson, 1994), (Baker et al., 2008), and (Pelánek, 2017)). In this regards, several approaches based on machine learning classifiers have been widely used to serve the estimation of student knowledge and the prediction of his future performance. Several research papers have experimentally proved that the Performance Factors Analysis (PFA) approach has shown high predictive accuracy against many other KT approaches (Gong et al., 2011), (González-Brenes et al., 2014), (Xiong et al., 2016), and (Minn et al., 2018). Even so, the overwhelming desire to innovate the existing knowledge tracing models prompts researchers to suggest advanced extensions in order to get more accurate predicting results. However, these extensions have been improved almost exclusively in a pedagogical sense. More precisely, researchers have exclusively evaluated the impact of some students' behavior during their learning process for applying the improvement. In fact, we do not deny the significant improvement that the analysis of student behavior and features correlation

CONTACT Amal Asselman asselman.amal@gmail.com LIROSA laboratory, Faculty of Sciences, Abdelmalek Essaadi University, Tetouan, Morocco

could ascribe to the original KT algorithms. Nevertheless, it should also be noted that the result obtained by most of the proposed contributions could be further improved if researchers take into consideration the technical aspects. Among them, we refer to the use of certain methods and techniques provided by data scientists to optimize predictive algorithms. Regarding the technical improvements that were resorted to, we cite the update made by Piech Chris et al. (Piech et al., 2015), Which has shown how it can achieve very high results by simply making some changes to the BKT approach through the adoption of Deep Learning (DL) algorithms. Due to the outstanding performance that it has shown, and to its ability to implement complex problems, this approach becomes commonly used by many researchers and widely applied to identify student performance. However, given that deep learning usually requires many environmental conditions, several researchers are obliged to use KT machine learning approaches to avoid certain DL requirements. From this paper, we aim to evaluate whether the KT machine learning approach, the PFA algorithm in our case, can be improved so that it becomes more powerful if its use is chosen instead of a deep learning approach for specific conditions.

In terms of efficiency, machine learning suggests a plethora of methods available for improving the results of its classical algorithms. As example, Ensemble Learning represents an extremely powerful machine learning paradigm for data classification. Moreover, although the success and the widespread use that deep learning has recently known, the ensemble learning algorithms are used, in several cases, as alternative methods to deal with many supervised learning problems since deep learning algorithms require lots of data, computational power, and large execution time for training and testing the model. In order to address these limitations, a new approach for enhancing PFA model has been proposed. The experiment has been performed to exploit the power of Ensemble Learning, as one of the most popular optimization techniques to improve student performance machine-learning algorithms. For this purpose, we have manipulated the original PFA as a basic algorithm used according to its higher performance to model cognitive learner profiles on various e-learning system. The main objective of this study is to identify to what extent the implemented model could improve the prediction result of future student performance. Our study has experimentally been validated using three real-world datasets from different Intelligent Tutoring System.

The rest of the paper is organized as follows. Section 2 provides a state-of-the-art methods and the related works about student modeling and Knowledge Tracing, then it highlights the theoretical background about Ensemble Learning. Section 3 introduces the machine learning algorithms that we refer to for constructing our models, it discusses the research methods applied in this study, then it presents the data collection used. Section 4 investigates the performance of the proposed approaches compared to the original PFA model, further; it discusses the implications and some requirements of the experimental models. Finally, Section 5 concludes the contribution and mentions the future scope of the work.

Related work

The research reported in this article is part of the optimization of tracing knowledge methods through machine learning techniques. In this context, we will review the methods closely related to this work: works relevant to knowledge tracing approaches are reviewed in the first section, and ensemble learning, as an optimization method of Machine Learning, is investigated in the second section.

Student Modeling and knowledge tracing

Adapting educational systems has become an efficient technique used to ensure a personalized and flexible learning process following the evolution of learners' profile. The personalization serves to monitor learner's behavior in order to respect his needs and his psychological state. Accordingly, modeling student features remains the key component of any adaptive educational system, as it stores all the information about the student such as his knowledge, interest, learning styles, etc. ((Koch, 2001), (Millán et al., 2010), and (Weibelzahl et al., 2020)). The construction of an effective student model allows the tutor to suggest the best hypermedia contents and the relevant multimedia resources to the student, and consequently, hide the content that can be very easy or highly complicated for him ((De Vrieze et al., 2005), (Brusilovsky & Millán, 2007), (Chrysafiadi & Virvou, 2015), and (Chrysafiadi et al., 2020)). Student modeling presents the process of tracking the entire lifecycle of a learner model; including the acquisition of student knowledge, construction, update, and exploitation of the student's model (Koch, 2001). In fact, the acquisition of student models recognized major difficulties due to the unpredictable change in learner behavior over time, lack of data about a student, and inaccurate measurement of student variables. As result, various learning systems still have limitations at the level of constructing a powerful student model. This means that these systems, for example, remain insufficient to identify more precisely the student cognitive features (such as his capacity of memorization), to determine student effect (such as student motivation), or to predict student performance during answering a new question ((Asselman et al., 2018), (Pelánek, 2017), and (Shahiri & Husain, 2015)).

In the literature, several studies described that the estimation of student features and the prediction of his future performance remain a challenging technique in the educational research area ((Baker et al., 2008), (Chrysafiadi & Virvou, 2015), and (Khajah et al., 2014)). Many approaches for tracing student's knowledge have been tried earlier, using the principle of machine learning methods, in order to develop an automatic detector of student behaviour ((Khajah et al., 2014), and (Hussain et al., 2019)). These methods offer the potential to support teaching and learning aspects that are considered time-consuming and difficult to manage. Among the methods we cite, classical Item Response Theory (IRT), which has been developed earlier using the logistic regression algorithm in order to estimate student's knowledge (Hambleton et al., 1991). It has been created to offer the modeling process the capacity to identify different student abilities and problem difficulties. A few years later, Corbett and Anderson (Corbett & Anderson, 1994) have proposed the Bayesian Knowledge Tracing (BKT) model to identify student-level progress over time. The proposed approach is intended to estimate whether a student has learned a given knowledge component (KC). By exploiting a student's sequence of problem-solving, the algorithm attempts to determine the point at which a skill is mastered. Many variations of the original BKT model have been introduced in order to fit different learning environments. For instance, the original version has been adopted through the identification of student guessing and slipping factors related to his prior knowledge (Baker et al., 2008). In addition, Pardos and Heffernan individualized the prior learning parameter by adding a student node to the BKT model (Pardos & Heffernan, 2010), then they introduced the difficulty of a Knowledge Component by adding additional nodes into the topology of the original BKT model (Pardos & Heffernan, 2011). However, the BKT still unable to deal with multiple concepts involved in the same task. For the reason, another approach is Performance Factors Analysis (PFA), presented by Pavlik et al. ((et al., 2009) and (et al., 2009)) has as purpose the estimation of student knowledge and the prediction of his future performance. The PFA approach has been used in several research studies due to its ability to handle multiple skill questions (Pelánek, 2017). In the literature, most of these studies (Gong et al., 2011), (González-Brenes et al., 2014), (Pelánek, 2015), (Xiong et al., 2016), and (Minn et al., 2018), have demonstrated that the basic PFA algorithm could predict student performance more efficiently compared to the BKT approach. Later, based on Deep Learning approaches, Piech Chris et al. have presented the Deep Knowledge Tracing model (DKT) (Piech et al., 2015). The proposed approach consists of training the tags of each realized exercise on a neural network that uses LSTM2 as the unit cell. The invented approach has become widely used by many researchers, and known various improvements over the days. However, several researchers still using machine learning approaches for tracing student behaviour according to the constraints that deep learning requires, such as the data length, the complexity of computation, and the higher training and testing time that it demands.

In addition to identify which basic approach remains the most effective for tracking student's knowledge, many advanced extensions to the cited original knowledge tracing algorithms have been developed. By studying several student behaviors, these extensions have mainly been constructed to increase the interpretability of the student model and to improve the prediction of student performance. In fact, all improvements proposed based on learner behavior analysis could be reapplied identically with most of the original KT approaches, rather than being exclusively applied to a specific model. These improvements have been applied in various aspects. Many researchers have focused their attention on the integration of the amount of help that a student requires for answering questions (Wang & Heffernan, 2011), (Wang & Heffernan, 2013), and (Asselman et al., 2020). In the same direction, several improvements have been accomplished based on the analysis of item response time (Xiong & Pardos, 2011), (Papoušek et al., 2015), (Rihák, 2015), (Pelánek, 2018), and (Chounta & Carvalho, 2019).

Generally, many researchers, in certain cases, are obliged to use machine learning approaches instead of using deep learning algorithms in order to avoid its constraints and requirements. As we could also notice that most of these researchers manipulate the PFA algorithm or its extensions because of its various advantages. In addition, researchers in the field of KT are most interested in studying exclusively the effect of students' behavior during their learning process. For the reason, machine learning could provide many powerful optimization methods that could be efficient to enhance technically the prediction of student performance, whether it is used alone or in parallel with any pedagogical improvements.

Ensemble learning

Optimization is an extremely important operation of machine learning. Generally, it involves several processing methods, such as the application of mathematical optimization techniques and the in-depth analysis of the data constituting the predictive model. Together, the two mentioned optimization processes are used to constitute the so-called "Ensemble Learning". Recently, this innovative method is becoming highly recommended by data scientists for improving machine learning algorithms, as it focuses on designing a robust model to obtain more accurate results in various practical tasks (Zhou, 2009). It is also recognized by "Multiple Classification Systems" (Woźniak et al., 2014). Ensemble Learning, being a branch of machine learning, created based on the following assumption: although it can be difficult to build a very accurate classifier (strong learner), it is much easier to find many approximate rules (weak learners) that come together to create a model with high generalization capacity (Zhou, 2012). Its use has led to an increased robustness and generalization of statistical models. Ensemble Learning can be used for both supervised learning tasks: classification as well as regression. Both of them have proven their performance due to their ranking in many prestigious machine learning competitions, such as the Kaggle Netflix competition, and KDD 2009. As shown in Figure 1, the concept of ensemble learning essentially boils down to the fact that a set of weak classifiers $F_n(x)$ with $n \in \{1, \ldots, N\}$ are combined to form the final strong classifier $F(x)$.

In general, during constructing an ensemble learning classifier, basic learners could be created using decision trees, neural networks, or other types of learning algorithms. Indeed, there are

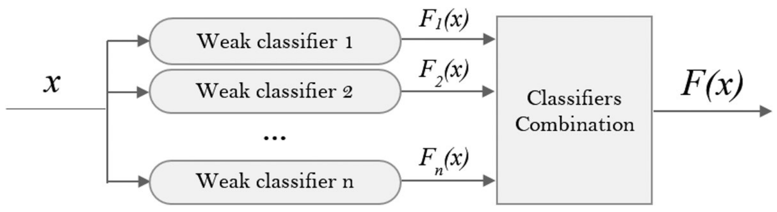

Figure 1. Concept of Ensemble classifiers (Zhou, 2012).

several Ensemble methods favor implementing their weak classifiers based on a single type of learning algorithm, while other methods use different basic algorithms to generate the strong classifier (Zhou, 2012). In fact, whenever developers use a single classifier, they mostly try to use more and more features to get better accuracy. However, the wrong choice of the number of used attributes of the created classifier can certainly lead to bias or variance errors. In practice, as the complexity of the prediction model increases, the resulting error decreases due to having a lower rate of bias. While, as the model becomes more and more complex, it will eventually face the problem of Overfitting, so that the model will start to decrease in performance level due to high variance. To deal with these problems, using Ensemble Learning denotes one of the best decisions made, as it has defined several effective methods to solve learning problems. As described in Figure 2, we introduce two of the most popular methods, namely BAGGing and Boosting.

- **BAGGing methods:** was invented by establishing multiple independent estimators instead of using only a single one, in order to obtain a powerful model with more efficient results (Breiman, 1996). The BAGGing method is created based on two techniques: 1) Bootstrap as a technique of splitting datasets, 2) AGGregation technique, which refers to the collection of the results obtained by each predictor variable to form the final prediction result. In the Bootstrap technique, new sub-datasets are created by deriving instances of the global dataset using the replacement procedure. This process is mainly used to randomly select dataset instances, where each instance can be selected multiple times to create sub-datasets. Subsequently, the aggregation technique is used to incorporate all possible prediction results in order to select the final prediction result. In the case of regression, the final result is selected by calculating the average of the results obtained by all weak learners, while, in the case of binary classification, the selection of the final result adopts the principle of majority vote.
- **Boosting methods:** is partially different from the BAGGing method in that it focuses on reducing the prediction error rather than minimizing the variance, and also because it is performed in a sequential manner (Freund & Schapire, 1997). In Boosting algorithms, each classifier is trained on data, taking into account the success of previous classifiers. At each instant t, the results of the model are weighted according to the results of the preceding instant $t-1$. After each training iteration, the weights are redistributed so that correctly predicted outcomes are given a lower weight, and output that is misclassified increases their weights to highlight more difficult cases. In this way, subsequent learners will focus on the more weighted instances during their training.

In our work, we refers to the two cited methods for constructing our approach.

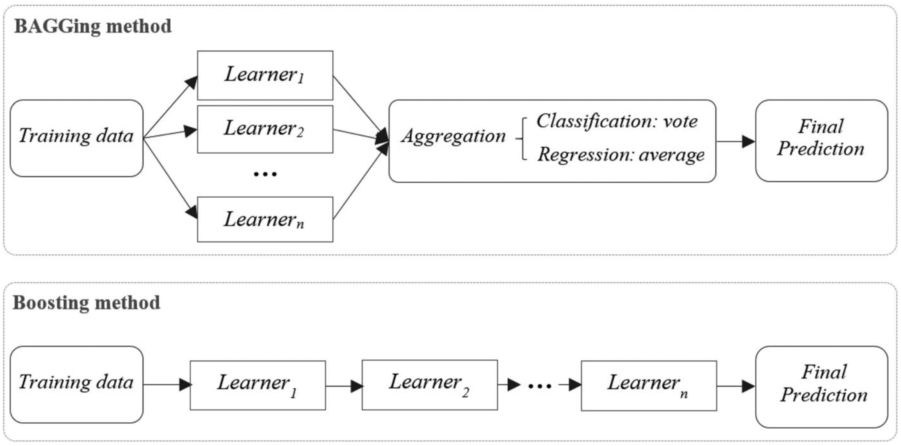

Figure 2. A simplified vision of BAGGing and Boosting methods.

Methodology

In this section, we describe the overall methodology of the proposed approach, and we present the details of its implementation. For the purpose, firstly, we present the machine learning basic approaches that we refer to for constructing our models, namely Performance Factors Analysis as a Logistic Regression classifier, Random Forest, AdaBoost and XGBoost algorithms. Then, we introduce the research method applied in this study. We describe all the datasets used for training and testing the models constructed, as well as the criteria of evaluation used for measure our classifiers.

Machine learning classifiers

Performance Factors Analysis (Logistic regression)

Performance Factors Analysis (PFA) ((et al., 2009) and (et al., 2009)) is an example of knowledge tracing approach. It has been extensively used to model the cognitive profiles of learners on various Intelligent Tutoring Systems for predicting their performance based on their past actions. It is an instance of the logistic regression classifier. It should be noted that PFA has been designed especially for being able to handle multiple skill problems (Pelánek, 2017). It has shown an effective estimation of student skill acquisition and an accurate prediction of future performance. The model estimation outperforms many other knowledge-tracing models such as Bayesian Knowledge Tracing ((Gong et al., 2011), (Wang & Heffernan, 2011), (González-Brenes et al., 2014), (Xiong et al., 2016), and (Minn et al., 2018)). The PFA original equations (1) and (2) take the following form (et al., 2009):

$$m(i, j \in KC_s, k \in Items, S, F) = \beta_k + \sum_{j \in KC_s} (\gamma_j S_{ij} + \rho_j F_{ij})$$

$$P(m) = \frac{1}{1 + e^{-m}} \tag{2}$$

The equation (1) represents the logit function. It aims to estimate a student's knowledge state. While the equation (2) is used to determinate the prediction of the correctness of his future item (Pelánek, 2017), where:

- S_{ij} and F_{ij} track respectively the prior successes and the prior failures for student i on KC_j.
- The parameters γ_j and ρ_j scale the effect of the observation counts on KC_j.
- The parameter β_k denotes the easiness of item k.

Fitting parameters of: β, γ, and ρ can easily be accomplished using gradient descent principles for standard logistic regression implementation. This classifier is an elaboration of the Rasch model from Item Response Theory, which has an equivalent form to equation (1) with γ and ρ set to 0 and using only a single β value.

Random Forest classifier

Random forest was developed combining the BAGGing method (Breiman, 1996) with the random variable selection, in order to merge a group of "weak learners" together to form a "strong learner". In this algorithm, each decision tree within the group is created by means of a sample with replacement from the training data. Each decision tree within the group acts as a base estimator to establish the class label of an unlabeled instance. This is accomplished through majority votes. Bootstrap sampling is used for each tree of Random Forest, and spliting data according to the presented problems type (Classification or Regression). For classification, the Gini index is used to split the data; for regression, minimizing the sum of the squares of the mean deviations can be selected to train each tree model. Benefits of using RFs are that the ensembles of trees are used without pruning.

In addition, RF is relatively robust to Overfitting, and standardization or normalization are not necessary because it is insensitive to the range of value. Two parameters should be adjusted for RF model: the number of trees and the number of features randomly sampled at each split.

Adaboost classifier

The AdaBoost algorithm (or adaptive Boosting), introduced in 1995 by Freund and Schapire (Freund & Schapire, 1997), is the most influential, classical and wellknown boosting algorithm that has solved many of the practical difficulties of the earlier boosting algorithms. The algorithm takes as input a training set $\{(x_i, y_i)\}^n_{, \text{p. }11}$ of n input examples and m features, where each x_i belongs to some domain or instance space X, and each label y_i is in some label set Y where $Y = \{-1, +1\}$ AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds $t = 1, \ldots, T$. The main idea of AdaBoost is to focus on instances (samples) that were previously misclassified to form a new model. It handles a distribution that represents the weights of each instance. This distribution, D_t, will change in every iteration t, i.e. every time AdaBoost calls the weak classifier. At the beginning, the weights of each sample are initialized to $D_1 = \frac{1}{N}$ for the first estimate of the model and change with each estimate. On each iteration, AdaBoost adjusts the weights of the data instances, so that the weights of data points misclassified by previous classifiers are increased and the weights of data points correctly classified by previous classifiers are decreased. AdaBoost trains a weak classifier, it returns a weak hypothesis, h_t, and we want to choose a h_t. that minimizes the weighted error:

$$\varepsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i] \tag{3}$$

Once the weak hypothesis has been received, AdaBoost chooses a parameter, αt, which measures the weak hypothesis's importance. That way, the more accuracy h_t gets, the more importance we assign to it. This parameter is defined as

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{4}$$

Which distributes more weight among well-classified examples than among misclassified examples. By taking into account α_t and h_t, AdaBoost updates the distribution for every example x_i with label y_i:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} exp(-\alpha_t) \text{ if } h_t(x_i) = y_i \\ exp(\alpha_t) \text{ if } h_t(x_i) \neq y_i \end{cases} \tag{5}$$

Where Z_t is the selected normalization factor, so D_t is equal to 1. After all the iterations T are complete, AdaBoost returns a final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{6}$$

XGBoost classifier

XGBoost stands for eXtreme Gradient Boosting (Chen & Guestrin, 2016). It represents an instance of Gradient Boosting Machine (GBM) as a technique used mainly in the construction of both regression and classification predictive modeling problems. Most existing GBM models are consistently outperformed other machine learning algorithms, ie. it has shown better performance on various machine learning reference data sets (Friedman, 2001). XGBoost is an ensemble method where new models are created to correct the residuals or errors of prior models then combined together to make the final prediction. Experimentally, XGBoost is relatively faster than many other ensemble classifiers (such as AdaBoost). The impact of the XGBoost algorithm has been widely recognized in many machine learning and data mining challenges, where it has become a more commonly used and popular tool among Kaggle's competitors and the data scientist of the industry (Nielsen, 2016). In addition, it is a parallelizable algorithm, i.e. it can harness the power of multi-core computers, which also allows training on very large data sets. Moreover, the XGBoost is free open source software available for use under the permissive Apache-2 license.[1]

For better explaining Boosting algorithm, we assume for a given data set: $D = (x_i, y_i): i = 1, \ldots, n,\ x_i \in \mathbb{R}^m,\ y_i \in \mathbb{R}$, we have n observations with m features. Let $\hat{y}_i$ be defined as the predict value by the model:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ \ f_k \in K \tag{7}$$

Chen and Guestrin (Chen & Guestrin, 2016) introduced the eXtreme Gradient Boosting (XGBoost) as described in equation (8). At each iteration of gradient boosting, the residual will be manipulated to correct the previous predictor that the specified loss function can be optimized. The $f_k(x_i)$ corresponds to the prediction value given by the k^{th} tree to the i^{th} sample. The set of functions f_k can be learned by minimizing the objective function:

$$Obj = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{8}$$

With

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\omega^2 \tag{9}$$

The first term l in the equation (8) is defined in the classification tree by $(\hat{y}_i, y_i) = \ y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i)\ln(1 + e^{-\hat{y}_i})$. It represents the loss function and it measures the difference between the predicted value $\hat{y}_i$ and the target value y_i. While the second term Ω in the equation (8) represents the regularization term; a factor used to measure the complexity of the tree f_k. Where γ and λ are the degrees of regularization. T and ω. used in the equation (9) are the numbers of leaves and the vector of values attributed to each leaf respectively

We can then optimise to find the f_k which minimises the objective function. Taking the Taylor expansion of the above fution to the second order allows us to easily accommodate different loss functions:

$$Obj^{(t)} \simeq \sum_{i=1}^{n}\left[l(y_i,\ \hat{y}_i^{(t-1)}) + g_i f_i(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t) \tag{10}$$

Where and $h_i = \partial^2_{\hat{y}^{(t-1)}}\ l(y_i, \hat{y}_i^{(t-1)})$ are first and second $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ order gradient statistics on the loss function. By removing the constant term, we obtain the following approximation of the equation (11) by expanding Ω as follows:

$$\begin{aligned} Obj^{(t)} &= \sum_{i=1}^{n}\left[g_i f_i(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i \in I_j} g_i \right) \omega_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda \right) \omega_j^2 \right] + \gamma T \end{aligned} \tag{11}$$

Where $I_j = \{i | q(x_i) = j\}$ denotes the instance set of leaf j. For a fixed tree structure q, the optimal weight w_j^* of leaf j and the corresponding optimal value can be calculated by: $w_j^* = -\frac{G_j^2}{H_j + \lambda}$.

And after substituting w_j^* into equation (11), there exists:

$$Obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \lambda T$$

Where $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$. In practice, it is impossible to enumerate all the possible tree structures q. A greedy algorithm that starts from a single leaf and iteratively adds branches to the tree is used instead. Whether adding a split to the existing tree structure can be decided by the following function:

$$G = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda} + \frac{\left(\sum_{i\in I_I} g_i\right)^2}{\sum_{i\in I_I} h_i+\lambda}\right] - \gamma$$

Assume that I_L and I_R are the instance sets of left and right nodes after the split.

Experimentation

Ensemble Learning is one of the most powerful machine learning optimization method, which gained great popularity in recent years. It has demonstrated a strong ability to improve many research works. Among them, according to Chen and Guestrin (Chen & Guestrin, 2016), Kaggle's blog in 2015 published that 17 out of 29 winning solutions used XGBoost in their models. Due to the advantages of ensemble learning classifiers over many other ML single classifiers, we assume that the use of BAGGing and Boosting methods could improve the prediction results of student performance in adaptive hypermedia learning systems. This process would allow tutors to become more able to select reasonably the appropriate multimedia sources and hypermedia content adapted to the profile of each learner. Our objective from this experiment is to identify whether the implication of the Ensemble Learning methods can improve the performance prediction, or the simple use of a single classifier may eventually lead to sufficient improvements. For this reason, we have first implemented the original PFA model as a reference to judge the results of the proposed approach. Then, we have implemented the FPA model based on the Random forest algorithm, as the BAGGing algorithm, due to its high performance in several research studies. In addition, we have tested the PFA model using AdaBoost and XGBoost classifiers as Boosting solutions.

The proposed approach is illustrated in Figure 3, to get a simplified view of how our approach is structured. To estimate future student performance, the implementation of a predictive model generally follows all the usual processes, as ensemble learning algorithms can be used very easily. In this case, researchers can use our contribution in parallel with the pedagogical adjustments they want to suggest, and then choose the ensemble learning algorithm that can give the best results.

In contrast to logistic regression classifiers, using the three ensemble learning algorithms requires developers to tune some different parameters in order to increase the performance of these models as well as prevent Overfitting problems. In order to automate the process and estimate the best hyper-parameters values of our proposed models, we have applied *BayesSearchCV* optimization algorithm for tuning the optimal values of parameters. In our experiment, the *BayesSearchCV* has been implemented in Python environment with the support of open-source libraries. The *param_grid* of *BayesSearchCV* requires a list of parameters and a range of values for each parameter of a specified estimator. For several attempts, we have provided a list of values to each hyper-parameter of the models to choose the best fitted one. Moreover, the parameters of the estimator that are used to apply these methods are optimized by cross-validated grid-search over a parameter grid in order to provide the best accuracy levels. Using AdaBoost classifiers require the tuning of two hyper-parameters: *n_estimator* and *learning_rate* taking the values described in Table 1.

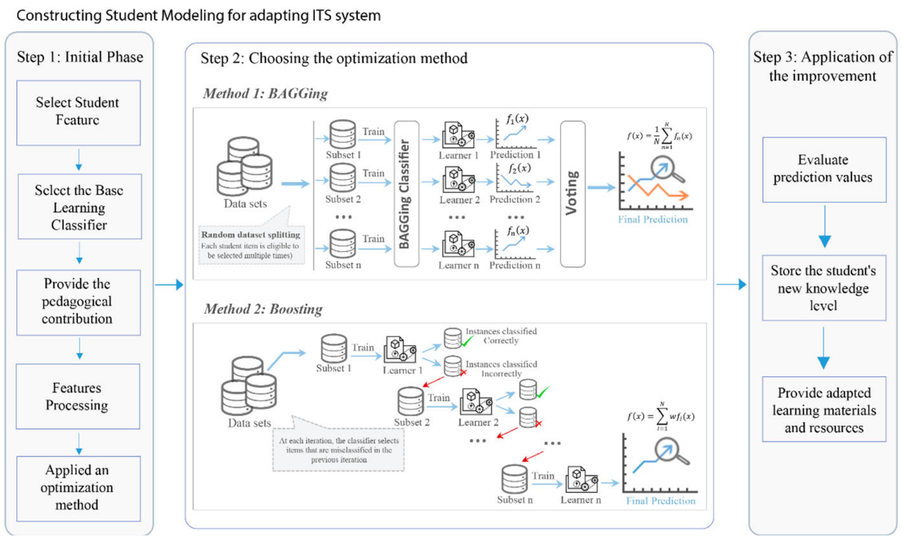
Constructing Student Modeling for adapting ITS system
Step 1: Initial Phase
Select Student Feature
Select the Base Learning Classifier
Provide the pedagogical contribution
Features Processing
Applied an optimization method
Step 2: Choosing the optimization method
Method 1: BAGGing
Data sets
Random dataset splitting
Each student item is eligible to be selected multiple times
Subset 1
Subset 2
Subset n
Train
BAGGing Classifier
Learner 1
Learner 2
Learner n
Prediction 1
Prediction 2
Prediction n
$f_1(x)$
$f_2(x)$
$f_n(x)$
Voting
$f(x) = \frac{1}{N}\sum_{n=1}^{N} f_n(x)$
Final Prediction
Method 2: Boosting
Data sets
At each iteration, the classifier selects items that are misclassified in the previous iteration
Subset 1
Learner 1
Instances classified Correctly
Instances classified Incorrectly
Subset 2
Learner 2
Subset n
Learner n
$f(x) = \sum_{i=1}^{N} wf_i(x)$
Final Prediction
Step 3: Application of the improvement
Evaluate prediction values
Store the student's new knowledge level
Provide adapted learning materials and resources

Figure 3. A simplified vision of our approach.

The use of Random Forest classifier requires tuning the following hyper-parameters:

- *n_estimators*: describe the number of trees in the foreset
- *max_depth*: define the max number of levels in each decision tree
- *min_samples_split*: represents the min number of data points placed in a node before the node is split
- *min_samples_leaf* : specifies the min number of data points allowed in a leaf node

The values of the parameters selected by the adjustment process are defined in Table 2:

For tuning our XGBoost model, we have set three categories of parameters: general parameters, task parameters, and booster parameters. Based on the first category, we aim to define the overall functionality of XGBoost and controls the booster type used in the model. In our work, we have set two attributes, namely: *booster* and *nthread*. The type of *booster* has identified by *gbtree* value, as our approach is implemented for classifying learner questions. While the *nthread* attribute is used to provide the number of parallel processing required for running the XGBoost algorithm. In our case, we have used our local machine for tuning the approach. For all used datasets, we have set this attribute to 7 threads to make the execution faster.

By referring to the learning task parameters, we have defined the optimization objective of the metric that we have calculated at each step. Concerning the objective parameter, we have selected *binary: logistic* to apply logistic regression for binary classification. This parameter returns the predicted values in the form of probabilities. And for evaluating the invented approach, we have set the *eval_metric* by error in order to create a strong model able to correct the residuals of simple classifiers to obtain better final prediction results.

According to the booster parameters, we have used the best set of hyper-parameter values chosen in the *BayesSearchCV* method, in order to control the model performance depending on

Table 1. Tuning booster parameters values used for AdaBoost method.

Hyper-parameters	ASSISTments data	Andes data	Simulated data
n_estimator	300	290	300
learning_rate	0.1	0.08	0.05

Table 2. Tuning booster parameters values used for Random Forest method.

Hyper-parameters	ASSISTments data	Andes data	Simulated data
n_estimator	300	250	300
min_samples_split	12	8	12
max_depth	12	40	64
min_samples_leaf	4	8	5

the booster that we have selected. Table 3 presents the best selected values of each parameter for all the tested datasets. This category contains several parameters appropriate for tree and linear boosters. As we present above the tree-specific parameters:

- *eta*: refers to a step size shrinkage used for controling the learning rate and preventing Overfitting. After each boosting step, we can directly get the weights of new features, and then eta shrinks the feature weights to make the boosting process more conservative. In practice, using a smaller *eta* makes our model more robust to Overfitting.
- *min_child_weight*: defines the minimum sum of instance weight needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min child weight, then the building process will give up further partitioning. The larger *min_child_weight* is, the more conservative the algorithm will be.
- *max_depth*: presents the maximum number of nodes allowed from the root to the farthest leaf of a tree. Increasing the value of this parameter will make our model more complex and more likely to overfit.
- *max_delta_step*: specifies the maximum delta step to allow for each tree's weight estimation. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help to make the update step more conservative. This parameter might help in case of the logistic regression when class is extremely imbalanced.
- *subsample*: presents the subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees, and this will prevent Overfitting. Subsampling will occur once in every boosting iteration.
- *colsample_bytree*: refers to the subsample ratio of columns when constructing each tree. Subsampling occurs once for every constructed tree.
- *colsample_bylevel*: defines the subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.
- *reg_alpha* and *reg_lambda*: refers, respectively, to *L1* and *L2* regularization term on weights.

Data sets description

For evaluating our hypothesis, we tested the proposed approach against the original PFA on three different datasets. Following is the description for each dataset:

- *ASSISTments 2009 – 2010*: is a real dataset extracted from the ASSISTments platform (Feng et al., 2009). It is an Intelligent Tutoring System available online mainly for teaching math courses for middle and high school students. The dataset was gathered from the 2009–2010 school years. It is available in ASSISTments website[2] for any experimental research. This version of a dataset is used due to the fact that we manipulated a multiskilling approach (PFA). The subset used in this work is about 123 skills (e.g. Division, Area of polygons, Pythagorean theorem, etc.), where 245 students are working on 13482 problems with a total number of 115956 responses. We have used 104361 items to train the model and 11595 to evaluate the prediction of future performance.

Table 3. Tuning booster parameters values used for XGBoost method.

Hyper-parameters	ASSISTments data	Andes data	Simulated data
eta	0.131	0.139	0.162
n_estimator	300	300	300
min_child_wight	1	1	9
max_depth	11	8	8
max_delt_step	0	0	0
subsample	1	0.8	1
colsample_bytree	0.949	0.978	0.894
colsample_bylevel	0.94	0.979	0.853
reg_lambda	6.52e-06	1	0.2
reg_alpha	0.140	0	0.0326

- *Andes 2011–2012*: is a real dataset extracted from the Andes Physics Tutor for teaching Physics Course specially, Mechanics. The dataset used in our experiment can be found on PSLC DataShop[3] (Koedinger et al., 2010). The used subset is constructed of 117 students and 119 skills. The used dataset consisted of a total number of 68573 responses, in which students were working on 11064 problems. We have split the dataset into two groups, where we have used 61716 items to train the models and 6857 to evaluate the accuracy of predicting future performance.
- *Synthetic dataset*: is generated according to a certain number of criteria. We have carefully defined them in order to create a data with a structure similar to the ASSISTments dataset. Among these criteria, we notice that our generated dataset has been constructed from 100 simulated students and 20 skills. We have also considered that each student answers between 8 and 15 questions per skill. Our dataset is consisted of a total number of 171459 responses, in which students were working on 1515 problems. Splitting dataset has been done where we have used 154314 items to train the models and 17145 to evaluate the accuracy of predicting future performance.

For all the three used datasets, we have split samples into training and test set by using a 10 K-fold cross-validation method.

Evaluation criterion

The next section aims to describe and discuss the results obtained with our models. The evaluation of the implemented models has been achieved with the help of the confusion matrix (Pardos & Yudelson, 2013). This latter allows the specification of the relationships between the attributes of real classes and those of predicted classes. By counting the number of correct and incorrect classifications in each possible value of the attribute in the confusion matrix, the effectiveness of the classification model can be estimated. In our context, the main objective of the use of this matrix is to evaluate the ability of the proposed models in predicting student performance. Given two classes, the confusion matrix is described in Table 4 (Pardos & Yudelson, 2013). *TP* refers to the case where the model correctly classifies the correct answers of learners. *TN* refers to the number of answers that are actually classified as incorrect, but the model predicts them as correct. *FP* presents the case where a learner does not respond correctly to an item, but the prediction model classifies the learner's response as correct. And *FN* refers to the case that even if the learner answers an item correctly, the prediction model classifies their answer as incorrect.

Table 4. Confusion Matrix for a binary classifier.

		Actual	
		Correct	Incorrect
Predicted	**Correct**	True Positive **(TP)**	False Positive **(FP)**
	Incorrect	False Negative **(FN)**	True Negative **(TN)**

Generally, for evaluating the performance values predicted by binary classification algorithms, ML provides a set of metrics to analyse the predictive accuracy. Among those, we have used:

- **Accuracy**: represents one of the most commonly used metrics for rating performance. It estimates the proportion of correctly classified of students' answers by measuring a ratio between the answers correctly classified and the total number of answers. (Accuracy $= \frac{TP + TN}{TP + TN + FP + FN}$)
- **Precision**: represents the percentage of positive students' answers that correctly classified divided by the total number of expected positive responses. (Precision $= \frac{TP}{TP + FP}$)
- **Recall**: represents the number of positive students' answers that are properly classified divided to the total number of answers that are supposed to be answered correctly.(Recall $= \frac{TP}{TP + FN}$)
- **F1-measures**: represents the harmonic mean of precision and recall. It is used as a statistical measure to rate performance in which high values of F1-measures indicate high classification performance. (F1-Score $= \frac{2 \times precision \times rappel}{precision + rappel}$)

Results

In order to test our experiment, we have implemented our models using an Intel Core i7 processor with 2.4 GHz frequency and 8 G of memory. Once these models have been established, we assessed their performance using the evaluation metrics described in the subsection (3.4). In this study, all the constructed models have been experimentally validated against the original formulation of PFA. First, we have started the analysis by comparing the misclassification rate of each algorithm. As a reference, we have used Table 5, which displays the confusion matrices obtained by the original PFA algorithm for all three datasets used. Tables 6, 7, and 8 respectively show the confusion matrices for all the implemented algorithms obtained based on ASSISTments, Andes, and the synthetic datasets.

In order to make these tables more readable, we have summarized their results using Figure 4.

From Figure 4, for the three tested datasets, the XGBoost classifier has shown the least number of misclassified instances. The figure also notices that, compared with the Radom Forest and the AdaBoost algorithms, the PFA logistic regression, as single classifiers, has obtained a high numbers of misclassified instances. Basically, based on the described results, it seems that the Gradient Boosting method provides the best performance compared to the other methods used.

After evaluating the models implemented using the confusion matrices, we use Table 9, to compare the results obtained by the original PFA model against the models of our proposed approach. The comparison has been made according to four important metrics that are mostly used for evaluating the prediction accuracy.

From table 9, we find that all the implemented models have shown a positive impact on the original PFA model, in which the best results are highlighted in bold type. By analysing the results obtained, we observe that our proposed XGBoost model has outperformed with the highest Accuracy score on the test set for the three used datasets. We have found that, for all the tested datasets,

Table 5. Confusion Matrices of PFA Original models for the three datasets.

	ASSISTments 2009–2010		Andes 2011–2012		**Simulated Dataset**	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Correct	8 933	2 441	4 837	1 662	10 829	4 599
Incorrect	93	128	124	234	422	1 295

Table 6. Confusion Matrices of the three models created based on the ASSISTments 2009–2010 dataset.

	ASSISTments 2009–2010		Andes 2011–2012		**Simulated Dataset**	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Correct	8 883	2 379	8 796	2 285	8 749	2 187
Incorrect	143	190	230	284	277	382

Table 7. Confusion Matrices of the three models created based on the Andes 2011–2012 dataset.

	ASSISTments 2009–2010		Andes 2011–2012		**Simulated Dataset**	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Correct	4 752	1 546	4 711	1 507	4 731	1 491
Incorrect	209	350	250	389	230	405

Table 8. Confusion Matrices of the three models created based on the synthetic dataset.

	ASSISTments 2009–2010		Andes 2011–2012		**Simulated Dataset**	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Correct	10 705	4 450	10 558	4 237	10 565	4 114
Incorrect	546	1 444	693	1 657	686	1 780

the accuracy metrics showed respectively an increase of (7.35%, 4.5%, 4.2%) compared to the original PFA algorithm. Regarding the other evaluated ensemble learning classifiers, we have acquired an improvement of 6.85% and 6.90%, respectively, for the Random Forest, and AdaBoost models during using the ASSISTments dataset. Similarly, for the other datasets, these classifiers have shown a close effect to the results obtained. We notice that the results of Random Forest and AdaBoost are not much different. Undeniably, using classification accuracy to measure the performance of our models gives an impression about how each model has performed, but, in machine learning it is not enough to truly judge the models. We have reinforced the comparison by using precision, recall, and F1-measure metrics. Regarding the XGBoost classifier, the obtained values confirm the results provided by the accuracy metric, where we mention that, for the three used datasets, this model has outperformed all other implemented models. In addition, both AdaBoost and Random Forest have improved the original PFA algorithm. We notice that compared to AdaBoost, Random Forest achieves better results when the data is small, and in case of large data sets, it is recommended to use a boosting algorithm. Accordingly, we can basically consider that the XGBoost classifier offers the best performances.

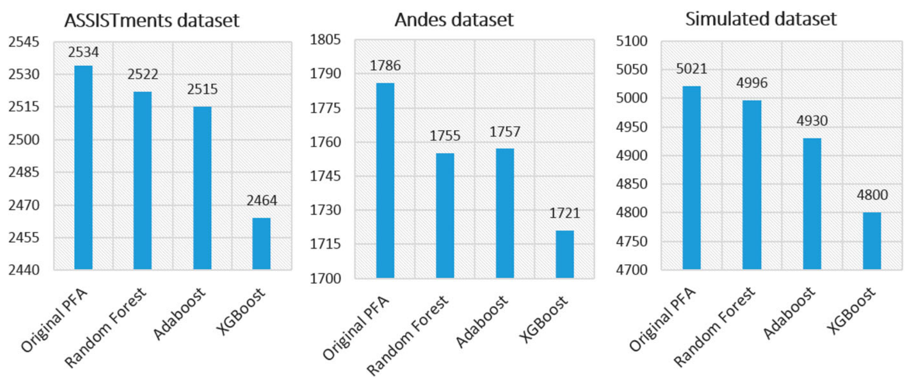

Figure 4. Number of incorrectly classified instances for student answers evaluation.

Table 9. Evaluation metric for the four classification models.

Dataset	Models	Accuracy	Precision	Recall	F1-score
ASSISTments 2009–2010	Original PFA	71.40%	73.97%	78.15%	70.21%
	Random Forest	78.25%	74.04%	78.24%	71.07%
	AdaBoost	78.30%	74.03%	78.31%	72.19%
	XGBoost	**78.75%**	**75.12%**	**78.75%**	**73.48%**
Andes dataset	Original PFA	70.46%	71.92%	73.95%	66.82%
	Random Forest	74.41%	71.90%	74.41%	68.96%
	AdaBoost	74.38%	71.65%	74.37%	69.46%
	XGBoost	**74.96%**	**72.65%**	**74.90%**	**70.86%**
Simulated dataset	Original PFA	67.80%	71.99%	70.71%	64.97%
	Random Forest	70.86%	71.30%	70.86%	65.78%
	AdaBoost	71.25%	71.07%	71.25%	67.02%
	XGBoost	**72.00%**	**72.05%**	**72.00%**	**68.11%**

In addition, we have evaluated our method based on the Area under the receiver operating characteristics curve (AUC-ROC curve). It is considered an efficient method for testing the predictive accuracy of classifier systems. The curve is a two-dimensional graphical plot that illustrates the performance of the proposed binary classifier, as its discrimination threshold is changed over the range of the predictor variable. In figure 5, we display the ROC curve to compare the performance of classifiers across the entire range of class distribution and error costs. The higher scores determine higher accuracies, where a perfect result would be the point (0, 1) indicating 0% false positives and 100% true positives. In our approach, we have used Python sickit-learn library to generate a false-positive rate and true positive rate for different values of this rounding threshold.

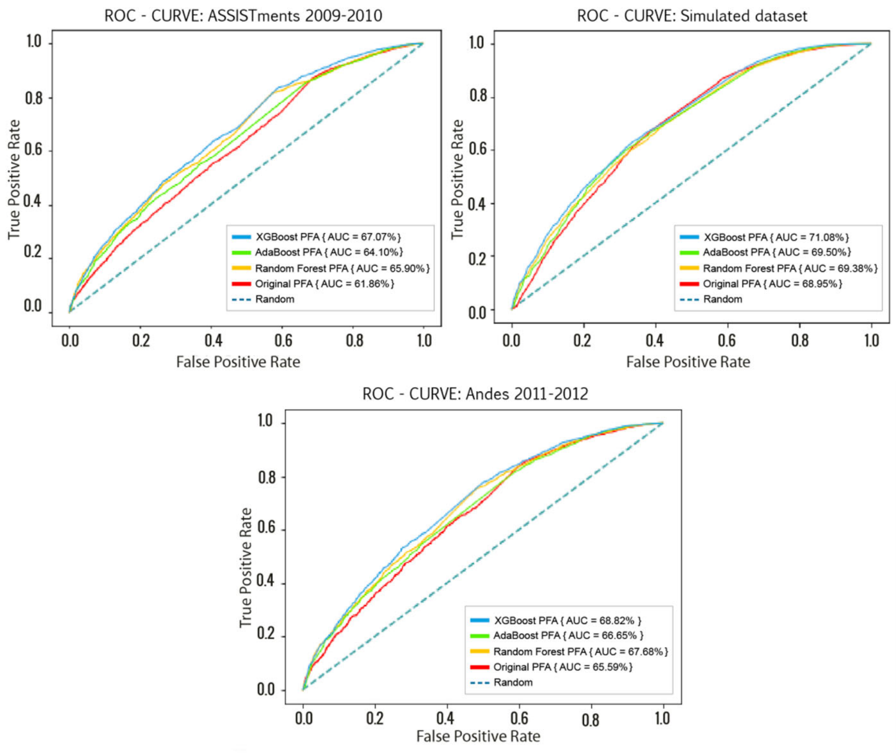

Figure 5. Receiver operation characteristics of the PFA based XGBoost approach.

According to the AUC metric, we can see from Figure 5 the variation that our models have represented. The prediction results from the three datasets provide an interesting demonstration of Gradient Boosting capacities. As illustrated by the ROC curve, the XGBoost clearly dominates the PFA Logistic Regression and all the other implemented algorithms, which means that PFA based XGBoost has substantially outperformed the basic model. From the results, as the ASSISTments 2009–2010 dataset mentions, the XGBoost model is about 5.2% more accurate than the original PFA model. Similarly, in the Andes dataset, XGBoost outperformed with the highest AUC score on the test set arriving up to 3.23% compared to the original PFA algorithm. The simulated dataset using the XGBoost model leads to an AUC of 71.08% which produces a 2.13% gain over the basic approach. As can be seen from the figure, other created models have also improved the original PFA model. We noticed that in the two real data sets, Random Forest has achieved good results compared with the AdaBoost classifier, while in the case of using a large data set, the improvement obtained by Random Forest was lower compared with the AdaBoost classifier.

Our model performances are also assessed according to the error cost. We have used for this purpose the logarithmic loss function as a statistical measure commonly used in machine learning for evaluating the quality of classification models. The Log loss metric mainly aims to penalize the false classifications. The goal of our proposed model is to reduce the prediction error value as much as possible. In general, a log loss closer to zero implies models with greater predictive quality where a low Log Loss value means a low uncertainty/entropy of the models. For the three used datasets, the prediction results in terms of this metric are presented in Figure 6.

From Figure 6, we notice that, for all the evaluated datasets, the use of the Gradient Boosting algorithm has marked a great potentiality to minimize the error rate compared to other tested ensemble learning techniques. The experimental results have shown that the PFA based XGBoost model, applied to the ASSISTments 2009 – 2010 dataset, has reduced the error rate by 0.21% on the test set compared to the original version of PFA model. Similarly, the figure illustrates that this model has the capability to decrease the error rate better compared to the original PFA model in the two other datasets. We notice that the error is reduced by 0.35% in the Andes dataset and by 0.46% in the simulated dataset. From the results, we notice also that models created based on Random Forest and AdaBoost classifiers have improved the PFA algorithm somewhat better than using single classifiers. In fact, the results obtained by the XGBoost model are very predictable, as the "Gradient Boosting Machine" method was mainly designed to reduce the error rate.

From all the results obtained, we can summarize that the three ensemble learning classifiers used have shown significant improvements compared to the single PFA classifier, where the XGBoost classifier has obtained the highest prediction performance due to its scalability. In the same context, several experiment results have demonstrated that most of XGBoost approach gives state-of-the-

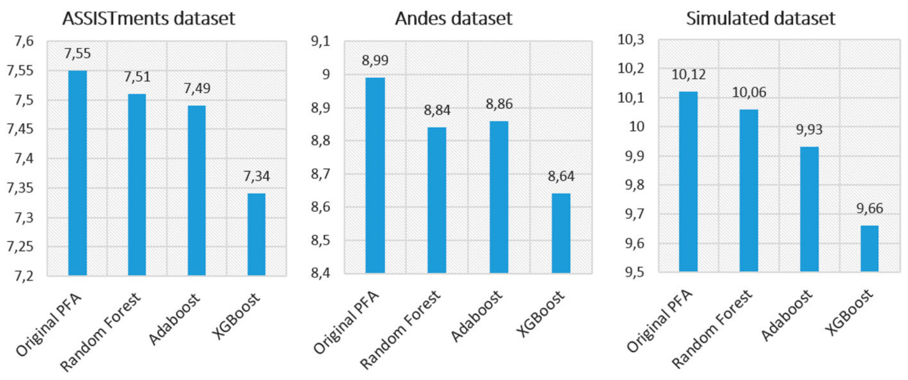

Figure 6. The error cost results of the four used classifiers.

Table 10. Training and testing time-consuming for the four classifiers.

Models	ASSISTments dataset		Andes dataset		Simulated dataset	
	Training	Testing	Training	Testing	Training	Testing
Original PFA	15.03	0.04	9.84	0.02	14.08	0.04
Random Forest	1017.46	19.39	550.93	11.30	1912.59	35.41
AdaBoost	114.6	1.74	106.79	1.57	134.21	4.55
XGBoost	129.8	0.32	74.05	0.17	110.2	0.26

art results on a wide range of problems (Maeta et al., 2018) and (Zhang et al., 2018). The impact of the approach has been widely recognized in a number of machine learning and data mining challenges. It is also important to note that these proposed classifiers might provide more significant results if implemented with a more efficient environment. In addition to the implementation process, by using a high-quality machine, it could also be possible to tune the XGBoost hyper-parameters model with a wide range of values, which may provide a more performing model.

Discussions

From the results of the experiments described in subsection 4, we found that the XGBoost model has significantly achieved higher predictive accuracy than the original PFA model, and has better outperformed the other evaluated algorithms. Therefore, we will pay special attention to it for further analysis. In fact, in order to obtain more robust results while using the XGBoost model, it is necessary to take into account certain requirements. In terms of the speed, as shown in Table 10, XGBoost has trained and tested the model at a relatively slow speed compared to the PFA logistic regression algorithm. In contrast, the XGBoost classifier remains less time consuming compared to other algorithms. In this analysis, we describe the testing execution time as a relevant feature for making the comparison. Apparently, We noticed that concerning the three used dataset, and compared with the original PFA approach, the manipulation of the XGBoost and AdaBoost algorithms lead to fairly high training and testing execution time, while the random Forest has taken a very long time, both in training and testing the models.

In addition to the execution time of the training and testing model, generally using the *GridSearchCV* technique for tuning parameters with all possible values may become a time-consuming challenge, especially with large parameter spaces. The cost of this technique grows exponentially with the number of parameters. In our research, we have used *BayesSearchCV* to deal with this problem, however, this method is not able to evaluate the model with all possible values, so that it is recommended to be used by trying repeatedly to define new values and then studying the changing behaviour of the model from time to time. Moreover, getting better accuracy at the moment of manipulating the gradient Boosting method requires a large number of samples for training the model. Where the XGBoost method requires a dataset with at least 10000 data points for providing better results. While it is very likely that most of these constraints' results could be improved when using a high-quality machine for implementing the boosting algorithm.

Conclusion

Many researchers still have to use machine learning rather than deep learning algorithms while conducting knowledge-tracing experiments. For this purpose, the aim of conducting this research is to focus on the importance of exploiting technical improvements along with the pedagogical contributions that could be proposed. Thus, researchers could obtain higher predictive accuracy, if the use of one of the machine learning algorithms is indispensable. In this work, we have applied some of the most powerful Ensemble Learning methods, namely Random Forest, AdaBoost, and XGBoost, to solve real-world classification problems in the educational system. The contribution aims to evaluate whether a single classifier can lead to sufficient optimization, and to specify to what extent the use of Ensemble Learning classifiers could improve the single KT classifier. The evaluation of our proposed models is done through a comparison against the original PFA model, as a more powerful Knowledge Tracing approach. For all datasets used, the results obtained have shown that Ensemble learning PFA versions are more valuable compared to the original PFA. Moreover, the results have demonstrated that XGBoost has the ability to predict student future acquisition with the highest performance. While, regardless of the remarkable improvement that can be provided by all ensemble learning models, it would be mentioning that the use of these classifiers requires certain environmental conditions in order to provide better results.

The work presented in this paper can be extended in many directions and open several opportunities for future development. As the best classifier model, XGBoost could be reinforced by taking into consideration the conditions discussed in subsection 5 to further enhance the prediction of future performance and obtain more significant results. Further analysis can also be performed to improve the mentioned model, we cite the use of boosting methods to analyse features' importance and avoid the computation complexity by selecting the most significant features.

Notes

1. eXtreme Gradient Boosting: https://github.com/dmlc/xgboost
2. ASSISTments Dataset: https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010
3. The Andes Physics Tutor Dataset: https://pslcdatashop.web.cmu.edu/Project?id=167

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Amal Asselman http://orcid.org/0000-0003-3960-2407
Mohamed Khaldi http://orcid.org/0000-0002-1593-1073
Souhaib Aammou http://orcid.org/0000-0002-4000-2073

References

Asselman, A., Khaldi, M., & Aammou, S. (2020). Evaluating the impact of prior required scaffolding items on the improvement of student performance prediction. *Education and Information Technologies*, 25, 3227–3249. https://doi.org/10.1007/s10639-019-10077-3

Asselman, A., Nasseh, A., & Aammou, S. (2018). Revealing strengths, weaknesses and prospects of Intelligent collaborative e-learning systems. *Advances in Science, Technology and Engineering Systems Journal*, *3*(3), 67–79. https://doi.org/10.25046/aj030310

Baker, R. S., Corbett, A. T., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In Beverley P. Woolf, Esma Aïmeur, Roger Nkambou, Susanne Lajoie (Eds.), *International conference on intelligent tutoring systems* (pp. 406–415). Springer, Berlin, Heidelberg.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Brusilovsky, P., & Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. Dans. In Peter Brusilovsky, Alfred Kobsa, Wolfgang Nejdl (Eds.), *The adaptive web* (pp. 3–53). Springer, Berlin, Heidelberg.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In New York, NY ACM 2016 (Ed.), *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). Association for Computing Machinery.

Chounta, I.-A., & Carvalho, P. F. (2019). Square it up! How to model step duration when predicting student performance. In *Proceedings of the 9th International Conference on learning analytics & knowledge* (pp. 330–334). Association for Computing Machinery.

Chrysafiadi, K., & Virvou, M. (2015). Student modeling for personalized education: A review of the literature. Dans. In *Advances in personalized Web-Based education* (pp. 1–24). Springer.

Chrysafiadi, K., Virvou, M., & Sakkopoulos, E. (2020). Optimizing programming language learning through student modeling in an adaptive web-based educational environment. In Maria Virvou, Efthimios Alepis, George A. Tsihrintzis, Lakhmi C. Jain (Ed.), *Dans machine learning paradigms* (pp. 205–223). Springer, Cham.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modelling and User-Adapted Interaction*, *4*(4), 253–278. https://doi.org/10.1007/BF01099821

De Vrieze, P. T., Van Bommel, P., Van Der Weide, T., & Klok, J. (2005). Adaptation in multimedia systems. *Multimedia Tools and Applications*, *25*(3), 333–343. https://doi.org/10.1007/s11042-005-6538-3

Feng, M., Heffernan, N., & Koedinger, K. (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, *19*(3), 243–266. https://doi.org/10.1007/s11257-009-9063-7

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232. doi:10.1214/aos/1013203451

Gong, Y., Beck, J. E., & Heffernan, N. T. (2011). How to construct more accurate student models: Comparing and optimizing knowledge tracing and performance factor analysis. *International Journal of Artificial Intelligence in Education*, *21*, 27–46.

González-Brenes, J., Huang, Y., & Brusilovsky, P. (2014). General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *The 7th International Conference on Educational Data mining* (pp. 84–91).

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Sage.

Hussain, M., Zhu, W., Zhang, W., Abidi, S. M., & Ali, S. (2019). Using machine learning to predict student difficulties from learning session data. *Artificial Intelligence Review*, *52*(1), 381–407. https://doi.org/10.1007/s10462-018-9620-8

Khajah, M. M., Huang, Y., González-Brenes, J. P., Mozer, M. C., & Brusilovsky, P. (2014). Integrating knowledge tracing and item response theory: A tale of two frameworks. *CEUR Workshop Proceedings*, *1181*, 7–15.

Koch, N. P. (2001). *Software Engineering for adaptive hypermedia systems-reference model*. Modeling Techniques and Development Process.

Koedinger, K. R., Baker, R. S., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010). A data repository for the EDM community: The PSLC DataShop. *Handbook of Educational Data Mining*, *43*, 43–56.

Maeta, K., Nishiyama, Y., Fujibayashi, K., Gunji, T., Sasabe, N., Iijima, K., & Naito, T. (2018). Prediction of glucose metabolism disorder risk using a machine learning algorithm: Pilot study. *JMIR Diabetes*, *3*(4), e10212. https://doi.org/10.2196/10212

Millán, E., Loboda, T., & Pérez-De-La-Cruz, J. L. (2010). Bayesian networks for student model engineering. *Computers & Education*, *55*(4), 1663–1683. https://doi.org/10.1016/j.compedu.2010.07.010

Minn, S., Yu, Y., Desmarais, M. C., Zhu, F., & Vie, J.-J. (2018). Deep knowledge tracing and dynamic student classification for knowledge tracing. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 1182–1187). IEEE Computer Society.

Nielsen, D. (2016). *Tree Boosting With XGBoost-Why Does XGBoost Win" Every" Machine Learning Competition?* Master's thesis, NTNU.

Papoušek, J., Pelánek, R., Řihák, J., & Stanislav, V. (2015). An analysis of response times in adaptive practice of geography facts. In *Proceedings of the 8th International Conference on Educational Data mining* (pp. 562–563).

Pardos, Z. A., & Heffernan, N. T. (2010). Modeling individualization in a Bayesian networks implementation of knowledge tracing. In Paul De Bra, Alfred Kobsa, David Chin (Ed.), *International conference on user modeling, adaptation, and personalization* (pp. 255–266). Springer, Berlin, Heidelberg.

Pardos, Z. A., & Heffernan, N. T. (2011). KT-IDEM: Introducing item difficulty to the knowledge tracing model. In Joseph A. Konstan, Ricardo Conejo, José L. Marzo, Nuria Oliver (Ed.), *International conference on user modeling, adaptation, and personalization* (pp. 243–254). Springer, Berlin, Heidelberg .

Pardos, Z. A., & Yudelson, M. V. (2013). Towards moment of learning accuracy. *AIED 2013 Workshops Proceedings*, *4*, 3.

Pavlik Jr, P., Cen, H., & Koedinger, K. (2009). Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. (ERIC, Éd.) *Online Submission*.

Pavlik Jr, P. I., Cen, H., & Koedinger, K. R. (2009). Performance Factors Analysis–A New Alternative to Knowledge Tracing. *Online Submission*.

Pelánek, R. (2015). Modeling students' memory for application in adaptive educational systems. *International Educational Data Mining Society*.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, *27*(3-5), 313–350. https://doi.org/10.1007/s11257-017-9193-2

Pelánek, R. (2018). Exploring the utility of response times and wrong answers for adaptive learning. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, (p. 18).

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In C. Cortes and N. Lawrence and D. Lee and M. Sugiyama and R. Garnett (Ed.), *Advances in neural information processing systems* (pp. 505–513).

Rihák, J. (2015). Use of time Information in models behind adaptive system for building fluency in mathematics. *International Educational Data Mining Society*.

Shahiri, A. M., & Husain, W. (2015). A review on predicting student's performance using data mining techniques. *Procedia Computer Science*, *72*, 414–422. https://doi.org/10.1016/j.procs.2015.12.157

Wang, Y., & Heffernan, N. (2013). Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes. In *International conference on artificial intelligence in education* (pp. 181–188). Springer.

Wang, Y., & Heffernan, N. T. (2011). The "Assistance" model: Leveraging how many hints and attempts a student needs. *Twenty-Fourth International FLAIRS Conference*.

Weibelzahl, S., Paramythis, A., & Masthoff, J. (2020). Evaluation of adaptive systems. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and personalization* (pp. 394–395). New York, NY: Association for Computing Machinery.

Woźniak, M., Grana, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3–17. https://doi.org/10.1016/j.inffus.2013.04.006

Xiong, X., & Pardos, Z. A. (2011). An analysis of response time data for improving student performance prediction.

Xiong, X., Zhao, S., Van Inwegen, E. G., & Beck, J. E. (2016). Going deeper with deep knowledge tracing. *International Educational Data Mining Society*.

Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., & Si, Y. (2018). A data-driven design for fault detection of wind turbines using random forests and XGboost. *IEEE Access*, *6*, 21020–21031. https://doi.org/10.1109/ACCESS.2018.2818678

Zhou, Z.-H. (2009). Ensemble learning. *Encyclopedia of Biometrics*, *1*, 270–273. https://doi.org/10.1007/978-0-387-73003-5_293

Zhou, Z.-H. (2012). *Ensemble methods: Foundations and algorithms*. CRC press.