


Article

Implementing AutoML in Educational Data Mining for Prediction Tasks

Maria Tsiakmaki, Georgios Kostopoulos, Sotiris Kotsiantis *  and Omiros Ragos

Department of Mathematics, University of Patras, 26504 Rio Patras, Greece; m.tsiakmaki@gmail.com (M.T.); kostg@sch.gr (G.K.); ragos@math.upatras.gr (O.R.)

* Correspondence: sotos@math.upatras.gr

Received: 29 November 2019; Accepted: 17 December 2019; Published: 20 December 2019



Abstract: Educational Data Mining (EDM) has emerged over the last two decades, concerning with the development and implementation of data mining methods in order to facilitate the analysis of vast amounts of data originating from a wide variety of educational contexts. Predicting students' progression and learning outcomes, such as dropout, performance and course grades, is regarded among the most important tasks of the EDM field. Therefore, applying appropriate machine learning algorithms for building accurate predictive models is of outmost importance for both educators and data scientists. Considering the high-dimensional input space and the complexity of machine learning algorithms, the process of building accurate and robust learning models requires advanced data science skills, while is time-consuming and error-prone in most cases. In addition, choosing the proper method for a given problem formulation and configuring the optimal parameters' values for a specific model is a demanding task, whilst it is often very difficult to understand and explain the produced results. In this context, the main purpose of the present study is to examine the potential use of advanced machine learning strategies on educational settings from the perspective of hyperparameter optimization. More specifically, we investigate the effectiveness of automated Machine Learning (autoML) for the task of predicting students' learning outcomes based on their participation in online learning platforms. At the same time, we limit the search space to tree-based and rule-based models in order to achieving transparent and interpretable results. To this end, a plethora of experiments were carried out, revealing that autoML tools achieve consistently superior results. Hopefully our work will help nonexpert users (e.g., educators and instructors) in the field of EDM to conduct experiments with appropriate automated parameter configurations, thus achieving highly accurate and comprehensible results.

Keywords: automatic machine learning; educational data mining; Bayesian optimization; early performance prediction

1. Introduction

Educational Data Mining (EDM) is the research field of using data mining methods and tools in educational settings [1,2]. Its main objective is to analyze these environments in order to find appropriate solutions to educational research issues [3], all of which are directed to improve teaching and learning [4]. Their results help students improve their learning performance, provide personalized recommendations, enhance the teaching performance, evaluate learning effectiveness, organize institutional resources and educational offer and many more [1,5].

Three common concerns that employ EDM techniques are the detection of whether a student is going to pass or fail a certain course, the prediction of students' final marks and the identification of students that are likely to drop out. The ability to predict students' performance and their underlying learning difficulties is a significant task and leads to benefits for both students and educational

institutions [6]. With a view to encouraging those students, remedial actions could be organized, such as early alerts and advising interventions [7]. In addition, the critical analysis of the reasons for failure and dropouts assist educators to improve their pedagogical basis and teaching approach [8]. Moreover, when used effectively, they can help institutions enhance learning experiences, develop appropriate strategies and ultimately, reduce dropout rates [9]. As such, we consider that effective tools for predicting student failures can be beneficial, especially at an early stage.

Related studies encompass a large collection of data mining tasks for studying different aspects of these problems. They include algorithms for attribute selection, association rule learning, classification and regression. Most of these methods depend on a wide range of hyperparameter choices with varying degrees of complexity and dimensions, which make them difficult for non-machine-learning experts to apply and even more to reason about. Meanwhile, the rapid development and the wide distribution of machine learning applications also point out the necessity of advanced data science skills in the relevant fields. Therefore, the need for machine learning methods that automate several of these design choices has been identified. The research area that promises to reduce the human input and effort on these processes is called automated Machine Learning or autoML. In experiments with several datasets, results from sophisticated automated optimization approaches compared favorably with results published from human experts inputs and the state of the art [10–13]. AutoML allows non-experts, such as educators and instructors, to conduct experiments and produce complex and effective learning models that ultimately support educational institutions.

At the same time, the need to provide transparent and explainable ML systems is also a factor to consider [14]. Such an effort aims to help scientists and nonexperts to understand and analyze ML models, their operational mechanism and the choices made towards their decision-making processes. Generally, it can be argued that methods that provide more linguistic models, such as rule and tree-based classifiers, are considered to be more comprehensible [15]. On the other hand, methods that are easy to understand and interpret tend to have lower predictive accuracy [16]. However, practical experience has demonstrated that, for some cases, the explainability feature is more important than the predictive accuracy [14,15]. AutoML could tackle the performance-transparency trade-off, providing tuned hyperparameters that improve the overall performance of the selected methods.

In the above context, our present work investigates the use of autoML in order to automate any part of the process of building machine learning models for the above mentioned three tasks: predict students who are prone to pass or fail, student's final grade and students at risk of dropout. We present a comparative study on classic educational data mining techniques and autoML. To equally highlight the importance of producing interpretable and explainable machine learning models [14,16], we restrict the configuration space by allowing only tree-based and rule-based classifiers (the choice of the classifier itself is considered as a hyperparameter). We prove that in most cases, the application of autoML on educational data improves model performance over traditional approaches. To the best of our knowledge, there is no research that demonstrates the use of advanced machine learning strategies to educational settings.

This paper is organized as follows. Section 2 is devoted to related work. We present a collection of studies that investigate the student's performance employing data mining techniques. Section 3 briefly introduces the Bayesian optimization search strategy that allows the efficient hyperparameter configuration. Section 4 describes our proposed approach. Section 5 illustrates our results. In Section 6 we discuss our findings, while Section 7 concludes our research considering some thoughts for future work.

2. Related Work

Predicting students' learning behavior and outcomes is regarded among the most important tasks of the EDM field. The main interest is mainly focused on three forms of predictive problems [17]. Predicting student performance (i.e., whether a student will pass or fail a course) covers a very wide area of research in the EDM field [18]. In addition, a plethora of studies has been published with

the aim of predicting students who are prone to drop out from a course, a very important problem, which principally concerns distance learning [19]. Finally, another common problem on prediction is estimating students' grades in a specific test, an exam or a course [20].

2.1. Related Work on Predicting Student Academic Performance

Mueen et al. (2016) employed Naïve Bayes (NB), Neural Networks (NNs), and Decision Trees (DTS) classification algorithms to predict the performance of undergraduate students [21]. The dataset was collected from two courses, both supported by a Learning Management System (LMS). The information retrieved included the access to teaching material, the performance on course assignments and the participation in discussion fora. The experimental results revealed that NB classifier outperformed the other two methods.

Student performance prediction models were also constructed and compared in a similar study [22]. Demographic features, academic background and behavioral metrics for student and parent participation in the learning process were exploited for this purpose. Apart from Artificial Neural Network (ANN), NB and DTS algorithms, the authors also compared several ensemble methods such as Bagging, Boosting and Random Forest (RF). The ANN model outperformed the other data mining techniques, while Boosting was the best ensemble method.

The prediction of whether a student should be considered as qualified or not was the research objective of Kaur et al. (2015) [23]. The authors experimented with four attribute evaluation methods and five classification algorithms, using a sample dataset of 152 regular high school students. The Multilayer Perceptron (MLP) was the best performing classifier among all other methods.

Guo et al. (2015) developed a prewarning system for students at risk of failing on the basis of a deep learning multiclass classification model [24]. The deep neural system developed in this work was a six-layer, fully connected feed-forward neural network with Rectified Linear Units (ReLU). The output layer was composed of five neurons with Softmax as a classifier. Each node on the output layer represented the students' final score {O, A, B, C, D}. The results showed that the proposed architecture acquired the highest accuracy values compared to three familiar classification algorithms: NB, MLP and Support Vector Machine (SVM).

The objective of Saa's study (2016) was to discover relations between students' personal and social factors, as well as their educational performance using data mining tasks [25]. The authors collected 270 records through online questionnaires and tested four classification algorithms, while the values of the class attribute were: excellent, very good, good, and pass. The best performing method was the Classification and Regression Tree (CART) classifier, scoring an accuracy measure of 40%.

In another comparative study regarding the effectiveness of EDM techniques [26], the authors compared four methods to predict students that are likely to fail in introductory programming courses at early stages. In addition, they demonstrated the importance of data preprocessing and algorithms fine-tuning tasks in the effectiveness of these techniques. Finally, the fine-tuned SVM was the best performer.

Predicting students' graduation performance at the end of a degree program was one of the three research goals examined by Asif et al. (2017) [27]. The data used comprised students' marks for all the courses in the four years of the program and variables related to students' pre-admission marks. The graduation mark (the class attribute) was divided into five possible values: A, B, C, D, E. The authors used several classification algorithms (DTS, k-Nearest Neighbor (kNN), NB, NNs and RF), while the maximum accuracy score (83.65%) was obtained by NB.

Finally, a recent study indicated the effectiveness of semisupervised learning (SSL) methods in students' performance prediction [28]. The authors evaluated the performance of SSL algorithms, namely Self-Training, Democratic, De-Tri-Training, Tri-Training, Co-Training, Random Subspace CO-training (RASCO) and Rel-RASCO. The best overall averaged accuracy score was obtained by Tri-Training algorithm with a C4.5 Decision Tree as the base classifier. Moreover, the Tri-Training algorithm performed better than the C4.5 Decision Tree supervised algorithm trained on the full dataset.

2.2. Related Work on Predicting Student Grade

Personalized multiple regression-based methods and matrix factorization approaches based on recommender systems were used by Elbadrawy et al. (2016) to forecast students' grades in future courses and in-class assessments [29]. Briefly, the first method was the course-specific regression, which predicted the grade that a student will achieve in a specific course as a sparse linear combination of the grades that the student obtained in past courses. The second method was the personalized linear multiregression, which employed a linear combination of k regression models, weighted on a per-student basis. The third method was a standard matrix factorization approach that approximated the observed entries of the student-course grade matrix. The fourth method was matrix factorization based on factorization machines. The evaluations showed that the factorization machines produced lower error rates for the next-term grade prediction.

Predicting student performance was also the main focus of the study conducted by Xu et al. (2017) [30]. Their goal was to predict the final cumulative Grade Point Average (GPA) of a student, given his/her background and performance states of the known grades and the predictions for the courses that have not been taken. For enabling such progressive predictions, the authors proposed a two-layer architecture. The first layer implements the base predictors for each course, given the performance state of graduate students on courses relevant to the targeted course. For discovering the relevant courses, a course clustering method was developed. In the second layer, ensemble-based predictors were developed, able to keep improving themselves by accumulating new student data over time. The authors' architecture was compared with four classic machine learning algorithms, named Linear Regression (LR), Logistic Regression (LogR), RF and kNN. The proposed method yielded the best prediction performance.

Predicting students' final grade was one of the two research goals also by Strecht et al. (2015) [31]. The authors evaluated various popular regression algorithms, i.e., Ordinary Least Squares, SVM, CART, kNN, RF and AdaBoost R2. The experiments were carried out using administrative data from the university's Student Information System (SIS) of Porto, concerning approximately 700 courses. The algorithms with best results overall were SVM, RF and AdaBoost R2.

The proposed method by Meier et al. (2015) made personalized and timely predictions of the grade of each student in a class [32]. Using data obtained from a pilot course, the authors' methodology suggested that it was effective to perform early in-class assessments such as quizzes, which result in timely performance prediction for each student. The study compared their proposed algorithm against four different prediction methods: two simple benchmarks; Single performance assessment and past assessments and weights and two well-known data mining algorithms; Linear Regression (LR) employing the Ordinary Least Squares (OLS) method and kNN algorithm ($k = 7$). The error of the proposed method decreased approximately linearly as more homework and in-class exam results were added to the model.

Sweeney et al. (2016) also presented the problem of student performance prediction as a regression task [33]. They explored three classes of methods for predicting the next-term grade of students. These were (1) simple baselines, (2) Matrix Factorization (MF)-based methods, and (3) common regression models. For the third category of methods, four different regression models were tested: RF, Stochastic Gradient Descent (SGD), kNN and personalized LR. The obtained results revealed that a hybrid of the RF model and the MF-based Factorization Machine (FM) was the best performer.

The first study that applied Semi-Supervised Regression (SSR) methods for regression tasks in educational settings was carried out by Kostopoulos et al. (2019) [34]. In order to predict final grades in a distance learning course, the authors proposed a Multi-Scheme Semi-Supervised Regression Approach (MSSRA) employing RF and a set of three k-NN algorithms as the base regressors. A plethora of features related to students' characteristics, academic performance and interactions within the learning platform throughout the academic year formed the training set. The results indicated that the proposed algorithm outperformed typical classical regression methods.

Finally, a recent study utilized eight familiar supervised learning algorithms – LR, Random Forests (RF), Sequential Minimal Optimization algorithm for regression problems (SMOreg), 5-NN, M5 Rules, M5, Gaussian processes (GP), Bagging – for predicting students' marks [35]. The training data contained selected demographic variables, students' first semester grades along with the number of examination attempts per course. The reported results seem rather satisfactory, ranging from 1.217 to 1.943. It was observed that RF, Bagging and SMOreg took precedence over the other methods.

2.3. Related Work on Predicting Student Dropout

One interested study used data gathered from 419 high schools students in Mexico [36]. The authors carried out experiments to predict dropout at different steps of the course, to select the best indicators of dropout. Results showed that their classifier (named ICRM2) could predict student dropout within the first 4–6 weeks of the course.

Student retention and the identification of potential problems as early as possible was the main aim of Zhang et al. (2010) [37]. The authors used data from the Thames Valley University systems that were related to the background and the academic activities of the students. Three algorithms, namely NB, SVM and DTS, were chosen and different configurations for each algorithm were tested in order to find the optimum result. Finally, NB was reported to have achieved the highest prediction accuracy.

Moreover, Delen (2010) used five years of institutional data along with several popular data mining techniques (four individual and three ensemble techniques), in order to build models to predict and explain the reasons behind students dropping out [38]. The data contained variables related to students' academic, financial, and demographic characteristics. The SVM produced the best results when compared to ANN, DTS and LogR. The information fusion-type ensemble model produced the best results when compared with the Bagging and Boosting ensembles.

Lykourantzou et al. (2009) presented a dropout prediction method for e-learning courses, based on three machine learning techniques: NNs, SVM and the probabilistic ensemble simplified fuzzy ARTMAP [39]. The results of these techniques were combined using three decision schemes. The dataset consisted of demographic attributes, prior academic performances, time-varying characteristics depicting the students' progress during the courses, as well as their level of engagement with the e-learning procedure. The decision scheme where a student was considered to be a dropout if at least one technique has classified this student as such, was reported to be the most appropriate solution for achieving and maintaining high accuracy, sensitivity and precision results in predicting at-risk students.

Superby et al. (2006) applied NNs, discriminant analysis, DTS and RF on survey data from three universities, to classify new students in low-risk, medium-risk, and high-risk categories [40]. The authors found that the scholastic history and socio-family background were the most significant predictors of students at risk. The least bad result of the four methods was reported by the NNs method, reaching the total rate of correctly classified students up to 57.35%.

Herzog's study (2006) examined the predictive accuracy of the DTS and NNs over the problem of predicting college freshmen retention [41]. The author used three sources to produce the data set: the institutional student information system for student demographic; the American College Test (ACT)'s Student Profile Section for parent income data; and the National Student Clearinghouse for identifying transfer-out students. Overall prediction results showed that the DTS and NNS performed substantially better to the LR baseline. Also, the different results from the three NNs variations confirmed the importance of exploring available setup options.

Finally, a recent study explored the usage of semi-supervised techniques for the task of drop out prediction [42]. The dataset consisted of 2 classes of 344 instances characterized by 12 attributes. The authors compared familiar SSL techniques, Self-Training, Co-Training, Democratic Co-Training, Tri-Training, RASCO and Rel-RASCO. In their two separate experiments, C4.5 and NB were the base classifiers, while C4.5 was the dominant supervised algorithm. The results revealed that Tri-Training (C4.5) algorithm outperformed the rest SSL algorithms as well as the supervised C4.5 Decision Tree.

To sum up, various researchers have investigated the problem of student’s performance prediction employing a plethora of data mining techniques. The results reveal that there is a strong relationship between students’ logged activities in LMSs and their academic achievements. Most of the proposed prediction models achieved notable results (accuracy is more than 80%). However, a variation in the outperformers is observed; i.e., there is no method that can be thought of or shown to be better than others for educational settings (Table 1). Even more, to the best of our knowledge, there is no research that demonstrates the use of advanced machine learning strategies to these settings, such as the autoML.

Table 1. Data mining algorithms as applied in educational settings.

Paper	Prediction Task	Metrics	Methods Compared	Outperformers
<i>Related work on predicting student academic performance</i>				
[21]	Binary Classification	Accuracy, Precision, Recall, Specificity	NB, NNs, DTs	NB
[22]	Binary Classification	Accuracy, Precision, Recall, F-measure	ANN, NB, DTs, RF, Bagging, Boosting	ANN and Boosting
[23]	Binary Classification	Accuracy, Precision, Recall, F-measure, ROC Area	NB, MLPs, SMO, J48, REPTree	MLP
[24]	Multiclass Classification	Accuracy	NB, MLPs, SVM	Deep Neural Network
[25]	Multiclass Classification	Accuracy, Precision, Recall	C4.5, NB, ID3, CART, CHAID	CART
[26]	Binary Classification	F-measure	NNs, J48, SVM, NB	SVM fine-tuned
[27]	Multiclass Classification	Accuracy, Kappa	DTs, NB, NNs, Rule Induction, 1-NN, RF	NB
[28]	Binary Classification	Accuracy, Specificity	De-Tri-Training, Self-Training, Democratic, Tri-Training, Co-Training, RASCO, Rel-RASCO, C4.5	Tri-Training (C4.5 as base learner)
<i>Related work on predicting student grade</i>				
[29]	Regression course grades Assessments Grades	RMSE, MAE	Regression-based methods, Matrix factorization-based methods	Factorization machine Depending on the records
[30]	Regression GPA	MAE	LR, LogR, RF, kNN	Proposed architecture
[31]	Regression course grades	RMSE	SVM, RF, AB.R2, kNN, OLS, CART	SVM
[32]	Regression course grades	Average absolute prediction error	Single performance assessment proposed architecture and Past Assessments, Weights, LR, 7-NN	Proposed architecture
[33]	Regression course grades	RMSE, MAE	Simple baselines, MF-based methods, regression models {RF, SGD, kNN, personalized LR}	Authors’ proposed hybrid model (FM-RF)
[34]	Regression course grades	MAE, RAE, RMSE, PCC	IBk, M5Rules, M5 Model Tree, LR, SMOreg, k-NN, RF, MSSRA	Proposed method (MSSRA)
[35]	Regression course grades	MAE	LR, RF, 5NN, M5 Rules, M5, SMOreg, GP, Bagging	RF, Bagging, SMOreg
<i>Related work on predicting student dropout</i>				
[36]	Classification (dropout)	Accuracy, TP rate, TN rate, GM	NB, SMO, IBk, JRip, J48, ICRM2	Proposed architecture (ICRM2)
[37]	Classification (dropout)	Accuracy	NB, SVM, DTs	NB

Table 1. Cont.

Paper	Prediction Task	Metrics	Methods Compared	Outperformers
<i>Related work on predicting student dropout</i>				
[38]	Classification (dropout)	Accuracy	ANN, SVM DTS, LR, Information fusion, Bagging, Boosting	Information fusion
[39]	Classification (dropout)	Accuracy, Sensitivity, Precision	3 decision schemes based on NN, SVM, and ARTMAP	Decision scheme 1
[40]	Classification (dropout)	Accuracy	NNs, DTs, RF, Discriminant Analysis	NNs
[41]	Classification (dropout)	Accuracy	NNs, DTs, LR	DTs
[42]	Classification (dropout)	Accuracy, Sensitivity	Self-Training, Co-Training, Democratic Co-Training, Tri-Training, RASCO, Rel-RASCO, C4.5	Tri-Training (C4.5)

3. Introduction to Bayesian Optimization for Hyperparameter Optimization

For the prediction of students' academic performance, we explore the use of autoML to automatically find the optimal learning model without human intervention. The task of constructing a learning model usually includes supplementary processes; the attributes selection, learning algorithm selection, and their hyperparameter optimization. Therefore, to model the problem of automatically tuning the machine learning pipeline for obtaining the optimal performance result (i.e., the goal of autoML), the overall hyperparameter configuration space covers the choice between various preprocessing and machine learning algorithms along with their relevant hyperparameters.

This optimization problem is currently addressed by various techniques. During this section, we will briefly discuss algorithms that are part of a powerful and popular approach, referred to as "black-box optimization" techniques. More specifically, our focus will be the Bayesian optimization algorithm, as this is the method that we employ for our experiments. We will also refer to prominent autoML software packages and to the importance of autoML, especially for the non-ML-expert users. At first, we provide some definitions related to both optimization and hyperparameter optimization problem.

3.1. Definitions

In general, optimization refers to the process of finding the best result under specified circumstances [43]. More formally, it is the (automatic) process of finding the value or a set of values of a function (a real valued function called the objective function) that maximizes (or minimizes) its result. It is consistent with the principle of maximum expected utility or minimum expected loss (risk) [44]. It is well known that there is no single method that can solve every optimization problem efficiently. It can be challenging to choose the best method for a given problem formulation, usually complex and computationally expensive processes are required. However, optimization methods can obtain high quality results with reasonable efforts.

In machine learning systems, the targets of automation include mechanisms that optimize machine learning pipelines, such as the feature engineering, model selection, hyperparameter selection, etc. The problem of identifying values for the hyperparameters that optimize the system's performance is called the problem of hyperparameter optimization [45]. Hyperparameters are the parameters whose values are set before the beginning of the learning process, e.g., the number of neighbor's k in the nearest neighbor algorithm, or the depth of the tree in tree-structured algorithms. In contrast, model parameters are parameters that are learned during the training process, e.g., the weights of neurons in

a neural network. The central focus of autoML is the hyperparameter optimization (HPO) of machine learning processes.

Formally, the general statement of the hyperparameter optimization problem is defined as [44]:

$$x^* = \operatorname{argmin}_{x \in X} f(x) \quad (1)$$

where f is the objective function, given a set of hyperparameters x from a hyperparameters space X . We are interested in finding x^* set that minimizes the expected loss (the value of $f(x)$). One important property of this function f is that its evaluation is expensive (costly) or even impossible to compute [46].

To make f more clear, consider the context of machine learning applications, where, for example, the function f can be a system that predicts student dropout rates (e.g., a set of preprocessing and classification algorithms) with adjustable parameters x (e.g., the learning algorithm or the depth of the tree when tree structured algorithms are tested), and an observable metric $y = f(x)$, on data collected from learning systems (e.g., data from a learning management system).

As manual tuning is an error-prone process that also requires time and experts in the field [47], various automatic configuration methods have been proposed. A popular approach is to treat the problem as black-box optimization. Grid search has been the traditional and the most basic, yet extremely costly method. A fairly efficient alternative is random search [48]. Other families of methods that have been applied are gradient-based algorithms [49], racing algorithms, e.g., [50], evolutionary optimization algorithms [51], and population-based search, e.g., [52]. Next, we will try to focus on another strategy employed to obtain the optimal set of hyperparameters, that of Bayesian optimization. Our research also leverages the advances of this method.

3.2. Bayesian Optimization

Bayesian optimization is an effective strategy for minimizing (or maximizing) objective functions that are costly to evaluate. The main advantage of this method is that it uses previous results in order to pick the next point to try, while dealing with the dilemma of exploration and exploitation. As such, it reaches the optimal solution with less number of evaluations [44,53].

Bayesian optimization uses the famous Bayes theorem. The theorem states that the posterior probability of a model M given data D $P(M|D)$ is proportional to the likelihood of D given M $P(D|M)$ multiplied by the prior probability $P(M)$. As for the hyperparameter optimization problem, model M should not be mistaken with the output model of machine learning algorithms. On the contrary, M is actually a regression model that represents our assumptions about f [54]:

$$P(f|D_{i:t}) \propto P(D_{i:t}|f)P(f) \quad (2)$$

where $D_{i:t} = \{x_{i:t}, f(x_{i:t})\}$ defines our accumulated observations of the objective function f on sequences of data samples $x_{i:t}$ [44]. The prior prescribes our belief (what we think we know) over a space of objective functions. The likelihood captures how likely the data we observed are, given our belief about the prior. These two combined, give us the posterior, which represents our belief about the objective function f . Additionally, the posterior conceptualizes the *surrogate* function, the function used to estimate f .

As it is much easier to optimize the surrogate probability model than the objective function, the Bayesian optimization selects the next set of hyperparameters to evaluate based on its performance on the surrogate. During the execution, the accuracy of the surrogate model is increased by continually incorporating the evaluations on the objective function.

The Bayesian optimization is considered as a sequential design strategy (formally Sequential model-based optimization (SMBO)) [10,55]. At first, a surrogate probabilistic regression model of the objective function is built. Until a budget limit is being reached, new samples of hyperparameters are sequentially selected. These new samples are selected by optimizing an acquisition function S , which

uses the surrogate model. Each suggested sample is applied on the true objective function and produce new evaluations. The new observations are used to update the surrogate model (Algorithm 1). There are several variants of the SMBO formalism, which are specified by the selection of the probabilistic model and the criteria (acquisition function) used to select the next hyperparameters.

In the model-based optimization literature, the most recognized choices for the surrogate model are the Gaussian Processes [56], Tree Parzen Estimators (TPE) [10,57], and Random Forests [58]. In this research, we selected Random Forests for our experiments. Some of the advantages of this method is that it can represent the uncertainty of a given prediction, and that it can handle categorical and conditional parameters [59]. Hutter et al. [54] suggested that tree-based models (i.e., TPE and Random Forests) work best for large configuration spaces and complicated optimization problems. The same conclusions are marked by the authors of [60]. They further notice that, in their experiment, Random Forests could obtain results of four more configurations at that same time budget compared to TPE and GP. The reason is their support on small number of folds at the cross-validation procedure. Random Forests are used by the Sequential Model-based Algorithm Configuration (SMAC) library (<https://github.com/automl/SMAC3>).

Algorithm 1: The sequential model-based optimization algorithm

Input: $f, M_0, T, S, H := \emptyset$
Output: x^* from H with minimal c

- 1: Function SMBO
- 2: for $t := 1$ to T do
- 3: $x^* := \operatorname{argmin} S(x, M_{t-1});$
- 4: $c := \operatorname{evaluateCost}(f(x^*));$
- 5: $H := H \cup \{(x^*, c)\};$
- 6: $M_t := \operatorname{fitNewModel}(H);$
- 7: end for

The role of the acquisition function is to decide which point the surrogate model should evaluate next. It determines the utility of the candidate data points, trading off exploration and exploitation. Exploration favors new, uncertain areas in the objective space. Exploitation benefits areas that are already known to have advantageous results [11]. The Expected Improvement is considered to be among the typically used acquisition functions [61]. Other strategies have also been suggested, such as the upper confidence bound (UCB), the Probability of Improvement [62], and the more recent proposed Gaussian process upper confidence bounds (GP-UCB) [63].

Several open-source Bayesian optimization software packages exist for various stages of autoML. Some libraries that can be used to optimize hyperparameters are Spearmint (<https://github.com/JasperSnoek/spearmint>) [11], Metric Optimization Engine (MOE) [64], Hyperopt [13], Sequential Model-based Algorithm Configuration (SMAC) [59]. SMAC framework enabled the autoML frameworks: Auto-WEKA and Auto-sklearn.

3.3. Use Cases

In the context of machine learning, each time we try a different set of hyperparameters, we build a model using the training dataset and create the predictions based on the validation dataset and the evaluation metric. Considering the high dimensional search space and the complexity of models, such as deep neural networks or ensemble methods, the specific process is practically intractable to be done by hand, while in most cases it requires advanced data science skills. Therefore, autoML processes make machine learning more accessible, while reducing the human expertise that is required and, finally, improving the performance of the model [45]. Moreover, automatic tuning is reproducible and generally supports the fair comparison of the produced results [57], while in the case that the search

space is limited by design to specific learning algorithms, the produced results can be more transparent and interpretable to the end users, as our study shows.

4. Method

4.1. Research Goal

The goal of this study is to examine the potential yield of advanced machine learning strategies to improve the prediction of student's performance based on their participation in online learning platforms. Specifically, we investigated the effectiveness of the Automated Machine Learning (autoML) in conjunction with educational data to early predict students' final performance. We experimented using autoML for the tasks of algorithm selection, hyperparameter tuning, feature selection and preprocessing. Furthermore, to achieve explainable machine learning decisions, we made available only tree and rule-based classifiers in the configuration space for the task of selecting a learning algorithm. We examined log data obtained from several blended courses that were using the Moodle platform. We studied whether, and to what extent, the use of the autoML could leverage the performance of the predictions and provide reasonable results rather early when compared with standard ML algorithms. Our work will hopefully help nonexpert users, such as teachers, to conduct experiments with the most appropriate settings and hence achieve improved results.

4.2. Procedure

The selected compulsory courses "Physical Chemistry I", "Physics III (Electricity—Magnetism)" and "Analytical Chemistry Laboratory" were held in the spring semester of 2017–2018 at the Aristotle University of Thessaloniki (Table 2). In total, 591 students attended the courses, 322 of which were male, and 269 females (mean = 295.5, standard deviation = 26.5). The total number of students of the first course was 282 (122 females and 160 male), in the second 180 (90 females and 90 male), and in the third 129 (57 females and 72 male). The final grade was based on weighted averages of grades that students received at the online assignments and the final examination. The students attended the courses between February 2018 and July 2018.

Table 2. Summary statistics for each course.

Course	Female	Male	Total	Course Modules in Moodle
Physical Chemistry I	122	160	282	forum, resource, page, assign, folder
Physics III	90	90	180	forum, resource, page, assign
Analytical Chemistry Lab	57	72	129	forum, resource, page, assign, folder, URL
Total	269	322	591	

For all courses, the face-to-face teaching was supported with online resources and activities over the Moodle learning platform. All of the material of the courses was added into sections as web pages, files or URLs. The materials were available to the students until the end of the semester. Most sections also contained learning activities that were evaluated for a grade. Each Moodle course preserved the default Announcements forum. Announcements were created by making posts in that forum. It should be noted that the courses were not directed or specially designed for the conduction of the experiments that are described in the research.

4.3. Data Collection

For the collection of the datasets, we developed a custom plugin for Moodle (Figure 1). The implemented extension computes an outline report of the course's activities. For each student, the outline calculates the number of views for each available module (e.g., activity, resource, folders, forum), the grades of each activity (e.g., assign, workshop, choice, quiz), the number of created posts (if any), aggregated event counters, and her/his final grade. By default, the report is computed from

the course starting date to the current date, but results can also be filtered by a specified date. Data are exported in various formats and for regression and classification machine learning tasks. Most of the data are retrieved from the log's table in conjunction with aggregate functions.

gender	span-lang-en-class-multilang-announcements-forum-span-span-lang-en-class-multilang-forum-anakoinwsewnspan_forum_views	uli-mathimatos_page	kanonismos-mathimatos_page	didaskontes-epikoinwnia_page	luseis-g-seiras-askisewn-2018_resource	luseis-b-seiras-askisewn-2018_resource	luseis-a-seiras-askisewn-2018_resource	diafaneies-mathimatos-febrouarios-2018_resource
m	3	1	3	1	2	3	5	1
m	2	1	1	1	2	2	2	4
f	6	18	2	2	2	2	4	6
m	2	2	1	2	2	2	4	1
f	8	1	1	1	1	1	1	2

Figure 1. Screenshot of the custom report plugin on Moodle. The report outline lists all the available learning activities of the course (modules) along with a corresponding score for each participant (student). Scores are calculated according to the resource type (e.g., for the ‘page’ module we count the total number of student access). The report can be exported in two formats (arff, xls) and for various data mining tasks (classification, regression). Results can be calculated until a specified date.

For each machine learning experiment (dropout, pass/fail, regression) we collected six samples of the exported reports, one for each month of the semester. We aimed at experimenting in order to be aware of the precision of the results at the time and to predict failures as soon as possible during the semester. Table 3 lists the total number of logs that were available per course in the Moodle platform. In total, more than 130,000 log events were parsed. As it is expected, the number of events is increased during the semester (Figure 2). The students’ interest in the course varies, depending on the course activities. In general, course registrations start in February, final examinations take place in June, and final grades are announced by professors in July. An exception is the “Physical Chemistry I” course, by which the logged interactions had not started until March. As such, we will not build models for the first period.

Table 3. Number of activity logs per month for each course (separate and aggregated). e.g., 7617 records were created for Physical Chemistry I in the logs table during the third month (April), and 9692 records were created until the end of the third month (i.e., 2075 + 7617).

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Physical Chemistry I	0	2075	7617	13,175	9298	787
	0	2075	9692	22,867	32,165	32,952
Physics III	1156	8007	7259	10,369	7760	617
	1156	9163	16,422	26,791	34,551	35,168
Analytical Chemistry Lab	5800	28,564	8499	17,054	1623	416
	5800	34,364	42,863	59,917	61,540	61,956

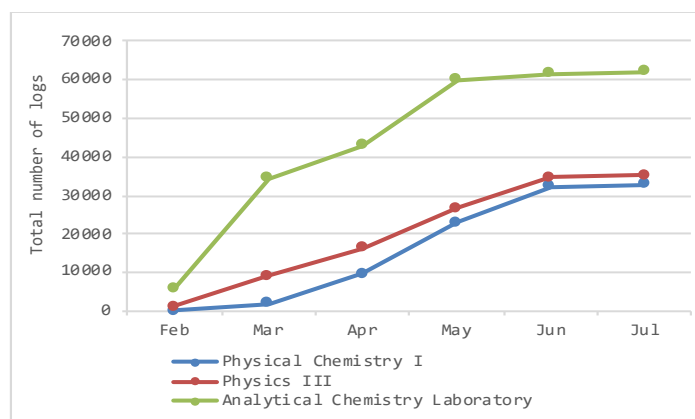


Figure 2. The total number of logs per month for each course. Logged activities are increased during the semester, as could be expected.

4.4. Data Analysis

Data related to 6 types of learning activities were collected and they are presented in Table 4. For each module, we calculated a numeric representation. Specifically, for the forum module, we counted the total number of times a student viewed the forum threads. Respectively, we counted the number of times a page, a resource, a folder, and a URL module had been accessed by each student. For the assign module, we counted the number of times a student accessed its description, and the grade that s/he took using the 0–10 scale. We also include a counter that aggregates the number of times a student viewed the course (course total views) and the total number of every kind of log written for a student for the specific course (course total activity). We were not able to examine additional demographic values apart from gender. The class attribute was the final grade that the student achieved, transformed to 0–10 scale, or nominal according to the supervised machine learning problem in question. Students that finally succeeded in a course are the ones that scored above or equal to 5. Dropout students were considered to be the ones whose final grade was an empty value. The dataset does not contain missing values. Students that did not access a learning activity score for 0. Learning resources that were not accessed by any student were not included in the experiments.

Table 4. Variables extracted from students’ Moodle use for each course.

Physical Chemistry I	Physics III	Analytical Chemistry Lab	Description	Possible Values
gender	gender	gender	Student’s gender	{female, male}
1 forum 7 pages 17 resources 2 folders 8 assign views	1 forum 6 pages 15 resources 9 assign views	1 forum 2 pages 4 resources 17 folders 1 URL 8 assign views	Total number of times a student accessed the resource	0 or positive integer
3 assigns	9 assigns	8 assigns	Student grade	[0,10]
course total views	course total views	course total views	The total number of course views for an individual student	0 or positive integer
course total activity	course total activity	course total activity	The total number of every kind of log written for an individual student	0 or positive integer
42 attributes	44 attributes	45 attributes		

The descriptive statistics are represented in Table 5. The mean (average) and standard deviation of the final grades in “Physical Chemistry I” were, respectively, 3.98 and 2.92, in “Physics III” 3.62 and 3.17, while in “Analytical Chemistry Laboratory” 6.27 and 2.73. The 48% of students passed the first course, 41% the second and 81% the third course. Only the 28% of students dropped out of the first course, 24% dropped out of the second and 6% dropped out of the third. It is concluded that Physical Chemistry I and Physics III were more challenging courses, while the high grades on average in the laboratory of Analytical Chemistry affirm that this course was easier.

Table 5. Descriptive statistics for each course.

	Regression				Classification		Classification	
	Min	Max	Mean	Std	Pass	Fail	Dropout	No Dropout
Physical Chemistry I	0	10	3.98	2.92	134	148	78	204
Physics III	0	10	3.62	3.17	74	106	44	136
Analytical Chemistry Lab	0	10	6.27	2.73	105	24	8	121

4.5. Feature Importance

During our research, additional procedures to understand more comprehensively the datasets were performed. More specifically, we used extremely randomized trees [65] to evaluate the importance of features on the classification and regression tasks (We used ExtraTreesClassifier and ExtraTreesRegressor ensemble methods from sklearn Python library, that return the feature importance (e.g., see <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>)). We therefore indicated the informative features for each course (Figure 3). Table 6 summarizes the results found by listing the module categories that were estimated above each average per course and per task.

Table 6. Features above the average importance per course and per task.

	Physical Chemistry I	Physics III	Analytical Chemistry Lab
Classification (Pass/Fail)	14 total views, 3 assign views, total activity, 4 resources, 3 assigns, 1 folder, forums	13 2 assign, 4 assign views, 4 resources, total views, total activity, gender	10 8 assigns, 1 resources, 1 assign views
Regression	9 total views, 2 assign views, total activity, 2 resources, 3 assigns	14 3 assign, 5 assign views, 3 resource, total views, total activity, gender	7 7 assigns
Classification (Dropout/No Dropout)	17 total views, 3 assign views, total activity, 5 resources, 3 assigns, 1 folder, 1 forum, 2 pages	10 1 assign, 5 assign views, 2 resources, total views, total activity	15 7 assign, 5 folder, 3 assign views, total views
Common Important Features	9 total views, total activity, 2 assign views, 2 resources, 2 assigns	8 total views, total activity, 1 resource, 4 assign views, 1 assign	6 6 assigns

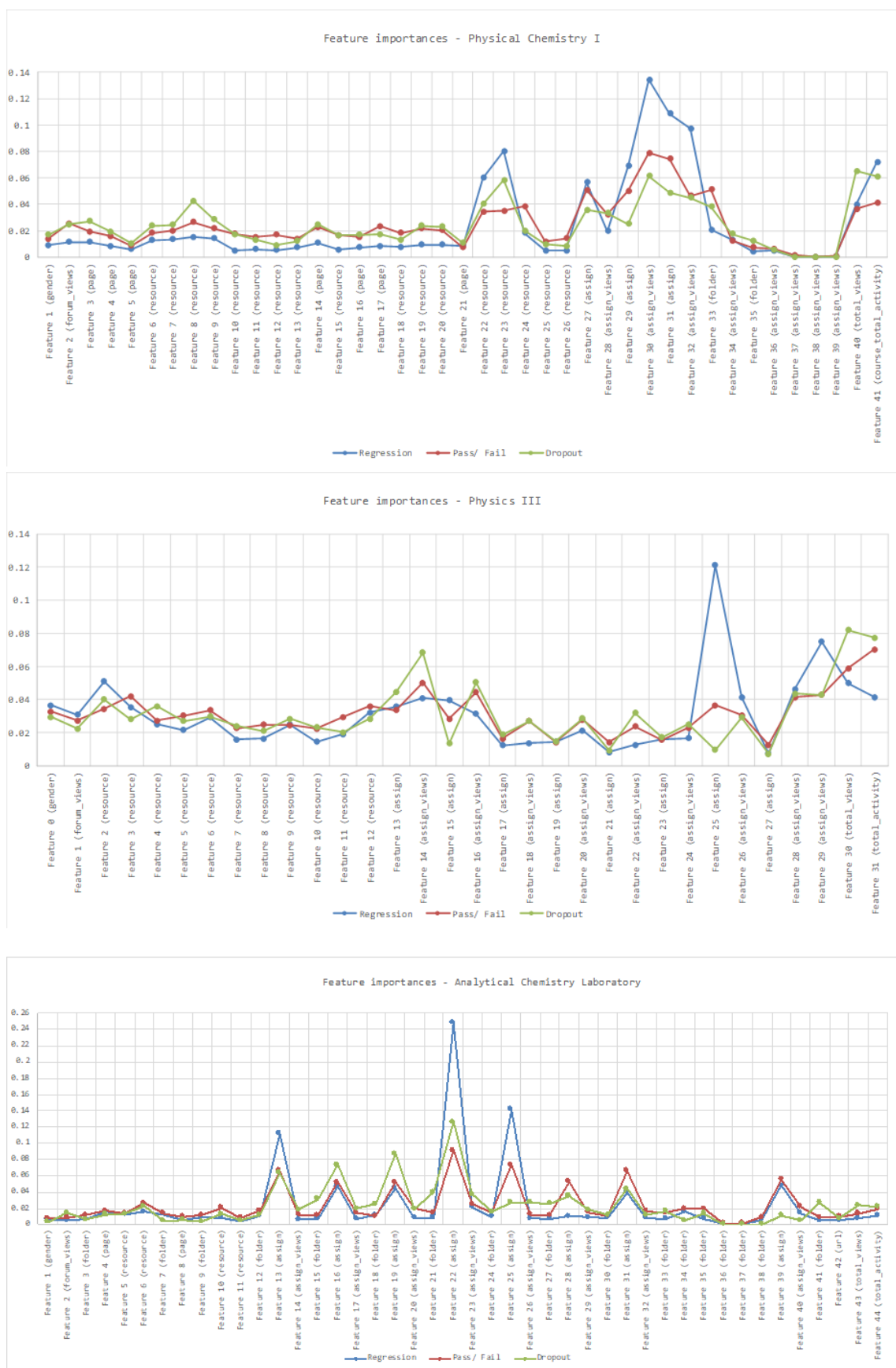


Figure 3. The plot suggests the features that are informative for each task.

However, such data approaches rarely can lead to conclusive results. As expected, features related to assignment grades were far more informative than views counters when considering regression tasks. A representative example is the laboratory course of Analytical Chemistry; as such type of courses concentrate on conducting assignments related to the theory of a related course. In-class exams were indicated as good predictors also by Meier et al. [32]. On the other hand, pass/fail classification tasks list among the effective and some features that are related to how many times a student accessed resources and pages. It is worth noting that among the high rated documents are the ones related to result announcements. Features 22 and 23 of the “Physical Chemistry I” course are related to resources were the results of the handwritten exams and the final grades were listed. Finally, dropout classification tasks list the widest variety of features among its important compared to the other tasks. Seventeen out of 42 features have estimated importance above the average of all the feature importance at the “Physical Chemistry I” course.

4.6. Evaluation Measures

For evaluating the performance of the classification models, we calculated the accuracy metric, which is defined in accordance to the confusion matrix (Table 7) as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \quad (3)$$

Table 7. Confusion matrix.

		Predicted Class	
		Fail/Dropout	Pass/No Dropout
Actual class	Fail/Dropout	TP	FN
	Pass/No Dropout	FP	TN

In addition, in order to evaluate the regression models, we make use of the Mean Absolute Error (MAE) measure, which is defined as follows:

$$\text{MAE} = \frac{\sum_{i=1}^N |\hat{G}_{ij} - G_{ij}|}{N} \quad (4)$$

Finally, the dropout classification accuracy was measured with the Receiver Operating Characteristic (ROC), since it is appropriate for datasets with imbalanced class distributions [66,67]. The ROC Curve is a two-dimensional graph that illustrates the performance trade-off of a given classification model [68].

Given such experimental set-up, it is necessary to use a statistical test to verify whether the improvement is statistically significant or not. We apply the paired, one tailed *t*-test to compare the maximum accuracy (or minimum error) obtained by a set of classic machine learning algorithms using their default parameter values (marked with a star *), with the results when using autoML. All *t*-tests have been performed with a significance level of $\alpha = 0.05$. As such, if the *p*-value is inferior or equal to 0.05, we conclude that the difference is significant.

4.7. Environment

To apply the data mining techniques, we used the WEKA implementation [69] without customizing the default parameter values. In addition, we employed the Auto-WEKA [70], the autoML implementation for WEKA that uses SMAC to determine the classifier with the best performance.

During the experiments, the classic algorithms were executed using the 10-fold cross-validation method. Therefore, each dataset was divided into 10 equally sized subsets (folds). The method was

repeated 10 times, and for each time 9 of the 10 subsets were used to form the training set, and the remaining 1 was used as the test/validation set.

4.8. ML Algorithms

For our study we used various classification and regression techniques for predicting the final student performance. We made sure to choose one representative example out of the six main categories of learners, i.e., Bayes classifiers, rule-based, tree-based, function-based, lazy and meta classifiers [71].

1. Bayes classifiers: Based on the Bayes theorem, the Bayes classifiers constitute a simple approach that often achieves impressive results. Naïve Bayes is a well-known probabilistic induction method [72].
2. Rule-based classifiers: In general, rule-based classifiers classify records by using a collection of “if . . . then . . . ” rules. PART uses separate-and-conquer. In each iteration the algorithm builds a partial C4.5 decision tree and makes the “best” leaf into a rule [73]. M5Rules is a rule learning algorithm for regression problems. It builds a model tree using M5 and makes the “best” leaf into a rule [74].
3. Tree-based classifiers: Tree-based classifiers is another approach to the problem of learning. An example is Random Forest classifier that generates a large number of random trees (i.e., forest) and uses the majority voting to classify a new instance [75].
4. Function-based classifiers: Function-based classifiers build a discriminant function that separates selected instances as widely as possible. SMO and SMOreg implement the support vector machine for classification [76–78] and regression [79,80] respectively.
5. Lazy classifiers: Lazy learners do not train a specific model. At the prediction time, they evaluate an unknown instance based on the most related instances stored as training data. IBk is the k-nearest neighbor’s classifier that is able to analyze the closest k number of training instances (nearest neighbors) and returns the most common class or the mean of k nearest neighbors for the classification and regression task respectively [81].
6. Meta classifiers: Meta classifiers either enhance a single classifier or combine several classifiers. Bagging predictors is a method for generating multiple versions of a predictor and using them in order to get an aggregated predictor [82].

On the other hand, Auto-WEKA was restricted to use tree-based classifiers – Decision Stump, J48, Logistic model tree (LMT), REPTree, RT and M5P – and rule-based classifiers – JRip, One Rule (OneR), PART and M5Rules.

5. Results

In this section, we present the main results of our experiments that were outlined in Section 4. We will show that the application of autoML technique in educational datasets significantly improves the efficiency of classic machine learning algorithms.

5.1. Predicting Pass/Fail Students

At first, we conducted a series of experiments to identify the effectiveness of classic data mining algorithms to predict students that are likely to fail at early enough stages. We performed 6 classic machine-learning algorithms on the datasets of 3 courses, split into chronological sets (month A, month B, etc.). An exception is the “Physical Chemistry I” course, which did not have any logs on the first month, and as such, we did not conduct experiments during that period. Tables 8–10 present the effectiveness results, represented by the accuracy measure.

We observe that Bagging and the Random Forest ensemble algorithms outperform the others in most cases in the first course, Naïve Bayes, PART and IBk algorithms outperform the others in most cases in the second course, and SMO and Random Forest algorithms outperform the others in most

cases in the third course. In total, SMO and Random Forest are noted to be among the best performers 9 times and Bagging 7 times. As such, we could not conclude one best performer for our given datasets.

Table 8. Overall accuracy results regarding the “Physical Chemistry I” course and Pass/Fail classification task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Naïve Bayes	-	59.22	68.09	75.18	75.18	75.18
Random Forest	-	62.06	78.01 *	82.62 *	81.56 *	81.56 *
Bagging	-	62.4	74.47	81.21	81.21	81.20
PART		63.12 *	73.76	72.70	75.89	74.11
SMO	-	58.16	74.82	79.43	80.85	80.14
IBk-5NN	-	59.93	70.92	78.01	79.43	79.08
Auto-WEKA	-	66.67 Random Tree	81.20 LMT	83.68 LMT	82.27 REPTree	81.56 PART

t-test: *p*-value = 0.0365, α = 0.05.

Table 9. Overall accuracy results regarding the “Physics III” course and Pass/Fail classification task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Naïve Bayes	53.33	61.67	61.67	63.33	67.78 *	67.78 *
Random Forest	52.78	54.44	60.56	60.00	61.67	60.56
Bagging	57.22	57.78	60.56	61.11	66.67	62.22
PART	62.22 *	63.33 *	58.89	62.22	55.00	52.22
SMO	56.11	61.67	64.44 *	65.56 *	64.44	63.89
IBk-5NN	56.67	58.89	60.56	60.56	60.56	60.00
Auto-WEKA	61.11 PART	73.33 J48	66.67 OneR	69.44 LMT	71.11 J48	70.56 JRip

t-test: *p*-value = 0.0317, α = 0.05.

Table 10. Overall accuracy results regarding the “Analytical Chemistry Laboratory” course and Pass/Fail classification task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Naïve Bayes	61.24	63.57	66.67	83.72	84.50	84.50
Random Forest	78.29	85.27 *	86.05 *	90.70 *	90.70	89.92
Bagging	79.84	85.27 *	84.50	88.37	88.37	88.37
PART	72.09	76.74	75.97	86.04	86.82	86.82
SMO	81.40	86.05	86.05 *	89.92	91.47 *	90.70 *
IBk-5NN	75.19	85.27 *	84.50	89.92	89.92	89.92
Auto-WEKA	81.40 LMT	86.82 JRip	86.05 LMT	93.02 LMT	93.02 LMT	93.02 LMT

t-test: *p*-value = 0.0223, α = 0.05.

In addition, we used Auto WEKA to run automated machine learning experiments for the corresponding datasets. From the results, we identify that in most cases, the accuracy was significantly increased. Auto-WEKA was able to optimize the results up to 10% (Figure 4). The suggested classifiers and hyperparameters again vary across the datasets, including LMT, J48, PART, and JRip, to name

a few. Overall, tree-based classifiers were suggested the most. More specifically, the LMT method was the outperformer 8 out of 17 times.

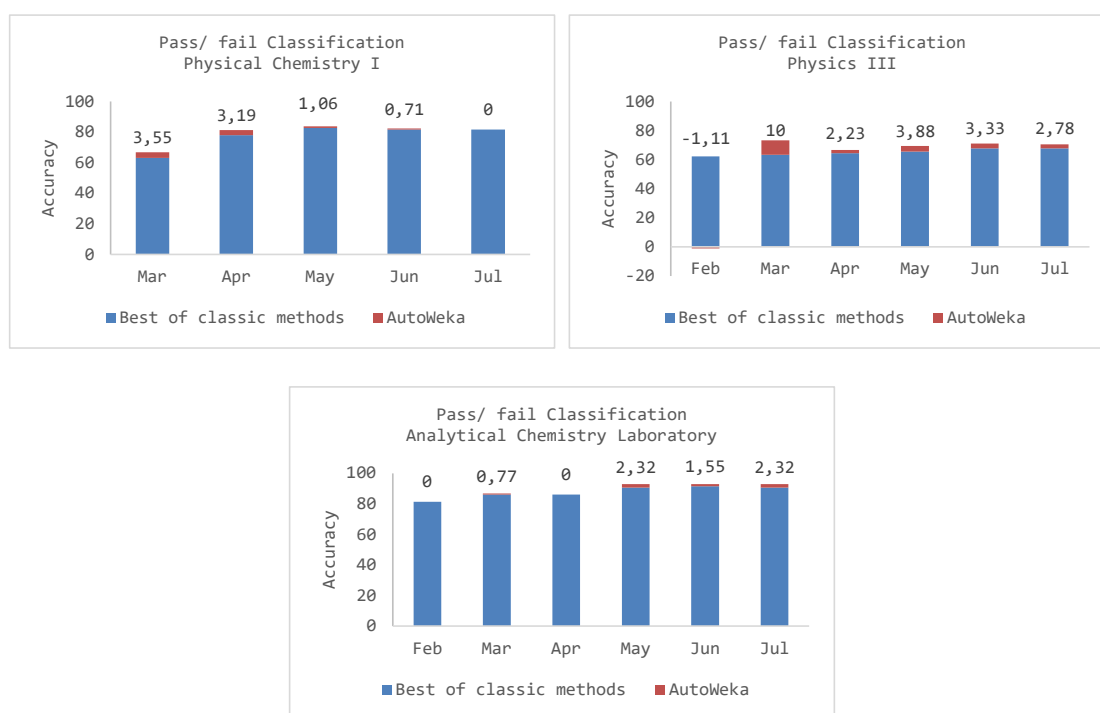


Figure 4. Comparative results of the effectiveness of autoML over classic EDM methods. The results indicate that in most cases, the effectiveness of the best classic classifier was improved when we applied autoML.

Finally, by applying the *t*-test on the results as shown in Tables 8–10, we obtained the following *p*-values: “Physical Chemistry I” *p*-value = 0.0365; “Physics III” *p*-value = 0.0317; Analytical Chemistry Laboratory *p*-value = 0.0223. Thus, we can conclude that the autoML presents a statistically significant increase when applied to specific educational datasets.

5.2. Predicting Students’ Academic Performance

In addition, we conducted a series of experiments to identify the effectiveness of classic data mining algorithms to predict students’ grades at early enough stages. Next, we compared the results with the predictions given by models created by applying the autoML. Similar to Section 5.1, the experiments comprise of six phases, one for each month of the semester. Tables 11–13 present the regression results, represented by the mean absolute error measure.

Depending on the dataset, we observe that Bagging and the Random Forest algorithms outperform the others in most cases in the first course, Bagging and SMOreg algorithms outperform the others in most cases in the second course, and Random Forest algorithm outperforms the others in most cases in the third course. In total, Bagging is noted to be among the best performers 10 times, Random Forest 9 times, and SMOreg 5 times. Overall, from the predictions generated we could not conclude one best performer for our given datasets.

In addition, we used Auto-WEKA to run the automated machine learning experiments for the corresponding datasets. From the results, we identify that in all cases, the mean absolute error was significantly decreased. Auto-WEKA was able to minimize the error from 0.0188 to 0.4055 (Figure 5). The suggested classifiers and hyperparameters again vary across the datasets, including M5P, Random Tree, REPTree and M5Rules. Overall, tree-based methods were primarily suggested. More specifically, the M5P and Random Tree were the outperformers 5 out of 17 times.

Table 11. Overall MAE results regarding the “Physical Chemistry I” course and regression task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Random Forest	-	2.4264	1.9346	1.7334 *	1.5784 *	1.5731 *
M5Rules	-	2.4504	2.0541	1.8244	1.8169	1.782
Bagging	-	2.3903 *	1.901 *	1.7495	1.6081	1.6136
SMOreg	-	2.4604	2.1352	1.9962	1.8998	1.9037
IBk-5NN	-	2.4433	2.1556	1.9011	1.7521	1.774
Auto-WEKA	-	1.9848 REPTree	1.8822 Random Tree	1.6715 M5P	1.4017 Random Tree	1.4255 Random Tree

t-test: *p*-value = 0.0366, α = 0.05.

Table 12. Overall MAE results regarding the “Physics III” course and regression task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Random Forest	2.944	2.676	2.6011	2.507	2.4429	2.4946
M5Rules	2.8632 *	2.6879	2.6901	2.5311	2.5291	2.5235
Bagging	2.8638	2.6932	2.5707	2.5179	2.4094 *	2.3855 *
SMOreg	3.1291	2.6246	2.5372 *	2.3305 *	2.4123	2.4305
IBk-5NN	2.8962	2.5815 *	2.5832	2.5777	2.5059	2.5448
Auto-WEKA	2.8194 M5P	2.0091 M5Rules	2.2386 Random Tree	2.1743 REPTree	2.2015 M5P	2.1276 M5Rules

t-test: *p*-value = 0.0021, α = 0.05.

Table 13. Overall MAE results regarding the “Analytical Chemistry Lab” course and regression task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Random Forest	2.1137	1.7123 *	1.62 *	1.381 *	1.3825 *	1.3864 *
M5Rules	2.1347	1.9989	1.6982	1.4233	1.5202	1.5492
Bagging	2.0715 *	1.795	1.7555	1.4894	1.4902	1.4864
SMOreg	2.1972	1.9377	1.7385	1.5705	1.5401	1.5303
IBk-5NN	2.3073	1.8891	1.7302	1.4574	1.4434	1.4426
Auto-WEKA	1.7935 REPTree	1.6067 REPTree	1.4947 M5P	1.2135 M5Rules	1.0402 M5P	1.2135 M5Rules

t-test: *p*-value = 0.0016, α = 0.05.

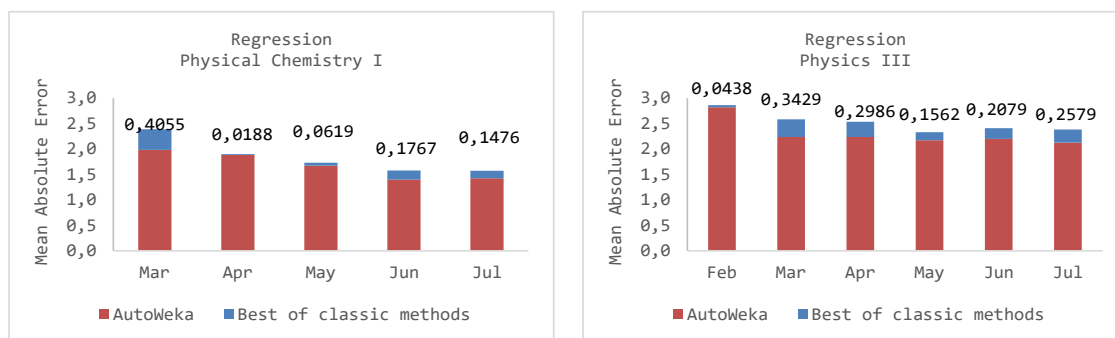


Figure 5. Cont.

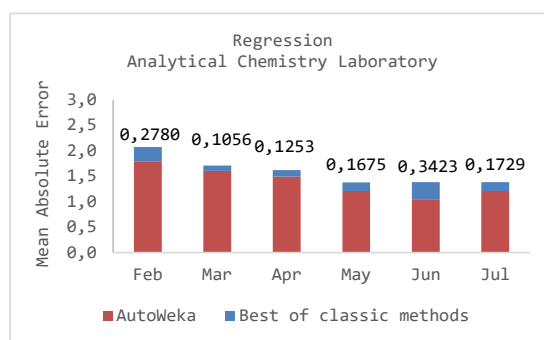


Figure 5. Comparative results of the effectiveness of autoML over classic ML methods. The results indicate that in all cases, the effectiveness of the best classic classifier was improved when we applied autoML.

Finally, by applying the *t*-test on the results as shown in Tables 11–13, we obtained the following *p*-values: “Physical Chemistry I” *p*-value = 0.0366; “Physics III” *p*-value = 0.0021; Analytical Chemistry Laboratory *p*-value = 0.0016. Similarly, a statistically significant decrease was observed in the overall measurements.

5.3. Predicting Dropout Students

Lastly, we conducted a series of experiments to identify the effectiveness of classic data mining algorithms to predict students that are likely to drop out at early enough stages. We compared the results with the predictions given by models created by applying the autoML. Similar to Sections 5.1 and 5.2, the experiments consist of six phases, one for each month of the semester. As the Analytical Chemistry Laboratory course displays a large level of class imbalance—the dropout class represents the 6% of the dataset—we did not include it in this category of experiments. Since it is easy to get high accuracy without actually making useful predictions in imbalanced datasets [66,83], for the drop out problem we used the ROC measure to compare the results of the classifiers. For the same reason, in Auto-WEKA we set the ‘area above ROC’ as the metric to optimize. Tables 14 and 15 present the drop results.

Depending on the dataset, we observe that SMO and Bagging algorithms outperform the others in most cases in the first course, and Bagging algorithm outperforms the others in most cases in the second course. In total, Bagging is noted to be among the best performers 7 times, and SMO 4 times. We could not result in one best performer.

Table 14. Overall ROC area results regarding the “Physical Chemistry I” course and dropout classification task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Naïve Bayes	-	0.520	0.731	0.793 *	0.817	0.817
Random Forest	-	0.569	0.763	0.777	0.871 *	0.869 *
Bagging	-	0.512	0.774 *	0.791	0.860	0.854
PART	-	0.599 *	0.710	0.740	0.796	0.787
SMO	-	0.500	0.500	0.688	0.820	0.818
IBk-5NN	-	0.594	0.700	0.758	0.804	0.811
Auto-WEKA	-	0.801 J48	0.863 LMT	0.828 LMT	0.928 LMT	0.896 LMT

t-test: *p*-value = 0.0308, α = 0.05.

Table 15. Overall ROC area results regarding the “Physics III” course and dropout classification task.

	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018
Naïve Bayes	0.463	0.680	0.710 *	0.699	0.743	0.744
Random Forest	0.487	0.697	0.704	0.689	0.708	0.716
Bagging	0.495	0.723 *	0.690	0.702	0.747 *	0.760 *
PART	0.477	0.590	0.660	0.722 *	0.651	0.742
SMO	0.500 *	0.500	0.500	0.500	0.496	0.496
IBk-5NN	0.486	0.649	0.632	0.655	0.740	0.737
Auto-WEKA	0.545	0.883	0.778	0.801	0.842	0.784
	Decision Stump	Random Tree	Random Tree	J48	LMT	LMT

t-test: *p*-value = 0.0039, α = 0.05.

Auto-WEKA experiments again proved to be more effective. From the results, we identify that in all cases, the ROCK curve measure was increased. Auto-WEKA was able to optimize the results from 0.018 to 0.202 (Figure 6). The suggested classifiers and hyperparameters again vary across the datasets, including LMT, Random Tree, and J48. Overall, only tree-based algorithms were suggested. More specifically, the LMT was the outperformer 6 out of 17 times.

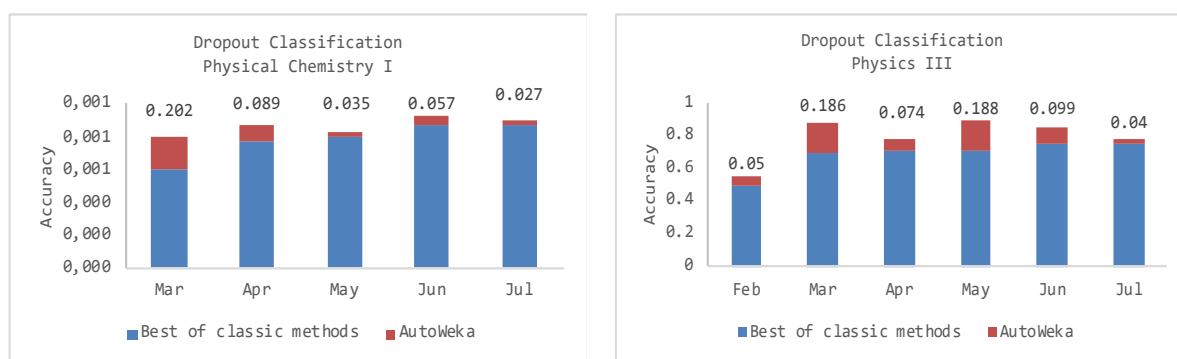


Figure 6. Comparative results of the effectiveness of autoML over classic ML methods. The results indicate that in all cases, the effectiveness of the best classic classifier was improved when we applied autoML.

Finally, by applying the *t*-test on the results as shown in Tables 14 and 15, we obtained the following *p*-values: “Physical Chemistry I” *p*-value = 0.0308; “Physics III” *p*-value = 0.0039. A statistically significant increase was observed in all courses.

6. Discussion

The results illustrated in previous sections unveil the effectiveness of the autoML methods in educational data mining processes. Without any human intervention, the suggested models report performance often much better of classic supervised learning techniques with default hyperparameter settings. Even from the early half of the semester, predictions have satisfactory values. The result models can help educators and instructors to identify weak students, improve retention and reduce academic failure rates. It can also lead to improved educational outcomes.

Finding appropriate models and hyperparameter configurations is one of the most difficult processes when building a machine learning solution and indeed, it is central to the pursuit of precision. The main advantage of autoML and tools like Auto-WEKA is that they provide an out-of-the-box mechanism to build reliable machine learning models without the need for advanced data science knowledge. People in IT, educational administration, teaching, research or learning support roles,

who want to explore how their students perform, are not always experts in educational data mining. Therefore, despite having basic knowledge, novice ML users can converge easily to a suitable algorithm and its related hyperparameter settings and develop trusted machine learning models to support their educational institutions.

In our comparative study we focus on classification and regression using educational data. We evaluated widely known ML algorithms from 6 different categories versus 2 allowed categories in Auto-WEKA, on 51 datasets exported from 3 compulsory courses, using the Moodle LMS as a complement to face-to-face lectures. For each course, we collected 6 samples of data—one for each month of the semester—in order to predict students’ final performance as soon as possible. The total number of students (591) who attended the courses produced more than 130,000 log events between February 2018 and July 2018. Logs were triggered by a variety of learning modules that structured the courses—forums, pages, resources, assignments, folders, and URLs. Assignment grades were scaled to a common metric. The final performance for each ML task—pass/fail, regression and dropout—was adjusted to represent the class attribute appropriately.

We further indicated the informative features for each course using the extremely randomized trees. We concluded that attributes related to assignment grades were more important than views counters when performing regression tasks. For pass/fail classification, apart from assignment grades, some features related to how many times a student accessed a resource or a page also had high importance scores. Lastly, dropout classification tasks yielded the widest variety of features among its important compared with the other tasks.

As previous studies have shown [17,84,85], there is no algorithm that is the best across all classification problems. A similar conclusion was reached by our experiments, as we could not result in one best performer in our educational data mining tasks illustrated in Section 5. We applied 6 well known supervised machine learning algorithms—Naïve Bayes, Random Forest, Bagging, PART/M5Rules, SMO/SMOreg and IBk—one representative of the six main learners’ categories. Each time we compared the performance of the classic models with the calculated by Auto-WEKA model. Auto-WEKA was set to allow only ten tree or rule-based classifiers. The automated one was in most cases better. Significant differences calculated by *t*-test were marked. We did not conclude an overall winner classifier suggested by the Auto-WEKA tool either. However, when we grouped the proposed classifiers under the 6 learner’s categories, it was clear that Auto-WEKA suggestions were mostly tree-based classifiers (Table 16).

Table 16. Classic outperformers and Auto-WEKA suggestions grouped under 6 learners’ categories. Twelve out of 17 proposed Auto-WEKA classifiers (71%) were tree classifiers. Similarly, 14 out of 17 regressors (82%) and 11 out of 11 (100%) suggested dropout models also belong to the tree classifiers category.

	Pass/Fail		Regression		Dropout	
	Best of Classic Methods	Auto-WEKA	Best of Classic Methods	Auto-WEKA	Best of Classic Methods	Auto-WEKA
bayes	5%	-	Not applicable	Not applicable	12%	-
functions	27%	-	19%	-	4%	-
lazy	9%	-	7%	-	8%	-
meta	21%	-	37%	-	32%	-
rules	9%	29%	4%	18%	12%	0%
tree	27%	71%	33%	82%	32%	100%
	functions/trees	trees	meta	trees	meta/trees	trees

There are several advantages to using decision trees in classification and prediction applications. Decision trees models are easy to interpret and explain [14]. Compared to other algorithms, they require

less effort for data preparation during pre-processing. A decision tree does not require normalization of data nor scaling of data.

Finally, automated hyperparameter optimization familiarized us with a wide range of models and configurations that were practically applied to our settings [10]. It allowed us to test models with many variables that would be complicated to be tuned by hand.

However, some limitations should also be noted. First, it was not clear what was the appropriate time limit that imposed on our (relatively small) datasets. We marked differences between the suggested models for the same datasets when Auto-WEKA was executed e.g., for 5 min and when it was executed for 1 or 2 h. In limited cases, as the time limit increased, the performance was not necessarily better. This behavior is probably due to the fact that the parameter space is too large to explore, and different randomizations (e.g., during the train/test splitting, or in the underline SMAC optimizer that is used) allow to explore a smaller or a larger part of it. Another possible explanation is the fact that datasets of such size are prone to overfitting and underfitting. In any case, the suggested model was generally beneficial over using default values. Secondly, autoML methods, by definition, take much longer to train. Such an effect could be afforded due to the reliability of the results.

7. Conclusions and Future Research Directions

This study has investigated the effectiveness of autoML techniques for the early identification of students' performance in three compulsory courses supported by the Moodle e-learning platform. In our experimental evaluation, we focused on classification and regression. We further limited the configuration space of autoML methods to allow only tree and rule-based classifiers, in order to enhance the interpretability and explainability of the resulting models. Our results provide evidence that tools optimizing hyperparameters rather than choosing default values achieve state-of-the-art performance in educational settings as well. The comparison we made reveals that in the majority of cases, hyperparameter optimization results in better performances than default values for a set of classic learning models. We also noted that in most cases, the proposed configuration included tree-based classifiers. On this basis, we believe that autoML procedures and tools like Auto-WEKA can help people in education—both experts and novices in the field of data science.

Moreover, the proposed method may serve as a significant aid in the early estimation students' performance, and thus enabling timely support and effective intervention strategies. Appropriate software extensions within learning management systems could be built, to enable non-expert users benefit from autoML. Meanwhile, such tools should not lack transparency. It is essential to incorporate features that enable the interpretability and explainability of the produced results to a certain extent. Understanding why a student is prone to fail can help to better align the learning activities and support with the students' needs. This assumption could be addressed in future studies. Overall, the potential use of automatic machine learning methods in the educational field opens up new horizons for educators so as to enhance their use of data coming from educational settings, and ultimately improve academic results.

Author Contributions: S.K. and O.R. conceived and designed the experiments; M.T. performed the experiments; M.T. and G.K. analyzed the data; S.K. contributed analysis tools; M.T. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Romero, C.; Ventura, S. Data mining in education. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2013**, *3*, 12–27. [[CrossRef](#)]
2. Bakhshinategh, B.; Zaiane, O.R.; ElAtia, S.; Ipperciel, D. Educational data mining applications and tasks: A survey of the last 10 years. *Educ. Inf. Technol.* **2018**, *23*, 537–553. [[CrossRef](#)]

3. Romero, C.; Ventura, S. Educational data mining: A review of the state of the art. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2010**, *40*, 601–618. [[CrossRef](#)]
4. Bousbia, N.; Belamri, I. Which Contribution Does EDM Provide to Computer-Based Learning Environments? In *Educational Data Mining*; Springer: Basel, Switzerland, 2014; pp. 3–28.
5. Romero, C.; Ventura, S. Educational data science in massive open online courses. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2017**, *7*, e1187. [[CrossRef](#)]
6. Wolff, A.; Zdrahal, Z.; Herrmannova, D.; Knoth, P. Predicting student performance from combined data sources. In *Educational Data Mining*; Springer: Basel, Switzerland, 2014; pp. 175–202.
7. Campbell, J.P.; DeBlois, P.B.; Oblinger, D.G. Academic analytics: A new tool for a new era. *Educ. Rev.* **2007**, *42*, 40–57.
8. Romero, C.; Ventura, S. Educational data mining: A survey from 1995 to 2005. *Expert Syst. Appl.* **2007**, *33*, 135–146. [[CrossRef](#)]
9. Daniel, B. Big Data and analytics in higher education: Opportunities and challenges. *Br. J. Educ. Technol.* **2015**, *46*, 904–920. [[CrossRef](#)]
10. Bergstra, J.S.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: New York, NY, USA, 2011.
11. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*; Curran Associates Inc.: New York, NY, USA, 2012.
12. Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013.
13. Bergstra, J.; Yamins, D.; Cox, D.D. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In Proceedings of the 12th Python in Science Conference, Brussels, Belgium, 21–25 August 2013.
14. Galitsky, B. Customers' Retention Requires an Explainability Feature in Machine Learning Systems They Use. In Proceedings of the 2018 AAAI Spring Symposium Series, Palo Alto, CA, USA, 26–28 March 2018.
15. Martens, D.; Vanthienen, J.; Verbeke, W.; Baesens, B. Performance of classification models from a user perspective. *Decis. Support Syst.* **2011**, *51*, 782–793. [[CrossRef](#)]
16. Došilović, F.K.; Brčić, M.; Hlupić, N. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*; IEEE: Opatija, Croatia, 2018.
17. Hämmäläinen, W.; Vinni, M. Classifiers for educational data mining. In *Handbook of Educational Data Mining*; CRC Press: Boca Raton, FL, USA, 2010; pp. 57–74.
18. Conijn, R.; Snijders, C.; Kleingeld, A.; Matzat, U. Predicting student performance from LMS data: A comparison of 17 blended courses using Moodle LMS. *IEEE Trans. Learn. Technol.* **2016**, *10*, 17–29. [[CrossRef](#)]
19. Márquez-Vera, C.; Cano, A.; Romero, C.; Ventura, S. Predicting student failure at school using genetic programming and different data mining approaches with high dimensional and imbalanced data. *Appl. Intell.* **2013**, *38*, 315–330. [[CrossRef](#)]
20. Moreno-Marcos, P.M.; Alario-Hoyos, C.; Muñoz-Merino, P.J.; Kloos, C.D. Prediction in MOOCs: A review and future research directions. *IEEE Trans. Learn. Technol.* **2018**, *12*. [[CrossRef](#)]
21. Mueen, A.; Zafar, B.; Manzoor, U. Modeling and predicting students' academic performance using data mining techniques. *Int. J. Mod. Educ. Comput. Sci.* **2016**, *8*, 36–42. [[CrossRef](#)]
22. Amrieh, E.A.; Hamtini, T.; Aljarah, I. Mining educational data to predict student's academic performance using ensemble methods. *Int. J. Database Theory Appl.* **2016**, *9*, 119–136. [[CrossRef](#)]
23. Kaur, P.; Singh, M.; Josan, G.S. Classification and prediction based data mining algorithms to predict slow learners in education sector. *Procedia Comput. Sci.* **2015**, *57*, 500–508. [[CrossRef](#)]
24. Guo, B.; Zhang, R.; Xu, G.; Shi, C.; Yang, L. Predicting students performance in educational data mining. In Proceedings of the 2015 International Symposium on Educational Technology (ISET), Wuhan, China, 27–29 July 2015.
25. Saa, A.A. Educational data mining & students' performance prediction. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 212–220.

26. Costa, E.B.; Fonseca, B.; Santana, M.A.; de Araújo, F.F.; Rego, J. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Comput. Hum. Behav.* **2017**, *73*, 247–256. [[CrossRef](#)]
27. Asif, R.; Merceron, A.; Ali, S.A.; Haider, N.G. Analyzing undergraduate students' performance using educational data mining. *Comput. Educ.* **2017**, *113*, 177–194. [[CrossRef](#)]
28. Kostopoulos, G.; Kotsiantis, S.; Pintelas, P. Predicting student performance in distance higher education using semi-supervised techniques. In *Model and Data Engineering*; Springer: New York, NY, USA, 2015; pp. 259–270.
29. Elbadrawy, A.; Polyzou, A.; Ren, Z.; Sweeney, M.; Karypis, G.; Rangwala, H. Predicting student performance using personalized analytics. *Computer* **2016**, *49*, 61–69. [[CrossRef](#)]
30. Xu, J.; Moon, K.H.; van der Schaar, M. A machine learning approach for tracking and predicting student performance in degree programs. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 742–753. [[CrossRef](#)]
31. Strecht, P.; Cruz, L.; Soares, C.; Mendes-Moreira, J.; Abreu, R. A Comparative Study of Classification and Regression Algorithms for Modelling Students' Academic Performance. In Proceedings of the 8th International Conference on Educational Data Mining, Madrid, Spain, 26–29 June 2015.
32. Meier, Y.; Xu, J.; Atan, O.; van der Schaar, M. Personalized grade prediction: A data mining approach. In Proceedings of the 2015 IEEE International Conference on Data Mining, Atlantic City, NJ, USA, 14–17 November 2015.
33. Sweeney, M.; Rangwala, H.; Lester, J.; Johri, A. Next-term student performance prediction: A recommender systems approach. *arXiv* **2016**, arXiv:1604.01840.
34. Kostopoulos, G.; Kotsiantis, S.; Fazakis, N.; Koutsonikos, G.; Pierrakeas, C. A Semi-Supervised Regression Algorithm for Grade Prediction of Students in Distance Learning Courses. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1940001. [[CrossRef](#)]
35. Tsiakmaki, M.; Kostopoulos, G.; Koutsonikos, G.; Pierrakeas, C.; Kotsiantis, S.; Ragos, O. Predicting University Students' Grades Based on Previous Academic Achievements. In Proceedings of the 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA), Zakynthos, Greece, 23–25 July 2018.
36. Márquez-Vera, C.; Cano, A.; Romero, C.; Noaman, A.Y.M.; Fardoun, H.M.; Ventura, S. Early dropout prediction using data mining: A case study with high school students. *Expert Syst.* **2016**, *33*, 107–124. [[CrossRef](#)]
37. Zhang, Y.; Oussena, S.; Clark, T.; Kim, H. Use Data Mining to Improve Student Retention in Higher Education-A Case Study. In Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 1, DISI, Funchal, Madeira, Portugal, 8–12 June 2010.
38. Delen, D. A comparative analysis of machine learning techniques for student retention management. *Decis. Support Syst.* **2010**, *49*, 498–506. [[CrossRef](#)]
39. Lykourantzou, I.; Giannoukos, I.; Nikolopoulos, V.; Mparadis, G.; Loumos, V. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Comput. Educ.* **2009**, *53*, 950–965. [[CrossRef](#)]
40. Superby, J.-F.; Vandamme, J.P.; Meskens, N. Determination of factors influencing the achievement of the first-year university students using data mining methods. In Proceedings of the Workshop on Educational Data Mining, Jhongli, Taiwan, 26–30 June 2006.
41. Herzog, S. Estimating student retention and degree-completion time: Decision trees and neural networks vis-à-vis regression. *New Dir. Inst. Res.* **2006**, *2006*, 17–33. [[CrossRef](#)]
42. Kostopoulos, G.; Kotsiantis, S.; Pintelas, P. Estimating student dropout in distance higher education using semi-supervised techniques. In Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, 1–3 October 2015.
43. Rao, S.S. *Engineering Optimization: Theory and Practice*; John Wiley & Sons: Toronto, ON, Canada, 2009.
44. Brochu, E. *Interactive Bayesian Optimization: Learning User Preferences for Graphics and Animation*; University of British Columbia: Vancouver, BC, Canada, 2010.
45. Feurer, M.; Hutter, F. Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Springer: New York, NY, USA, 2019; pp. 3–33.
46. Brochu, E.; Cora, V.M.; de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv* **2010**, arXiv:1012.2599.

47. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning-Methods, Systems, Challenges*; Springer: New York, NY, USA, 2019.
48. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
49. Bengio, Y. Gradient-based optimization of hyperparameters. *Neural Comput.* **2000**, *12*, 1889–1900. [[CrossRef](#)]
50. Maron, O.; Moore, A.W. The racing algorithm: Model selection for lazy learners. *Artif. Intell. Rev.* **1997**, *11*, 193–225. [[CrossRef](#)]
51. Simon, D. *Evolutionary Optimization Algorithms*; Wiley: Hoboken, NJ, USA, 2013.
52. Guo, X.C.; Yang, J.H.; Wu, C.G.; Wang, C.Y.; Liang, Y.C. A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. *Neurocomputing* **2008**, *71*, 3211–3215. [[CrossRef](#)]
53. Dewancker, I.; McCourt, M.; Clark, S. *Bayesian Optimization Primer*; SigOpt. 2015. Available online: https://app.sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf (accessed on 12 June 2019).
54. Hutter, F.; Lücke, J.; Schmidt-Thieme, L. Beyond manual tuning of hyperparameters. *Künstliche Intell.* **2015**, *29*, 329–337. [[CrossRef](#)]
55. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE* **2016**, *104*, 148–175. [[CrossRef](#)]
56. Williams, C.K.I.; Rasmussen, C.E. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 2.
57. Bergstra, J.; Yamins, D.; Cox, D.D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
58. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Trees*; Wadsworth Int. Group: Belmont, CA, USA, 1984; Volume 37, pp. 237–251.
59. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2011.
60. Eggenberger, K.; Feurer, M.; Hutter, F.; Bergstra, J.; Snoek, J.; Hoos, H.; Leyton-Brown, K. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In Proceedings of the NIPS Workshop on Bayesian Optimization in Theory and Practice, Lake Tahoe, NV, USA, 10 December 2013.
61. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [[CrossRef](#)]
62. Kushner, H.J. A new method of locating the maximum point of an arbitrary multippeak curve in the presence of noise. *J. Basic Eng.* **1964**, *86*, 97–106. [[CrossRef](#)]
63. Srinivas, N.; Krause, A.; Kakade, S.M.; Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv* **2009**, arXiv:0912.3995.
64. Clark, S.; Liu, E.; Frazier, P.; Wang, J.; Oktay, D.; Vesdapunt, N. MOE: A Global, Black Box Optimization Engine for Real World Metric Optimization. 2014. Available online: <https://github.com/Yelp/MOE> (accessed on 12 June 2019).
65. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [[CrossRef](#)]
66. Jeni, L.A.; Cohn, J.F.; de la Torre, F. Facing imbalanced data—recommendations for the use of performance metrics. In Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, Geneva, Switzerland, 2–5 September 2013.
67. Ling, C.X.; Huang, J.; Zhang, H. AUC: A better measure than accuracy in comparing learning algorithms. In Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence, Halifax, NS, Canada, 11–13 June 2003.
68. Provost, F.; Fawcett, T. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, USA, 14–17 August 1997.
69. Frank, E.; Hall, M.A.; Witten, I.H. *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed.; Morgan Kaufmann: Massachusetts, MA, USA, 2016.
70. Kotthoff, L.; Thornton, C.; Hoos, H.H.; Hutter, F.; Leyton-Brown, K. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *J. Mach. Learn. Res.* **2017**, *18*, 826–830.

71. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. *Data Mining: Practical machine learning tools and techniques*; Morgan Kaufmann: Massachusetts, MA, USA, 2016.
72. John, G.H.; Langley, P. Estimating continuous distributions in Bayesian classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–20 August 1995.
73. Eibe, F.; Witten, I.H. *Generating Accurate Rule Sets without Global Optimization*; University of Waikato, Department of Computer Science: Hamilton, New Zealand, 1998.
74. Holmes, G.; Hall, M.; Prank, E. Generating rule sets from model trees. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Sydney, Australia, 6–10 December 1999.
75. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
76. Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods—Support Vector Learning*; Schoelkopf, B., Burges, C., Smola, A., Eds.; MIT Press: Massachusetts, MA, USA, 1998.
77. Keerthi, S.S.; Shevade, S.K.; Bhattacharyya, C.; Murthy, K.R.K. Improvements to Platt’s SMO Algorithm for SVM Classifier Design. *Neural Comput.* **2001**, *13*, 637–649. [[CrossRef](#)]
78. Hastie, T.; Tibshirani, R. Classification by Pairwise Coupling. In *Advances in Neural Information Processing Systems*; MIT Press: Massachusetts, MA, USA, 1998.
79. Shevade, S.K.; Keerthi, S.S.; Bhattacharyya, C.; Murthy, K.R.K. Improvements to the SMO Algorithm for SVM Regression. *IEEE Transactions on Neural Netw.* **2000**, *11*. [[CrossRef](#)]
80. Smola, A.J.; Schoelkopf, B. *A Tutorial on Support Vector Regression*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1998.
81. Aha, D.; Kibler, D. Instance-based learning algorithms. *Mach. Learn.* **1991**, *6*, 37–66. [[CrossRef](#)]
82. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [[CrossRef](#)]
83. Kim, B.-H.; Vizitei, E.; Ganapathi, V. GritNet: Student performance prediction with deep learning. *arXiv* **2018**, arXiv:1804.07405.
84. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006.
85. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).