Massive LMS Log Data Analysis for the Early Prediction of Course-Agnostic Student Performance

Moises Riestra-González[a], Maria del Puerto Paule-Ruíz[b], Francisco Ortin[b,c,*]

[a] Accenture SL, Applied Intelligence Department, c/ Jimena Fernandez de la Vega 140, Edificio Asturias, offices 1 A-E, 33202, Gijon, Spain

[b] University of Oviedo, Computer Science Department, c/Federico Garcia Lorca 18, 33007, Oviedo, Spain

[c] Cork Institute of Technology, Computer Science Department, Rossa Avenue, Bishopstown, T12 P928, Cork, Ireland

---

Notice: This is the authors' version of a work accepted for publication in Computers & Education journal. Please, cite this document as:

Moises Riestra-González, Maria del Puerto Paule-Ruíz, Francisco Ortin. Massive LMS Log Data Analysis for the Early Prediction of Course-Agnostic Student Performance. Computers & Education, volume 163, pp. 104108-104128 (2021), doi: 10.1016/j.compedu.2020.104108

Massive LMS Log Data Analysis for the Early Prediction of Course-Agnostic Student Performance

Moises Riestra-González[a], Maria del Puerto Paule-Ruiz[b], Francisco Ortin[b,c,*]

[a] *Accenture SL, Applied Intelligence Department, c/ Jimena Fernandez de la Vega 140, Edificio Asturias, offices 1 A-E, 33202, Gijon, Spain*

[b] *University of Oviedo, Computer Science Department, c/Federico Garcia Lorca 18, 33007, Oviedo, Spain*

[c] *Cork Institute of Technology, Computer Science Department, Rossa Avenue, Bishopstown, T12 P928, Cork, Ireland*

Abstract

The early prediction of students' performance is a valuable resource to improve their learning. If we are able to detect at-risk students in the initial stages of the course, we will have more time to improve their performance. Likewise, excellent students could be motivated with customized additional activities. This is why there are research works aimed to early detect students' performance. Some of them try to achieve it with the analysis of LMS log files, which store information about student interaction with the LMS. Many works create predictive models with the log files generated for the whole course, but those models are not useful for early prediction because the actual log information used for predicting is different to the one used to train the models. Other works do create predictive models with the log information retrieved at the early stages of courses, but they are just focused on a particular type of course.

In this work, we use machine learning to create models for the early prediction of students' performance in solving LMS assignments, by just analyzing the LMS log files generated up to the moment of prediction. Moreover, our models are course agnostic, because the datasets are created with all the University of Oviedo courses for one academic year. We predict students' performance at 10%, 25%, 33% and 50% of the course length. Our objective is not to predict the exact student's mark in LMS assignments, but to detect at-risk, fail and excellent students in the early stages of the course. That is why we create different classification models for each of those three student groups. Decision tree, naïve Bayes, logistic regression, multilayer perceptron (MLP) neural network, and support vector machine models are created and evaluated. Accuracies of all the models grow as the moment of prediction increases. Although all the algorithms but naïve Bayes show accuracy differences lower than 5%, MLP obtains the best performance: from 80.1% accuracy when 10% of the course has been delivered to 90.1% when half of it has taken place. We also discuss the LMS log entries that

most influence the students' performance. By using a clustering algorithm, we detect six different clusters of students regarding their interaction with the LMS. Analyzing the interaction patterns of each cluster, we find that those patterns are repeated in all the early stages of the course. Finally, we show how four out of those six student-LMS interaction patterns have a strong correlation with students' performance.

Keywords: Learning management systems, early prediction, interaction patterns, student performance, machine learning

1. Introduction

Higher education institutions have incorporated information and communication technologies to support the way lecturers teach and students learn. One widespread example of that support is the utilization of Learning Management Systems (LMS), whose usage has grown significantly in the last years [1]. An LMS is a software application for the delivery, documentation, tracking, reporting and administration of educational courses and learning, training and development programs [2]. By using the Internet, LMSs represent a valuable tool to transform the traditional face-to-face courses into blended and online programs [3]. Moreover, LMSs are also used to support the majority of face-to-face courses, because of the functionalities they provide such as course content management, communication, assignment delivery and assessment, online questionnaires and quizzes, and student grading, among others [4].

LMSs generate log files that contain data about user interaction (e.g., course and resource view, assignment submission and evaluation, and quiz and forum interaction). Such information has been used to create predictive models for different purposes such as foreseeing student performance [3], detecting procrastination [5] and clustering students [6]. Data in log files describe patterns of how students interact with LMSs, and such patterns may involve some correlation with their performance. However, the patterns found are commonly discovered for one single course [6, 7] or a set of courses sharing similar methodologies and/or structures [8]. An open question is if LMSs interaction patterns could be found from massive log data taken from multiple heterogeneous courses with different methodologies, knowledge areas, learning formats (i.e., online, blended and face-to-face), duration and evaluation systems.

Another important feature of the predictive models built from LMS log files is their early prediction capability. If one model is able to identify at-risk students in the initial stages of the course, lecturers could devise ways of supporting their learning [9].

*Corresponding author

Email addresses: moises.riestra@accenture.com (Moises Riestra-González), paule@uniovi.es (Maria del Puerto Paule-Ruiz), ortin@uniovi.es (Francisco Ortin)

URL: http://www.reflection.uniovi.es/ortin (Francisco Ortin)

Likewise, students identified as excellent could be motivated with additional activities throughout the course. However, most works commonly build their predictive models from data gathered from whole the course, neglecting the practical value of an early prediction, while the course is in progress [10]. To build reliable models that predict student performance at early stages of the course, such models should be trained and validated only with log information gathered before the moment of prediction [10]. In this way, those models could be eventually used in real scenarios, when the course is in progress and log information available is only that generated so far.

The main contribution of this work is the use of massive LMS log data (15,994 students and 8.5 million log entries) for multiple (669) heterogeneous (face-to-face, online and blended) courses in order to predict student performance in solving LMS assignments at the early stages of the course, regardless the particular characteristics of each course. In this way, lecturers can develop early intervention strategies for both at-risk and excellent students when it is still viable to improve their performance.

The dataset created from the LMS log files is also used to create course-agnostic student clusters, and calculate their correlation with students' performance. We study the clusters to analyze how students are classified depending on behaviors such as course and resource view, procrastination, quiz and assignment evaluation, and forum participation. Cluster information can be used to create adaptive educational systems [11, 12] and intelligent tutoring systems that, depending on the student cluster, could refine the scaffolding provided to students, particularly important in online courses [13]. Moreover, students' clusters and their LMS log information could be used by instructors to provide adaptive feedback, customized assessment, and more personalized attention as needed [14].

1.1. Research questions

To see to what extent we achieve the expected contribution, we address the following four research questions:

1. Is it possible to predict the students' performance in solving the LMS assignments, by just analyzing their LMS logs when only 10%, 25%, 33% and 50% of the course has been completed?
2. Which are independent variables derived from the LMS logs that most influence the students' performance?
3. Are there student clusters that group students regarding their usage of the LMS, regardless the course features?
4. Is there any correlation between those student clusters and their performance?

For the early prediction of student performance in solving the LMS assignments, we use different supervised learning algorithms from a dataset built from the LMS logs.

Then, we study an interpretable white-box model to analyze the influence of the independent LMS variables on the dependent one (students' performance). Student clusters are built with the same dataset, merging similar features with feature agglomeration, and then running the k-means unsupervised learning algorithm. We finally study each cluster and their correlation with students' performance.

The rest of this paper is structured as follows. Next section details related work and Section 3 describes how the dataset is built. The methodology, evaluation and discussion of the predictive models is presented in Section 4. Section 5 applies the same structure to describe student clustering. Section 6 presents the conclusions and future work.

2. Related work

Predicting students' academic performance is a challenge already faced by the educational scientific community. We first describe the existing works closer to our approach. Such works are those that a) perform student's performance prediction by using LMS information; b) are not course-specific, considering multiple heterogeneous courses; and c) undertake the predictions with LMS information gathered before the moment of prediction.

Conijn *et al.* analyze 17 blended courses with 4,989 students using Moodle LMS [3]. Their objective is to predict students' final grades from LMS predictor variables and from in-between assessment grades, using logistic (pass/fail) and standard regression (final grade) models. They predict students' performance for the first 10 weeks. Accuracies slightly improve as the week of prediction increases, with a notable improvement after week 5, when in-between assessment grades become available. At week 5, the regression model showed 0.43 adjusted R^2, and the pass/fail binary classifier obtained 67% accuracy in week 3. Unlike our system, Conijn *et al.* do not create specific models to detect excellent and severe at-risk students. Their target variable is the final exam, whereas ours is LMS assignments.

Costa *et al.* compare the effectiveness of existing Educational Data Mining (EDM) techniques for the early identification of students likely to fail two introductory programming courses [15]: a distance course with 262 students and 10-week duration, and an on-campus course with 161 students. Unlike the system described in this article, datasets include not only LMS interaction information, but also other data such as age, gender, civil status, city, income, enrollment year, discipline and student performance in the weekly activities and exams. The highest F_1-measures for week 1 were 0.77 and 0.8 for the two courses. These values grow as the moment of prediction increases. Although the work is not specific for one course, two courses might not be sufficient to create course-agnostic models.

Tomasevic *et al.* build classification and regression models for the task of predicting student exam marks [16]. They use the Open University Learning Analytics Dataset

(OULAD), which stores student demographics, student's performance in course assessments, and student engagement (not just LMS log data). Models are built with only two courses from the OULAD dataset. Unlike our proposal, their classification model just distinguishes between fail and pass students. Prediction takes place at the moment immediately before the final exam, but also after different intermediate assessments. F_1-measures grow as the moment of prediction increases: from 78% (first assessment) up to 94.9% (the sixth one). For the moment before the final exam, F_1-measure is 96.6%. The regression models to predict the finals show a similar pattern. Neural network models provide the best results for both regression and classification.

As in our study, the research work of Cobo *et al.* clusters students from different courses, considering their interaction with LMSs [17]. They use an agglomerative hierarchical clustering algorithm to identify the different participation profiles adopted by learners in online discussion forums. The experiments conducted analyze the activity carried out by learners within the forums of three different courses in an online Telecommunications degree. Information is taken from an asynchronous web-based teaching environment, and the participation of learners in the discussion forums is not mandatory. The whole dataset involves 672 students in 18 different classrooms and 2,842 posts. Five different clusters are found. Inactive learners (shirkers), who neither read nor write messages, have very low performance. For only readers (lurkers), active students correlate with pass, and non-active with fail. Finally, most active read and write learners (workers), correlate with pass. Apart from the number of courses, the main difference with our approach is that they do not use all the log information in the LMS.

2.1. Other research works aimed at predicting students' performance

There are other research works that use information retrieved from LMSs to predict students' performance. The following ones use multiple courses, but do not provide early prediction, while the course is in progress. Instead, they use all LMS data generated through the whole course, reducing their early prediction capability.

Gerritsen uses Moodle log files of 17 courses to forecast whether a student will pass or fail a course (binary classification) [18]. Out of seven models, multilayer perceptron outperformed the other classifiers, managing to detect at-risk students with 66.1% accuracy.

Gašević *et al.* build different logistic regression models for nine undergraduate courses to predict students' performance (pass or fail) [19]. They use LMS logs and student information from the institutional student information system. They build one model including all the courses, and one model per course. They compute the area under the ROC (Receiving Operating Characteristic) curve (AUC) values. The model for all the courses have an acceptable accuracy ($0.5 \leq \text{AUC} < 0.7$). However, those models specifically built for a particular course achieve excellent ($0.8 \leq \text{AUC} < 0.9$) or outstanding ($\text{AUC} \geq 0.9$) performance.

Romero *et al.* try to predict students' final marks with LMS data, such as the quizzes and assignments passed and failed, and the time spent on quizzes, forums and assignments [20]. They compare the performance of the following data mining techniques for classifying students: statistical methods, decision trees, rule induction and neural networks. They use several classification methods in seven different Moodle courses, obtaining the best performance with CART decision trees (65%).

The motivation in [21] is to study the portability of students' performance predictive models among courses with the same degree and similar level of LMS usage. They create J48 decision trees models from 24 courses to classify pass/fail students. When porting models between courses in the same degree, AUC values fall between 0.09 and 0.28. These loses range from 0.22 to 0.25 when porting models between courses with similar level of Moodle usage.

Some works are aimed at early predicting students' performance, but they create course-specific models. In [10] the authors build three datasets from LMS data to determine how early the warning system can accurately predict students' performance. Those datasets collect statistics for the first four, eight, and thirteen weeks of an online course with 330 students. CART decision tree classifiers provide 95% accuracy in week 4, with no significant difference for weeks 8 and 13.

The work in [22] proposes different data mining approaches to predict if students pass or fail the course, using forum participation indicators. They use 114 university students in a first-year course in computer science. They build one set of classification and clustering models at the end of the course, and another one in the middle of it. Average accuracies in the middle of the course were between 70% and 80%, and 80%-90% for the model built with the entire course data.

Marbouti *et al.* create three logistic regression models to identify at-risk students in a first-year engineering course at weeks 2, 4 and 9 [23]. For the week-2 and week-4 models, they use as predictors attendance records, homework, and quiz grades. For the week-9 model, they include mid-term exam grades. The models identify at-risk students with overall accuracy of 79% at week 2, 90% at week 4, and 98% at week 9.

The Early Warning System (EWS) plug-in for Moodle predicts students' performance in a first-year Communication and Information Literacy course in week 4 of the semester [24]. A linear regression model is used to foresee the numeric final mark, which is simply computed as the sum of all coursework marks at the end of week 14. The linear regression model showed an adjusted R^2 of 0.608.

Some other works create predictive models for a single course, using LMS information stored throughout the whole course. Macfadyen and Dawson collect LMS data including 'whole term' counts for frequency of usage of course materials and tools supporting content delivery, engagement and discussion, assessment, and administration/management [9]. They build a regression model to predict students' grades, and a logistic regression classifier to detect students at risk of failure, in an online course.

The regression model explains more than 30% of the variation in students' final grades; while the classifier identifies at-risk students with 70.3% accuracy.

Ljubobratović *et al.* create random forest models to predict student success on the base of input predictors (lectures, quizzes, labs and videos) extracted from Moodle logs, with 96.3% accuracy [25]. They study the dependence of predictors on the target value, finding that scores in labs and quizzes have the strongest influence on the final mark.

2.2. Other research works aimed at student clustering

The following research works are aimed at clustering students regarding their interaction with LMSs. They cluster students of one particular course rather than creating course-agnostic groups. Talavera and Gaudioso cluster students to discover student-LMS interaction patterns from an Internet course [26]. Their dataset includes LMS log data, background knowledge, demographic data and student interests taken from surveys filled in by the students. The Expectation Maximization (EM) algorithm found six clusters showing a non-perfect but still adequate correlation with students' performance [26].

Hung *et al.* take 17,934 LSM log entries for 98 undergraduate students in a course to discover students' online learning behaviors [27]. The k-means clustering algorithm found three clusters: clusters 1 and 2 group students with performance above the average and high LMS interaction; whereas cluster 3 relate low interaction patterns to low performance.

Cerezo *et al.* use data extracted from Moodle logs to cluster students of an eLearning course to study their asynchronous learning process [6]. K-means and EM algorithms are used, finding that procrastination and socialization variables are the most determinant for cluster membership. Out of the four clusters, three groups show differences in the final marks of students.

The PPP algorithm predicts students' performance through procrastination behaviors using assignment submission data extracted from Moodle [28]. Students of one course are grouped into different procrastination behavior categories. Three clusters are found: procrastinator, procrastinator candidate and non-procrastinator. Then, students are classified according to their performance, using different classification algorithms and their procrastination class, with 96% accuracy.

López *et al.* propose a classification via clustering approach to predict students' final marks in a university course, by just using forum data [29]. They use different clustering algorithms to predict whether students pass or fail the course, based on their usage of the Moodle forum. The Expectation Maximization clustering algorithm obtained similar accuracy to traditional supervised machine learning algorithms.

Another work uses LMS information not to classify students but courses [30]. LMS information from 612 courses are used, finding four different clusters: 50.5% of the courses show inactive utilization of the LMS, 24.3% present significantly higher use of

Original LMS log files			Filtered LMS log files		
Courses	Students	LMS entries (student actions)	Courses	Students	LMS entries (student actions)
5,112	29,602	47,097,824	699	15,944	8,540,418

Table 1: LMS log data.

Q&A and work groups; 18% of the courses make higher utilization of lecture notes, links and discussion forums; and 7.2% use more resources and assignments.

3. Dataset

To create different predictive models, we build a dataset from LMS log files. In particular, we took all the log files of the LMS used by all the courses delivered in the University of Oviedo, during the academic course 2017/2018. For that academic year, the LMS hosted 5,112 courses in which 29,602 students were enrolled. Those courses comprise online, on-campus and blended modules, of both undergraduate and graduate degrees, and multiple disciplines (arts, humanities, science, engineering, health, social sciences and law). University of Oviedo uses the widespread Moodle LMS to support most of its courses [31].

The purpose of considering many courses of different disciplines, formats, durations and methodologies is to make our models sufficiently general for distinct courses. Single-course predictive models are commonly able to provide better performance, but they are hard to port to other courses. Previous works have measured from 9% to 28% accuracy drop when a model is used in another course of the same degree, and from 22% to 25% loss for courses sharing similar usage of the LMS [21]. Likewise, Gašević *et al.* show that accuracies of course-specific models to predict students' performance are from 60% to 80% higher than course-agnostic models, using the same information extracted from the LMS [19].

All the student interactions with the Moodle LMS are recorded in log files, which is the only information we used to build the predictive and clustering models presented in this article. All the log files were anonymized for the sake of confidentiality.

3.1. Course filtering

As mentioned, we only work with data taken from LMS log files. We do not have the student grades for the courses they are enrolled in, because of data protection reasons. Therefore, we measure students' performance by using their marks in the LMS assignments, as detailed in Section 3.5. This make us filter those courses with no assignments in the LMS, as shown in Table 1. After that filtering, we have 699 courses with almost 16,000 students and more than 8.5 million log entries.

3.2. Estimate of course duration

The early prediction of student performance strongly depends on the duration of each course, because different time-dependent variables (features) are used [10]. However, the courses delivered by a university in an academic year have many different durations. They could be annual, semi-annual and even one- or two-weeks long, particularly in graduate degrees. Unfortunately, the anonymization process deleted all the information about the course, including its length. Therefore, we must define a mechanism to estimate course duration from the LMS log files.

Log data from the Moodle LMS indicates when a course starts, but not when it ends. To estimate that value, we took 31 different courses that we know their exact duration, and tried to estimate their end date by analyzing the student log files. To that aim, we ordered the log entries by date and computed the percentile values, ignoring outliers (i.e., entries with dates outside the interval $[Q_1 - 1.5IQR,\ Q_3 + 1.5IQR]$[1]) [32]. We then computed the percentile that best matches the course end date, finding that value to be 95%. That is, 5% of the log entries (ignoring outliers) take place when the course has already ended.

3.3. Moments of prediction

We want to create early predictors of student performance, so that lecturers could devise ways of supporting their learning. An important decision is to define the moment of prediction. If we are able to detect at-risk students at the beginning of the course, it will be easier to increase their performance (we will have more time for it). However, we will have fewer entries in the LMS log files, and hence prediction accuracy will probably be not very high. It seems quite reasonable to think that the later the prediction is, the better its accuracy will be. Therefore, we build different models for various moments of prediction to analyze this expected trade-off between model accuracy and moment of prediction. The moments of prediction used were 10%, 25%, 33% and 50% of the course duration.

3.4. Feature engineering

LMS log files contain raw data about actions undertaken by students (Appendix A) and evaluation tasks (Appendix B). Student actions include course access, lecturer and external (URL) resource view, quiz submission and view, and forum interaction. Some entries related to evaluation tasks include assignment submission and view, and assignment evaluation.

We process raw data in LMS log files to extract the features of the dataset used to build our predictive models. In this way, many absolute variables are converted into

[1] Q stands for quartile and IQR stands for interquartile range.

relative features, including time dependent ones. For example, we create a `ResourceViewPct` feature that holds the percentage of accesses to lecturer resources, relative to the total accesses to lecturer resources for all the students in that course. Likewise, `CourseViewTime1` indicates the first time the student accessed the course, relative to its duration (i.e., percentage of the course length).

It is common to propose optional assignments to students. Optional assignments influence students' grades, but usually less than mandatory ones. Therefore, we thought it could be useful to include features about the potential optionality of assignments. We estimated such optionality the following way. We took 30 control courses that provide both optional and mandatory assignments in the LMS. For such courses, we computed the percentage of students submitting each assignment, and compute the submission threshold that differentiates optional from mandatory assignments. In all the cases, assignments with less than 40% submissions were optional. Therefore, we used that value to create different evaluation features for optional and mandatory assignments (Appendix B). For example, `AccomplishOptionalPctGraded` provides the percentage of optional assignments delivered by the student until the moment of prediction. `AccomplishMandatoryPercentileGrade` tells us the student's average mark in the mandatory assignments, relative to (percentile) the rest of the students in the same course.

It is worth noting that all the feature values are always computed with the log information obtained before the moment of prediction. Since the purpose of our work is the early prediction of at-risk and excellent students, we only use the LMS log information generated by students until the moment of prediction. In this way, when we build a model to predict student performance at 10% of the course duration, it is highly probable that we do not have any assignment evaluation data to perform such prediction.

Different discretization and rebalance processing techniques were used to test if better models could be obtained. All the numeric data were discretized using equal width, equal frequency and ChiMerge methods, with 3, 4 and 5 intervals [33]. The rebalance techniques used were random undersampling, Tomek link, random oversampling and SMOTE [34]. We achieved no benefits on the performance of the models created using those discretization and rebalance techniques, so neither of the processes were finally applied to the dataset.

3.5. Dependent (target) variable

Since we do not have the final grades of students, we define student performance as their marks in the assignments published in the LMS (see Research Question 1). To this aim, we compute the dependent (target) variable as an aggregation of all the student's marks for the LMS assignments retrieved throughout the course. Given that the assignment marks before the moment of prediction are included as independent variables, we check that no model is created with 50% (or more) of the student's marks

used to compute the dependent variable.

Rather than just defining one aggregate of students' marks, we tried different equations, and selected the one with closer values to the final grades. We took the 30 control courses used in the previous section, for which we know the students' final grades. Then, we computed the two following equations with different values of α (assignment marks and submissions are for each student):

$$10 \times \left(\alpha \frac{\sum mandatory\ assignment\ marks}{mandatory\ assignment\ submissions} + (1-\alpha) \frac{\sum optional\ assignment\ mark}{optional\ assignment\ submissions} \right) \quad (1)$$

$$10 \times \left(\alpha\ \frac{\sum mandatory\ assignment\ marks}{mandatory\ assignments} + (1-\alpha)\ \frac{\sum optional\ assignment\ mark}{optional\ assignments} \right) \quad (2)$$

Both equations give an α weight (value between 0 and 1) to mandatory assignments and 1-α to the optional ones. The first equation computes the average values of assignments submitted only, whereas the second one considers the mark of a non-submitted assignment as zero.

Table 2[2] shows differences between the two equations and the actual students' grades, for different values of α. Such differences are computed as mean square error (MSE), root-mean-square error (RMSE), mean absolute error (MAE), coefficient of determination (R^2) and adjusted R^2 [35]. We also include the Person's and Spearman's correlation coefficients. We took the second equation with α=0.5 as our target to represent students' performance, since that aggregate is the closest one to the final grade for the 30 control courses used. That equation has a strong correlation with final grades and explains 47.16% of its variation.

4. Predictive models

We create different predictive models from the dataset described in Section 3 to answer Research Question 1. Such models will show us to what extent it is possible to predict students' performance on the LMS assignments, at different moments of prediction.

Section 3.5 defined the target variable as a continuous value between 0 and 10. However, our intention is not to predict that exact value (regression models), but to detect both at-risk and excellent students (classification models) to customize, reinforce and improve their learning. For these two kinds of students, we define two values in the 0

[2]To interpret the values of MSE, RMSE and MAE, please notice that marks are normalized between 0 and 1.

Equation	α	MSE	RMSE	MAE	R^2	Adjusted R^2	Pearson	Spearman
	1	0.0070	0.0881	0.0756	0.3439	0.3146	0.6110	0.6272
	0.75	0.0064	0.0810	0.0694	0.3742	0.3424	0.6650	0.6826
(1)	0.5	0.0063	0.0793	0.0680	0.3820	0.3495	0.6789	0.6969
	0.25	0.0068	0.0861	0.0738	0.3519	0.3219	0.6253	0.6419
	0	0.0077	0.0967	0.0830	0.3132	0.2865	0.5565	0.5713
	1	0.0062	0.0841	0.0710	0.4111	0.4043	0.7165	0.7798
	0.75	0.0055	0.0755	0.0638	0.4579	0.4503	0.7981	0.8685
(2)	**0.5**	**0.0054**	**0.0733**	**0.0619**	**0.4716**	**0.4638**	**0.8218**	**0.8944**
	0.25	0.0059	0.0798	0.0674	0.4334	0.4262	0.7553	0.8220
	0	0.0068	0.0922	0.0778	0.3751	0.3689	0.6537	0.7114

Table 2: Aggregates used to compute students' performance, and their difference with final students' grades (bold font shows the lowest differences).

Mark thresholds:	≤ 2.5	< 5	≥ 8.5
Percentage of students in the dataset:	25.76%	39.39%	20.34%

Table 3: Number of students in each group regarding their marks.

to 10 scale to create binary classification models. We take 2.5 as the threshold mark for at-risk students, and 8.5 for excellent ones. We also take 5.0 as another mark threshold to differentiate pass students from fail ones[3]. Table 3 shows how predicting whether a student will pass or not will be more difficult than detecting at-risk and excellent students, because pass/fail students are much closer to be balanced (61%/39%) in our dataset than at-risk (25.76%) and excellent (20.34%) students.

These three mark thresholds define three different binary classifiers. Since we also consider four different moments of prediction (10%, 25%, 33% and 50%), we create a total of 12 different models.

4.1. Methodology

We divide the dataset into 80% of the instances for training and 20% for testing, using a stratified random sampling method [36]. We repeat the training plus testing process 30 times, measuring the mean, standard deviation and 95% confidence intervals of accuracy, F_1-measure and AUC (Area Under Curve) values [36]. Data split is random and stratified to ensure that the proportions between classes are the same in each fold, as they are in the whole dataset.

We use the following algorithms to build each of the 12 the binary classifiers mentioned above: CART Decision Trees (DT), Naïve Bayes (NB), Logistic Regression (LG), Multilayer Perceptron (MLP) and Support Vector Machines (SVM). The algorithm im-

[3]In the Spanish educational system, 5 points out of 10 is the standard minimum grade to pass any course.

plementations are those from the scikit-learn 0.22.2 framework [37], namely the Python classes `DecisionTreeClassifier` (DT), `GaussianNB` (NB), `LogisticRegression` (LG), `MLPClassifier` (MLP) and `SVC` (SVM). We normalize all the feature values between 0 and 1, since some classifiers such as MLP and SVN show better performance with normalized features [38]. All the code is implemented in Python 3.7.6.

For all the models, we first perform a feature selection process and then hyper-parameter tuning. Features are selected with the scikit-learn `RFECV` feature ranking algorithm with recursive feature elimination [37]. The best number of features is selected with stratified randomized 3-fold cross validation (`StratifiedKFold`), accuracy as the evaluation metric, and with at least 10 features selected for each model.

For hyper-parameter tuning, we start with the default hyper-parameters provided by scikit-learn. Then, we select the best hyper-parameters for each algorithm with exhaustive parallel search across common parameter values (`RandomizedSearchCV`), using stratified randomized 3-fold cross validation (`StratifiedKFold`). Accuracy is the metric used to measure the performance of each hyper-parameter combination.

Appendix C details all the hyper-parameters used for each algorithm [39]. For the particular case of the MLP artificial network, we try with different number of hidden layers (from 1 to 3) with 10 to 30 units per layer, obtaining the best accuracy with 1 layer and 20 units.

All the algorithms, including feature selection and hyper-parameter tuning, are executed in parallel using all the cores in our server. We use a Dell PowerEdge R530 server with two Intel Xeon E5-2620 v4 2.1GHz microprocessors (32 cores) with 128GB DDR4 2400MHz RAM memory, running CentOS operating system 7.4-1708 for 64 bits.

4.2. Results

Figure 1 shows accuracies of all the classifiers for the three mark thresholds and growing moments of prediction. Figure 1 represents as baseline the prediction that takes the most frequent class, because the dataset is not balanced.

Table 4 details the AUC values obtained for the 5 algorithms implementing the 12 models. AUC measures the entire two-dimensional area underneath the ROC curve, which considers the diagnostic ability of a binary classifier taking into account its true- and false-positive rates [40]. Table 4 shows in bold font the greatest AUC values. When there are not significant differences statistically between two algorithms (p-value$<$0.05 for *Student's t* test), multiple accuracies are displayed in bold font.

Figure 2 shows the ROC curves for the 12 classifiers built with MLP. This is the only algorithm used in Figure 2, because it obtains the best AUC results in all the scenarios. Table 5 compares MLP accuracy and the most frequent class, for all the models. Table 6 details the best $F_{0.5}$-, F_1- and F_2-measures in order to discuss the cost of false positives and negatives (type I and type II errors) in different models.

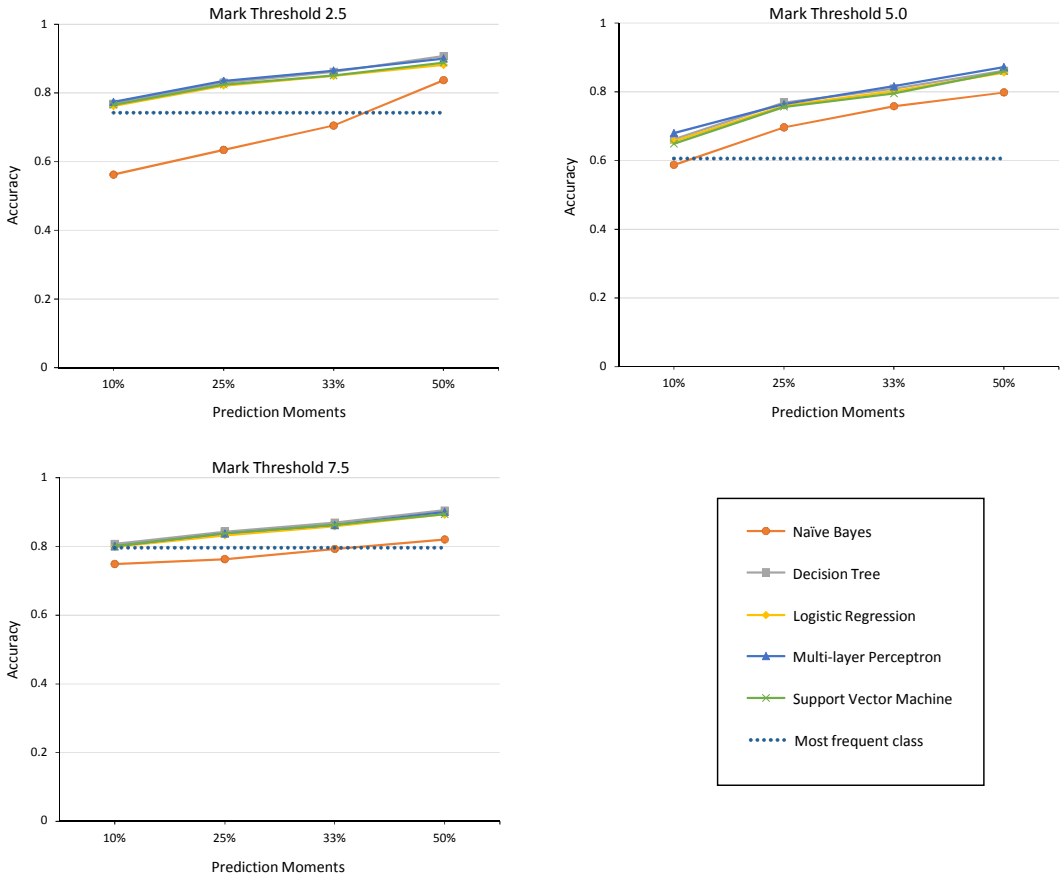

Figure 1: Model accuracies for 3 mark thresholds and growing moments of prediction.

4.3. Discussion

Figure 1 and Table 4 show how the predictive models are more accurate as the moment of prediction increases, for the three mark thresholds. The log files generated by the LMS when half of the course has taken place have more information than those generated for just 10% of the course, thus producing more accurate models. That is why it is not advisable to build early predictive models with information taken from the whole course [10].

Apart from naïve Bayes, the rest of the algorithms behave quite similar, showing differences in their accuracy lower than 5%. They always perform better than the most frequent class.

For the moment of prediction 10%, MLP is able to classify pass/fail students (mark 5.0) with 68% of accuracy, 7.38% more than the baseline (Table 5). This difference drops to 3.14% for at-risk students (mark 2.5) and just 0.42% for excellent ones (mark 8.5). For these two last groups, there is less room from improvement, since data are not balanced.

As mentioned, prediction accuracy keeps growing as the moment of prediction grows.

Moment of prediction	Mark threshold 2.5	5.0	8.5	Algorithm
10%	0.70501	0.67395	0.60651	NB
	0.73664	0.68949	0.67235	DT
	0.72470	0.71220	0.65633	LR
	0.76016	**0.72783**	**0.68783**	MLP
	0.70990	0.71260	0.63956	SVC
25%	0.79181	0.74912	0.70559	NB
	0.84986	0.82854	**0.82014**	DT
	0.84773	0.83737	0.78587	LR
	0.87717	**0.84596**	**0.82083**	MLP
	0.85885	0.83609	0.79264	SVC
33%	0.81336	0.81923	0.75494	NB
	0.89871	0.88324	**0.88024**	DT
	0.88105	0.88386	0.83148	LR
	0.91724	**0.89223**	**0.88046**	MLP
	0.89199	0.88375	0.83206	SVC
50%	0.87335	0.88090	0.80257	NB
	0.93633	0.93506	**0.93515**	DT
	0.92493	0.93435	0.87315	LR
	0.95834	**0.94669**	**0.93199**	MLP
	0.93858	0.93388	0.89550	SVC

Table 4: AUC measures for all the models and algorithms. Bold font indicates highest value. Multiple values are in bold font when there is no significant difference.

At half the course, we are able to reach accuracies higher than 90% for both at-risk and excellent student detection (these two values are higher than 83% for 25% of the course). For pass/fail classification, we get 87.2% accuracy at 50% of the course duration (Table 5).

Table 4 analyzes differences among algorithms. In all the cases, MLP is the algorithm that performs the best. For three models, there is no significant difference between MLP and CART decision trees (DT). Figure 2 shows the prediction capability of MLP models regarding the moment of prediction, for the three different mark thresholds. In that figure, we can clearly see how the classifier performance grows as the moment of prediction increases.

One important discussion to be considered is the cost of false positives and false negatives. In this scenario, we consider positive as a student with a mark higher than the mark threshold (2.5, 5.0 or 8.5). For 2.5 and 5.0 models, we would rather have models with high precision, reducing the number of at-risk and fail students wrongly classified (reduce false positives). That is, the cost of wrong classification of at-risk and fail students is higher than the cost of erroneously classifying not-at-risk and pass students. For these case scenarios, the $F_{0.5}$-measure is used to give twice the weight to precision as recall [41]—F_1-measure gives the same weight to precision and recall.

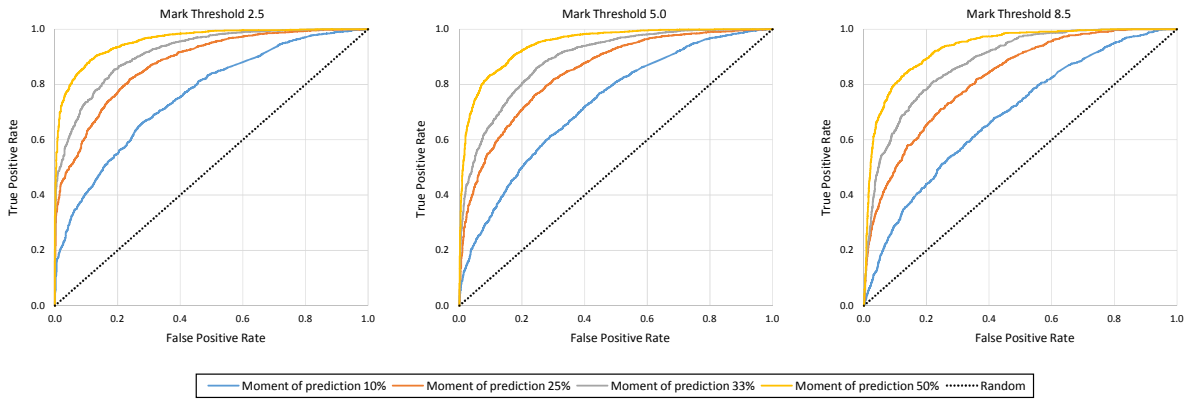

Figure 2: ROC curves for MLP classifiers.

Mark threshold	Classifier	Moment of prediction 10%	25%	33%	50%
2.5	Most Frequent Class	74.24%	74.24%	74.24%	74.24%
	MLP	77.38%	83.49%	86.50%	90.02%
5	Most Frequent Class	60.61%	60.61%	60.61%	60.61%
	MLP	67.99%	76.37%	81.61%	87.17%
8.5	Most Frequent Class	79.66%	79.66%	79.66%	79.66%
	MLP	80.08%	83.75%	86.24%	90.06%

Table 5: MLP accuracy vs the most frequent class in all the datasets.

Mark threshold	F-measures	Moment of prediction 10%	25%	33%	50%
2.5	F_1	MLP (0.8634)	MLP (0.8944)	MLP (0.9121)	DT (0.9382)
	$F_{0.5}$	MLP (0.8130)	MLP (0.8681)	DT (0.8970)	DT (0.9300)
5.0	F_1	MLP (0.7527)	MLP (0.8129)	MLP (0.8525)	MLP (0.8941)
	$F_{0.5}$	MLP (0.7133)	DT (0.7948)	DT (0.8382)	MLP (0.8781)
8.5	F_1	DT (0.8909)	DT (0.9074)	DT (0.9217)	DT (0.9419)
	F_2	DT (0.9370)	DT (0.9457)	DT (0.9464)	DT (0.9540)

Table 6: Best $F_{0.5}$-, F_1- and F_2-measures for different mark thresholds and moments of prediction.

For excellent students (mark≥8.5), we may give more weight to recall than to precision. The idea is to allow students to perform optional additional activities to increase their learning. Therefore, we want to reduce the number of excellent students not detected by our model (false negatives). F_2-measure doubles the weight given to recall.

Table 6 shows the best $F_{0.5}$- and F_1-measures for 2.5 and 5.0 models, and the best F_1- and F_2-measures for the models predicting excellent students. The two best algorithms are MLP and DT. Lecturers may use one of these two algorithms depending on the weight they want to give to the misclassification of students, for the three different mark thresholds.

After all these analyses, we can answer **Research Question 1**: it is possible to predict the students' performance in solving the LMS assignments, by analyzing their LMS logs when only 10%, 25%, 33% and 50% of the course has been completed. For 10% of the course, there is a statistically significant difference with the most frequent class only for 2.5 and 5.0 mark thresholds; all the differences are significant for 25% on. Prediction accuracy grows as the moment of prediction increases. The two best classifiers are MLP and DT, to be chosen depending on the mark threshold and the costs to be given to false positives and false negatives.

4.4. Analysis of independent variables

Besides the performance of predictive models, it is also interesting to analyze the generated models to see to what extent the LMS log entries influence students' performance (Research Question 2). Not all the algorithms we use create interpretable white-box models. DT has this capability, since the paths from root to leaf nodes can be interpreted as classification rules [42]. Such classification rules are based on the values of the features. Since DT models also provides high accuracy to classify students by their performance (Tables 4 and 6), we study DT models to analyze dependencies among independent variables and the target.

When building a DT, information gain is used to determine which feature provides the maximum information about a class. One measure of information gain is Gini importance, which computes the importance of each feature as the sum over the number of splits that include that feature, relative to the number of samples it splits [43, 44]. Table 7 shows the five features with greatest Gini importance for all the models. 57% of the most important variables belong to evaluation tasks (Appendix B) and 43% are related to actions performed by students (Appendix A). For the moment of prediction 10%, fewer evaluation variables are used because it is less common to have evaluated assignments. The weight of evaluation variables grows as the moment of prediction increases. That weight is 43%, 56%, 62% and 64% for, respectively, 10%, 25%, 33% and 50% prediction moments.

The independent variable with the strongest influence on the target is `Accomplish-MandatoryGrade`, which describes the average mark of mandatory assignments before the moment of prediction, considering non-submitted assignments as zero mark. It

is the most influential variable in all the models but one, where it is the second most influential. Evaluation variables (`Accomplish*Grade`) are more important as the moment of prediction increases, because at the very early stages of the course many students do not have any assignment evaluation.

Table 7 shows that student accesses to resources uploaded by lecturers (`ResourceViewPct` and `ResourceViewUniquePct`) are important variables to detect at-risk and fail students (mark thresholds 2.5 and 5.0), predicting such students when the values of those variables are low. On the contrary, this kind of variables are not decisive to classify excellent students.

The `CourseViewPct` variable measures the percentage of course accesses relative to the accesses of the rest of students enrolled in the same course. This variable is only important to detect excellent students. It seems that such learners access the course LMS more often than the rest of the students, particularly at the early stages of the course.

The independent variables associated to discussion forums (`ForumViewForumPct` and `ForumViewDiscussionPct`) do not appear as the five most influential variables, for any of the models (Table 7). It seems that student interaction with forums is not correlated with their performance, for course-agnostic models. Likewise, students' performance does not strongly depend on quiz variables, since only two independent variables are used for the 12 predictive models (Table 7).

These analyses comprise the response to **Research Question 2**, aimed at identifying the variables that most influence the students' performance using course-agnostic LMS log data. Evaluation variables are more important as the moment of prediction increases; a small number of accesses to lecturer's resources identify at-risk and fail students; a high number of course accesses at early stages detect excellent students; quizzes have very low influence on students' performance; and discussion forums none.

5. Student clustering

The objective of this section is to answer research questions 3 and 4. We first see if it is possible to cluster students by their interaction with the LMS, regardless the characteristics, methodology and discipline of the course. After identifying such clusters, we analyze whether there is any correlation between the student groups found and their performance in solving the assignments published in the LMS.

5.1. Methodology

The input dataset to obtain student clusters is the one described in Section 3. The dependent variable defined to estimate student's performance (Section 3.5) is not included, since we use unsupervised leaning to find the clusters. For tuples with missing values, we use the mean substitution method, based on replacing null values with

	Prediction Moment			
	10%	25%	33%	50%
Mark Threshold 2.5	AccomplishMandatory-Grade (0.36)	AccomplishMandatory-Grade (0.66)	AccomplishMandatory-Grade (0.29)	AccomplishMandatory-PctGraded (0.38)
	ResourceViewUniquePct (0.12)	ResourceViewUniquePct (0.08)	AccomplishMandatory-PctGraded (0.27)	AccomplishMandatory-Grade (0.35)
	AssignSubmitUniquePct (0.07)	AssignSubmitUniquePct (0.04)	AccomplishMandatory-PercentileGrade (0.11)	AccomplishMandatory (0.09)
	AccomplishOptionalPct-Graded (0.05)	AssignViewUniquePct (0.02)	AccomplishMandatory (0.06)	ResourceViewUniquePct (0.02)
	AccomplishOptional-PercentileGrade (0.04)	AccomplishMandatory (0.02)	AssignViewUniquePct (0.04)	AssignViewUniquePct (0.02)
Mark Threshold 5.0	AccomplishMandatory-Grade (0.17)	AccomplishMandatory-Grade (0.56)	AccomplishMandatory-Grade (0.72)	AccomplishMandatory-Grade (0.77)
	QuizCloseAttempt-UniquePct (0.12)	AccomplishMandatory-PercentileGrade (0.17)	AssignViewUniquePct (0.05)	AccomplishMandatory-PercentileGrade (0.09)
	AssignSubmitPct (0.07)	AssignViewUniquePct (0.03)	AssignSubmitUniquePct (0.03)	AssignViewUniquePct (0.02)
	ResourceViewPct (0.06)	ResourceViewUniquePct (0.03)	AccomplishMandatory-PctGraded (0.02)	AccomplishMandatory (0.02)
	AccomplishMandatory-PctGraded (0.06)	AssignSubmitUniquePct (0.02)	ResourceViewUniquePct (0.02)	AssignSubmitUniquePct (0.01)
Mark Threshold 8.5	AccomplishMandatory-Grade (0.42)	AccomplishMandatory-Grade (0.76)	AccomplishMandatory-Grade (0.8)	AccomplishMandatory-Grade (0.74)
	CourseViewPct (0.18)	AccomplishMandatory-PctGraded (0.07)	AccomplishMandatory (0.08)	AccomplishMandatory (0.1)
	AccomplishMandatory-PercentileGrade (0.13)	CourseViewPct (0.06)	CourseViewPct (0.04)	dsurseViewPct (0.05)
	AccomplishOptionalPct-Graded (0.09)	AccomplishMandatory-PercentileGrade (0.05)	QuizAttemptPct (0.01)	AccomplishMandatory-PercentileGrade (0.04)
	AccomplishMandatory-PctGraded (0.07)	AssignViewPct (0.01)	AssignViewPct (0.01)	AccomplishMandatory-PctGraded (0.01)

Table 7: Independent variables with the greatest Gini importance to classify at-risk, fail/pass and excellent students (respective mark thresholds 2.5, 5.0 and 8.5).

the average value for that feature in the whole dataset [45, 46]. Additionally, all the variables are z-score normalized (μ=0 and σ=1) to facilitate their comparison. We create four unsupervised classifiers, one for each moment of prediction (mark thresholds are not included, because they are targets of supervised classifiers).

Our dataset has 63 different features, representing a high dimensional space. Since cluster algorithms work with distance methods to identify the clusters, they are not accurate for high dimensional spaces [47]. This is the reason why it is better to reduce the number of dimensions before executing the clustering algorithm [48]. In this work, we tried the PCA (Principal Component Analysis) and FA (Feature Agglomeration) dimension reduction algorithms [49]. With PCA, we need 20 variables to explain 80% of the total variance of the dataset, whereas with FA we get to explain the same percentage of variance with just 4 variables. Therefore, we used FA (with ward criterion to compute distances) to reduce the dimensions of our dataset to four, improving the performance of the clustering algorithm and the interpretation of the generated clusters.

What follows is a description of the four variables found by the FA algorithm, considering the variables they group (details are presented in Appendix D):

- URL and assignment access (UAA). This variable mostly aggregates submission and view of URLs and assignments in the LMS.
- Mandatory and optional assignment evaluation (MOA). The majority type of variables aggregated by MOA is evaluation of assignments.
- Quiz access time (QAT). All the variables aggregated by QAT refer to quiz actions.
- Course and resource view (CIR). Every variable in CIR refers to course and resource views.

After reducing the dimensionality of the dataset, we apply the k-means algorithm to obtain the student clusters. However, k-means receives as a parameter the number of clusters to be created. Therefore, we use the gap statistic method to determine the optimal number of clusters to be created [50]. When running k-means, centroid were initialized with random values and we used the Euclidean distance.

ANOVA analyses are run to see if there are significant differences of the aggregated variables among the clusters found. Afterwards, we use Tukey's Honest Significant Difference (HSD) post-hoc tests to see exactly where those differences lie (i.e., to find out which specific aggregate variables are different one another)—Research Question 3. All these analyses were carried out for the four models (one for each moment of prediction).

5.2. Results

The gap statistic method identified six clusters to be created for the four moments of prediction. Figure 3 and Table 8 detail the distribution of the four aggregate variables in the six clusters created by the k-means algorithm, for each moment of prediction.

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
10%	N	12,162	1,691	837	756	4,198	5,616
	UAA	0.0284±0.256	-1.1638±0.608	0.2016±0.543	-0.1940±0.700	0.2811±0.530	0.0748±0.386
	MOA	-0.1506±0.328	0.3153±0.471	0.5061±0.729	0.6515±0.708	-0.0237±0.345	0.0859±0.412
	QAT	-0.0021±0.174	-0.0092±0.389	2.3969±0.873	-2.6879±1.194	0.0328±0.250	-0.0127±0.286
	CIR	0.0841±0.250	-0.6530±0.465	0.1862±0.807	-0.4885±0.590	1.1991±0.489	-0.8440±0.331
25%	N	7,088	1,897	8,600	3,353	1,315	3,007
	UAA	0.1469±0.379	-0.0436±0.545	0.0658±0.329	-0.7834±0.416	0.1455±0.504	0.3028±0.503
	MOA	-0.0010±0.389	0.4188±0.641	-0.1851±0.381	0.2586±0.435	0.2128±0.570	-0.1140±0.359
	QAT	0.0153±0.274	-1.6652±0.580	0.0066±0.212	0.0002±0.274	2.1413±0.799	0.0591±0.324
	CIR	-0.5945±0.269	-0.3984±0.474	0.2397±0.298	-0.5274±0.356	0.1075±0.799	1.5082±0.556
33%	N	4,134	7,438	2,460	1,452	7,266	2,510
	UAA	-0.6901±0.392	0.0702±0.331	-0.0471±0.507	0.1817±0.524	0.1800±0.387	0.3482±0.527
	MOA	0.2394±0.429	-0.2061±0.406	0.3432±0.607	0.1563±0.519	-0.0245±0.394	-0.1393±0.390
	QAT	0.0166±0.262	0.0095±0.215	-1.4321±0.488	2.0941±0.754	0.0180±0.266	0.0846±0.357
	CIR	-0.4895±0.329	0.3148±0.320	-0.3755±0.433	0.1150±0.799	-0.5191±0.263	1.6775±0.626
50%	N	3,055	8,051	1,520	1,627	4,957	6,050
	UAA	-0.0712±0.454	0.1980±0.382	0.4268±0.586	0.1841±0.518	0.1479±0.421	-0.5054±0.409
	MOA	0.2717±0.488	-0.2022±0.414	-0.2664±0.433	0.0859±0.537	-0.0985±0.401	0.2563±0.452
	QAT	-1.2791±0.422	0.0112±0.242	0.1999±0.559	1.9876±0.758	0.0109±0.265	0.0374±0.266
	CIR	-0.3566±0.371	-0.3543±0.293	2.1397±0.671	0.0930±0.734	0.6423±0.346	-0.4374±0.301

Table 8: Number of students per group (N), and variable average ± standard deviation values per cluster, for the four different moments of prediction (10%, 25%, 33% and 50%).

For all the variables, ANOVA found differences among the six groups (p-value<0.05), for every moment of prediction. Therefore, the Tukey's post-hoc test can be applied to analyze inter-cluster differences among the values of each variable [51]. Figure 4 shows the results of those tests, which could be summarized the following way:

- 10% of the course. UAA, MOA and CIR variables show significant differences for all the clusters (i.e., p-value<0.05 for all the Tukey's tests). For QAT, there is no difference among Clusters 1, 2 and 6.
- 25% of the course. MOA and CIR are significantly different for all the clusters. UAA values are not different between Clusters 1 and 5. The same happens in Clusters 1, 3 and 4 for QAT.
- 33% of the course. All the clusters show different values for MOA and CIR variables. For UAA, there is no significant difference between Clusters 4 and 5. Clusters 1, 2 and 5 show similar values for QAT.
- 50% of the course. There is no significant difference between: Clusters 2 and 4 for UAA, Clusters 1 and 6 for MOA, Clusters 2 and 5 for QAT and Clusters 1 and 2 for CIR.

5.3. Discussion

It is worth noting than each cluster set is different for each moment of prediction, since we used different datasets to create the clusters. Therefore, we should analyze the characteristics of each cluster for each moment of prediction and see if there are

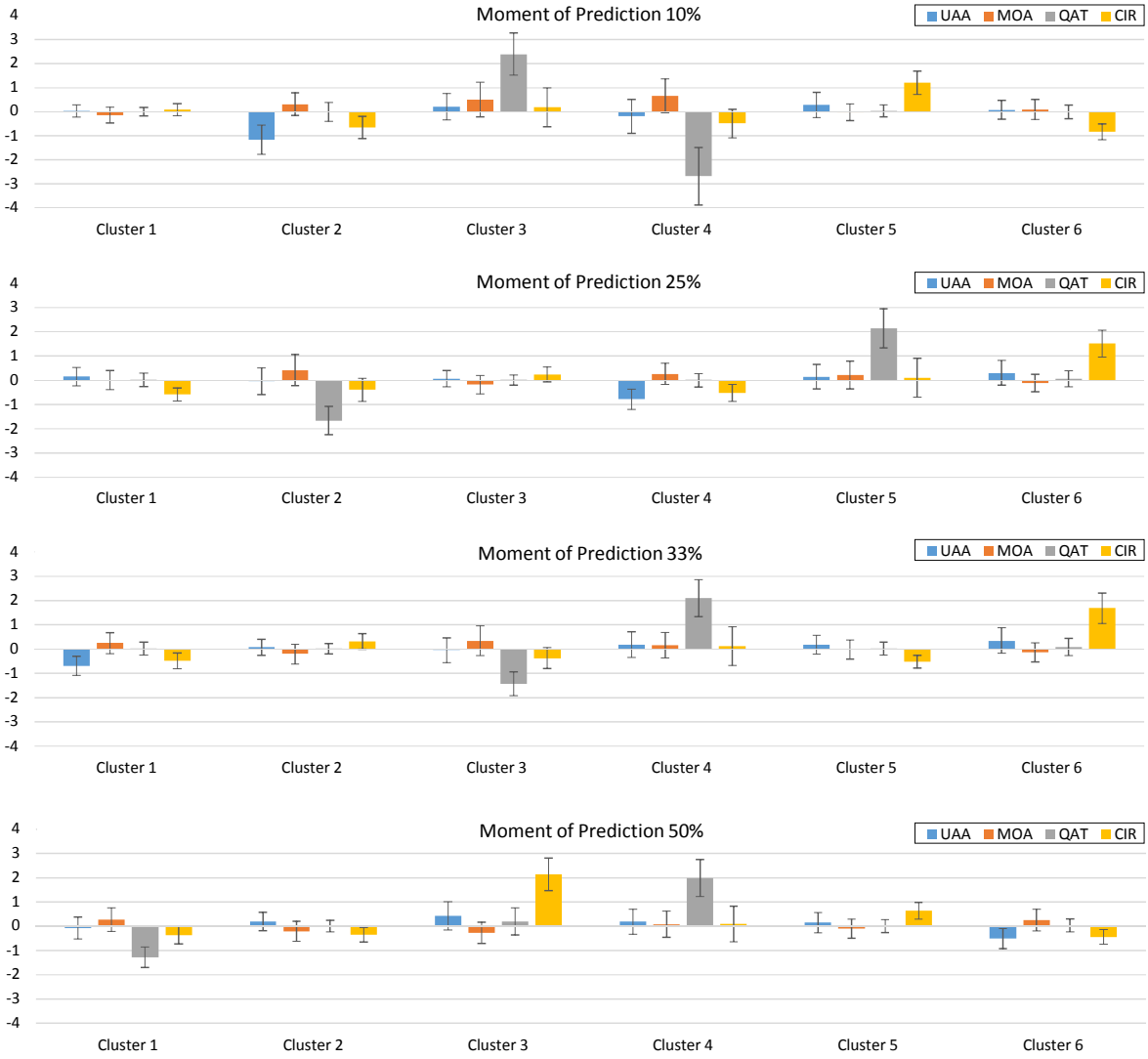

Figure 3: Average variable values per cluster for different moments of prediction; whiskers represent standard deviations.

similarities among different moments of prediction. To this aim, we analyze how the variable values determine cluster membership (Tukey's test), and then see if those cluster memberships are repeated in other moments of predictions. That is, we see if k-means generated similar clusters for different moments of prediction.

Since variables were normalized with μ=0 and σ=1 (Section 5.1), we consider a variable to be very low if it is below -1, low between -1 and -0.5, average between -0.5 and 0.5, high between 0.5 and 1, and very high if its value is greater than 1. By analyzing the values of the four aggregate variables, we can identify the following student clusters, valid for all the moments of prediction:

– QAT variable. As shown in Figure 4, there are two extreme values for QAT

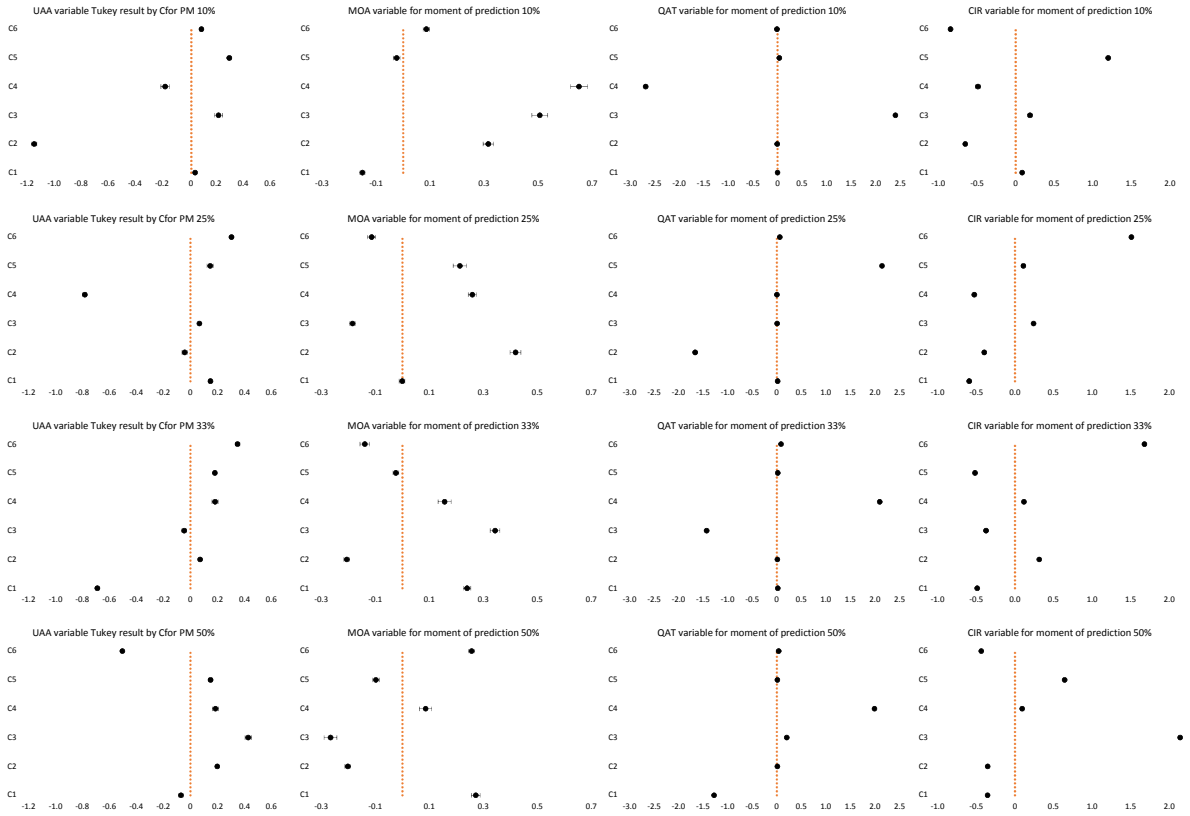

Figure 4: Inter-cluster variable values for different moments of prediction (whiskers represent 95% confidence intervals; dotted line represents mean values; Cn stands for Cluster n).

that determine two clusters, and this pattern is repeated in all the moments of prediction.

- $QAT{\downarrow}{\downarrow}$. A very low value of QAT define a group of students who answer quizzes significantly faster than the rest of the students. Table 9 shows how this condition holds for Clusters 4, 2, 3 and 1 in, respectively, the moments of prediction 10%, 25%, 33%, and 50%.
- $QAT{\uparrow}{\uparrow}$. On the other hand, there is one cluster of students with very low values of QAT for all the moments of prediction.

– $UAA{\downarrow}$. Low (and very low) values for UAA define one cluster in each moment of prediction (Figure 4). This cluster represents students that view external URLs and assignments in the LMS significantly earlier than the rest of the students.
– $CIR{\uparrow}{\uparrow}$. Figure 4 shows how the four moments of prediction have one cluster for very high values of CIR. The students in this group procrastinate when it comes to viewing course contents and the resources uploaded by lecturers.
– $MOA{\downarrow}CIR{\uparrow}$. There is a cluster for all the prediction moments with assignment evaluations below the average (MOA), and course and lecturer's resources views (CIR) above the mean. Students in this cluster cannot hold the previous

Moment of prediction	$QAT\downarrow\downarrow$	$QAT\uparrow\uparrow$	$UAA\downarrow$	$CIR\uparrow\uparrow$	$MOA\downarrow CIR\uparrow$	*Average*
10%	C4	C3	C2	C5	C1	C6
25%	C2	C5	C4	C6	C3	C1
33%	C3	C4	C1	C6	C2	C5
50%	C1	C4	C6	C3	C5	C2

Table 9: Correspondence between clusters for particular moment of predictions and clusters for any moment of prediction (Cn stands for Cluster n).

conditions—i.e., very low or high QAT, and (very) low UAA.
- *Average*. The last cluster is made up of the rest of the students, who have average values for most of the aggregate variables (Figure 4).

We can now answer **Research Question 3**. There are six student clusters, valid for all the prediction moments, that group students regarding their interaction with LMS, regardless particular course characteristics. Remarkably, the only variable related to evaluation (MOA) does not determine group membership by their own. Table 9 shows the correspondence between clusters found by the k-means algorithm for each particular moment of prediction and the six student groups identified for any moment of prediction.

5.4. Correlation between student clusters and students' performance

We now analyze whether there is any dependence between the clusters identified in the previous section and the students' performance continuous variable defined in Section 3.5. We run ANOVA tests, showing that such relationship exits (p-value$<$0.05). Then, we conducted Tukey's HSD post-hoc tests to see, for each pair of clusters, if there are significant differences in the performance of their students.

Figure 5 shows the results of Tukey's post-hoc tests (overlapping intervals mean no significant difference). The following relationships between student clusters and their performance are identified:

- The highest performance is obtained by those students who answer quizzes significantly faster than the rest ($QAT\downarrow\downarrow$)—differences are sometimes not significant with other groups.
- The students with the lowest marks are those that procrastinate when it comes to viewing lecturer resources and course contents, and with assignment evaluations below the average ($MOA\downarrow CIR\uparrow$)—differences are not significant with the next cluster for the 50% prediction moment.
- Students with very late view of lecturer resources and course contents are the ones with the second worst performance ($CIR\uparrow\uparrow$)—together with $MOA\downarrow CIR\uparrow$ in the 50% of course length.

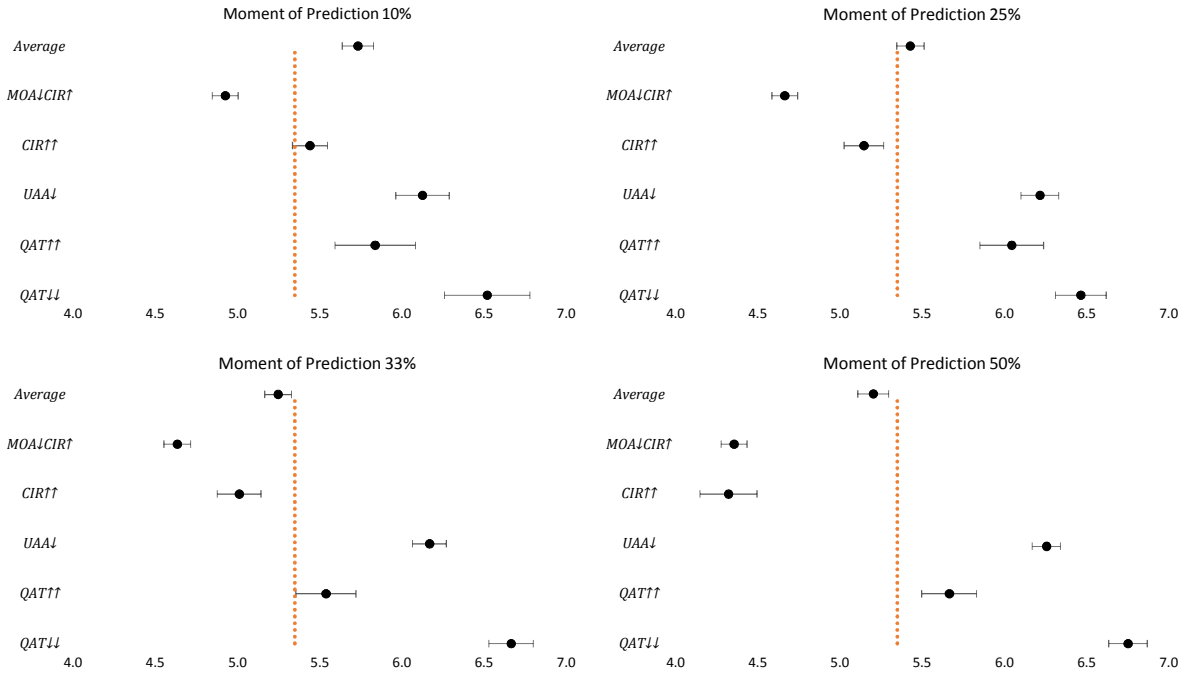

Figure 5: Students' performance for each cluster (whiskers represent 95% confidence intervals; dotted line represents mean values). Overlapping intervals represent non-significant difference.

- $UAA\downarrow$ is the cluster with the second best performance: students that view external URLs and assignments in the LMS significantly earlier than the rest of the students.
- The average and $QAT\uparrow\uparrow$ clusters get average performance, and sometimes there are not significant differences between both groups.

The response to **Research Question 4** is that four (out of the six) clusters identified for any moment of prediction are correlated with the performance of students belonging to such groups. This correlation holds for heterogeneous courses with different methodologies, knowledge areas, learning formats and duration.

6. Conclusions

Students' performance in solving the LMS assignments can be predicted at the early stages of the course by using the log files generated by the LMS. Moreover, we find six student-LMS interaction patterns that are repeated in all the initial stages of the course, and four of them are correlated with the performance of the students belonging to them. An important characteristic of our work is that the predictive and clustering models were created with multiple heterogeneous courses, so they do not depend on a particular learning methodology, discipline, format or duration.

At 10% of the course, prediction accuracies are better than the most-frequent-class classification for at-risk and fail students. For 25% of the course on, excellent students are also detected. Accuracy grows up to 93% when half of the course has been delivered. MLP and DT are the algorithms with the highest accuracy, which can be selected depending on the particular costs of misclassification.

The log entries related with the actions performed by students show a strong influence on students' performance in the very early stages of the course. Later, evaluation tasks become more influential. Procrastination when it comes to viewing the resources uploaded by lecturers has an important influence on at-risk and fail students. Course view actions at the beginning of the course strongly influence the detection of excellent students.

LMS log files generated at the beginning of the course are also useful to detect student clusters regarding their interaction with LMSs. Six clusters are identified for all the early moments of prediction analyzed. Four of those clusters have strong correlation with students' performance, and they do not depend on the particular features of a course. Patterns such as early answer to quizzes, and promptly LMS resource and assignment view are related to high performance; whereas procrastination when it comes to viewing resources and course contents is associated to low performance.

We plan to include the predictive models in a customized Moodle version to be used by the University of Oviedo. In this way, lecturers could receive messages about students' performance predicted at the early stages of the courses. We also would like to enrich the predictive models with other external sources of information such as demographic data [52] and information taken from student-LMS interaction in former courses. Another line of work could be analyzing if the order of certain actions would be related with students' performance, where LSTM recurrent networks could be used [53].

The dataset used to create the models, the original collection of log files, the reusable Python models, the source code used to build and evaluate the models, data obtained and the hyper-parameters used can be freely downloaded from

`https://github.com/moisesriestra/moodle-early-performance-prediction`

Acknowledgments

This work has been partially funded by the Spanish Department of Science, Innovation and Universities: project RTI2018-099235-B-I00. The authors have also received funds from the University of Oviedo through its support to official research groups (GR-2011-0040).

Appendix A. Variables related to students' actions

The following enumeration depicts the variables describing student actions, computed from the LMS log files. All the variables are relative to the moment of prediction. That is, no data after that moment are considered to compute the variables.

- `CourseViewPct`: percentage of accesses to the course, relative to the total accesses all the students in that course.
- `CourseViewTime{1,2,3,4,5}`: first to fifth time the student accessed the course, relative to its duration (i.e., percentage of the course length).
- `CourseViewTimePct`: geometric mean of the five `CourseViewTime{1,2,3,4,5}` values.
- `ResourceViewPct`: percentage of accesses to lecturer resources, relative to the total accesses to lecturer resources for all the students in that course.
- `ResourceViewTime{1,2,3,4,5}`: first to fifth time the student view a lecturer resource, relative to the course duration (percentage).
- `ResourceViewTimePct`: geometric mean of the five `ResourceViewTime{1,2,3,4,5}` values.
- `ResourceViewUniquePct`: percentage of lecturer resources viewed by the student, relative to the total number of lecturer resources in the course.
- `UrlViewPct`: percentage of URL views, relative to the total URL views for all the students in that course.
- `UrlViewTime{1,2,3,4,5}`: first to fifth time the student viewed a URL, relative to the course duration (percentage).
- `UrlViewTimePct`: geometric mean of the five `UrlViewTime{1,2,3,4,5}` values.
- `UrlViewUniquePct`: percentage of URLs viewed by the student, relative to the total number of URLs in the course.
- `AssignViewPct`: percentage of assignment views, relative to the total assignment views for all the students in that course.
- `AssignViewTime{1,2,3}`: first to third time the student viewed an assignment, relative to the course duration (percentage).
- `AssignViewTimePct`: geometric mean of the three `AssignViewTime{1,2,3}` values.
- `AssignViewUniquePct`: percentage of assignments viewed by the student, relative to the total number of assignments in the course.
- `QuizViewPct`: percentage of quiz views, relative to the total quiz views for all the students in that course.
- `QuizViewTime{1,2,3}`: first to third time the student viewed a quiz, relative to the course duration (percentage).
- `QuizViewTimePct`: geometric mean of the three `QuizViewTime{1,2,3}` values.
- `QuizViewUniquePct`: percentage of quizzes viewed by the student, relative to the total number of quizzes in the course.
- `AssignSubmitPct`: percentage of assignment submissions, relative to the total assignment submissions for all the students in that course.

- `AssignSubmitTime{1,2,3}`: first to third time the student submitted an assignment, relative to the course duration (percentage).
- `AssignSubmitTimePct`: geometric mean of the three `AssignSubmitTime{1,2,3}` values.
- `AssignSubmitUniquePct`: percentage of assignments submitted by the student, relative to the total number of assignments in the course.
- `QuizAttemptPct`: a quiz attempt is when the student starts a quiz, so this variable measures the percentage of quiz attempts, relative to the total quiz attempts for all the students in that course.
- `QuizAttemptTime{1,2,3}`: first to third time the student started a quiz, relative to the course duration.
- `QuizAttemptTimePct`: geometric mean of the three `QuizAttemptTime{1,2,3}` values.
- `QuizAttemptUniquePct`: percentage of quizzes started by the student, relative to the total number of quizzes in the course.
- `QuizCloseAttemptPct`: percentage of quiz submissions, relative to the total quiz submissions for all the students in that course.
- `QuizCloseAttemptTime{1,2,3}`: first to third time the student submitted a quiz, relative to the course duration.
- `QuizCloseAttemptTimePct`: geometric mean of the three `QuizCloseAttemptTime{1,2,3}` values.
- `QuizCloseAttemptUniquePct`: percentage of quizzes submitted by the student, relative to the total number of quizzes in the course.
- `ForumViewForumPct`: percentage of forum views, relative to the total number of forum views for all the students in the course.
- `ForumViewDiscussionPct`: percentage of discussion views, relative to the total number of discussion views for all the students in the course.

Appendix B. Variables related to assignment tasks

What follows is the independent variable list related to assignment tasks before the moment of prediction (no data after that moment are included in the model).

- `AccomplishMandatory`: indicates whether any mandatory assignment has been submitted.
- `AccomplishMandatoryGrade`: average mark of mandatory assignments, considering non-submitted assignments as zero mark.
- `AccomplishMandatoryPctGraded`: percentage of mandatory assignments submitted until prediction time, relative to the total number of mandatory assignments in the course.
- `AccomplishMandatoryPercentileGrade`: student's average mark in the mandatory assignments, relative to (percentile) the rest of the students in the same course.
- `AccomplishOptional`: indicates whether any optional assignment has been submitted.

- `AccomplishOptionalGrade`: average mark of optional assignments, considering non-submitted assignments as zero mark.
- `AccomplishOptionalPctGraded`: percentage of optional assignments delivered by the student until the moment of prediction.
- `AccomplishOptionalPercentileGrade`: student's average mark in the optional assignments, relative to (percentile) the rest of the students in the same course.

Appendix C. Hyper-parameters

We now detail the hyper-parameters selected for each predictive model, using the hyper-parameter tuning method described in Section 4.1. For those hyper-parameters not listed, we took the default value provided by scikit-learn.

- Decision tree:
  - Mark threshold 2.5:
    - 10%, 25%, 33% and 50%: `splitter` = `random`, `presort` = `False`, `max_features` = `None`, `max_depth` = 10, `criterion` = `gini`, `class_weight` = `None`.
  - Mark threshold 5.0:
    - 10%: `splitter` = `best`, `presort` = `False`, `max_features` = `sqrt`, `max_depth` = 10, `criterion` = `gini`, `class_weight` = `None`.
    - 25% and 50%: `splitter` = `random`, `presort` = `True`, `max_features` = `None`, `max_depth` = 10, `criterion` = `gini`, `class_weight` = `balanced`.
    - 33%: `splitter` = `random`, `presort` = `True`, `max_features` = `None`, `max_depth` = 10, `criterion` = `entropy`, `class_weight` = `balanced`.
  - Mark threshold 8.5:
    - 10%, 25%, 33% and 50%: `splitter` = `best`, `presort` = `True`, `max_features` = `None`, `max_depth` = 5, `criterion` = `entropy`, `class_weight` = `None`.
- Naïve Bayes:
  - Mark threshold 2.5, 5.0 and 8.5:
    - 10%, 25%, 33% and 50%: `var_smoothing` = 1e-09.
- Logistic regression:
  - Mark thresholds 2.5:
    - 10% and 25%: `tol` = 0.0001, `solver` = `liblinear`, `penalty` = `l1`, `max_iter` = 200.
    - 33%: `tol` = 0.001, `solver` = `liblinear`, `penalty` = `l2`, `max_iter` = 50.
    - 50%: `tol` = 0.01, `solver` = `liblinear`, `penalty` = `l2`, `max_iter` = 100.
  - Mark threshold 5.0:
    - 10%: `tol` = 1e-05, `solver` = `liblinear`, `penalty` = `l2`, `max_iter` = 200.
    - 25% and 33%: `tol` = 0.001, `solver` = `liblinear`, `penalty` = `l1`, `max_iter` = 100.

    - 50%: `tol` = 1e-05, `solver` = `liblinear`, `penalty` = `l1`, `max_iter` = 200.
  - Mark threshold 8.5:
    - 10%: `tol` = 1e-05, `solver` = `liblinear`, `penalty` = `l2`, `max_iter` = 50.
    - 25%: `tol` = 0.01, `solver` = `liblinear`, `penalty` = `l1`, `max_iter` = 100.
    - 33%: `tol` = 0.0001, `solver` = `liblinear`, `penalty` = `l1`, `max_iter` = 100.
    - 50%: `tol` = 0.01, `solver` = `liblinear`, `penalty` = `l2`, `max_iter` = 100.
- Multi-layer perceptron:
  - Mark threshold 2.5:
    - 10%, 25%, 33% and 50%: `solver` = `lbfgs`, `learning_rate` = `constant`, `hidden_layer_sizes` = 20, `alpha` = 0.1, `activation` = `tanh`.
  - Mark threshold 5.0:
    - 10% and 50%: `solver` = `lbfgs`, `learning_rate` = `adaptive`, `hidden_layer_sizes` = 20, `alpha` = 0.01, `activation` = `relu`.
    - 25%: `solver` = `adam`, `learning_rate` = `constant`, `hidden_layer_sizes` = 20, `alpha` = 0.001, `activation` = `relu`.
    - 33%: `solver` = `lbfgs`, `learning_rate` = `invscaling`, `hidden_layer_sizes` = 20, `alpha` = 0.1, `activation` = `relu`.
  - Mark threshold 8.5:
    - 10%: `solver` = `adam`, `learning_rate` = `adaptive`, `hidden_layer_sizes` = 20, `alpha` = 0.001, `activation` = `relu`.
    - 25%: `solver` = `lbfgs`, `learning_rate` = `adaptive`, `hidden_layer_sizes` = 20, `alpha` = 1, `activation` = `relu`.
    - 33% and 50%: `solver` = `lbfgs`, `learning_rate` = `invscaling`, `hidden_layer_sizes` = 20, `alpha` = 1, `activation` = `relu`.
- Support vector machine:
  - Mark thresholds 2.5, 5.0 and 8.5.
    - 10%, 25%, 33% and 50%: `tol` = 0.01, `probability` = `True`, `kernel` = `rbf`, `gamma` = `scale`, `cache_size` = 4096, `C` = 1.

Appendix D. Aggregate variables

Table D.1 shows the aggregate variables obtained by the feature agglomeration algorithm.

References

[1] R. Li, J. T. Singh, J. Bunk, Technology Tools in Distance Education: A Review of Faculty Adoption, in: EdMedia+ Innovate Learning, Association for the Advancement of Computing in Education (AACE), 2018, pp. 1982–1987.

[2] R. K. Ellis, Field guide to learning management systems, ASTD learning circuits (2009) 1–8.

URL and assignment access (UAA)	Mandatory and optional assignment evaluation (MOA)	Quiz access time (QAT)	Course and resource view (CIR)
UrlViewTime1	AccomplishMandatory	QuizViewTime1	CourseViewtime1
UrlViewTime2	AccomplishMandatoryGrade	QuizViewTime2	CourseViewtime2
UrlViewTime3	AccomplishMandatoryPctGraded	QuizViewTime3	CourseViewtime3
UrlViewTime4	AccomplishMandatoryPercentileGrade	QuizViewTimePct	CourseViewtime4
UrlViewTime5	AccomplishOptional	QuizAttemptTime1	CourseViewtime5
UrlViewTimePct	AccomplishOptional	QuizAttemptTime1	CourseViewtime5
AssignViewTime1	AccomplishOptionalGrade	QuizAttemptTime2	CourseViewtimePct
AssignViewTime2	AccomplishOptionalPctGraded	QuizAttemptTime3	ResourceViewTime1
AssignViewTime3	AccomplishOptionalPercentileGrade	QuizAttemptTimePct	ResourceViewTime2
AssignViewTimePct	CourseViewPct	QuizCloseAttemptTime1	ResourceViewTime3
AssignSubmitTime1	ResourceViewPct	QuizCloseAttemptTime2	ResourceViewTime4
AssignSubmitTime2	ResourceViewUniquePct	QuizCloseAttemptTime3	ResourceViewTime5
AssignSubmitTime3	UrlViewPct	QuizCloseAttemptTimePct	ResourceViewTimePct
AssignSubmitTimePct	UrlViewUniquePct		
	AssignViewPct		
	AssignViewUniquePct		
	QuizViewPct		
	QuizViewUniquePct		
	AssignSubmitPct		
	AssignSubmitUniquePct		
	QuizAttemptPct		
	QuizAttemptUniquePct		
	QuizCloseUniquePct		
	QuizCloseAttemptUniquePct		
	ForumViewForumPct		
	ForumViewDiscussionPct		

Table D.1: Aggregate variables.

[3] R. Conijn, C. Snijders, A. Kleingeld, U. Matzat, Predicting student performance from LMS data: A comparison of 17 blended courses using Moodle LMS, IEEE Transactions on Learning Technologies 10 (1) (2017) 17–29.

[4] M. Llamas, M. Caeiro, M. Castro, I. Plaza, E. Tovar, Use of LMS functionalities in engineering education, in: 2011 Frontiers in Education Conference (FIE), IEEE, 2011, pp. S1G–1.

[5] B. W. Tuckman, Relations of academic procrastination, rationalizations, and performance in a web course with deadlines, Psychological reports 96 (3) (2005) 1015–1021.

[6] R. Cerezo, M. Sánchez-Santillán, M. P. Paule-Ruiz, J. C. Núñez, Students' LMS interaction patterns and their relationship with achievement: A case study in higher education, Computers & Education 96 (2016) 42–54.

[7] N. Kadoić, D. Oreški, Analysis of student behavior and success based on logs in Moodle, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2018, pp. 0654–0659.

[8] C. R. Henrie, R. Bodily, R. Larsen, C. R. Graham, Exploring the potential of LMS log data as a proxy measure of student engagement, Journal of Computing in Higher Education 30 (2) (2018) 344–362.

[9] L. P. Macfadyen, S. Dawson, Mining LMS data to develop an "early warning system" for educators: A proof of concept, Computers & education 54 (2) (2010) 588–599.

[10] Y.-H. Hu, C.-L. Lo, S.-P. Shih, Developing early warning systems to predict students' online learning performance, Computers in Human Behavior 36 (2014) 469–478.

[11] P. Brusilovsky, Methods and techniques of adaptive hypermedia, User Modeling and User-Adapted Interaction 16 (1996) 87–129.

[12] M. del Puerto Paule Ruiz, M. J. F. Díaz, F. Ortin, J. R. P. Pérez, Adaptation in current e-learning systems, Computer Standards & Interfaces 30 (1) (2008) 62 – 70.

[13] N. Dabbagh, A. Kitsantas, Using Web-based Pedagogical Tools as Scaffolds for Self-regulated Learning, Instructional Science 33 (5) (2005) 513–540.

[14] T. M. Kelly, N. Nanjiani, The Business Case for E-Learning, Cisco Press, 2004.

[15] E. B. Costa, B. Fonseca, M. A. Santana, F. F. de Araújo, J. Rego, Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses, Computers in Human Behavior 73 (2017) 247–256.

[16] N. Tomasevic, N. Gvozdenovic, S. Vranes, An overview and comparison of supervised data mining techniques for student exam performance prediction, Computers & education 143 (103676) (2020) 1–18.

[17] G. Cobo, D. García-Solórzano, J. A. Morán, E. Santamaría, C. Monzo, J. Melenchón, Using agglomerative hierarchical clustering to model learner participation profiles in online discussion forums, in: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, 2012, pp. 248–251.

[18] L. Gerritsen, Predicting student performance with Neural Networks, Ph.D. thesis, Doctoral dissertation, Tilburg University (2017).

[19] D. Gašević, S. Dawson, T. Rogers, D. Gasevic, Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success, The Internet and Higher Education 28 (2016) 68–84.

[20] C. Romero, P. G. Espejo, A. Zafra, J. R. Romero, S. Ventura, Web usage mining for predicting final marks of students that use Moodle courses, Computer Applications in Engineering Education 21 (1) (2013) 135–146.

[21] J. López-Zambrano, J. A. Lara, C. Romero, Towards Portability of Models for Predicting Students' Final Performance in University Courses Starting from Moodle Logs, Applied Sciences 10 (1) (2020) 354.

[22] C. Romero, M.-I. López, J.-M. Luna, S. Ventura, Predicting students' final performance from participation in on-line discussion forums, Computers & Education 68 (2013) 458–472.

[23] M. F. Marbouti, H. A. Diefes-Dux, Building course-specific regression-based models to identify at-risk students, in: The american society for engineering educators annual conference, 2015, pp. 1–10.

[24] A. Jokhan, B. Sharma, S. Singh, Early warning system as a predictor for student performance in higher education blended courses, Studies in Higher Education 44 (11) (2019) 1900–1911.

[25] D. Ljubobratović, M. Matetić, Using LMS Activity Logs to Predict Student Failure with Random Forest Algorithm, The Future of Information Sciences (2019) 113.

[26] L. Talavera, E. Gaudioso, Mining student data to characterize similar behavior groups in unstructured collaboration spaces, in: Workshop on artificial intelligence in CSCL. 16th European conference on artificial intelligence, 2004, pp. 17–23.

[27] J.-L. Hung, K. Zhang, Revealing online learning behaviors and activity patterns and making predictions with data mining techniques in online teaching, MERLOT Journal of Online Learning and Teaching (Dec. 2008).

[28] D. Hooshyar, M. Pedaste, Y. Yang, Mining Educational Data to Predict Students' Performance through Procrastination Behavior, Entropy 22 (1) (2020) 12.

[29] M. I. Lopez, J. M. Luna, C. Romero, S. Ventura, Classification via clustering for predicting final marks based on student participation in forums., International Educational Data Mining Society (Jun. 2012).

[30] Y. Park, J. H. Yu, I.-H. Jo, Clustering blended learning courses by online behavior data: A case study in a Korean higher education institute, The Internet and Higher Education 29 (2016) 1–11.

[31] J. Cole, H. Foster, Using Moodle: Teaching with the popular open source course management system, 2nd Edition, O'Reilly Media, Inc., 2007.

[32] J. W. Tukey, Exploratory Data Analysis, Addison-Wesley, 1977.

[33] S. Garcia, J. Luengo, J. A. Sáez, V. Lopez, F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, IEEE Transactions on Knowledge and Data Engineering 25 (4) (2012) 734–750.

[34] N. Rout, D. Mishra, M. K. Mallick, Handling imbalanced data: a survey, in: International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications, Springer, 2018, pp. 431–443.

[35] T. O. Kvålseth, Cautionary note about R^2, The American Statistician 39 (4) (1985) 279–285.

[36] Z. Reitermanová, Data splitting, in: Proceedings of the 19th Annual Conference of Doctoral Student, WDS, 2010, pp. 31–26.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[38] B. K. Singh, K. Verma, A. S. Thoke, Investigations on impact of feature normalization techniques on classifier's performance in breast tumor classification, International Journal of Computer Applications 116 (19) (2015).

[39] M. Riestra Gonzalez, Moodle early performance prediction, `https://github.com/moisesriestra/moodle-early-performance-prediction` (2020).

[40] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, in: Proceedings of the 23rd International Conference on Machine Learning, ICML'06, Association for Computing Machinery, New York, NY, USA, 2006, p. 233–240.

[41] C. J. V. Rijsbergen, Information Retrieval, 2nd Edition, Butterworth-Heinemann, USA, 1979.

[42] F. Ortin, O. Rodriguez-Prieto, N. Pascual, M. Garcia, Heterogeneous tree structure classification to label Java programmers according to their expertise level, Future Generation Computer Systems 105 (2020) 380–394.

[43] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, F. A. Hamprecht, A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data, BMC bioinformatics 10 (1) (2009) 213.

[44] S. Nembrini, I. R. König, M. N. Wright, The revival of the Gini importance?, Bioinformatics 34 (21) (2018) 3711–3718.

[45] G. L. Schlomer, S. Bauman, N. A. Card, Best practices for missing data management in counseling psychology., Journal of Counseling psychology 57 (1) (2010) 1.

[46] S. M. Fox-Wasylyshyn, M. M. El-Masri, Handling missing data in self-report measures, Research in nursing & health 28 (6) (2005) 488–495.

[47] N. Tomašev, M. Radovanovič, D. Mladenič, M. Ivanovič, The role of hubness in clustering high-dimensional data, in: Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I, PAKDD'11, Springer-Verlag, Berlin, Heidelberg, 2011, p. 183–195.

[48] P. Mitra, C. A. Murthy, S. K. Pal, Unsupervised feature selection using feature similarity, IEEE transactions on pattern analysis and machine intelligence 24 (3) (2002) 301–312.

[49] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1151–1157.

[50] S. Trivedi, Z. A. Pardos, N. T. Heffernan, Clustering students to generate an ensemble to improve standard test score predictions, in: International Conference on Artificial Intelligence in Education, Springer, 2011, pp. 377–384.

[51] H. Abdi, L. J. Williams, Tukey's honestly significant difference (HSD) test, in: Encyclopedia of Research Design, SAGE, 2010, pp. 1–5.

[52] J. Kuzilek, M. Hlosta, D. Herrmannova, Z. Zdrahal, A. Wolff, OU Analyse: analysing at-risk students at The Open University, Learning Analytics Review (2015) 1–16.

[53] F. Okubo, T. Yamashita, A. Shimada, H. Ogata, A neural network approach for students' performance prediction, in: Proceedings of the Seventh International Learning Analytics & Knowledge Conference, 2017, pp. 598–599.