Predictive models of academic success: a case study with version control systems

Ángel Manuel Guerrero-Higueras
Research Institute of Applied Sciences in Cybersecurity.
University of León.
León, Spain.
am.guerrero@unileon.es

Noemí DeCastro-García
Deparment of mathematics.
Universidad de León.
University of León.
León, Spain.
ainhoa.alvarez@ehu.eus

Vicente Matellán
Supercomputación Castilla y León.
León, Spain.
vicente.matellan@scayle.es

Miguel Á. Conde
Robotics Group - University of León
León, Spain
mcong@unileon.es

ABSTRACT

Version Control Systems are commonly used by Information and Communication Technology professionals. These systems allow monitoring programmers activity working in a project. Thus, Version Control Systems are also used by educational institutions. The aim of this work is to demonstrate that the academic success of students may be predicted by monitoring their interaction with a Version Control System. In order to do so, we have built a Machine Learning model to predict student results in a specific practical assignment of the *Ampliación de Sistemas Operativos* subject, from the second course of the degree in Computer Science of the University of León, through their interaction with a Git repository. To build the model, several classifiers and predictors have been evaluated. In order to do so, we have developed Model Evaluator (MoEv), a tool to evaluate different Machine Learning models in order to get the most suitable for a specific problem. Prior to the model development, a feature selection of the input data is done. The resulting model has been trained using results from 2016–2017 course and later validated using results from 2017–2018 course. Results conclude that the model predict students' success with a success high percentage.

TEEM'18, October 24-26, 2018, Salamanca, Spain
© 2018 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6518-5/18/10...$15.00
http://dx.doi.org/10.1145/3284179.3284235

CCS CONCEPTS

• **Applied computing** → E-learning; **Applied computing** → Education

KEYWORDS

competency-based learning, visual learning analytics, feedback

ACM Reference format:
Ángel Manuel Guerrero-Higueras, Noemí Decastro-García, Vicente Matellan and Miguel Ángel Conde. 2018. Predictive models of academic success: a case study with version control systems. In *Proceedings of the 6th International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2018) (Salamanca, Spain, October 24-26, 2018), F. J. García-Peñalvo Ed. ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/3284179.3284235

1 INTRODUCTION

The study of learning analytics has been defined as the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs (see [26]). This field, together with educational data mining, has a high potential for understanding and optimizing the learning and teaching processes. For instance, the analysis of data from student institutional systems (SISs), or the interaction of the students with the Learning Management Systems (LMSs), provides the teachers a powerful way to identify patterns that can be used to predict the achievements of learning outcomes, propose insights regarding teaching interventions, or make decisions about the adequacy of a resource.

One of the most studied issues in learning analytics is obtaining a predictive model for the student's grades. In particular, it is very common to try to obtain the risk (or the probability) of that a student passes or fails a course [25]. Predictive analytics has an important consequence in the education because they let us identify the students at-risk situation. This information allows the teachers to carry out proactive strategies in order to contribute to have a high-quality educational system. The research in this topic is focused on obtaining specific predictive models that only have high accuracy over concrete scenarios depending on the field, the size of the samples, or the data source, see [13]. This limitation is something expected due to the models are computed by learning algorithms that require quality data for each case. This fact makes difficult to get a general overview of the effect of different features on the academic success, and also getting a general and easy-to-use tool.

The main goal of this work is to present an evaluation tool, Model Evaluator (MoEv), which automatically study different parametric and non-parametric learning algorithms in order to choose the best one predictive model for the classification of students in fail or pass. In this way, the tool could be applied to different courses, fields or learning situations. The above tool is based on the idea of [16], but it includes a very essential phase for the educational use: the automatized selection of the discriminant features. If the challenge is obtaining a general tool, it is necessary that the data source could be of a different nature, and then the software will determine what is the important information for the model. The most common types of the data source in the predictive educational models are the data stores in SISs, see [20], the trace data recorded by LMSs and other learning environments, [1], or hybrid data sources composed by the ones described above [2].

On the other hand, the emergence of Information and Communication Technologies (ICTs) have changed the landscape of the teaching and learning processes. In some fields, as engineering or computer science, it is very usual that teachers need to employ advanced tools and applications in their courses with the aim to give to the students a meaningful learning experience. In software engineering, the management of changes in the components of a software product is known as version control [11]. It is called version, revision or edition, to the state of the product at a given time. Version management can be done manually, although it is advisable to use some tool to facilitate this task. These tools are known as Version Control Systems (VCSs) [27]. Among the most popular there are the following: CVS, Subversion [23] or Git [28]. A VCS must provide, at least, the following features: storage for the different elements to be managed (source code, images, documentation, etc.); edition the stored elements (creation, deletion, modification, naming, etc.); and registration and labelling of all actions carried out, of so that they allow an element to be returned to a state previous.

A highly demanded skill in computer science professional profiles is the use of VCSs. For this reason, another goal of this work is to answer the following research questions:

Question–1: Are there features that we can extract from the students' interactions with this type of systems that are related with the academic success?

Question–2: Can we build a model that allows predicting students' success at a practical assignment, by monitoring their use of a VCS?

In order to obtain data that allow us to answer the above questions, we have carried out a case study in a course about operating systems of the second year of the degree in Computer Science at the University of León. The preliminary results presented at [18], concluded that analyzing the students' activity at VCSs allows to predict their results by using tree-based algorithms. However, results were different depending on the chosen features. As at [18], in this work we also follow the method proposed at [16] to select the most suitable model, but in this work a feature analysis is done prior to the model development. To build the model, a training dataset have been used. In addition, in order to ensure an optimal generalization, we validate the selected model by using a second dataset.

The rest of the paper is organized as follows: Section 2 describes the empirical evaluation process of the classification algorithms. The experimental environment, materials, and methods used are also described at section 2. Section 3 summarizes the results of the evaluation. The discussion of the results is developed in Section 4. Finally, section 5 presents the conclusions and future lines of research.

2 MATERIALS AND METHODS

This section describes all the used elements to build and evaluate the model for predicting student academic success from their interaction with VCSs. Among these elements are included the following: a practical assignment that *Ampliación de Sistemas Operativos* (ASSOO) students need to pass, a VCS platform to gather data, and a tool to get an optimized model.

2.1 Input data

The ASSOO course belongs to the second year of the degree in Computer Science of the University of León. The course broadens knowledge about operating systems. In particular, it addresses the internal functioning of storage management, both volatile (memory management) and non-volatile (file management). Issues related to security in operating systems are also addressed.

The main practical assignment that ASSOO students need to pass consists of implementing an inode-based file system called *Ampliación de Sistemas Operativos* File System (ASSOOFS). According to the proposed specification, this file system must work on computers that run the Linux operating system. Therefore, students have to implement a module for the Linux kernel [5] that supports, at least, the following operations: mounting of devices formatted with this file system; creation, reading, and writing of regular files; creation of new directories and the visualization of the content of existing directories.

ASSOOFS assignment must be fulfilled individually. Students are encouraged to use a VCS during the completion of the task, specifically a Git repository. Git follows a distributed scheme, and contrary to other systems that follow the client-server models, each copy of the repository includes the story complete of all the changes made [8]. In order to provide some organizing capabilities and private repositories for students, the GitHub Classroom platform was used [15]. GitHub is a web-based hosting service for software development projects that utilize the Git revision control system. In addition, GitHub Classroom allows assigning tasks to students, or groups of students, framed in the same centralized organization: the ASSOO course in our case.

Regarding the collected data obtained from the practical assignment described above, the following variables coming from students' activity on their repositories have been considered:

- Anonymized student identifier (*id*). It is just considered in order to differentiate between students.
- Number of commit operations carried out by the student (*commits*).
- Number of days where there is at least one commit operation (*days*).
- Average number of commit operations per day (*commits/day*).
- Number of lines of code added during the assignment completion (additions).
- Number of lines deleted during the assignment completion (*deletions*).
- The GitHub platform calls Issue to the problems detected and documented in a software project for a later fix. The *issues* variable represents the number of issues opened by students.
- The *closed* variable represents the number of issues closed.

In addition to the data obtained from the GitHub Classroom platform, we have also considered the grade of the students on a proof carried out to control the authorship of the code in their repositories. This authorship proof allows us verifying that the students really worked in the content of their repository. On the proof, students are requested to introduce a minor change on their code in a given period of time. For instance, maintaining a modification date in the inodes. The requested change is trivial for students who really understand the code of their repository. The proof is carried out at class and no interaction is allowed with neither other students nor the outside. The authorship proof has two possible results: "1" if the student passed the proof; "0" otherwise.

Our target is a binary variable with two possible values: "AP", for those students who will finish the practical assignment successfully; and "SS", for those who not.

The data obtained from the ASSOO course come from two groups of students:

1. 46 students who tried the ASSOOFS assignment from 2016–2017 ASSOO course. The data includes the features mentioned above. It is a balanced sample with 21 students labelled with "AP", and 25 students with the label "SS". These data are used to train and test the prediction model. We will refer to them as *training dataset*.
2. 40 students who tried the ASSOOFS assignment from 2017–2018 ASSOO course, whose number of "AP" and "SS" is 21and 19, respectively. These data are used to validate the prediction model. We will refer to them as *validation dataset*.

2.2 Model design

To get the best predictive model we have built the MoEv tool following the method proposed at [16] but adding a feature selection phase to the process. To build MoEv, the Scikit-learn library has been used [22]. Figure 1 show the steps of the methodology used.

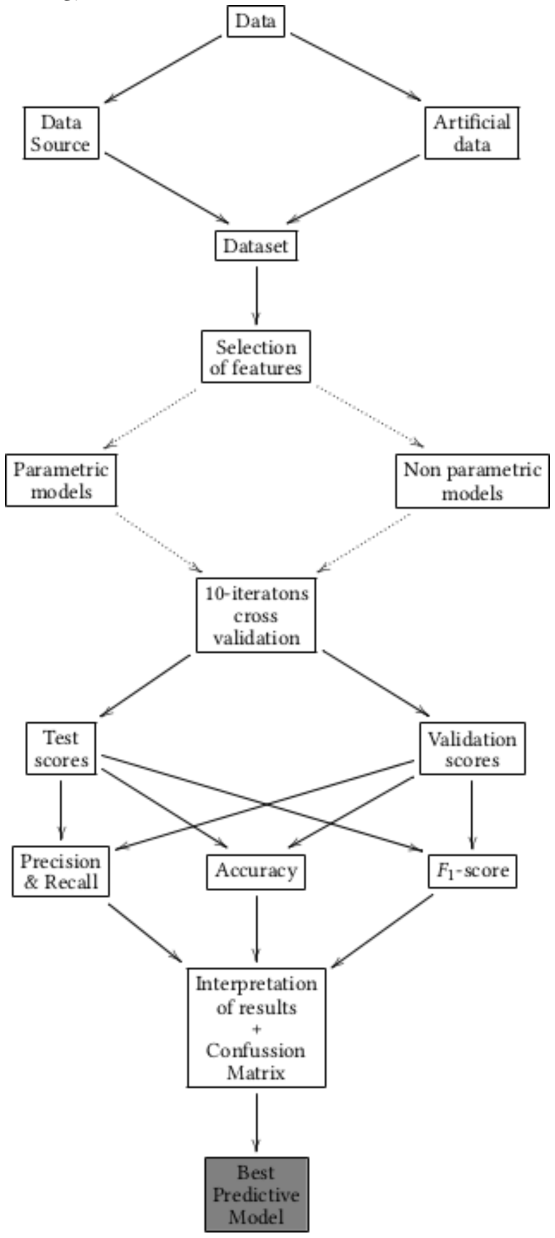

Figure 1: MoEv design

We start from a database that contains two different type of data. On the one hand, we have variables that directly come from a data source as SISs or LMSs (raw data). In our case, we manage the following raw variables: id, commits, additions, deletions, issues and closed.

On the other hand, we have variables that a researcher constructs based on the raw data, or that they are provided by other sources as observation or face-to-face activity (synthetic data). We manage the following synthetic variables: days, commits/day, and authorship proof.

We will refer to these variables as features. In addition, we need a target variable (class); namely, a variable with the labels of classification that let us train and test the supervised learning model. As mentioned above, our target variable has two possible values: "AP", and "SS".

As shown at Figure 1, once we have the database, the following step is determining what are the most significant features in order to obtain a classification model based on the target variable. Feature selection is a procedure that selects the features that contribute most to the classification or the prediction. If we perform a feature selection prior to build the model, it is probably that we achieve to reduce overfitting, improve the accuracy and, obviously, reduce the training time. We have different available feature selection methods as the previous stage in machine learning: univariate selection, recursive feature elimination and the computation of the feature importance by learning algorithms. In this case, due to the nature of the database and because the data are not well modelled with a normal distribution, we have implemented the last ones. The ensemble methods are a good option for this goal because allow us to choose the type of algorithm that we want to use.

Results obtained in a preliminary study, see [18], show that the tree-based algorithms get good results with our database. Thus, we have chosen an extremely randomized tree [14] to compute the feature selection. Features with importance higher than a specific threshold are chosen. Feature importance is computed as the Gini coefficient (G).

Finally, to generate a model to predict a binary variable with possible values "AP" and "SS", from mixed input data (quantitative and qualitative variables), two types of Machine Learning (ML) algorithms may be used: classifiers and predictors, whereby considering the first ones will be better. Also, supervised classification techniques can be divided into two different categories: parametric and non-parametric models. In the first case, we have a fixed and finite number of parameters because we assume a specific form for the classification map. This machine learning algorithms could be effective because they are simple to understand and they do not need much training data to work. However, they are adequate to specific problems and could turn out to be too limited. On the other hand, in the non-parametric algorithms, the number of parameters is unknown and it could grow depending on the training set. These models are more flexible and powerful, but they require more training data and have more risk to get a high overfitting. Since the goal of the design is obtaining a tool that is the most possible general, we have included both types of machine learning algorithms in order that it fits as many educational situations as possible, see Table 1.

We have evaluated the following well-known methods that we think are the more promising ones: Linear Discriminant Analysis (LDA), Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), Adaptive Boosting (AB), Classification And Regression Tree (CART), Random Forest (RF), Multi-Layer Perceptron (MLP), and K-Nearest Neighbors (KNN).

Table 1: Parametric and non parametric models included in the MoEv tool.

Parametric model	Non parametric model
Linear Discriminant Analysis	Adaptive Boosting
Logistic Regression	Classification And Regression Tree
Naive Bayes	Random Forest
Support Vector Machine	Multi-Layer Perceptron
	K-Nearest Neighbors

LDA: Parametric method that assumes that distributions of the data are multivariate Gaussian [10]. Also, LDA assumes knowledge of population parameters. In another case, the maximum likelihood estimator can be used. LDA uses Bayesian approaches to select the category which maximizes the conditional probability (see [3], [19] or [21]).

LR: Linear methods are intended for regressions in which the target value is expected to be a linear combination of the input variables. LR, despite its name, is a linear model for classification rather than regression. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

NB: This method is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features, see [10] and [29].

SVM: SVMs are a set of supervised learning methods used for classification, regression and outliers detection. A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks [6]. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. We use Support Vector Classification (SVC) models to perform multi-class classification.

AB: Ensemble methods are techniques that combine different basic classifiers turning a weak learner into a more accurate method. Boosting is one of the most successful types of ensemble methods, and AB one of the most popular boosting algorithms.

CART: A decision tree is a method which predicts the label associated with an instance by traveling from a root node of a tree to a leaf [12]. It is a non-parametric method in which the trees are grown in an iterative, top-down process.

RF: Classifier consisting of a collection of decision trees, in which each tree is constructed by applying an algorithm to the training set and an additional random vector that is sampled via bootstrap re-sampling [4].

MLP: An artificial neural network is a model inspired by the structure of the brain. Neural networks are used when the type of relationship between inputs and outputs is not known. It is supposed that the network is organized in layers (an input layer, an output layer, and hidden layers). A MLP consists of multiple layers of nodes in a directed graph so that each layer is fully connected to the next one. A MLP is a modification of the standard linear perceptron and, the best characteristic is that it is able to distinguish data which is not linearly separable. An MLP uses back-propagation for training the network, see [24] and [7].

KNN: Although the nearest neighbors concept is the foundation of many other learning methods, notably unsupervised, supervised neighbor-based learning is also available to classify data with discrete labels. It is a non-parametric technique which classifies new observations based on the distance to observation in the training set. A good presentation of the analysis is given in [9] and [10].

Regarding the final decision, MoEv computes some well-known Key Performance Indicators (KPIs). To evaluate the above models with these input data, we have performed a k-iteration crossvalidation analysis for selecting the most suitable learning algorithm. Moreover, the accuracy classification score has been used to evaluate the performance of the models. The accuracy classification score is computed as follows, where $\sum T_p$ is the number of true positives, and P $\sum T_n$ is the number of true negatives:

$$accuracy = \frac{\sum T_p + \sum T_n}{\sum totaldata}$$

The three models with the highest accuracy classification score are pre-selected for in-depth evaluation by considering the following KPIs: Precision (P), Recall (R), and F1-score; all of which were obtained through the confusion matrix.

The Precision (P) is computed as follows, where $\sum F_p$ is the number of false positives:

$$P = \frac{\sum T_p}{\sum T_p + \sum F_p}$$

The Recall (R) is computed follows, where $\sum F_n$ is the number of false negatives.

$$R = \frac{\sum T_p}{\sum T_p + \sum F_n}$$

These quantities are also related to the F1-score, which is defined as the harmonic mean of precision and recall as follows:

$$F_1 = 2\frac{P \times R}{P \times R}$$

3 RESULTS

Figure 2 shows feature importances computed as the Gini coefficient. Features with a low Gini coefficient (G ≤ 0.1) have been discarded. Thus, the selected features are the following: authorship proof (G = 0.21), commits (G = 0.16), commits/day (G = 0.14), additions (G = 0.14), and days (G = 0.13).

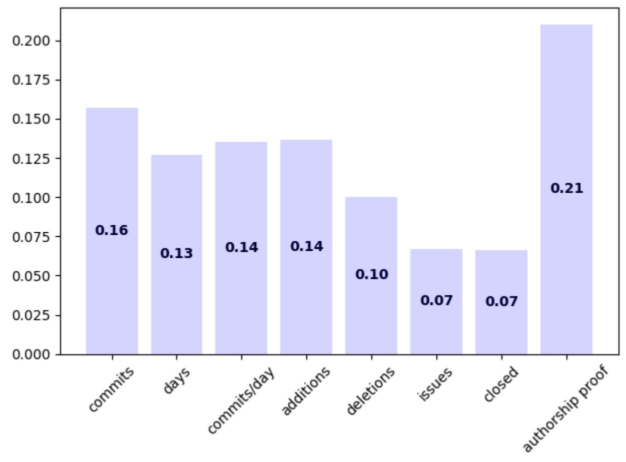

Figure 2: Features importances

Table 2 shows the accuracy classification score for test and validation datasets computed by the MoEv tool in a 10-iterations run. Since the generalization of the model is essential, models are ordered according to their validation score. The highest scores for the validation dataset are highlighted in bold.

Table 2: Accuracy classification score.

Classifier	Test score	Validation score
NB	0.8	**0.825**
RF	0.8	**0.8**
LDA	0.8	**0.725**
MLP	0.5	0.65
CART	0.4	0.6
AB	0.4	0.5
LR	0.7	0.475
KNN	0.6	0.475
SVN	0.4	0.525

Fig 3 shows the confusion matrix computed for the highlighted models: NB, RF, and LDA; using the test dataset. Table 3 shows the precision, recall and F1-score for the test dataset, also from highlighted models.

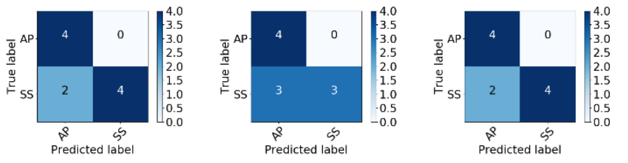

Figure 3: Confusion matrix for the NB (left), RF (center), and LDA (right) classifiers evaluated using the test dataset.

Table 3: Precision, recall and F_1-score for the test dataset.

Classifier	Class	P	R	F_1-score	#examples
NB	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg/total	0.87	0.80	0.80	10
RF	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg/total	0.87	0.80	0.80	10
LDA	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg/total	0.87	0.80	0.80	10

Figure 4 shows the confusion matrix computed for the highlighted models using the validation dataset. Table 4 shows the precision, recall and F1-score also using the validation dataset, for the highlighted models.

4 DISCUSSION

Feature importances at Figure 2 show that authorship proof is the most discriminant variable. This proof measures the students' knowledge about the content of their repositories. This feature is required to verify that students have really worked in the assignment and no one else. On the other hand, it is logical that demonstrating knowledge about the code contained in their repository has an important weight when it comes to predicting academic success in that specific task.

Regarding to other variables, commits, additions, days, and commits/day are the most discriminant (G > 0.1). They all come from student's interaction with the VCS so we can assert that this interaction has to do with the academic success. On the other hand, there are not too big differences among these features, so their importances might change with a different dataset.

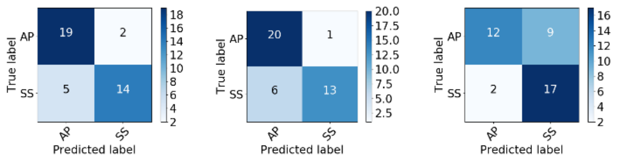

Figure 4: Confusion matrix for the NB (left), RF (center), and LDA (right) classifiers evaluated using the validation dataset.

Table 4: Precision, recall and F_1-score for the validation dataset.

Classifier	Class	P	R	F_1-score	#examples
NB	AP	0.79	0.90	0.84	21
	SS	0.88	0.74	0.80	19
	avg/total	0.83	0.82	0.82	40
RF	AP	0.76	0.90	0.83	21
	SS	0,87	0.68	0.76	19
	avg/total	0.81	0.80	0.80	40
LDA	AP	0.86	0.57	0.69	21
	SS	0.65	0.89	0.76	19
	avg/total	0.76	0.72	0.72	40

Table 2 shows the accuracy classification computed by MoEv. Models with higher scores for the validation dataset are pre-selected for in-depth analysis in order to ensure an optimal generalization. According to this criterion, NB, RF, and LDA get the highest accuracy for the validation dataset. They also are the models with the highest accuracy for the test dataset. This might be an indicator that both datasets are similar. We could make an only dataset and thus we could train models with a bigger dataset. However, in order to do so, we need to verify if there are statistically meaningful differences between both datasets. A similar analysis is done at [17].

Once the best generalizable models are considered, a deeper analysis with the confusion matrix of each one is given. Another important item that should be analyzed is the sensitivity of the model for detecting a pass: i.e., the rate of true passes that the model classifies incorrectly. Fig 3 and 4, and Tables 3 and 4, show that the NB classifier gets better values for precision (P), recall (R) and F1-score than RF and LDA in both test and validation stages.

5 CONCLUSIONS

A major contribution of the work described in this paper is the MoEv tool, which allows to build prediction models for different problems by performing a feature selection prior to a cross-validation analysis to select the most suitable model. In this work, the goal is to build a model to predict academic success by monitoring students' activity at VCSs.

The paper also poses two research questions: firstly, *are there features that we can extract from the students' interactions with this type of systems that are related with the academic success?*, and secondly, *can we build a model that allows predicting students' success at a practical assignment, by monitoring their use of a VCS?*

With regard to the first question, the feature analysis carried out show the importance of each feature. This allow to identify which ones have a greater weight in the model. This is the first step to obtaining a classification model that allows to predict the academic success of students. Results show that some features related with students' interaction with the VCS are discriminant. However, including more features, such as a sort of validation exam (like the authorship proof described at section 2) increase model accuracy.

Relative to the second question posed, the MoEv tools provide a prediction model by evaluating several classifiers. There is future works to do due to optimize the selected model by tuning its hyperparameters, but results are enough to assert that we can predict students' results at a practical assignment with a success high percentage.

Further works should face the accuracy improvement. In order to do so, in addition to hyper-parameters tuning, it would be desirable to increase training data. A first approach may be done by combining both test and validation dataset. However, a prior analysis is required in order assert that there are not statistical meaningful differences between both datasets.

It would be also interesting to include additional predictor variables in the models. For example, variables about interactions with other educational resources might be included for which the environment would need some changes.

REFERENCES

[1] Ángel F Agudo-Peregrina, Santiago Iglesias-Pradas, Miguel Ángel Conde-González, and Ángel Hernández-García. 2014. Can we predict success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning. Computers in human behavior 31 (2014), 542–550.
[2] Rebecca Barber and Mike Sharkey. 2012. Course correction: Using analytics to predict course success. In Proceedings of the 2nd international conference on learning analytics and knowledge. ACM, 259–262.
[3] Christopher M Bishop. 2006. Pattern recognition. Machine Learning 128 (2006), 1–58.
[4] Leo Breiman. 2001. Random forests. Machine learning 45, 1 (2001), 5–32.
[5] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. 2005. Linux Device Drivers: Where the Kernel Meets the Hardware. O'Reilly Media, Inc.
[6] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. Machine learning 20, 3 (1995), 273–297.
[7] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) 2, 4 (1989), 303–314.
[8] Brian De Alwis and Jonathan Sillito. 2009. Why are software projects moving from centralized to decentralized version control systems? In Proceedings of the 2009 ICSE Workshop on cooperative and human aspects on software engineering. IEEE Computer Society, 36–39.
[9] Luc Devroye, László Györfi, and Gábor Lugosi. 2013. A probabilistic theory of pattern recognition. Vol. 31. Springer Science & Business Media.
[10] Richard O Duda, Peter E Hart, and David G Stork. 2012. Pattern classification.
[11] John Wiley & Sons. [11] Michael Fischer, Martin Pinzger, and Harald Gall. 2003. Populating a release history database from version control and bug tracking systems. In Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on. IEEE, 23–32.
[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2009. The elements of statistical learning Ed. 2. Vol. 1. Springer series in statistics Springer, Berlin.
[13] Dragan Gašević, Shane Dawson, Tim Rogers, and Danijela Gasevic. 2016. 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.
[14] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. Machine learning 63, 1 (2006), 3–42.
[15] Terry Griffin and Shawn Seals. 2013. Github in the classroom: Not just for group projects. Journal of Computing Sciences in Colleges 28, 4 (2013), 74–74.
[16] Ángel Manuel Guerrero-Higueras, Noemí DeCastro-García, and Vicente Matellán. 2018. Detection of Cyber-attacks to indoor real time localization systems for autonomous robots. Robotics and Autonomous Systems 99 (2018), 75–83.
[17] Ángel Manuel Guerrero-Higueras, Noemí DeCastro-García, Francisco Javier Rodríguez-Lera, and Vicente Matellán. 2017. Empirical analysis of cyber-attacks to an indoor real time localization system for autonomous robots. Computers & Security. 70 (2017), 422–435.
[18] Ángel Manuel Guerrero-Higueras, Vicente Matellán-Olivera, Gonzalo Esteban-Costales, Camino Fernández-Llamas, Francisco Jesús Rodríguez-Sedano, and Conde. Miguel Ángel. 2018. Model for evaluating student performance through their interaction with version control systems. In Learning Analytics Summer Institute (LASI). SNOLA.
[19] Daphne Koller and Nir Friedman. 2009. Probabilistic graphical models: principles and techniques. MIT press.
[20] Z Kovacic. 2012. Predicting student success by mining enrolment data. Research in Higher Education Journal 15 (2012).
[21] Kevin P Murphy. 2012. Machine learning: a probabilistic perspective. MIT press.
[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vinxent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.
[23] C Michael Pilato, Ben Collins-Sussman, and Brian W Fitzpatrick. 2008. Version Control with Subversion: Next Generation Open Source Version Control. O'Reilly Media, Inc.
[24] David E Rummelhart. 1986. Learning internal representations by error propagation. Parallel distributed processing (1986).
[25] George Siemens, Shane Dawson, and Grace Lynch. 2013. Improving the quality and productivity of the higher education sector. Policy and Strategy for Systems-Level Deployment of Learning Analytics. Canberra, Australia: Society for Learning Analytics Research for the Australian Office for Learning and Teaching (2013).
[26] George Siemens and Dragan Gasevic. 2012. Learning and knowledge analytics. Educational Technology & Society 15, 3 (2012), 1–2.
[27] Diomidis Spinellis. 2005. Version control systems. IEEE Software 22, 5 (2005), 108–109.
[28] Linus Torvalds and Junio Hamano. 2010. Git: Fast version control system. http:// git-scm.com (2010).
[29] Harry Zhang. 2004. The optimality of naive Bayes. AA 1, 2 (2004), 3.