Research Article

Ángel Manuel Guerrero-Higueras*, Noemí DeCastro-García, Francisco Javier Rodriguez-Lera, Vicente Matellán, and Miguel Ángel Conde

Predicting academic success through students' interaction with Version Control Systems

https://doi.org/10.1515/comp-2019-0012
Received May 31, 2019; accepted Jul 23, 2019

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 evaluate if 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 which predicts student results in a specific practical assignment of the Operating Systems Extension 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 Machine Learning models in order to get the most suitable for a specific problem. Prior to the model development, a feature selection from 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 predicts students' success with a success high percentage.

Keywords: Version Control System, Machine Learning, Learning analytics

***Corresponding Author: Ángel Manuel Guerrero-Higueras:** Dept. Mech, Computer and Aerospace Eng., Universidad de León, León, Spain; Email: am.guerrero@unileon.es
Noemı DeCastro-Garcıa: Deparment of mathematics, Universidad de León, León, Spain; Email: ncasg@unileon.es
Francisco Javier Rodriguez-Lera: Dept. Mech, Computer and Aerospace Eng., Universidad de León, León, Spain; Email: fjrodl@unileon.es
Vicente Matellán: Supercomputación Castilla y León (SCAyLE), León, Spain; Email: vicente.matellan@scayle.es
Miguel Ángel Conde: Dept. Mech, Computer and Aerospace Eng., Universidad de León, León, Spain; Email: mcong@unileon.es

Acronyms

AB Adaptive Boosting.
CART Classification And Regression Tree.
ICT Information and communication technology.
KNN K-Nearest Neighbors.
KPI Key Performance Indicator.
LDA Linear Discriminant Analysis.
LMS Learning Management System.
LR Logistic Regression.
ML Machine Learning.
MLP Multi-Layer Perceptron.
MoEv Model Evaluator.
NB Naive Bayes.
OSE Operating Systems Extension.
OSEFS Operating Systems Extension File System.
RF Random Forest.
SIS Student Institutional System.
VCS Version Control System.

1 Introduction

The emergence of Information and communication technologys (ICTs) has changed the teaching and learning processes. Teachers can employ a lot of tools in their classes with the aim to improve students learning. In addition, students can use different applications in their education center, and beyond it. However, Is it possible to say if a tool is improving student performance? If we can assert this, it would be possible to use the tool that better fits with specific lessons or students. There are several studies regarding this, and this issue is especially related to Learning Analytics and Educational Data Mining.

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 [1]). 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), provide 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 the research of learning analytics is obtaining a predictive model for the academic student's grades. In particular, it is very common to try to obtain the risk (or the probability) of that a student fails or passes a course [2]. Predictive analytics has an important consequence in 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 having a high-quality educational system. The research in this question is focused on obtaining specific predictive models that only have high accuracy over concrete scenarios that depending on the field, the size of the samples, or the data source, see [3]. This limitation is something expected due to the models are computed by advanced learning algorithms that require good and quality data for each case. This fact creates difficulties in order to get a general overview of the effect of different features on academic success, and for getting a general and easy tool for this type of learning analytics.

A goal of this work is to present Model Evaluator (MoEv), an evaluation tool which automatically studies 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. The tool was first used in [4]. Here we propose that MoEv could be applied to different courses, fields, or learning situations. MoEv is based on the idea of [5], 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 [6], the trace data recorded by LMSs and other learning environments, [7], or hybrid data sources composed by the ones described above [8].

On the other hand, in some fields as engineering or computer science, it is very usual that teachers employ advanced tools and applications in their courses with the aim to give to the students a meaningful learning experience as close as possible to the professional world. For instance, in software engineering, the changing management in the components of a software product or its configuration is known as version control [9]. It is called version, revision or edition, to the state of the product at a specific time. Version control 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) [10].

A VCS should provide, at least, the following features: storage for the different elements to be managed such as source code, images, and documentation; edition the stored elements (creation, deletion, modification, renaming, etc.); and registration and labeling of all actions carried out, of so that they allow an element to be returned to a state previous. Among the most popular there are the following: CVS, Subversion [11] or Git [12].

One of the knowledge more highly demanded in ICT professional profiles is the use of VCSs. So, a 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 to 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 [13], concluded that analyzing the students' activity at VCSs allows predicting their results by using tree-based algorithms. An in-depth analysis presented at [4] shown that results were different depending on the chosen features. [4] also follow de method proposed at [5] to select the most suitable model, but a previous feature analysis was done prior to the model development. In both [4, 13] a training dataset was used to build the model. In addition, in order to ensure an optimal generalization, in [5] a validation of the selected model was done by using a second dataset. Regarding the features, [4] concluded that considering additional features to the ones from students' interaction with VCS improve the accuracy. In this work, we explore the results without adding any external feature, which a priori would ensure an optimal validation.

The rest of the paper is organized as follows: Section 2 describes the empirical evaluation of the classification algorithms presenting the experimental environment, materials, and methods used. Section 3 summarizes the results of the evaluation. The discussion of the results is developed in Section 4. Section 5 presents the conclusions and

future lines of research. Finally, the main references are given.

2 Experimental procedures

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: a practical assignment which Operating Systems Extension (OSE) students need to pass, a VCS platform to gather data, and the MoEv tool to get an optimized model.

Input data

The OSE 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 OSE students need to pass consists of implementing an inode-based file system called Operating Systems Extension File System (OSEFS). 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 [14] that supports, at least, the following operations: mounting of devices formatted with this system; creation, reading, and writing of regular files; creation of new directories and the visualization of the content of existing directories.

This is an individual assignment and each student is encouraged to use a VCS during the completion of the task. For the development of OSEFS, students are encouraged to use 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 [15]. In order to provide some organizing capabilities and private repositories for students, the GitHub Classroom platform was used [16]. 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 OSE 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 date (*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 issued 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 student repositories. This *authorship proof* allows us to verify that the students really worked in the content of their repository. The *authorship proof* has two possible results: "1" if the student passed the proof; "0" otherwise.

The *authorship proof* is different from the other variables since it is not obtained directly from the students' interaction with VCSs. In [4] is shown that this feature is the most significant. Nonetheless, in this work, we intend to evaluate the results without considering the *authorship proof* to ensure optimal validation.

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 OSE course come from two groups of students. The first group include 46 students who tried the OSEFS assignment from 2016–2017 OSE course. The data includes the features mentioned above. It is a balanced sample with 21 students labeled 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 the training dataset.

The second group include 40 students who tried the OSEFS assignment from 2017–2018 OSE course, whose number of "AP" and "SS" is 21 and 19, respectively. These

data are used to validate the prediction model. We will refer to them as validation dataset.

Model design

To get the best predictive model we have built the MoEv tool following the method proposed at [5] but adding a feature selection phase to the process. To build MoEv, the Scikit-learn library has been used [17]. Figure 1 show the steps of the methodology used.

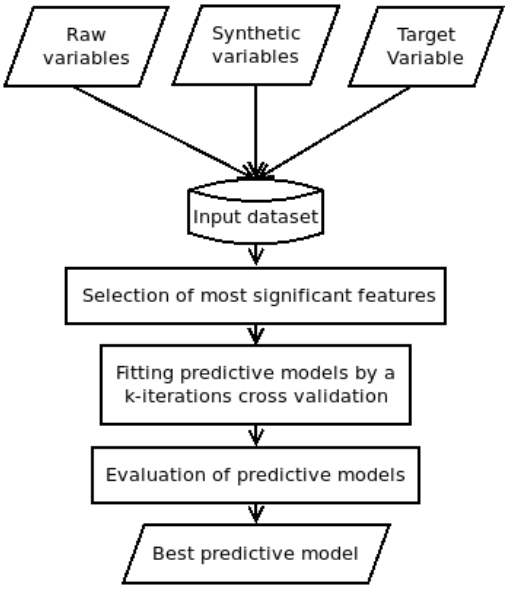

Figure 1: MoEv operation steps.

We start from an input dataset 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 raw and synthetic 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 in Figure 1, once we have the input dataset, 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 building 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 modeled 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 [13], show that the tree-based algorithms get good results with our database. Thus, we have chosen an extremely randomized tree [18] to compute the feature selection. Features with importance higher than a specific threshold are chosen. Feature importance is computed as the Gini coefficient ($\mathcal{G}$).

Once selected the most significant features, several models are fitted 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 adequated 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 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.

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

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

Parametric model	Non parametric model
Linear Discriminant Analysis	Ensemble methods
Naive Bayes	Decision Tree
Logisic Regression	Random Forest
	Multilayer-perceptron
	K- Nearest Neighbours
	Classifier

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 [19]. It is a non-parametric method in which the trees are grown in an iterative, top-down process.

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 [20] and [21].

LDA Parametric method that assumes that distributions of the data are multivariate Gaussian [21]. 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 [22], [23] or [24]).

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.

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 [25] and [26].

NB This method is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features, see [21] and [27].

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 [28].

To fit the above models with the input dataset, MoEv performs a k-iteration cross-validation. Finally, to get the most suitable learning algorithms MoEv has to evaluate the previous models. In order to do so, it computes some well-known Key Performance Indicators (KPIs). Moreover, the accuracy classification score has been used to evaluate the performance of the models. The accuracy classification score is computed as shown at equation 1, where $\sum T_p$ is the number of true positives, and $\sum T_n$ is the number of true negatives.

$$accuracy = \frac{\sum T_p + \sum T_n}{\sum \text{total data}} \tag{1}$$

The three models with the highest accuracy classification score have been pre-selected for in-depth evaluation by considering the following KPIs: Precision (P), Recall (R), and F_1-score; all of which were obtained through the confusion matrix.

The Precision (P) is computed as shown at equation 2, where $\sum F_p$ is the number of false positives.

$$P = \frac{\sum T_p}{\sum T_p + \sum F_p} \tag{2}$$

The Recall (R) is computed at equation 3, where $\sum F_n$ is the number of false negatives.

$$R = \frac{\sum T_p}{\sum T_p + \sum F_n} \tag{3}$$

These quantities are also related to the F_1-score, which is defined as the harmonic mean of precision and recall as shown at equation 4.

$$F_1 = 2\frac{P \times R}{P + R} \tag{4}$$

3 Results

Figure 2 shows feature importances computed as the Gini coefficient as shown in [4]. In [4], features with a low Gini

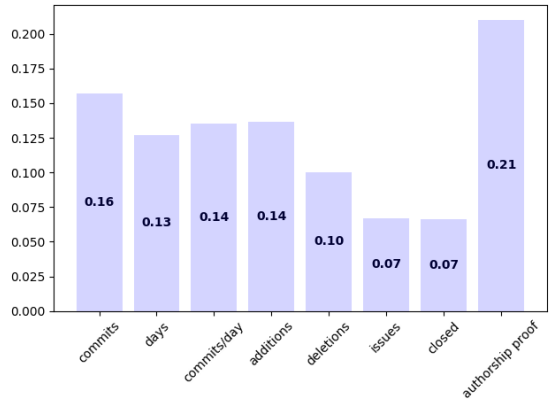

Figure 2: Features importances as shown in [4].

Table 2: Accuracy classification score as shown in [4].

Classifier	Test score	Validation score
NB	0.8	**0.8**
RF	0.8	**0.8**
LDA	0.8	**0.7**
MLP	0.5	0.7
CART	0.4	0.6
AB	0.4	0.5
LR	0.7	0.5
KNN	0.6	0.5

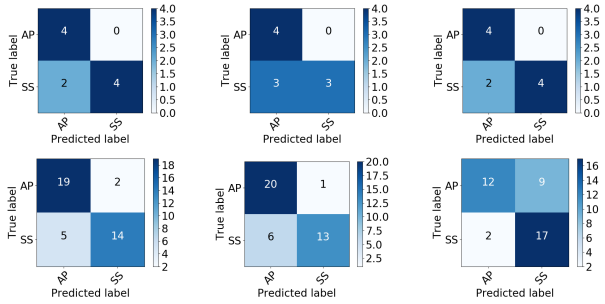

Figure 3: Top: Confusion matrix for the NB (left), RF (center), and LDA (right) classifiers evaluated using the test dataset. Bottom: Same data using the validation dataset.

coefficient ($\mathcal{G} \leq 0.1$) were discarded. The selected features were following: *authorship proof* ($\mathcal{G} = 0.21$), *commits* ($\mathcal{G} = 0.16$), *commits/day* ($\mathcal{G} = 0.14$), *additions* ($\mathcal{G} = 0.14$), and *days* ($\mathcal{G} = 0.13$).

As mentioned above, *authorship proof* is the most significant feature. According to this, Table 2 shows the accuracy classification score for test and validation datasets computed by the MoEv tool in a 10-iterations execution. Since the generalization of the model is essential, models

Table 3: Accuracy classification score without considering *authorship proof*.

Classifier	Test score	Validation score
RF	0.7	**0.7**
MLP	0.4	**0.7**
LDA	0.6	**0.6**
CART	0.6	0.6
NB	0.5	0.5
KNN	0.6	0.5
AB	0.4	0.5
LR	0.6	0.4

Table 4: 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

Table 5: 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

are ordered according to their validation score. The highest scores for the validation dataset are highlighted in bold.

Table 3 also shows the accuracy classification score for test and validation datasets computed the same way but without considering the *authorship proof*.

Figure 3–top shows the confusion matrix computed for the highlighted models: NB, RF, and MLP; using the test dataset. Figure 3–bottom shoes the same data using the validation dataset.

Table 4 shows the precision, recall and F_1-score for the test dataset, also from highlighted models.

Table 5 shows the precision, recall and F_1-score also using the validation dataset, for the highlighted models.

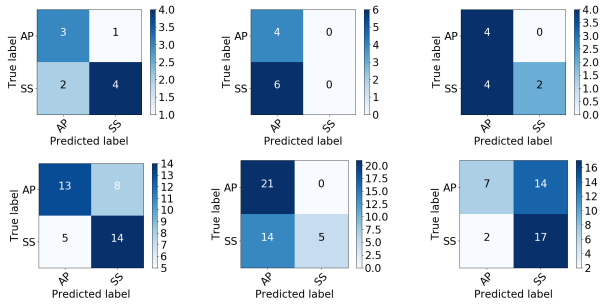

Figure 4: Top: Confusion matrix for the RF (left), MLP (center), and LDA (right) classifiers evaluated using the test dataset without considering *authorship proof*. Bottom: Same data using the validation dataset.

Table 6: Precision, recall and F_1-score for the test dataset without considering *authorship proof*.

Classifier	Class	*P*	*R*	F_1-score	#examples
	AP	0.60	0.75	0.67	4
RF	SS	0.80	0.77	0.73	6
	avg/total	0.72	0.70	0.70	10
	AP	0.40	1.00	0.57	4
RF	SS	0.00	0.00	0.00	6
	avg/total	0.16	0.40	0.23	10
	AP	0.50	1.00	0.67	4
LDA	SS	1.00	0.33	0.50	6
	avg/total	0.80	0.60	0.57	10

Table 7: Precision, recall and F_1-score for the validation dataset without considering *authorship proof*.

Classifier	Class	*P*	*R*	F_1-score	#examples
	AP	0.72	0.62	0.67	21
RF	SS	0.64	0.74	0.68	19
	avg/total	0.68	0.68	0.67	40
	AP	0.60	1.00	0.75	21
RF	SS	1.00	0.26	0.42	19
	avg/total	0.79	0.65	0.57	40
	AP	0.78	0.33	0.47	21
LDA	SS	0.55	0.89	0.68	19
	avg/total	0.67	0.60	0.57	40

Figure 4–top shows the confusion matrix computed for the highlighted models: RF, MLP, and LDA; using the test datasetwithout considering *authorship proof*. Figure 4–bottom shoes the same data using the validation dataset.

Table 6 shows the precision, recall and F_1-score for the test dataset, also from highlighted models without considering *authorship proof*. Table 7 shows the precision, recall and F_1-score also using the validation dataset, for the highlighted models without considering *authorship proof*.

4 Discussion

Feature importances at Figure 2 show that *authorship proof* is the most discriminant variable. This proof measures the students' knowledge about the contents of their repositories. It 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.

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

Table 2 shows the accuracy classification computed by MoEv. Models with higher scores for the validation dataset are pre-selected for an 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 [29].

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. Figure 3 and 4, and Tables 4 and 5, show that the NB classifier gets better values for precision (P), recall (R) and F_1-score than RF and LDA in both test and validation stages.

Table 2 shows the accuracy classification computed by MoEv without considering the *authorship proof*. As in the previous analysis, models with higher scores for the validation dataset are pre-selected for and in-depth analysis in order to ensure an optimal generalization. According to this criterion, RF, MLP, and LDA get the highest accuracy for the validation dataset. They are not the models with the highest accuracy for the test dataset.

Again, once the best generalizable models are considered, a deeper analysis with the confusion matrix of each one is given. According to this criterion, again RF, MLP, and LDA get the highest accuracy for the validation dataset. Figure 4, and Tables 6, and 7, show that the RF

classifier gets better values for precision (P), recall (R) and F_1-score than MLP and LDA in both test and validation stages.

5 Conclusions

A major contribution of the work described in this paper is a tool, known as MoEv, which allows building 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 allows identifying which ones have a greater weight in the model. This is the first step to obtaining a classification model that allows predicting the academic success of students. Results show that some features related to students interaction with the VCS are discriminant. However, include more features, such as *authorship proof*, increase models accuracy.

Relative to the second question posed, the MoEv tools provide a prediction model by evaluating several classifiers. There are future works to do due to optimizing the selected model by tuning its hyper-parameters, 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 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 to assert that there are not statistically meaningful differences between both datasets.

References

[1] Siemens G., Gasevic D., Guest editorial-Learning and knowledge analytics., Educational Technology & Society, 15(3), 2012, 1–2

[2] Siemens G., Dawson S., Lynch G., 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

[3] Gašević D., Dawson S., Rogers T., Gasevic D., 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

[4] Guerrero-Higueras Á.M., DeCastro-García N., Matellán V., Conde M.Á., Predictive models of academic success: a case study with version control systems, in Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality, ACM, 2018, 306–312

[5] Guerrero-Higueras Á.M., DeCastro-García N., Matellán V., Detection of Cyber-attacks to indoor real time localization systems for autonomous robots, Robotics and Autonomous Systems, 99, 2018, 75–83

[6] Kovacic Z., Predicting student success by mining enrolment data., Research in Higher Education Journal, 15, 2012

[7] Agudo-Peregrina Á.F., Iglesias-Pradas S., Conde-González M.Á., Hernández-García Á., 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

[8] Barber R., Sharkey M., Course correction: Using analytics to predict course success, in Proceedings of the 2nd international conference on learning analytics and knowledge, ACM, 2012, 259–262

[9] Fischer M., Pinzger M., Gall H., Populating a release history database from version control and bug tracking systems, in Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on, IEEE, 2003, 23–32

[10] Spinellis D., Version control systems, IEEE Software, 22(5), 2005, 108–109

[11] Pilato C.M., Collins-Sussman B., Fitzpatrick B.W., Version Control with Subversion: Next Generation Open Source Version Control, " O'Reilly Media, Inc.", 2008

[12] Torvalds L., Hamano J., Git: Fast version control system, http://git-scm.com, 2010

[13] Guerrero-Higueras A.M., Matellán-Olivera V., Esteban-Costales G., Fernández-Llamas C., Rodríguez-Sedano F.J., Ángel C.M., Model for evaluating student performance through their interaction with version control systems, in Learning Analytics Summer Institute (LASI), SNOLA, 2018

[14] Corbet J., Rubini A., Kroah-Hartman G., Linux Device Drivers: Where the Kernel Meets the Hardware, "O'Reilly Media, Inc.", 2005

[15] De Alwis B., Sillito J., 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, 2009, 36–39

[16] Griffin T., Seals S., Github in the classroom: Not just for group projects, Journal of Computing Sciences in Colleges, 28(4), 2013, 74–74

[17] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E., Scikit-learn: Machine Learning in Python, Journal

of Machine Learning Research, 12, 2011, 2825–2830
[18] Geurts P., Ernst D., Wehenkel L., Extremely randomized trees, Machine learning, 63(1), 2006, 3–42
[19] Friedman J., Hastie T., Tibshirani R., The elements of statistical learning Ed. 2, volume 1, Springer series in statistics Springer, Berlin, 2009
[20] Devroye L., Györfi L., Lugosi G., A probabilistic theory of pattern recognition, volume 31, Springer Science & Business Media, 2013
[21] Duda R.O., Hart P.E., Stork D.G., Pattern classification, John Wiley & Sons, 2012
[22] Bishop C.M., Pattern recognition, Machine Learning, 128, 2006, 1–58
[23] Koller D., Friedman N., Probabilistic graphical models: principles and techniques, MIT press, 2009
[24] Murphy K.P., Machine learning: a probabilistic perspective, MIT press, 2012
[25] Rummelhart D.E., Learning internal representations by error propagation, Parallel distributed processing, 1986
[26] Cybenko G., Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems (MCSS), 2(4), 1989, 303–314
[27] Zhang H., The optimality of naive Bayes, AA, 1(2), 2004, 3
[28] Breiman L., Random forests, Machine learning, 45(1), 2001, 5–32
[29] Guerrero-Higueras Á.M., DeCastro-García N., Rodríguez-Lera F.J., Matellán V., Empirical analysis of cyber-attacks to an indoor real time localization system for autonomous robots, Computers & Security., 70, 2017, 422–435