applied sciences

MDPI

Article

Academic Success Assessment through Version Control Systems

Ángel Manuel Guerrero-Higueras *, Camino Fernández Llamas, Lidia Sánchez González, Alexis Gutierrez Fernández, Gonzalo Esteban Costales and Miguel Ángel Conde González

Robotics group, dept. of Mechanical, Computer Science, and Aerospace Engineering, University of León, Campus de Vegazana s/n, 24071 León, Spain; camino.fernandez@unileon.es (C.F.L.); lidia.sanchez@unileon.es (L.S.G.); alexis.gutierrez@unileon.es (A.G.F.); gestc@unileon.es (G.E.C.); mcong@unileon.es (M.C.G.)

* Correspondence: am.guerrero@unileon.es

Received: 16 January 2020; Accepted: 18 February 2020; Published: 21 February 2020

check for updates

Abstract: Version control systems' usage is a highly demanded skill in information and communication technology professionals. Thus, their usage should be encouraged by educational institutions. This work demonstrates that it is possible to assess if a student can pass a computer science-related subject by monitoring its interaction with a version control system. This paper proposes a methodology that compares the performance of several machine learning models so as to select the appropriate predicting model for the assessment of the students' achievements. To fit predicting models, three subjects of the Degree in Computer Science at the University of León are considered to obtain the dataset: computer organization, computer programming, and operating systems extension. The common aspect of these subjects is their assignments, which are based on developing one or several programs with programming languages such as C or Java. To monitor the practical assignments and individual performance, a Git repository is employed allowing students to store source code, documentation, and supporting control versions. According to the presented experience, there is a huge correlation between the level of interaction for each student and the achieved grades.

Keywords: computer programming; version control system; machine learning; learning analytics

1. Introduction

Learning Analytics (LA) is defined in [1] as the measurement, collection, analysis, and reporting of data about learners and their contexts in order to understand and optimize the learning process and its environments. The use of LA provides many advantages such as carrying out data mining in education and also helping lecturers to optimize their teaching as they obtain current and actual feedback of each student's learning process. Therefore, by analyzing data from both Student Institutional Systems (SISs) or Learning Management Systems (LMSs), information about the students' interaction is gathered. Such data allow lecturers to identify patterns that point out if a resource is adequate or if the learning outcomes can be achieved.

According to recent literature, LA research is widely applied to estimate academic results. In particular, in [2], the students' probability to pass or fail a course was determined. By employing predictive analytics, those students whose chances to pass a subject were lower could be identified, so an intervention could be adapted in order to mitigate the academic failure. In other words, the impact on education was truly positive. However, these methods depend on the field where they are applied, the sample features, or the data source [3], as well as what has an influence on the obtained accuracy. In existing methodologies that use learning algorithms, the quality data are an essential requirement;

such a drawback avoids detecting the influence of different features on academic success, as well as designing a ready-to-use tool that can be applied in general.

On the contrary, the Module Evaluator (MoEv) tool presented in [4,5] handled these problems by evaluating different parametric and non-parametric learning algorithms, automatically identifying the more accurate one for a specific problem. MoEv follows the methodology proposed in [6], but it appends a crucial step so as to generalize models: the feature extraction is done automatically by choosing the most discriminant variables. In order to design a general tool that makes it possible to analyze any data source, the first stage is to identify the most valuable information for the model. Usually, predictive models on education consider data sources such as SISs [7], the trace data recorded by LMSs and other learning environments [8], or hybrid data sources formed by a combination of them [9].

On the other hand, the use of a Version Control System (VCS) is an indispensable skill for Information and Communications Technology (ICT) professionals and demanded by companies [10]. Thus, it is widely used by lecturers during practical assignments so as to provide a meaningful learning experience.

In order to facilitate software development, there are specific tools that manage the code changing named version control [11]. In this field, a version means how the product is at a certain time. Moreover, these tools usually provide version management since it is a hard task if it is done manually. Not only a VCS has to provide storage for the source code, images, or documentation, but also they can be editable, keeping a record of every performed action so as to revert changes at any time in case this is needed. This last feature is very important since it makes it possible to return an element to its previous state, which is quite common in programming. Some examples of popular VCSs are CVS, Subversion [12], or Git [13].

The authors aim to estimate students' academic achievements by supervising their daily performance with VCSs. In order to do so, there are two issues with which to deal. The first one is if academic success is related to the considered variables that are computed from the students' interaction with VCSs. The second is if it is possible to design a model that analyses those features in order to estimate the students' success automatically.

Preliminary results of this approach were presented in [4,5,14]. A proof of concept about using the features extracted from the interaction with VCSs to predict academic success was presented in [14]. To build the prediction models, a single dataset gathered data from the Operating Systems Extension Subject of the degree in Computer Science at the University of León. Specifically, the dataset contained data from the 2016-2017 course. In [4,5], in order to build more accurate predictions models, the authors used two different datasets also gathering data from the Operating Systems Extension Subject. The first dataset, which contained data from the 2016-2017 course, was used to fit and test the models. The second one, which contained data from the 2017-2018 course, was used to evaluate that such models ensured an optimal generalization. This subject was selected because students are required to complete a practical assignment, which implies developing software solutions so as to pass the course. In the required practical assignments of this subject, a VCS was employed to store both source code and documentation. Regarding the first issue described before, both works demonstrated that some variables related to students' interaction with VCSs were discriminant. Relating to the second one mentioned earlier, both works presented models that allowed the prediction of students' success by analyzing such variables. The main difference between [4] and [5] had to do with the contents of the datasets. In [4], the dataset contained not only variables related to the interaction with VCSs, but also some additional variables, which increased model accuracy. In [5], only variables related to the interaction with VCSs were used. The accuracy of the models described in [4] was higher, but the authors considered the models presented in [5] more interesting since they may be used to build a general and ready-to-use tool.

To ensure an optimal generalization of prediction models, the use of a much bigger dataset was proposed in [15]. A new dataset was built not only gathering data from the Operating Systems Extension subject, but also from Computer Programming I and Computer Organization subjects. These subjects also belong to the degree in Computer Science at the University of León. As in

Operating Systems Extension, students had to develop software solutions using a VCS to store both source code and documentation.

Besides the data related to the interaction with VCSs, the work in [15] proposed including some additional variables in the dataset extracted from the source code. These were the total number of source code lines of the software solution, the number of used subroutines, or the number of variables or objects, among others. The authors considered that these variables may be used to analyze the quality of the source code and, thus, study the evolution of students' programming skills.

In this work, the authors built new prediction models by using the dataset presented in [15]. The new models presented a higher accuracy than the ones described in [5] and a similar accuracy to the models in [4]. It is important to point out that the optimized models proposed here, contrary to the models presented in [4], ensured an optimal generalization since they just used variables related to the interaction with VCSs and their contents.

The organization of this paper is as follows: Section 2 includes a description of the empirical evaluation of the considered classification algorithms, the experience details, the materials, and methods used; Section 3 shows the obtained results; the discussion of the results is presented in Section 4; Section 5 gathers the conclusions and future work.

2. Materials and Methods

This section includes a description of all the elements used to design and evaluate models to estimate students' academic success through their interaction with a VCSs. First, a dataset was built by gathering data from several practical assignments from subjects of the degree in Computer Science at the University of León. In order to do so, a commercial VCS was used. Then, MoEv was employed to get an optimized model. Each element is further outlined below.

2.1. Version Control System

All the considered practical assignments required the use of a VCS; in particular, a Git repository. Git follows a distributed scheme where each copy of the repository also encloses the whole change history, unlike other systems that follow the client-server models [16]. As the repositories were public, the GitHub Classroom platform was considered so as to use the organizing functionalities that it presents and also to offer private repositories for students [17]. GitHub is a website and service that uses the Git revision control system facilitating the development of software projects. Otherwise, GitHub Classroom provides different useful functionalities for lecturers such as task assignments for students or groups of students all in the same hosting service.

2.2. Input Data

In this paper, the considered data were extracted from the practical assignments proposed in different subjects of the Computer Science Degree at the University of León. In particular, the subjects that participated in this experience were Operating Systems Extension, Computer Programming I, and Computer Organization.

There were two subjects for first-year students: Computer Programming I and Computer Organization. In the case of Computer Programming I, it deals with the fundamentals of software programming, that is design, implementation, and testing. Therefore, students acquire the competence of designing at a certain abstract level for different algorithms and patterns. In addition to this, students learn to design and code test cases.

Computer Organization is also a first-year course, and its objectives are related to the fundamentals of computers such as datapath, control unit, memory, or input/output. In the practical lessons, students learn how to program in both C and assembler languages.

Finally, the Operating Systems Extension is a second-year course where students deal with operating systems. More specifically, students acquire knowledge about how the main memory works (volatile) and how the file system is managed (non-volatile). This subject also addresses security issues in operating systems.

The reason to select the previously mentioned subjects was that all of them involve practical assignments where students have to develop a program in a certain language. Moreover, the use of a Git repository is mandatory in order to store both source code and documentation. Table 1 shows the number of samples for each subject.

Table 1. Number of samples in the dataset.

Subject	Year	Course	Practical Assignments	Support
Computer Programming I	1°	2018–2019	3	372
Computer Organization	1°	2018–2019	5	240
Operating Systems Extension	2°	2016–2017	1	72
		2017–2018	1	47

Regarding the feature extraction, the dataset was formed as follows. Considering the practical assignments developed by students and its activity on the repository, different variables were considered:

- An identifier to make students anonymous (*id*): it provided differentiation among the samples.
- Commits counter (*commits*): it represented the number of commit operations made by a given student.
- Number of days where at least one commit was done (*days*).
- Average number of commit operations per day (*commits/day*).
- Number of additional lines appended to the source code during the assignment development (*additions*).
- Number of deleted lines removed from the source code during the assignment development (*deletions*).
- Number of issues opened (*issues-opened*) during the assignment.
- Number of issues closed (*issues-closed*) during the assignment.

In [15], the authors proposed to collect some additional variables extracted from the source code: the total number of source code lines, the number of subroutines, the number of variables or objects, the number of classes, the number of loops, the cyclomatic complexity, the number of files, some object-oriented programming related features (i.e., inheritance and polymorphism usage), the number of README files, dependencies, etc. They also considered variables computed by the measure of the interaction with VCS. These data may be later used to analyze the quality of the source code and, thus, study the evolution of students' programming skills. Specifically, in this work, the following additional variables were considered: the number of source code files written in C (including header files), Java, and the assembler; as well as the number of source code lines in each file.

2.3. Models Design and Evaluation

In this work, the MoEv tool was employed in order to achieve the model that performed the best prediction. The followed methodology was presented in [4,5]. Figure 1 shows a scheme of the applied methodology.

The feature vector of each sample was formed by two kinds of variables. One type was those variables extracted from raw data given by an SIS or an LMS directly. In this work, the feature vector had the following raw data elements: id, commits, additions, deletions, issues-opened, and issues-closed. In addition to this, source code metrics also are considered as raw data. Besides raw data, there were synthetic data represented by the following variables: days (number of days with at least a commit operation) and commits/day (average commit operations per day).

Once the feature vector was built, a target variable (class) was included. That was a component that determined the class to which a certain feature vector belonged. This class element allowed the model to train and test the supervised learning procedure. The considered target was a binary variable with a Pvalue if a given student will pass the course and a Fvalue if a given student will fail.

Following the methodology shown in Figure 1, after collecting the data step, those features that were more significant had to be identified so as to obtain a classification model that assessed the target variable. This stage, named feature selection, was a procedure that determined the contribution of each feature, identifying the higher influences when a classification or prediction was done. By performing a feature selection before building the model, the overfitting was reduced, improving the accuracy and, therefore, the training time.

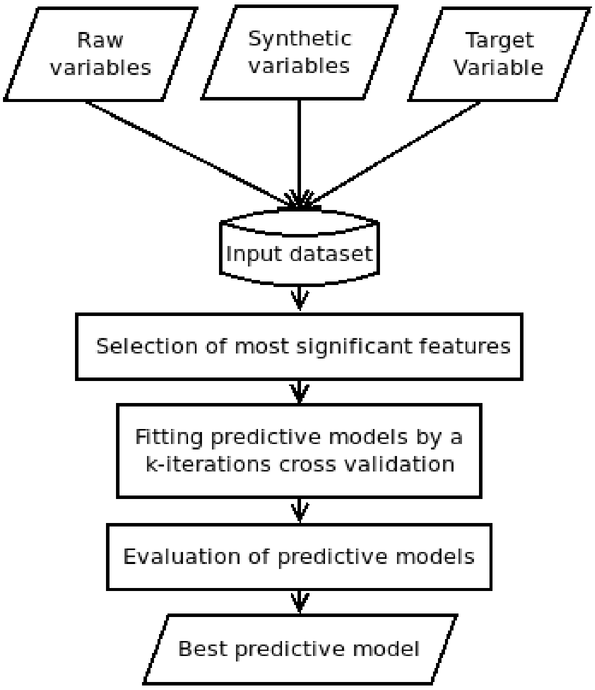
Raw variables
Synthetic variables
Target Variable
Input dataset
Selection of most significant features
Fitting predictive models by a k-iterations cross validation
Evaluation of predictive models
Best predictive model

Figure 1. MoEv operation scheme as shown in [5].

According to the scores presented in [14], the tree-base algorithm achieved good rates for the database considered in this paper. Thus, the feature selection was computed by employing an extremely randomized tree [18]. The decision criterion to select a feature was based on the value of the Gini coefficient ($\mathcal{G}$) computed for each feature; if $\mathcal{G}$ was higher than a certain threshold, the feature was then significant; in this paper, the threshold value was fixed to 0.1, obtained empirically.

After the feature selection, several prediction models were fitted. We used 80% of the input data to train the models and 20% to test them. The following Machine Learning (ML) algorithms were considered since they are well-known in this field:

- Adaptive Boosting (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.
- Classification And Regression Tree (CART). A decision tree is a method that 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.
- K-Nearest Neighbors (KNN). Although the nearest neighbor 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 that classifies new observations based on the distance to the observation in the training set. A good presentation of the analysis was given in [20,21].
- Linear Discriminant Analysis (LDA). This is a parametric method that assumes that distributions of the data are multivariate Gaussian [21]. Furthermore, 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 [22–24].

- Logistic Regression (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.
- Multi-Layer Perceptron (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. The network is organized into layers of nodes (an input layer, an output layer, and one or more hidden layers). These layers are organized in a directed graph so that each layer is fully connected to the next one. An MLP is a modification of the standard linear perceptron whose best characteristic is that it is able to distinguish data that are not linearly separable. An MLP uses back-propagation for training the network; see [25,26].
- Naive Bayes (NB). This method is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features; see [21,27].
- Random Forest (RF). This is a 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].

Finally, experiments were carried out with the MoEv tool that supports k-iteration cross-validation so as to identify the prediction algorithm more appropriate for a given problem. To perform it, MoEv calculated a set of well-known Key Performance Indicators (KPIs). Each model performance was measured by considering the accuracy of classification (see Equation (1)):

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

T_p being the number of students who pass the course and the model predicts so (true positives) and T_n the number of students who do not pass the course and the model predicts so (true negatives).

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

The Precision (P) is computed as follows (Equation (2)):

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

F_p being the number of students who do not pass the course and the model says they pass (false positives).

The Recall (R) is defined in Equation (3), where F_n is the number of students who pass the course and the model says they fail (false negatives):

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

The F_1-score relates those metrics since it is the harmonic mean of precision and recall, as shown in Equation (4).

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

3. Results

Feature importance was computed with the Gini coefficient, as is shown in Figure 2. Features with a low Gini coefficient were discarded ($\mathcal{G} < 0.1$). Selected features were days ($\mathcal{G} = 0.27$), lines ($\mathcal{G} = 0.24$), additions ($\mathcal{G} = 0.16$), commits ($\mathcal{G} = 0.14$), and files ($\mathcal{G} = 0.1$).

Table 2 gathers the accuracy calculated by using a 10-iteration execution of MoEv. As shown in the table, the best results were obtained with RF (0.78), KNN (0.72), and AB (0.7).

In Figure 3, the confusion matrices for RF, KNN, and AB models are presented.

Precision (P), Recall (R), and F_1-score obtained through the confusion matrix from Figure 3 are gathered in Table 3.

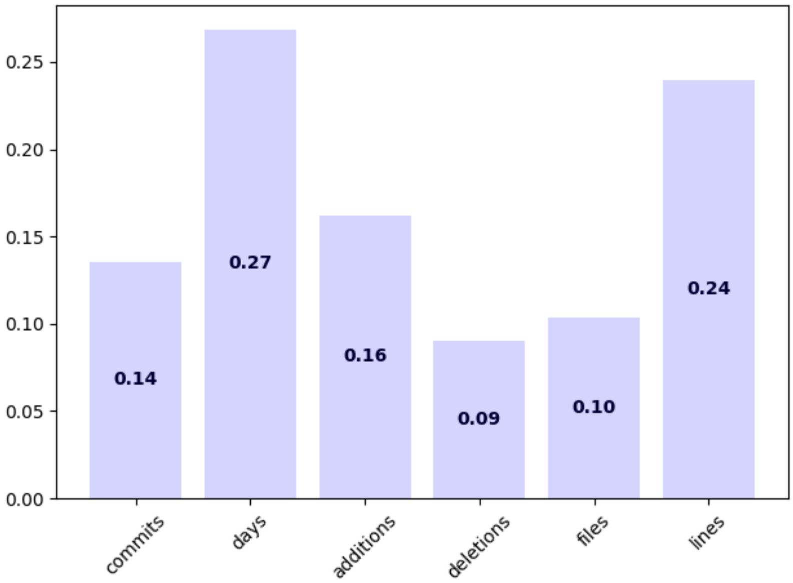

Figure 2. Feature importance.

Table 2. Accuracy classification scores.

	Classifier							
	RF	**KNN**	**AB**	**LR**	**LDA**	**CART**	**NB**	**MLP**
Score	**0.78**	**0.72**	**0.70**	0.67	0.67	0.67	0.63	0.63

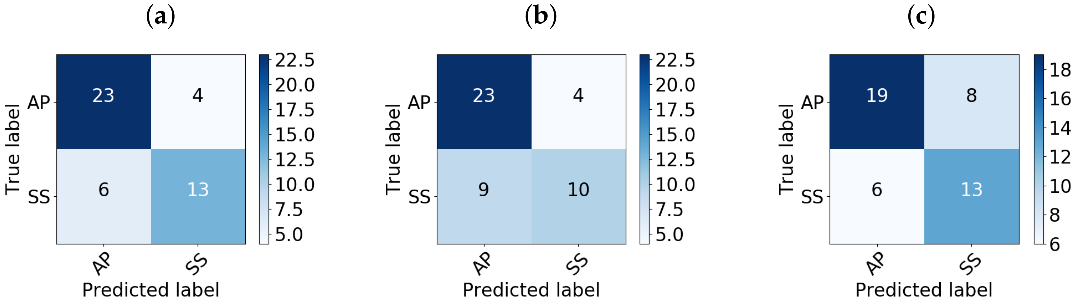

Figure 3. Confusion matrix for RF (**a**), KNN (**b**), and AB (**c**) models.

Table 3. Precision, recall and F_1-score for the RF, KNN, and AB models.

Classifier	Class	P	R	F_1-Score	Support
RF	P	0.79	0.85	0.82	27
	F	0.76	0.68	0.82	19
	avg/total	0.78	0.78	0.78	46
KNN	P	0.72	0.85	0.78	27
	F	0.71	0.53	0.61	19
	avg/total	0.72	0.72	0.71	46
AB	P	0.76	0.70	0.73	27
	F	0.62	0.68	0.65	19
	avg/total	0.70	0.70	0.79	46

4. Discussion

Feature importance in Figure 2 showed that the number of days with at least one commit operation (days) was the most discriminant feature ($\mathcal{G} = 0.27$). This is logical since working in a software solution usually implies different phases and milestones. A low value in this feature points out that the student had not achieved the expected milestones. In addition to this, the number of days with at least one commit operation had higher importance than the number of commit operations itself; this had to do with the fact that several milestones usually cannot be achieved the same day. Having multiple commit operations the same day usually means that students are using several commit operations for a single milestone. This may happen if there are unexpected errors due to untested software or even that the students are trying to cheat the model.

The next discriminant feature was the number of source code lines ($\mathcal{G} = 0.24$). This allowed us to assert that the contents of a repository may be quite useful to predict academic success. Including more additional features such as the ones proposed in [15] in addition to the number of files and lines may result in higher accuracy. However, it is important to point out that this requires a static analysis of the source code. This analysis is different for each programming language, so it complicates considerably the data collection, although programming paradigms are quite common between languages. This is an important drawback since data collection is actually quite complicated and time demanding. However, the number of programming languages that are used during the Computer Science degree is not so high, so the method presented in this paper can be easily adapted to other subjects.

Regarding the models' accuracy, RF, KNN, and AB had the highest scores, as shown in Table 2. RF was also the model with the highest accuracy score in [5]. Table 4 shows the complete comparison between the accuracy scores obtained in this work shown in Table 2 and the accuracy scores obtained in [5]. The first line at Table 4 shows the accuracy score obtained in this work. The second line shows the accuracy scores obtained in [5]. It is important to point out that all models increased their accuracy scores. This was due to the fact that in this work, the training dataset was much bigger and complete (since it included data from several subjects) compared to the dataset used in [5]. In addition, considering additional significant features also implied an accuracy improvement.

Table 4. Accuracy scores' comparison between the models proposed in this work (first line) and the models proposed in [5] (second line).

	Classifier							
	RF	**KNN**	**AB**	**LR**	**LDA**	**CART**	**NB**	**MLP**
Score of the models proposed at this work	0.78	0.72	0.70	0.67	0.67	0.67	0.63	0.63
Score of the models proposed in [5]	0.70	0.60	0.40	0.60	0.60	0.60	0.50	0.40

Finally, considering the most accurate models, i.e., RF, KNN, and AB, a deeper analysis through their confusion matrices was given in order to analyze their sensitivity for detecting Ps: i.e., the rate of true Ps that the model classified incorrectly. Figure 3 and Table 3 show that the RF classifier obtained better values for precision (P), recall (R), and F_1-score than KNN and AB in both the test and validation stages.

5. Conclusions

The methodology presented in this paper allowed lecturers to predict the results students achieved by supervising their daily performance using a VCS. The employed models were fitted by using a dataset that gathered data from different subjects of the Computer Science degree at the University of León. All the considered subjects had two common items: all of them required students to develop software solutions and to use a VCS for storing source code and documentation.

The dataset included variables from the students' interaction with VCSs such as the number of commit operations, source code lines added and deleted, etc. In addition, the dataset included data related to the contents of students' repositories such as the number of source code files and lines. These additional variables, although not coming from the interaction with VCS, were source code features; thus, the models ensured an optimal generalization.

The results presented in Section 3 and discussed in Section 4 demonstrated that the models' accuracy scores increased by using a bigger dataset, not only including additional variables, but also much more samples. The accuracy was also similar to the results obtained with models trained with datasets containing specific features, and thus, and contrary to these models, the ones proposed in this paper ensured an optimal generalization.

Further works should face the usage of additional features collected within the source code itself to analyze the quality of the source code and, thus, study the evolution of students' programming skills.

Author Contributions: Conceptualization, Á.M.G.H.; methodology, Á.M.G.H., C.F.L., L.S.G., and M.Á.C.; software, Á.M.G.H., A.G.F., and G.E.C.; validation, C.F.L., L.S.G., and M.Á.C.; investigation, Á.M.G.H., C.F.L., and L.S.G.; resources, C.F.L.; data curation, A.G.F., and G.E.C.; writing—original draft preparation, Á.M.G.H. and L.S.G.; writing—review and editing, C.F.L., A.G.F., G.E.C., and M.Á.C.; visualization, Á.M.G.H.; supervision, M.Á.C.; project administration, Á.M.G.H.; funding acquisition, C.F.L. All authors read and agreed to the published version of the manuscript.

Funding: The research described in this article was partially funded by the Support Plan for Teaching Innovation Groups (PAGID) of the University of León 2018 and by the European Commission under Grant 2018-1-ES01-KA201-050939: ROBOSTEAM–Integrating Steam and Computational Thinking development by using Robotics and Physical Devices.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

Abbreviations

List of abbreviations used in this paper:

AB	Adaptive Boosting
CART	Classification And Regression Tree
ICT	Information and Communications Technology
KNN	K-Nearest Neighbors
KPI	Key Performance Indicator
LA	Learning Analytics
LDA	Linear Discriminant Analysis
LMS	Learning Management System
LR	Logistic Regression
ML	Machine Learning

MLP	Multi-Layer Perceptron
MoEv	Module Evaluator
NB	Naive Bayes
RF	Random Forest
SIS	Student Institutional System
VCS	Version Control System

References

1. Siemens, G.; Gasevic, D. Guest editorial-Learning and knowledge analytics. *Educ. Technol. Soc.* **2012**, *15*, 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; Society for Learning Analytics Research for the Australian Office for Learning and Teaching: Canberra, Australia, 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. *Internet High. Educ.* **2016**, *28*, 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, Salamanca, Spain, 24–26 October 2018; pp. 306–312.
5. Guerrero-Higueras, Á.M.; DeCastro-García, N.; Rodriguez-Lera, F.J.; Matellán, V.; Conde, M.Á. Predicting academic success through students' interaction with Version Control Systems. *Open Comput. Sci.* **2019**, *9*, 243–251.
6. Guerrero-Higueras, Á.M.; DeCastro-García, N.; Matellán, V. Detection of Cyber-attacks to indoor real time localization systems for autonomous robots. *Robot. Auton. Syst.* **2018**, *99*, 75–83.
7. Kovacic, Z. Predicting student success by mining enrolment data. *Res. High. Educ. J.* **2012**, *15*, 1–20.
8. 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. *Comput. Hum. Behav.* **2014**, *31*, 542–550.
9. Barber, R.; Sharkey, M. Course correction: Using analytics to predict course success. In Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, Vancouver, BC, Canada, 29 April–2 May 2012; pp. 259–262.
10. Spinellis, D. Version control systems. *IEEE Softw.* **2005**, *22*, 108–109.
11. Fischer, M.; Pinzger, M.; Gall, H. Populating a release history database from version control and bug tracking systems. In Proceedings of the International Conference on Software Maintenance, Amsterdam, The Netherlands, 22–26 September 2003; pp. 23–32.
12. Pilato, C.M.; Collins-Sussman, B.; Fitzpatrick, B.W. *Version Control with Subversion: Next Generation Open Source Version Control*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2008.
13. Torvalds, L.; Hamano, J. Git: Fast Version Control System. Available online: http://git-scm.com (accessed on February 21st 2020).
14. 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 Proceedings of the Learning Analytics Summer Institute (LASI), New York, NY, USA, 11–13 June 2018.
15. Guerrero-Higueras, Á.M.; Sánchez-González, L.; Fernández-Llamas, C.; Conde, M.Á.; Lera, F.J.R.; Sedano, F.J.R.; Costales, G.E.; Matellán, V. Prediction of academic success through interaction with version control systems. In Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain, 24–26 October 2018; pp. 284–289.
16. 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, Vancouver, BC, Canada, 17 May 2009; pp. 36–39.
17. Griffin, T.; Seals, S. Github in the classroom: Not just for group projects. *J. Comput. Sci. Coll.* **2013**, *28*, 74–74.
18. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42.
19. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 1.
20. Devroye, L.; Györfi, L.; Lugosi, G. *A Probabilistic Theory of Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 31.

21. Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
22. Bishop, C.M. Pattern recognition. *Mach. Learn.* **2006**, *128*, 1–58.
23. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009.
24. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
25. Rummelhart, D.E. Learning internal representations by error propagation. In *Parallel Distributed Processing*; 1986.
26. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314.
27. Zhang, H. The optimality of naive Bayes. *AA* **2004**, *1*, 3.
28. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32.