Model for Evaluating Student Performance Through Their Interaction With Version Control Systems

Ángel Manuel Guerrero-Higueras[1], Vicente Matellán-Olivera[3], Gonzalo Esteban Costales[1], Camino Fernández-Llamas[2], Francisco Jesús Rodríguez-Sedano[2], and Miguel Ángel Conde[2]

[1] Research Institute on Applied Sciences in Cybersecurity (RIASC). Universidad de León, Av. de los Jesuitas s/n. ES-24008 León (Spain).
am.guerrero@unileon.es

[2] Robotics Group. Universidad de León, Av. de los Jesuitas s/n. ES-24008 León (Spain).
{camino.fernandez,fjrods,mcong}@unileon.es

[3] Supercomputación de Castilla y León (SCAYLE), Campus de Vegazana s/n, ES.24071 León (Spain).
vicente.matellan@fcsc.es

Abstract. Version Control Systems are commonly used for Information and communication technology professionals. They also allows to follow the activity of a single programmer working in a project. For these reasons, Version Control Systems are also used by educational institutions. The aim of this work is to demonstrate that the student performance may be evaluated, and even predicted, by monitoring their interaction with a Version Control System. In order to do so we have build a Machine Learnings model to predict student results in a specific task 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.

Keywords: Version Control System, Machine Learning, Learning analytics.

1 Introduction

The emergence of the Information and Communication Technologies have change the landscape of 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 application to learn 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 to this, and this issue is specially link to trends such as Learning Analytics and Educational Data Mining.

The most accepted definition of learning analytics considers that it comprises "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" [1]. Learning analytics facilitates discovery of "hidden" knowledge about teaching and learning processes (see [2,3]). Therefore, the use of learning analytics allows learners and instructors to obtain and visualize information about different issues and between them the suitability of contents and/or tools and their impact in students' performance [4]. Educational institutions and instructors could use the information obtained by applying these techniques to make changes in the courses in order to improve the whole learning process and experience [5].

In this case the idea is to explore how students' performance is affected by the use of Version Control Systems (VCSs). VCSs facilitate the management of changes in the components of a software product or its configuration[6]. The version, release or edition is the state of this product in a specific moment. But why to use such tools? This is because it is a high demanded tool for future computer science engineers and it is introduced as a tool of several Computer Science Subjects.

The aim of this work is to build a model that allows to predict student results at a practical assignment by monitoring their use of a VCS. We assume the premise that the students activity with this type of systems is an indicator of the evolution of their progress.

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.

2 Materials and Methods

This section describes all the elements and the methodology used to build and evaluate the model for predicting student results. Among the elements used there are a specific practical assignment to provide student results, and a VCS. Regarding the methodology, a set of classifying algorithms have been evaluated by analysing some well-known Key Performance Indicators (KPIs).

2.1 Practical assigment: ASSOOFS

The Ampliación de Sistemas Operativos (ASSOO) subject from the second course of the degree in Computer Science of the University of León 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.

Main practical assignment consists on 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 [7] 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.

2.2 GitHub Classrrom

In software engineering, it is known as control of versions to the management of the changes that are made on the elements of some product [6]. 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 VCSs [8]. Among the most popular there are the following: CVS, Subversion [9] or Git [10].

A VCS must provide, at least, the following features:

- Storage for the different elements to be managed (source code, images, documentation).
- Edition the stored elements (creation, deletion, modification, renaming, etc.).
- Registration and labelling of all actions carried out, of so that they allow an element to be returned to a state previous.

For the development of ASSOOFS, 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 [11].

In order to provide some organizing capabilities and private repositories for students the GitHub Classroom platform was used [12]. GitHub is a web-based hosting service for software development projects that utilize the Git revision control system. In addition, GitHub Classroom allows to assign tasks to students, or groups of students, framed in the same centralized organization: ASSOO students in our case.

Features Regarding the input data to predict results, usually called features in a Machine Learning (ML) context, we have considered the following information coming from students activity on their repositories:

- *Commits*: total number of commit operations carried by the student.
- *#Days with commit operations*: total number of days where there is at least one commit operation.
- *Commits/date*: average number of commit operations per date.

- *Additions*: number of lines of code added during the assignment completion.
- *Deletions*: number of lines deleted during the assignment completion.

In addition to the above data, all obtained from the GitHub Classroom platform, we have also considered the students grade on a proof carried out to control the authorship of the code in student repositories. This authorship proof allows 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.

Input data explained above will be used by the model to predict a class: AP, for those students who will finish the practical assignment successfully; and SS, for those who not.

2.3 Model

We want to generate a model whose inputs are quantitative, while its output is a discrete value: AP, and SS. Two types of ML algorithms may be used: classifiers and predictors, whereby considering the first ones will be better. We have evaluated 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).

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 travelling from a root node of a tree to a leaf [13]. It is a non-parametric method in which the trees are grown in an iterative, top-down process.

KNN Although nearest neighbours is the foundation of many other learning methods, notably unsupervised, supervised neighbour-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 [14] and [15].

LDA Parametric method that assumes that distributions of the data are multivariate Gaussian [15]. 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 [16], [17] or [18]).

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 (input layer, output layer and hidden layers). An MLP consists of multiple layers of nodes 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 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 [19] and [20].

NB This method is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features, see [15] and [21].

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 boostrap re-sampling [22].

To evaluate the previous methods, the implementation of the Scikit-learn library has been used [23].

2.4 Methodology

In order to train the models we have use the data obtained by the ASSOO students from de 2016–2017 course presented at [24]. These data includes the features mentioned at section 2.2 for the 46 students who tried the ASSOOFS assignment. We carried out 2 kind of analysis: in the first one we do not include the authorship proof as an input feature; in the second one, we do.

To evaluate the above algorithms with there input data, we have followed the method proposed at [25] to select the model which better fits our problem. The method proposes a 10-iteration cross-validation 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 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

Table 1–left shows the accuracy classification score for all evaluated models without consider the authorship proof as an input feature. The highest scores for the validation dataset are highlighted in bold. Table 1–right show Precision (P), Recall (R), and F_1-score for the highlighted models: RF, CART, and LR.

Table 1. Accuracy classification score without consider the authorship proof (left) and Precision (P), Recall (R), and F_1-score for highlighted models (right).

Model	**Score**
RF	**0.8**
CART	**0.7**
LR	**0.6**
LDA	0.6
KNN	0.6
AB	0.6
MLP	0.6
NB	0.5

Classifier	**Class**	P	R	F_1**-score**	**#examples**
RF	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg.	0.87	0.80	0.80	10
CART	AP	0.60	0.75	0.67	4
	SS	0.80	0.67	0.73	6
	avg.	0.72	0.70	0.70	10
LR	AP	0.50	1.00	0.67	4
	SS	1.00	0.33	0.50	6
	avg.	0.80	0.60	0.57	10

Fig 1 shows the confusion matrix computed from the highlighted models: RF, CART, and LR.

Table 2–left shows the accuracy classification score for all evaluated models considering the authorship proof as an input feature. The highest scores for the validation dataset are highlighted in bold. Table 1–right show Precision (P), Recall (R), and F_1-score for or the highlighted models: RF, LR, and NB.

Fig 2 shows the confusion matrix computed from the highlighted models: RF, LR, and NB.

4 Discussion

According to the above results, as shown at Table 1–left, RF classifier works better (accuracy score = 0.8) than any other for selected features, in this case:

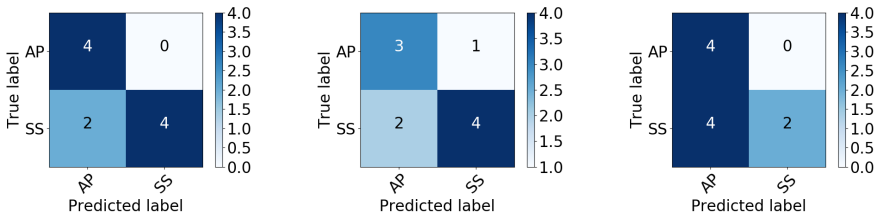

Fig. 1. Confusion matrix for the RF (left), CART (center), and LR (right) classifier.

Table 2. Accuracy classification score considering the authorship proof (left) and Precision (P), Recall (R), and F_1-score for highlighted models (right).

Model	Score
RF	**0.9**
LR	**0.8**
NB	**0.8**
LDA	0.7
KNN	0.6
MLP	0.6
CART	0.5
AB	0.5

Classifier	Class	P	R	F_1-score	#examples
RF	AP	0.80	1.00	0.89	4
	SS	1.00	0.83	0.91	6
	avg.	0.92	0.90	0.90	10
LR	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg.	0.87	0.80	0.80	10
NB	AP	0.67	1.00	0.80	4
	SS	1.00	0.67	0.80	6
	avg.	0.87	0.80	0.80	10

Commits, #days with commit operations, commits/date, additions, and deletions. CART classifier works slightly worse (accuracy score = 0.7) than RF, while all the other classifiers offer very poor results.

Once the best models are pre-selected, a deeper analysis with the confusion matrix of each one is given. Another important item that should be analysed is the sensitivity of the model for detecting a passed assignment (AP): i.e., the rate of APs that the model classifies incorrectly. Table 1–right and Fig 1, show that the RF classifier gets better average values for Precision (P), Recall (R) and F_1-score than CART and LR.

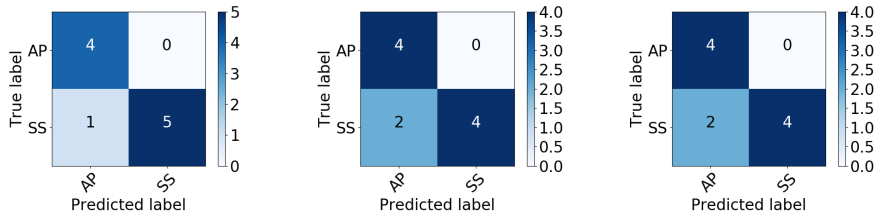

Fig. 2. Confusion matrix for the RF, (left) LR (center), and NB (right) classifier.

Table 2–left shows the results by considering an additional feature: the result of an authorship proof made by students. Accuracy is clearly better considering this feature. Again, it is the RF classifier the one with best results (accuracy score = 0.9). LR and NB classifiers work slightly worse, both with a 0.8 accuracy. All the other classifiers work worse. Regarding the sensitivity of the three bests models for detecting a passed assignment (AP), Table 2–right and Fig 2, show that the RF classifier gets better average values for precision (P), recall (R) and F_1-score than LR and NB.

It is important to note that models have been trained with a small dataset. We have data for just 46 students. Results might change considerably with a bigger dataset. ASSOO students of the 2017–2018 course, have done the same practical assignment, so we plan to repeat the analysis with a bigger dataset when 2017–2018 students finish their work.

5 Conclusions

This work aim to build a model to predict students results by monitoring their activity at VCSs. We start from the premise that analysing the students activity at VCSs allows to predict their results.

To build the model several classifiers have been evaluated. In addition to select the best classifier, we have demonstrated that our premise is true due to the fact that we can predict the students results with a success high percentage. However, the models were evaluated using a small dataset. It would be desirable to get a larger volume of data to perform the analysis.

Regarding the chosen features, we observe that in addition to consider the repository activity, adding an authoring proof helps to increase the accuracy.

Future work will be related to the tuning the hyper-parameters of models in order to obtain better results. In addition, we need to increase de training dataset.

References

1. Siemens, G., Long, P.: Penetrating the fog: Analytics in learning and education. EDUCAUSE review **46**(5), 30 (2011)
2. Hernández-García, Á., González-González, I., Jiménez-Zarco, A.I., Chaparro-Peláez, J.: Applying social learning analytics to message boards in online distance learning: A case study. Computers in Human Behavior **47**, 68–80 (2015)
3. 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**, 542–550 (2014)
4. Conde, M.Á., Hernández-García, Á., Oliveira, A.: Endless horizons?: addressing current concerns about learning analytics. In: Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality. pp. 259–262. ACM (2015)

5. Conde, M.Á., Hernández-García, Á.: Learning analytics for educational decision making. Computers in Human Behavior (47), 1–3 (2015)
6. 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. pp. 23–32. IEEE (2003)
7. Corbet, J., Rubini, A., Kroah-Hartman, G.: Linux Device Drivers: Where the Kernel Meets the Hardware. " O'Reilly Media, Inc." (2005)
8. Spinellis, D.: Version control systems. IEEE Software **22**(5), 108–109 (2005)
9. Pilato, C.M., Collins-Sussman, B., Fitzpatrick, B.W.: Version Control with Subversion: Next Generation Open Source Version Control. " O'Reilly Media, Inc." (2008)
10. Torvalds, L., Hamano, J.: Git: Fast version control system. http://git-scm.com (2010)
11. 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. pp. 36–39. IEEE Computer Society (2009)
12. Griffin, T., Seals, S.: Github in the classroom: Not just for group projects. Journal of Computing Sciences in Colleges **28**(4), 74–74 (2013)
13. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning Ed. 2, vol. 1. Springer series in statistics Springer, Berlin (2009)
14. Devroye, L., Györfi, L., Lugosi, G.: A probabilistic theory of pattern recognition, vol. 31. Springer Science & Business Media (2013)
15. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. John Wiley & Sons (2012)
16. Bishop, C.M.: Pattern recognition. Machine Learning **128**, 1–58 (2006)
17. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
18. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
19. Rummelhart, D.E.: Learning internal representations by error propagation. Parallel distributed processing (1986)
20. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) **2**(4), 303–314 (1989)
21. Zhang, H.: The optimality of naive bayes. AA **1**(2), 3 (2004)
22. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
23. 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**, 2825–2830 (2011)
24. Guerrero-Higueras, Á.M., Conde, M.Á., Matellán, V.: Using version control systems to apply peer review techniques in engineering education. In: IV Congreso Internacional sobre Aprendizaje, Innovación y Competitividad (CINAIC) (2017)
25. 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**, 75–83 (2018)