computers

MDPI

Article

On Predicting Exam Performance Using Version Control Systems' Features

Lorenzo Canale [1,2,†], Luca Cagliero [1,*,†], Laura Farinetti [1,†] and Marco Torchiano [1,†]

[1] Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy; laura.farinetti@polito.it (L.F.)
[2] Centre for Research and Technological Innovation, Radiotelevisione Italiana (RAI), Via Giovanni Carlo Cavalli 6, 10129 Torino, Italy
* Correspondence: luca.cagliero@polito.it; Tel.: +39-011-090-7179
† These authors contributed equally to this work.

Abstract: The advent of Version Control Systems (VCS) in computer science education has significantly improved the learning experience. The Learning Analytics community has started to analyze the interactions between students and VCSs to evaluate the behavioral and cognitive aspects of the learning process. Within the aforesaid scope, a promising research direction is the use of Artificial Intelligence (AI) to predict students' exam outcomes early based on VCS usage data. Previous AI-based solutions have two main drawbacks: (i) They rely on static models, which disregard temporal changes in the student–VCS interactions. (ii) AI reasoning is not transparent to end-users. This paper proposes a time-dependent approach to early predict student performance from VCS data. It applies and compares different classification models trained at various course stages. To gain insights into exam performance predictions it combines classification with explainable AI techniques. It visualizes the explanations of the time-varying performance predictors. The results of a real case study show that the effect of VCS-based features on the exam success rate is relevant much earlier than the end of the course, whereas the timely submission of the first lab assignment is a reliable predictor of the exam grade.

Keywords: version control systems; machine learning; explainable AI; learning analytics

Citation: Canale, L.; Cagliero, L.; Farinetti, L.; Torchiano, M. On Predicting Exam Performance Using Version Control Systems' Features. *Computers* **2024**, *13*, 150. https://doi.org/10.3390/computers13060150

Academic Editor: Paolo Bellavista

Received: 12 May 2024
Revised: 26 May 2024
Accepted: 3 June 2024
Published: 9 June 2024

1. Introduction

Version control systems (VCS) are tools aimed at helping teams of software developers to manage code changes. In computer science education, the use of VCSs is established. For example, the GitHub collaborative platform has shown to enhance the learning experience under multiple aspects [1]. In university-level computer science courses, VCSs have gained importance as collaboration platforms in the development of application software projects: they not only support the learning and evaluation processes but also foster students' participation and ease the collaboration between team members [2,3].

Statistical analyses of the interactions between students and VCSs have shown that VCS-based data are effective in characterizing the relationship between the utilization of VCSs and the exam grades. In this regard, various VCS-based features have already been considered, e.g., the selection proposed by [4]. They incorporate information relevant to both behavioral and cognitive engagement levels of the student [5].

This paper analyzes the activities of the students of a university-level Object-Oriented Programming course by monitoring their interactions with a VCS. The purpose is to predict various exam performance indicators early such as the exam success rate, the exam grade, the lab dropout rate (when a student stops attending the laboratory activities from a certain point on), and the number of exam attempts prior to passing the exam. The purpose is to allow timely interventions focused on at-risk situations [6].

Previous works on early prediction of students' performance based on VCS usage data presented by [7,8] have two main drawbacks:

- *Time invariance.* Existing ML models are commonly trained once on the complete set of VCS usage statistics. However, since VCS data are time-evolving, there is a need for time-dependent model analyses and comparisons to allow early prediction of the students' exam performance well before the end of the course.
- *Limited explainability*. AI often produces non-transparent outcomes, which cannot be explained to the teacher. This can lead to trust issues when predictions fail. To overcome this issue, there is a need to involve human experts in the decision-making process.

To address the aforesaid challenges, the present work presents the following contributions.

- *Time-dependent ML model training.* Computer science courses typically entail a sequence of assignments and tests. VCSs continuously trace students' activities on log files throughout the course. The VCS usage data collected in the first part of the course is likely to be correlated with the students' performance. Hence, we investigate the suitability of intermediate VCS usage data to perform early student performance prediction. To this end, we re-train the classifiers on the available statistics to dynamically update the predictive models depending on the availability of up-to-date information.
- *Visual classifier explanation.* We compare the classification models by exploiting a state-of-the-art explainable AI method, namely Shapley Additive explanations (SHAP) [9]. SHAP provides teachers with a visual interpretation of the time-dependent impact of the VCS-based feature values taken by a student on the exam performance predictions.

We carried out an empirical analysis of the presented methodology on real VCS usage data collected during an Object-Oriented Programming course. The main results are summarized below:

- *Predictive power of early VCS-based models.* The VCS usage patterns identified at the very preliminary course stages are, to a good approximation, accurate predictors of the students' exam performance.
- *Negative impact of lab dropout.* The dropout of a student from the course laboratories can be detected by monitoring the VCS commits. The attendance of laboratory activities has shown to be highly discriminating of both exam success rate and grade. Stopping laboratory attendance in the first part of the course has shown to negatively affect exam performance.
- *Positive impact of exam attempt.* The a priori knowledge of the outcomes of past exam sessions has shown to enhance the quality of classifier predictions provided that VCS-based features are considered in the ML model as well.
- *Correlation between exam grades and laboratory attendance.* To obtain high grades, students usually need to attend most of the course laboratories. The attendance to the laboratories at the beginning of the course has shown to be not sufficient to achieve high grades.

The source code of our methodology is available at https://github.com/Loricanal/VESPE (accessed on 1 June 2024). The source data are available for research purposes upon request to the authors.

A paper outline follows. Section 2 provides a literature review. Section 3 introduces the research context. Section 4 presents the proposed architecture. Section 5 discusses the main results. Finally, Section 6 concludes the work and discusses the future research agenda.

2. Literature Review

The learning analytics community has already explored the use of Version Control Systems in education. For example, Refs. [2,3] studied the role of GitHub (https://github.com/ (latest access: 1 May 2024)) as a collaborative resource in large software engineering

classes. Thanks to the functionalities provided by the VCS, the majority of the learners have shown to actively contribute to the project through code changes, reviews, and comments. Refs. [10,11] studied the learners' adherence to the iterative software development model as well as the extent to which students interact with each other even in the absence of specific commitments. They show the benefits of using VCSs for individual learning activities.

The correlation between the utilization of VCSs and the exam grades was studied by [4,7,12,13]. They carried out statistical analyses to verify the dependence between a selection of VCS-based features and learners' outcomes. For example, the number and frequency of GitHub commits have shown to be correlated with the exam outcomes.

Parallel attempts to use Machine Learning techniques on top of VCS-based features have been made. For example, Refs. [8,14–16] extracted from the VCS log files a subset of features summarizing the most relevant students' interactions with VCS, collected them into a training dataset, and trained a classification model. The classifier is exploited to predict students' performance. The main limitations of the existing ML-based solutions are (i) the time-invariance of the trained models, which disregard the temporal evolution of the learner's interactions with the VCS, and (ii) the limited interpretability of the trained models, which hinders an in-depth analysis of the ML reasoning.

Position of the present work. We address the limitations of the previous works as follows: (i) We study how the impact of VCS-based features on ML model performance changes over time as a result of the temporal changes in the student–VCS interactions. (ii) We generate visual explanations of the classifiers using an established explainable AI method [9]. The provided visualizations allow for a detailed comparison between the models trained at different course stages.

3. Context of Research

The research presented in this paper draws its data from an object-oriented (OOP) programming course in the second year of the Computer Engineering B.Sc. degree. held every year in spring. The course consists of 70 taught hours and 20 h of laboratory activities devoted to the development of programming assignments. The language of choice for the course is Java (currently, version 8 is used). While previous programming courses in the curriculum adopted paper-based exams, in this one, the teachers opted for a computer-based exam to improve students' practice. All the exercises and submissions are conducted in the laboratory under the teacher's supervision.

Laboratory practices are meant to develop software testing and software configuration skills in compliance with SWECOM 1.0 [17]. Moreover, automated testing appears particularly suited to respond to the Millennials' need for frequent feedback [18]. The tests are performed using the JUnit 5 automated testing framework. Test cases are written as methods within Java classes. Such methods include statements that exercise the code under tests and assertions that check the actual return values vs. the expected ones. The JUnit framework executes all test cases, and for each one, it collects the outcome, i.e., Success (correct execution and all assertions verified), Failure (correct execution but at least one assertion not verified), or Error (incorrect execution). A detailed description of the key practices used in laboratory is given in [19].

4. The Presented Methodology

We present here the Visual Explainable Student performance PrEdictor (VESPE), a machine learning-based method aimed at predicting students' exam performance based on VCS-based features. VESPE allows end-users to apply ML models trained at different course stages and compare them by exploiting ad hoc visual explanations. VESPE end-users are teachers/educators, mainly from the Computer Science area because a minimum level of data and computational literacy are required.

VESPE supports the integration of an arbitrary set of student-related features. They include both time-dependent VCS-based features (e.g., the number of commits) and time-invariant ones (e.g., year of enrollment). Furthermore, it integrates a variety of prediction

targets related to the students' exam performance (e.g., exam success, exam grade). Our purpose is to study the impact of VCS-based features on exam performance. Hence, hereafter we will disregard the time-invariant features as their analysis is out of the scope of the present work.

Figure 1 shows the architecture of the proposed methodology where the end-user can feed VCS data, specify prediction targets and related options and then the system produces the time-varying predictions and visualizes the model explanations. A more detailed description of each block follows.

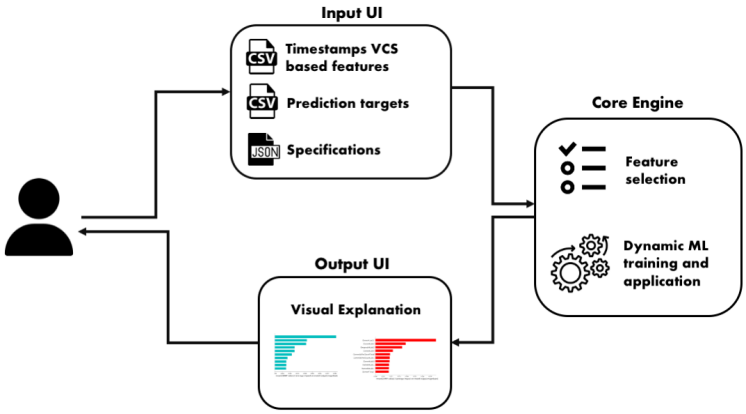


Figure 1. The VESPE architecture.

4.1. Input UI

4.1.1. VCS Usage Data

The input VCS usage data are stored in a relational dataset [20]. Each record corresponds to a different student whereas each attribute is associated with a distinct feature. Analogously, the target dataset stores the values of the target classes for each student in separate attributes. Hence, the target of the prediction could be dynamically set up according to the end-user goals.

4.1.2. Feature Engineering

We focus on the activities carried out by the students on the VCS and related to the course laboratories. The considered features and the corresponding categorizations are reported in Tables 1 and 2. They include both VCS-based feature categories considered in the previous works and newly proposed ones. Specifically, the features related to assignment submissions and commits have already been considered in past research studies. For instance, Ref. [8] carried out an in-depth feature analysis. Conversely, to the best of our knowledge, categories Lab dropout and Exam attempts are new. A complementary goal of our research work is to understand whether the newly proposed VCS-based feature categories are correlated with students' exam performance.

Lab dropout indicates whether a student stops attending the laboratory (Dropout N indicates that the student stopped attending the course after the N-th laboratory). It is an indicator of students' engagement level and, in particular, it is related to both persistence and completion. Notice that stopping lab attendance does not necessarily imply course dropout [21].

Exam attempts indicate the number of exam attempts made by a student. We recall that a student could attend the official exam of a course up to four times per year. Repeated exam attempts can be due to either exam failures or to the rejection of the exam grade.

Table 1. Description of the VCS features extracted from [22].

Feature	Description
Commit numbers ($D = 6$)	
CommitCount#Lab {1,2,3,4,5}	Number of commits for each lab
CommitCount#Total	Total number of commits
Commit quality ($D = 18$)	
Passed#Lab{1,2,3,4,5}	Number of tests passed for each lab
Passed#Total	Total number of passed test
Error#Lab{1,2,3,4,5}	Number of tests that raised errors for each lab
Error#Total	Total number of tests that raised errors
Failed#Lab{1,2,3,4,5}	Number of tests failed for each lab
Failed#Total	Total number of failed tests
Commit frequency ($D = 6$)	
CommitsPerDay#Lab{1,2,3,4,5}	Average number of commits per day for each lab
CommitsPerDay#Total	Average number of commits per day
Active days ($D = 6$)	
ActiveDays#Lab{1,2,3,4,5}	Number of days in which at least 1 commit was carried out for each lab
ActiveDays#Total	Total number of days in which at least 1 commit was carried out
Submitted labs ($D = 6$)	
Done#Lab{1,2,3,4,5}	Lab were submitted or not
DoneLab#Total	Total number of submitted lab
Dropout ($D = 4$)	
Dropout#Lab{2,3,4,5}	Dropout#LabN: the student submitted labs 1 … N − 1

Table 2. Description of Exam attempts features extracted from [22].

Feature	Description
PreviousAttempt	The feature indicates whether the examination was already attempted by the student in the first session; it takes the value 1 if it was attempted (regardless of the outcome) and takes the value 0 if it was not attempted.
PreviousAttemptPass	The feature indicates whether the examination had already been attempted by the student in the first session and passed (0 if it was not attempted or failed, 1 if it was attempted and passed). In practice, this indicates the students who refused the grade.
PreviousAttemptFail	The feature indicates whether the examination had already been attempted by the student in the first session and failed (0 if not attempted or attempted and passed, 1 if attempted and failed).

4.1.3. Model Training

We train multiple ML models at different time points. Specifically, we train a separate model after each laboratory session (which requires tight student–VCS interactions). Notice that the available feature set changes over time because the values of some of the VCS-based features in Tables 1 and 2 are missing or need updates.

The data used to train the ML algorithms at time t_n contains only the features f_x whose time stamp $t_x \leq t_n$. For instance, let us suppose that lab 1 and lab 2 terminated at times t_1 and t_2, respectively. Then, at time t_x $[t_1 \geq t_x \geq t_2]$, ML training can be run on a dataset describing the activities carried out in lab 1, whereas the features related to lab 2 are not available.

In the performed experiments we considered as reference time stamps t_1, t_2, ..., t_n the end of each lab of the course. Hence, at the end of an arbitrary lab x, we re-train the ML model by using only the features describing the labs from 1 to x.

4.1.4. Prediction Targets

A student who passes the exam receives a grade. Grades are discretized into bins as follows: grade A is a fairly high grade, grade B accounts for students who obtained an intermediate grade, whereas grade C is relative to the students who passed the exam but with a low grade.

According to the exam statistics, almost all students passed the exam. Students who received a high grade (A) are quite numerous in the first two exam session attempts, whereas getting high grades was less likely in Exams 3 and 4. For the latter exam sessions, the number of students giving the exam is too small to have enough data samples to make accurate predictions.

According to the aforesaid statistics, in our experiments we will focus on a subset of prediction targets for which a sufficiently large set of training samples is available:

Specifically, target Success takes values (1) Pass if the student passed the exam, (2) Fail if the student fails all the attempts, or (3) Dropout if the student performs a course dropout. Similarly, targets Success Exam 1 and Success Exam 2, respectively, indicate whether the student passed the exam at the first/second call or not. Target Grade indicates the exam grade A, B, or C (valid only for the students who passed the exam).

4.1.5. Configuration Settings

The system allows end-users to specify the desired configuration settings. The specifications include: (i) a *target type*, which discriminates between different ML model types (i.e., regression for continuous targets, classification for binary/categorical ones); (ii) a *feature categorization*, which partitions the feature set in the VCS usage dataset into homogeneous groups according to their semantic meaning; (iii) a *timeline*, which indicates for each feature the time stamp at which it takes value in the dataset records; and (iv) the algorithm parameters.

4.2. Feature Selection

The VCS-based features are automatically evaluated and filtered prior to ML training according to their correlation with the selected target.

VESPE integrates a variety of feature selection algorithms available in the the Scikit-Learn library [23], including (i) the unsupervised methods based on the Pearson's correlation (Prs) and Spearman's rank coefficient (Spr) and (ii) the supervised approaches based on statistical methods, i.e., ANOVA correlation coefficient (Anv), Chi-Squared (Chi^2), Mutual Information (MI), and recursive feature elimination, i.e., RFE, RFECV.

4.3. Classification Algorithms

We train classification models representative of various families of algorithms such as ensemble methods, decision tree classifiers, SVM classifiers, and Bayesian models. VESPE integrates the implementations provided by the Scikit-Learn library [23]. A detailed descriptions of the VESPE architecture is given in [22].

4.4. Model Explanation

VESPE adopts the SHAP library to visualize the global explanations [9]. The graphical interface allows end-users to quickly access a summarized view of results. A detailed description of the interface can be found in [22].

5. Experimental Results

All experiments were run on a machine with 1.8 GHz Intel® (Santa Clara, CA, USA) Core® i5, 8 GB of RAM and running macOS High Sierra. We aim to explore the following research directions:

1. Estimate when the exam success rate becomes predictable and identify the most discriminating VCS-based features.
2. Assess the predictability of class dropouts.
3. Analyze the impact of laboratory attendance on exam grades.
4. Study the correlation between exam success and previous exam attempts.

5.1. Evaluation Metrics

We evaluate the performance of the classification algorithms using the standard precision, recall, and F1-Score metrics [20]. Since we are mainly interested in evaluating classifiers' performance on the minority classes (e.g., course dropouts or exam failures), we also consider the Balanced Accuracy, defined by the percentage of correctly classified students weighted by per-class frequencies [24].

5.2. When Does the Exam Success Rate Become Predictable?

Figure 2 shows the variation in the performance metrics over time. Notably, the quality of the predictions is fairly high even at the early course stages. This indicates that the attendance to the first laboratories is already discriminating for predicting exam success.

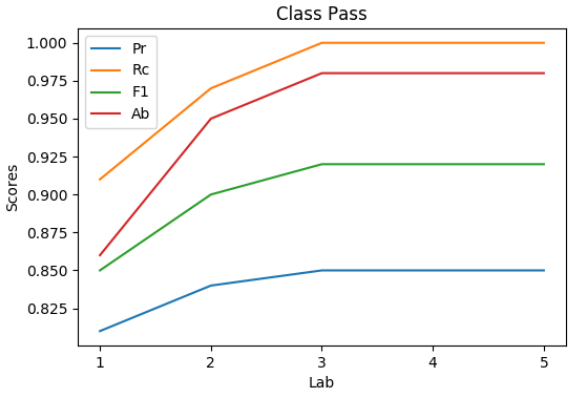
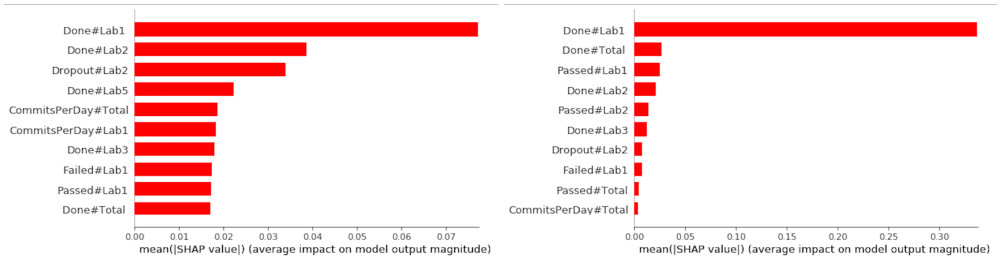


Figure 2. Time dependence of the early predictions for class Pass.

Figure 3 graphically shows the contribution of each Lab feature to the classifier predictions. Feature Done#Lab1 turns out to be the most discriminating one. It indicates that students who attended the first laboratory are likely pass the exam (see Figure 3a). Similar results hold for the target Success Exam 1 (see Figure 3b).

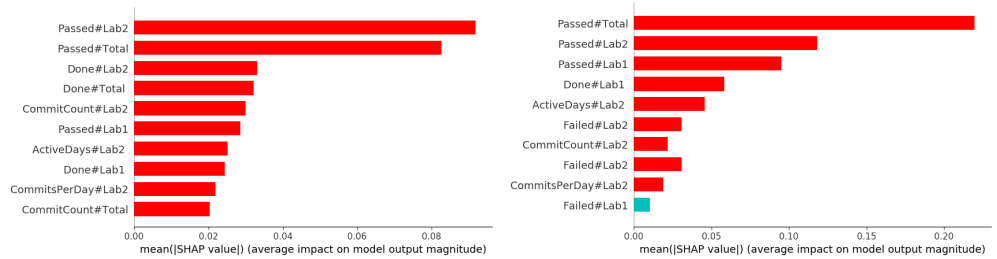
The second most important feature is Done#Lab2, indicating that the attendance to the second laboratory is beneficial as well. Dropout#Lab2 has also an important role. The reason is that Dropout#Lab2 is implicitly related to the student participation in Lab1, which appeared to be the most discriminating feature (Done#Lab1).

Since most of the previous findings pointed out that the first labs were the most significant and the results had shown that the prediction quality was good already after the second lab, we chose to analyze the impact of the features at this time point as well (Figure 4a). The most significant features are related to the commit quality: the higher the number of passed tests (especially in the first lab), the higher the likelihood of passing the exam. The feature types Submitted labs, Commit frequency, and Commit count still have a positive impact on the prediction.



(**a**) Class Pass (**b**) Class Pass Exam 1

Figure 3. Explanations for class Pass and Pass Exam 1.



(**a**) Class Pass (**b**) Class Pass Exam 1

Figure 4. Explanations of Pass and Pass Exam 1. Models trained after the second laboratory.

We can conclude that in the instant of time after the second lab, when only features derivable from the first two labs are used, ML predictions mainly rely on the quality of the student's work. Hence, by identifying students who completed the first two labs and passed the tests, we can accurately identify the students who are likely to pass the exam. Conversely, at the end of all labs, doing the first labs remains significant but it seems also important to add activity on some of the remaining labs.

Figure 5 compares two examples of students after laboratory 2, i.e., an at-risk student and an highly engaged one. The former one submitted just the solution of the first laboratory and failed to pass several tests, whereas the latter passed most of the tests related to the first two laboratories. The teacher can early detect the risky situations and actively involve at-risk students accordingly.

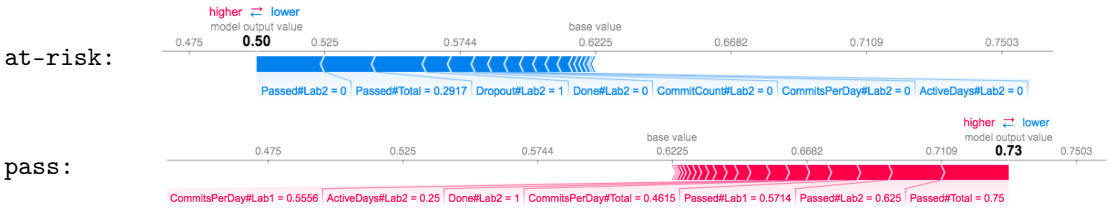


Figure 5. Force plot after lab 2 for two representative students, at-risk and pass, respectively. GNB classifier.

5.3. Predictability of Class Dropout

A Dropout student is deemed as critical because she/he has never attended any course exam. The early identification of these students is particularly useful for optimizing a critical parameter for universities, i.e., the on-time graduation rate.

Figure 6 shows that classifier performance is good well before the end of the course. It increases over time while reaching a steady state after the third course laboratory (when class dropouts become very unlikely).

Figure 7 reports the visual explanations associated with the aforesaid ML models. The main VCS-based features turned out to be negatively correlated with the classifier outcomes. A dropout at labs 2 and 3 (Dropout#Lab2, Dropout#Lab3) negatively affected the prediction. The student has completed at least one lab and it is a sufficient condition

to not be marked as Dropout student in this case study. We can conclude that a student is primarily marked as a dropout when either she/he has not attended any laboratory or she/he attended only the first laboratory with little effort. Figure 8 shows an example of a student classified as Dropout because she/he did not attend the first two laboratories.

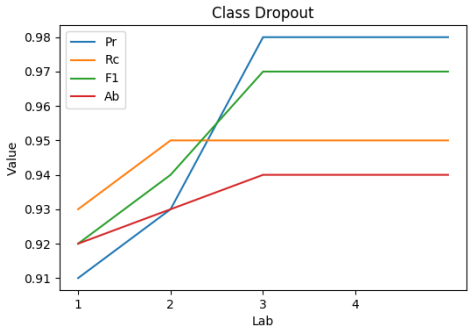


Figure 6. Early prediction of class Dropout based on incomplete feature sets (Dynamic ML). MLP classifier.

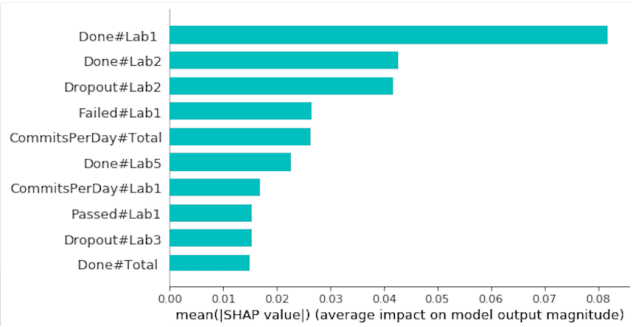


Figure 7. Summary bar plot for class Dropout. MLP classifier.

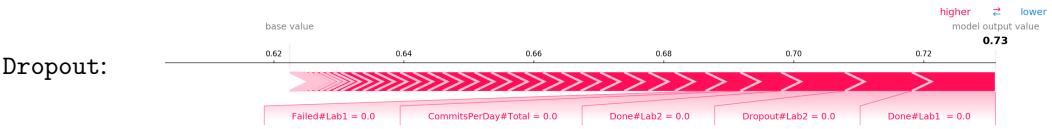


Figure 8. Force plot of a dropout student after laboratory 3. MLP classifier.

5.4. Correlation between Exam Success and Previous Exam Attempts

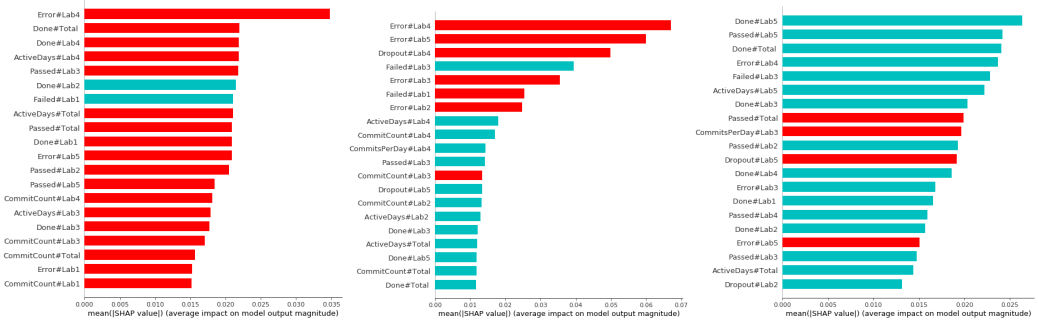
Classifier performance on classes Grade and Grade Exam 1 was fairly low (see Table 3). Unlike exam success and course dropout, VCS-based features have shown to be not very discriminating for the aforesaid targets.

Table 3. Classification performance for class Grade Exam 1.

Features	Alg.	Class Grade C				Class Grade B				Class Grade A				Avg Scores	
		Pr	Rc	F1	Ab	Pr	Rc	F1	Ab	Pr	Rc	F1	Ab	F1	Ab
Commit count	GP	1.00	0.40	0.56	0.70	0.95	0.34	0.49	0.66	0.56	0.98	0.72	0.67	0.59	0.68
Commit quality	SVM	1.00	0.42	0.59	0.71	0.74	0.54	0.59	0.69	0.62	0.86	0.72	0.71	0.63	0.70
Commit frequency	LGR	1.00	0.45	0.61	0.72	0.63	0.54	0.58	0.66	0.62	0.80	0.70	0.69	0.63	0.69
Active days	DT	0.54	0.57	0.55	0.74	0.61	0.60	0.60	0.67	0.63	0.62	0.63	0.66	0.53	0.69
Submitted labs	GP	1.00	0.50	0.66	0.75	0.75	0.35	0.47	0.63	0.57	0.91	0.70	0.67	0.61	0.68
Dropout labs	KN	0.76	0.62	0.59	0.75	0.63	0.69	0.58	0.66	0.83	0.46	0.53	0.65	0.57	0.69
Lab	MLP	0.78	0.48	0.57	0.72	0.63	0.59	0.60	0.67	0.63	0.73	0.67	0.68	0.61	0.69
Lab with feature selection	MLP	0.78	0.48	0.57	0.72	0.63	0.59	0.60	0.67	0.63	0.73	0.67	0.68	0.61	0.69

Figure 9 shows the visual explanations related to the Grade class. Grade explanations are quite diversified. Class Grade A consists of highly engaged students in terms not only of attendance (e.g., of laboratory 3) but also of notified errors (e.g., in laboratory 4). In fact, learning by doing (i.e., trial and error) is known to be beneficial for beginner programmers.

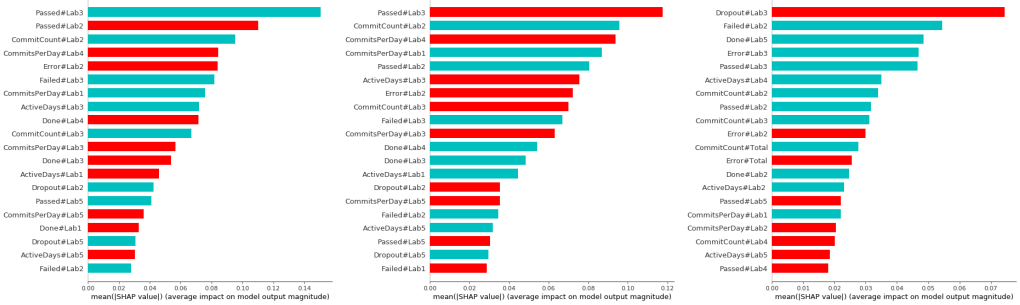
Class Grade B encompass students who either completed at least the first three laboratory or completed all the activities of the last laboratories but with a limited number of commits. Finally, class Grade C includes students who neither completed the first nor the last laboratories. Importantly, submissions to lab 5 only are not sufficient to achieve a better grade.



(**a**) Grade A (**b**) Grade B (**c**) Grade C

Figure 9. Explanations for class Grade. GNB classifier.

Figure 10 shows the visual explanations for the Grade Exam 1 class. Class Grade A Exam 1 is composed of students who have been active for most of the laboratories, especially those who have completed labs 1, 3, and 4 (Done#Lab1, Done#Lab3, Done#Lab4 have a positive impact); this class includes also the students who have been active for a significant number of days in labs 1 and 5 (ActiveDays#Lab1, ActiveDays#Lab5). Although Done#Lab2 is not present among the most significant features, students in this class have completed a significant activity in this lab (number of errors before the correct solution —Error#Lab1— has a positive impact, and number of failed tests —Passed#Lab2— also has a positive impact). Hence, we can conclude that having completed lab 2 is not a sufficient condition to obtain a high grade, but putting a strong effort into it is likely paying off.



(**a**) Grade A Exam 1 (**b**) Grade B Exam 1 (**c**) Grade C Exam 1

Figure 10. Explanations for class Grade in Exam 1. MLP classifier.

The main behavioral difference between the students who belong to class Grade A Exam 1 and those who belong to class Grade B Exam 1 is that the latter puts more effort into lab 3 activities (Passed#Lab3, tActiveDays#Lab3, CommitCount#Lab3, CommitsPerDay#Lab3 has a positive impact), put less effort in lab 2 and did not do lab 4 (Done#Lab4 has a negative impact). Conversely, for class Grade A, the effort seems more constant during all labs.

Finally, the most significant feature associated with the class Grade C Exam 1 is the dropout from the third laboratory onward (Dropout-#Lab3). Coherently, the features

Done#Lab5, Error#Lab3 and Passed#Lab3 indicate that the activities carried out in the third and fifth lab have a negative impact. Students who were not or less active from the second half of the course onward were therefore classified as Grade C Exam 1.

5.5. Correlation between Exam Success and Previous Exam Attempts

We also study the predictability of the exam performance from the second attempt on. Exam success has shown to be not correlated with the number of previous attempts. Conversely, the combination of previous exam attempts and laboratory attendance is discriminating, confirming the significance of the previously results.

6. Conclusions and Future Works

In this work, we explored the time-dependent interactions between the students of an Object-Oriented Programming course and the Version Control Systems used to accomplish lab activities. It applied Machine Learning models on a dataset collecting various behavioral and cognitive descriptors of the students' activities in order to forecast students' performance.

According to the results, we recommend the following actions:

- To early prevent exam failures, teachers are recommended to alert the students who are not active in the first two/three laboratories.
- To prevent student dropout, teachers can encourage student attendance to the very first laboratories in order to learn the necessary fundamentals.
- Students who achieved medium or low grades are likely to have attended only part of the scheduled laboratories. Hence, teachers should push students to attend all the laboratories with a view to the upcoming exam.
- To obtain the highest grade (A), it is necessary to persist in the activities on the VCS for the entire course duration to obtain a high grade. Hence, teachers should encourage students who pursue higher grades to complete all the assignments.
- Including in the ML model the knowledge about the previous exam attempts can be beneficial, but only in combination with other VCS-based features.

As future work, we plan to extend our study firstly to other courses in the area of computer science and then to other disciplines. Furthermore, we aim to tightly integrate the AI technologies described in this paper in the official platforms used in our university in order to simplify the technology and allow teachers to perform deeper analysis.

Author Contributions: Conceptualization, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), L.F., and M.T.; methodology, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), L.F., and M.T.; software, L.C. (Lorenzo Canale), and L.C. (Luca Cagliero); validation, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; formal analysis, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; investigation, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F; resources, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; data curation, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; writing—original draft preparation, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; writing—review and editing, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; visualization, L.C. (Lorenzo Canale), L.C. (Luca Cagliero), and L.F.; supervision, L.C. (Luca Cagliero), L.F., and M.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The research uses publicly available data released by third parties.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zagalsky, A.; Feliciano, J.; Storey, M.A.; Zhao, Y.; Wang, W. The Emergence of GitHub as a Collaborative Platform for Education. In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15), Vancouver, BC, Canada, 14–18 March 2015; pp. 1906–1917. [CrossRef]
2. Zakiah, A.; Fauzan, M. Collaborative Learning Model of Software Engineering using Github for informatics student. In Proceedings of the 2016 4th International Conference on Cyber and IT Service Management, Bandung, Indonesia, 26–27 April 2016; pp. 1–5. [CrossRef]
3. Feliciano, J.; Storey, M.; Zagalsky, A. Student Experiences Using GitHub in Software Engineering Courses: A Case Study. In Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), Austin, TX, USA, 14–22 May 2016; pp. 422–431.
4. Hsing, C.; Gennarelli, V. Using GitHub in the Classroom Predicts Student Learning Outcomes and Classroom Experiences: Findings from a Survey of Students and Teachers. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), Minneapolis, MN, USA, 27 February–2 March 2019; pp. 672–678. [CrossRef]
5. Bond, M.; Buntins, K.; Bedenlier, S.; Zawacki-Richter, O.; Kerres, M. Mapping research in student engagement and educational technology in higher education: A systematic evidence map. *Int. J. Educ. Technol. High. Educ.* **2020**, *17*, 2. [CrossRef]
6. Conijn, R.; Snijders, C.; Kleingeld, A.; Matzat, U. Predicting Student Performance from LMS Data: A Comparison of 17 Blended Courses Using Moodle LMS. *IEEE Trans. Learn. Technol.* **2017**, *10*, 17–29. [CrossRef]
7. Isacson, J.; Lindblom, E. Correlation of User Behaviour Patterns and Assignment Supplements in KTH-GitHub Repositories. Bachelor's Thesis, KTH, Stockholm, Sweden, 2017.
8. Guerrero-Higueras, A.; Llamas, C.; Sánchez, L.; Fernández, A.; Costales, G.; Conde-González, M. Academic Success Assessment through Version Control Systems. *Appl. Sci.* **2020**, *10*, 1492. [CrossRef]
9. Lundberg, S.M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J.M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.I. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2020**, *2*, 2522–5839. [CrossRef]
10. Griffin, T.; Seals, S. GitHub in the classroom: Not just for group projects. *J. Comput. Sci. Coll.* **2013**, *28*, 74.
11. Baumstark, L.; Orsega, M. Quantifying Introductory CS Students' Iterative Software Process by Mining Version Control System Repositories. *J. Comput. Sci. Coll.* **2016**, *31*, 97–104.
12. Tushev, M.; Williams, G.; Mahmoud, A. Using GitHub in large software engineering classes. An exploratory case study. *Comput. Sci. Educ.* **2020**, *30*, 155–186. [CrossRef]
13. Sprint, G.; Conci, J. Mining GitHub Classroom Commit Behavior in Elective and Introductory Computer Science Courses. *J. Comput. Sci. Coll.* **2019**, *35*, 76–84.
14. Guerrero-Higueras, A.M.; DeCastro-García, N.; Matellán, V.; Conde, M.A. 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 (TEEM'18), Salamanca, Spain, 24–26 October 2018; pp. 306–312. [CrossRef]
15. Guerrero-Higueras, Á.M.; DeCastroGarcía, N.; RodriguezLera, F.J.; Matellán, V.; Ángel Conde, M. Predicting academic success through students' interaction with Version Control Systems. *Open Comput. Sci.* **2019**, *9*, 243–251. [CrossRef]
16. Mierle, K.; Laven, K.; Roweis, S.; Wilson, G. Mining Student CVS Repositories for Performance Indicators. *SIGSOFT Softw. Eng. Notes* **2005**, *30*, 1–5. [CrossRef]
17. IEEE-CS. *Software Engineering Competency Model*; Technical Report; IEEE Computer Society Press: Washington, DC, USA, 2014.
18. Myers, K.K.; Sadaghiani, K. Millennials in the Workplace: A Communication Perspective on Millennials' Organizational Relationships and Performance. *J. Bus. Psychol.* **2010**, *25*, 225–238. [CrossRef]
19. Torchiano, M.; Bruno, G. Integrating software engineering key practices into an OOP massive in-classroom course: An experience report. In Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials (SEEM '18), Gothenburg, Sweden, 2 June 2018; pp. 64–71. [CrossRef]
20. Zaki, M.J.; Meira, W., Jr. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2020. [CrossRef]
21. Xavier, M.; Meneses, J. A Literature Review on the Definitions of Dropout in Online Higher Education. In Proceedings of the EDEN 2020 Annual Conference, Virtual, 22–24 June 2020. [CrossRef]
22. Canale, L. *Artificial Intelligence Methodologies to Early Predict Student Outcome and Enrich Learning Material*; Politecnico di Torino: Torino, Italy, 2022.
23. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
24. Brodersen, K.H.; Ong, C.S.; Stephan, K.E.; Buhmann, J.M. The Balanced Accuracy and Its Posterior Distribution. In Proceedings of the 20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23–26 August 2010; IEEE Computer Society: Washington, DC, USA, 2010; pp. 3121–3124. [CrossRef]