International Journal of Engineering Education Vol. 39, No. 2, pp. 476–487, 2023
Printed in Great Britain

0949-149X/91 $3.00+0.00

Applying Project-Based Learning to Teach Software Analytics and Best Practices in Data Science*

SILVERIO MARTÍNEZ-FERNÁNDEZ[1]**, CRISTINA GÓMEZ[1] and VALENTINA LENARDUZZI[2]
[1] Universitat Politècnica de Catalunya, C/ Jordi Girona 1-3, 08034 Barcelona, Spain. E-mail: silverio.martinez@upc.edu, cristina.gomez@upc.edu
[2] University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland. E-mail: valentina.Lenarduzzi@oulu.fi

Due to recent industry needs, synergies between data science and software engineering are starting to be present in data science and engineering academic programs. Two synergies are: applying data science to manage the quality of the software (software analytics) and applying software engineering best practices in data science projects to ensure quality attributes such as maintainability and reproducibility. The lack of these synergies on academic programs have been argued to be an educational problem. Hence, it becomes necessary to explore how to teach software analytics and software engineering best practices in data science programs. In this context, we provide hands-on for conducting laboratories applying project-based learning in order to teach software analytics and software engineering best practices to data science students. We aim at improving the software engineering skills of data science students in order to produce software of higher quality by software analytics. We focus in two skills: following a process and software engineering best practices. We apply project-based learning as main teaching methodology to reach the intended outcomes. This teaching experience shows the introduction of project-based learning in a laboratory, where students applied data science and best software engineering practices to analyze and detect improvements in software quality. We carried out a case study in two academic semesters with 63 data science bachelor students. The students found the synergies of the project positive for their learning. In the project, they highlighted both utility of using a CRISP-DM data mining process and best software engineering practices like a software project structure convention applied to a data science project.

Keywords: project-based learning; software analytics; software quality; data science; software engineering

1. Introduction

Due to the increasing adoption of Big Data technologies, data science has gained attention in multiple knowledge areas, including software engineering. Software companies have a need to understand the learning skills required to organize multi-disciplinary teams with expert software and data professionals. For example, Microsoft and IBM have reported the need to organize multi-disciplinary teams composed of data scientists and software engineers to carry out software analytics projects (about data-driven software development and quality improvement) [1, 2]. This has led to a new career path: data scientists in software teams [1]. Indeed, software analytics (performed by data scientists in software teams) guides practitioners in decision making throughout the software development process [3].

There has been recent research on how to teach software analytics. For example, Hassan [4] describes an experience report on teaching a computing graduate course in mining software repositories. Zhang et al. [5] have given tutorials to researchers and practitioners on software analytics. However, up to our knowledge, teaching software analytics in data science programs has not been explored yet. As argued by Zhang et al. [5]: ''it remains an open challenge for university educators or industry trainers on how to effectively teach and train students and practitioners to equip them with skills and knowledge required for conducting software analytics''. Kästner et al. [6] also argue that not teaching the synergies of data science and software engineering in academic programs is an education problem. Therefore, we identify the need of learning activities to teach software analytics to data science students to guide them on how to perform better on software analytics projects highly demanded by industry.

The main goal of this paper is to present the teaching experience of designing, executing and evaluating a new activity using the project-based learning (PBL) methodology [7] in the laboratory of the Advanced Topics in Data Engineering 2 (TAED2) course of the Bachelor's Degree in Data Science and Engineering (GCED) at the Universitat Politècnica de Catalunya (UPC). The objective of this activity is to carry out a software analytics project (hereafter, the project) to detect improvements in the quality of open source software. The project has two innovative software engineering aspects for the students: (i) the application of data science in a new domain (software quality problem

** Corresponding author: silverio.martinez@upc.edu

* Accepted 23 December 2022.

detection); and (ii) the conduction of best software engineering practices.

The design, execution, and evaluation of this activity has four main contributions:

(i) To evaluate the usefulness of a data mining process (CRISP-DM) perceived by data science students in the context of their software analytics projects.
(ii) To evaluate the usefulness of best software engineering practices perceived by data science students in the context of their software analytics projects.
(iii) To analyze which laboratory activities help data science students to learn about how to develop a software analytics project.
(iv) To study the feasibility of applying PBL in software analytics projects executed by data science students to improve the software quality.

In this paper, we present aggregated results from two semesters (preliminary results for the pilot semester are presented in [8]). Section 2 introduces the background on PBL and software analytics. Section 3 shows the related work on teaching software analytics, pointing out that the application of PBL in the domain is novel. Section 4 presents the project-based course for teaching software analytics. Section 5 reports the research methodology to evaluate the project-based course for teaching software analytics. Section 6 shows the evaluation results and reflects on strengths and considerations for future improvements. Section 7 presents the limitations of our work. Finally, Section 8 concludes this paper.

2. Background

In the following, we provide backgrounds on PBL and software analytics.

2.1 Project-Based Learning (PBL) and Applications

PBL is a methodology that allows students to acquire new knowledge through projects related to real-world challenges and problems, which has been extensively adopted in engineering education and its benefits have been reported in the literature [9]. For example, Zhao et al. [10] introduced a PBL methodology to redesign an IoT course of the fourth-year for electronics and information engineering at Central China Normal University. The new methodology enabled 220 students to make progress gradually from understanding IoT knowledge. Results before and after the IoT course redesign showed that students' final grades and their self-efficacy related to the topic were significantly bettered. In the same line, Liu et al. [9] analyzed the learning experiences with PBL activities of 21 engineering students from a first-year Introduction to Engineering course from a Chinese university. The analysis suggested that, although the students perceived difficult to choose and develop a project at the beginning, most students' self-efficacy was improved in the process of finishing the project.

PBL has also demonstrated its effectiveness even with complex concepts and topics. For instance, Foss et al. [11] presented a PBL case study that allowed the students (30 students divided into 5 groups) to understand the cross disciplinary nature of product quality as an intersection of materials science, manufacturing engineering, and engineering design areas. Moreover, the PBL methodology demonstrated to be effective to develop different students' skills, as teamwork, oral and written communication, and problem solving. Similarly, Del Rey Castillo et al. [12] explored whether a PBL activity would improve learning of complex concepts related to concrete manufacturing. They used four assets to make the evaluation: observations made by lecturers, student evaluations, a targeted survey, and course grades. The results showed that students were satisfied overall with the PBL activity and they were more satisfied with face-to-face activity than with the on-line counterpart. The results also pointed out that student learning about the topic improved when complex theory was integrated with practical learning activities using a project-based approach.

This work shows also empirical evidence of the benefits of applying PBL in a course of a bachelor degree.

2.2 Software Analytics

Software analytics consists of using data-driven approaches to obtain revealing and actionable information to help software professionals in their data-related tasks [13]. It aims to improve the productivity of the development process, the quality of software systems, and/or the end-user experience [14]. In this laboratory, the project focuses on detecting improvements in the quality of software systems from the analysis of their data. To do so, we start by executing mining tasks on raw data (e.g., source code, bug repositories) to discover patterns (e.g., build a bug prediction model), and finally produce actionable information (e.g., schedule specific quality assurance tasks, allocate resources).

In the context of teaching software analytics, we show the most important artifacts used in this paper together with a cogently articulated rationale for why they were adopted in this subject (CRISP-DM as data mining process, best software engineering practices, and the dataset used).

2.2.1 CRISP-DM

CRISP-DM is the acronym for "Cross-industry standard process for data mining". According to two surveys conducted in 2014 and 2007, it is the most widely used data mining process in the industry [15]. In fact, at present, adaptations of it are applied not only in data mining, but in the context of machine learning [16]. Other data mining processes are KDD [17] and SEMMA (see Kurgan et al. for comparison [18]). CRISP-DM was the result of an European research project. The goal was to create a standard process for the data mining community. It is based on the practical experience of how practitioners conduct data mining projects. Among the 300 companies that collaborated in a series of workshops, three composed the core of the consortium: Daimler Chrysler (later Daimler- Benz), SPSS (later ISL), and NCR. We emphasize that the standard is non-proprietary, freely available, and agnostic with respect to industry, tools, or application domains. One of the main contributions of CRISP-DM is its reference model. This reference model contains six phases (see Fig 1): business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The standard also includes the methodology, user guide, and reporting template for each of the project phases. All information can be found in the user guide [19].

Recently, the use of this process in software development teams has been shown to be effective for data-driven software quality improvement in enterprises [20], even with some adaptations [21, 22]. For this reason, we study its feasibility and usefulness in an academic context in this subject and teaching experience.

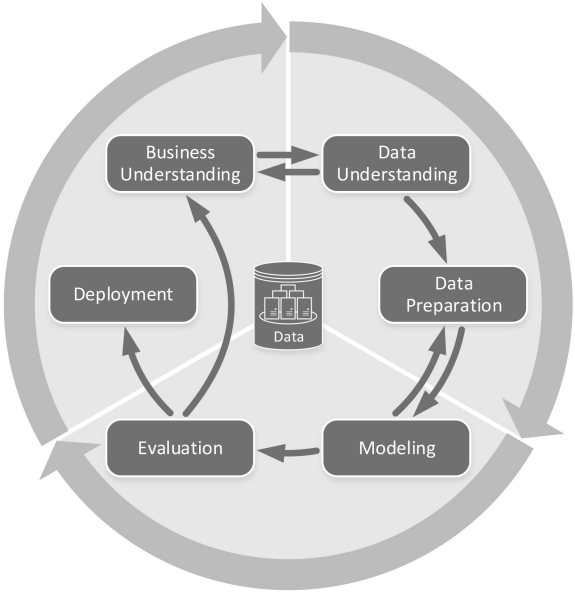

Fig. 1. CRISP-DM phases.

2.2.2 Software Engineering Best Practices: Data Science Project Structure

Software engineering best practices are key in data science projects [23]. Recent work present linters (like *mllint* and *pynblint*) for data scientists so that they can identify missing best practices in Jupyter Notebooks [24, 25]. As the linters work as a checklist, students shall use one linter and include at least three best practices of their choice.

An example of best practice is using a project structure. Data science projects contain scripts, interactive workbooks, configuration files, data files, reports, documentation, tests and figures. A common problem with these data science projects is their reproducibility. If any data scientist or developer needs to run a script, interactive notebook, or artificial intelligence model, they need to know where the correct version of the models, training data, etc. is located. This problem is analogous to that encountered by software development teams when they need to find some artifact within the structure of their software project. For this reason, there has recently been a push for multiple data science project structure conventions, such as the one offered for Tensorflow [26] and Azure [27]. At the time of the start of the subject, the top-rated data science project structure on GitHub was Cookiecutter Data Science [28], which motivated its choice. Cookiecutter Data Science project structure: (i) provides a generic repository structure for models, data, configuration, etc.; (ii) allows starting a project customized by author, project name, license, Python version, etc.; (iii) specifies a naming convention for notebooks, provides a directory for trained models, and a folder structure to separate raw data from external/processed data.

2.2.3 The TechDebt Dataset

There are multiple datasets to perform software analytics for various software engineering problems (e.g., cost estimation, requirements elicitation, etc.) [29]. They can be found in proprietary repositories such as PROMISE [30] or even in Kaggle. In this teaching experience we focus on a specific software analytics problem: technical debt. Technical debt reflects the implicit cost of the additional rework caused by implementing a rapid solution at a particular point of development instead of using a higher quality, but costlier alternative that would take longer. To study this problem, we use the "TechDebt" dataset [31]. At the time of the start of the course, the TechDebt dataset consisted of 33 open-source Java projects with different data calculated and/or extracted by different tools. For each project we can find: metrics of static code analysis performed by the SonarQube tool, Jira backlogs

tasks, commits, refactoring data extracted by the RefactorMiner tool, and bugs identified by the SZZ algorithm (for more information, see [31]). The TechDebt dataset, with its documentation and previous data cleaning work, makes it feasible to complete the project in seven weeks.

3. Related Work on Teaching Software Analytics

Due to the relevance of software analytics in industry, in the last decade, there has been a proliferation of related courses, some of them associated with mining software repositories (MSR) conference. A subset is depicted in Table 1.

There are other two related areas in the synergies of data science and software engineering, which are worth mentioning. First, the application of software analytics on students' projects to monitor their evolution and quality [33]. However, in our subject we do software analytics on top open-source projects from the TechDebt dataset, rather than on students' projects. Second, with the latest wave of artificial intelligence and deep learning, we can also see the need to teach software engineering techniques to create systems based on artificial intelligence [34]. Three relevant teaching experiences have been carried out in the last year at Carnegie Mellon University [6], Fontys University Eindhoven [35], and the University of Bari [36].

To end up, this research has been motivated because, as we can see in Table 1, there is no course on software analytics in data science programs taught with PBL. Hence, the teaching experience of this paper is innovative in applying PBL for teaching software analytics to data science students, as well as teaching best software engineering practices (e.g., project structure convention) to this student profile at UPC.

4. The Project-Based Course for Software Analytics and Best Practices

In this section we explain the specific teaching objectives of the subject, and we focus on the object of study of this teaching experience: the software analytics project to be carried out by the students.

4.1 Teaching Objectives

TAED2 is a compulsory course and it is part of the seventh quarter of the GCED at UPC. The laboratory of the TAED2 course focuses in two teaching objectives, in which the subject is the student, a verb related to a level of Bloom's taxonomy [37, 38] is used, and the content and circumstance are specified. These teaching objectives are:

(i) To interpret and apply the basic concepts of Software Engineering, especially in relation to the use and exploitation of data.
(ii) To apply and analyze concepts and methods related to the use of data from the development process in the quality management of the software system.

They are covered in depth by the project proposed in this teaching experience, reaching levels 3 (application) and 4 (analysis) of Bloom's taxonomy.

4.2 The Software Analytics Project

The project is the main activity of the TAED2 laboratory. The PBL methodology has been chosen due to the need for multidisciplinary teams in software analytics, the complexity of the objective to be achieved, and to encourage the active and critical participation of students in identifying opportunities for software quality improvement. PBL is an active learning methodology in which students work on a real-world problem over a

Table 1. Related courses about software analytics

Course title	University/Organization	Teaching methodology	Main assets	Language	Ref.
Data Analysis in Software Engineering using R	University of Alcala, University of the Basque Country	Tutorial with notes	Data mining, data sources, models	R	[29]
Machine Learning for Software Engineering	University of Delft	Seminar	Replication of papers or proposing a new approach	Python	[32]
Teaching and Training for Software Analytics	Microsoft Research Asia, North Carolina State University	Tutorial with first-hand experiences	Industry experiences	n/a	[5]
Mining Software Repositories	Queen's University	Lecturing (first two weeks), then seminar-style	Data mining techniques	R	[4]
This work	Universitat Politècnica de Catalunya	Project-based learning	CRISP-DM, software engineering best practices, TechDebt dataset	Python	This work

period of time. Furthermore, PBL has been proved to be one of the most engaging elements for students [39]. The project teams in the TAED2 laboratory are composed of 3–4 students. The project is organized in an iterative and incremental way, following the CRISP-DM phases, with flexibility to jump to previous phases if necessary, as recommended by the process itself. During the project execution, the lecturer gives weekly feedback on the project and solves doubts.

Since the goal of the lab sessions is not to teach new programming languages or environments but to apply those learned in the first three years of the GCED, the lab sessions were designed to use Python and Jupyter Notebook. Still, teams were free to choose other languages and tools if they justify their decision.

The project is developed in seven weeks, as shown in Table 2. In the first session, the project teams are formed. The dataset is also introduced and its characteristics are explained. From the second week on, the teams present their progress in the project and receive weekly feedback. They start from CRISP-DM phase 1, specifying a business objective as part of the business understanding, to the deployment and final presentation.

In addition to PBL, the integration of other complementary methodologies is noteworthy. An example of a flipped classroom can be found in [40]. In the first week, each student in a team is assigned to read one paper. The available papers were the following: [1, 31, 41, 42] in the 2020 Fall and [1, 31, 43, 44] in the 2021 Fall. These papers help the students to choose a software analytics target and to understand the dataset. Between the first two sessions, students individually read the assigned paper. In the second week, they must agree on a summary with components of other teams that have the same assigned paper and present the summary of each paper to their team members. Furthermore, each week the students have to study (outside the classroom) the activities for a specific CRISP-DM phase in order to apply them as the project progresses.

At the midpoint of the laboratory sessions, an ''Ask-me-anything'' session is held via videoconference. That session, in the 2020 Fall and 2021 Fall, involved three data science panelists who are part of software development teams. The panelists were of different gender, place of work (company, research institute, and university), and country. After a brief introduction for one hour, the students could ask any questions, such as advice about the project or about their future career. The 2021 session is available on [45].

Another activity is the digital book, accessible by all the enrolled students, which is created with the project reports of every team. The book, which is a Google Docs document, allows each team to see the summary of other teams' projects. Thus, each team can see what the other teams are doing before the final presentation. This book is inspired by other books created in other subjects (e.g., software architecture in the University of Delft) that encourage collaborative learning [46]. In the 2020 Fall was created one week before the last (week 6), and in the 2021 Fall it was created in the first week.

The evaluation of the delivered projects consists of the final report and the developed software. Project teams are provided with a rubric to learn how both the final report and the software are evaluated. The report must follow the CRISP-DM guidelines and the software code has to follow a subset of software engineering best practices.

5. Research Methodology

We designed our empirical study based on the guidelines defined by Wohlin [47]. In this section, we describe the empirical study, including the goal and the research questions, the study instruments, the study execution, and the data analysis.

5.1 Research Goal and Research Questions

Our evaluation goal is defined as: *Analyze* a laboratory following project-based learning *for the purpose of* teaching software analytics and software engineering best practices *with respect to* the useful-

Table 2. Project schedule

Week	Tasks/results of each lab session
1	Project start: Teams created and Google drive folders configured. Introduction to the dataset. Preliminary definition of the software analytics objective.
2	Discussions based on assigned readings. Presentation of initial selection of the project objective.
3	Presentation of business understanding and the initial steps of understanding the data.
4	Presentation of data understanding (and deadline to refine the objective if necessary). Start of data preparation. Ask me anything session with experts.
5	Presentation of data preparation and modeling steps.
6	Presentation of modeling steps (if there are any updates) and evaluation.
7	Final presentation, including reflection of the deployment step. Delivery of: (a) final report (following CRISP-DM) and (b) software (following project structure). Project closure.

ness and feasibility *from the point of view of* students and lecturers *in the context of* a subject of a bachelor degree on data science and engineering.

We defined the following Research Questions (RQ):

RQ$_1$ Do data science students find the application of *CRISP-DM* useful to achieve the objective of a software analytics project?
We aim at exploring a data mining methodology (i.e., CRISP-DM) that is feasible and useful in a teaching context to guide a software analytics project.

RQ$_2$ Do data science students find *software engineering best practices* useful to promote correctness, reproducibility and quality of software analytics projects?
We aim at facilitating the quality aspects (e.g., reproducibility) of the code developed in software analytics projects by means of best practices (e.g., project structure).

RQ$_3$ Do *collaborative learning activities* help data science students to learn in a software analytics project?
We aim at exploring the interaction among students from diverse teams to learn from each other by means of a digital book and sessions with external experts.

RQ$_4$ Do *laboratory team work* help data science students to learn in a software analytics project?
We aim at understanding the potential benefits and drawbacks of PBL (e.g., real problem challenge, conducting team work) to learn software analytics.

5.2 Study Instruments

To execute and perform the evaluation of our study, two instruments were designed: (i) An initial questionnaire to capture the knowledge of the students before their projects kick-off and (ii) a final questionnaire for collecting the students' satisfaction about the activities performed in the lab sessions, and the students' opinions about the artifacts used in their projects.

In order to allow the replication of this study, we made these instruments available in our online appendix [48].

5.3 Study Execution

We performed our study in two semesters of TAED2 (2020 Fall and 2021 Fall). In total, 63 students were enrolled in the subject, 33 in the first semester and 30 in the second one. They created 17 teams of 3–4 students each to develop the software analytics project in the laboratory sessions.

In the following we explain two aspects related to the study execution: the specific objectives the students selected to develop their projects, and how we analyzed the responses of the study instruments.

Software analytics objectives selected in the projects: Within the general objective of improving software quality from exploiting the data generated in software development processes, each team had to define its specific objective focused on one particular aspect. To do so, they had been provided with a list of ideas, and it was possible to work on a new proposed idea.

In the 2020 Fall, five teams focused on predicting and classifying commits with respect to bugs or code smells. For example, one team integrated their prediction model into a system in which, based on commit characteristics (such as description, number of lines added/modified/removed, percentage of comments, code complexity, etc.), they predicted the probability of the presence of bugs in the code. One team classified different types of commits, without elaborating prediction algorithms. Two teams focused on classifying developers by creating profiles based on the characteristics of their code (e.g., use of comments, complexity of functions, performing refactoring tasks, work on resolving bugs, etc.). One of these two teams published its work in [49]. This type of developer profile is used in software companies for staff recruitment. Another team classified projects by their maintainability and sustainability over time.

In the 2021 Fall, the final presentations were recorded (available footnotes below). One team focused on studying commits for predicting the text of the description of prospective commits [50]. This could help developers to write better commits descriptions, and not forgetting key information. Two teams classified developer types, and created scripts for project managers to balance teams according to developer experience [51, 52]. The rest of the projects conducted data exploration tasks (e.g., refactoring and faults correlation) [53].

Data Analysis: To analyze each answer of the final questionnaire, we computed the mode (probably the most suitable for easy interpretation [54]) and the mean of students' responses of each semester and for both aggregated semesters in order to identify the common evaluation rate. Moreover, we calculated for each answer, in the aggregated case, the minimum, the maximum and the median rate.

6. Evaluation Results

Bearing in mind our research goal, to evaluate the new activity introduced in the laboratory sessions (i.e., the project), we collected and analyzed the data

provided by the students from the two instruments designed (initial and final questionnaire). We aggregated the data of the two semesters.

This section details the results of the evaluation and presents the answers to RQ1, RQ2, RQ3 and RQ4. Moreover, the feedback and opinion of the faculty staff and students after carrying out their projects, as well as executed improvements between the semesters are exposed.

6.1 Initial Questionnaire

To prepare the laboratory sessions, the initial questionnaire was conducted two weeks prior to the start of those sessions in both semesters. Participation in the questionnaire was optional and reached 92% (58 students out of 63).

The questionnaire had questions about the students' background in programming languages and development environments, their experience in data mining processes and data science projects, and their knowledge about software engineering. To measure the experience, we used an ordinal scale with the following values for experience: none (never used), basic (used in one project or subject), intermediate (used in 2 to 5 subjects or projects), advanced (used in more than 5 subjects or projects), and n/a (not applicable).

Regarding programming languages, 55 students out of 58 (95%) had advanced knowledge of Python. C++ and R were also known, with 53 students having intermediate to advanced knowledge. Regarding Java, only 20 students (34%) had some knowledge. Although not explicitly asked about them, in the open section at least 16 students mentioned knowledge about Matlab and SQL.

The development environment most used by the students is Jupyter Notebook, with advanced knowledge by 52 of the students, followed by R Studio, with 35 students reporting advanced knowledge.

Prior to the fourth year of the bachelor degree, 95% of the students had no knowledge of data mining processes, such as CRISP-DM, KDD, or SEMMA. Moreover, 69% of the students did not know what Software Engineering is.

Finally, 29 students had previous experience in conducting data science projects (50%), although none in the context of software-related data. Examples of data science projects conducted prior to TAED2 by students include: predicting whether a user will click on web advertisements, creating movie subtitles, developing image recognition, developing models to detect diseases and text classification algorithms.

6.2 Final Questionnaire

The final questionnaire was conducted after the students finished their projects in both semesters. Participation in the voluntary questionnaire was 76% (48 students out of 63). This questionnaire aimed to explore the degree of the students' satisfaction with the activities and tasks performed in the TAED2 laboratory. To measure the answers, we used an ordinal scale from 1 to 5 (the same as in the initial questionnaire). Table 3 shows the questions, the results for each semester and the aggregated results for both semesters.

Regarding **RQ1 (Do data science students find the application of *CRISP-DM* useful to achieve the objective of a software analytics project?)**, the first

Table 3. Results of final questionnaires

	2020 Fall		2021 Fall		Aggregated 2020 and 2021 years					
Question	**N**	**Mode**	**N**	**Mode**	**N**	**Min**	**Max**	**Mean**	**Median**	**Mode**
I find the application of CRISP-DM useful to achieve the objectives of our project.	29	4	19	4	48	1	5	4.17	4	4
I believe that using the Cookiecutter Data Science project structure is useful to promote correctness and reproducibility.	28	5	19	4	47	2	5	4.09	4	4
I find using software engineering best practices useful to foster the quality of our project.	n/a	n/a	19	4	n/a	n/a	n/a	n/a	n/a	n/a
The "ask me anything" session helped me to learn	27	4	16	4	43	2	5	4.07	4	4
The common digital book in the lab helped me to learn	29	3	17	4	46	1	5	3.46	4	4
The lab project helped me learn	29	4	19	4	48	1	5	3.83	4	4
Teamwork in the laboratory helped me to learn	29	5	19	4	48	2	5	4.19	4	4
For my team in the lab, it was feasible to do a data science project in a software context.	29	5	19	5	48	2	5	4.38	4.5	5

question asked about the usefulness of applying a data mining process (CRISP-DM) in the laboratory classes. The students found the application of CRISP-DM useful to achieve the objectives defined in their projects (41 out of 63 students agree or strongly agree with the sentence "*I find applying CRISP-DM useful to achieve the goals of our project*", with a Mode = 4 and a Mean = 4.17). This question was followed by another question with open space for leaving comments. In this open-ended question, 25 out of 48 students said that they would use CRISP-DM again in other projects. Some students emphasized the way of working promoted by CRISP-DM "*I will definitely take it into account because is a very sensible and well-structured manner of working*", "*I think it's a great work structure and I will probably use it in the future*" and "*CRISP-DM has helped us a lot when it comes to organizing work and doing the tasks in the corresponding order*". However, some students explicitly expressed some doubts regarding their future use. Some examples are: "*I would only use it if it was a very big project between a lot of people*" and "*I find it a bit rigid and repetitive*".

About **RQ2 (Do *data science students find software engineering best practices* (e.g., Cookiecutter template) useful to promote correctness, reproducibility and quality of software analytics projects?)**, the second question asked about the usefulness of applying a project structure. As Table 3 shows, students found useful to use the "Cookiecutter Data Science" project structure to promote correctness and reproducibility (Mode = 4 and Mean = 4.09). This project structure only received 2 negative comments: "*I find the Cookiecutter structure too rigid, many directories are not always useful*" and "*I think it can be a structure too strict that may not adapt well to all kind (or sizes) of projects*". Among the positive comments we can find, for example: "*It is rather interesting to see the balance between freedom of organization, exploration and 'fresh-eyes' work, and the orderliness achieved with project structures like 'Cookiecutter Data Science'*", "*I found Cookiecutter pretty useful, as following a well-defined structure ensures to keep all things ordered and thus, speed up performing modifications as well as finding code*". However, from the teaching staff, we have to indicate that during the project several doubts regarding some optional sections of the template were raised.

In the 2021 Fall, we added a new question in the final questionnaire to know if the use of software engineering best practices in the students' projects promoted the quality of their projects. Only 2 out of 19 students disagree with this promotion whereas 16 students (84%) agree or strongly agree. Only one student manifested her neutrality.

Regarding **RQ3 (Do *collaborative learning activities* help data science students to learn in a software analytics project?)**, the "Ask-me-anything" session was well received (Mode = 4 and Mean = 4.07). In fact, students felt that it could have been even longer than one hour. Since it was a videoconference, the panelists did show their interest in seeing the students' faces during the session, and not just students asking questions.

The worst received activity was the digital book (although the Mode was 4, only 24 out of 48 students considered that the common digital book in the laboratory helped them to learn). Students did not give explicit feedback on the problems. A hypothesis to investigate in next semesters is whether the book would be better received if it was created from the beginning of the project.

Regarding **RQ4 (Do *laboratory team work* help data science students to learn in a software analytics project?)**, the last three questions of the questionnaire were about aspects of PBL. Students agree or strongly agree (34 students out of 48) that applying PBL was useful for them to learn (Mode = 4 and Mean = 3.83). Most students (83%) recognize the teamwork as a key factor to learn (Mode = 4 and Mean = 4.19). Moreover, 44 out 48 students declared that their data science project was feasible in a software context (Mode = 5 and Mean = 4.38).

These questions were followed by another open-ended question to leave comments on problems or challenges solved during the project. Thirteen students explicitly mentioned that they struggled with the data comprehension part. Some examples are: "*The quality of the data was surely a problem*" and "*The main problems were to understand the data and the exact meaning of the variables*". Fifteen students indicated that the main challenge was doing the project in seven weeks: e.g., "*The main problem was the time, we did not feel like had enough time to properly work in the project*" and "*The main problem was the lack of time to properly develop our analysis*".

Finally, as in the initial questionnaire, knowledge about software engineering and CRISP-DM was asked in the final questionnaire, after finalizing the project. For software engineering, 92% of the students said they knew what it was, compared to 31% in the initial questionnaire while for CRISP-DM 98% said they had experience, compared to 5% initially. In addition, in the last question on overall satisfaction with the laboratory, only 10% were dissatisfied with the laboratory activities.

6.3 Discussions and Improvements

In general, from the faculty staff we make a positive evaluation and we are satisfied with the students' academic performance (total average of the whole

course grade: 7.6; and average grade of the software analytics project: 8.8).

With the experience of the 2020 Fall, there was enough data to analyze student feedback and make the improvements below.

Regarding the evaluation rubric, in the 2021 Fall we made it simpler by explicitly showing which parts of the project structure are optional and including an evaluation point at the middle of the software analytics project with half of the report (up to the understanding of the data).

Regarding the content, in the 2020 Fall we saw that students have shown a lot of interest in the data science project structure convention. For this reason, we improved a learning objective of the subject and add more best software engineering practices for this type of projects [25, 44]. As one student declared in the open comments: ''*more education on best practices*''. Best software engineering practices facilitate the deployment of the resulting model obtained in the project.

Finally, to solve the main challenges mentioned (difficulty in understanding the data and limited time), we introduced in the 2021 Fall, in its first week, more detailed concepts of the dataset variables (in both semesters it was done briefly with documentation of the dataset), and a page limit on the report was set to reduce the time spent on documentation.

Regarding the comments from students in the 2021 Fall, there were fewer suggestions for improvement, which could show that the laboratory is more mature. Still, five students mentioned that they would like to have freedom in choosing the dataset, and highlight their interest in best practices such as APIs. We will consider these suggestions in future semesters. Remarkably, students appreciated the PBL methodology.

7. Limitations

As with any study of teaching innovation, the above results of this project have some limitations:

- The surveys were conducted in English, the language in which the subject is taught, which is not mostly the mother tongue of all students. Therefore, the way the questions were formulated could have been a limiting factor. However, the lecturers took care to word the questions concisely, and the students were able to answer most of them using an ordinal scale from 1 to 5. This led to the main advantage of the questions, which is using a similar method of data collection. This makes the questions easy to understand and answer, and students do not feel compelled to express their opinion, allowing them to remain neutral.
- The age and gender of the students were not considered in the study. However, these factors probably did not influence the results because they were similar among them. It is worth noting the large number of women among the student body (>33%) being a STEM degree.
- The final questionnaire was completed at the end of the last lab session, so students might have felt fatigued when completing them. As before, since the questions were worded concisely and answered using a scale of 1 to 5, it is not very likely that students felt fatigued. These questionnaires were anonymous and optional. Therefore, they had no impact on subject grades.
- At the beginning of the course, students had to choose their team mates to do the software analytics project. There was no student who could not find a group. The way the teams were formed could influence the final grades of their members (e.g., students who already know each other tend to find it easier to work together).
- The results come from a limited population of 63 students. However, this sample of students is convenient for explorative RQs in a teaching innovation research [55].

8. Conclusions

This article has shown the experience of teaching software analytics and best practices to data science students. To do so, we explored the application of a well-known teaching methodology, project-based learning (PBL), in two semesters with 63 students. We report the rationale of the selection of the main artifacts of the laboratory, as well as complementary activities.

The results show that both PBL and the chosen artifacts are useful for the intended outcome: teaching software analytics to detect opportunities to improve the software in open source repositories to data science students. The taught synergies of data science and software engineering were positively perceived by the students, namely: the application of CRISP-DM in a software analytics project (RQ1); and the adoption of best software engineering practices (RQ2). Moreover, the students perceived positive collaborative learning activities such as the ''Ask-me-anything'' session (RQ3). Finally, the students considered that PBL characteristics such as team work helped to achieve the learning objectives (RQ4).

As future work, the inclusion of more software engineering best practices in TAED2 will be studied. Concretely, we will introduce the *pynblint* linter from [25]. In addition, in the last phase of

deployment, it will be considered to evaluate the creation of a component for a software system based on the model created and evaluated. Examples of components already created by TAED2 teams have been: a web application where the features of a commit were included and predicted the probability of containing bugs, and a component to predict the next words in the description of a commit.

Acknowledgments – This paper was partly funded by a teaching innovation project of ICE@UPC-BarcelonaTech (entitled "Audiovisual and digital material for data engineering, a teaching innovation project with open science"), and the "Beatriz Galindo" Spanish Program BEA-GAL18/00064.

References

1. M. Kim, T. Zimmermann, R. DeLine and A. Begel, Data scientists in software teams: State of the art and challenges, *IEEE Transactions on Software Engineering*, **44**(11), pp. 1024–1038, 2017.
2. P. Santhanam, E. Farchi and V. Pankratius, Engineering reliable deep learning systems, *arXiv preprint arXiv:1910.12582*, 2019.
3. A. Bener, A. Tosun, B. Caglayan, E. Kocaguneli and G. Calikli, Lessons learned from software analytics in practice, in C. Bird, T. Menzies and T. Zimmermann (eds), *The art and science of analyzing software data*, Elsevier, pp. 453–489, 2015.
4. E. Hassan, Raising MSR researchers: an experience report on teaching a graduate seminar course in mining software repositories (MSR), in *Proceedings of the 13th International Conference on Mining Software Repositories*, Austin, 14–22 May, pp. 121–125, 2016.
5. D. Zhang, Y. Dang, S. Han and T. Xie, Teaching and training for software analytics, in *2012 IEEE 25th Conference on Software Engineering Education and Training*, Nanjing, 17–19 April, pp. 92–92, 2012.
6. C. Kästner and E. Kang, Teaching software engineering for AI-enabled systems, in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training*, Seoul, 27 June–19 July, pp. 45–48, 2020.
7. L. Dym, A. M. Agogino, O. Eris, D. D. Frey and L. J. Leifer, Engineering design thinking, teaching, and learning, *Journal of engineering education*, **94**(1), pp. 103–120, 2005.
8. S. Martínez-Fernández, C. Gómez and X. Franch, Aprendizaje basado en proyectos de analítica de software en estudios de ciencia e ingeniería de datos, *Jornadas de Enseñanza Universitaria de la Informática*, Valencia, 7–8 July, pp. 35–42, 2021.
9. R. Liu, J. Zhu, W. Li, T. Ju and L. Chen, Investigating Engineering Students' Experiences of Self-Regulated Learning in Project-Based Learning Activities, *International Journal of Engineering Education*, **38**(5), pp. 1422–1433, 2022.
10. L. Zhao, S. Qu and S. Sun, Applying Project-Based Learning and an Integrated Laboratory Platform to Teach Internet of Things, *International Journal of Engineering Education*, **38**(5), pp. 1291–1306, 2022.
11. M. Foss, Y. Liu and S. Yarahmadian, Project-Based Learning in a Virtual Setting: A Case Study on Materials and Manufacturing Process and Applied Statistics, *International Journal of Engineering Education*, **38**(5), pp. 1377–1388, 2022.
12. E. Del Rey Castillo and C. Donald, Making the Abstract Concrete: A Project-Based Laboratory Activity on Concrete Manufacturing for Civil Engineering Students, *International Journal of Engineering Education*, **38**(5), pp. 1443–1457, 2022.
13. H. Gall, T. Menzies, L. Williams and T. Zimmermann, Software development analytics (dagstuhl seminar 14261), in *Dagstuhl Reports*, **4**(6), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
14. D. Zhang, S. Han, Y. Dang, J.-G. Lou, H. Zhang, and T. Xie, Software analytics in practice, *IEEE software*, **30**(5), pp. 30–37, 2013.
15. CRISP-DM, still the top methodology for analytics, data mining, or data science projects, https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html, Accessed 29 November 2022.
16. S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters and K.-R. Müller, Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology, *Machine Learning and Knowledge Extraction*, **3**(2), pp. 392–413, 2021.
17. R. J. Brachman and T. Anand, The process of knowledge discovery in databases: A first sketch, in *KDD workshop*, **3**, Seattle, pp. 1–12, 1994.
18. L. A. Kurgan and P. Musilek, A survey of knowledge discovery and data mining process models, *The Knowledge Engineering Review*, **21**(1), pp. 1–24, 2006.
19. P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer and R. Wirth, The CRISP-DM user guide, in *4th CRISP-DM SIG Workshop*, Brussels, 18 March, 1999.
20. C. Ebert, J. Heidrich, S. Martínez-Fernández and A. Trendowicz, Data science: technologies for better software, *IEEE software*, **36**(6), pp. 66–72, 2019.
21. F. Martínez-Plumed, L. Contreras-Ochando, C. Ferri, J. Hernández- Orallo, M. Kull, N. Lachiche, M. J. Ramírez-Quintana and P. Flach, Crisp-dm twenty years later: From data mining processes to data science trajectories, *IEEE Transactions on Knowledge and Data Engineering*, **33**(8), pp. 3048–3061, 2019.
22. V. Plotnikova, M. Dumas and F. P. Milani, Applying the crisp-dm data mining process in the financial services industry: Elicitation of adaptation requirements, *Data & Knowledge Engineering*, **139**, p. 102013, 2022.
23. D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo and D. Dennison, Hidden technical debt in machine learning systems, *Advances in Neural Information Processing Systems*, **2**, December, pp. 2503–2511, 2015.
24. B. van Oort, L. Cruz, B. Loni and A. van Deursen, "project smells" – experiences in analysing the software quality of ml projects with mllint, *arXiv preprint arXiv:2201.08246*, 2022.
25. L. Quaranta, F. Calefato and F. Lanubile, Eliciting best practices for collaboration with computational notebooks, *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1), April, pp. 1–41, 2022.
26. Tensorflow-Project-Template, https://github.com/Mrgemy95/Tensorflow-Project-Template, Accessed 29 November 2022.
27. Azure-TDSP_ProjectTemplate, https://github.com/Azure/Azure-TDSP-ProjectTemplate, Accessed 29 November 2022.
28. Cookiecutter-data-science, https://github.com/drivendata/cookiecutter-data-science, Accessed 29 November 2022.
29. D. Rodriguez and J. Dolado, Data Analysis in Software Engineering using R, https://danrodgar.github.io/DASE/, Accessed July 2022.
30. Promise Software Engineering Repository, http://promise.site.uottawa.ca/SERepository/, Accessed 29 November 2022.

31. V. Lenarduzzi, N. Saarimáki and D. Taibi, The technical debt dataset, in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, Recife, September, pp. 2–11, 2019.
32. G. Gousios and M. Aniche, IN4334 – Machine Learning for Software Engineering, https://gousios.org/courses/ml4se/, Accessed July 2022.
33. F. Koetter, M. Kochanowski, M. Kintz, B. Kersjes, I. Bogicevic and S. Wagner, Assessing software quality of agile student projects by data-mining software repositories, in *11th International Conference on Computer Supported Education*, Crete, May, pp. 244–251, 2019.
34. S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, Trendowicz, A. M. Vollmer and S. Wagner, Software Engineering for AI-Based Systems: A Survey, *ACM Transactions on Software Engineering and Methodology*, **31**(2), pp. 1–59, 2022.
35. P. Heck and G. Schouten, Turning software engineers into AI engineers, *arXiv preprint arXiv:2011.01590*, 2020.
36. http://www.di.uniba.it/∼ lanubile/teaching.htm, Accessed 29 November 2022.
37. M. Besterfield-Sacre, L. J. Shuman, H. Wolfe, C. J. Atman, J. McGourty, R. L. Miller, B. M. Olds and G. M. Rogers, Defining the outcomes: A framework for EC-2000, *IEEE Transactions on Education*, **43**(2), pp. 100–110, 2000.
38. B. S. Bloom, C. of College and U. Examiners, *Taxonomy of Educational Objectives*, Longmans, Green New York, **2**, 1964.
39. M. A. Almulla, The effectiveness of the project-based learning (PBL) approach as a way to engage students in learning, *Sage Open*, **10**(3), pp. 1–15, 2020.
40. J. Bishop and M. A. Verleger, The flipped classroom: A survey of the research, in *ASEE Annual Conference & Exposition*, Atlanta, June, pp. 23–1200, 2013.
41. A. Begel and T. Zimmermann, Analyze this! 145 questions for data scientists in software engineering, in *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, May, pp. 12–23, 2014.
42. E. Scott, F. Milani and D. Pfahl, Data science and empirical software engineering, in M. Felderer, G.H. Travassos (eds.), *Contemporary empirical methods in software engineering*, Springer, pp. 217–233, 2020.
43. F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora and M. Di Penta, How open source projects use static code analysis tools in continuous integration pipelines, in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories*, Buenos Aires, May, pp. 334–344, 2017.
44. A. Serban, K. van der Blom, H. Hoos and J. Visser, Adoption and effects of software engineering best practices in machine learning, in *14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Bari, October, pp. 1–12, 2020.
45. Data science in software teams and AI engineering, https://zonavideo.upc.edu/video/616976440210803a34256956, Accessed 29 November 2022.
46. A. Van Deursen, M. Aniche, J. Aué, R. Slag, M. De Jong, A. Nederlof and E. Bouwers, A collaborative approach to teaching software architecture, in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle, March, pp. 591–596, 2017.
47. C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell and A. Wesslén, Experimentation in Software Engineering: An Introduction, Springer, 2000.
48. Materials for Project-based learning for teaching software analytics and best practices in data science studies, https://figshare.com/s/829d7d7f48e3b943830b, Accessed 29 November 2022.
49. C. A. González, L. A. Zumel, J. A. Acero, V. Lenarduzzi, S. Martínez-Fernández and S. R. Rodríguez, A preliminary investigation of developer profiles based on their activities and code quality: Who does what? in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security*, Hainan, December, pp. 938–945, 2021.
50. TAED2 Software Analytics. Project Causal LM for commit messages, https://zonavideo.upc.edu/video/618bd3810210805a8f1732d8, Accessed 29 November 2022.
51. TAED2 Software Analytics Project. Programmers segmentation based on their coding skills and assignment of a quality rating, https://zonavideo.upc.edu/video/618bc5820210805a1f54898f, Accessed 29 November 2022.
52. TAED2 Software Analytics Project. Human Resource Management Through Developer Technical Analysis, https://zonavideo.upc.edu/video/618bcda60210805a907fca17, Accessed 29 November 2022.
53. Advanced Topics in Data Engineering, https://zonavideo.upc.edu/series/615edd7a02108030f56d4d00, Accessed 29 November 2022.
54. Evaluation Cookbook, http://www.icbl.hw.ac.uk/ltdi/cookbook/cookbook.pdf, Accessed 29 November 2022.
55. D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka and M. Oivo, Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, **23**(1), pp. 452–489, 2018.

Silverio Martínez-Fernández is an Assistant Professor at UPC-BarcelonaTech since January 2020. He received his PhD Degree in Computing from the Universitat Politècnica de Catalunya in 2016. He was a Post-Doctoral Fellow of the European Research Consortium for Informatics and Mathematics (2016–2018) and operative project manager (2018–2019) in Fraunhofer IESE (Germany). Researcher with more than 50 peer-reviewed publications and H-factor 17 (according to Google Scholar). His interests include Empirical Software Engineering, Green AI, AI-based systems, Software Analytics, and Reference Architectures. In EU framework programmes, he acted as Evaluation WP leader in Q-Rapids (H2020, RIA), and participated in DESIRA (H2020, RIA). He has been PC co-chair in several international conferences and workshops: ESELAW@CIbSE 2021, PROFES 2019, CESI@ICSE 2018 and QuASD@PROFES 2017–2018. He is Editorial Board Member of the SCI-indexed journal IET Software (IEE). He has also been reviewer of multiple journals (e.g., IST, JSS, EMSE, ASE, DKE) and PC member of international conferences (e.g., ESEM, ICSME, ECSA, PROFES, CIbSE). Contact him at silverio.martinez(at)upc.edu.

Cristina Gómez Seoane received her PhD Degree in Informatics from the Universitat Politècnica de Catalunya in 2003. She is a member of the GESSI research group at the same university. Her main interests have been, and are, conceptual modeling, software engineering, software architectures and object-oriented design. She has taught extensively on these topics and also conducted research on these topics, which has been published in international journals and conferences.

Valentina Lenarduzzi is an Assistant Professor (tenure track) at University of Oulu (Finland). She is Finnish consortium and WG5 leader in the COST Action EUGAIN European Network For Gender Balance in Informatics. Her research activities are related to contemporary software development practices and methodologies, including data analysis in software engineering, software quality, software maintenance and evolution, focusing on Technical Debt as well as code and architectural smells. She got the PhD in Computer Science in 2015 and was a postdoctoral researcher at the Free University of Bozen-Bolzano, (Italy), at the Tampere University (Finland), and LUT University (Finland). Moreover, she was a visiting researcher at the University of Kaiserslautern (TUK) and the Fraunhofer Institute for Experimental Software Engineering IESE (Germany). She served as a program committee member of various international conferences (e.g., ICPC, ICSME, ESEM), and for various international journals (e.g., TSE, EMSE, JSS, IST) in the field of software engineering. She has been program co-chair of OSS 2021, TechDebt 2022, SEAA2023, PROFES 2023, and QUATIC2023. Moreover, she served in different organization roles (i.e. short papers, emerging results, and publicity chair) in several conferences such as ESEM 2021, SANER 2022, PROFES 2022, EASE2023, ESEM2023, and SANER 2024. She was also one of the organizers of the last edition of MaLTeSQuE workshop (2022) collocated with ESEC/FSE. Dr. Lenarduzzi is recognized by the Journal of Systems and Software (JSS) as one of the most active SE researchers in top-quality journals in the period 2013 to 2020.