Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in INGEGNERIA
INFORMATICA, GESTIONALE E DELL'AUTOMAZIONE

How students solve problems during Educational Robotics activities: identification and real-time measurement of problem-solving patterns

Ph.D. Dissertation of:

Lorenzo Cesaretti

Advisor:

Prof. David Scaradozzi

Curriculum supervisor:

Prof. Francesco Piazza

XXXII edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in INGEGNERIA
INFORMATICA, GESTIONALE E DELL'AUTOMAZIONE

Come gli studenti risolvono problemi durante le esperienze di Robotica Educativa: identificazione e misurazione in tempo reale delle strategie di problem-solving

Ph.D. Dissertation of:

Lorenzo Cesaretti

Advisor:

Prof. David Scaradozzi

Curriculum supervisor:

Prof. Francesco Piazza

XXXII edition - new series

Università Politecnica delle Marche
Dipartimento di Ingegneria dell'Informazione (DII)
Via Brecce Bianche — 60131 - Ancona, Italy

Acknowledgements

I would like to thank my supervisor, professor David Scaradozzi: his advises, his continuous support and his guidance were fundamental during these three years. A special thanks goes also to professor Eleni Mangina (University College Dublin): she helped me in the study and development of machine learning techniques applied to our dataset during my visiting period in Dublin.
All the research results, all the projects, all the "stuff" we created were possible thanks to a special colleague and friend, Laura Screpanti. Thanks for all the "scientific" (and not scientific) chats we had during the PhD. Thanks also to Arianna: her great professionalism has made possible an amazing event (Fablearn Italy 2019), the "icing on the cake" at the end of the PhD. Thanks to Nicolò, for cooking for me during our visiting period at Hopkins Marine Station (California).
Three years ago, I decided to follow my dreams (in practice: I quit a job, founded a startup and started a PhD) because the love of my life, my wife Giada, pushed me towards what is really important for me. We are trying to put into practice some words that I love to read:
"The very basic core of a man's living spirit is his passion for adventure. The joy of life comes from our encounters with new experiences, and hence there is no greater joy than to have an endlessly changing horizon, for each day to have a new and different sun." (Jon Krakauer, Into the Wild).
Thanks to my mother Mirella and my dad Giancarlo: they taught me that great sacrifice and hard work (and sometimes sleep a few hours in a night) is needed to create excellent things.
Thanks to my sister, Elisa, her husband Francesco and their daughter, Giorgia (the other love in my life) for all the laughs every Sunday evening. Thanks to my parents in law Valeria and Renzo, for boosting me in this adventure and for giving me my first Educational Robotics book.
Thanks to Elisa and Michele, unique friends, every day involved in the building, brick by brick, of our projects TALENT and Weturtle. Their decision to invest in my research project was a turning point in my life.
Thanks to all the TALENT collaborators and friends: Federico, Luca, Nico, Susanna, Roberto, Alessandra, Elena. Their patience in collecting log files following the protocol was incredible. Thanks to all my friends Francesco, Cecilia, Alessio, Giulia, Marco, Kelly, Marco, Cecilia, Gian-Luca, Elisa, Jacopo, Giulia, Don Andrea, Maila and Paolo for being with me during this journey. Thanks to Michele and Alessandro (and to Pizzeria La Grotta) for all the relaxing moments with good food and good beer (Coke for Michele).
Finally, thanks to all the students and teachers involved in the experimentation presented in the following pages: three years ago I left a job as robotics engineer, but I found the most beautiful job in the world.

“Don't settle down and sit in one place. Move around, be nomadic, make each day a new horizon. You are still going to live a long time, Ron, and it would be a shame if you did not take the opportunity to revolutionize your life and move into an entirely new realm of experience.”
(Jon Krakauer, Into the Wild)

“Space is big. You just won't believe how vastly, hugely, mind-bogglingly big it is. I mean, you may think it's a long way down the road to the chemist's, but that's just peanuts to space.”
(Douglas Adams, The Hitchhiker's Guide to the Galaxy)

This dissertation is dedicated to all my grandparents (Gina, Duilio, Santa, Ugo).

Abstract

This dissertation aims to provide the results through the utilisation of data mining and machine learning techniques for the assessment with Educational Robotics (ER).
This research work has three main objectives: identify different patterns in the students' problem-solving trajectories; predict the students' team final performance, with a particular focus on the identification of learners with difficulties in the resolution of the ER challenges; analyse the correlation of the discovered patterns of students' problem-solving with the evaluation given by the educators.
We analysed the literature on Educational Robotics' traditional evaluation and Educational Data Mining for assessment in constructionist environments.
An experimentation with 455 students in 16 primary and secondary schools from Italy was conducted, through updating Lego Mindstorms EV3 programming blocks in order to record log files containing the coding sequences designed by the students (within team work), during the resolution of two preliminary Robotics' exercises (Exercise A and B).
The collected data were analysed based on data mining methodology. We utilised five machine learning techniques (logistic regression, support vector machine, K-nearest neighbors, random forests and Multilayer perceptron neural network) to predict the students' performance, comparing two approaches:

- a supervised approach, calculating a feature matrix as input for the algorithms characterised by two parts: the team's past problem-solving activity (thirteen parameters extracted from the log files) and the learners' current activity (three indicators for Exercise A and four indicators for Exercise B); and
- a mixed approach, applying an unsupervised technique (the k-means algorithm) to calculate the team's past problem-solving activity, and considering the same indicators of the supervised approach representing the students' current activity.

Firstly, we wanted to verify if similar findings emerged comparing younger students and older students, so we divided the entire dataset in two subsets (students younger than 12 years old and students older than 12 years old) and applied the supervised and mixed approach in these two subgroups for the first exercise, and a clustering analysis for the second exercise. This process demonstrated that similar problem-solving strategies were applied by both younger and older students, so we aggregated the dataset and performed the supervised and the mixed approach comparing the performances of these two techniques considering the entire dataset.
The results have highlighted that MLP neural network with the mixed approach outperformed the other techniques, and that three learning styles were predominantly emerged from the data mining. Furthermore, we deeply analysed the pedagogical meaning of these three different approaches and the correlation of the discovered patterns with the performance obtained by learners. We denote the added value of data mining and machine learning applied to Educational Robotics research and highlight the significance of further implications. Finally, we discuss the future further development of this work from educational and technical view.

Contents

List of Figures

List of Figures in Appendices

List of Tables

List of Tables in Appendices

Chapter 1.

1 Introduction

1.1 *Educational Robotics*

Nowadays Educational Robotics (ER) is increasingly spreading in schools all over the world (Miller & Nourbakhsh, 2016), thanks to teachers and educators that are using this approach during the course of their standard lessons. The expression "ER" is more and more cited in the educational context and policymakers have decided to invest funds in the students' and teachers' training (see for example in Italy the National Plan for Digital School, PNSD (2019), PON Digital Creativity - PON Creatività Digitale (2019) and "In Summer we learn STEM - In Estate si imparano le STEM" (2019)).

In this situation also the academic community is paying more attention to the study of the effects of the use of robots in the educational environment: Angel-Fernandez and Vincze (2018) showed how the number of scientific papers using the words "robotics" and "education", or the expression "ER" has increased in the last two decades.

But how can we define ER? It is not enough introducing robots in an educational setting to propose an ER activity (Scaradozzi, Screpanti & Cesaretti, 2019): ER is characterized by a workflow that allows students to design, build and program robotic artefacts, creatively solve problems and carry out meaningful projects.

ER can facilitate active teaching and learning, promote active reasoning and critical thinking, and also enhance students' engagement and interest, while motivating them to address complex or abstract subjects (Cheng, Sun & Chen, 2018). ER is a growing field with the potential to significantly impact the nature of engineering and science education at all levels (Cho, 2011; Barker, Nugent, Grandgenett & Adamchuk, 2012).

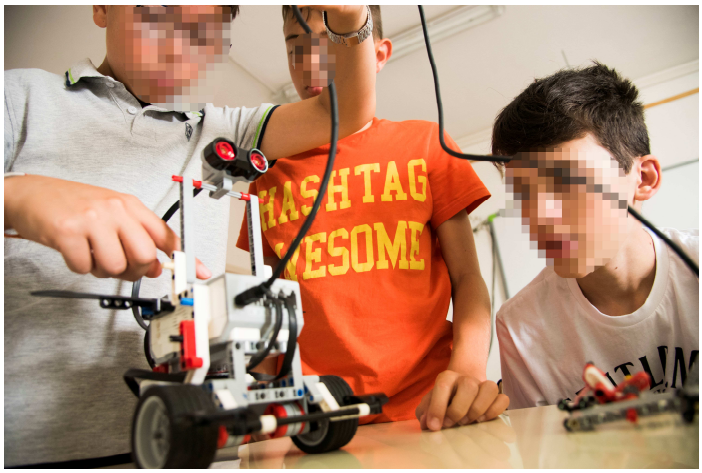

Figure 1. Robot built by students during an ER activity

In 1980 mathematician and pedagogist Seymour Papert proposed the constructionist approach (Papert, 1980), the pedagogical theory at the base of ER: the construction of a personal and meaningful artefact (a robotic artefact, considering ER) promotes the construction of deep knowledge and "can provide children with new possibilities for learning, thinking, and growing emotionally as well as cognitively" (Papert, 1980). Moreover, the sharing of this product with the educational community (peers' group, teachers, parents) makes the artefact a "public entity" (Papert, 1991) and provide a richer learning experience to the pupils. Papert contributed to the design of one of the first examples of Educational Robotic kits, the Lego / Logo construction kit (Resnick, Ocko & Papert, 1988), proposed in 1988 by the MIT Media Laboratory.
Lego Mindstorms EV3 Education (2019) is the kit chosen for the experimentation presented in this dissertation, and it's one of the successors of the Lego / Logo kit (and the name of the kit, "Mindstorms", is a way to thank Papert remembering the title of his main publication "Mindstorms. Children, Computers and Powerful Ideas"). Teachers can propose to carry out ER activities in their class through the use of Lego Mindstorms EV3. Since constructionism is the pedagogical approach at the base of ER, when students design, build, program, debug and share a robot artefact they explore the everyday phenomena in their lives in a new and playful way (Resnick, Martin, Sargent & Silverman, 1996), improving their problem-solving and thinking capabilities.

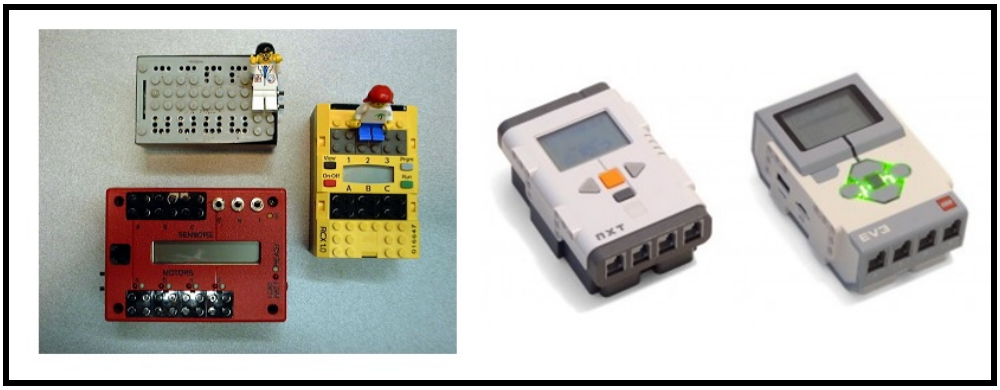

Figure 2. From Lego/Lego (grey brick in the upper left corner), to Lego Mindstorms EV3 block (grey and white brick on the right-hand side)

The evaluation of constructionist activities could be difficult for teachers: as Berland, Baker and Blikstein (2014) suggest, what students learn thanks to a constructionist approach (such as ER) is hardly detected via analysing scores obtained through standard tests (questionnaires, multiple choice questions, etc.). These activities are characterised by a personal design process: documenting, analysing and evaluating this process can be costly in terms of time for an educator. Moreover, what and how to assess the artefact designed by students it's a question without a clear answer from the academic community. Some alternative strategies have been proposed to evaluate not only the final product, but also the underlying process. Papert (1980) proposed to consider feedback provided by detailed peer critiques or actual use of a student's tool in an authentic setting (as reported by Berland et

al. (2014)), other researchers proposed students' interviews, the development and analysis of portfolios (Brennan & Resnick, 2012; Tangdhanakanond, Pitiyanuwat & Archwamety, 2006) and MIT (Playful Journey Lab) in collaboration with MakerEd proposed the "Beyond Rubrics Toolkit", a collection of tools to support embedded assessments in maker-centered classrooms (Introducing the Beyond Rubrics Toolkit, 2019).

The latest developments in the field of data mining and machine learning consent to explore and extract new, valuable and non-trivial information and relationships in large volumes of data (Kantardzic, 2011), instead of trying to test prior hypotheses (Collins, Schapire & Singer, 2004). Applying data mining and machine learning methods to data collected from the educational environments (Educational Data Mining, EDM) can allow to predict and classify students' behaviours and discover latent structural regularities to large educational dataset (Berland et al., 2014). The raw data coming from educational systems (for example web platform used by students) are transformed into useful and meaningful information that could be strongly helpful for the educational research and practice (Dutt, Ismail & Herawan, 2017). In recent years some research projects in the educational fields used these techniques to predict the students' success probability, analysing their behaviour during the interaction with software applications or with online platforms (Beck & Woolf, 2000; Merceron & Yacef, 2004; Piech, Bassen, Huang, Ganguli, Sahami, Guibas, et al., 2015; Ornelas & Ordonez, 2017), or analysing school career related data (Asif, Merceron, Ali & Haider, 2017; Fernandes, Holanda, Victorino, Borges, Carvalho & Van Erven, 2019). Creating systems able to gather and organize data during open-ended and constructionist activities could let us deeply explore the problem-solving and design process performed by learners, using machine learning algorithms; these developments could give a great support to teachers during the evaluation phase of learners. Artificial Intelligence will aid educators in the analysis of the fine-grained data collected during the interaction between pupils and technological devices (laptops, tablets, educational robots, etc.), and will give interesting insights related to the students' usage of these materials, to their cognitive process and to their problem-solving styles (Luckin, 2017); from a metacognitive point of view this approach could improve learners' self-awareness and could allow them to discover the best personalised ways to successfully reach educational objectives (Winne & Baker, 2013).

In the ER field machine learning and data mining techniques can offer a strong contribute: data related to the utilization of the robot could be recorded and analysed, in order to verify how students have designed the robotic artefact, how they have solved a problem or how they have created a programming sequence.

This dissertation presents a research work that aims to implement this approach: thanks to the upgrade of the Lego Mindstorms EV3 programming blocks it was possible to register some log files containing the programming sequences created by 455 Italian Primary and Secondary school students (organized in teams), during the resolution of **two introductory Robotics exercises**. Using the collected data as inputs for machine learning algorithms, this work explores the following ideas:

- identification of different patterns in the students' problem-solving trajectories;
- accurate prediction of students' team final performance; and
- correlation of the discovered patterns of students' problem-solving with the evaluation given by the educators.

The Literature Review with analysis of related work is provided within the following section. The traditional assessment methods applied to ER and EDM for evaluation in constructionist environments were taken into consideration. A description of the tracking system designed, how the data were collected during the experimentation and the details of the machine learning algorithms that have been applied are reported in the second section of this dissertation; moreover, in this section an abstract description of the data structure is given, with the aim of making this experimentation replicable using different robotic kits. Chapter 3 (Results) at first presents some insights of the collected data, in order to allow some preliminary considerations related to the presented approach; then it proposes the results obtained in terms of prediction and the patterns discovered based on the input data. Finally, the conclusions of the analysis based on the results from an educational point of view (Chapter 4) are presented along with the future developments in order to improve the presented approach.

1.2 Literature Review

1.2.1 Traditional assessment methods in Educational Robotics

ER have been considered as an educational approach that allows students to design, build, program and debug robotic artefacts (both hardware and software components) (Denis & Hubert, 2001). Educators have considered different strategies to utilise this approach, related to the learning environment where the activities take place (formal or non-formal), to the impact on curriculum of the ER activities and to the evaluation procedures chosen in order to quantify learning outcomes (qualitative, quantitative or mixed methods) (Scaradozzi, Screpanti & Cesaretti, 2019). ER not only helps students in the exploration of the main concepts of Robotics (Castro, Cecchi, Valente, Buselli, Salvini, & Dario, 2018), but also allows them to build knowledge related to Science Technology Math Engineering (STEM) disciplines (Cristoforis, Pedre, Nitsche, Fischer, Pessacg, & Di Pietro, 2013; Kim, Kim, Yuan, Hill, Doshi, & Thai, 2015) and in some cases also related to Arts and Humanities (Montero & Jormanainen, 2016; Jeon, FakhrHosseini, Barnes, Duford, Zhang, Ryan et al., 2016).
In the ER field evaluation is considered a crucial activity and researchers have identified lack of quantitative analysis on how robotics can improve skills and increase learning achievements in students (Benitti 2012; Alimisis, 2013). For this reason, in the last years, different studies focused on the assessment of ER projects, adopting qualitative, quantitative or mixed methods for the collection of educational data.
ER activities, as previously reported, are based on the constructionist theory: students learn when they are involved in meaningful experiences, characterized by the construction of an artefact (a robot, a mechanism, a computer program, etc.). Furthermore, in this kind of activity students usually have to design a solution for a given open-ended problem, related to specific software aspects (for example creating a pre-defined robotic behaviour) or to hardware aspects (designing and assembling a robot that meets specific requirements).
Generally, there are unlimited options that students can explore and implement.
These features make ER activities particularly suitable for a qualitative evaluation: through observations, interviews, focus groups, essays and projects' analysis, teachers can

extrapolate useful information about the problem-solving and learning paths adopted by their students. Nevertheless, this methodology could be strongly affected by the human bias and by external factors. Several research studies in the ER field reported this type of assessment: Denis and Hubert (2001) observed interactions among students during ER activities, and thanks to the introduction of an observation grid they tried to analyse improvements in teamwork and problem-solving skills. Liu (2010) interviewed students previously involved in ER lessons, investigating the learners' perceptions of educational robots and learning of robotics. This study proposed three clusters resulting from the interviews' analysis (robotics as a plaything, as a source of employment, and as a way to high technology). Elkin, Sullivan and Bers (2014) evaluated the introduction of a robotics' curriculum in a Montessori classroom, collecting surveys, interviews, and impressions from the personal blog written by the teacher responsible for the project and they proposed best practices for integrating programming and engineering concepts into Montessori education.
Qualitative assessment was useful also to evaluate activities outside school hours, for example during robotics training camps, as experimented by Ucgul and Cagiltay (2014). Interviews with children and instructors, observations, field notes, and camp evaluation forms were utilised by the researchers to discover the best educational approach to adopt in this type of experience, and the participants' preferred activity. After-school program activities related to STEM are another example of non-curricular opportunities where the same approach was applied (Sahin, Ayar, & Adiguzel, 2014).
On the other hand, quantitative assessment focuses on standardized tests, questionnaires and exercises that allow to obtain more reliable numeric outputs, so that students' performances can be analysed and compared. Summarising learning through a quantifiable parameter sometimes could be reductive, especially in an open-ended constructionist environment, where what is learned indirectly results in quantitative test outcomes (Berland et al., 2014). In spite of this consideration, the only way to manage large scale studies until now, trying to discover the correlation between ER and the development of skills or a better understanding of curricular subjects, is the quantitative approach. Some research projects propose a pre and post assessment (administering a questionnaire), evaluating the students' technical (programming, robotics) and social skills (teamwork, problem-solving, self-efficacy) and their attitude towards STEM (Kandlhofer & Steinbauer, 2016; Atmatzidou & Demetriadis, 2016; Castro et al., 2018) or the teachers' attitude toward ER (Scaradozzi, Screpanti, Cesaretti, Storti, & Mazzieri, 2019). Other case studies propose only a post evaluation (Cesaretti, Storti, Mazzieri, Screpanti, Paesani, & Scaradozzi, 2017; Screpanti, Cesaretti, Storti, Mazzieri, Longhi, Brandoni et al., 2018; Screpanti, Cesaretti, Marchetti, Baione, & Scaradozzi, 2018).
To obtain the benefits from both the approaches, especially in a complex field like ER, some studies show a mixed approach, using quantitative and qualitative tools, during teachers' training courses and activities with students (Kim et al., 2015; Chalmers, 2018; Ferrarelli, Villa, Attolini, Cesareni, Micale, Sansone et al., 2018; Angeli & Valanides, 2019). In this case, the possibility to deeply analyse the students' learning process requires an expensive work in terms of time required and human resources and it's not applicable to a large number of students. This dissertation focuses on a potential improvement of the evaluation within ER activities based on the recordings from every single step performed by the students during the design of their solution. From these recordings, the investigation of the most important features takes place in order to identify different patterns in the

problem-solving path, which could be then analysed in relation to the performance reached by the students.
An ER activity is usually characterized by a cyclical experimentation, where learners modify their programming sequence or their robot's hardware structure, trying to reach a specific goal set by the teacher or by themselves. A suitable model that represents this way of working is the "Think Make Improve" (TMI) cycle (Martinez & Stager, 2013) where the following steps are taking place (see also Figure 3):

- students usually start thinking about how to solve a problem or how to create a prototype that meets certain requirements (Think); they can adopt a "tinkering", a "planning" approach or a mix of them, and it's important that the educator doesn't force them to take a certain and more scholastic path (Resnick & Rosenbaum, 2013);
- students try to realise the solution by building and programming the robot (Make);
- at the end of the cycle, students observe their artefacts and try to debug or improve them (Improve); in this phase they discuss with peers and/or with the expert, and play with their robot, trying to figure out what's going wrong (or what to improve). From the analysis of the real feedback of the robot, students decide what changes to implement in the software or in the hardware.

The cycle starts again with "Think" phase: if the students' team identify a feature to modify, they have to decide what changes to implement (hardware or software) and the best strategy to pursue.

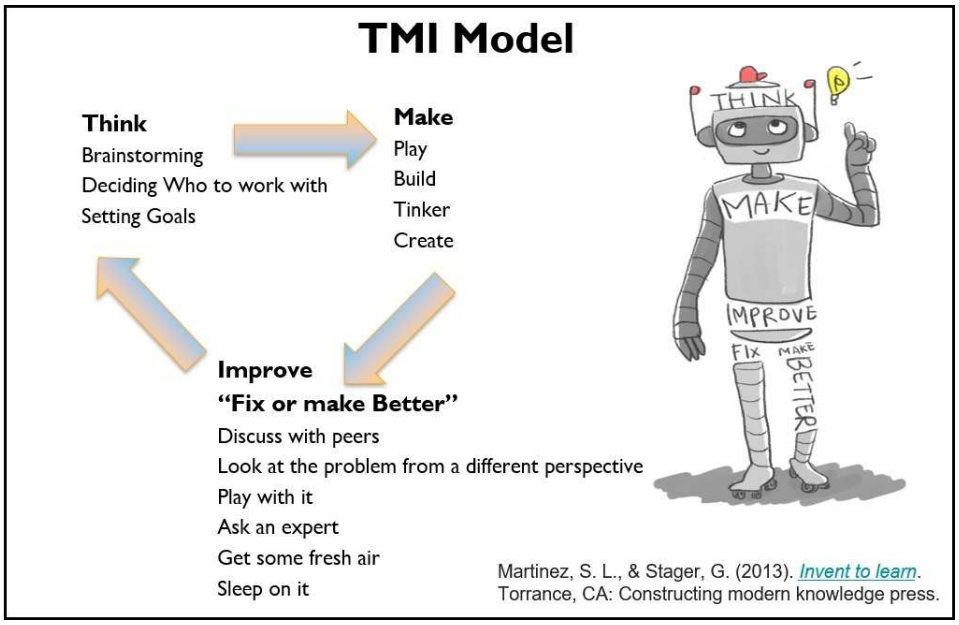

Figure 3. The TMI cycle (Martinez & Stager, 2013).

A similar model was developed by LEGO, the "4C" framework (LEGO 4C Framework, 2019):

- Connect: pupils are presented with a challenge or task that is open-ended and that places them in the position of solution-seekers.
- Construct: students build prototypes to solve the challenge or to carry out the task (when students construct artefacts in the world, they simultaneously construct knowledge in their minds)
- Contemplate: learners are given the opportunity to consider what they have learned and to talk about and share insights they have gained during the Construct phase. In this phase they could decide what to improve in their artefact.
- Continue: every task ends with a new task that builds on what has just been learned. This phase is designed to keep the learner in 'a state of Flow' (Csikszentmihalyi, 1997).

TMI is the framework for our experimentation phase, therefore features extracted from this process will be meaningful characteristic from this way of approaching an ER activity. This could be useful, amongst other things, because it lays the basis of a systematical analysis of the impact of ER into such framework. Thus, EDM and machine learning will enhance the perspective of assessment.

1.2.2 Assessment of students' problem-solving performances

In this PhD dissertation we will focus on the identification of learners' problem-solving patterns during ER activities. In the educational context other studies proposed some strategies for an assessment of the students' problem-solving performances.
Greiff, Holt, and Funke (2013) classified problem-solving in three categories (based on PISA 2003, PISA 2012 and PISA 2015):

- analytical problem-solving [paper-pencil based analytical problem solving, defined as "an individual's capacity to use cognitive processes to resolve real, cross disciplinary situations where the solution path is not immediately obvious" (OECD, 2004)];
- interactive problem-solving [characterized by the dynamic interaction between a problem solver and the problem to generate and integrate information about the problem; administering interactive problems through computers or technological devices, it's possible to gather and analyse process data (computer-generated log files) created by the interaction between students and the software interface];
- collaborative problem-solving (the problem solver interacts with a task and collaborates with several problem solvers during the resolution process).

A theoretical formulation for the assessment of paper-pencil based analytical problem solving (Deek, Hiltz, Kimmel and Rotter, 1999) divided the students' process into three phases: formulating the problem, planning the solution, designing the solution; this method took into consideration the thinking skills required by students learning how to program. Furthermore, this formulation proposed four dimensions for the evaluation of the learners' final product: efficiency, reliability, readability, solution correctness.

Considering the interactive problem-solving Baker and Mayer (1999) proposed some components required to assess student problem solving in technology environments; starting from a definition of problem solving, they identified some issues in the assessment of this skill and proposed some techniques and standards for measuring the quality of student understanding (the cognitive analysis, the CRESST model of learning).
Some experimentations of software environments related to problem-solving (interactive problem-solving) were implemented by Bottge, Rueda, Kwon, Grant, and LaRoque (2009) and Shute, Wang, Greiff, Zhao and Moore (2016); the first one compared a computer-based test and paper pencil test: data collected thanks to the computer-based test revealed that the weak students were able to navigate the test, spent about the same amount of time solving the subproblems as the more advanced students, and made use of the learning scaffolds. The second one measured middle-school students' problem solving skills using a videogame ("Use Your Brainz"); these researchers chose Bayesian networks to analyse data gathered during three-hours session of students' playing and compared this analysis with two external problem solving measures (Raven's Progressive Matrices and MicroDYN): their results indicated that the problem solving evaluation obtained from the videogame significantly correlated with the external measures.
Cooper, Cox Jr, Nammouz, Case and Stevens (2008) tried to assess collaborative problem solving: they involved in a research study 713 students enrolled in the first semester of a general chemistry course for science and engineering majors; they utilised a suite of software tools and pedagogies of small-group student collaborations to assess both student problem-solving strategies and student abilities as they change over time: their results showed that using small group interaction improved students' problem solving skill.

1.2.3 Machine learning and data mining for assessment in constructionist environments

There are several examples of the application of data mining and machine learning techniques for the evaluation of students during constructionist activities, but **all these research projects concern programming**, and not the use of robots. Table 1 denotes an overview of the main aspects of these studies (features selected, implemented machine learning techniques and obtained results) and provide the motivation for the work described in this dissertation.

To the best of our knowledge, only one experimentation used data mining and machine learning in the field of ER (Jormanainen and Sutinen, 2012), but in a very small scale. They designed a new visual programming environment that allowed to program the Lego Mindstorms RCX and gathered data about the students' interactions with the fundamental functions of the software. The study's objective was the creation of an Open Monitoring Environment, a system that tried to find the current students' learning behaviours during robotic activities, showing them to the teachers. This system, using trees algorithm (J48 implementation), classified the data entries into four classes that marked the observed students' group's progress, indicated by colour: white (neutral situation), if students were not progressing without showing problems, green if students were progressing without problems, yellow if teacher's intervention was required soon (high probability that students

were facing problems), red if teacher's intervention was certainly required. There were some weaknesses in this experimentation; first, the kit utilised in the study was obsolete, as in 2012 the Lego Mindstorms NXT (the new model) had been on the market since 2006 (Lego Robotics History, 2019); second, it was a very small scale study (only 12 students and 4 teachers from primary school were involved); third, the authors defined their new programming environment graphical, but students had to create textual sequences of instructions (clicking on a set of given functions), maybe a block-based approach could be more friendly for primary school pupils.

Ahmed, Lubold and Walker (2018) presented an interesting system that gives feedback to pupils in real-time, while they are programming the Lego Mindstorms EV3 robot. But in this case, the researchers did not train their system using machine learning techniques but using deterministic rules.

Berland, Martin, Benton, Petrick Smith and Davis (2013) registered students' programming actions and used clustering (k-means algorithm) to study different pathways of novice programmers (53 high school girls participated to the study). This led to the identification of three general patterns (Tinkering, Exploring, and Refining) and to the definition of the EXTIRE framework. They analysed how students' programming activity changed over time and how these changes relate to the quality of the programs that students were writing. The researchers discovered that the quality of the programming sequences designed by the students was higher in two of the above-mentioned phases (tinkering and refine).

Blikstein, Worsley, Piech, Sahami, Cooper and Koller (2014) presented a study that focused on how students learn computer programming, analysing if different programming styles affect students' achievements. The research project was based on data drawn from 154,000 code snapshots of computer programs under development by approximately 370 students enrolled in an introductory undergraduate programming course. They used methods from machine learning to discover patterns in the data and tried to predict midterm and final exam grades. They performed 2 different types of analysis: the first one looked for patterns across several assignments and tried to find correlations between these patterns and students' assignment results or exam grades. They considered how many lines of code and characters were added, modified and deleted, without looking at the content of the code: indeed, extracting this kind of data is a good choice taking into account computational cost and usefulness. The second one examined in detail one single assignment, building a machine learning–induced progression maps (using Hidden Markov Model (Rabiner & Juang, 1986) to represent students' programming states), and demonstrating that the topologies of such maps are correlated with course performance. In particular, they used a variety of machine-learning techniques to translate the myriad code logs into maps of states that show the progress of the students' work and the correlation of this work with student final performance.

These two research projects (Berland et al., 2013; Blikstein et al. 2014) extended the previous work by Turkle and Papert (1992), that proposed two opposite approaches to programming: the "bricoleur scientist" (the tinkerer, in these studies), that prefers "a negotiational approach and concrete forms of reasoning", and the "planner scientist", that prefers "an abstract thinking and systematic planning".

Chao (2016) analysed the patterns in visual programming showed by novice programmers during problem-solving activities related to computational problems, and the differences in the students' computational design and performances within these different patterns. The

participants in this study were 158 college students majoring in information communication from three classes at the same university in northern Taiwan. Chao chose different indicators for the study (classified into 3 categories, Computational practice, Computational design and Computational performance), using only five of them to identify diverse students' behaviours: sequence, selection, simple iteration, nested iteration and testing. The participants' frequencies of the five indicators were analysed by cluster analysis (Ward's minimum variance (Ward, 1963) and k-means algorithm (Bock, 2007)); four clusters were obtained: sequent approach (students recognizable by their prevalent linear progression in solving computational problems, using mostly sequence or simple iteration); selective approach (students identifiable by their selective and divergent approach to computational problem-solving, using frequently nested iterations and selection); repetitious approach (participants appeared capable in discovering repeated patterns, and with a very high number of simple iteration blocks); trial approach (students that showed a trial and error approach, applying different control-flow blocks during the programming activity and evaluating the effect of their implementation). Comparing the computational performance among these patterns, the results showed that the trial approach seemed to have a significantly lower performance and a worse capacity to create efficient programming sequences than the sequent approach and the repetitious approach. The selective approach and the repetitious approach showed good outcomes related to the programs' efficiency.

Wang, Sy, Liu and Piech (2017) collected log data from one of the most popular platforms for introducing young students to Computer Science and programming, Code.org. They fed the embedded program submission sequences into a recurrent neural network (a Long Short Term Memory architecture (Hochreiter & Schmidhuber, 1997)) and trained it on two tasks of predicting the student's future performance. They represented each program using an Abstract Syntax Tree, and converted into program embeddings using a recursive neural network. This research study implemented two experiments, using the "Exercise 18" in Code.org environment (1263360 code submissions, made by 263569 students). Knowing the student's trajectory (namely the sequence of code submission attempts), the success or the fail in the completion of the next programming exercise in the same course was predicted. Furthermore, the researchers evaluated the possibility to predict the success in a programming exercise considering only a certain number of student's attempts to solve that test. This work presented interesting aspects in terms of the transformation of students' code attempts in program embeddings over time (other studies are characterized by hand-picked features), and the distinct clusters obtained through this representation, that can predict future students' performance.

Bey, Pérez-Sanagustín and Broisin (2019) identified three clusters in a dataset created collecting programs from 100 students registered to a three-week course on the essential of Shell programming; they also determined some students' behavioral trajectories correlated with their performance at the final exam. In this paper unsupervised clustering techniques were applied (mixture Gaussian Clustering algorithm) for automatically identifying learners' programming behaviour.

Filvà et al. (2019) used k-means technique on data generated by students' clicks in Scratch (and not on handpicked features), with the objective to categorize learners' behaviour in programming activities: they identified three different patterns and a strong correlation between these behaviours and the evaluation given by some teachers involved in the research project, using a rubric for programming assessment. The patterns identified were:

the “Blocked development” (those students who are behind in the coding of the program), the “Development at a normal pace” (those students who have a balance of development between the graphics interface of the program and the coding) and the “Rapid development based on trial–error” (those students who make rapid changes in the program and constantly check the results). These researchers affirmed that there is a correlation between those projects in which students make rapid iterations of trial and error and good results.

To evaluate different aspects of constructionist activities, other works relied also on external sensors (cameras, microphones, physiologic sensors), and automated techniques, like text analysis, speech analysis, handwriting analysis, and others (Blikstein & Worsley, 2016). A key for future developments and experimentations will probably be connected to the availability and cost of implementation of such technological solutions for classroom assessment. External sensors may be more expensive, whereas embedded software solutions and machine learning algorithms could be effective and reliable in extracting evidence of students’ learning process and helping teachers to provide personalised feedback to students.

The majority of the studies presented above (Berland et al., 2013; Blikstein et al. 2014; Chao, 2016; Wang et al., 2017; Bey et al., 2019, Filva et al. 2019) applied machine learning techniques and data mining approach to data gathered from students during programming activities without the presence of physical robots, obtaining good results in the identification of different patterns in specific tasks. Some of the research results show also a correlation between the patterns and the final grades obtained by students. This dissertation takes the existing work further through the utilisation of educational robots. ERs have a distinctive feature, compared to the programming activity of a virtual avatar as the student can seek constant feedback from the physical object during the programming process, through a number of observations and analysis of the robot’s feedback. These processes promote a recursive activity that tries to optimise the prototype’s behaviour and to reach the desired performance (Sullivan, 2008; Mikropoulos & Bellou, 2013). Within the research described in this dissertation, the data mining techniques applied to data collected during ER projects will allow an analysis of how students understand the robot’s feedback. In a field like ER where quantitative measures are fundamental but maybe restrictive, the use of these new techniques could add important insight to the state of the art. Furthermore, the growing spread of ER projects in schools all over the world and the growing interest on this approach shown by teachers and policy-makers justify the research of new approaches that allow a rich and deeper evaluation of the students’ results. The study proposed by Jormanainen and Sutinen (2012) doesn’t attempt to investigate diverse learning paths, but to identify when students have difficulties in problem-solving and programming tasks. These specific gaps in the current research have prompted the research described in this doctoral thesis.

Table 1. Features, Machine Learning techniques and outcomes of studies carried out in constructionist environments.

Paper	Features selected	ML techniques	Results
Blikstein et al. (2014) [1st experiment]	Code update differential, characterised by: number of lines added, lines deleted, lines modified, characters added, characters removed, characters modified.	Simple regression between exam grades and average size (calculated considering one data point per assignment, they had 4 assignments) of the code updates per student.	No significant results.
Blikstein et al. (2014) [2nd experiment]	Code update differential (see above).	X-means clustering algorithm (applied to data obtained dividing each assignment into four equal time-based segments and extracted the average of each segment).	The clusters appeared to be significantly different from each other; the results with regard to grades showed effect sizes on the order of 0.2 (quite small).
Blikstein et al. (2014) [3rd experiment]	Code update differential (see above).	X-means clustering algorithm (applied to the previous dataset, divided calculating proportion of large, medium, and small changes).	The clusters (2) were quite distinct. The difference in average final grades corresponded to a Cohen's d of 0.21 (quite small).

Blikstein et al. (2014) [4th experiment]	Code update curves (combination of frequency and size in changes made by students).	Dynamic time warping and scaled dynamic time warping distance (to calculate the difference between two given code update curves). Considering four assignments, calculated (for the same student) how different his or her curve was related the other assignment.	The results indicated that there was a connection between students' grades and the amount of change in their programming patterns. The results were not statistically strong.
Blikstein et al. (2014) [5th experiment]	Modelling of a student's trajectories as an HMM (Hidden Markov Model, each state of the model represents a milestones of the assignment).	Two clustering algorithms: k-medioid and hierarchical agglomerative clustering (to compute the different states of the HMM). Expectation maximization algorithm to compute both the transition and emission probabilities in the state diagram. To find patterns in students transition, clustered the paths through the HMM using a method developed by Smyth (1997).	After defining the 3 clusters, they found that the group a student was clustered into was predictive of his or her midterm grade. The distinction between the groups contained in two clusters was particularly large, having a mean midterm score difference of 7.9% ($p < .04$, two-tailed t test)."
Berland et al. (2013)	Measures of Individual Program States (these measures were calculated for each program state): Action (number of action	X-Means clustering algorithm	They found 6 different clusters (Active, Balanced, Compact, Logical, Minimal, Testbed), divided these clusters in 3 phases

	primitives), Logic (number of logic and sensor primitives), Unique primitives (number of unique action, logic, and sensor primitives), Length (number of primitives, equivalent to lines of code in other programming languages), Coverage (the percentage of possible combinations of sensor inputs for which the program state generates actions).		(Explore, Tinkering, Refine) and found that Quality Program is higher in 2 of the above cited phases (Tinkering and Refine). The number of program quality scores (low, average, and high) differed significantly for each period, $\chi 2(4, N = 53) = 441.35$, $p < .05$ quality programs per period became significantly better over time ($F = 5.639$, $p < .01$, partial $\eta 2 = .163$).
Jormanainen & Sutinen (2012)	6 events: Add statement, Add command to code, Remove line, Upload program to robot, Compiling errors, Sum of all these events.	Decision trees, decision tables, Bayesian networks, and multi-layer perceptrons to predict students' progress. To measure the accuracy of the tested algorithms, we used the 10-fold cross-validation method.	The J48 algorithm (87.1% accuracy), best-first decision tree (BFTree) algorithm (82.3% accuracy), and multi-layer perceptrons (MLP) algorithm (82.3% accuracy) outperformed other algorithms in the classification of the data in 4 states (green, white, yellow, red).

Chao (2016)	Related to Computational practice (5 measures): Sequence, Selection, Simple iteration, Nested iteration Testing.	Ward's minimum variance method (to identify number of clusters), followed by the k-mean cluster analysis (on the identified cluster number). Separate ANOVAs and post hoc tests (for descriptive purpose) with the cluster group as an independent factor were conducted to determine the difference of the participants' computational practices among different clusters by comparing the indicators.	They obtained 4 clusters of different approaches (Sequent approach, Selective approach, Repetitious approach and Trial approach). Trial approach groups' had a significantly lower performance in solving the computational problems and in designing efficient sequences. Comparing the differences between groups in different clusters they obtained: for students' performance F(ANOVA) = 4.386 ($p < .001$), for sequences' efficiency F(ANOVA) = 9.139 ($p < .0001$).
Wang et al. (2017) [1st experiment, task A]	A student's trajectory consists of all the program submissions, which are represented as ASTs (that contain all the information about a program and can be mapped back into a program). These ASTs are converted into program embeddings using a recursive neural network.	Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN).	For both the pathscore baseline model and the LSTM model, they used 90% of the data set to perform training and validation and the remaining 10% for testing. The LSTM model outperforms the path score baseline by around 5% on test accuracy at every trajectory length.

Wang et al. (2017) [2nd experiment, task B]	Same features of their 1st experiment, from ASTs they calculated program embeddings.	LSTM RNN	They compared LSTM results with a Logistic regression algorithm. Logistic regression had higher performance considering Precision (0.7), but LSMT had higher performance considering Recall (0.8).
Bey et al. (2019)	Number of Submissions, Average time between two submissions, Average number of changes, Percentage of syntactical errors, Time standard deviation (the standard deviation of the average time between two submissions), Code standard deviation (the standard deviation of the average number of changes).	Mixture Gaussian Clustering algorithm	Cluster 1: students who submit frequently a large number of submissions with short time intervals between submissions, and who make irregular changes in their source code characterized by the most important number of syntactical errors. Cluster 2: students who do not submit an important number of submissions, as they spend more time to make significant changes in their source code. Students of this cluster make however a significant number of syntactical errors as well. Cluster 3: students who submit the lowest number of submissions even if they need a short period of time between two submissions.

Filva et al. (2019)	Clickstream	K-means cluster analysis	Strong correlation (0.93) between the evaluation by rubric and the clicks to the green flag button. The p-value of the statistical test is less than 0.05.

Chapter 2.

2 Methodology

2.1 *Methodology description*

This dissertation aims to analyse different problem-solving paths that characterize students' team workflow during an ER lesson by focusing on the step-by-step software development carried out by learners in order to solve the challenges proposed by the educators.
In order to capture behaviours from the field and to perform a deep analysis of the programming sequences designed by students (and of the problem-solving strategies that could be extrapolated), we updated the original Lego Mindstorms EV3 software blocks, in order to write in a log file on an SD card all the attempts carried out by each students' team involved in the research project, during the resolution of two given exercises ("Software updated into Lego Mindstorms EV3 blocks", in Figure 4). Sixteen schools were involved in the experimentation, and data were gathered in order to provide input to the clustering and prediction algorithms ("Data acquisition", in Figure 4).
Then, the dataset was divided into two subsets (schools with students younger than 12 years old and schools with students older than 12 years old): the latter were developed to transform the log files of these two subsets in two features' matrices ("Algorithmic problem solving for input data transformation", in Figure 4); which were then evaluated by different algorithms: both **supervised approach**, using Logistic Regression, Support Vector Machine (SVM), K-nearest neighbors (KNN), Random Forest, and a **mixed approach**, using both unsupervised K-means clustering, and the previously cited supervised algorithms. This strategy aimed to verify and highlight different patterns in the students' behaviours, and if these patterns are related to the students' performances ("Data clustering and performance prediction", in Figure 4); a model validation was performed to calculate the reliability of the system [using the repeated 10 fold cross validation (Kim, 2009), "Clusters' model validation" in Figure 4], and a detailed analysis of the clusters in relation to the results obtained by the students was implemented to verify the pedagogical meaning of the project results ["Correlation results (Cluster analysis vs. Student performance)", in Figure 4].
Similar problem-solving behaviours between the two groups (younger students and older students) have emerged during the analysis: for this reason, the dataset was aggregated again, and the procedure was repeated (from "Data clustering and performance prediction" to "Correlation results"), to verify if the same results were achieved considering the whole dataset. Thanks to this larger dataset, it was possible to use another machine learning technique: the performance of a Multilayer Perceptron (MLP) neural network was compared with the previously cited techniques (Logistic Regression, SVM, KNN, Random Forest).
Figure 4 shows also the teachers'/educators' workflow during our experimentation: at the beginning of the research project ("Software and firmware installation", see Figure 4)

teacher had to install updated software and firmware in the school devices (laptops, robotic kits); then, he/she could start the ER activity with students ("ER activity with students", see Figure 4), at the end of which it was necessary to download log files and upload them to a cloud storage ("Download log files from EV3 brick and upload them to a cloud storage", see Figure 4); when the ML analysis of the collected data was completed, the teacher gave his/her support for the analysis of the results, trying to compare the findings obtained by the ML algorithms with his/her knowledge about students' behaviours during standard educational activities ("Clustering results validation from a pedagogical point of view", see Figure 4). Finally, teachers tried to identify some strategies to help students with worst ER performance results, reflecting also on the problem-solving style showed by them.

As stated before, the students' workflow in an ER activity could be represented with the TMI cycle. Students, after having observed their robot's behaviour, discuss in the peer group (or with the expert) about how to improve their artefact and modifies their software to obtain the desired performance, and this procedure usually is repeated many times (Figure 5 shows a diagram of this workflow). This students' cyclical activity of modifying the programming sequence after having observed the robot's feedback, tracked by our system, is the input to the following ER Data Analytics.

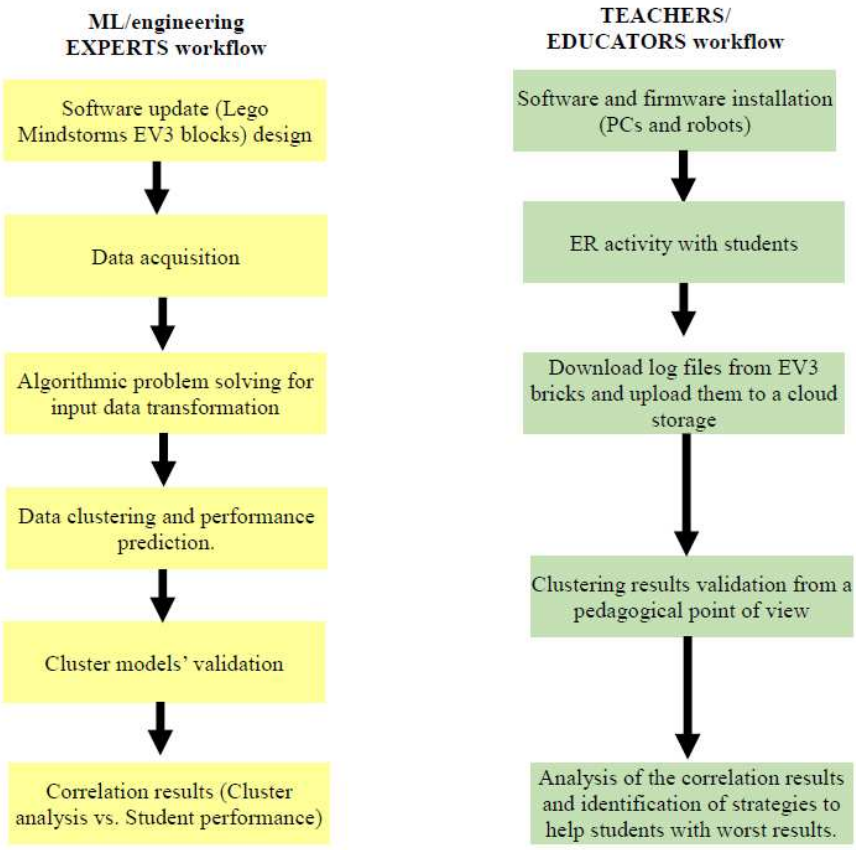

Figure 4. ML expert and teacher workflow in this experimentation

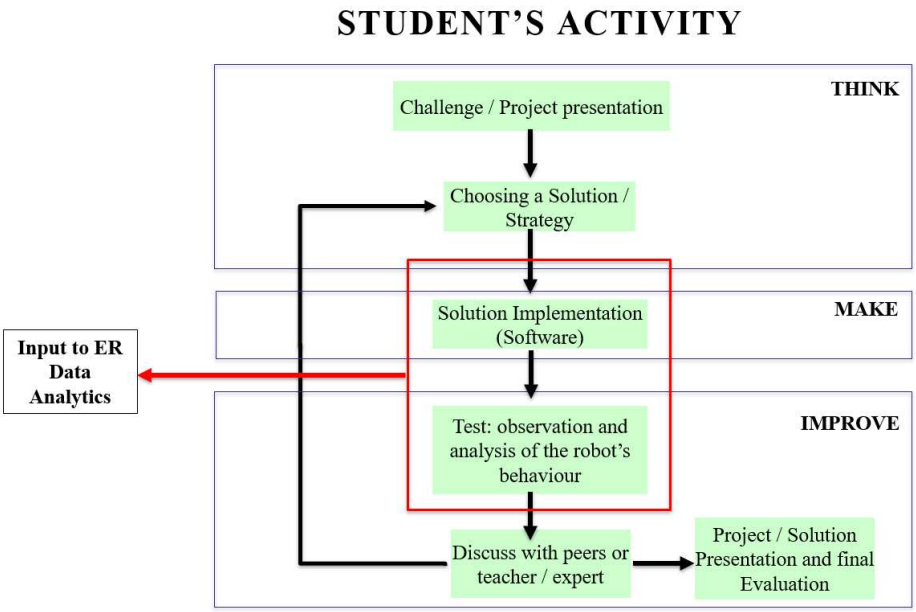

Figure 5. Student's activities workflow during an ER intervention

For the research purposes of the work described in this dissertation, the original blocks were modified based on the software suite Lego Mindstorms EV3 Developer kit (2019) with the objective to keep track of all the programming sequences designed by the students' team involved in Educational Robotics labs. Figure 6 denotes the development of this software update:

- during the students' team interaction with Lego Mindstorms software, the Lego block input data are defined (students create the programming sequences and define this parameters);
- the EV3 brick executes a programming block and simultaneously performs a Log Writing routine (it doesn't only perform the standard function of that block, for example turning on/off the motors, reading a value from a sensor, executing a loop or a conditional statement, etc.). It writes on a log file a string containing the principal information of the block (block type, options and parameters set by the students' team).
- the standard LEGO function, at the end of the process generates a double output: the standard behaviour of the robot and a log file with all the information about the executed blocks.

We substituted the original version of Lego blocks with the one shown in Figure 6 (using the same name and VIX extension of the original blocks) to obtain this new behaviour.

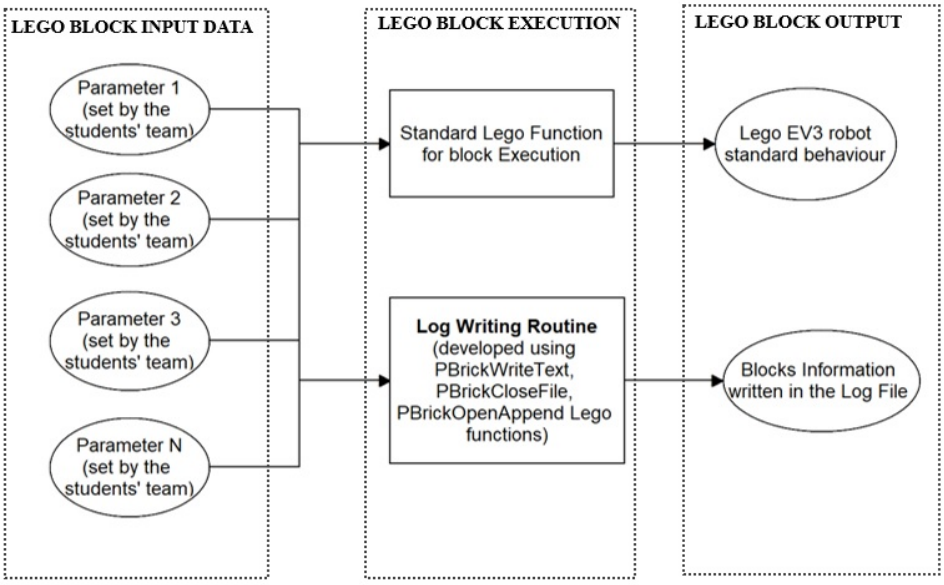

Figure 6. Structure of the Lego Mindstorms EV3 blocks software update

2.2 *Participants and procedure*

Stakeholders involved in the research project were students from sixteen Italian Primary and Secondary schools, located in the Emilia Romagna, Lazio and Marche regions. The total number of students involved in this study is 455. The experimentation was carried out from March 2018 to October 2019. We used a convenience sampling method to involve students in the experimentation: we collected data thanks to the Robotics courses organized in schools by the startup TALENT (partner in the research project).
Before starting the ER course, students were asked to compile a questionnaire, comprising 3 types of questions:

- Demographic questions (gender, age);
- School-related questions (the category of their favourite subject: TS stands for Technological/Scientific, AH stands for Artistic/Humanistic);
- Technology-related questions (which technologies they use at school or at home, if they use videogames).

Personal data, School-related data and Technology-related data are presented in Table 2 and Table 3.
Figure 7 represents the histogram of students' usage of technology at school or at home by category.

Table 2. Summary of data describing the sample of students collected by a questionnaire (schools 1-8).

	School 1	School 2	School 3	School 4	School 5	School 6	School 7	School 8
Students (#)	62	22	24	21	19	25	24	23
Teams (#)	20	6	6	5	5	8	6	6
Gender (%)								
M	61,29	63,64	37,50	38,10	31,58	72,00	54,17	39,13
F	38,71	36,36	62,50	61,90	68,42	28,00	45,83	60,87
Age (years)								
Mean (SD)	17,29 (0,55)	11,45 (0,50)	10,08 (0,65)	11,7 (0,47)	11,63 (0,83)	15,92 (0,28)	12,00 (0,46)	11,87 (1,29)
Min	17	11	9	11	10	15	11	10
Max	20	12	11	12	13	16	13	14
Discipline (%)								

TS	74,19	81,82	75,00	66,67	73,68	76,00	41,67	65,22
AH	24,19	18,18	12,50	33,33	26,32	8,00	50,00	30,43
Both	1,61	0,00	4,17	0,00	0,00	0,00	8,33	0,00
No answer	0,00	0,00	8,33	0,00	0,00	16,00	0,00	4,35
Technologies (%)								
Tablet	96,77	86,36	75,00	95,24	89,47	100,00	100,00	91,30
Smartphone	98,39	72,73	50,00	80,95	68,42	100,00	91,67	86,96
Text/Image Editor	95,16	72,73	25,00	57,14	84,21	88,00	83,33	91,30
Audio/video Editor	43,55	45,45	20,83	14,29	47,37	52,00	45,83	30,43
Web Browser	98,39	95,45	58,33	71,43	89,47	96,00	87,50	82,61

Arduino board	4,84	0,00	4,17	9,52	0,00	16,00	8,33	47,83
Programming env.	32,26	18,18	16,67	61,90	52,63	44,00	20,83	73,91
Ed. software	88,71	31,82	20,83	47,62	47,37	56,00	58,33	69,57
Robot	1,61	22,73	29,17	52,38	31,58	0,00	29,17	73,91
Other	3,23	0,00	20,83	19,05	5,26	4,00	14,67	4,53
Video Games (%)								
Yes	66,13	77,27	83,33	90,48	78,95	72,00	58,33	73,91
No	33,87	22,73	16,67	9,52	21,05	28,00	41,67	26,09

Table 3. Summary of data describing the sample of students collected by a questionnaire (schools 9-16).

	School 9	School 10	School 11	School 12	School 13	School 14	School 15	School 16
Students (#)	25	28	23	25	26	68	21	19
Teams (#)	6	6	6	8	7	17	5	5
Gender (%)								
M	60,00	71,43	60,87	40,00	42,31	55,88	71,43	63,16
F	40,00	28,57	39,13	60,00	57,69	44,12	28,57	36,84
Age (years)								
Mean (SD)	12,54 (0,51)	9,64 (0,56)	12,43 (0,95)	11,08 (0,70)	10,27 (0,78)	9,09 (0,69)	11,14 (1,11)	10,21 (0,98)
Min	12	9	11	10	9	8	9	8
Max	13	11	14	12	12	10	13	11

Discipline (%)								
TS	56,00	82,14	43,48	76,00	61,54	30,88	95,24	78,95
AH	44,00	0,00	43,48	20,00	30,77	33,82	4,76	10,53
Both	0,00	17,86	13,04	4,00	7,69	33,82	0,00	10,53
No answer	0,00	0,00	0,00	0,00	0,00	1,47	0,00	0,00
Technologies (%)								
Tablet	88,00	96,43	91,30	96,00	88,46	94,12	90,48	78,95
Smartphone	84,00	78,57	91,30	100,00	76,92	82,35	80,95	57,89
Text/Image Editor	84,00	35,71	82,61	72,00	50,00	45,59	80,95	47,37
Audio/video Editor	28,00	42,86	21,74	48,00	46,15	17,65	38,10	47,37

Web Browser	88,00	67,86	82,61	84,00	65,38	61,76	90,48	57,89
Arduino board	8,00	14,29	0,00	16,00	11,54	4,41	19,05	0,00
Programming env.	14,00	42,86	8,70	32,00	46,15	48,53	71,43	47,37
Ed. software	52,00	3,57	43,48	64,00	46,15	44,12	33,33	21,05
Robot	16,00	28,57	0,00	40,00	42,31	98,53	57,14	63,16
Other	16,00	7,14	4,35	16,00	3,85	2,94	4,76	5,26
Video Games (%)								
Yes	80,00	82,14	86,96	80,00	76,92	85,29	85,71	78,95
No	20,00	17,86	14,04	20,00	23,08	14,29	14,29	21,05

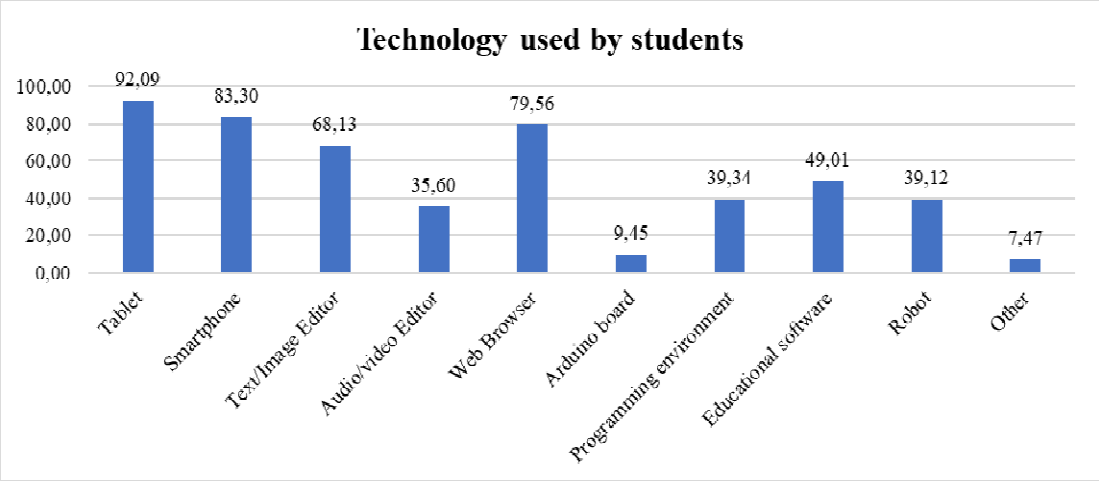

Figure 7. The columns report the percentage of how many students use a specific technology at school or at home

As Table 2 and Table 3 show, we have 11 schools with students younger than 12 years old (School 2, School 3, School 4, School 5, School 8, School 10, School 12, School 13, School 14, School 15 and School 16) and 5 schools with students older than 12 years old (School 1, School 6, School 7, School 9 and School 11). The total number of teams in the younger students' sub-dataset is 76 and the total number of teams in the older students' sub-dataset is 45.

In a typical Robotics challenge proposed by the educators involved in the experimentation, students are divided into teams of 3 to 4 students who worked together to design software or hardware solutions, trying to solve a challenge in a specific amount of time. In our experimentation the teacher created the work groups based on his/her own educational experience, trying to create balanced teams (mixing weak students with more proficient students). Each students' group has at its disposal one Lego Mindstorms EV3 Education kit and one laptop.

At first the educator explains in depth the task that the robot has to complete and gives to the students some theoretical information about the programming blocks necessary to solve the challenge. After this explanation students start to work on the exercise, and usually they can test their programming sequences as many times as they want, using all the time available for the design of their solution. At the end of this designing and testing phase, students' teams have to present their work, showing if the robot works and can perform correctly the task to the educator.

In this final moment, the instructor evaluates the performance of the robot: analysing the robot's behaviour, he/she reports in a table the precision (in terms of centimetres, considering the type of exercises reported in the following sections) obtained by each students' team, and communicates this performance evaluation to the students.

At the end of a lesson, the educator downloads and saves the log files generated by the system from each EV3 robot. With the software development presented in this dissertation, it's possible to visualise all the tests made by the students' teams during their designing phase, because each sequence created by the students and performed by the robot is tracked in the log files. The following sections show how the log files could be transformed into the problem-solving trajectory of each students' team involved in the lab. The standard approach usually adopted by teachers is the **evaluation of the final product** of the learning process (it could be a project report, the resolution of a problem, a presentation etc.) but as some researchers proposed (Blikstein et al., 2014; Resnick, 2017), new technologies allow to implement the analysis and **evaluation of the process** that student carried out to obtain the product.
Within the research study presented in this thesis, we will cluster the instructors' evaluation of the robot behaviour in two classes, **completed** and **not completed**; we will define a threshold in terms of robot's precision for each exercise: if the error obtained by the group was < threshold (in centimetres) the challenge is considered completed for that pupils' team, otherwise it is considered not completed.

Students involved in the research project attended a course that was an introduction to Robotics; the main topics of the course were:

- Introduction to Robotics. How to build a simple mobile robot and what are its main hardware features.
- How to turn on robot motors. What is the open-loop control.
- Introduction to the control flow statements: what is a "loop" and how to use it.
- How to collect data from sensors and how to use them trying to make the robot smarter.
- Introduction to the control flow statements: what is an "if-then-else" and how to use it.
- What is a programming bug, and how to implement the debugging activity.
- Final Challenge.

Each topic was characterized by some exercises and challenges: the educators proposed them to verify the students' understanding of the above Robotics concepts. In this thesis two of these exercises will be analysed in order to validate our approach.
During the course students designed different solutions for the challenges proposed by their instructors using two tools:

- Lego Mindstorms EV3 Education kit (2019)
- Lego Mindstorms EV3 Education software (2019): this programming environment is characterized by a visual approach: students can create sequence of instructions selecting blocks from the Blocks List Area (see Figure 8) and placing them in a certain order in the Programming Area. Blocks are divided into 6 main categories, identifiable by 6 colours: Actions (green), Flow Control (orange), Sensors (yellow), Data Operations (red), Advanced (dark blue), My Blocks (light blue).

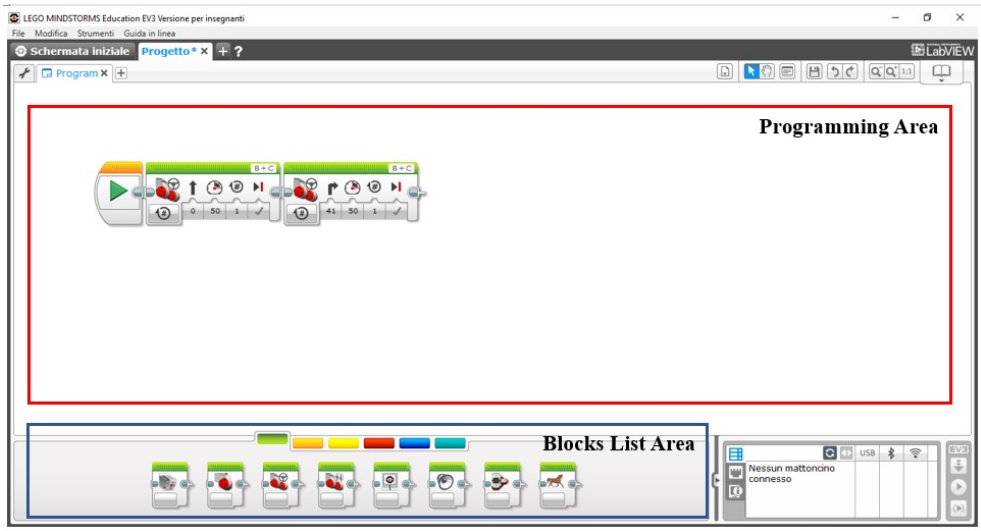

Figure 8. Lego Mindstorms Education EV3 Education screen

2.2.1 Exercise A: the robot has to cover a given distance

Lego Mindstorms EV3 Education software offers different ways to turn on robot's motors: for example, choosing the "Move Steering" block, you can select one of the 4 different options in the following figure (On for Rotations, On for Degrees, On for Seconds, On).
After selecting the "On for Rotations" option, students' team can modify the numeric parameter highlighted in Figure 9, to specify how many rotations the motors (and the wheels connected to the motors) have to perform.

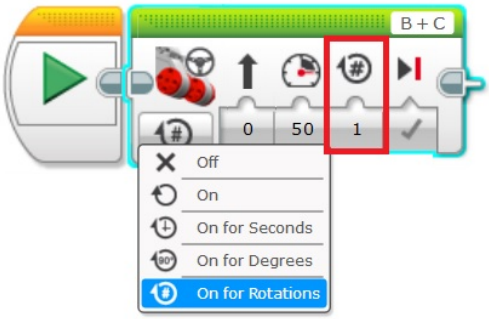

Figure 9. The "Move Steering" Block

With the same procedure students can decide to define the number of degrees or the number of seconds of the motors' rotation. The first two numeric parameters represent the steering of the robot and the motors' speed. There isn't a direct way to specify a distance that the

robot as to perform: for this reason, the first exercise that educators proposed to students was to:

Program the robot so that it covers a given distance (1 m), trying to be as precise as possible.

Trying to solve this exercise, students' teams involved in the research project had to take into account some constraints:

- the amount of time within they had to design and test their solution (15 minutes for higher secondary school classes, 20 minutes for lower secondary school classes);
- during the available time the teams could test the programming sequence as many times as they wanted;
- they were not allowed to use measuring instruments (set squares, rulers, etc.) to verify the distance covered by the robot on the floor; they could utilize the instruments only to measure some robot's parameters (for example the radius of the wheel).

This exercise wanted to stress the problem-solving skills of the students: some of them discovered that there were some cables with a known length inside the Lego Mindstorms box, and they were allowed to use them as a reference object for their programming tests.
Students' teams approaches were very different: some pupils measured the radius of the wheel and calculated its circumference; other students tried to determine the robot speed, with the objective to quantify the number of seconds to set in the programming block; someone else adopted a more practical and "trial and error approach", for example using the cables inside the box or estimating the dimensions of the floor tiles as a reference measurement.
In this exercise the instructor gave a final evaluation related to the precision obtained by the team: if the error was <= 4 cm, the educator considered the challenge completed; if the error was > 4 cm the educator considered the challenge not completed.

2.2.2 Exercise B: the robot has to stop at a given distance from the wall, using the ultrasonic sensor

In the second exercise students had to use the ultrasonic sensor and the conditional block, with the aim of creating this behaviour:

Program the robot so that it stops at a given distance from the wall (25 cm), trying to be as precise as possible.

For this exercise, students' teams had to consider:

- the amount of time within they had to design and test their solution (20 minutes for higher secondary school classes, 30 minutes for lower secondary school classes);
- during the available time the teams could test the programming sequence as many times as they wanted.

Solving this problem is usually quite complex for students, because there are some variables to analyse:

- how to set the condition related to the value measured by the ultrasonic sensor;
- how to use the loop block: in this case robot had to repeat continuously the measure using the ultrasonic sensor;
- how to compensate the braking distance: when the EV3 brick turn off the motors, the robot doesn't stop immediately.

In this exercise the instructor provided the final evaluation related to the precision of the challenge obtained by the team: if the error was <= 3 cm, the educator considered the challenge completed; if the error was > 3 cm the educator considered the challenge not completed at all.

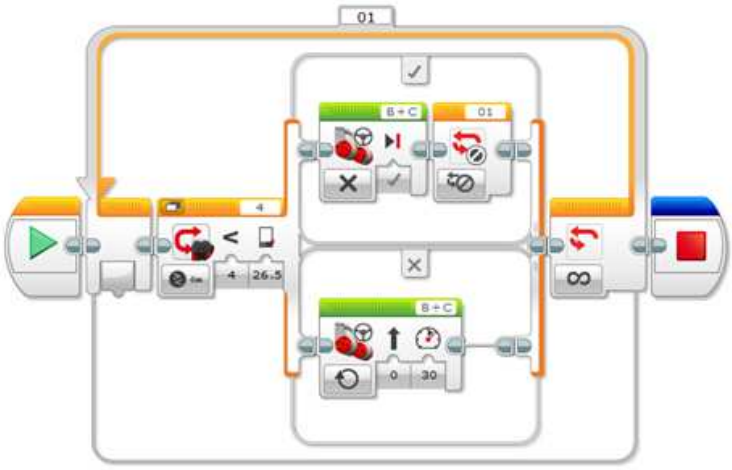

Figure 10. A possible solution for the Ex. B

2.3 *Data preparation*

Students' teams designed 2187 programming sequences to solve the Exercise A (1404 designed by younger students and 783 designed by older students), and 4252 programming sequences to solve the Exercise B (3087 designed by younger students and 1165 designed by older students). In the Appendix A are reported all the technical steps performed to obtain a mathematical representation from the log files collected during the experimentation. This mathematical representation of the log files allows to calculate 13 indicators for each programming sequence:

- **Motors**: the n° of Motor blocks in the sequence; to calculate this indicator, a Python function counts how many blocks of Motor category are present in the sequence.

- **Loops**: the n° of Loop blocks in the sequence.
- **Conditionals**: the n° of Conditional and Sensors blocks in the sequence.
- **Others**: the n° of blocks in the sequence belonging to different categories than Motors, Loops and Conditionals.
- **High Values**: the n° of Motors blocks in the sequence with a Rotations parameter >= 20, or with a Seconds parameter >= 15.
- **Added**: the n° of blocks added, compared to the previous sequence; to calculate this indicator, a Python function counts how many blocks with different Block Name or different Block Option have been added considering two contiguous sequences.
- **Deleted**: the n° of blocks deleted, compared to the previous sequence; to calculate this indicator, a Python function counts how many blocks in a specific sequence have been deleted in the next sequence (only if they are not classified as "Changed").
- **Changed**: the n° of blocks changed, compared to the previous sequence; to calculate this indicator, a Python function counts how many blocks with same Block Name and same Block Option but different parameters are present comparing two contiguous sequences.
- **Equal**: the n° of the same blocks, compared to the previous sequence; to calculate this indicator, a Python function counts how many blocks with same Block Name, same Block Option and exactly the same parameters are present comparing two contiguous sequences.
- **Delta Motors**: amount of change in Motor blocks parameters (first, second or third parameter), compared to the previous sequence (calculated only for blocks of the "Changed" category); to calculate this indicator, a Python function performs the pseudocode as shown in Figure 11 (and the flowchart in Figure 12). Then, for each block in the current sequence only the lowest value of the Euclidean distance is selected (supposing that this difference represents the modification made by the students' team) and all these selected values are summed to obtain the overall Delta Motors.
- **Delta Loops**: amount of change in Loop blocks parameters, compared to the previous sequence; the procedure is the same as the one explained for the Delta Motors, but considering the Loop category.
- **Delta Conditionals**: amount of change in Conditional blocks parameters, compared to the previous sequence; the procedure is the same as the one explained for the Delta Motors, but considering the Conditional category.
- **Delta Others**: amount of change in Other blocks parameters, compared to the previous sequence; the procedure is the same as the one explained for the Delta Motors, but considering the Other categories.

Each programming test realised by the students' team can be represented as a vector composed by these thirteen elements.
The previously presented thirteen indicators were employed to represent the participants' activities in the robot programming activity; for each test contained in the log file this vector is calculated, and at the end of this procedure a sort of trajectory of the problem-

solving strategy implemented by the students' team is obtained. As proposed by some previous researches (Berland et al., 2013; Chao, 2016) the representation of a point in the students' designing trajectory could be done choosing the block type that characterise the programming environment. Lego Mindstorms EV3 blocks mostly used by students in this experimentation were Motors, Loops and Conditionals (that includes also Sensors blocks), but usually, in an ER activity, students change their programming sequence observing the robot's feedback. Thus, it's important to consider also the difference (delta) between two sequences, not only the number of blocks in each relevant category. Motors, Loops, Conditionals and Sensors can be considered also some key concepts in ER and Computational curriculums (Grover, & Pea, 2013; Scaradozzi, Sorbi, Pedale, Valzano, & Vergine, 2015; Scaradozzi, Cesaretti, Screpanti, Costa, Zingaretti, & Valzano, 2019; Allsop, 2019).

Figure 13 (Microsoft MakeCode for Lego Mindstorms EV3), Figure 14 (Scratch extension for Lego Mindstorms EV3) and Figure 15 (mBlock for mBot robot) show other software interfaces that allow to design programming sequences for educational robots: all of them are characterised by blocks divided into Categories, and each block has parameters in input; they seem quite similar to Lego Mindstorms EV3 Education software, and probably a strategy similar to the one presented in this dissertation (tracking programming sequences and log transformation into vectors) could be implemented also for these software environments.

```
#The algorithm iterates over all the blocks in sequence 1
for i in range(0, length_seq1):
    #The algorithm iterates over all the blocks in sequence 2
    for y in range(0,length_seq2):
        #If the 1st block feature and the 2nd block feature (of the i-th block(sequence 1) and
        of the y-th block (sequence 2)) are equal
        if block_seq1[i][1] == block_seq2[y][1] and block_seq1[i][2] == block_seq2[y][2]:
        #The Euclidean distance is calculated considering the 1st, 2nd and 3rd block parameter
            delta[i][y] = euclidean_distance(block_seq1[i][1:3] - block_seq2[y][1:3])
```

Figure 11. Pseudocode for the delta calculation

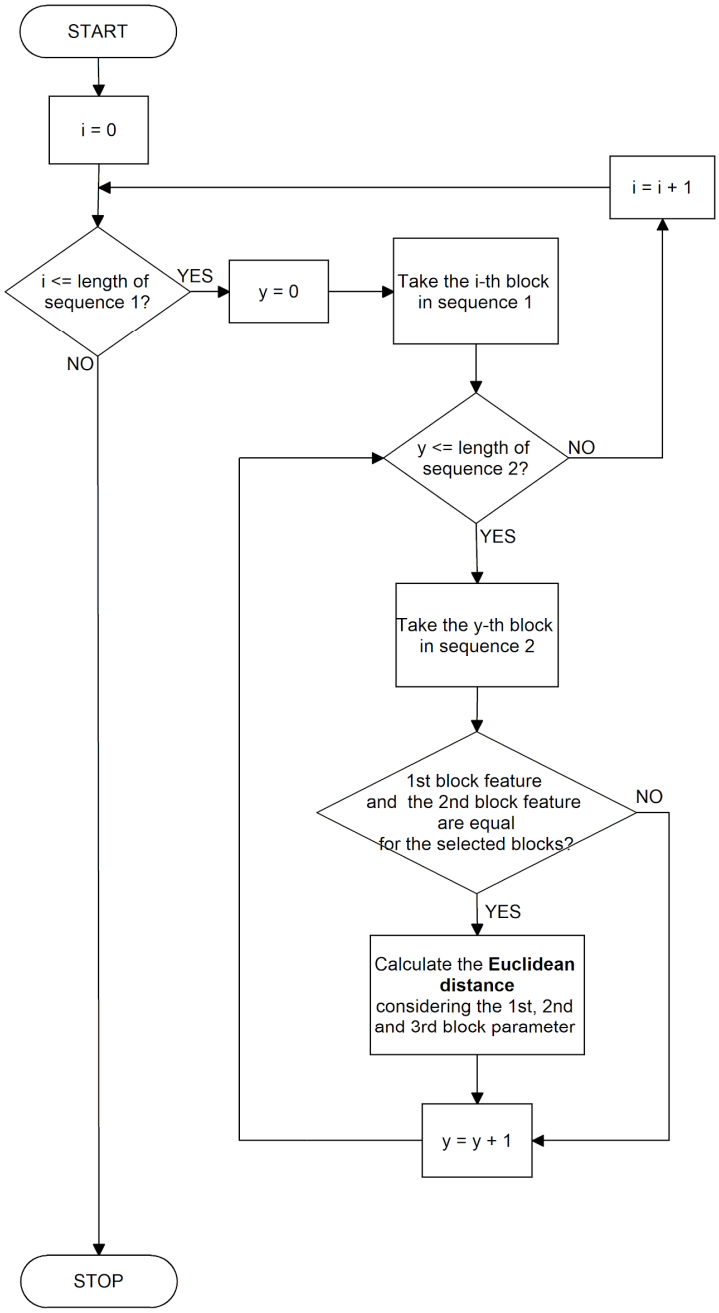

Figure 12. Flowchart for the delta calculation

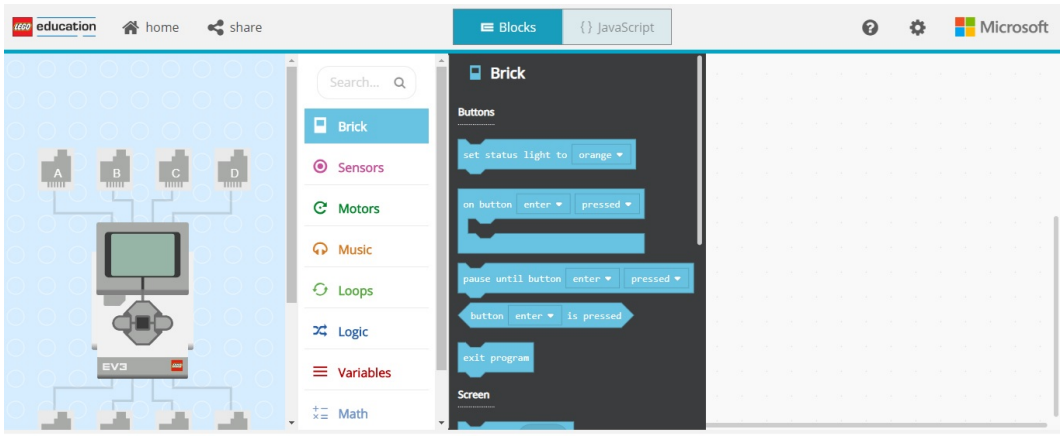

Figure 13. Microsoft MakeCode for Lego Mindstorms EV3 software interface

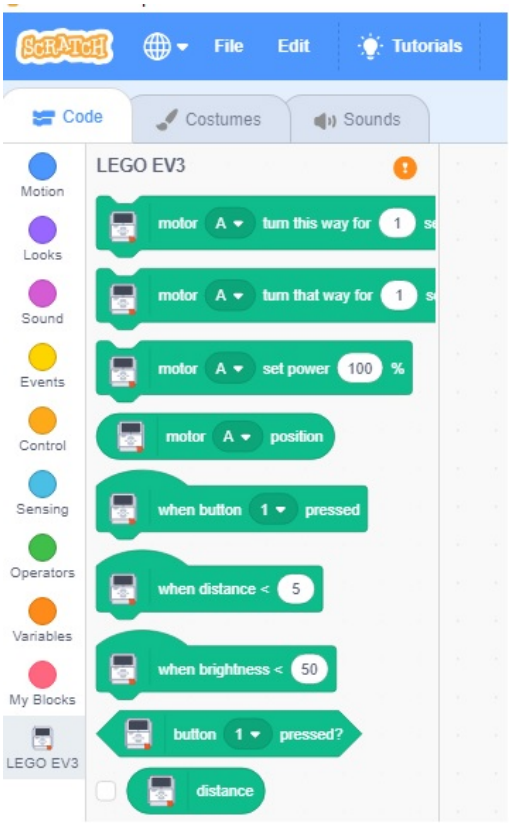

Figure 14. Lego Mindstorms EV3 Scratch extension

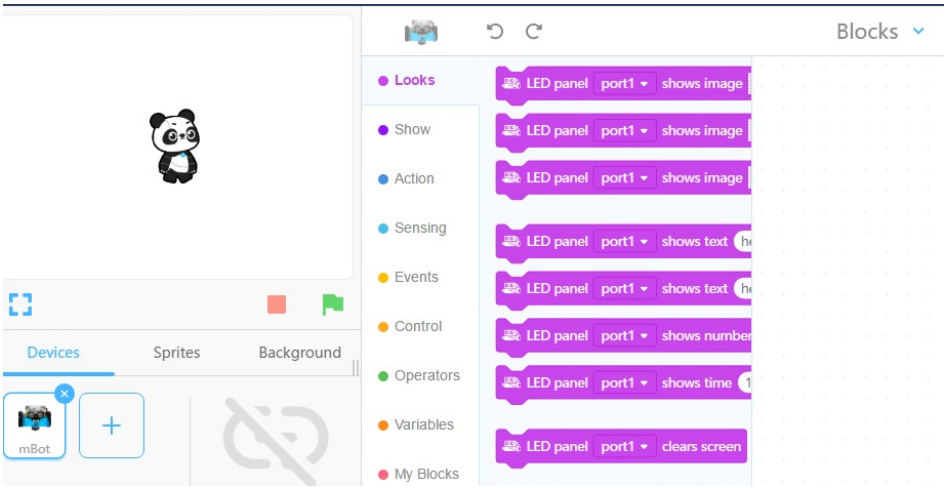

Figure 15. mBlock (mBot robot) software interface

2.4 *Machine learning algorithms*

We adopted two approaches in the analysis of the exercise logs:

- A **supervised approach**, calculating the mean value and the standard deviation for each indicator presented in the previous section, taking into account all the trials performed by a students' team to solve an exercise: the problem-solving trajectory is summarized by the mean number and the standard deviation of Motors blocks in all the programming sequences designed to reach the objective; the mean number and the standard deviation of the Loops blocks; the mean number and the standard deviation of the Conditionals blocks, etc.; this part of the of the feature matrix summarizes the **team's past problem-solving activity** and it is unvaried both for Exercise A and B. If we want to add also an **insight of the learners' current activity**, we can integrate to the feature matrix other indicators, chosen based on the exercise type: the final performance in Exercise A is closely linked to the parameters that students can set for the Move Steering block (see Figure 9); for this reason, we decided to add to the feature matrix the "Rotations parameter" (the amount of movement in rotations, defined by students in the Move Steering block), the "Seconds parameter" (the amount of movement in seconds, defined by students in the Move Steering block) and the "Motors Speed" parameter (the Motor speed level, defined by students in the Move Steering block). The final performance in Exercise B is closely linked to the parameters that students can set for the Move Steering block, for the Conditional block and to the presence of the Motors OFF block (see Figure 10); for this reason, we decided to add to the feature matrix the "Compare Type" parameter (the symbol =, <, >, <=, >=, ≠ defined by students in the Conditional block), the "Ultrasonic sensor threshold" parameter (the value to compare sensor data to, defined by students in the

Conditional block) the "Motors Speed" parameter (the Motor speed level, defined by students in the Move Steering block) and the "Motor OFF presence" parameter (1 if the Motor OFF block is present, 0 otherwise).
Then, we compared the performances of four different machine learning algorithms (Logistic Regression (Hosmer & Lemeshow, 2000), Support Vector Machine (Cristianini and Shawe-Taylor, 2000), k-nearest neighbors (Cover and Hart, 1967), and Random Forest classifier (Ho, 1995)), in the prediction of the students' teams final result using as input the feature matrix described in the previous paragraph.

If we want to represent the students' team problem-solving activity at the n-th trial (t_n) applying the supervised approach, we can adopt the following schemes (Figure 16 and Figure 17), for Exercise A and B:

SUPERVISED APPROACH – EX. A

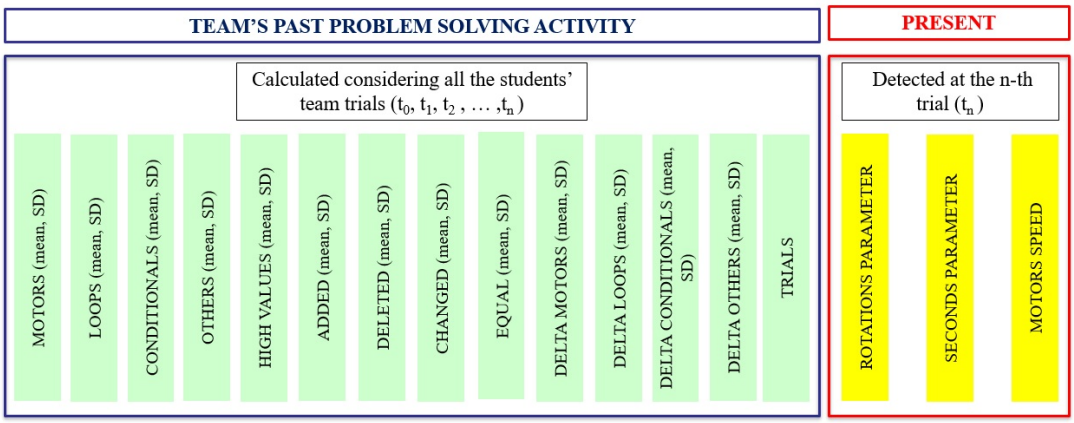

Figure 16. Team problem-solving activity representation (at the n-th trial) for the Ex. A passed as input to the machine learning algorithms (supervised approach)

SUPERVISED APPROACH – EX. B

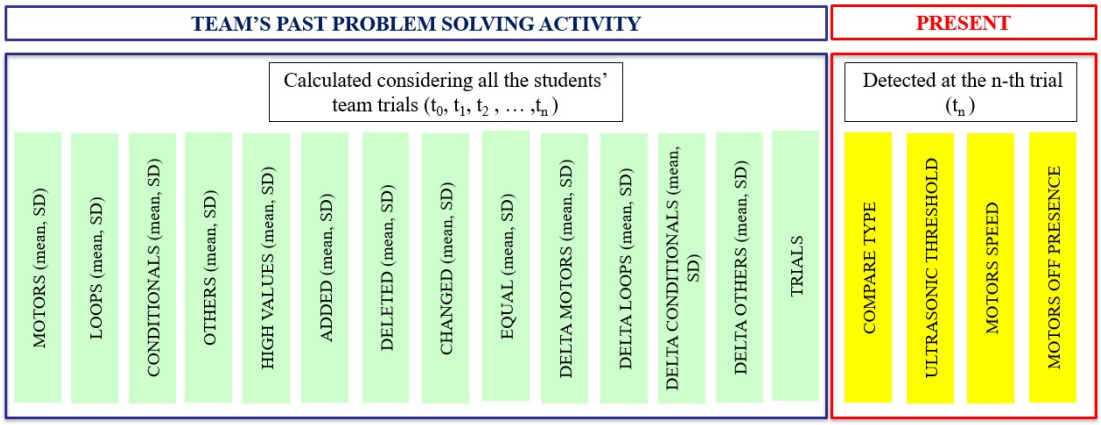

Figure 17. Team problem-solving activity representation (at the n-th trial) for the Ex. B passed as input to the machine learning algorithms (supervised approach)

- A **mixed approach**, combining the benefits of both supervised and unsupervised methods: a k-means algorithm (Steinley, 2006) was applied to calculate in which clusters the programming sequences could be divided. Following the clustering, the percentage of each cluster in the programming activity of the students' groups was calculated: our system counted how many sequences belonging to each cluster are in the students' team log file generated during the exercise, and divided this value by the total number of programming sequences designed by the team.
  These new features were then used to create a feature matrix as an input for the previously cited four supervised algorithms. With this approach we have a number of features (representing the team's past problem-solving activity) equal to the number of clusters identified by the k-means algorithm. Also in this case we added to the model an insight of the learners' current activity, integrating Rotations parameter, Seconds parameter and Motors speed (detected at the n-th programming trial) for Exercise A, and Compare Type, Ultrasonic threshold, Motors speed and Motors Off presence (detected at the n-th programming trial) for Exercise B.

If we want to represent the students' team problem-solving activity at the n-th trial (t_n) applying the mixed approach, we can adopt the following schemes (Figure 16 and Figure 17), for Exercise A and B:

MIXED APPROACH – EX. A

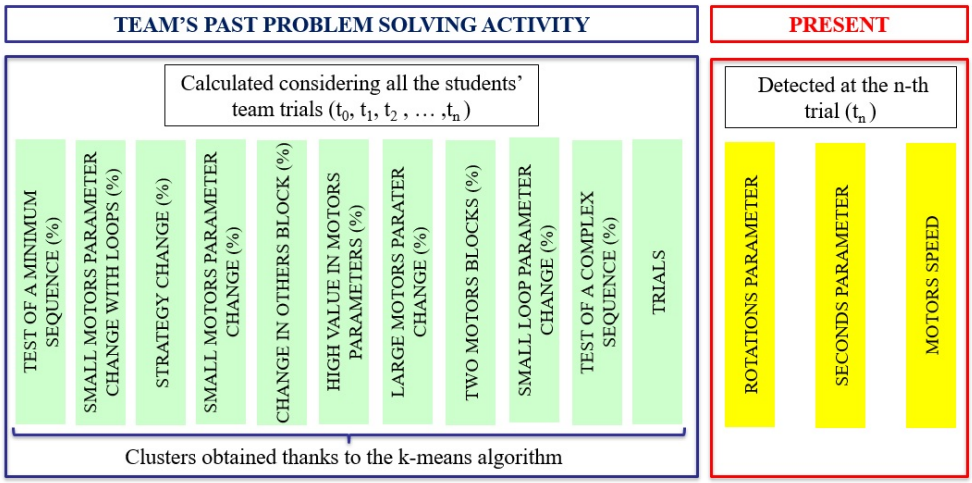

Figure 18. Team problem-solving activity representation (at the n-th trial) for the Ex. A passed as input to the machine learning algorithms (mixed approach)

MIXED APPROACH – EX. B

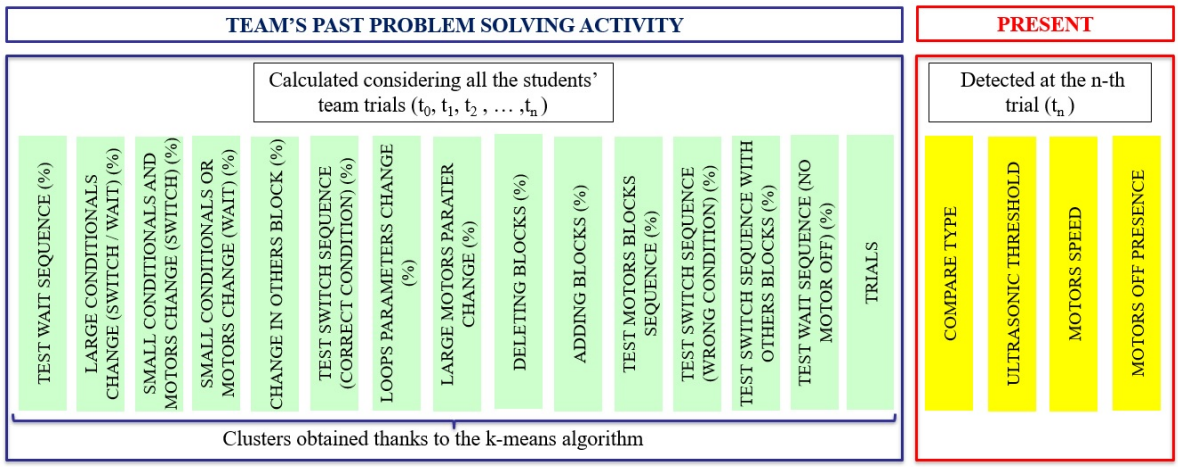

Figure 19. Team problem-solving activity representation (at the n-th trial) for the Ex. B passed as input to the machine learning algorithms (mixed approach)

We applied these machine learning techniques following this strategy: at first, we wanted to verify if similar behaviours come up comparing younger students and older students, so we divided the entire dataset in two subsets (students younger than 12 years old and students older than 12 years old) and applied the supervised and mixed approach in these two subgroups.

Then, after demonstrating that similar problem-solving strategies are emerged both for younger learners and for older learners, we aggregated the dataset again, performed the supervised and the mixed approach and compared the performances of these two techniques considering the entire dataset.

To evaluate the performances of these different strategies in the prediction of the educational results four parameters were calculated (Huang and Ling, 2005):

- Accuracy (number of correct predictions / total number of predictions)
- Mean Precision (calculated considering the average value between precision in the prediction of students' positive performance and negative performance)
- Mean Recall (calculated considering the average value between the recall in the prediction of students' positive performance and negative performance)
- Mean F1 – Score (calculated considering the average value between the F1-score in the prediction of students' positive performance and negative performance)

To obtain these parameters a repeated 10-fold cross validation was performed (Kim, 2009), so that the average value and standard deviation of the previous four parameters repeating the 10-fold validation multiple times were calculated.

Considering a larger dataset, we decided to apply also a Multilayer Perceptron Neural Network (using the feature matrix generated for the mixed approach) in the prediction of the team's final performance. This last algorithm outperformed the previous ones, so we decided to test it also for a real prediction: being t_n the n-th programming sequence designed by the students' team (where n is the number of team's trials to solve the exercise), we deleted the last trial in the log file and considered only $n-1$ programming sequences to generate the feature matrix to pass as input for the neural network. We tried to consider trials from t_0 to $t_{(n-1)}$ to test if it's possible to predict the students' final result before they ended the Robotics exercise. Then, we tested the MLP neural performance in terms of prediction also for $n-2$ programming sequences (considering programming sequences from t_0 to $t_{(n-2)}$).

Technical details about the machine learning algorithms developed in this research project are presented in the Appendix B.

Chapter 3.

3 Results

Chapter 3 presents the results of the research project:

- at first, some data insights of the collected dataset are presented, analysing behaviours that characterise students' problem-solving activity, without applying machine learning techniques;
- then it proposes the results obtained from the input data in terms of prediction and the programming patterns discovered, considering younger and older students divided into two sub-datasets in order to demonstrate that similar behaviours emerged in both of them;
- after this separate study we aggregate the dataset again and perform the same analysis, verifying if different Machine Learning performances are obtained and how problem-solving styles are related to learners' achievements; thanks to a larger dataset we are allowed to apply Multilayer Perceptron Neural Network with the aim to predict students' final results.

3.1 *Data Insights*

In this section we present some data insights obtained thanks to our tracking system. These preliminary graphs are not obtained applying machine learning techniques, but they can help teachers to reflect about students' problem-solving activity. The following diagrams have been obtained simply selecting some programming parameters deeply connected to the challenges and visualizing them into a chart.

3.1.1 Exercise A

Regarding the Exercise A "program the robot so that it covers a given distance, trying to be as precise as possible", some students realised that there were some cables with a known length inside the Lego Mindstorms EV3 box, and they were allowed to use them as a reference object for the trials. As previously stated, this task was tricky because in the Lego Mindstorms EV3 software there are not blocks in which the designer can set a specific distance to cover. The trainer presented only one block for the challenge: the "Move Steering" function, where students can set three modes to control the motors ("On for seconds", "On for degrees" or "On for rotations"), the steering of the robot and the motors' speed.

Students' teams mainly focused on the change in the last block parameter: some groups calculated the wheel's circumference (in order to obtain the number of rotations to set); other groups tried to measure the robot's speed (in order to calculate the number of seconds to set); other groups adopted a more practical and "trial and error" approach, for example using the cable inside the box or the floor tiles as a reference measurement. These different approaches to the solution to a given task seem to fit into the two different styles in

problem solving proposed by Turkle and Papert (1992), the "bricoleur scientist" and the "planner scientist".

Figure 20 shows the sequence of the Rotations parameter (the number of the rotations set for the motors) chosen by one of the groups involved in the experimentation: 10 tests were conducted by the team (the last one was the final competition), all of them with a rotation parameter equal to 5.63 rotations. In this case planning seems to be the prevalent approach adopted by the group, probably with an initial mathematical calculus (the wheel circumference) and then verification tests to check the robot's behaviour. This team obtained a 1 cm error from the desired measure.

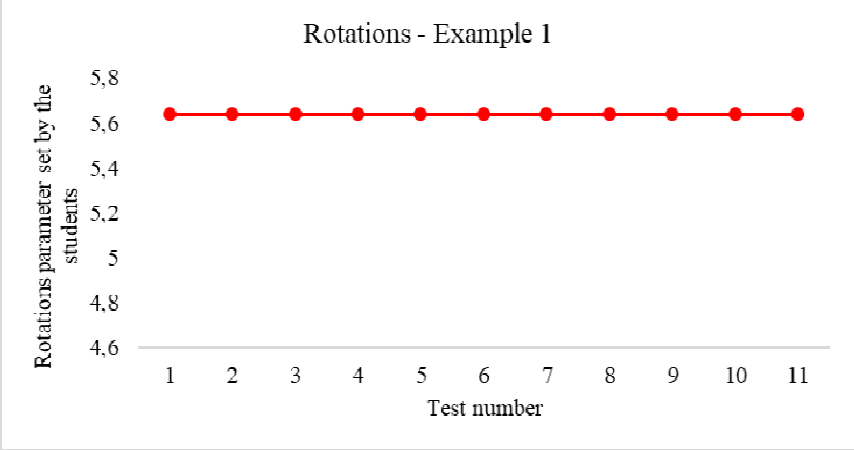

Figure 20. Rotations graph – Example 1

Figure 21 shows the Rotations parameter set by another group: they performed 14 tests (the last one was the final competition), and their strategy is represented by a broken line ranging from a minimum value of 1 rotation to a maximum value of 8. They probably did a first check of the robot's behaviour setting 1 rotation for the motors, then they refined the parameter step-by-step, gradually approaching the right value. In this case tinkering seems to be the prevalent approach adopted by the group, probably with a "trial and error" pathway more pronounced compared to the Example 1. This team obtained a 1.5 cm error from the desired measure.

Figure 21. Rotations graph – Example 2

Figure 22 shows another behaviour quite common in our dataset: the team performed a large number of tests (25), and their strategy is represented by a broken line ranging from a minimum value of 1 rotation to a maximum value of 80. In this case tinkering seems to be the prevalent approach adopted by the group, but they probably were struggling during the exercise resolution: they set multiple times the Rotations parameter to 1 (16 times), then they tried values very far from the right solution (40, 80, 30, etc.). This team obtained a 48 cm error from the desired measure.

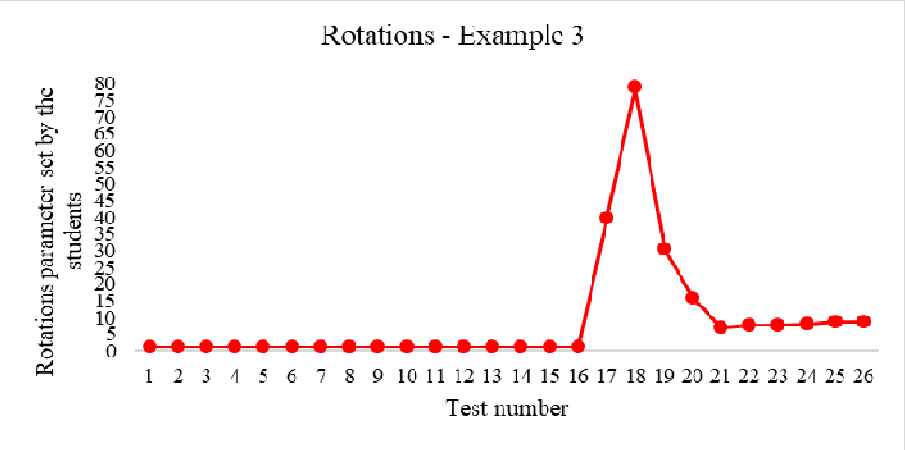

Figure 22. Rotations graph – Example 3

Figure 23 shows the sequence of the Seconds parameter selected by one of the teams: 13 tests were conducted by the group, and the students' strategy is represented by a broken line ranging from a minimum value of 2 seconds to a maximum value of 4. In this case, at the end of the activity this group reported to have tried a planning approach, trying to measure

the speed of the robot. But something went wrong, and this team obtained a 10 cm error from the desired measure.

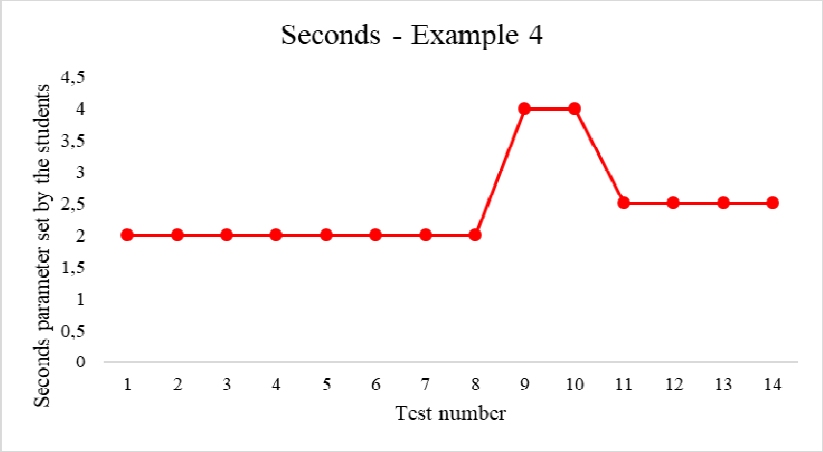

Figure 23. Seconds graph – Example 4

These graphs (Figure 20, Figure 21, Figure 22 and Figure 23) show different approaches to Exercise A: planning and tinkering can lead students to achieve both positive and negative results; so it could be useful to design systems that help teachers to recognise in real time when learners are struggling with a challenge.

3.1.2 Exercise B

Regarding the Exercise B "program the robot so that it stops at a given distance from the wall, trying to be as precise as possible", students had to use Motors, Loop and Conditional blocks. They didn't know how to set the condition related to the value measured by the ultrasonic sensor (it was their first use of an ultrasonic sensor): some of them started trying some possible values and observing the feedback of the robot; some of them started reflecting on the task and schematizing the problem, creating the programming sequence only after this phase.

Figure 24 shows the sequence of the Delta calculated between Conditional blocks considering contiguous sequences of one of the teams in the research project. They performed 17 tests (the last one was the final competition), and their strategy is represented by a broken line ranging from a minimum value of 0 (Delta in Conditional blocks) to a maximum value of 8. This group never changed the symbol in the Conditional block (this modification is highlighted with a Delta equal to 40 or 80, as showed by the following students' team, Figure 25): they started setting the right condition from the first test, and then they refined the Ultrasonic threshold step by step, obtaining a 1 cm error from the desired measure.

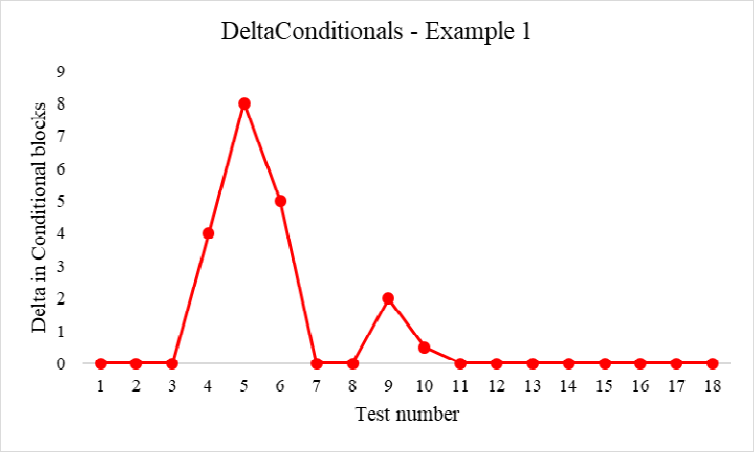

Figure 24. Delta Conditionals graph – Example 1

Figure 25 shows the problem-solving activity of another group: they performed 48 tests (the last one was the final competition), and their strategy is represented by a broken line ranging from a minimum value of 0 (Delta in Conditional blocks) to a maximum value of 80 (Delta equal to 40 or 80 represents a change in the conditional symbol (=, <, >, <=, >=, ≠)). They modified the Ultrasonic condition multiple times with large changes, modifying 5 times the symbol in the Conditional block, especially in the second part of their programming activity. This team obtained a 7 cm error from the desired measure.

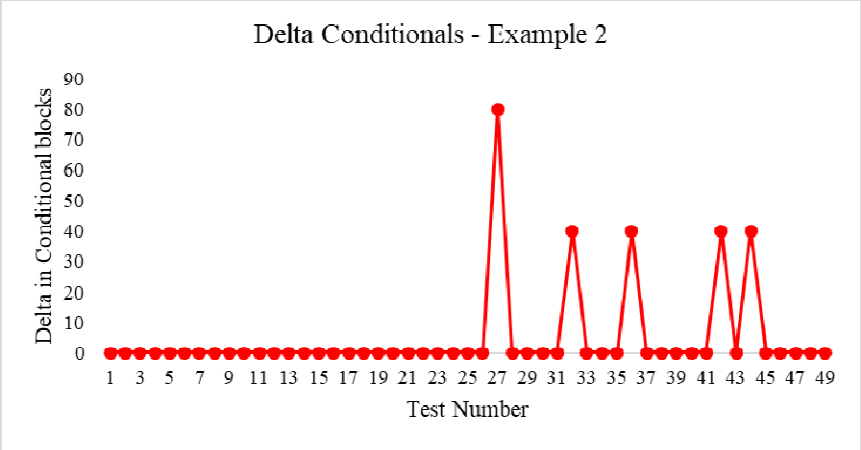

Figure 25. Delta Conditionals graph – Example 2

This preliminary analysis shows how such a tool can provide teachers with complementary information on students learning. The following sections present the results connected to the machine learning techniques in terms of prediction and problem-solving styles identification.

3.2 *Younger students*

3.2.1 Exercise A

Figure 26 and Figure 27 show the resulted performances in comparison within the four machine learning algorithms in the prediction of the younger students' teams result, analysing Exercise A. In this exercise the 32,3% of teams of this subset of participants had a negative performance (see Figure 28).

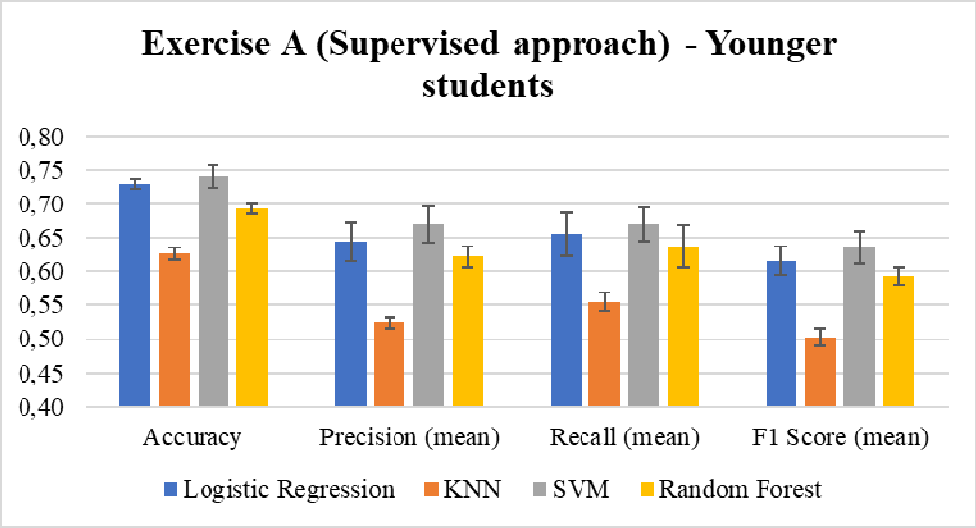

Figure 26. Results for the Ex. A (considering only younger students) obtained applying the supervised approach

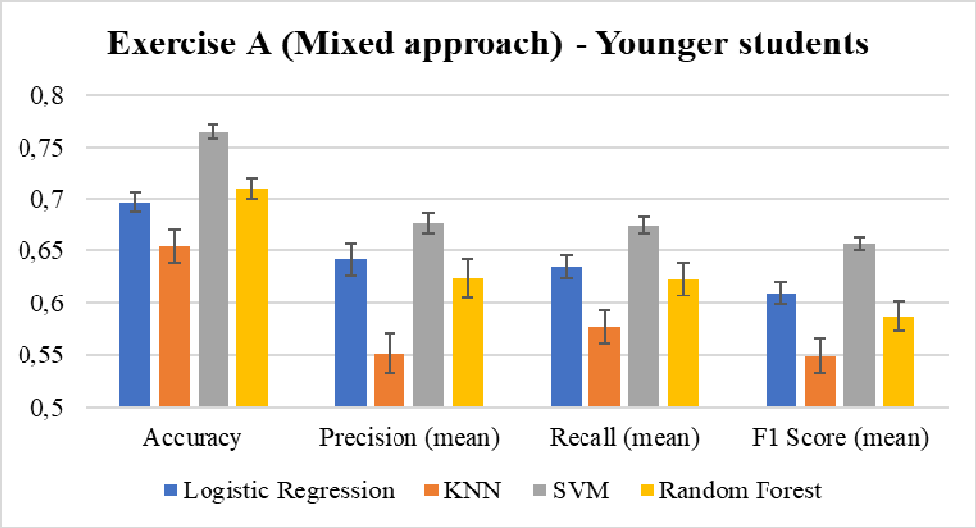

Figure 27. Results for the Ex. A (considering only younger students) obtained applying the mixed approach

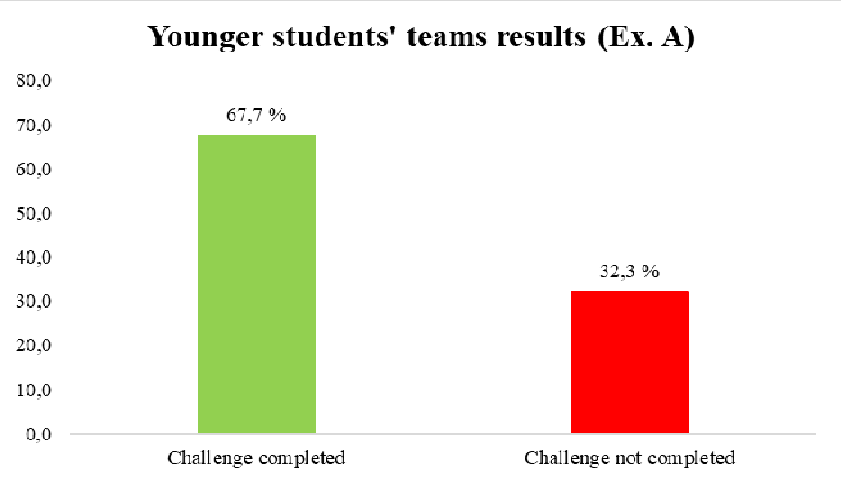

Figure 28. Younger students' teams results for the Ex. A

For both the approaches in the analysis of the exercises A, the best performance was obtained with the SVM algorithms, but comparing the supervised and the mixed approach a better trend emerges applying the second one as shown in Table 4.

Table 4. Best performance parameters for Exercise A (younger students).

	SVM (supervised) Ex. A - Younger	SVM (mixed) Ex. A - Younger
Accuracy	0,74	**0,76**
Mean Precision	0,67	**0,68**
Mean Recall	0,67	**0,67**
Mean F1 Score	0,64	**0,66**

Table 5 show the clusters resulting from the k-means algorithms; to select the number of clusters in which divide the programming sequences generated by the younger students (solving the Exercise A) the Elbow Method (see Figure 29) was applied (Kodinariya and Makwana, 2013). Table 6 contains the mean values and standard deviation values of the thirteen indicators (presented in the previous chapter) calculated for each cluster, for Exercise A (#: number of sequences). In Table 6 only nine indicators are considered, because younger students did not use Loops and Conditional blocks for the Exercise A.

Table 5. Clusters obtained with the k-means algorithm applied on Ex. A (considering only younger students).

	Name	#	%	Description
1	**SMALL MOTORS PARAMETERS CHANGE**	346	25,9	The team is refining its Motors parameters.
2	**HIGH VALUE IN MOTORS PARAMETERS**	142	10,1	The Motors blocks are characterized by high values of rotations or seconds parameters.
3	**TEST OF A MINIMUM SEQUENCE**	848	60,4	The team is testing the same previous programming sequence (with only Motors blocks).
4	**STRATEGY CHANGE**	22	1,6	The team is changing its strategy (1 block deleted, 1 added), probably modifying the Move Steering block options (Rotations, Degrees, Seconds).
5	**LARGE MOTORS PARAMETERS CHANGE**	22	1,6	The team is changing strongly its Motors parameters.
6	**CHANGE IN OTHERS BLOCK**	1	0,1	The team is changing something not connected to the challenge (Others blocks are not essential in Ex. A).

7	**TEST OF A COMPLEX SEQUENCE**	5	0,4	The team is testing the same programming sequence (with Others and Motors block).

Table 6. Mean values and standard deviation (M (SD)) calculated for the 13 indicators for each cluster in Table 5.

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaOthers	Motors	Others	HighValues
1	0,86 (0,37)	0,85 (0,35)	0,00 (0,00)	0,00 (0,00)	2,28 (4,58)	0,00 (0,00)	1,00 (0,00)	1,01 (0,10)	0,00 (0,05)
2	1,92 (0,50)	0,01 (0,12)	0,00 (0,00)	0,00 (0,00)	0,05 (0,40)	0,00 (0,00)	1,00 (0,00)	1,04 (0,20)	1,00 (0,00)
3	2,01 (0,10)	0,00 (0,00)	0,00 (0,03)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	1,01 (0,10)	0,00 (0,00)
4	1,14 (0,35)	0,00 (0,00)	1,00 (0,44)	0,95 (0,21)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	1,14 (0,47)	0,05 (0,21)
5	1,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	51,40 (18,04)	0,00 (0,00)	1,00 (0,00)	1,00 (0,00)	0,36 (0,50)
6	2,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	1,00 (0,00)	2,00 (0,00)	0,00 (0,00)
7	4,40 (0,55)	0,40 (0,55)	0,20 (0,45)	0,00 (0,00)	1,16 (1,59)	0,00 (0,00)	1,00 (0,00)	4,00 (0,00)	0,00 (0,00)

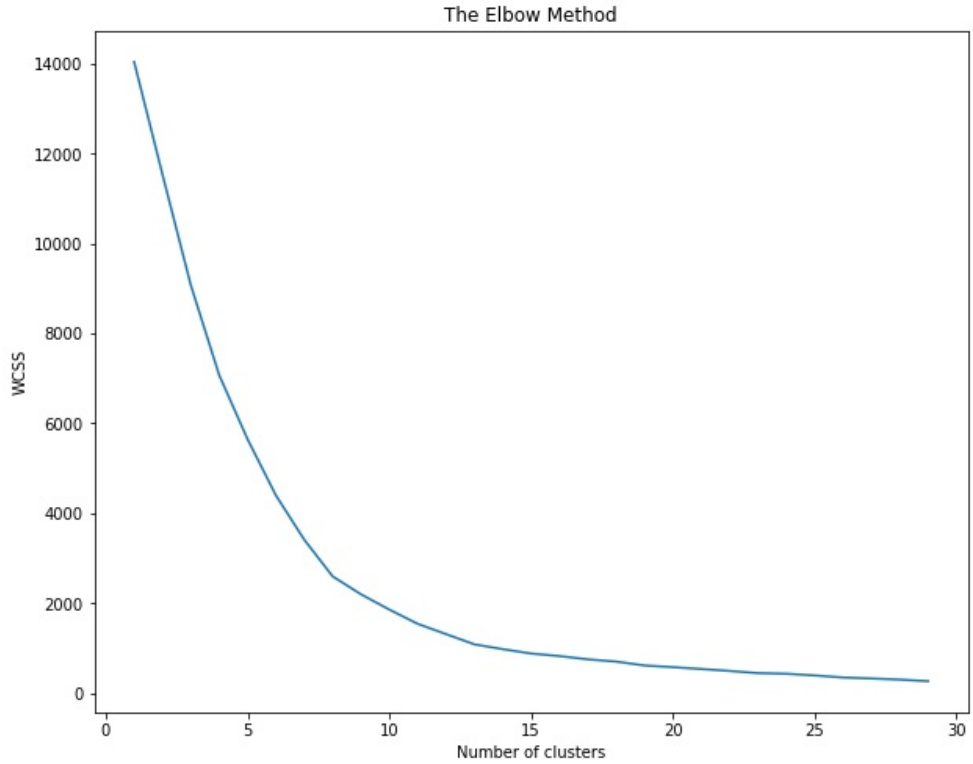

Figure 29. The Elbow Method for the Ex. A (considering only younger students) (Within-Cluster-Sum-of-Squares (WCSS) over the Number of clusters)

Calculating the Pearson correlation coefficient between the features extracted with the k-means algorithms (presented in Table 5) and the final results obtained by younger students' teams, three features show a statistically significant negative correlation in the Exercise A:

- the percentage of sequences of the cluster named "LARGE MOTORS PARAMETERS CHANGE" (Pearson correlation coefficient (PCC) = -0.31, p-value < 0.02)
- the number of trials (PCC = -0.32, p-value = 0.01) and
- the percentage of sequences of the cluster named "HIGH VALUE IN MOTORS PARAMETERS" (PCC = -0.32, p-value = 0.01);

High values of these features indicate higher probability of a negative performance.

3.2.2 Exercise B

In this exercise the 39,1% of the younger students' teams had a negative performance, while the 6,3% of older students' groups had a negative performance (see Figure 30 and Figure 36).
Being the negative performance examples more concentrated in the students younger than 12 years old, we decided to not perform the prediction analysis for this exercise (too few "negative performance" examples for the older students' subset), and to analyse the students' teams problem-solving behaviours through the clustering analysis.

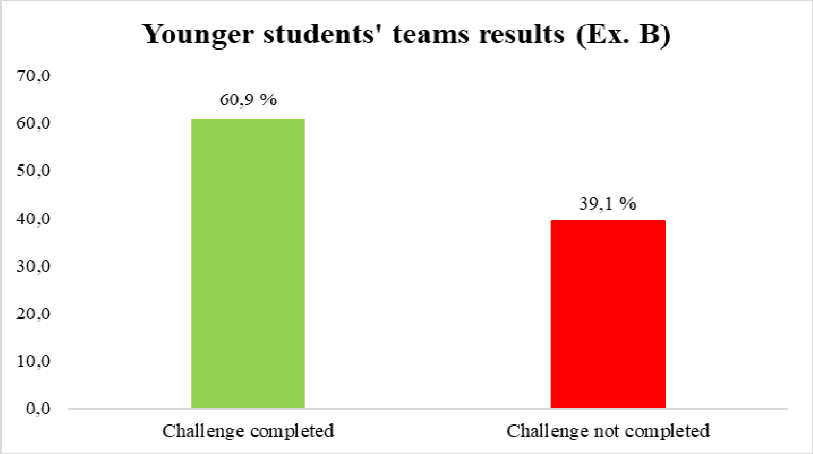

Figure 30. Younger students' teams results for the Ex. B

Table 7 show the clusters resulting from the k-means algorithms applied considering the younger students' teams during the resolution of Exercise B; to select the number of clusters in which divide the programming sequences generated by the younger students the Elbow Method (see Figure 31) was applied. Table 8 contains the mean values and standard

deviation values of the thirteen indicators (presented in the previous chapter) calculated for each cluster, for Exercise B (#: number of sequences).

Table 7. Clusters obtained with the k-means algorithm applied on the Ex. B (considering only younger students).

	Name	#	%	Description
1	**TEST "WAIT" SEQUENCE WITHOUT "TURN OFF" BLOCK**	1140	36,9	The team is testing the same previous programming sequence (with Motors and Conditionals (Wait) blocks, but without the Turn Off motor block).
2	**LARGE CONDITIONAL PARAMETERS CHANGE**	156	5,1	The team is strongly changing its Conditional parameters.
3	**TEST "SWITCH" SEQUENCE WITH OTHERS BLOCKS**	46	1,5	The team is testing the same previous and complex programming sequence (with Motors, Loops, Others and Conditionals (Switch) blocks).
4	**TEST "SWITCH" SEQUENCE**	440	14,3	The team is testing the same previous programming sequence (with Motors, Loops and Conditionals (Switch) blocks).
5	**DELETING BLOCKS**	2	0,1	The team is deleting some blocks in the sequence.
6	**LARGE MOTORS PARAMETERS CHANGE (SWITCH / WAIT BLOCKS)**	35	1,1	The team is changing strongly its Motors parameters.
7	**SMALL CONDITIONALS PARAMETERS CHANGE ("WAIT" SEQUENCE)**	387	12,5	The team is refining its conditionals parameters in a "Wait" sequence (there aren't Loops blocks in these cluster).
8	**TEST MOTORS BLOCKS SEQUENCE**	69	2,2	The team is testing the same previous programming sequence (with only Motors blocks).
9	**VERY LARGE CONDITIONAL**	68	2,2	The team is very strongly changing its Conditional parameters (both

	PARAMETERS CHANGE (SWITCH / WAIT SEQUENCE)			ultrasonic threshold and condition).
10	**ADDING BLOCKS**	64	2,1	The team is adding some blocks in the sequence.
11	**TEST "WAIT" SEQUENCE (NO LOOPS)**	628	20,3	The team is testing the same previous programming sequence (with Motors and Conditionals blocks (Wait), and without Loop blocks).
12	**SMALL CONDITIONALS PARAMETERS CHANGE ("SWITCH" SEQUENCE)**	52	1,7	The team is refining its Conditionals parameters (with "Switch" and Loop blocks in the sequence).

Table 8. Mean values and standard deviation (M (SD)) calculated for the 13 indicators for each cluster in Table 7.

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaConditionals	DeltaOthers	Motors	Loops	CondWait	Others
1	3,80 (0,88)	0,00 (0,00)	0,02 (0,14)	0,03 (0,16)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,94 (0,24)	0,04 (0,09)	2,01 (0,09)	1,01 (0,10)
2	3,35 (1,21)	1,10 (0,30)	0,32 (0,47)	0,21 (0,41)	1,50 (6,02)	38,82 (7,70)	0,00 (0,00)	1,34 (0,54)	0,35 (0,79)	2,06 (0,23)	1,02 (0,14)
3	11,04 (2,87)	0,41 (0,80)	0,39 (0,95)	0,13 (0,34)	0,00 (0,00)	0,95 (2,41)	0,00 (0,00)	1,98 (0,15)	3,70 (1,50)	2,96 (0,30)	3,74 (0,20)
4	8,10 (0,82)	0,22 (0,60)	0,05 (0,22)	0,05 (0,22)	0,12 (1,67)	0,40 (1,66)	0,00 (0,00)	1,93 (0,29)	2,63 (0,52)	2,59 (0,50)	1,20 (0,40)
5	6,00 (1,41)	2,00 (0,00)	0,5 (0,71)	1,00 (1,41)	25,00 (35,36)	20,00 (28,28)	1,00 (0,00)	1,5 (0,71)	2,00 (0,00)	2,5 (0,71)	2,5 (0,71)
6	2,78 (1,40)	1,49 (0,51)	0,20 (0,58)	0,14 (0,43)	64,99 (26,37)	18,50 (24,07)	0,00 (0,00)	1,20 (0,41)	0,23 (0,65)	1,94 (0,54)	1,03 (0,17)
7	3,43 (0,66)	1,04 (0,22)	0,03 (0,17)	0,03 (0,17)	1,67 (5,60)	3,14 (4,05)	0,00 (0,00)	1,42 (0,50)	0,05 (0,32)	2,03 (0,16)	1,00 (0,00)
8	1,64 (0,75)	0,09 (0,28)	0,04 (0,21)	0,01 (0,12)	0,55 (2,42)	0,00 (0,00)	0,00 (0,00)	0,80 (0,44)	0,01 (0,12)	0,25 (0,43)	1,00 (0,00)
9	3,91 (1,48)	1,03 (0,17)	0,40 (0,65)	0,19 (0,55)	0,74 (6,06)	94,40 (38,92)	0,00 (0,00)	1,47 (0,53)	0,60 (1,01)	2,18 (0,38)	1,09 (0,29)
10	4,27 (2,41)	0,30 (0,49)	2,59 (1,16)	0,30 (0,55)	1,88 (7,40)	5,91 (13,80)	0,00 (0,00)	1,70 (0,52)	1,58 (1,29)	2,55 (0,56)	1,22 (0,42)

11	4,92 (0,52)	0,00 (0,00)	0,04 (0,18)	0,01 (0,09)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	2,00 (0,04)	0,01 (0,17)	1,99 (0,07)	1,00 (0,04)
12	5,00 (2,38)	0,25 (0,50)	0,25 (0,59)	2,35 (0,94)	1,04 (6,93)	5,20 (13,48)	0,00 (0,00)	1,13 (0,44)	1,50 (1,18)	1,65 (0,84)	1,17 (0,43)

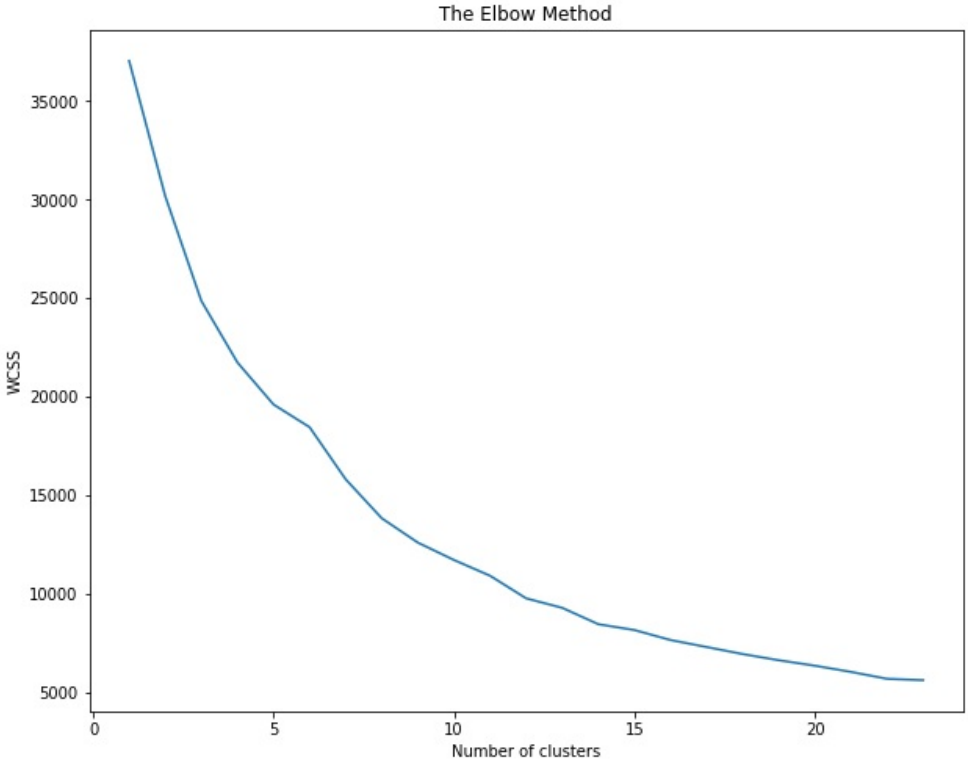

Figure 31. The Elbow Method for the Ex. B (considering only younger students) (Within-Cluster-Sum-of-Squares (WCSS) over the Number of clusters)

Exercise A and B have a different number of clusters because younger students used blocks from different categories to solve them: in the Exercise A only a Motor block was required (but some students teams explored blocks from other categories also in this challenge), in the exercise B Motor blocks, Conditional blocks, Loop blocks were essential.

3.3 *Older students*

3.3.1 Exercise A

Figure 32 and Figure 33 show the resulted performances in comparison within the four machine learning algorithms in the prediction of the older students' teams result, analysing

Exercise A. In this exercise the 23,8% of teams of this subset of participants had a negative performance (see Figure 34).

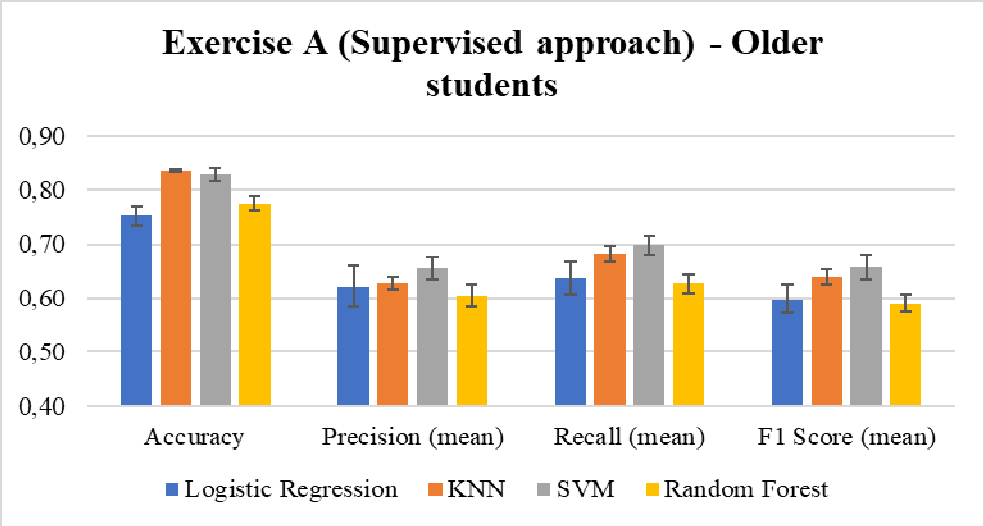

Figure 32. Results for the Ex. A (considering only older students) obtained applying the supervised approach

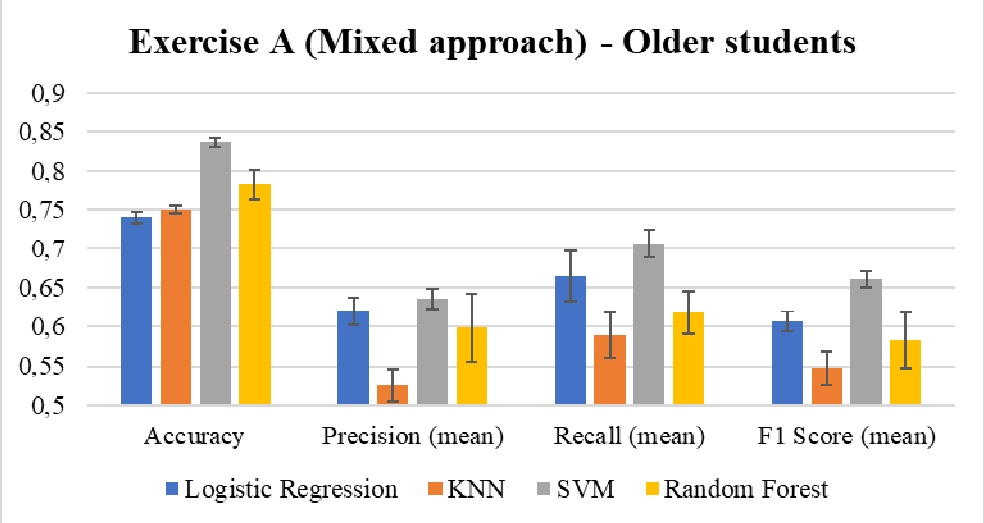

Figure 33. Results for the Ex. A (considering only older students) obtained applying the mixed approach

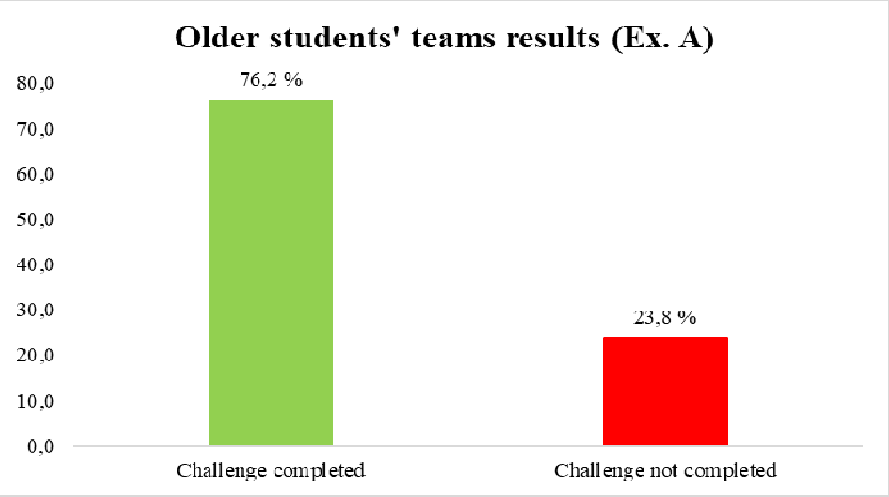

Figure 34. Older students' teams results for Ex. A

For both the approaches in the analysis of the Exercise A, the best performance was obtained with the SVM algorithms, but comparing the supervised and the mixed approach a better trend emerges applying the second one as shown in Table 9.

Table 9. Best performance parameters for the Ex. A (older students).

	SVM (supervised) Ex. A - Older	**SVM (mixed) Ex. A - Older**
Accuracy	0,83	**0,84**
Mean Precision	0,66	**0,65**
Mean Recall	0,70	**0,71**
Mean F1 Score	0,66	**0,66**

Table 10 show the clusters resulting from the k-means algorithms; to select the number of clusters in which divide the programming sequences for the Exercise A generated by the older students the Elbow Method (see Figure 35) was applied. Table 11 contains the mean values and standard deviation values of the thirteen indicators (presented in the previous chapter) calculated for each cluster, for Exercise A (#: number of sequences). In Table 11 only eleven indicators are considered, because older students did not use Conditional blocks for the Exercise A.

Table 10. Clusters obtained with the k-means algorithm applied on the Ex. A (considering only older students).

	Name	#	%	Description
1	**TEST OF A MINIMUM SEQUENCE**	476	60,8	The team is testing the same previous programming sequence (with only Motors blocks).
2	**SMALL MOTORS PARAMETERS CHANGE WITH LOOPS**	34	4,3	The team is refining its Motors parameters, in a sequence with also Loops blocks.
3	**STRATEGY CHANGE**	20	2,6	The team is changing its strategy (1 block deleted, 1 added), probably modifying the Move Steering block options (Rotations, Degrees, Seconds).
4	**SMALL MOTORS PARAMETERS CHANGE**	134	17,1	The team is refining its Motors parameters.
5	**CHANGE IN OTHERS BLOCK**	1	0,13	The team is changing something not connected to the challenge (Others blocks are not essential in Ex. A).
6	**HIGH VALUE IN MOTORS PARAMETERS**	29	3,7	The Motors blocks are characterized by high values of rotations or seconds parameters.
7	**LARGE MOTORS PARAMETERS CHANGE**	22	2,8	The team is changing strongly its Motors parameters.
8	**TWO MOTOR BLOCKS**	9	1,2	The team is testing a sequence with two Motors blocks.
9	**SMALL LOOPS PARAMETER CHANGE**	2	0,3	The team is refining its Loops parameters.
10	**TEST OF A COMPLEX SEQUENCE**	56	7,2	The team is testing the same programming sequence (with Others and Motors block).

Table 11. Mean values and standard deviation (M (SD)) calculated for the 13 indicators for each cluster in Table 10.

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaLoops	DeltaOthers	Motors	Loops	Others	HighValues
1	1,86 (0,51)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)
2	4,06 (1,18)	0,12 (0,33)	0,00 (0,00)	0,00 (0,00)	0,84 (4,25)	0,01 (0,07)	0,00 (0,00)	1,00 (0,00)	2,06 (0,24)	1,38 (0,49)	0,00 (0,00)
3	1,45 (0,60)	0,15 (0,37)	1,05 (0,69)	0,90 (0,45)	1,90 (5,90)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,30 (0,73)	1,35 (0,49)	0,10 (0,31)

4	1,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	2,90 (4,82)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)
5	2,00 (0.00)	1,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	17,00 (0,00)	1,00 (0,00)	0,00 (0,00)	3,00 (0,00)	0,00 (0,00)
6	1,59 (0,78)	0,07 (0,26)	0,00 (0,00)	0,00 (0,00)	0,89 (3,36)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	1,00 (0,00)
7	1,05 (0,21)	1,00 (0,00)	0,05 (0,21)	0,00 (0,00)	49,95 (11,53)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,09 (0,29)	0,14 (0,35)
8	2,67 (0,71)	0,11 (0,33)	0,44 (0,73)	0,33 (0,50)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	2,11 (0,33)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)
9	3,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,25 (0,21)	0,00 (0,00)	1,00 (0,00)	2,00 (0.00)	1,00 (0,00)	0,00 (0,00)
10	2,55 (0,76)	0,32 (0,47)	0,04 (0,19)	0,00 (0,00)	0,08 (0,21)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	2,02 (0,13)	0,00 (0,00)

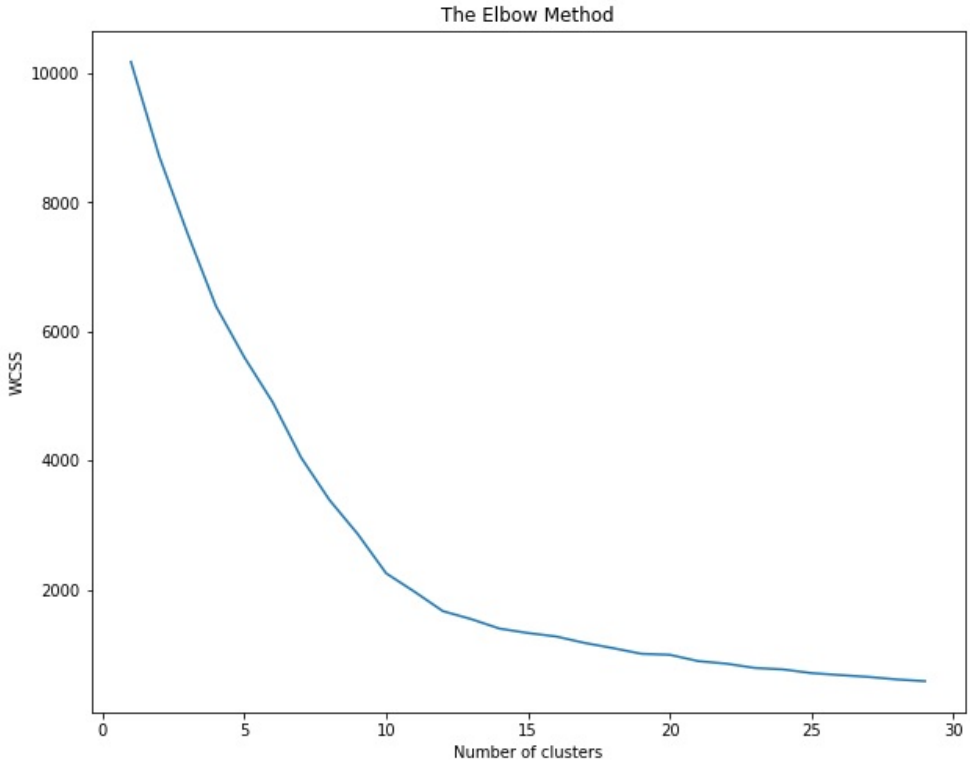

Figure 35. The Elbow Method for the Ex. A (considering only older students) (Within-Cluster-Sum-of-Squares (WCSS) over the Number of clusters)

Calculating the Pearson correlation coefficient between the features extracted with the k-means algorithms (presented in Table 11) and the final results obtained by older students'

teams, one feature shows a statistically significant negative correlation in the Exercise A: the percentage of sequences of the cluster named "LARGE MOTORS PARAMETERS CHANGE" (PCC = -0.45, p-value < 0.01); high values of these features indicate higher probability of a negative performance.

3.3.2 Exercise B

In this exercise only the 6,25% of the older students' teams had a negative performance (see Figure 36).

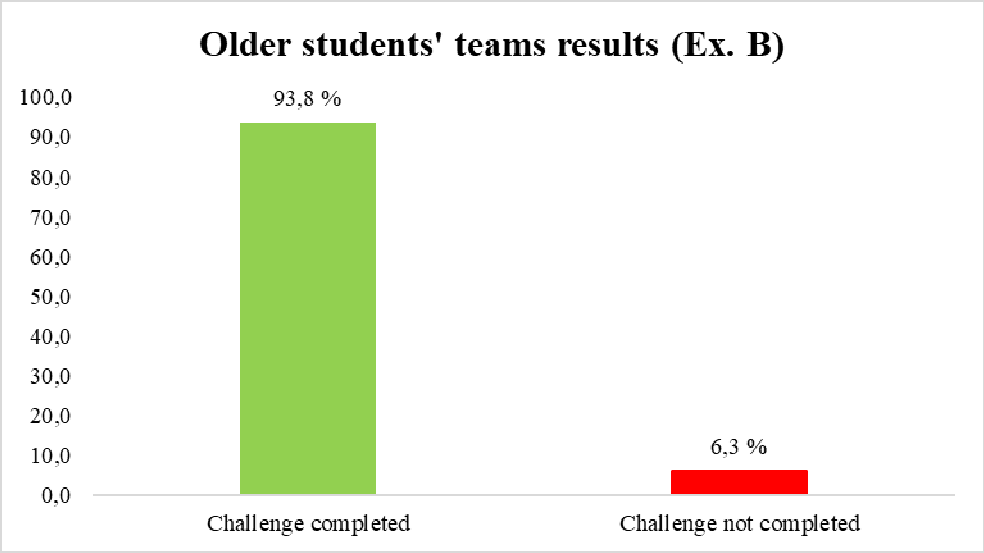

Figure 36. Older students' teams results for the Ex. B

Table 12 show the clusters resulting from the k-means algorithms applied considering the older students' teams; to select the number of clusters in which divide the programming sequences (designed for Exercise B) generated by the older students the Elbow Method (see Figure 37) was applied. Table 12 contains the mean values and standard deviation values of the twelve indicators (presented in the previous section) calculated for each cluster, for Exercise B (#: number of sequences).

Table 12. Clusters obtained with the k-means algorithm applied on the Ex. B (considering only older students).

cluster	Name	#	%	Description
1	**TEST "SWITCH" SEQUENCE (WRONG CONDITION)**	175	15,0	The team is testing the same previous programming sequence (with Motors, Loops and Conditionals (Switch) blocks) but the condition is wrong, indeed only one Case

				(see Lego Mindstorms EV3 Switch block (2019)) was executed.
2	**LARGE MOTORS PARAMETERS CHANGE**	25	2,2	The team is changing strongly its Motors parameters.
3	**CHANGE IN OTHERS BLOCK**	6	0,52	The team is changing something not connected to the challenge (Others blocks are not essential in Ex. B).
4	**ADDING BLOCKS AND DELETING BLOCKS**	12	1,0	The team is adding and deleting some blocks in the sequence.
5	**LARGE CONDITIONAL PARAMETERS CHANGE (SWITCH / WAIT BLOCKS)**	41	3,5	The team is strongly changing its Conditional parameters.
6	**SMALL CONDITIONALS PARAMETERS CHANGE ("SWITCH" SEQUENCE)**	154	13,22	The team is refining its conditionals parameters (with "Switch" and Loop blocks in the sequence).
7	**TEST "SWITCH" SEQUENCE WITH OTHERS BLOCKS**	103	8,8	The team is testing the same previous and complex programming sequence (with Motors, Loops, Others and Conditionals (Switch) blocks).
8	**ADDING BLOCKS**	69	5,9	The team is adding some blocks in the sequence.
9	**DELETING BLOCKS**	70	6,0	The team is deleting some blocks in the sequence.
10	**TEST WITH NO MOTORS BLOCKS**	75	6,4	The team is testing a sequence without Motors blocks.
11	**SMALL LOOPS PARAMETER CHANGE**	1	0,1	The team is refining its Loops parameters.
12	**TEST "WAIT" SEQUENCE (NO LOOPS)**	125	10,7	The team is testing the same previous programming sequence (with Motors and Conditionals blocks (Wait), and without Loop blocks).
13	**TEST "SWITCH" SEQUENCE**	303	26,0	The team is testing the same previous programming sequence (with Motors, Loops and Conditionals (Switch) blocks).

14	**VERY LARGE CONDITIONAL PARAMETERS CHANGE (SWITCH / WAIT SEQUENCE)**	6	0,5	The team is very strongly changing its Conditional parameters (both ultrasonic threshold and condition).

Table 13. Mean values and standard deviation (M (SD)) calculated for the 13 indicators for each cluster in Table 12

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaLoops	DeltaConditionals	DeltaOthers	Motors	Loops	CondWait	Others
1	7,63 (1,15)	0,17 (0,37)	0,09 (0,29)	0,06 (0,24)	0,54 (2,82)	0,00 (0,00)	0,15 (0,61)	0,00 (0,00)	1,45 (0,50)	2,26 (0,44)	2,02 (0,17)	1,10 (0,30)
2	4,84 (2,01)	1,40 (0,71)	0,56 (1,16)	0,40 (1,19)	52,51 (13,71)	0,00 (0,00)	9,60 (20,37)	0,00 (0,00)	1,68 (0,48)	1,44 (1,19)	2,28 (0,46)	1,00 (0,00)
3	7,50 (1,05)	2,83 (0,75)	0,00 (0,00)	0,00 (0,00)	6,67 (8,76)	0,00 (0,00)	0,60 (0,67)	20,00 (3,85)	2,00 (0,00)	3,00 (0,00)	3,00 (0,00)	2,33 (0,52)
4	7,42 (1,62)	0,67 (0,98)	1,83 (1,70)	1,08 (1,16)	6,17 (14,46)	0,00 (0,00)	3,50 (11,51)	0,00 (0,00)	1,83 (0,39)	1,33 (0,49)	5,50 (0,80)	1,08 (0,29)
5	5,20 (1,21)	1,12 (0,40)	1,12 (1,14)	0,41 (0,74)	0,61 (3,90)	0,00 (0,00)	54,23 (21,92)	0,00 (0,00)	1,44 (0,59)	2,32 (0,93)	2,51 (0,55)	1,12 (0,33)
6	7,01 (0,78)	1,99 (0,34)	0,06 (0,24)	0,04 (0,19)	1,18 (0,19)	0,00 (0,00)	4,19 (5,55)	0,00 (0,00)	1,97 (0,16)	2,81 (0,46)	3,00 (0,20)	1,25 (0,51)
7	11,68 (1,72)	0,12 (0,38)	0,22 (0,59)	0,07 (0,29)	0,63 (2,68)	0,00 (0,00)	0,11 (0,83)	0,02 (0,14)	1,99 (0,10)	2,99 (0,10)	2,99 (0,10)	4,17 (1,07)
8	5,77 (1,44)	0,23 (0,42)	2,46 (0,88)	0,28 (0,51)	1,05 (4,43)	0,00 (0,00)	1,86 (6,52)	0,00 (0,00)	1,81 (0,43)	2,29 (0,75)	2,62 (0,69)	1,12 (0,63)
9	6,23 (1,97)	0,37 (0,51)	0,26 (0,47)	2,31 (0,63)	0,48 (2,37)	0,00 (0,00)	4,69 (11,13)	0,00 (0,00)	0,99 (0,47)	2,11 (0,73)	2,04 (0,49)	1,43 (1,02)
10	2,21 (2,15)	0,01 (0,11)	0,20 (0,57)	0,07 (0,25)	0,00 (0,00)	0,00 (0,00)	0,33 (2,89)	0,00 (0,00)	0,24 (0,43)	0,97 (1,01)	1,99 (0,51)	1,19 (0,39)
11	5,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	20,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	2,00 (0,00)	2,00 (0,00)	1,00 (0,00)
12	4,31 (1,03)	0,10 (0,32)	0,18 (0,43)	0,12 (0,33)	0,31 (2,34)	0,00 (0,00)	0,41 (2,45)	0,00 (0,00)	1,62 (0,55)	0,08 (0,33)	2,06 (0,28)	1,01 (0,09)
13	8,83 (1,10)	0,00 (0,06)	0,03 (0,16)	0,04 (0,19)	0,00 (0,06)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,98 (0,16)	2,78 (0,50)	3,03 (0,16)	1,15 (0,36)
14	5,17 (1,72)	2,00 (0,63)	0,67 (1,21)	0,17 (0,41)	2,50 (6,12)	0,00 (0,00)	155,41 (5,66)	0,00 (0,00)	1,67 (0,51)	2,33 (0,51)	2,83 (0,41)	1,00 (0,00)

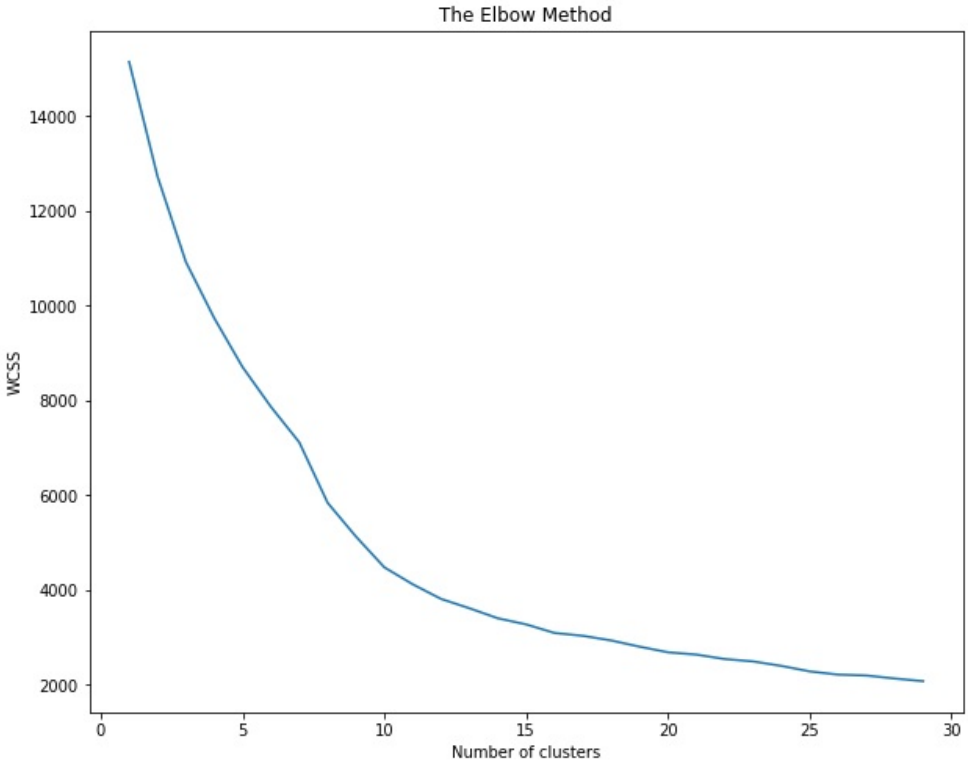

Figure 37. The Elbow Method for the Ex. B (considering only older students) (Within-Cluster-Sum-of-Squares (WCSS) over the Number of clusters)

Exercise A and B have a different number of clusters because older students used blocks from different categories to solve them: in the Exercise A only a Motor block was required (but some students teams explored blocks from other categories also in this challenge), in the exercise B Motor blocks, Conditional blocks, Loop blocks were essential.

3.4 *Complete dataset*

The previous analysis showed how problem-solving activity carried out by students, regardless of the age, was quite similar:

- Considering the Exercise A, applying clustering on programming sequences designed by younger students, 7 clusters are obtained; the same 7 clusters emerged analysing the sequences created by older students, but in this case, we have 3 more clusters: SMALL MOTORS PARAMETERS CHANGE WITH LOOPS, SMALL LOOPS PARAMETER CHANGE (some older students decided to solve the Ex. A using Loop blocks), TWO MOTORS BLOCKS (some older students decided to

solve the Exercise A using two Motors blocks instead of one). Therefore, considering this challenge, some of the older participants discovered different problem-solving paths. For both the students' subsets the cluster named "LARGE MOTORS PARAMETERS CHANGE" has a statistically significant negative correlation with the teams' final results. In terms of prediction the SVM algorithm (using the mixed approach) outperformed the other techniques, and this trend is verified for both the sub-datasets.

- Considering the Exercise B, as previously declared, only the k-means algorithm was applied to study the different types of programming sequences created by younger and older students; the clustering analysis shows that similar clusters emerged from the sub-datasets: the participants tried to solve the challenge mainly using two types of sequences, the Wait sequence (see Figure 38) and the Switch sequence (see Figure 39); younger students largely chose the first one (the Wait block), while the older largely chose the second one (the Switch block). It's important to underline that these two solutions are equivalent.

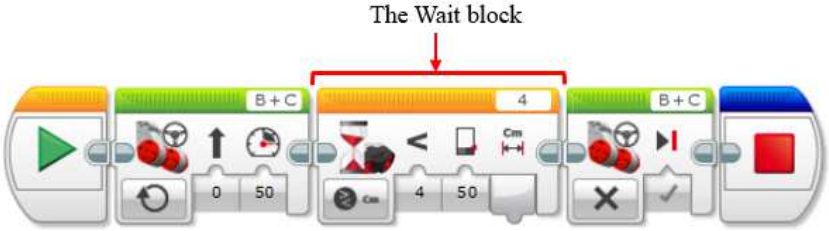

Figure 38. Example of the usage of the Wait Block setting a condition on the Ultrasonic Sensor

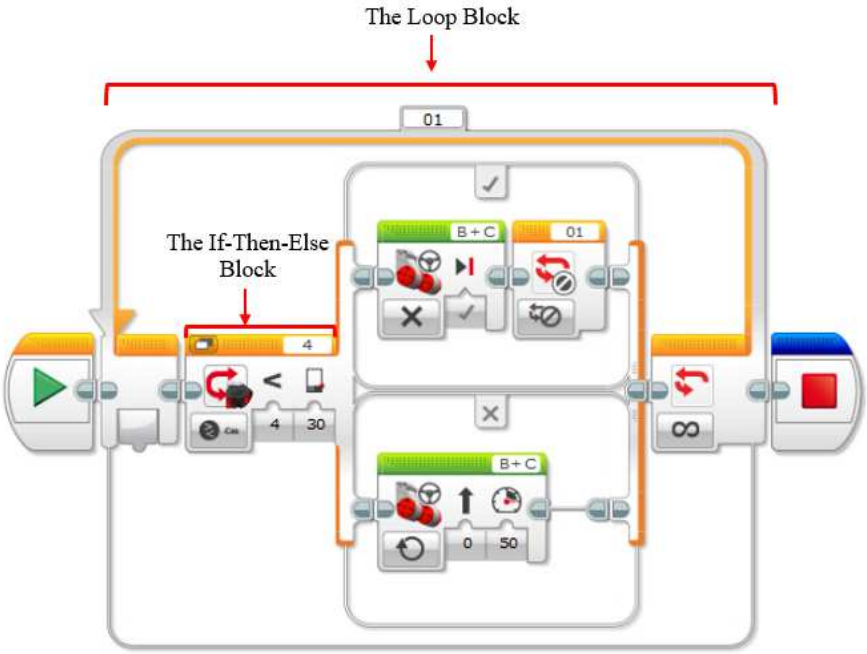

Figure 39. Example of the usage of the Switch and Loop Blocks setting a condition on the Ultrasonic Sensor

Being the prediction and clustering results for younger and older students very similar, we have decided to aggregate the two sub-datasets and test the previously cited techniques, trying to obtain better performances with a larger dataset (Jain & Chandrasekaran, 1982).

3.4.1 Prediction performance of supervised and mixed approach

Figure 40, Figure 41, Figure 42 and Figure 43 show the resulted performances in comparison within the four machine learning algorithms in the prediction of all the students' teams result, analysing Exercise A and B.

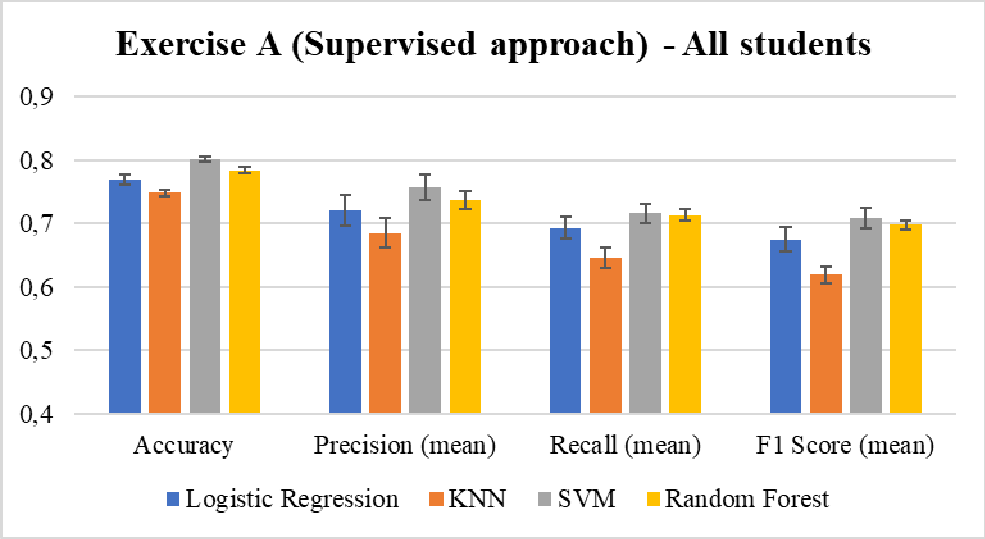

Figure 40. Results for the Ex. A obtained applying the supervised approach (considering the whole dataset)

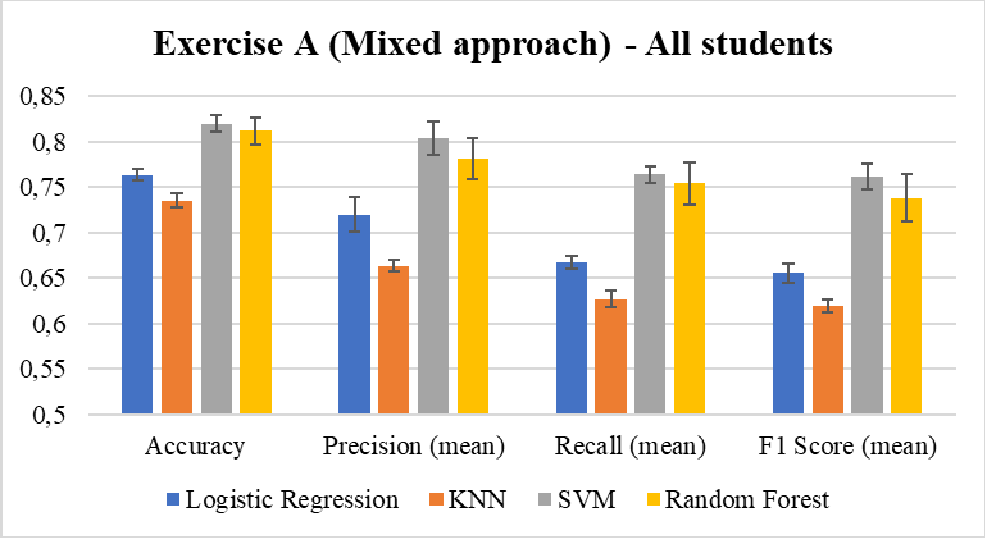

Figure 41. Results for the Ex. A obtained applying the mixed approach (considering the whole dataset)

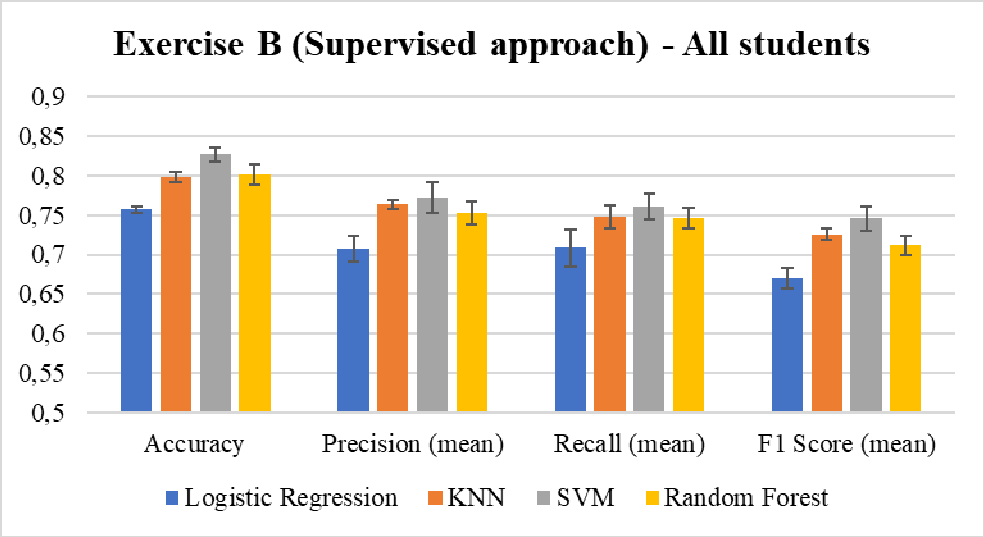

Figure 42. Results for the Ex. B obtained applying the supervised approach (considering the whole dataset)

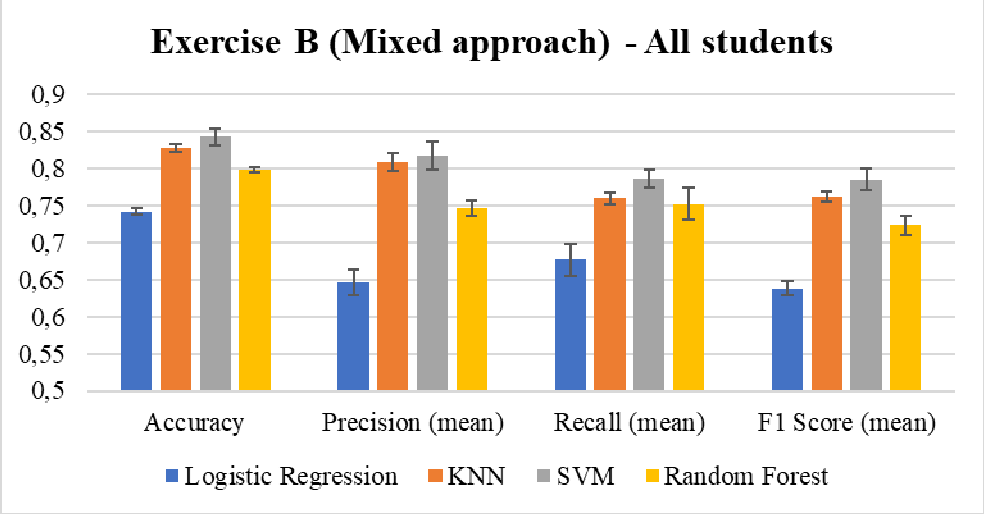

Figure 43. Results for the Ex. B obtained applying the mixed approach (considering the whole dataset)

For both the approaches in the analysis of the exercises A and B, the best performance was obtained with the SVM algorithms, but comparing the supervised and the mixed approach a better trend emerges applying the second one as shown in Table 14.

Table 14. Best performance parameters for the Ex. A and B, considering the whole dataset.

	SVM (supervised) Ex. A	SVM (mixed) Ex. A	SVM (supervised) Ex. B	SVM (mixed) Ex. B
Accuracy	0,80	**0,82**	0,83	**0,84**
Mean Precision	0,76	**0,80**	0,77	**0,82**
Mean Recall	0,72	**0,76**	0,76	**0,79**
Mean F1 Score	0,71	**0,76**	0,75	**0,79**

Table 15 and Table 17 show the clusters resulting from the k-means algorithms; to select the number of clusters in which divide the programming sequences generated by the students the Elbow Method (see Figure 44) was applied. Table 16 and Table 18 contains the mean values and standard deviation values of the thirteen indicators (presented in the previous chapter) calculated for each cluster, both for Exercise A and B (#: number of sequences). In Table 10 only ten indicators are considered, because students did not use Conditional blocks for the Exercise A.

Table 15. Clusters obtained with the k-means algorithm applied on the Ex. A.

	Name	#	%	Description
1	**TEST OF A MINIMUM SEQUENCE**	1282	58,6	The team is testing the same previous programming sequence (with only Motors blocks).
2	**LARGE MOTORS PARAMETERS CHANGE**	47	2,2	The team is changing strongly its Motors parameters.
3	**STRATEGY CHANGE**	40	1,8	The team is changing its strategy (1 block deleted, 1 added), probably modifying the Move Steering block options (Rotations, Degrees, Seconds).
4	**SMALL MOTORS PARAMETERS CHANGE**	523	23,9	The team is refining its Motors parameters.
5	**CHANGE IN OTHERS BLOCK**	1	0,05	The team is changing something not connected to the challenge (Others blocks are not essential in Ex. A).

6	**SMALL LOOPS PARAMETER CHANGE**	2	0,09	The team is refining its Loops parameters.
7	**SMALL MOTORS PARAMETERS CHANGE WITH LOOPS**	36	1,6	The team is refining its Motors parameters, in a sequence with also Loops blocks.
8	**HIGH VALUE MOTORS PARAMETERS**	166	7,6	The motor blocks are characterized by high values of rotations or seconds parameters.
9	**TEST OF A COMPLEX SEQUENCE**	81	3,7	The team is testing the same programming sequence (with Others and Motors block).
10	**TWO MOTOR BLOCKS**	9	0,4	The team is testing a sequence with two Motors blocks.

Table 16. Mean values and standard deviation (M (SD)) calculated for the 11 indicators for each cluster in Table 15.

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaLoops	DeltaOthers	Motors	Loops	Others	HighValues
1	2,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)
2	1,02 (0,15)	1,00 (0,00)	0,02 (0,15)	0,00 (0,00)	49,10 (15,69)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,04 (0,20)	0,23 (0,43)
3	1,23 (0,42)	0,05 (0,22)	0,98 (0,53)	0,98 (0,28)	0,33 (1,64)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,05 (0,32)	1,20 (0,46)	0,05 (0,22)
4	0,84 (0,37)	0,84 (0,37)	0,00 (0,00)	0,00 (0,00)	2,09 (4,07)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)
5	2,00 (0,00)	1,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	17,00 (0,00)	1,00 (0,00)	0,00 (0,00)	3,00 (0,00)	0,00 (0,00)
6	3,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,25 (0,21)	0,00 (0,00)	1,00 (0,00)	2,00 (0,00)	1,00 (0,00)	0,00 (0,00)
7	3,97 (1,21)	0,14 (0,35)	0,11 (0,46)	0,00 (0,00)	1,49 (5,77)	0,01 (0,07)	0,00 (0,00)	1,00 (0,00)	2,06 (0,23)	1,42 (0,5)	0,03 (0,17)
8	1,83 (0,55)	0,02 (0,13)	0,00 (0,00)	0,00 (0,00)	0,28 (2,15)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)	1,00 (0,00)	1,00 (0,00)
9	2,65 (0,87)	0,32 (0,47)	0,05 (0,22)	0,00 (0,00)	0,53 (1,90)	0,00 (0,00)	0,01 (0,11)	1,00 (0,00)	0,00 (0,00)	2,14 (0,49)	0,07 (0,26)
10	2,67 (0,71)	0,11 (0,33)	0,44 (0,73)	0,33 (0,5)	0,00 (0,01)	0,00 (0,00)	0,00 (0,00)	2,11 (0,33)	0,00 (0,00)	1,00 (0,00)	0,00 (0,00)

Table 17. Clusters obtained with the k-means algorithm applied on the Ex. B.

cluster	Name	#	%	Description
1	**TEST "WAIT" SEQUENCE (NO LOOPS)**	845	19,9	The team is testing the same previous programming sequence (with Motors and conditionals blocks (Wait), and without Loop blocks).
2	**LARGE CONDITIONAL PARAMETERS CHANGE (SWITCH / WAIT BLOCKS)**	91	2,1	The team is strongly changing its Conditional parameters.
3	**SMALL CONDITIONALS PARAMETERS CHANGE ("SWITCH" SEQUENCE)**	230	5,4	The team is refining its conditionals parameters (with "Switch" and Loop blocks in the sequence).
4	**SMALL CONDITIONALS PARAMETERS CHANGE ("WAIT" SEQUENCE)**	389	9,1	The team is refining its conditionals parameters in a "Wait" sequence (there aren't Loops blocks in these cluster).
5	**CHANGE IN OTHERS BLOCK**	6	0,1	The team is changing something not connected to the challenge (Others blocks are not essential in Ex. B).
6	**TEST "SWITCH" SEQUENCE**	515	12,1	The team is testing the same previous programming sequence (with Motors, Loops and Conditionals (Switch) blocks).
7	**SMALL LOOPS PARAMETER CHANGE**	1	0,02	The team is refining its Loops parameters.
8	**LARGE MOTORS PARAMETERS CHANGE**	66	1,6	The team is changing strongly its Motors parameters.
9	**DELETING BLOCKS**	116	2,7	The team is deleting some blocks in the sequence.

10	**ADDING BLOCKS**	157	3,7	The team is adding some blocks in the sequence.
11	**TEST MOTORS BLOCKS SEQUENCE**	63	1,5	The team is testing the same previous programming sequence (with only Motors blocks).
12	**TEST "SWITCH" SEQUENCE (WRONG CONDITION)**	371	8,7	The team is testing the same previous programming sequence (with Motors, Loops and Conditionals (Switch) blocks) but the condition is wrong, indeed only one Case (see Lego Mindstorms EV3 Switch block (2019)) was executed.
13	**TEST "SWITCH" SEQUENCE WITH OTHERS BLOCKS**	144	3,4	The team is testing the same previous and complex programming sequence (with Motors, Loops, Others and Conditionals (Switch) blocks).
14	**TEST "WAIT" SEQUENCE WITHOUT "TURN OFF" BLOCK**	1258	29,6	The team is testing the same previous programming sequence (with Motors and Conditionals (Wait) blocks, but without the Turn Off motor block).

Table 18. Mean values and standard deviation (M (SD)) calculated for the 12 indicators for each cluster in Table 17.

	Equal	Modified	Added	Deleted	DeltaMotors	DeltaLoops	DeltaConditionals	DeltaOthers	Motors	Loops	CondWait	Others
1	4,73 (0,69)	0,18 (0,38)	0,04 (0,19)	0,01 (0,11)	0,07 (0,68)	0,00 (0,00)	0,42 (1,47)	0,00 (0,00)	2,01 (0,08)	0,03 (0,25)	2,01 (0,10)	1,00 (0,03)
2	4,20 (1,59)	1,12 (0,36)	0,49 (0,79)	0,24 (0,60)	0,82 (5,54)	0,00 (0,00)	95,73 (37,95)	0,00 (0,00)	1,50 (0,52)	0,95 (1,22)	2,26 (0,44)	1,09 (0,28)
3	7,07 1,02	1,97 (0,35)	0,07 (0,25)	0,04 (0,18)	1,45 5,36	0,00 (0,00)	6,12 10,01	0,00 (0,00)	1,97 0,16	2,77 0,64	3,03 0,33	1,31 0,58
4	3,15 0,78	1,03 0,17	0,15 0,36	0,13 (0,33)	1,65 (5,45)	0,00 (0,00)	16,52 (17,53)	0,00 (0,00)	1,12 (0,39)	0,17 (0,58)	2,01 (0,10)	1,01 (0,09)
5	7,50 (1,04)	2,83 (0,75)	0,00 (0,00)	0,00 (0,00)	6,67 (8,76)	0,00 (0,00)	0,60 (0,67)	20,00 (3,85)	2,00 (0,00)	3,00 (0,00)	3,00 (0,00)	2,33 (0,52)
6	8,79 (0,78)	0,01 (0,09)	0,05 (0,23)	0,04 (0,22)	0,01 (0,14)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,97 (0,19)	2,64 (0,58)	3,04 (0,25)	1,22 (0,42)
7	5,00 (0,00)	1,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	20,00 (0,00)	0,00 (0,00)	0,00 (0,00)	1,00 (0,00)	2,00 (0,00)	2,00 (0,00)	1,00 (0,00)
8	3,52 (1,92)	1,48 (0,56)	0,35 (0,79)	0,16 (0,79)	56,78 (23,58)	0,00 (0,00)	16,78 (22,87)	0,02 (0,12)	1,41 (0,49)	0,64 (1,02)	2,06 (0,51)	1,05 (0,27)

9	5,60 (1,93)	0,40 (0,56)	0,32 (0,61)	2,40 (0,82)	1,92 (8,66)	0,00 (0,00)	6,65 (14,21)	0,01 (0,09)	1,07 (0,49)	1,88 (0,91)	2,00 (0,62)	1,19 (0,57)
10	5,08 (2,06)	0,31 (0,49)	2,55 (1,02)	0,29 (0,56)	1,23 (5,58)	0,00 (0,00)	6,17 (14,05)	0,00 (0,00)	1,71 (0,54)	1,96 (1,10)	2,68 (0,82)	1,21 (0,62)
11	1,51 (0,80)	0,11 (0,32)	0,05 (0,21)	0,21 (0,60)	0,61 (2,53)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,97 (0,25)	0,03 (0,18)	0,03 (0,18)	1,03 (0,18)
12	7,64 (0,91)	0,09 (0,29)	0,04 (0,20)	0,07 (0,26)	0,15 (1,48)	0,00 (0,00)	0,36 (3,06)	0,00 (0,00)	1,66 (0,49)	2,52 (0,50)	2,01 (0,10)	1,07 (0,26)
13	11,60 (1,89)	0,10 (0,37)	0,22 (0,57)	0,17 (0,49)	0,45 (2,28)	0,00 (0,00)	0,13 (0,89)	0,01 (0,12)	1,95 (0,22)	3,10 (0,77)	2,94 (0,27)	4,10 (1,05)
14	3,67 (1,04)	0,00 (0,00)	0,03 (0,18)	0,03 (0,18)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,00 (0,00)	0,89 (0,31)	0,07 (0,31)	2,00 (0,18)	1,02 (0,13)

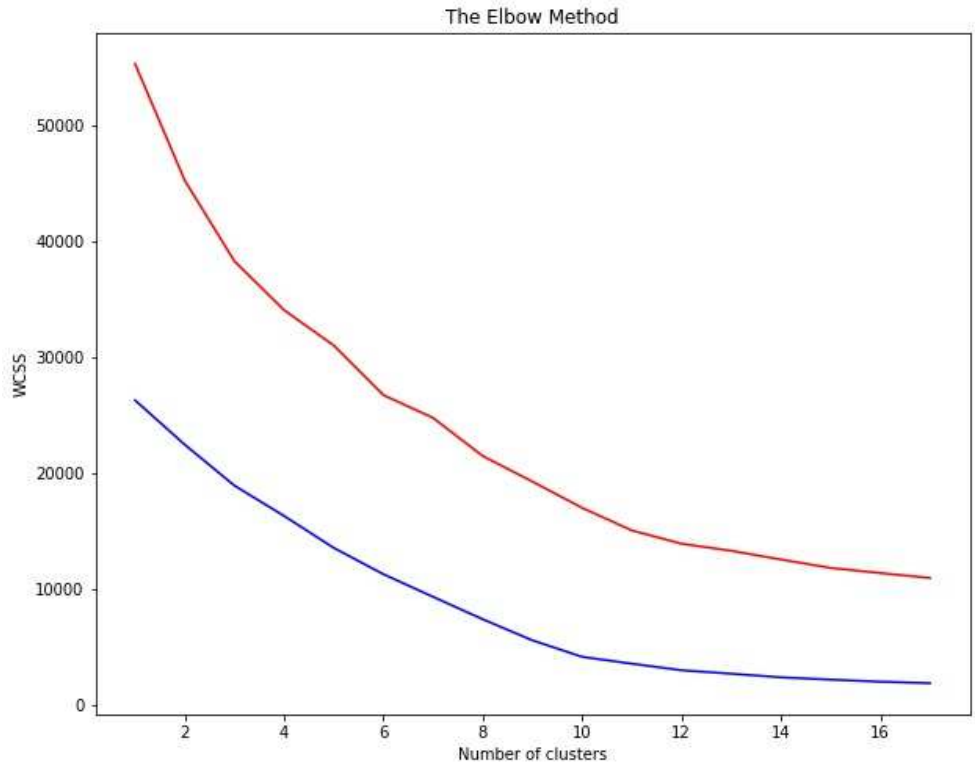

Figure 44. The Elbow Method for the Ex. A (blue curve) and B (red curve) (Within-Cluster-Sum-of-Squares (WCSS) over the Number of clusters)

3.4.2 Problem solving styles

As an example, Figure 45 shows the number of the programming sequences (related to Exercise A) belonging to each cluster in the problem-solving activity performed by 3 students' teams; these three behaviours are the most common in the analysed dataset. Analysing this graph, it's possible to acquire further knowledge in terms of the learning style of the teams:

- The "**Mathematical / planning**" group has a large prevalence of programming sequences in cluster 1 (TEST OF A MINIMUM SEQUENCE), and only 1 sequence in the SMALL MOTORS PARAMETERS CHANGE (cluster 4): they probably used a mathematical formula to decide the parameters of their robot, and applied just one small refinement; they had also a low number of trials (11), considering the average number of trials for this challenge, that is equal to 21.
- The "**Tinkering with prevalent refining behaviour**" group has a prevalence of programming sequences in the SMALL MOTORS PARAMETERS CHANGE (cluster 4), and some sequences in the TEST OF A MINIMUM SEQUENCE cluster. They probably used an heuristic approach to decide the parameters of their robot, and repeatedly refine them analysing the feedback of the robot; they had a number of trials (19) close to the average number of trials for this challenge (21).
- The "**Tinkering with significantly high changes**" group is the only one with some programming sequences in the LARGE MOTORS PARAMETERS CHANGE (cluster 2), and other sequences in the SMALL MOTORS PARAMETERS CHANGE and TEST OF A MINIMUM SEQUENCE clusters. They also probably used an heuristic approach to decide the parameters of their robot, but a certain number of large modifications could indicate a difficulty in the understanding of the robot feedback; they had a number of trials (26) greater than the average number of trials for this challenge (21).

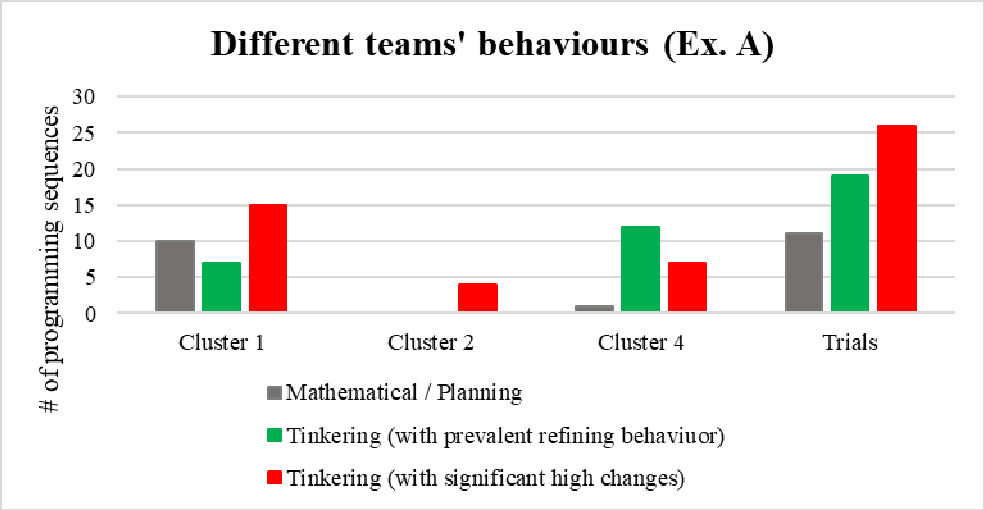

Figure 45. Different problem-solving styles that emerge from the Ex. A

Different reflections can be proposed for the other clusters (not present for the teams taken as models above): STRATEGY CHANGE is typical for teams with a Tinkering approach (both of them), indeed sometimes they tried multiple strategy solving a problem; SMALL MOTORS PARAMETERS CHANGE WITH LOOPS and SMALL LOOPS PARAMETER CHANGE contains sequences characterized by the presence of Loop blocks, an alternative way to solve the Exercise A: the educator does not explain this type of blocks before the Exercise A, so groups that use them already know something about programming. The presence of this type of sequences in the teams' programming activity adds nothing about the problem-solving style of the group. CHANGE IN OTHERS BLOCK cluster contains sequences with changes in blocks not directly connected to the exercise (Sounds, Display, Brick Lights, etc.): usually students who complete the challenge before the deadline try this kind of blocks. Finally, the "HIGH VALUE MOTORS PARAMETERS" cluster contains sequences with Rotations or Seconds values (in the Move Steering block) very high, and for this reason very far from the right solution: this type of sequences could characterise the activity of "Mathematical / planning" groups, when they choose wrong formulas or strategies (or when they get the math wrong), or it could characterise the activity of "Tinkering" groups when they are struggling with the analysis of the robot's feedback.

Figure 46 shows the number of teams of each problem-solving styles in our dataset for Exercise A.

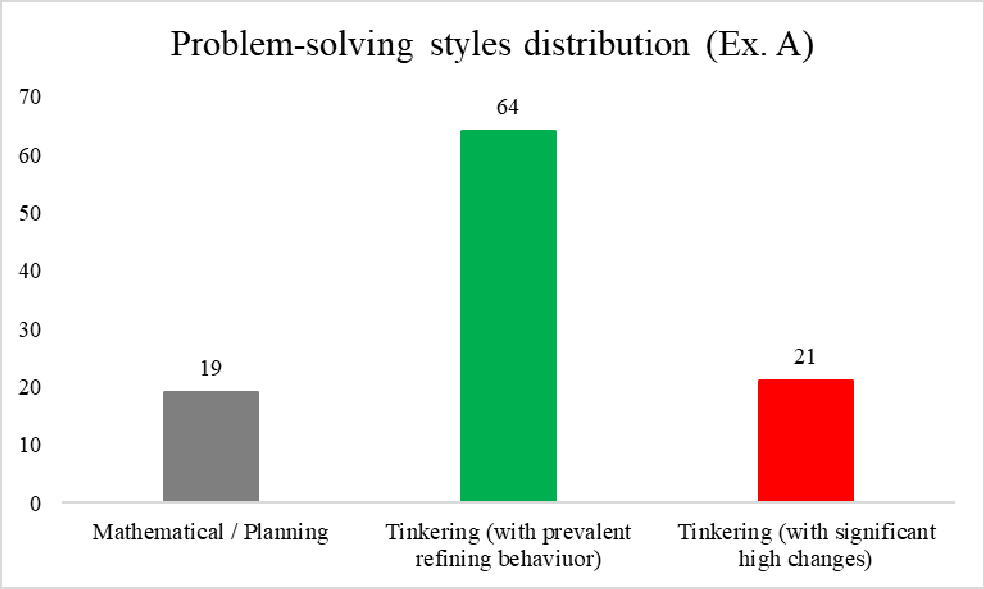

Figure 46. Problem-solving styles showed by students' teams in the Ex. A

Figure 47 shows the same analysis of Figure 45 but related to Exercise B; also in this case, analysing the graph we have acquired similar trends in the learning style of the selected teams:

- The "**Mathematical / planning**" group has a large prevalence of programming sequences in cluster 13 (TEST "SWITCH" SEQUENCE WITH OTHERS BLOCKS): they probably decided their strategy and their algorithm before starting the test process, without applying further changes; they had also a low number of trials (17), considering the average number of trials for this challenge, that is equal to 40.
- The "**Tinkering with prevalent refining behaviour**" group is characterized by a significant number (10) of programming sequences in the SMALL CONDITIONALS PARAMETER CHANGE (WITH LOOPS) (cluster 3). They probably used a heuristic approach to decide the parameters of their robot, and repeatedly refine them analysing the feedback of the robot; they had a number of trials (14) lower than the average number of trials for this challenge (40).
- The "**Tinkering with significantly high changes**" group is the only one with some programming sequences (5) in the LARGE CONDITIONAL PARAMETERS CHANGE (SWITCH / WAIT BLOCKS) (cluster 3), and a high number of sequences (16) in the SMALL CONDITIONALS PARAMETERS CHANGE ("SWITCH" SEQUENCE) cluster and a high number of sequences (18) in the TEST "SWITCH" SEQUENCE (WRONG CONDITION). They also used a heuristic approach to decide the parameters of their robot, but a certain number of large modifications and a high number of tests characterised by a wrong condition could indicate some difficulties in the understanding of the robot feedback; they

had a number of trials (57) greater than the average number of trials for this challenge (40).

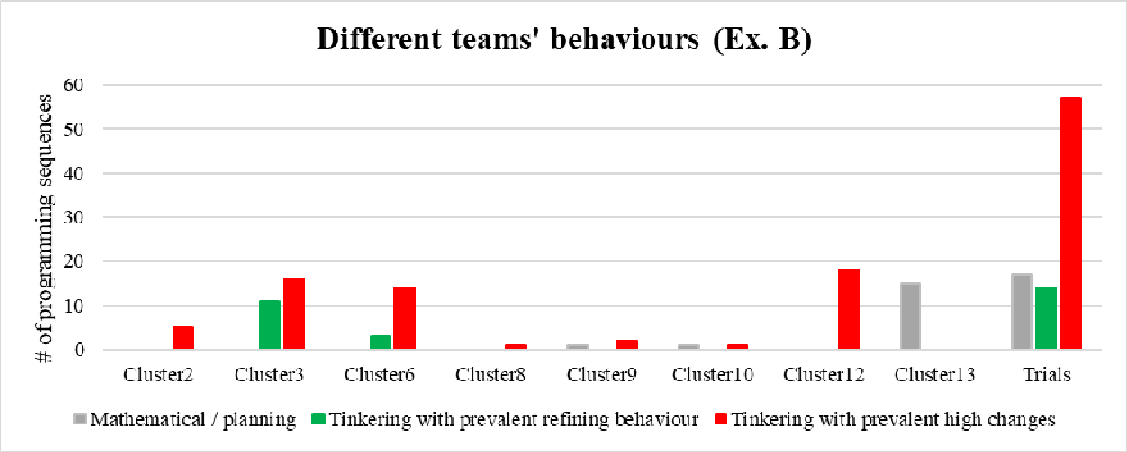

Figure 47. Different problem-solving styles that emerge from the Ex. B

We can propose the same problem-solving styles for teams that chose to design programming sequences using the Wait block (see Figure 38 in section 3.4): a high percentage of TEST "WAIT" SEQUENCE (NO LOOPS) compared to other clusters, could indicate a "Mathematical / planning" approach; a high percentage of SMALL CONDITIONALS PARAMETERS CHANGE ("WAIT" SEQUENCE) compared to other clusters, could indicate a "Tinkering with prevalent refining behaviour" approach; the presence of LARGE CONDITIONAL PARAMETERS CHANGE (SWITCH / WAIT BLOCKS) sequences could indicate a "Tinkering with significantly high changes" approach.
For the Exercise B, we identified also a cluster that could highlight difficulties in the challenge resolution: some teams decided not to use the "Turn Off" Motor block (this type of programming sequences are contained in the cluster TEST "WAIT" SEQUENCE WITHOUT "TURN OFF" BLOCK), so when the condition set for the ultrasonic sensor is true, the motors continue to rotate until the friction stops them (for an example of this sequence see Figure 48). Maybe students that designed this type of sequence did not understand how to use the "Turn Off" Motor block, and that utilising it makes the robot more precise.

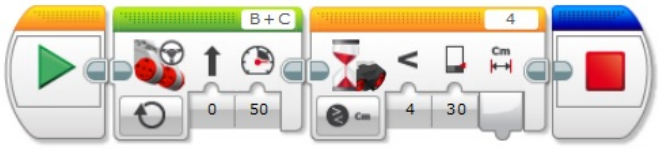

Figure 48. A programming sequence designed for the Ex. B (characterised by the Wait conditional block), without the "Turn Off" Motor block

Figure 49 shows the number of teams for each problem-solving styles in our dataset for Exercise B.

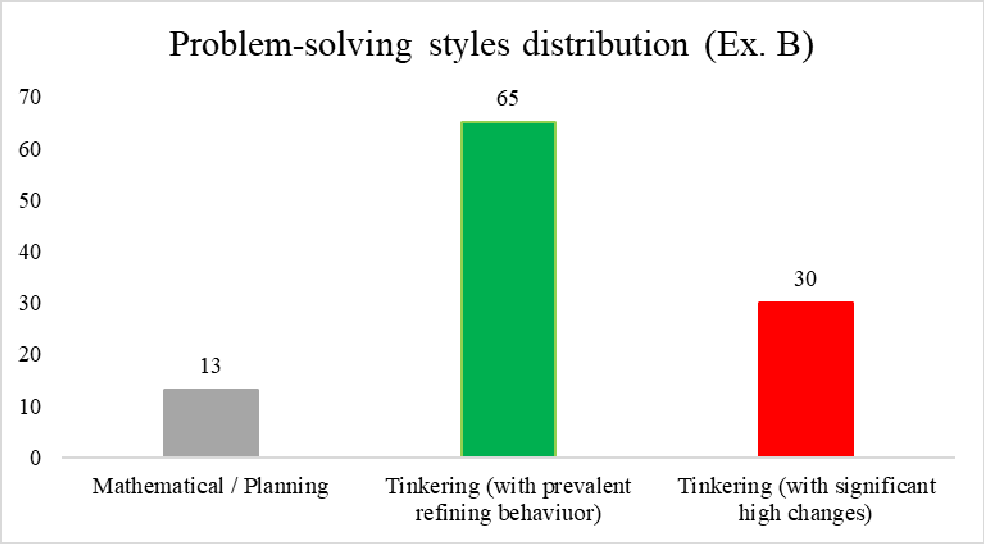

Figure 49. Problem-solving styles showed by students' teams in the Ex. B

Calculating the Pearson correlation coefficient between the features extracted with the k-means algorithms (presented in Table 15 and Table 17) and the final results obtained by students' teams, three features show a statistically significant negative correlation in the Exercise A:

- the number of trials (PCC = -0.24, p-value = 0.01);
- the percentage of sequences of the cluster named "HIGH VALUE MOTORS PARAMETERS" (PCC = -0.26, p-value < 0.01);
- the percentage of sequences of the cluster named "LARGE MOTORS PARAMETERS CHANGE" (PCC = -0.37, p-value < 0.0001).

High values of these features indicate higher probability of a negative performance (see Figure 50).

For the Exercise B, three features show a statistically significant negative correlation:

- the percentage of sequences of the cluster named "SMALL CONDITIONALS PARAMETERS CHANGE ("WAIT" SEQUENCE)" (PCC = -0.24, p-value = 0.01);
- the number of trials (PCC = -0.34, p-value < 0,001)
- the percentage of sequences of the cluster named "TEST "WAIT" SEQUENCE WITHOUT "TURN OFF" BLOCK" (PCC = -0.51, p-value < 0,000001)

Teams with a high percentage of these clusters used the "Wait" Conditional block (this block waits that the condition set by the students becomes true, and then executes the following block in the sequence, see Figure 38) instead of the Loop in combination with the If-Then-Else (the classical structure designed to verify repeatedly a condition in a loop, see Figure 39): it was an alternative and compact way to create a sequence to solve the Exercise B. High number of modifications in this Wait block (so changes related to the condition created to discriminate which action to perform, based on the value detected by the ultrasonic sensor) could reveal a lack of understanding of the "Conditional" concept, and thus a greater difficulty in the exercise resolution. A prevalence of "TEST "WAIT" SEQUENCE WITHOUT "TURN OFF" BLOCK" programming sequences could indicate that students didn't understand how to use the "Turn Off" Motors block, and its importance for making the robot more precise during the braking phase.

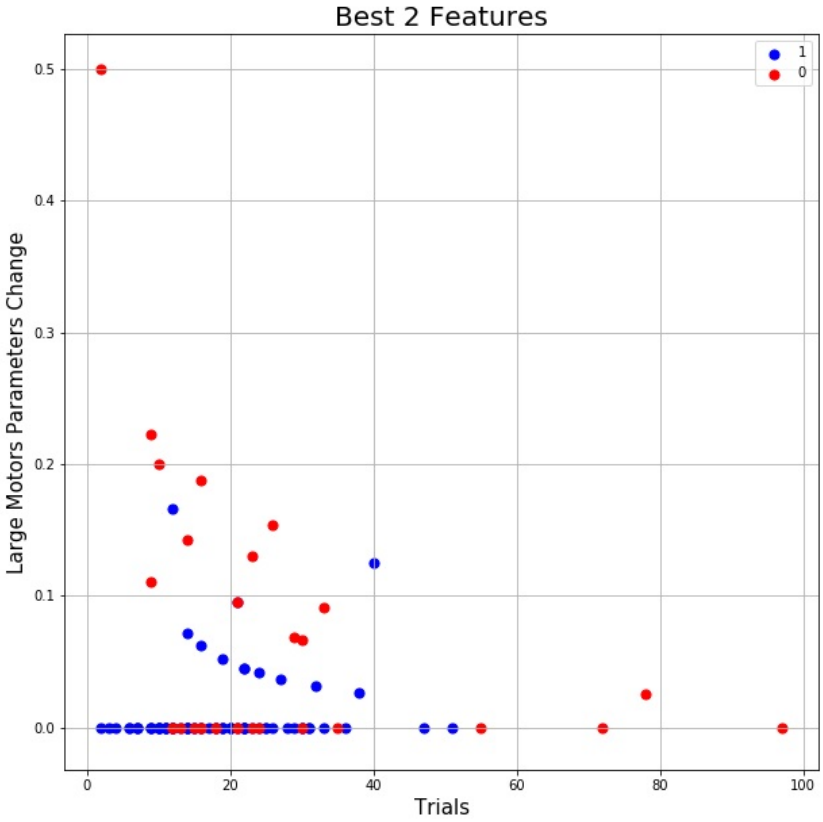

Figure 50. Best correlated features for the Ex. A (1: positive performance, 0: negative performance)

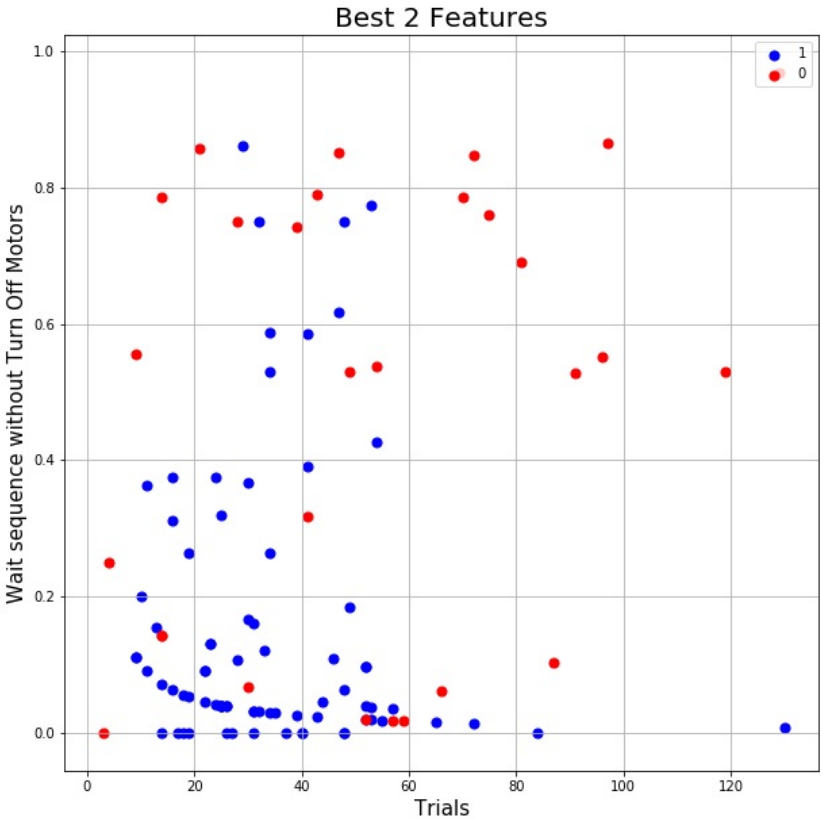

Figure 51. Best correlated features for the Ex. B (1: positive performance, 0: negative performance)

3.4.3 MLP Neural Network prediction performance

Taking into account the previous results, that showed a better performance of the mixed approach, we decided to use the same features presented in Chapter 2 (see Figure 18 and Figure 19) calculated using the k-means algorithm and the indicators chosen to represent the learners' current activity as input to a MLP neural network, with the following structure: one input layer (Input Layer $\in \mathbb{R}^{14}$ for Exercise A and Input Layer $\in \mathbb{R}^{19}$ for Exercise B), 3 hidden layers with the same number of nodes (Hidden Layer (1) $\in \mathbb{R}^{40}$, Hidden Layer (2) $\in \mathbb{R}^{40}$ and Hidden Layer (3) $\in \mathbb{R}^{10}$) for both the exercise, and an output layer (Output Layer $\in \mathbb{R}^{1}$). Figure 52 shows the structure of the MLP neural network.

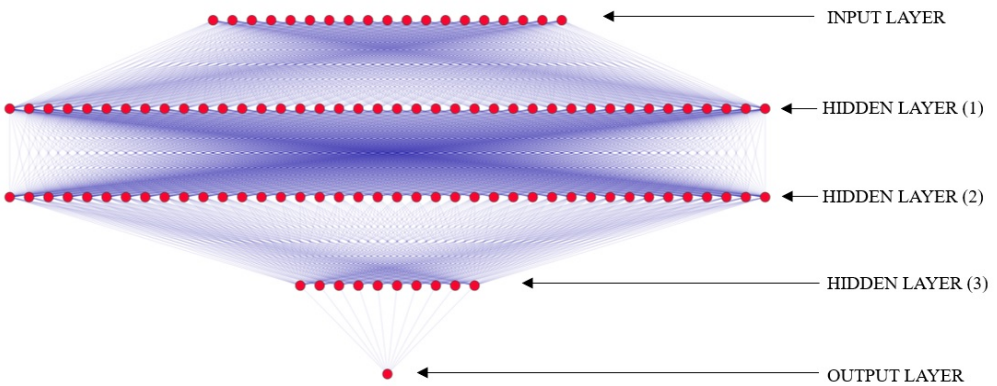

Figure 52. MLP Neural network structure

Figure 53 and Figure 54 show the resulted performances in comparison within the best machine learning technique emerged in the previous section (SVM) and the MLP neural network in the prediction of all the students' teams result (considering all their programming sequences designed during the activity), analysing Exercise A and B.

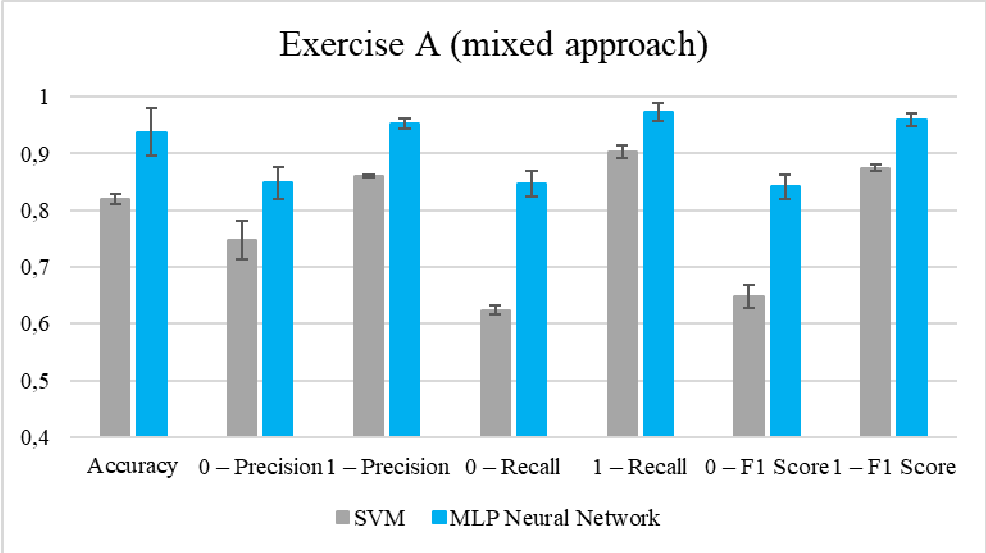

Figure 53. Prediction results for the Ex. A comparing the previous best ML technique (SVM) and the MLP neural network, applying the mixed approach (considering the whole dataset) - (1: positive performance, 0: negative performance)

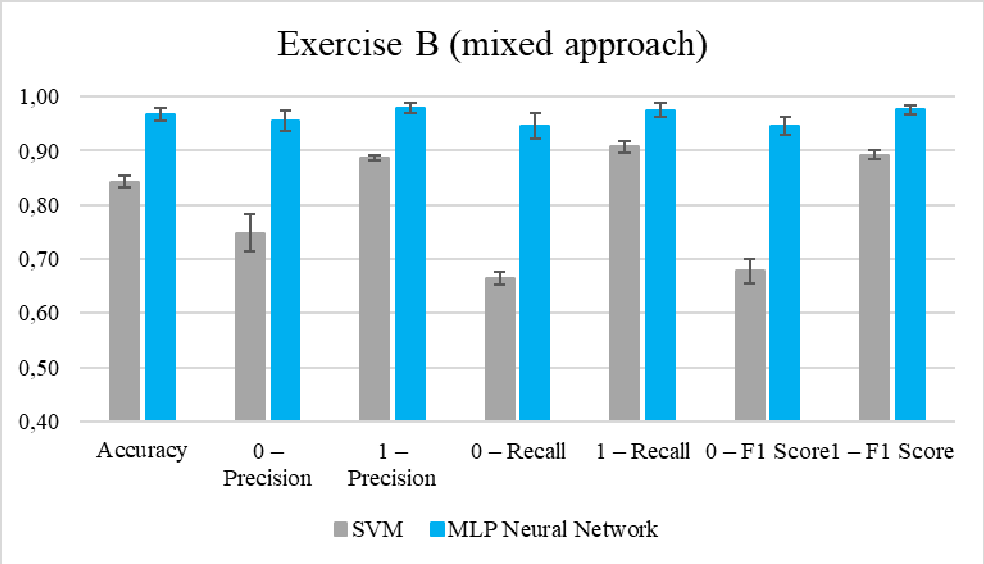

Figure 54. Prediction results for the Ex. B comparing the previous best ML technique (SVM) and the MLP neural network, applying the mixed approach (considering the whole dataset) - (1: positive performance, 0: negative performance)

The MLP neural network technique outperformed the SVM algorithm for both the exercises, as shown in Table 19 and Table 20.

Table 19. SVM and MLP neural network detailed performance for Ex. A (1: positive performance, 0: negative performance)

ML algorithm	Accuracy	0 – Precision	1 – Precision	0 – Recall	1 – Recall	0 – F1 Score	1 – F1 Score
SVM	0,82 (0,01)	0,75 (0,03)	0,86 (0,00)	0,62 (0,01)	0,90 (0,01)	0,65 (0,02)	0,88 (0,01)
MLP Neural Network	0,94 (0,04)	0,85 (0,03)	0,95 (0,01)	0,85 (0,02)	0,97 (0,02)	0,84 (0,02)	0,96 (0,01)

Table 20. SVM and MLP neural network detailed performance for Ex B (1: positive performance, 0: negative performance)

ML algorithm	Accuracy	0 – Precision	1 – Precision	0 – Recall	1 – Recall	0 – F1 Score	1 – F1 Score
SVM	0,84 (0,01)	0,75 (0,03)	0,89 (0,00)	0,67 (0,01)	0,91 (0,01)	0,68 (0,02)	0,89 (0,01)
MLP Neural Network	0,97 (0,01)	0,96 (0,02)	0,98 (0,01)	0,95 (0,02)	0,98 (0,01)	0,95 (0,02)	0,98 (0,01)

To test the performances of the MLP neural network in a real prediction task, we deleted the last programming sequence in the log file designed by each students' group involved in the research project: with this operation we considered only *n-1* programming sequences in the creation of the feature matrix to pass as input for the neural network. We want to simulate a real time prediction task: learners have not yet finished the exercise (this is the

situation we want to create deleting their last trial from the team's log file), and the MLP neural network tries to predict their final performance.
Then, we tested the MLP neural performance in terms of prediction also for *n-2* programming sequences (considering programming sequences from t_0 to $t_{(n-2)}$), eliminating 2 trials from the log files. This is a test to verify if it's possible to anticipate even more the prediction of students' teams with difficulties: in a real time implementation of the system this aspect could be very significant for educators that have to identify pupils struggling on a challenge.

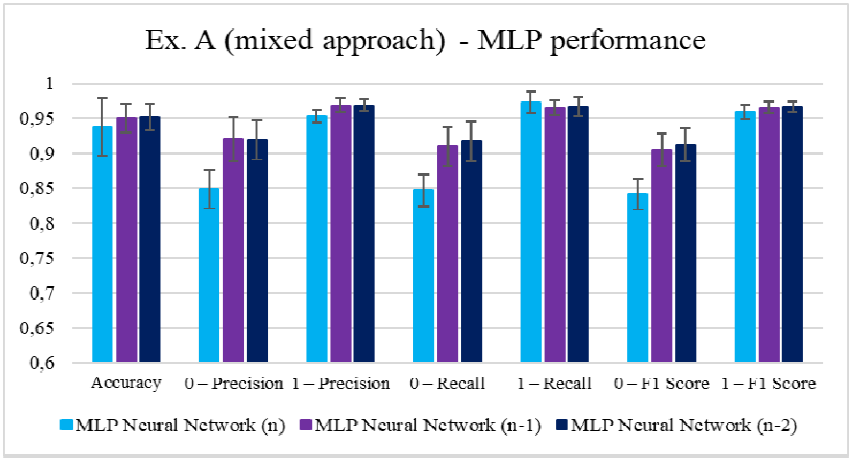

Figure 55. Prediction results for the Ex. A applying the MLP neural network considering n, n-1, n-2 programming sequences for each log files in our dataset (1: positive performance, 0: negative performance)

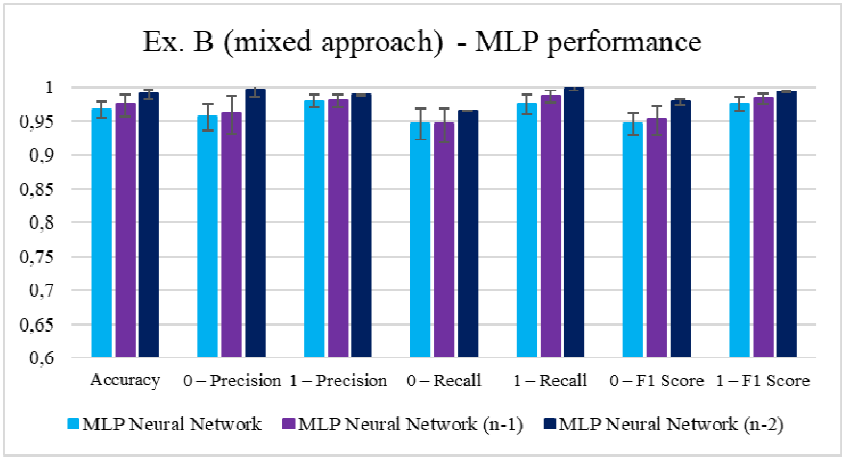

Figure 56. Prediction results for the Exercise B applying the MLP neural network considering n, n-1, n-2 programming sequences for each log files in our dataset (1: positive performance, 0: negative performance)

Figure 55 and Figure 56 (and in detail Table 21 and Table 22) show that better performances are obtained using the dataset with n-1 and n-2 trials (considering programming sequences from t_0 to $t_{(n-1)}$ and programming sequences from t_0 to $t_{(n-s)}$). This is due to the fact that preparing these datasets for the machine learning procedure, we had to delete 3 groups for the Exercise A and 1 group for the Exercise B: the programming activity of these teams was characterised by only 2 trials, so having designed such a low number of programming sequences we could not use their log data for this prediction analysis. Analysing the improvement in the prediction performances for the n-1 and n-2 datasets, we can hypothesize that these 4 groups are outliers.

Table 21. MLP neural network detailed performance (considering n, n-1, n-2 programming sequences in each log file) for Ex. A (1: positive performance, 0: negative performance)

ML algorithm	Accuracy	0 – Precision	1 – Precision	0 – Recall	1 – Recall	0 – F1 Score	1 – F1 Score
MLP Neural Network (n)	0,94 (0,04)	0,85 (0,03)	0,95 (0,01)	0,85 (0,02)	0,97 (0,02)	0,84 (0,02)	0,96 (0,01)
MLP Neural Network (n-1)	0,95 (0,02)	0,92 (0,03)	0,97 (0,01)	0,91 (0,03)	0,97 (0,01)	0,91 (0,02)	0,97 (0,01)
MLP Neural Network (n-2)	0,95 (0,02)	0,92 (0,03)	0,97 (0,01)	0,92 (0,03)	0,97 (0,01)	0,91 (0,02)	0,97 (0,01)

Table 22. MLP neural network detailed performance (considering n, n-1, n-2 programming sequences in each log file) for Ex. B (1: positive performance, 0: negative performance)

ML algorithm	Acc.	0 – Precision	1 – Precision	0 – Recall	1 – Recall	0 – F1 Score	1 – F1 Score
MLP Neural Network (n)	0,97 (0,01)	0,96 (0,02)	0,98 (0,01)	0,95 (0,02)	0,98 (0,01)	0,95 (0,02)	0,98 (0,01)
MLP Neural Network (n-1)	0,98 (0,01)	0,96 (0,02)	0,98 (0,01)	0,95 (0,02)	0,99 (0,01)	0,95 (0,02)	0,98 (0,01)
MLP Neural Network (n-2)	0,99 (0,01)	1,00 (0,01)	0,99 (0,00)	0,97 (0,00)	1,00 (0,00)	0,98 (0,00)	0,99 (0,00)

Finally, Figure 57, Figure 58, Figure 59, Figure 60, Figure 61 and Figure 62 show the ROC curve (Huang and Ling, 2005) for Exercise A and B considering n trials, n-1 trials and n-2 trials. As the following graphs show, the MLP neural network trained in our research project is an excellent classifier, considering the problem faced in this experimentation.

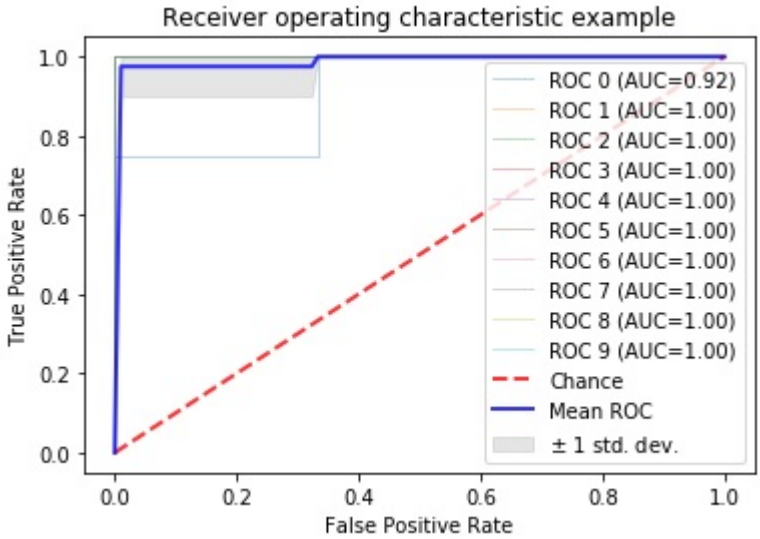

Figure 57. MLP neural network ROC curve for the Ex. A (considering all the n trials for each students' team in the experimentation)

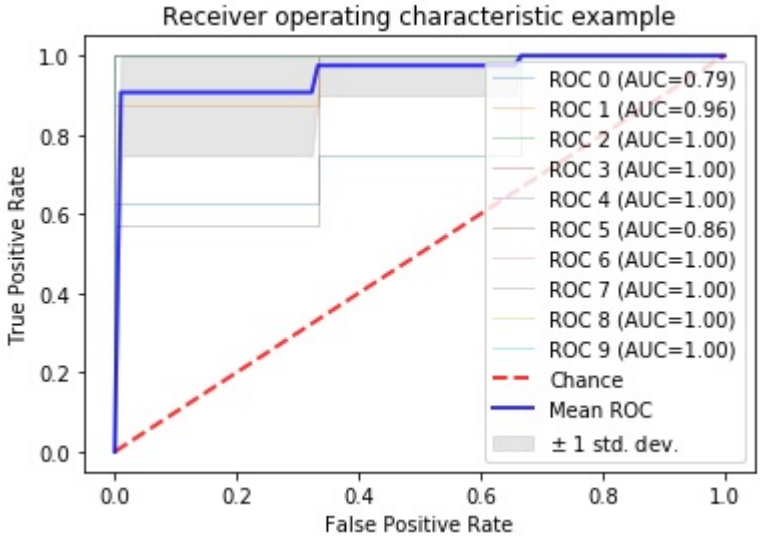

Figure 58. MLP neural network ROC curve for the Ex. A (considering n-1 trials for each students' team in the experimentation)

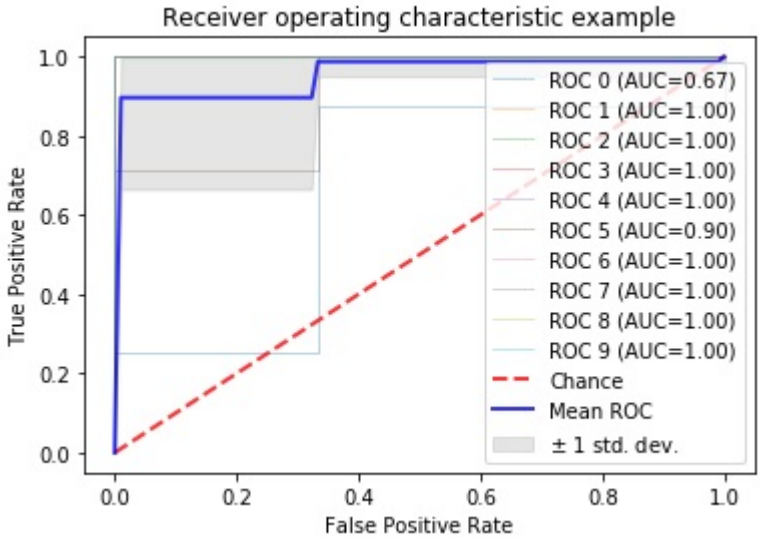

Figure 59. MLP neural network ROC curve for the Ex. A (considering n-2 trials for each students' team in the experimentation)

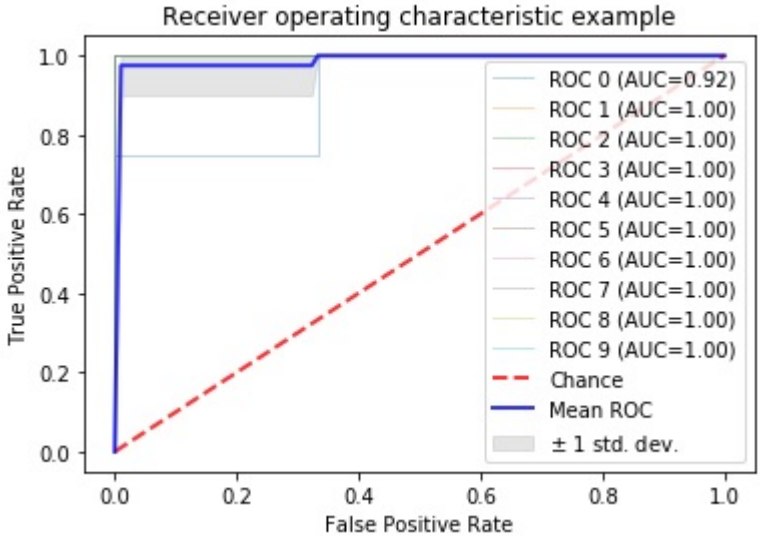

Figure 60. MLP neural network ROC curve for the Ex. B (considering all the n trials for each students' team in the experimentation)

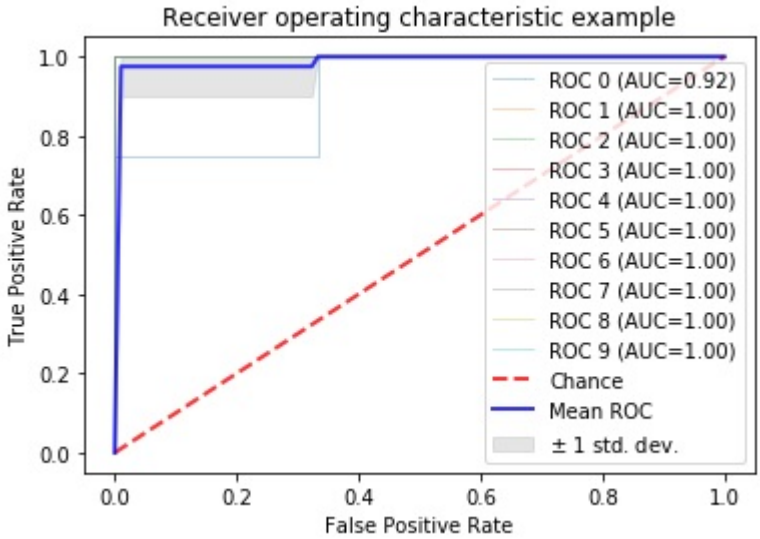

Figure 61. MLP neural network ROC curve for the Ex. B (considering n-1 trials for each students' team in the experimentation)

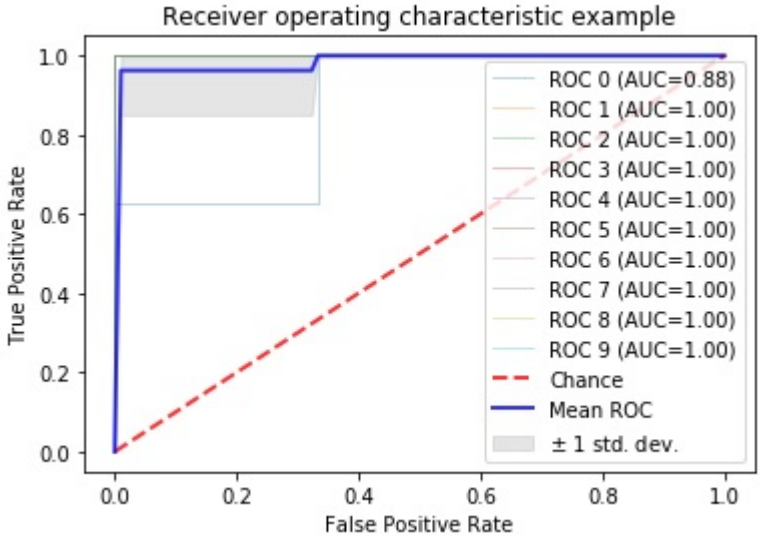

Figure 62. MLP neural network ROC curve for the Ex. B (considering n-2 trials for each students' team in the experimentation)

In summary, in this chapter we performed the following steps:

1. With a separate machine learning analysis we demonstrated that younger students and older students solved the Exercise A and Exercise B with similar strategies: section 3.2.1 and section 3.3.1 show similar prediction performances (SVM with mixed approach outperformed the other techniques for both the sub-datasets) and similar programming clusters for Exercise A; we could not implement a comparison of the prediction performances of the Exercise B because the "not completed" results were mainly concentrated in the younger students' subset; we analysed this challenge realizing a cluster analysis (k-means algorithm), and section 3.2.2 and 3.3.2 show how similar programming clusters emerged from both the sub-datasets.
2. After having demonstrated similar behaviours for younger and older students, we aggregated the dataset and performed the machine learning analysis with the whole sample. We obtained similar clusters for Exercise A and B also applying the k-means algorithm with this larger dataset, but the accuracy, the mean precision, the mean recall and the mean F1 score values for Exercise A were better analysing the complete dataset instead of considering the two sub-datasets (younger and older students); the SVM algorithm with mixed approach resulted the best solution at this stage of the analysis. For Exercise B we do not have a term of comparison for the prediction performances, but we achieved similar results to the Exercise A with the same technique (SVM with mixed approach).
3. Having available a larger dataset, we decided to apply another powerful machine learning technique, the MLP neural network algorithm (in combination with the mixed approach, the best solution emerged from the previous steps). This methodology outperformed all the previous algorithms: we obtained accuracy, precision, recall and F1 score even higher than 0.90 both for Exercise A and B.
4. The last step was to test the MLP neural network in a "simulation" of a real-time situation: we eliminated first the last programming sequence from the log files (the last trial designed by the students' team), and then the last two programming sequences. The MLP neural network has proven to be an excellent classifier also for these tasks.

In the next chapter we will go in depth in the educational implications of these results.

Chapter 4.

4 Concluding Remarks

This dissertation addresses the application of data mining and machine learning techniques for the evaluation of students' performance during educational robotics activities. Obtaining real time information from a learning system is of paramount importance for the teachers to react during ER activities. The technology, the data and the information obtained with a studied and designed tracking system has been presented and specific patterns along with specific features have been identified. These features characterized the problem-solving trajectories performed by students' teams involved in the experimentation and have shown the strong correlation with their performances in the exercises' resolution.

More specifically, three students' problem-solving styles are predominantly emerged during the research project: the "Mathematical/planning" style, the "Tinkering with prevalent refining behaviour" style and the "Tinkering with significantly high changes". Groups that adopted a "Mathematical / planning" style have an approach that utilises a well-defined mathematical strategy (in the case of Exercise A for example it could be the circumference formula, in the case of Exercise B it could be a flowchart that corresponds a priori to the robot's behaviour).
The "Mathematical / planning" style determines a very low numbers of changes to the programming sequence, a clear prevalence of robot's behaviour verification tests (without modification compared to the previous sequence) and usually a low number of total tests during the amount of time that students had to design their solution. This strategy is not necessarily a symptom of a positive performance, since learners could incorrectly analyse the problem, select a wrong formula, and/or introduce some errors in the calculations. This approach is close to the "planner scientist" style initially proposed by Turkle and Papert (1992).
Teams with a "Tinkering with prevalent refining behaviour" style have a heuristic approach. The groups start their problem-solving process from a practical experience, observing the robot feedback (in the case of Exercise A they set a specific parameter to turn on the motors and compare the robot's behaviour with a reference measure; in the case of Exercise B they could use the Conditional blocks using the default parameters to verify how the robot reacts) and then refine the blocks parameters implementing small changes, trying to incrementally move towards their objective. Students that prefer this approach mainly base their work on the analysis of the robot's feedback: examining the robot performing a program allows them to decide the next parameter changes; consequently, a careful observation is a crucial step to reach a positive result for the exercise.
"Tinkering with significantly high changes" groups show a similar approach to the one just presented, but they have a certain percentage of high parameters modifications between two contiguous sequences compared to the "Tinkering with prevalent refining behaviour" learners: this behaviour could indicate a difficulty in the interpretation of robot's feedback (they feel far from reaching the goal, so strongly modify the sequence), and therefore a

greater propensity to perform tests multiple times, since the students that adopt this style of working have a number of total trials higher than the mean value calculated for the other teams. Both these Tinkering styles are close to the "bricoleur scientist" proposed by Turkle and Papert (1992).
Analysing the performances of five machine learning algorithms (logistic regression, KNN, SVM, random forest, MLP neural network) in the prediction of the students' teams' results, for Exercises A and B, the MLP neural network used in combination with the features calculated by the k-means algorithm and the indicators that represents the learners' current activity (what we defined the "mixed approach", see Figure 18 and Figure 19 in chapter 2) outperformed the other techniques. The same MLP neural network shows excellent results also in a simulation of a real prediction, deleting the last programming sequences from the log files generated by the tracking system.
Classifying the different types of sequences designed by learners and calculating the percentage of them in the problem-solving trajectory performed by teams, instead of using the mean and standard deviation values of the 13 indicators presented in the Methodology chapter allows to obtain a better representation of the students' activity, with prediction results.
We consider very important from a pedagogical point of view the recall indicator for the students' groups who showed a negative performance. Recognizing in advance those teams with difficulties in the exercise resolution could allow teachers to give them some suggestions to solve the challenge; the MLP neural network algorithm reached:

- a 0,85 value of this indicator for Exercise A (considering all (n) the programming sequences in the log file);
- a 0,91 value of this indicator for Exercise A (considering n-1 programming sequences in the log file);
- a 0,92 value of this indicator for Exercise A (considering n-2 programming sequences in the log file);
- a 0,95 value of this indicator for Exercise B (considering all (n) the programming sequences in the log file);
- a 0,95 value of this indicator for Exercise B (considering n-1 programming sequences in the log file);
- a 0,97 value of this indicator for Exercise B (considering n-2 programming sequences in the log file).

We are quite close to identify each students' group that is struggling with the Educational Robotics challenge, and this result is very important for further developments of the system, implementing a real-time use of the tool by teachers and educators.
The results presented in this dissertation seem to show connections with previous research. On the one hand, Blikstein et al. (2014) identified a specific cluster for students with a "steadier incremental steps" strategy of programming, similar to the "Tinkering with prevalent refining behaviour", and stated that students in that cluster obtained better performance in the resolution of the exercise and in the final grade. On the other hand, Chao (2016) observed that students in a cluster named "Trial and Error" (with the greater number of trials to solve the exercise) achieved the worst performance in terms of exercise score, showing agreement with the results in the current PhD thesis, in which the number of trials has a negative correlation with the final students result. This finding is opposed to

Filvà et al. (2019): their analysis showed a positive correlation between projects with rapid iterations of trial and error and good results; but they considered a different kind of exercise, more creative and unstructured compared to the one analysed by Chao (2016) and to the two challenges considered in this dissertation. This particular aspect could be further investigated in future works.
Moreover, Dong, Marwan, Catete, Prince and Barnes (2019) recognized three different subcategories related to tinkering in their experimentation (test-based tinkering, prototype-based tinkering and construction-based tinkering). The test based tinkering category could be associated with the "Tinkering with prevalent refining behaviour" and "Tinkering with significantly high changes" behaviours of the present dissertation, since the analysed exercises are introduction to Robotics, and don't allow students to design more complex features or artefact; analysing more creative challenges could be interesting to evaluate if new styles appear similar to those identified by Dong et al. (2019).
This dissertation has presented a number of innovative elements. To the best of the authors' knowledge, for the first time an experimentation involving a fair number of students (455) has been conducted gathering programming sequences designed by students during Educational Robotics activities and automatically analysing them, using machine learning techniques. In this analysis, the authors used the programming sequences to calculate 13 indicators and detect patterns in the students' problem-solving activity, studying the relations between these patterns and the performances obtained by learners. Valid data were obtained from 105 teams for Exercise A and from 108 teams from Exercise B, with a total of 455 students. Although it's not sufficient to obtain a large dataset (large data results in better generalization), since Educational Robotics is an approach characterized by teamwork (3-4 students work together during the resolution of a challenge, integrating their different competencies), the promising results presented in this PhD thesis has encouraged our research group to involve new classes in the experimentation, in order to continue the validation of the approach. Increasing the sample will allow the identification of a greater number of problem-solving styles (not only planners vs. tinkerers) and the analysis of the combination of these styles within different teams that might adopt a mixed strategy.
Future work of this dissertation includes further development of the exercises' specification within the analysis (in the Exercise A students had to cover a given distance with the robot and in Exercise B they had to stop the robot at a given distance in front of the wall). In order to obtain more general results, the approach described in this thesis can be applied to a larger set of challenges, considering all the motors and sensors included in the Lego Mindstorms EV3 kit. Moreover, further creative exercises can be analysed, where the educator doesn't define precise challenge specifications, but he/she proposes only a general context that learners have to consider. Within this context, each team would decide autonomously the specifications of its robotic artefact and design programming sequences to reach the predetermined objectives.
A planned improvement of this research study is a deep analysis about how many programming sequences can be deleted from the log files maintaining good performances in terms of prediction, or in other words how much is possible to anticipate the prediction of a positive or a negative students' result, in an Educational Robotics activity. We demonstrated that eliminating two trials realised by students' teams from our dataset doesn't compromise prediction performance, but this aspect of the research project probably deserves more study.

Another improvement for future development of this study will be time tracking in the log files generated by the system in order to consider not only the programming sequences designed by students' teams, but also how they are distributed on the temporal axis within specific time intervals defined from the educator. Authors intend also to utilise recurrent neural network, in particular the long short-term memory (Hochreiter & Schmidhuber, 1997) autoencoders (a structure specifically designed to support sequences of input data), in order to translate the programming sequences created by students into fixed-length vectors (compress representation of the input data), maintaining high level of information content. As a result, these vectors obtained from the autoencoders compression will be used as input features for supervised and/or unsupervised algorithms.
Collection of complementary data, recordings of all the actions performed by students on the laptop during the programming sequence design (for example capturing the pointing device movements) could add even further useful information to the log file recorded. Another planned development is the update of the current system design with a personalised e-learning system: an educational recommender system could give real-time feedback to teachers and students involved in Educational Robotics activities, or propose personalised learning path to learners, similar to the programming tutoring system designed by Klašnja-Milićević, Vesin, Ivanović, and Budimac (2011).

The approach presented in this research was the initial step for the implementation of new evaluation strategies connected to Educational Robotics activities, based on combination of the strengths of both quantitative and qualitative methods. It's important to underline that the educator remains a key figure in the educational process, even by adopting machine learning techniques: he/she doesn't only collect complementary data with respect to the tracking system, but has a fundamental role in the validation of these new assessment strategies, observing the real behaviour of the students and verifying the veracity of the results obtained using data mining and machine learning methods. New large-scale experimentations will permit the educational experts to analyse more in depth the students' problem-solving and design process, and will allow a profound understanding of the Educational Robotics' benefits within pedagogical praxis.
The main research objective of this dissertation was the preliminary design of an approach that, with further developments and improvements, can help teachers to evaluate in depth Educational Robotics activities, to identify students with more difficulties and to determine different learners' problem-solving styles (so that personalised feedbacks and suggestions can be given). In the next future it would be very interesting to propose data insights also to students, in order to allow them a deep reflection about how they solved a problem, how they design a programming sequence, how they work during the Educational Robotics activity: this would allow to realise a true metacognition, putting into practice the visionary words of Seymour Papert (1972):
"I believe with Dewey, Montessori, and Piaget that children learn by doing and by thinking about what they do. And so the fundamental ingredients of educational innovation must be better things to do and **better ways to think about oneself doing these things**".

References

Ahmed I., Lubold N., and Walker E. (2018) ROBIN: Using a Programmable Robot to Provide Feedback and Encouragement on Programming Tasks. In: Penstein Rosé C. et al. (eds) Artificial Intelligence in Education. AIED 2018. Lecture Notes in Computer Science, vol 10948. Springer, Cham.

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. Themes in Science and Technology Education, 6(1), 63-71.

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. International journal of child-computer interaction, 19, 30-55.

Angel-Fernandez, J. M., & Vincze, M. (2018) Towards a Formal Definition of Educational Robotics. In Philipp Zech, Justus Piater Eds., Proceedings of the Austrian Robotics Workshop 2018. Conference series, Innsbruck university press. doi: 10.15203/3187-22-1

Angeli, C., & Valanides, N. (2019, in press). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. Computers in Human Behavior. https://doi.org/10.1016/j.chb.2019.03.018

Asif, R., Merceron, A., Ali, S. A., & Haider, N. G. (2017). Analyzing undergraduate students' performance using educational data mining. Computers & Education, 113 (Supplement C), 177-194.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661-670.

Baker, E. L., & Mayer, R. E. (1999). Computer-based assessment of problem solving. Computers in human behavior, 15(3-4), 269-282.

Barker, B. S., Nugent, G., Grandgenett, N., & Adamchuk, V.I. (Ed.). (2012). Robots in K-12 education: A new technology for learning: A new technology for learning. IGI Global.

Beck, J. E., & Woolf, B. P. (2000). High-level student modeling with machine learning. In Proceedings of Fifth International Conference on Intelligent Tutoring Systems, Alagoas, Brazil (pp. 584–593).

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. Computers & Education, 58(3), 978-988.

Berland, M., Baker, R. S., & Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. Technology, Knowledge and Learning, 19(1-2), 205-220.

Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. Journal of the Learning Sciences, 22(4), 564-599.

Bey A., Pérez-Sanagustín M., and Broisin J. (2019) Unsupervised Automatic Detection of Learners' Programming Behavior. In: Scheffel M., Broisin J., Pammer-Schindler V., Ioannou A., Schneider J. (eds) Transforming Learning with Meaningful Technologies. EC-TEL 2019. Lecture Notes in Computer Science, vol 11722. Springer, Cham.

Blikstein, P., & Worsley, M. (2016). Multimodal Learning Analytics and Education Data Mining: using computational technologies to measure complex learning tasks. Journal of Learning Analytics, 3(2), 220-238.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. Journal of the Learning Sciences, 23(4), 561-599.

Bock, H. H. (2007), Clustering Methods: A History of k-Means Algorithms, in P. Brito, P. Bertrand, G. Cucumel, & F. De Carvalho (Eds.), Selected Contributions in Data Analysis and Classification (pp. 161–172). Heidelberg: Springer Verlag.

Bottge, B. A., Rueda, E., Kwon, J. M., Grant, T., & LaRoque, P. (2009). Assessing and tracking students' problem solving performances in anchored learning environments. Educational Technology Research and Development, 57(4), 529-552.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In Annual American Educational Research Association meeting, Vancouver, BC, Canada.

Castro, E., Cecchi, F., Valente, M., Buselli, E., Salvini, P., & Dario, P. (2018). Can educational robotics introduce young children to robotics and how can we measure it?. Journal of Computer Assisted Learning, 34(6), 970-977.

Cesaretti, L., Storti, M., Mazzieri, E., Screpanti, L., Paesani, A., & Scaradozzi, D. (2017). An innovative approach to School-Work turnover programme with Educational Robotics. Mondo Digitale, 16(72), 2017-5.

Chalmers, C. (2018). Robotics and computational thinking in primary school. International Journal of Child-Computer Interaction, 17, 93-100.

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers & Education, 95, 202-215.

Cheng, Y. W., Sun, P. C., & Chen, N. S. (2018). The essential applications of educational robot: Requirement analysis from the perspectives of experts, researchers and instructors. Computers & education, 126, 399-416.

Cho, H. K. (2011, October). On the use of robot-enhanced learning activities across formal and informal K-12 curricula. In 2011 11th International Conference on Control, Automation and Systems (pp. 1059-1063). IEEE.

Collins, M., Schapire, R. E., & Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. Machine Learning, 48(1-3), 253-285.

Cooper, M. M., Cox Jr, C. T., Nammouz, M., Case, E., & Stevens, R. (2008). An assessment of the effect of collaborative groups on students' problem-solving strategies and abilities. Journal of Chemical Education, 85(6), 866.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.

Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to Support Vector Machines. Cambridge, UK: Cambridge University Press.

Cristoforis, P. D., Pedre, S., Nitsche, M., Fischer, T., Pessacg, F., & Di Pietro, C. (2013). A behavior-based approach for educational robotics activities. IEEE Transactions on Education, 56(1), 61-66.

Csikszentmihalyi, M. (1997). Finding flow: The psychology of engagement with everyday life. Basic Books.

Deek, F. P., Hiltz, S. R., Kimmel, H., & Rotter, N. (1999). Cognitive assessment of students' problem solving and program development skills. Journal of Engineering Education, 88(3), 317-326.

Denis, B., & Hubert, S. (2001). Collaborative learning in an educational robotics environment. Computers in Human Behavior, 17(5-6), 465-480.
Dong, Y., Marwan, S., Catete, V., Price, T., & Barnes, T. (2019, February). Defining Tinkering Behavior in Open-ended Block-based Programming Assignments. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 1204-1210). Minneapolis, USA.
Dutt, A., Ismail, M. A., & Herawan, T. (2017). A systematic review on educational data mining. IEEE Access, 5, 15991-16005.
Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. Journal of Information Technology Education: Innovations in Practice, 13, 153-169.
Fernandes, E., Holanda, M., Victorino, M., Borges, V., Carvalho, R., & Van Erven, G. (2019). Educational data mining: Predictive analysis of academic performance of public school students in the capital of Brazil. Journal of Business Research, 94, 335-343.
Ferrarelli, P., Villa, W., Attolini, M., Cesareni, D., Micale, F., Sansone, N., Pantalone, L.C., & Iocchi, L. (2018, April). Improving Students' Concepts About Newtonian Mechanics Using Mobile Robots. In International Conference on Robotics and Education (RiE) 2017 (pp. 113-124). Springer, Cham.
Filvà, D. A., Forment, M. A., García-Peñalvo, F. J., Escudero, D. F., and Casañ, M. J. (2019). Clickstream for learning analytics to assess students' behavior with Scratch. Future Generation Computer Systems, 93, 673-686.
Greiff, S., Holt, D. V., & Funke, J. (2013). Perspectives on problem solving in educational assessment: Analytical, interactive, and collaborative problem solving. Journal of Problem Solving, 5(2).
Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational researcher, 42(1), 38-43.
Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd International Conference on Document Analysis and Recognition (Vol. 1, pp. 278-282). IEEE, Montreal, Canada.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
Hosmer, D. W., & Lemeshow, S. (2000). Applied Logistic Regression, 2nd ed. New York: Wiley.
Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering, 17(3), 299-310.
In Summer we learn STEM - In Estate si imparano le STEM (2019). Retrieved from https://www.campustore.it/in-estate-si-imparano-le-stem.html
Introducing the Beyond Rubrics Toolkit (2019), Maker assessment tools designed by MIT Playful Journey Lab and MakerEd. Retrieved from https://makered.org/beyondrubrics/partners/
Jain, A. K., & Chandrasekaran, B. (1982). 39 Dimensionality and sample size considerations in pattern recognition practice. Handbook of statistics, 2, 835-855.
Jeon, M., FakhrHosseini, M., Barnes, J., Duford, Z., Zhang, R., Ryan, J., & Vasey, E. (2016, March). Making Live Theatre with Multiple Robots as Actors: Bringing Robots to Rural Schools to Promote STEAM Education for Underserved Students. In The Eleventh

ACM/IEEE International Conference on Human Robot Interaction, (pp. 445-446), doi: 10.1109/HRI.2016.7451798.
Jormanainen, I., & Sutinen, E. (2012). Using data mining to support teacher's intervention in a robotics class., Proceedings of the 2012 IEEE Fourth International Conference on Digital Game and Intelligent Toy Enhanced Learning (pp. 39-46). Washington, DC: IEEE.
Kandlhofer, M., & Steinbauer, G. (2016). Evaluating the impact of educational robotics on pupils' technical-and social-skills and science related attitudes. Robotics and Autonomous Systems, 75, 679-685.
Kantardzic, M. (2011). Data Mining: Concepts, Models, Methods, and Algorithms. John Wiley & Sons, Hoboken, New Jersey.
Kim, J. H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational statistics & data analysis, 53(11), 3735-3745.
Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. Computers & Education, 91, 14-31.
Klašnja-Milićević, A., Vesin, B., Ivanović, M., & Budimac, Z. (2011). E-Learning personalization based on hybrid recommendation strategy and learning style identification. Computers & Education, 56(3), 885-899.
Kodinariya, T. M., & Makwana, P. R. (2013). Review on determining number of Cluster in K-Means Clustering. International Journal of Advance Research in Computer Science and Management Studies, 1(6), 90-95.
LEGO 4C Framework (2019), Retrieved from:
https://le-www-live-s.legocdn.com/sc/media/files/marketing-tools/lego-education-manifesto-d218aa7fac50c89c1b307b8f1ab94b16.pdf
Lego Mindstorms EV3 Developer kit. (2019), Lego EV3 Mindstorms Software Developer kit official website. Retrieved from https://education.lego.com/en-au/support/mindstorms-ev3/developer-kits
Lego Mindstorms EV3 Education kit. (2019), Lego EV3 Mindstorms Education official website. Retrieved from https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set-/5003400
Lego Mindstorms EV3 Education software. (2019), Lego EV3 Mindstorms Education Software official website. Retrieved from https://education.lego.com/en-us/downloads/mindstorms-ev3/software
Lego Robotics History (2019), Lego Mindstorms History official website. Retrieved from https://www.lego.com/en-us/mindstorms/history
Liu, E. Z. F. (2010). Early adolescents' perceptions of educational robots and learning of robotics. British Journal of Educational Technology, 41(3), E44-E47.
Luckin, R. (2017). The Implications of Artificial Intelligence for Teachers and Schooling. In L. Loble, T. Creenaune, & J. Hayes (Eds.), Future frontiers: education for an AI world,109. Melbourne University Press & New South Wales Department of Education.109-125.
Martinez, S. L., & Stager, G. (2013). Invent to learn: Making, tinkering, and engineering in the classroom. Torrance, CA: Constructing modern knowledge press.
Merceron, A., & Yacef, K. (2004). Mining student data captured from a web-based tutoring tool: Initial exploration and results. Journal of Interactive Learning Research, 15(4), 319–346.

Mikropoulos, T. A., & Bellou, I. (2013). Educational robotics as mindtools. Themes in Science and Technology Education, 6(1), 5-14.
Miller, D. P., & Nourbakhsh, I. (2016). Robotics for education. In B. Siciliano, & O. Khatib, Springer Handbook of Robotics (pp. 2115-2134). Springer, Cham.
Montero, C. S., & Jormanainen, I. (2016, November). Theater Meets Robot–Toward Inclusive STEAM Education. In International Conference EduRobotics 2016 (pp. 34-40). Springer, Cham.
National Plan for Digital School, PNSD. Retrieved from https://www.miur.gov.it/scuola-digitale
OECD (2004), The PISA 2003 Assessment Framework: Mathematics, Reading, Science and Problem Solving Knowledge and Skills, PISA, OECD Publishing, Paris, https://doi.org/10.1787/9789264101739-en.
Ornelas, F., & Ordonez, C. (2017). Predicting Student Success: A Naïve Bayesian Application to Community College Data. Technology, Knowledge and Learning, 22(3), 299-315.
Papert, S. (1972). Teaching children thinking. Programmed Learning and Educational Technology, 9(5), 245-255.
Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books.
Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), Constructionism (pp. 1–11). Norwood, NJ: Ablex Publishing.
Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M. Guibas, L. J. & Sohl-Dickstein, J. (2015) Deep knowledge tracing. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. and R. Garnett, editors, Advances in Neural Information Processing Systems 28, 505–513. Curran Associates, Inc.
PON Digital Creativity - PON Creatività Digitale (2019). Retrieved from https://www.istruzione.it/pon//avviso_cittadinanza-creativita.html
Rabiner, L. R., & Juang, B. H. (1986). An introduction to Hidden Markov Models. ASSP magazine, IEEE, 3(1), 4-16.
Resnick, M. (2017). Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play. MIT Press. Cambridge MA.
Resnick, M., Martin, F., Sargent, R., & Silverman, B. (1996). Programmable Bricks: Toys to think with. IBM Systems journal, 35(3.4), 443-452.
Resnick, M., Ocko, S., & Papert, S. (1988). “Lego, LOGO, and Design.” Children's Environments Quarterly, 5(4), New York: Children's Environments Research Group, The City University of New York.
Resnick, M., & Rosenbaum, E. (2013). Designing for tinkerability. In M. Honey & D. Kanter (Eds.), Design, make, play: Growing the next generation of STEM innovators (pp. 163–181). New York: Routledge.
Sahin, A., Ayar, M. C., & Adiguzel, T. (2014). STEM Related After-School Program Activities and Associated Outcomes on Student Learning. Educational Sciences: Theory and Practice, 14(1), 309-322.
Scaradozzi, D., Cesaretti, L., Screpanti, L., Costa, D., Zingaretti, S., & Valzano, M. (2019). “Innovative tools for teaching Marine Robotics, IoT and Control Strategies since the Primary school”. In Daniela, L. (ed.), Smart Learning with Educational Robotics - Using Robots to Scaffold Learning Outcomes. Springer. DOI: 10.1007/978-3-030-19913-5, https://www.springer.com/gp/book/9783030199128.

Scaradozzi, D., Screpanti, L., Cesaretti, L., Storti, M., & Mazzieri, E. (2019). Implementation and assessment methodologies of teachers' training courses for STEM activities. Technology, Knowledge and Learning, 24(2), 247-268.

Scaradozzi, D., Screpanti, L., & Cesaretti, L. (2019). Towards a definition of educational robotics: a classification of tools, experiences and assessments. In Daniela, L. (ed.), Smart Learning with Educational Robotics (pp. 63-92). Springer, Cham. DOI: 10.1007/978-3-030-19913-5, https://www.springer.com/gp/book/9783030199128.

Scaradozzi, D., Sorbi, L., Pedale, A., Valzano, M., & Vergine, C. (2015). Teaching robotics at the primary school: an innovative approach. Procedia-Social and Behavioral Sciences, 174, 3838-3846.

Screpanti, L., Cesaretti, L., Storti, M., Mazzieri, E., Longhi, A., Brandoni, M., & Scaradozzi, D. (2018). Advancing K12 education through Educational Robotics to shape the citizens of the future. In Proceedings of DIDAMATICA 2018 (pp. 117-126). AICA. Cesena, Italy.

Screpanti, L., Cesaretti, L., Marchetti, L., Baione, A., & Scaradozzi, D. (2018, July). An Educational Robotics activity to promote gender equality in STEM education. In Proceedings of the eighteenth International Conference on Information, Communication Technologies in Education (ICICTE 2018) (pp. 336-346). Chania, Crete, Greece.

Shute, V. J., Wang, L., Greiff, S., Zhao, W., & Moore, G. (2016). Measuring problem solving skills via stealth assessment in an engaging video game. Computers in Human Behavior, 63, 106-117.

Smyth, P. (1997). Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), Advances in neural information processing systems 9: Proceedings of the 1996 conference (pp. 648–654). Cambridge, MA: MIT Press.

Steinley, D. (2006). K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology, 59(1), 1-34.

Sullivan, F. R. (2008). Robotics and science literacy: Thinking skills, science process skills and systems understanding. Journal of Research in Science Teaching, 45(3), 373-394.

Tangdhanakanond, K., Pitiyanuwat, S., & Archwamety, T. (2006). A development of portfolio for learning assessment of students taught by full-scale constructionism approach at Darunsikkhalai School. Research in the Schools, 13(2), 24.

Turkle, S., & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. Journal of Mathematical Behavior, 11(1), 3-33.

Ucgul, M., & Cagiltay, K. (2014). Design and development issues for educational robotics training camps. International Journal of Technology and Design Education, 24(2), 203-222.

Wang, L., Sy, A., Liu, L., & Piech, C. (2017). Learning to Represent Student Knowledge on Programming Exercises Using Deep Learning. In Proceedings of the 10th International Conference on Educational Data Mining (EDM) (pp. 324–329). Wuhan, China.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 58(301), 236-244.

Winne, P. H., & Baker, R. S. (2013). The potentials of educational data mining for researching metacognition, motivation and self-regulated learning. JEDM| Journal of Educational Data Mining, 5(1), 1-8.

Appendix A.

The tracking system: technical details

A.1. Data collection

We modified Lego Mindstorms EV3 original blocks in order to write (as a string) the programming sequence created by the students on a log file each time the students upload and run a program on the EV3 brick. Figure A 2 shows a very simple example of the log file, generated by the execution of the sequence in Figure A 1:

Figure A 1. A programming sequence example designed in Lego Mindstorms EV3 software

```
START
Name: Program
Time: 2350 sec

MoveSteering, OnForSeconds, Steering = 0, Speed = 100, Seconds = 10, MotorPorts = 123;
MoveSteering, OnForRotations, Steering = 50, Speed = 100, Rotations = 1, MotorPorts = 123;
MoveSteering, OnForseconds, Steering = 0,  Speed = 100, Seconds = 5, MotorPorts = 123;
STOP PROGRAM;

STOP
Name: Program
Time: 2367 sec
```

Figure A 2. Log File generated by the execution of the sequence in Fig. A.1

The line with the string "START" indicates the starting point of the log file. "Program name" indicates the name of the program assigned by the students (in this case, "Program"). The line with the string "Time" contains the number of seconds elapsed since the robot is turned on (2350 s in the example). Then there are the lines representing the three blocks in the sequence (in this example, 3 Move Steering blocks, as Figure A 2 shows); the start block is not recorded by the authors' system. After that, there are three lines indicating the end of the program: it's easy to calculate the duration of the execution simply subtracting the "start time" to the "stop time".
The lines representing the blocks in the sequence have always the same structure:

1st block feature: it is the name of the block selected by the students in the Lego Mindstorms Education Software; each block performs a different functionality. In the example the name of the block is MoveSteering.

2nd block feature: it is the block option chosen by the students; each block has different options: for example, in the Move Steering block it's possible to select 5 different options (see Figure 9 in "Methodology" chapter): On for Seconds, On for Degrees, On for Rotations, On, Off.

1st block parameter: it is the first parameter in the block, that students can set as they want; it's meaning depends on the block and on the option chosen by the students. In this case we have the robot's steering as the first parameter of the Move Steering block, with On for Seconds option.

2nd block parameter: it is the second parameter in the block, that students can set as they want; it's meaning depends on the block and on the option chosen by the students. In this case we have the motors' speed as the second parameter of the Move Steering block, with On for Seconds option.

3rd block parameter: it is the third parameter in the block, that students can set as they want; it's meaning depends on the block and on the option chosen by the students. In this case we have the number of seconds of motors' rotation at a given speed as the third parameter of the Move Steering block, with On for Seconds option.

4th block parameter: it is the fourth parameter in the block, that students can set as they want; it usually represents the brick ports where the students have connected motors or sensors.

Table A 1 contains the most commonly used block names, out of 43 with their specific options.

Table A 1. Examples of LEGO EV3 block names and options.

BLOCK NAME	BLOCK OPTIONS
Large Motor	OnForRotations, OnForSeconds, OnForDegrees, On, Off
Medium Motor	OnForRotations, OnForSeconds, OnForDegrees, On, Off
Move Steering	OnForRotations, OnForSeconds, OnForDegrees, On, Off
Move Tank	OnForRotations, OnForSeconds, OnForDegrees, On, Off

Display	Image, Text (Pixels, Grid), Shapes (Line, Circle, Rectangle, Point), Clear
Display	Clear
Sound	PlayFile, PlayTone, PlayNote, Stop
Brick Light	On, Off, Reset
Loop	Count, Unlimited, Time, Logic, BrickButtons, Color Sensor (Color, ReflLight, AmbLight), etc.

Figure A 3 shows the file log for a generic group N, that tries to solve the Exercise A: the team executes three tests using the Move Steering Block - On for Rotations, but in their first test rotations parameter is set to 1, while in the second and third test is set to 5.5.

```
START
Program name: Exercise1
Time: 1765 sec

MoveSteering, OnForRotations, Steering=0, Speed=50, Rotations=1, MotorPorts=123;
STOP PROGRAM;

STOP
Program name: Exercise1
Time: 1766 sec
```

Team N – First test

```
START
Program name: Exercise1
Time: 1842 sec

MoveSteering, OnForRotations, Steering=0, Speed=50, Rotations=5.5, MotorPorts=123
STOP PROGRAM;

STOP
Program name: Exercise1
Time: 1845 sec
```

Team N – Second test

```
START
Program name: Test
Time: 1902 sec

MoveSteering, OnForRotations, Steering=0, Speed=50, Rotations=5.5, MotorPorts=123
STOP PROGRAM;

STOP
Program name: Test
Time: 1905 sec
```

Team N – Third test

Figure A 3. An example of log file with 3 tests

A.2. Data preparation

Log files described in the previous section are not suitable for machine learning algorithms. The first transformation is needed to convert the text log files in numerical matrices. Authors, working in Python, translated each row in the log file in a row of a matrix. An example of this transformation is presented in the Table A 2: it shows the transformation of the log file in Figure A 3.

Table A 2. Example of log file transformation.

N° Test	1st Block Feature (numeric)	2nd Block Feature (numeric)	1st parameter (numeric)	2nd parameter (numeric)	3rd parameter (numeric)	Position of the block in the sequence
1	1	1	0	100	1	1
1	43	0	0	0	0	2
2	1	1	0	100	5.5	1
2	43	0	0	0	0	2
3	1	1	0	100	5.5	1
3	43	0	0	0	0	2

The first element in the row (n° of test), represents the number of the test the block belongs to. Considering the two elements 1st Block Feature and 2nd Block Feature, to perform this conversion, author created a mapping system that takes in input a string (the 1st block feature and the 2nd block feature presented in the previous subsection) and gives in output a numeric variable, as explained in Table A 3.

Table A 3. Mapping of 1st Block Feature string and 2nd Block Feature string into numeric value.

1st Block Feature (string)	1st Block Feature (numeric)	2nd Block Feature (string)	2nd Block Feature (numeric)	1st parameter	2nd parameter	3rd parameter
MoveSteering	1	OnForRotations	1	Robot steering	Motors speed	# of rotations

		OnForSeconds	2	Robot steering	Motors speed	# of seconds
		OnForDegrees	3	Robot steering	Motors speed	# of degrees
		On	4	Robot steering	Motors speed	0
		Off	5	0	0	0
MoveTank	2	OnForRotations	1	First Motor Speed	Second Motor Speed	# of rotations
		OnForSeconds	2	First Motor Speed	Second Motor Speed	# of seconds
		OnForDegrees	3	First Motor Speed	Second Motor Speed	# of degrees
		On	4	First Motor Speed	Second Motor Speed	0
		Off	5	0	0	0
LargeMotor	3	OnForRotations	1	Motor Speed	# of rotations	0
		OnForSeconds	2	Motor Speed	# of seconds	0
		OnForDegrees	3	Motor Speed	# of degrees	0
		On	4	Motor Speed	0	0
		Off	5	0	0	0
Loop	6	Start Loop Cycle	1	0	0	0

		End Loop Cycle, Count Mode	2	# of iterations	0	0
Stop Program	43	This block is not characterized by any option	0	0	0	0
…	…	...	…	…	…	…

The 1st parameter (numeric), 2nd parameter (numeric) and 3rd parameter (numeric) are the values contained in the three elements 1st block parameter (string), 2nd block parameter (string) and 3rd block parameter (string). Authors didn't consider the 4th parameter (string) because it is a standard value: it represents the brick ports to which students connected their motors and sensors, and learners don't change this value during their programming activity. The "Position of the block in the sequence" column represents the sequential order of the blocks within a certain programming sequence.

The second, fourth and sixth row of the Table A 2 represents the block STOP PROGRAM, that doesn't have any options and any parameters.

Each log file contains a different number of tests (hence a different number of sequences), because usually the educator gave a specific amount of time to solve the challenge, but not a limited number of tests, as explained in the previous section.

In the Figure A 4, two more complex programming sequences, containing the loop block, are presented; supposing that the students' team that designed these sequences, first runs the one on the top of the Figure A 4 and then the one on the bottom, the log file generated will be similar to Figure A 5.

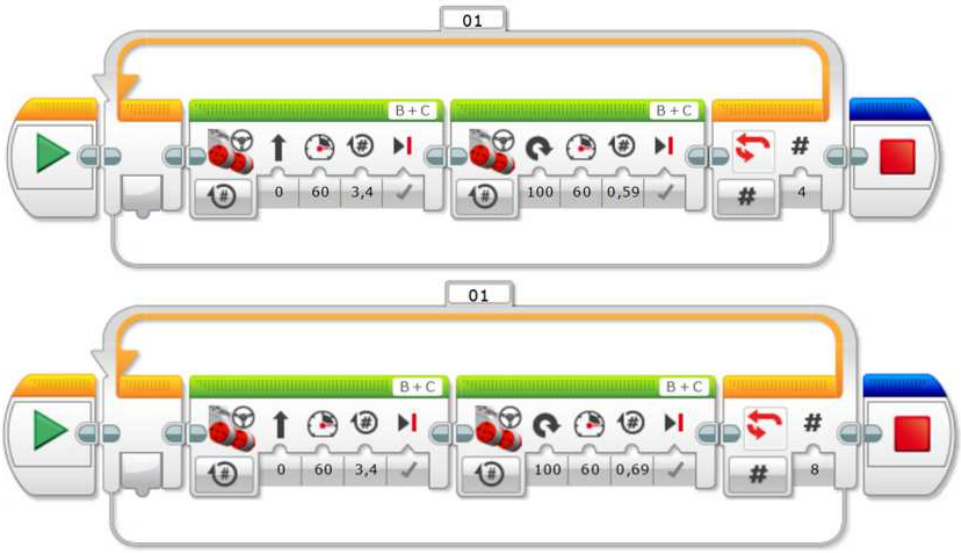

Figure A 4. Two example tests containing the loop block

```
START
Nome: Program
Tempo: 5539 sec

Start Loop Cycle;
MoveSteering, OnForRotations, Steering = 0, Speed = 60, Rotations = 3.4, MotorPorts = 123;
MoveSteering, OnForRotations, Steering = 100, Speed = 60, Rotations = 0.59, MotorPorts = 123;
EndLoopCycle, CountMode, Iterations To Run = 4;
STOP PROGRAM;

STOP
Nome: Program
Tempo: 5549 sec
```

Team N - first test

```
START
Nome: Program
Tempo: 5641 sec

Start Loop Cycle;
MoveSteering, OnForRotations,  Steering = 0, Speed = 60, Rotations = 3.4, MotorPorts = 123;
MoveSteering, OnForRotations, Steering = 100, Speed = 60, Rotations = 0.69, MotorPorts = 123;
EndLoopCycle, CountMode, Iterations To Run = 8;
STOP PROGRAM;

STOP
Nome: Program
Tempo: 5650 sec
```

Team N - second test

Figure A 5. The log file related to the Figure A.4

Table A 4 shows the results of an application of the above-mentioned procedure; where a block doesn't have a parameter, the value assigned is 0 (for example we have this situation with the loop block).

Table A 4. Transformation into a matrix of log file in Figure A.5

N° Test	1st Block Feature (numeric)	2nd Block Feature (numeric)	1st parameter (numeric)	2nd parameter (numeric)	3rd parameter (numeric)	Block Position in the sequence
1	6	1	0	0	0	1
1	1	1	0	60	3.4	2
1	1	1	100	60	0.59	3
1	6	2	4	0	0	4
1	43	0	0	0	0	5
2	6	1	0	0	0	1
2	1	1	0	60	3.4	2
2	1	1	100	60	0.69	3
2	6	2	8	0	0	4
2	43	0	0	0	0	5

Appendix B.

Machine learning algorithms technical details

This Appendix B presents some ML algorithms technical details, considering all the applied techniques (Logistic Regression, KNN, SVM, Random Forest, MLP Neural Network) and both the approaches (supervised approach and mixed approach).
The following tables show the parameters set using Python 3.5 and the scikit-learn library.

Table B 1. ML parameters for the supervised approach, Ex. A, younger students' teams

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty='l2', C = 0.1
KNN	KNeighborsClassifier	N° neighbors = 6
SVM	SVC	kernel='linear', C = 30
Random Forest		N° estimators = 10

Table B 2. ML parameters for the supervised approach, Ex. A, older students' teams

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty='l2', C = 10
KNN	KNeighborsClassifier	N° neighbors = 6
SVM	SVC	kernel='poly', Degree = 2, C = 100
Random Forest	RandomForestClassifier	N° estimators = 10

Table B 3. ML parameters for the supervised approach, Ex. A, whole dataset

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty='l2', C = 100
KNN	KNeighborsClassifier	N° neighbors = 6
SVM	SVC	kernel='linear', C = 20
Random Forest	RandomForestClassifier	N° estimators = 10

Table B 4. ML parameters for the supervised approach, Ex. B, whole dataset

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty='l2', C = 1
KNN	KNeighborsClassifier	N° neighbors = 4
SVM	SVC	kernel='poly', Degree = 2 C = 200
Random Forest	RandomForestClassifier	N° estimators = 10

Table B 5. ML parameters for the mixed approach, Ex. A, younger students' teams

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty = 'l2', C = 10
KNN	KNeighborsClassifier	n° neighbors = 4
SVM	SVC	kernel='rbf', C = 7
Random Forest	RandomForestClassifier	n° estimators=10

Table B 6. ML parameters for the mixed approach, Ex. A, older students' teams

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty = 'l2', C = 100
KNN	KNeighborsClassifier	n° neighbors = 4
SVM	SVC	kernel='linear', C = 1
Random Forest	RandomForestClassifier	n° estimators=10

Table B 7. ML parameters for the mixed approach, Ex. A, whole dataset

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty = 'l2', C = 1000
KNN	KNeighborsClassifier	n° neighbors = 4
SVM	SVC	kernel='poly', Degree = 2 C = 3800
Random Forest	RandomForestClassifier	n° estimators=10

Table B 8. ML parameters for the mixed approach, Ex. B, whole dataset

ML algorithm	Scikit-learn function	Scikit-learn Parameters (best performance)
Logistic Regression	LogisticRegression	solver = 'liblinear', penalty = 'l2', C = 1000
KNN	KNeighborsClassifier	n° neighbors = 4
SVM	SVC	kernel='rbf' C = 1000
Random Forest	RandomForestClassifier	n° estimators=10

Figure B 1 shows the parameters used to train the MPL neural network that obtained the performances presented in Results chapter. We utilised the same structure and the same parameters testing the network with n programming sequences, n-1 programming sequences and n-2 programming sequences.

```
#Neural Net stucture
classifier = Sequential()
#First Hidden Layer
classifier.add(Dense(40, activation='tanh', kernel_initializer='VarianceScaling', input_dim=15))
#Second  Hidden Layer
classifier.add(Dense(40, activation='tanh', kernel_initializer='VarianceScaling'))
#Third  Hidden Layer
classifier.add(Dense(10, activation='tanh', kernel_initializer='VarianceScaling'))
#Output Layer
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='VarianceScaling'))

sgd = optimizers.SGD(lr=0.1, decay=1e-2, momentum=0.9, nesterov=True)
```

Figure B 1. MLP Neural network structure, with parameters set to obtain best performances presented in Results chapters