Machine Learning for High School Students

Radu Mariescu-Istodor
University of Eastern Finland
School of Computing
radum@cs.uef.fi

Ilkka Jormanainen
University of Eastern Finland
School of Computing
ilkka.jormanainen@uef.fi

ABSTRACT

In this study, we developed a machine learning method for object recognition that can be implemented using knowledge that high school students attain during their normal math and IT classes. We then tailored a two-hour interactive lesson in which the students were divided into groups to implement solutions to six distinct problems required by the method. The solutions were later put together by the teacher into a working web application (HTML + JavaScript). The lesson was taught on three occasions in Romanian schools to students between 13 and 19 years old. The students were excited about the lesson, and the collected data measuring students' intrinsic motivation suggests that the given tasks and the type of instruction were motivating them. The students also found the lesson achievable regardless the level of their previous programming background. The students were even able to suggest viable improvements to the method. The lesson is presented in short in this (17 minute) YouTube video[1]. Furthermore, we utilized the developed machine learning tool in a workshop with primary school children. Observations from this workshop suggest wider applicability of the tool, as well as further research questions on machine learning in K-12 settings.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Applied computing → Education**.

KEYWORDS

machine learning, object recognition, school, education, HTML, JavaScript

ACM Reference Format:
Radu Mariescu-Istodor and Ilkka Jormanainen. 2019. Machine Learning for High School Students. In *19th Koli Calling International Conference on Computing Education Research (Koli Calling '19), November 21–24, 2019, Koli, Finland*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3364510.3364520

[1]https://youtu.be/QXB1ytG95gs

Koli Calling '19, November 21–24, 2019, Koli, Finland

ACM ISBN 978-1-4503-7715-7/19/11…$15.00
https://doi.org/10.1145/3364510.3364520

1 INTRODUCTION

Simple machine learning tasks, such as image or speech recognition have recently gained attention also in K-12 computing and ICT education. Tools, such as Scratch with its many variants have features to program and use machine learning driven projects. Similarly, cloud-based services such as IBM Watson or Google Machine Learning Kit make advanced machine learning capabilities approachable for programmers in all levels, and suitable front-ends for kids to use these tools have been developed [6]. These technologies are also behind web-services, such as Machine Learning for Kids[2]. The aim of many of these projects is to expose young learners to principles of machine learning and artificial intelligence. Often, the tools engage students in a typical cycle of supervised learning approach including steps such as data collection and entry, data visualization, feature engineering, model building, and testing. These projects are fit to prepare the students for the computing tasks and challenges they are going to face in the future when entering the job market [4].

However, use of the ready-made tools, such as Scratch, to teach machine learning principles leave part of the process in a black box. Principles of the system functionality in lower levels of abstraction remain unclear to the learners. For example, how a cloud-driven image recognition system actually makes a distinction between the training samples? It is obvious that these questions are not of interest in all learning contexts, but if the aim is to get a comprehensive view about how machine learning systems work and eventually teach the capable students to design and develop such systems, instead of just using them, we argue that it is difficult to achieve with these "black-boxed" tools. Programming has a recognized part in many school curriculum worldwide (see for example [10], [15]). This also implicates that we may expect the students to possess the necessary programming skills to complete the tasks, at least in higher grades.

Our objective was to design a machine learning method that can be understood by school students with knowledge they normally gain during their programming classes. The motivation for this comes in multiple forms. First, it introduces students to an actively studied research problem of our time. Second, it demonstrates to students that things they know already can be put together to build a powerful application. Thirdly, the exercise for building a machine learning system from scratch provides students with insights about how these systems work in low level, and what actually happens "behind the scenes" when a machine learning system is trained for example for image recognition.

A common approach to 'lure' people into the field is to present how easy something is to do. This video[3] literally starts with "6

[2]https://machinelearningforkids.co.uk/
[3]https://www.youtube.com/watch?v=cKxRvEZd3Mw

lines of code is what it takes to write your first machine learning program". While it shows that it is possible to make something powerful very easy... it is not your own. It is someone else's code packaged in sophisticated libraries. Not only that it is not 'your' program, but you also don't understand it. This is also what happens with Scratch and similar tools - they do not help to comprehend how machine learning features actually work.

For our experiment, we built a simple web-based machine learning tool for image recognition. The tool was designed and implemented with standard HTML5 and JavaScript without the help of any ready-made libraries, as described later in this paper. During the tutorial in classes, the students were provided with a bare-bone application and they were asked to implement the missing essential parts of the tool. The students' programmed modules were then integrated in a functional tool and demonstrated with sample pictures. After the workshop, the students were devised with a questionnaire to assess their intrinsic motivation towards the task and the learning approach in general.

This paper is organized as follows. First, we describe the implemented machine learning tool and the computational principles that were utilized in the tool. Then we explain the research setting and data collection instruments. After discussing the results, we draw some lines for future work.

2 MACHINE LEARNING IN K-12 EDUCATION

Machine learning can be considered as a vital part of future computational skills [4]. Hence, it is justified to include machine learning education as part of the computational thinking teaching agenda also in K-12 level. However, as of today, only few attempts to teach data science or machine learning for primary or secondary school kids appear in the literature, and many of them are still in draft level [5]. The few examples available describe projects where school kids have been taught data science in a context that is suitable for them. For example, Srikant and Aggarwal [17] describe an experiment where they engaged students between 10-15 years to build a friend predictor tool with Microsoft Excel. The authors describe that the kids were able experience how to use data science to successfully solve a relevant problem, and in doing so, also appreciate the power and the applicability of such a technique. Srikant and Aggarwal [17] describe a half-day workshop "fairly successful", indicating that the exercise pleased both the students and the educators.

Obviously, one can consider that highly popular educational robotics projects in K-12 education [1] teach also machine learning principles. However, often the tasks in these projects are to teach the basics of procedural or event-driven programming, instead of higher abstraction level steps that are needed in machine learning (data labelling, training, and evaluation of the model). When moving to tertiary education, there is a vast amount of research available on how robots have been used to teach machine learning for students in computer science and similar fields (for example [18], [20]). As such, these examples are not suitable to replicate in general K-12 education due to the required previous knowledge that the students should have.

When people indeed explain the inner workings of a machine learning method, whether in formal or informal education, they usually use decision trees[4,5,6,7]. Decision trees are often quoted as the basic or simplest method because they resemble the human decision making process and it is easy to understand examples of how they work [8], [16]. However, decision trees are automatically built by deciding how to split the data based on criteria like information gain or variance reduction [11] which are not known to high-school students. Some attempts to open also this process can be found from literature (for example [19]).

Unlike decision trees, the Nearest neighbor(s) algorithm does not have training steps that are difficult to comprehend. In its simplest form, the algorithm can work by simply calculating the distance to all points and selecting the nearest one(s). This is essentially a search for the minimum and is easy to explain in an educational setting. Indexing methods do exist which make the search faster, however, when the dataset is small, they are not necessary.

3 LEARN MACHINE LEARNING - THE TOOL

The developed machine learning application has a simple interface (see Figure 1). It uses an HTML canvas element (A) to display the camera images, a div element (B) where the name of the object will appear, an input text element (C) where a user can enter the name of the object and a button element (D) to use when learning new objects. Pressing the button counts as the training step. The testing step happens all the time when the app is on. The app classifies every image coming from the camera (24 frames / second) and the predicted name is shown in real-time. If no objects are learned yet, the app displays the question mark. If an object is classified incorrectly, the user can fix the mistake by specifying the correct name at any time (training step again).

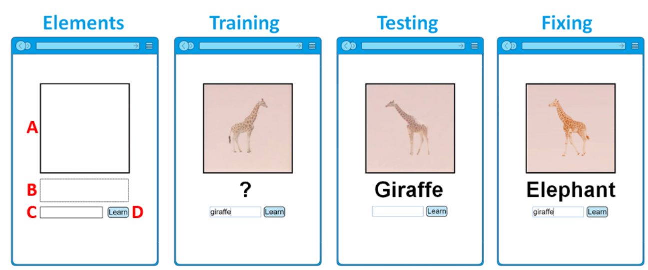

Figure 1: App elements and its usage.

3.1 Image recognition method

The core image recognition method works on gray-scale images because they can be simply represented as two-dimensional arrays. To implement the method we first threshold the image to isolate the object within. Then we calculate the bounding box of the object (see Figure 2). Two features are then calculated as follows:

- Aspect Ratio: shorter edge divided by the longer edge

[4]https://medium.com/analytics-vidhya/a-guide-to-machine-learning-in-r-for-beginners-decision-trees-c24dfd490abb

[5]https://lethalbrains.com/learn-ml-algorithms-by-coding-decision-trees-439ac503c9a4

[6]https://adityashrm21.github.io/Decision-Trees

[7]https://medium.com/x8-the-ai-community/decision-trees-an-intuitive-introduction-86c2b39c1a6c

- Fullness: proportion of black pixels inside the bounding box

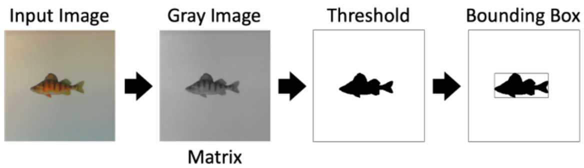

Figure 2: Isolating the object within the image.

These two properties are surprisingly effective to tell apart a variety of objects. During the learning stage, objects are stored in memory as 2D points and associated with their name (class), which users enter in a text field. In the classification step, new objects are classified by looking at the nearest neighbor in the feature space (see Figure 3).

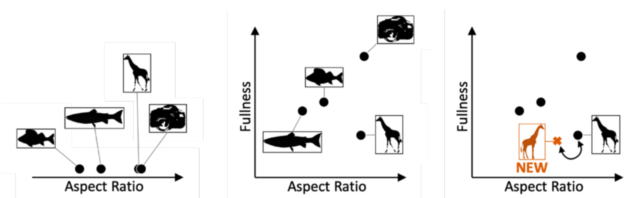

Figure 3: Objects represented by their Aspect Ratio (left), their Aspect Ratio and their Fullness (middle) and a new Object being classified as its nearest neighbor in the 2D feature space.

4 MACHINE LEARNING TUTORIAL

We designed a two-hour lesson plan for teaching the proposed method. Alternative plans with different durations ranging from 30 minutes to many hours can exist and we will discuss those in the later sections, however, we now focus on the two-hour lesson. During the lesson, the teacher, with the help of the students, builds a web app (HTML and JavaScript) that learns to recognize objects when shown to the camera. The lesson is structured into 5 modules:

- M1 - introduction to video images (15 min)
- M2 - method description (20 min)
- M3 - group work (30 min)
- M4 - assembling the app (15 min)
- M5 - experimentation (20 min)

In the first module (M1), basic information on what images are and how to work with them is taught. The topics include how to access the camera using HTML and JavaScript, how to display images from the camera using the HTML Canvas element, and how to obtain the matrix of pixel values from the camera images multiple times per second.

To aid with the explanation, the teacher used PowerPoint slides and a simple web app that only displays the camera input and prepares the pixel matrix. This app will be complemented with functionality implemented by the students as the lesson progresses. The source code of this web app is small (< 90 lines of code), however, it contains elements that are new to students such as handling the permissions to access the camera and converting the image signal into the matrix format. The main point to teach in this module is that gray-scale images are just matrices of numeric values and are automatically produced frequently (24 times per second).

After learning that an image can be represented as a matrix of integer numbers with values between 0 (black) and 255 (white), the students felt more comfortable because processing matrices is part of the 10th grade school curriculum in Romania [8] (most subjects were 10th grade or higher).

The next step of the lesson (M2) is to introduce the machine learning method. The PowerPoint slides were used for this purpose to identify six distinct components (tasks) that are required to solve the problem (see Figure 4). The steps are introduced one by one, starting with the need to isolate the object and continuing with extracting the features. The *Aspect Ratio* is the first feature introduced and demonstrated on four objects. Two objects: camera and giraffe are shown to be hard to distinguish based on that alone (see Figure 3), however, the camera has much more black pixels and thus, adding the *Fullness* feature makes the method much more capable. This way of teaching has several benefits:

(1) Aspect Ratio is presented by itself allowing students to get familiar with it.
(2) The need of using another feature becomes apparent.
(3) Students often propose the fullness feature themselves, increasing self-confidence.
(4) The fact that we add another feature suggests that we may be able to add even more.

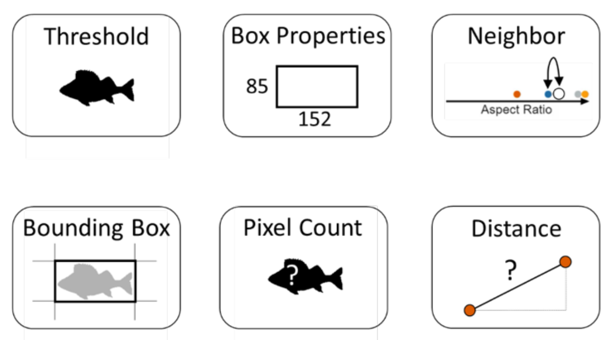

Figure 4: Six tasks (the detailed tasks can be found in Appendix 1).

After the six problems are introduced, students were divided into groups (M3) with each group focusing on a different task. If the students are of different age groups and of different backgrounds (we experienced students with 1h vs 7h computer science lessons per week), it is recommended that groups are balanced. Alternatively, if a student is not familiar with matrices, for example, he/she may be given tasks T5 and T6 which do not require such knowledge.

[8] http://ctice.md/ctice2013/?page_id=292

Students should write code in an environment that can be synchronized with the teacher, such as Google Docs. In our case, we designed a special environment for the students to write code in. The environment accomplishes three things:

- checks the code for errors and notifies upon successful completion
- measures the time needed to finalize the task
- synchronizes the source code with the teacher in real-time

While the students solved the sub-problems, the teacher was helping by visiting the groups and focusing on those that need help. Students were asked to write the code in any language that is familiar to them (typically C, C++, Python and JavaScript) and later the teacher gave advice on how to convert the code into JavaScript. During this time, a slide showing language specific differences was always on. Students that finished the task quickly were encouraged to try another task or to help the others.

After all tasks are solved, teacher completes the application by assembling the source code received from the groups and writing the remaining functionality in front of the students. The steps that the teacher does are:

- Calls the functions implemented by the students (T1 - T4) to calculate the Aspect Ratio and Fullness values and prints them on the screen for testing. Teacher tests with different objects to confirm that functionality is as expected (5 min).
- Writes function 'learn' to record the two properties as a 2D point and the name entered in the text box when pressing the Learn button (2 min).
- Updates the nearest neighbor function from T5 to work in two dimensions by using the Euclidean distance function from T6. If class consists of more experienced (typically older) students, discuss about the generalized Euclidean distance that works in higher dimensions and that any number of features can be used (5 min).
- Writes function 'classify' that updates the output field to show the name of the nearest neighbor from memory. To convince the students that it works, we recommend testing briefly by using the teacher as the subject and typing his name so that the app learns his appearance. And then, teacher lifts one hand up and enters the text 'hello'. Then the teacher repeats lifting his hand up and the app should alternate saying his name and the word 'hello'. This accomplishes two things. First, it rapidly tests that the app can categorize objects and second, it demonstrates that the app can have another purpose (3 min).

When moving on to experimentation (M5), the teacher presents the more sophisticated version of the application (Figure 5). The students are told that the app functions in the same way, but the additional features let us understand better how it works. These features are:

- a slider for selecting the threshold value
- a canvas displaying the thresholded image
- a visual representation of the 2D feature space which updates in real-time
- a listing of the learned items

While experimenting, several things can be investigated such as:

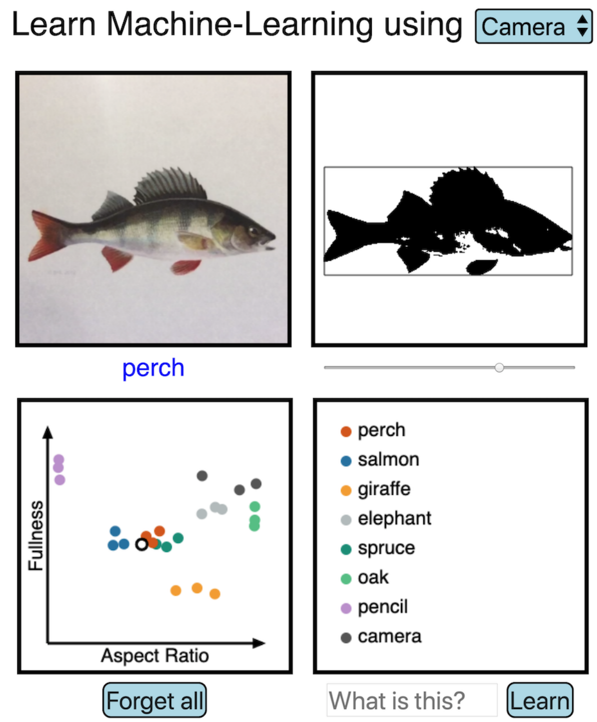

Figure 5: A complete version of the application

- ability to recognize letters
- ability to recognize objects up-side-down
- optical illusions
- ability to recognize people (students)
- ability to recognize hand gestures (rock, paper, scissors)

This has lead to interesting discussions and surprising outcomes like implementing a simple application that can count the number of squats by teaching the app the difference between the 'up' state and the 'down' state.

5 RESEARCH SETTING

The lesson was taught on three occasions for three distinct sets of students in Romanian high schools. The first school provides intensive classes on computer science and students had previous programming experience. The first set (S1) consisted of the best students of different age groups from the school. The second set (S2) was of students from a same class of the school. The third set (S3) was a mix of students from six schools in another city in Romania. The students (n=51) were recruited for the experiment by their teachers or tutors. The students were between 13-19 years old. Descriptive statistics of the participant groups are shown in Table 1.

5.1 Self-Determination Theory and Intrinsic Motivation Inventory

Theoretical background of the research lies in self-determination theory (SDT), which provides a framework to study an individual's

Table 1: Statistics of the students from the three teaching sessions.

Set	Size	Gender (M/F)	Programming competence	Age
S1	22	17/5	Very high	13-19
S2	11	7/4	High	17-18
S3	18	14/4	Medium	13-18

motivation and emotions, and development of these aspects. In the core of the model are the individual's motivation, personality, and optimal actions. Self-motivation and self-determination are defined by the intrinsic and extrinsic factors. It has been shown that intrinsic motivation has a direct connection to individual's optimal learning and skill development. It has been argued by Deci and Ryan [3] that development of students' intrinsic motivation should be a priority in teaching and learning processes. Intrinsic motivation supports activities that are naturally enjoyable and interesting for an individual. Also, personal values have a strong influence on individual's intrinsic motivation [13].

The main research instrument used in the study was *Intrinsic Motivation Inventory* (IMI) [12], [14]. The IMI is a multidimensional instrument to assess participants' self-determination and subjective experience related to a task. The IMI instrument has been used successfully in different contexts, also in computing [2]. The instrument's validity has been also verified [9]. The IMI instrument is often modified to suit for a specific purpose. From the seven subscales of IMI (Interest/Enjoyment, Perceived Competence, Effort/Importance, Pressure/Tension, Perceived Choice, Value/Usefulness, Relatedness) we chose the following to assess the students' motivation towards the machine learning lesson and this type of instruction model:

- Interest/Enjoyment
- Perceived Competence
- Effort/Importance
- Value/Usefulness

5.2 Pre-lesson questionnaire

We first evaluated the students using a pre-lesson questionnaire which they filled before the lesson starts. Second, during the lesson, we measured the time necessary for student groups to accomplish their task.

The pre-lesson questionnaire contained four statements that were planned to reflect to the IMI categories in the post-lesson questionnaire. However, the connection is only indicative and this information was used primarily to map students' background and existing knowledge. In addition to the statements, the pre-questionnaire was used to get information about students' previous programming experience and languages they have been using earlier. Most of students in S1 indicated that they had previous knowledge about C++ or Java. Other languages appearing in the questionnaire were Python, Pascal, and Scratch. Less than 50% of the students indicated that they had previous experience with HTML or JavaScript.

(1) I enjoy programming. (Interest/Enjoyment)
(2) I am good at programming. (Perceived Competence)
(3) Programming is one of my hobbies. (Effort/Importance)
(4) Programming skills are important for my future. (Value/Usefulness)

Using the pre-lesson questionnaire, we found students (n=51) to be generally very interested in programming ($\overline{x}$=4.33) and to consider it as their hobby ($\overline{x}$=3.31). Students considered themselves to be good at programming ($\overline{x}$=3.82) and they think it is a very important skill for their future ($\overline{x}$=4.08). Students in S1 and S2 considered themselves significantly better programmers than students in S3. Students from S1 see a significantly higher value than other sets.

5.3 Post-lesson questionnaire

After the lesson, the students were given a post-lesson questionnaire. The post-questionnaire design was based on the IMI instrument. Each category in the post-lesson questionnaire had five statements (total 20). The statements were answered with five-step Likert scale (Strongly disagree .. Strongly agree). The original IMI instrument statements were modified to fit better in our activities. For example, the original statement "I enjoyed this activity very much" was modified as follows: "I enjoyed doing the machine learning exercise very much". In the analysis phase, the students' answers were grouped according to the selected IMI categories, and mean values were calculated for each category. These values were used to correlate students' answers against their age, gender, and the perceived motivational issues in the pre-questionnaire.

The following questions were modified from the original IMI instrument and were used in the questionnaire. The questionnaire was translated to Romanian language so the students were able to answer by using their native language. As it is typical for the IMI instrument, there are overlapping questions between the categories. The questions were presented to the students in mixed order to make this overlapping less salient for the participants.

Interest/Enjoyment

- I enjoyed doing the machine learning exercise very much.
- The machine learning exercise was fun to do.
- I think this was a boring exercise.
- I think making this machine learning exercise was very interesting.
- I think that this exercise was quite enjoyable.

Perceived Competence

- I think I was pretty good at making the machine learning system.
- I think I did pretty well at programming the machine learning system, compared to other students.
- After programming the machine learning system for a while, I felt pretty competent.
- I am satisfied with my performance while programming the machine learning system.
- This was an exercise that I couldn't do very well.

Effort / importance

- I put a lot of effort in making the machine learning system.
- I tried very hard on making the machine learning system.
- It was important to me to do well in programming the machine learning system.

- I put much effort in programming the machine learning system.
- I didn't try very hard to do well at this exercise.

Value / usefulness

- I think that doing this exercise is useful for learning programming.
- I think this is important to do because it helps me to become a better programmer.
- I would be willing to do the machine learning exercise again because it has some value to me.
- I think that doing this exercise helped me to understand better what machine learning is.
- I believe doing this machine learning exercise was beneficial to me.

6 RESULTS

The students' answers in the post-lesson questionnaire were summarized across the categories (Table 2). We found that students perceived the lesson very interesting ($\bar{x}$=4.50, *SD*=0.47) and they felt competent during the lesson ($\bar{x}$=3.42, *SD*=0.69). The lesson was not too difficult ($\bar{x}$=3.29, *SD*=0.74) but at the same time, students found it very useful ($\bar{x}$=4.37, *SD*=0.43). It is remarkable that sets have similar opinions for the usefulness. This means that the lesson was interesting for both very good students in S1 (who arguably do not find a challenge in any of the six tasks, but presumably are interested in the overall app), and to the less experienced (or average) students in S2 and S3, for whom the lesson should be also accessible.

Table 2: Summary of the post-questionnaire (IMI instrument categories.

Category	S1	S2	S3	Mean
Interest/Enjoyment	4.60	4.71	4.22	4.50
Perceived Competence	3.50	3.49	3.28	3.42
Effort/Importance	3.50	3.36	3.04	3.29
Value/Usefulness	4.50	4.47	4.20	4.37

There is no significant difference between the genders apart from the fact that females tend not to consider programming as a hobby, while at the same time, see a higher value in the lesson given. This fact, however, can also be due to not having a large enough sample for calculating statistics. A number of 43/51 students were able to solve at least one task as a group, however, 5/8 students had to leave during the lesson. Students required 16 minutes on average to complete a task. The maximum time required to complete a task was 40 minutes, however, the time measurement is not entirely trustworthy, because often groups had a minor issue such as declaring a variable as `int a;` (specific to C language) instead of `var a;` (specific to JavaScript. Time was also ticking during a 10 minute break that some students chose to have.

We found that students adapted to JavaScript quite quickly even though many of them were not familiar with the language. In fact, there was almost no correlation between knowing the language beforehand and performance on the programming task (Pearsons = 0.09). Exceptions are a couple of students who were only familiar with Pascal, whose syntax is very different from JavaScript. Their algorithm was correct, however, they needed significant help from the teacher to port the source code.

Age correlated to the ability to solve the tasks (Pearsons = 0.31), however, there was no correlation (Pearsons = 0.09) when only S1 and S2 were considered. This is because those groups are experienced enough while in S3 the age correlation exists because age correlates more with experience.

7 DISCUSSION

Machine learning, as part of artificial intelligence, is a rapidly emerging field in all levels of society where information systems are used to support everyday life of people. Hence, it is vital that machine learning and similar topics are considered also in computing and computational thinking education in K-12 level. Keeping this in mind, we planned and presented a two-hour tutorial in three different high schools in Romania. The results show that the students perceived our web-based tool well, and they were able to implement the system during the tutorial walk-through. It is remarkable that not all students were familiar with JavaScript, which was the programming language in our web-based implementation. The experience running the tutorial shows also that this kind of collaborative working approach suits well for high school students, and they are capable to come up with new and unexpected ideas. The students were well capable to transfer their previous programming knowledge to new implementation paradigm and programming language. The experiment presented in this article focused mostly on how to implement such a machine learning system. Obviously, the particular challenge is possible to conduct with high school or similar level students, as it requires some mathematical and computational concepts, such as Euclidean distance and matrix manipulation operations. These skills are taught latest in the 10th grade in Romanian school context. The results show that when guided carefully, variance in these background skills does not interfere with the project implementation.

For younger students, the same tool can be nevertheless used in other ways. During the experiments in Romania, we conducted an unplanned ad-hoc machine learning workshop in Finland for primary school kids (grades 1-6, age 7-12 years). The intention of this workshop was not to collect research material, as the call for conducting the session came in the last minute and we did not have time to prepare suitable data collection instruments and a research protocol. The following discussion is therefore based on our observations and some materials produced in and for the workshop. Despite this rather anecdotal scientific evidence, we hope that the lessons learned help to frame a more rigorous research framework for future interventions.

During a 30 minutes workshop session for each grade level of the school (about 40 students at time), we explained the principles of machine learning to the students, modifying the explanation according to student group's age. The main focus was on demonstrating how the machine learning system (Figure 5) is trained to recognize a fish, an elephant, and a giraffe. This was done with printed, clearly identifiable pictures of these animals, as well as with hand-drawn illustrations.

After the system was demonstrated and trained with an adequate number of samples, the school kids were asked to draw one of these animals and try if the system is capable to recognize the drawing (Figure 6). When the system was not able to recognize the drawings correctly (as it indeed happened in many cases!), the reasons for this unexpected behavior were discussed with the students. During this discussion, more details of underlying concepts of the system were described for them, such as that aspect ratio of an elephant and a giraffe need to be different in order to make a difference between the animals. The students often drew very "thin" elephants, making them to mix easily with the trained giraffe samples. Another issue with the drawings was that the students tend to draw only outlines of the animals, perhaps only with some decorations. This led easily to a situation where the Fullness property was too low comparing to the trained samples. After fixing or redrawing the picture, most of the students were able to get the system to recognize their piece of art. It was remarkable that the students were really eager to fix or redraw their picture, in order to make the system to work with their very own illustrations. They seemed to feel a big pride for being able to draw an animal that a computer program can recognize properly. Likewise to findings in [17], creating own dataset for the machine learning task seems to motivate students. This sense of ownership and increased motivation is a known phenomenon in ICT education when working for example with educational robotics [7], and it is definitely one of the interesting aspects for further studies.

Another interesting observation emerged when some older students (10-12 years) were engaged in more detailed discussions about the recognition principles with the help of Aspect Ratio - Fullness diagram (bottom left part in Figure 5). When the students saw the system trying to classify their sample in real-time incorrectly, they were eager to suggest to further train it with their drawing. Sometimes this led to the desired end result, and the system was actually more capable to recognize the typical drawings of the students. Sometimes, this led to a situation where the trained sample clusters got too close to each other, and reliability of the system was decreased. These results were very valuable lessons to learn for us and evidently also for the students, who demonstrated a clear understanding about why the system behaved as it did. This ad-hoc workshop was an encouraging experience for us, and explaining the machine learning principles and using the system with more than 200 school kids over three hours revealed many research problems and open questions that could be addressed in the future.

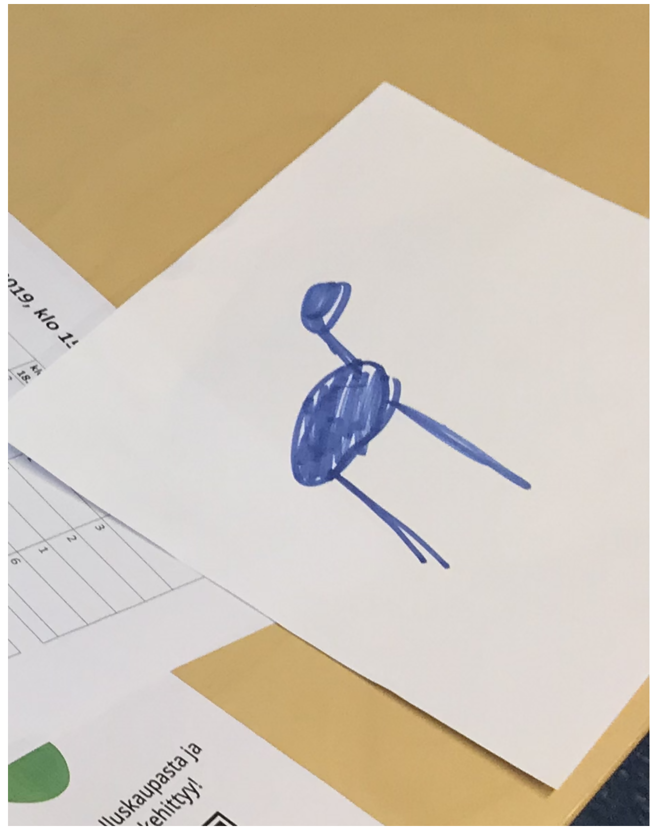

Figure 6: A hand-drawn giraffe created during the machine learning workshop for the primary school kids.

ACKNOWLEDGMENTS

We thank students and the teachers from Colegiul Naţional Traian, Liceul Teoretic William Shakespeare, Liceul Teoretic Vlad Ţepeş, Liceul Teoretic Grigore Moisil, Colegiul Naţional Pedagogic Carmen Sylva, Bartók Béla Elméleti Líceum, and Liceul Teoretic Nikolaus Lenau schools (Romania), and Kylmäoja primary school (Finland), for participating to the study.

REFERENCES

[1] Fabiane Barreto Vavassori Benitti. 2012. Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education* 58, 3 (2012), 978 – 988. https://doi.org/10.1016/j.compedu.2011.10.006

[2] K. Chin, Z. Hong, and Y. Chen. 2014. Impact of Using an Educational Robot-Based Learning System on Students' Motivation in Elementary Education. *IEEE Transactions on Learning Technologies* 7, 4 (Oct 2014), 333–345. https://doi.org/10.1109/TLT.2014.2346756

[3] Edward L Deci and Richard M Ryan. 1987. The support of autonomy and the control of behavior. *Journal of personality and social psychology* 53, 6 (1987), 1024.

[4] Peter J. Denning and Matti Tedre. 2019. *Computational Thinking*. MIT Press.

[5] Birte Heinemann, Simone Opel, Lea Budde, Carsten Schulte, Daniel Frischemeier, Rolf Biehler, Susanne Podworny, and Thomas Wassong. 2018. Drafting a Data Science Curriculum for Secondary Schools. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. ACM, New York, NY, USA, Article 17, 5 pages. https://doi.org/10.1145/3279720.3279737

[6] Ken Kahn and Niall Winters. 2017. Child-Friendly Programming Interfaces to AI Cloud Services. In *Data Driven Approaches in Digital Education*, Élise Lavoué, Hendrik Drachsler, Katrien Verbert, Julien Broisin, and Mar Pérez-Sanagustín (Eds.). Springer International Publishing, Cham, 566–570.

[7] Fatima Kaloti-Hallak, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2015. Students' Attitudes and Motivation During Robotics Activities. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15)*. ACM, New York, NY, USA, 102–110. https://doi.org/10.1145/2818314.2818317

[8] Chun Fu Lin, Yu chu Yeh, Yu Hsin Hung, and Ray I Chang. 2013. Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education* 68 (2013), 199 – 210. https://doi.org/10.1016/j.compedu.2013.05.009

[9] Edward McAuley, Terry Duncan, and Vance V. Tammen. 1989. Psychometric Properties of the Intrinsic Motivation Inventory in a Competitive Sport Setting: A Confirmatory Factor Analysis. *Research Quarterly for Exercise and Sport* 60, 1 (1989), 48–58. https://doi.org/10.1080/02701367.1989.10607413 arXiv:https://doi.org/10.1080/02701367.1989.10607413 PMID: 2489825.

[10] Keith Quille, Roisin Faherty, Susan Bergin, and Brett A. Becker. 2018. Second Level Computer Science: The Irish K-12 Journey Begins. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. ACM, New York, NY, USA, Article 22, 5 pages. https://doi.org/10.1145/3279720.3279742

[11] J Ross Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.

[12] Richard M. Ryan. 1982. Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of Personality and Social Psychology* 43, 3 (1982), 450 – 461.

[13] Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25, 1 (2000), 54–67.

[14] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55, 1 (2000), 68 – 78.

[15] José-Manuel Sáez-López, Marcos Román-González, and Esteban Váquez-Cano. 2016. Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch in five schools. *Computers & Education* 97 (2016), 129 – 141. https://doi.org/10.1016/j.compedu.2016.03.003

[16] Gerald F. Smith. 2003. Beyond Critical Thinking And Decision Making: Teaching Business Students How To Think. *Journal of Management Education* 27, 1 (2003), 24–51. https://doi.org/10.1177/1052562902239247 arXiv:https://doi.org/10.1177/1052562902239247

[17] Shashank Srikant and Varun Aggarwal. 2017. Introducing Data Science to School Kids. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 561–566. https://doi.org/10.1145/3017680.3017717

[18] Tapani Toivonen, Ilkka Jormanainen, and Markku Tukiainen. 2018. An Open Robotics Environment Motivates Students to Learn the Key Concepts of Artificial Neural Networks and Reinforcement Learning. In *Robotics in Education*, Wilfried Lepuschitz, Munir Merdan, Gottfried Koppensteiner, Richard Balogh, and David Obdržálek (Eds.). Springer International Publishing, Cham, 317–328.

[19] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H Witten. 2001. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies* 55, 3 (2001), 281 – 292. https://doi.org/10.1006/ijhc.2001.0499

[20] Daniel Wong, Ryan Zink, and Sven Koenig. 2010. Teaching artificial intelligence and robotics via games. In *First AAAI Symposium on Educational Advances in Artificial Intelligence*.

Appendix 1. The six tasks for the tutorial lesson.

Threshold

Write a function that counts how many times a given value appears in a given pixel matrix. The matrix is square. Matrix rows and columns begin at 1.

```
EXAMPLE
matrix =
255, 255, 255, 255, 255, 255
255, 255, 255,   0, 255, 255
255, 255,   0,   0, 255, 255
255,   0,   0,   0,   0, 255
255, 255, 255, 255, 255, 255
255, 255, 255, 255, 255, 255
matrixSize = 6
value      = 0

OUTPUT -> 7
```

Bounding Box

Write a function that obtains the bounding box of the region containing zeros in a given square matrix. The bounding box must be defined by the following attributes {xMin, xMax, yMin, yMax}. Matrix rows and columns begin at 1.

```
EXAMPLE
matrix =
255, 255, 255, 255, 255, 255
255, 255, 255,   0, 255, 255
255, 255,   0,   0, 255, 255
255,   0,   0,   0,   0, 255
255, 255, 255, 255, 255, 255
255, 255, 255, 255, 255, 255
matrix size = 6

OUTPUT
{
       xMin: 2,
       xMax: 5,
       yMin: 2,
       yMax: 4
}
```

Box Properties

Write a function that calculates properties of a given box:
- length
- width
- area

The box has attributes { xMin, xMax, yMin, yMax }. Access the box attributes as box.xMin, box.xMax, ...

```
EXAMPLE
box =
{
       xMin: 2,
       xMax: 5,
       yMin: 2,
       yMax: 4
}

OUTPUT
{
       length : 4 ,
       width  : 3 ,
       area   : 12
}
```

Pixel Count

Write a function that counts how many times a given value appears in a given pixel matrix. The matrix is square. Matrix rows and columns begin at 1.

```
EXAMPLE
matrix =
255, 255, 255, 255, 255, 255
255, 255, 255,   0, 255, 255
255, 255,   0,   0, 255, 255
255,   0,   0,   0,   0, 255
255, 255, 255, 255, 255, 255
255, 255, 255, 255, 255, 255
matrixSize = 6
value      = 0

OUTPUT -> 7
```

Neighbor

Write a function that given a sample point, finds the nearest value in an array of points and returns its index. First element of the points array has index 1. Use the absolute value Math.abs(-1)=1;

```
EXAMPLE 1
samplePoint = 42
points      = 32, 64, 128, 8, 16
pointCount  = 5

OUTPUT -> 1
```

Distance

Write a function that calculates the Euclidean distance between two points. Each point is represented as an array with two values. Math.sqrt(x) returns the squared root of a variable x.

```
EXAMPLE
p1 = (1, 4)
p2 = (5, 1)

OUTPUT -> 5
```