LearningML: A Tool to Foster Computational Thinking Skills Through Practical Artificial Intelligence Projects

LearningML: una herramienta para fomentar las habilidades de Pensamiento Computacional mediante proyectos prácticos de Inteligencia Artificial

Juan David Rodríguez-García
INTEF. Madrid, Spain.
juanda.rodriguez@educacion.gob.es

Jesús Moreno-León
Programamos. Madrid, Spain.
jesus.moreno@programamos.es

Marcos Román-González
Universidad Nacional de Educación a Distancia (UNED). Madrid, Spain.
mroman@edu.uned.es

Gregorio Robles
Universidad Rey Juan Carlos. Madrid, Spain.
grex@gsyc.urjc.es

Abstract

The use of Artificial Intelligence (AI) offers new and thriving opportunities, but introduces also new risks and ethical issues that should be dealt with. We argue that the introduction of AI contents at schools through practical, hands-on, projects is the way to go to educate conscientious and critical citizens of the future, to awaken vocations among youth people, as well as to foster students' computational thinking skills. However, most existing programming platforms for education lack some of the required educational features to develop a complete understanding of AI. In this paper we present `LearningML`, a new platform aimed at learning supervised Machine Learning (ML), one of the most successful AI techniques that is in the basis of almost every current AI application. This work describes the main functionalities of the tool and discusses some decisions taken during its design. For its conception, we have taken into account lessons learned from the research literature on introducing AI in school and from the analysis of other educational tools built with the aim to allow learners to use ML. We offer as well some promising results obtained after a preliminary testing pilot workshop. Finally, the next steps in the development of `LearningML` are presented, focused on the face and instructional validation of the tool.

Keywords: Computational Thinking, Educational Tools, Learning by doing, Machine Learning

Resumen

El uso de Inteligencia Artificial (IA) ofrece nuevas y prósperas oportunidades, pero también introduce nuevos riesgos y cuestiones éticas que deben abordarse. Sostenemos que la introducción de contenidos de inteligencia artificial en las escuelas a través de proyectos prácticos es el camino a seguir para educar ciudadanos conscientes y críticos, para despertar vocaciones entre los jóvenes, y para fomentar las habilidades de pensamiento computacional de los estudiantes. Sin embargo, la mayoría de las plataformas educativas de programación existentes carecen de algunas características necesarias para desarrollar proyectos completos de IA y, en consecuencia, se requieren nuevas herramientas. En este artículo presentamos `LearningML`, una nueva plataforma dirigida al Aprendizaje Automático (ML) supervisado, una de las técnicas de IA más exitosas que se encuentra en la base de casi todas las aplicaciones actuales de IA. Este trabajo describe las principales funcionalidades de la herramienta y discute algunas decisiones tomadas durante su diseño, para el que hemos tenido en cuenta las lecciones aprendidas al revisar trabajos anteriores realizados para introducir la IA en la escuela y el análisis de otras soluciones destinada al aprendizaje de ML. Analizamos, además, los prometedores resultados que hemos obtenido en la realización de un taller preliminar para probar la eficacia de la herramienta. Por último, presentamos los próximos pasos en el desarrollo de `LearningML`, que se centran en la validación, tanto aparente como instruccional, de la herramienta.

Palabras clave: Pensamiento computacional, Herramientas educativas, Aprender haciendo, Aprendizaje automático.

Introduction

Some sixty years after Dartmouth's workshop in 1956, a seminal event commonly agreed upon as its birth, Artificial Intelligence (AI) has transcended academia and science fiction and has landed in society as another everyday technology. Any individual having a mobile phone is using, being more or less aware of it, AI based applications. Even more, some of his decisions can be influenced by an AI algorithm. recommendation systems, facial recognition, speech-to-text conversion, language translation, web searching, self-parking cars, chatbots, virtual conversational assistants, email spam filters, and predictive writing are some examples of everyday tasks related to AI that are carried out by most of the people, regardless of their professional activity or even social scale. And of course, children are not an exception. A brief introduction on the current status of AI is presented in the *Background* section of this paper.

Governments around the world, worried about benefits and risks AI poses, are developing policies, strategic plans, and all kind of initiatives around this subject as suggested by the myriad of reports being written in last years (Pedró, Subosa, Rivas, & Valverde 2019; Tuomi, 2019; Ministerio de Ciencia, Innovación y Universidades, 2019). These reports raise education as a necessary cornerstone to enable successful adoption of AI in society. As a consequence, teaching and learning of AI contents at school is a clear line of action that must be developed if we are to educate conscientious citizens with a correct idea of what AI is and how it affects them. Even more, introducing AI contents in school is necessary to awaken vocations among young people and to address the increasing number of STEM and AI positions expected in the near future. In the *Related Research* section, we review some of the works done so far.

We argue that CT could be an appropriate framework to introduce AI contents in school. Although there is no consensus about a clear definition of CT, most people agree that problem representation and problem solving using computer power is a basic feature characterizing this skill (Moreno-León, Robles, Román-González, & Rodríguez-García, 2019). Therefore, both coding and unplugged practical activities have been shown as effective instruments for CT development. Since AI is a subfield of Computer Science

(CS) having much to do with programming, practical AI projects could be an engaging way for both CT skill development and AI content introduction. However, although successful programming platforms as `Scratch` (Resnick, 2009) allow children to develop practical programming applications, they are not enough to develop complete AI projects, thus new educational tools intended to fill this gap are demanded. In the *Educational ML tools* section, we analyze some of these tools and compare their main features in a summarizing table.

The analysis of the state of the art (including the research literature of tools devoted to teach ML) has been the starting point of `LearningML`, a tool we have developed and that is thoroughly described in the next section. `LearningML` is an educational tool aimed to teach and learn the fundamentals of ML in school. It has been designed with the "*low floor, high ceiling and wide walls*" principles (Resnick et al., 2009) as a guideline. That is, the tool should be very easy to start with (low floor) but also provide opportunities to create increasingly complex projects over time (high ceiling). Also, it should offer the possibility to support different types of projects in order to engage a wide range of interests and learning styles (wide walls).

Next, we offer some promising results of an educational intervention with `LearningML`, obtained with preliminary testing pilot workshops. Two small workshops were organized, with a total of n=14 participating students. Even though the number of participants is low, we could observe significant learning with a considerable effect size.

Finally, we present some conclusions and discuss the next steps regarding the development of `LearningML`, mainly focused on its validation in educational scenarios.

Background

AI is a phrase coined by McCarthy, Minsky, Rochester and Shannon during a workshop held in Dartmouth in the summer of 1956, which aimed to gather a selected group of scientists to work "*on the basis of conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it*" (McCarthy, Minsky, Rochester, & Shannon, 2006).

This pioneer work, generally agreed among computer scientists as the birth of AI, was really ambitious and the main problem it raised is nowadays known as *General Artificial Intelligence*, something that has not yet been achieved.

Since this seminal work, AI has been a prolific CS field that has attracted many researchers, and institutions around the world have invested lots of resources in its development. Although AI research has been alternating between optimistic periods followed by pessimistic ones, after more than sixty years of research lots of related subfields have emerged. Planning problem solving, natural language processing, knowledge representation, expert systems, neural networks, machine learning, robotic and computer vision are some of the most successful and broadly used in current applications. However, although really impressive results have been reached in all of these fields, we are far from something similar to an artificial mechanism able of behaving like a human brain.

Among all the computer science fields yielded by the AI development, Machine Learning (ML) has emerged as one of the most successful ones, used by all kinds of applications intended to perform classification, prediction and recognition tasks by means of computers. Due to its success, ML and AI are frequently, but wrongly, understood as the same thing. A great part of AI's recent success has been possible mainly to the power of ML. Examples of this are AlphaGo[1] --a program able to learn to play go and even win the world champion--, autonomous driving, facial and speech recognition, or language translation.

There are lots of problems which can be precisely solved top-down by applying known algorithms. Obtaining the greatest common divisor (GCD) of two numbers is a clear example of this type of problems. A well-defined set of computations, known as *Euclidean algorithms*, will always return the right solution: divide the greatest number by the less one, if the remainder is zero then the smaller number is the GCD, else divide the smaller number by the division remainder and repeat this step until a zero remainder is found, being the last dividend the solution.

This kind of problems, having more or less complexity, are the most suitable to be solved by computers. In fact, they are the type of problems we use in school when teaching to code. However, there are other types of problems that cannot be solved precisely by a finite set of computations; for example, to recognize cats and dogs among a set of images of cats and dogs. It is remarkable that this problem is easily solved by a human brain, however, it has been for decades a very difficult problem to be addressed by computers. For this reason, it is considered as an AI problem.

When we do not have an algorithm able to be applied to our problem, but we have lots of data related with its solution, we can deal with the problem from another perspective: we can use these data to infer possible solutions. This is known as a bottom-up strategy. And this is the essence of ML techniques: to gather a large amount of data from which some useful pattern can be extracted to build a model capable of yielding probable solutions to the problem; for instance, to determine if the image presented as input in our last example represents a cat or a dog, and what the probability is of being correct.

So, ML is the process of programming computers to optimize a performance criterion using example data or past experience. ML is used to create useful approximations for processes to solve tasks that we do not have algorithms for, but that we do have relevant data to learn patterns from (Tang, 2019). The term *learning* is a metaphor which reflects the fact that the rules to solve the problem are found through an intensive analysis of the available data.

The numerous algorithms composing the ML family belong to one of the following types: *supervised learning*, *unsupervised learning* and *reinforcement learning*. In *supervised learning*, data from which an inference model is to be built must be classified manually (by a human being). Afterwards these data are used by the ML algorithm to build a model that serves as well when other data is used.

[1] https://deepmind.com/research/case-studies/alphago-the-story-so-far

Unsupervised learning encompasses those algorithms intended to extract some patterns from a set of unlabeled data. Therefore, a classification “by hand” is not required. Sometimes this kind of algorithms is used to perform an initial automatic data labelling.

Finally, the algorithms belonging to *reinforcement* learning build their models by testing possible solutions; those that maximize some reward function are maintained while those that score low according to that function are eliminated. Here, too, labeled data is not necessary.

Although current ML research is producing powerful new techniques, its success is mainly due to the large amount of data available and the enormous computing power of today's computers. It should be noted that most of the ML algorithms are computationally very intensive. However, the number of ML-based applications increases every day and all of us use them regularly being more or less aware of the underlying technology and their consequences. As the results shown by ML models are not necessarily accurate and may even be biased, we must be very critical and attentive to the results obtained. This is one more reason to teach AI content in school.

But, although teaching ML in school would be very suitable to foster critical thinking and to prepare future generations for a highly technological world, can we really do it? We believe so, and this is the main hypothesis that we try to demonstrate with our `LearningML` tool. Even more, we think that ML practical projects can be a very suitable instrument to foster the development of Computational Thinking (CT) skills.

Related Research

In Rodríguez-García, Moreno-León, Román-González, & Robles (2019), a non-systematic literature review on teaching and learning of AI contents in school is made. This revision shows some works suggesting the introduction of AI content in CT frameworks as well as practical activities intended to reveal fundamentals of AI and ML to students. Afterwards an educational resource developed by the institution of the first author (INTEF) aimed to teach ML in schools with `Scratch`[2] and `Machine Learning for Kids`[3] is described. Finally, the testimonials of three educators who have implemented their own version of this resource are depicted.

Brummelen, Shen, & Patton (2019) propose the extension of Brennan-Resnick CT framework (Brennan & Resnick, 2012) adding key elements of ML: *classification*, *prediction* and *generation* are proposed as new concepts, *training*, *validating* and *testing* as new practices and *evaluation* as a new perspective. A new framework for CT based on a disciplinary perspective which takes into account the practice of CT used in STEM workplaces is suggested in Malyn-Smith, Lee, Martin, Grover, Evans, & Pillai (2018), being ML one of the practices pointed out by the authors.

One of the most relevant initiatives carried out to introduce AI contents at school is *AI4k12* which is broadly described in Touretzky, Gardner-McCune, Martin, & Seehorn

[2] https://scratch.mit.edu/
[3] https://machinelearningforkids.co.uk/

(2019). Being aware of the importance of AI in our lives, the partners that constitute this initiative wonder *what should every child know about AI*. Thus, the authors compile a list of curated resources to work with AI in K-12 levels (*Touretzkyds/ai4k1*2, n.d.). Via a collaboration between AI experts and K-12 teachers, the authors develop a helpful framework to teach AI built on top of five ideas: *i) computers perceive the world using sensors, ii) agents maintain models/representation of the world and use them for reasoning, iii) computers can learn from data, iv) agents interacting with humans is a substantial challenge for AI developers, and v) AI applications can impact society in both positive and negative ways*. These ideas are organized in four grade bands: K-2, 3-5, 6-8 and 9-12.

Every technology has been the source of new ethical issues, and surely AI is one of the most fruitful in this aspect. This is the subject of the fifth idea of the AI4k12 framework and one of the basis for the curricula that Ali, Payne, Williams, Park & Breazeal (2019) are developing to teach AI in school, which emphasize constructionist learning, ethics and creativity. In the authors' opinion, the platforms that allow children to create their AI projects are focused solely on technical aspects. This is not enough given the important ethical issues yielded by AI. Three different projects that engage primary and middle schools' students in AI education with ethical, constructionism and creativity as design guidelines, are presented in this paper.

How to address moral, ethical and philosophical issues in designing AI courses is the matter undertaken in Burton, Goldsmith, Koenig, Kuipers, Mattei & Walsh (2017). As instructors we want to develop contents that not only prepares students to be AI practitioners, but also to understand the moral, ethical and philosophical impacts that AI will have in society. In this paper, authors provide some practical case studies and links to resources for use by AI educators. They also provide concrete suggestion on how to integrate AI ethics into a general AI course and how to teach a stand-alone AI ethics course.

In Hitron, Wald, Erel & Zuckerman (2018) a relevant experiment to study whether 10-12 years old children can understand basic ML concepts is carried out. To asses children's understanding they applied an experimental design including a pretest, a "Wizard of Oz" (WoZ) (Dahlbäck, Jönsson, & Ahrenberg, 1993) based ML activity intended to recognize gestures, and a posttest. The participants were provided with a digital stick-like device and were informed that this device was able to record their hand movement and send the data to a computer. This stage was termed "sampling stage". The goal of the activity was to build a system able to recognize tennis serves. They were then invited to evaluate their training by testing the accuracy of the system, and informed that they can iterate between sampling and evaluation stages. Obviously during the experiment none of the children was aware about the fake ML system being simulated by researchers. The findings of this experiment suggest that children are able to understand basic ML concepts and can even apply their own examples and evaluate them in an interactive way.

In a posterior investigation, these researchers carried out a similar experiment (Hitron, Orlev, Wald, Shami, Erel & Zuckerman, 2019) including a pretest, a ML activity intended to recognize gestures, and a posttest. However, this time the ML system was improved by shifting from a WoZ simulation to a real ML gesture system. The results were similar; they found that children in the age range of 10-13 could understand basic ML concepts, apply their knowledge in different contexts and generate accurate and meaningful new examples for real-world ML applications. Furthermore, they revealed that a direct experience with data labeling and evaluation is a key element to achieve the understanding of concepts. Taking into account these findings, the authors recommend ML product designers to integrate direct experience with accessible building blocks, uncovering some black boxes of the ML process and allowing children to collect data by themselves, receive feedback on system's accuracy, and iterate in a process of trial and error.

"Does the computational experience with basic AI algorithms affect the perception of the capability of AI to affect the lives of common people?" is the research question addressed in Estevez, Garate, & Graña (2019), a follow-up of the work reported in Estevez, Garate, Guede & Graña (2019). Following a design-based research approach (The Design-Based Research Collective, 2003) they propose an educational workshop for undergraduate students in the age range of 16-17, focused on two specific AI tasks: data clustering and artificial neural network (ANN) learning. To solve these tasks, the authors selected two simple enough algorithms to be implemented in `Scratch` but powerful enough to produce non-trivial results: K-means for the clustering problem and an application of a least-square algorithm and a logic activation function for the ANN. In both cases the mathematical background was chosen taking into account the mathematical curricula of the participants. Students who took part in the workshop changed their opinion and perception about what AI is and how it affects their lives.

Educational ML tools

A large part of the tasks in which current ML applications are being very successful, such as image or speech recognition, language translation or text classification, are solved by means of *supervised learning*. And, up to our knowledge, all the existing tools intended to teach ML in primary and secondary levels are designed to teach this kind of learning. Consequently, we briefly describe how supervised learning work.

Although several algorithms can be chosen to perform supervised learning, all of them have a similar *operating scheme*: first, a large enough set of data that a human agent has to label "by hand" is gathered. Second, these labeled data, called *dataset*, are given as input to some of the supervised ML algorithms, yielding a model able to fit the data correctly, after an intensive computational operation. The model should also be able to give a correct answer when new data is presented. This is called the "generalization power of the model". The output is not always correct, and some wrong answers have to be expected. Hence, in a third step, the model performance is to be evaluated. This is done by feeding the model with test data, evaluating whether more labeled data needs to be added to improve the model. Last, once the model performs good enough, it can be used to develop an "intelligent" application. These steps resemble the way humans learn and generalize from experience, although this has to be understood as a metaphor helping to understand the process.

Taking into account this operational scheme, we propose the following architecture that describes a general tool for learning and teaching ML.

A ML tool should be composed by two components: the *ML Platform* and the

Programming Platform.

The *ML Platform* allows to carry out the above-mentioned ML operating scheme, summarized by the following steps:

1) Build a dataset by adding data examples to be labeled (*training)*,

2) Build a model from this dataset by means of some of the available supervised ML algorithms (*learning*),

3) Evaluate the performance of the model with test data (*evaluating*), and

4) Export the model to a programming platform (*exporting*).

On the other hand, the *Programming Platform* is needed to code an application that is going to use the ML model created by the *ML Platform*. The fourth step of the ML operating scheme interfaces both platforms. The scheme will be very useful to describe and compare tools that have been selected for review in this paper.

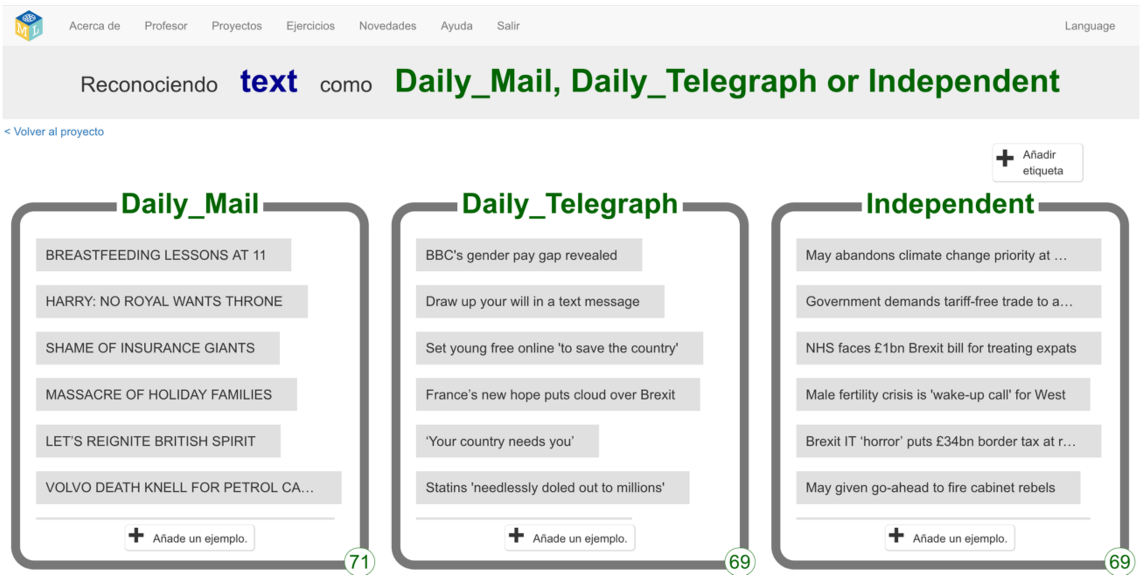
In the following, we will describe and compare tools according to this scheme.

Machine Learning for Kids

Machine Learning for Kids4 (`ML4K`) (Lane, 2018) offers a *ML Platform* and can export its model to three different *Programming Platforms*: `Scratch`, `MIT App Inventor` and `Python`.

The ML Platform is a web application accessible from any standard web browser. Although it can be tested without registration, an account has to be created if data and model storage is wanted. It allows to generate models for text, image, sound and numerical recognition. All four tasks (training, learning, evaluating and exporting) are allowed by this tool.

The training step is achieved by defining buckets associated to each of the labels of the classification problem. Once these buckets have been defined, example data (texts, images, sounds or numbers) can be easily added to their corresponding bucket according to their label (see Figure 1). Therefore, gathering and labeling of data are performed by users, uncovering and revealing one of the crucial process of supervised ML. This fact, as justified in Hitron, Orlev, Wald, Shamir, Erel, & Zuckerman (2019), helps the student to better understand ML.



Reconociendo **text** como **Daily_Mail, Daily_Telegraph or Independent**

Daily_Mail

- BREASTFEEDING LESSONS AT 11
- HARRY: NO ROYAL WANTS THRONE
- SHAME OF INSURANCE GIANTS
- MASSACRE OF HOLIDAY FAMILIES
- LET'S REIGNITE BRITISH SPIRIT
- VOLVO DEATH KNELL FOR PETROL CA...

Daily_Telegraph

- BBC's gender pay gap revealed
- Draw up your will in a text message
- Set young free online 'to save the country'
- France's new hope puts cloud over Brexit
- 'Your country needs you'
- Statins 'needlessly doled out to millions'

Independent

- May abandons climate change priority at ...
- Government demands tariff-free trade to a...
- NHS faces £1bn Brexit bill for treating expats
- Male fertility crisis is 'wake-up call' for West
- Brexit IT 'horror' puts £34bn border tax at r...
- May given go-ahead to fire cabinet rebels

Fig 1. `ML4K` ML Platform training step.

Once the training step has been completed, the dataset is sent off via an HTTP request, to IBM Watson servers to generate the ML model. *Watson* is the Artificial Intelligence Service offered by IBM. This guarantees the quality of the obtained model, but also entails some disadvantages. First, the service is offered with some limitations which only allows for a small number of models. Second, the *learning* step requires too much time and is unpredictable, since requests for model generation are queued up and have to wait their turn to be executed. Third, in addition to the account needed for the `ML4K` administration panel, one more account is required in IBM Watson services and, although

4 https://machinelearningforkids.co.uk/

a free plan is enough to use the tool, this last registration process can be too intricate for people without technical training, as Watson is a service aimed at the professional world of software development.

Moreover, the *training* step is presented as a black box; nothing about the underlying algorithm can be known. This fact, especially once the student has assimilated the operating scheme of supervised ML and wants to gain further insight about the ML algorithm, constitutes a pedagogical drawback.

In the *evaluation* step, its generalization power is evaluated. The model gives as output the predicted label to which the input belongs, and the confidence value representing the hit probability. In order to gain understanding about ML, it is very helpful to check some test data and even use data somehow different to those that have been used in the *training* step. This way students can understand a key concept in supervised ML: the model performance is as good as the training data used. Once the *evaluation* step has finished, the model can be improved if desired by adding new labeled data and rebuilding the model again. That is, iterating through the steps of the operating scheme.

The models are accessed through `HTTP` requests to Watson servers, where they are hosted. In the *exporting* step, three different programming platforms can be used: a modified copy of `Scratch` (fork) with some new blocks able to request Watson models, `MIT App Inventor`[5] with a custom extension to access those models, and a `Python` template with the code required to request the models.

In summary, `ML4K` is a very complete tool to learn ML although the need to create an IBM Watson account and the limitations imposed by that account pose some inconveniences to be used in a school context.

Cognimates

`Cognimates`[6] (Druga, 2018) is a tool that offers both the ML platform, called `Cognimates Studio`, and the programming platform, called `Codelab`, which is a `Scratch` fork.

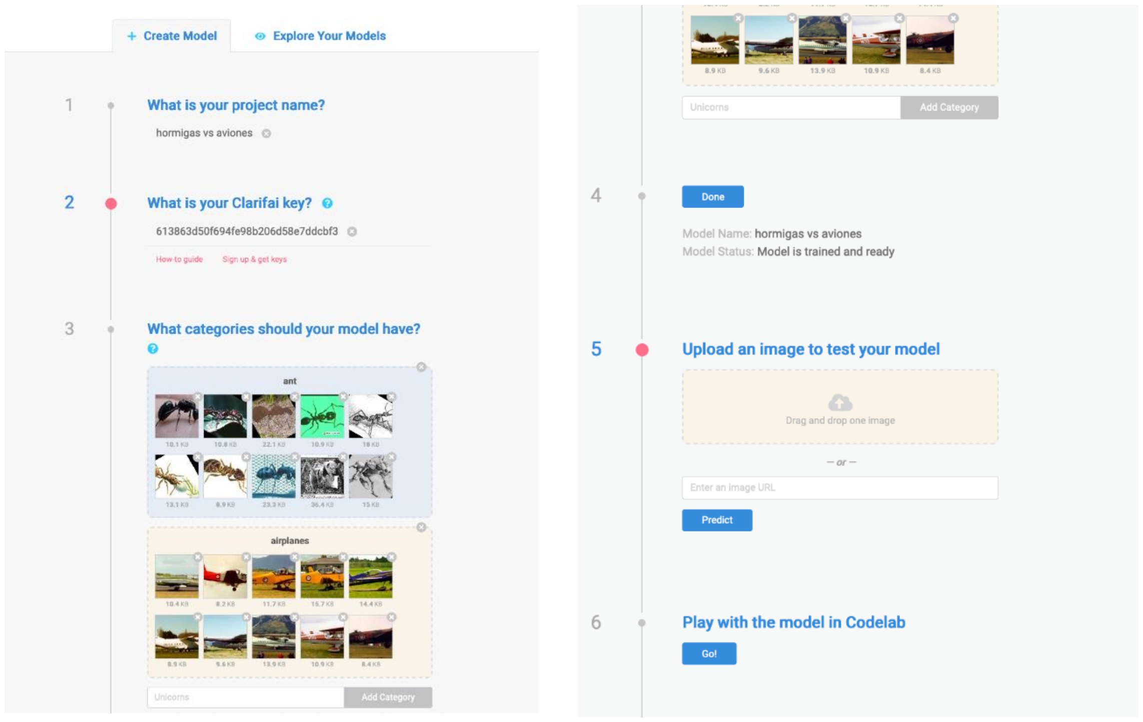
The ML Platform is a web application that can be used without registration. This is a great advantage in the school context, since it is accessed with just one click. It allows to build models for text and image recognition and to get started with some pretrained ML models. In the latter case, the ML operating scheme is skipped and the application can be directly coded. This tool allows to perform the four tasks of the ML operating scheme.

As in `ML4K`, the *training* step is performed by adding labels which lead to buckets where example texts or image have to be added (see Figure 2).

[5] https://appinventor.mit.edu/

[6] http://cognimates.me/home/

The *learning* step is performed by sending the dataset to a ML cloud service. The *uClassify* service[7] is used when dealing with text recognition, while the *clarifAI* service[8] is the ML back-end for image recognition problems. Therefore, creating an account in these services is mandatory if new ML models are to be built. Moreover, one API key is needed to access *clarifAI* and two API keys (reading/writing) are required to access *uClassify*, since they work as REST web services. These API keys are requested when the learner enters the ML `Cognimate Studio` platform. This, as in the case of `ML4K`, is a big drawback, since i) the registration process is not easy for non-experts, and ii) the API key, being a common term among software developers, is confusing for school students and teachers. As in `ML4K`, this step is presented as a black box to the learner, who does not know anything about the applied learning algorithm.



1 What is your project name?
hormigas vs aviones

2 What is your Clarifai key?
613863d50f694fe98b206d58e7ddcbf3
How to guide   Sign up & get keys

3 What categories should your model have?
Unicorns

4 Done
Model Name: hormigas vs aviones
Model Status: Model is trained and ready

5 Upload an image to test your model
Enter an image URL

6 Play with the model in Codelab

Fig 2. `Cognimates` ML Platform *training* step.

The *evaluation* step is carried out by adding some test images or texts, depending on the type of problem. The model yields as response the predicted label to which the data belongs, along with a confidence score expressed as a percentage.

Finally, once the model is working as expected, it can be exported to the programming platform to code the "intelligent" application that will make use of it. A Scratch fork, `Codelab`, is used as the programming platform, which includes ML blocks organized in five sections: *i) "your text blocks", ii) "vision training", iii) "text training", iv) "emotions", and v) "twitter"*. The blocks of the first three sections are used together with the models built with `Cognimates Studio`. The "emotion" section includes some blocks that can be used to make sentimental analysis (text classification based on

[7] https://www.uclassify.com

[8] https://www.clarifai.com

sentiment categories) by means of a pretrained model. As there is no need to train and build any model when using the blocks in this section, coding AI applications is easier. However, this hides the entire operating scheme. The "twitter" blocks can be used to get tweets from Twitter users or hashtags.

In summary, `Cognimates` constitutes a complete solution to work with ML in the classroom, although it requires to register in two cloud ML services to make use of ML platform. This poses a serious drawback when being used by novice people or in a classroom context where every little task can become an unbeatable barrier.

Machine Learning with Snap!

In Kahn & Winters (2017) the use of AI cloud services via a web connection, such as IBM Watson, Microsoft Cognitive services and Amazon AI services, is proposed as a pedagogical resource to teach and learn ML and AI contents. In order to be suitable to be used by children, the technical complexity posed by the API of these services has to be overcome somehow. The challenge addressed in the paper is how to provide interfaces that are much easier to use and yet support most of the functionality of these AI services.

Instead of creating a new programming environment for children, the authors chose to enhance an existing one, adding text-to-speech, speech-to-text and image recognition to `Snap!`[9]. The implementation performed by the authors, however, does not contain an ML platform, but only a programming platform, where new blocks request through `HTTP` these new added services. Therefore, no ML operating scheme is available. Both speech and image recognition are performed by means of pretrained models provided by the AI cloud services. This fact, in our opinion, constitutes a serious drawback of this tool, since hiding all the ML process behind black boxes does not help to get insight and to think about how ML works. Nevertheless, it could be a good starting point to introduce ML and AI contents in the classroom.

Teachable Machine

`Teachable Machine`[10] is a web-based tool aimed to create fast and easy ML models that are accessible to everyone. This application only provides a ML platform and does not require user registration to be used. Two different versions are available.

The first one is designed to quickly show how supervised ML works, allowing learners to create image models able to classify into three classes. The user interface has three different buckets that represent three labels respectively with one button each (as shown in Figure 3). The images, which come from the webcam, are added to each bucket by pressing the corresponding button. Although limited to three labels, the *training* step is thus explicitly carried out by the learner.

As the images are being labeled, the application is training the new model. Hence the *learning* step is taken place continuously and without the learner intervention which, once again, has no knowledge or control over the ML algorithm. Finally, when no button is

[9] https://snap.berkeley.edu/
[10] https://teachablemachine.withgoogle.com/

pressed, the frames taken by the webcam are used as test images. By means of three bar gauges, the application shows the probability of an image belonging to each of the labels. This constitutes the *evaluation* step. No *exporting* step is available.

Since there is no programming platform, nothing can be done with this model, just evaluating and experimenting with the tool to better understand the fundamentals of ML, especially becoming aware of the main role played by data. A little drawback is that the three ML steps --*training*, *learning* and *evaluating*-- take place almost simultaneously, not being clearly separated. This can be a bit confusing when trying to understand the operating scheme of ML. In any case, it is an extremely easy to use and a highly recommendable tool to start learning about ML and AI.

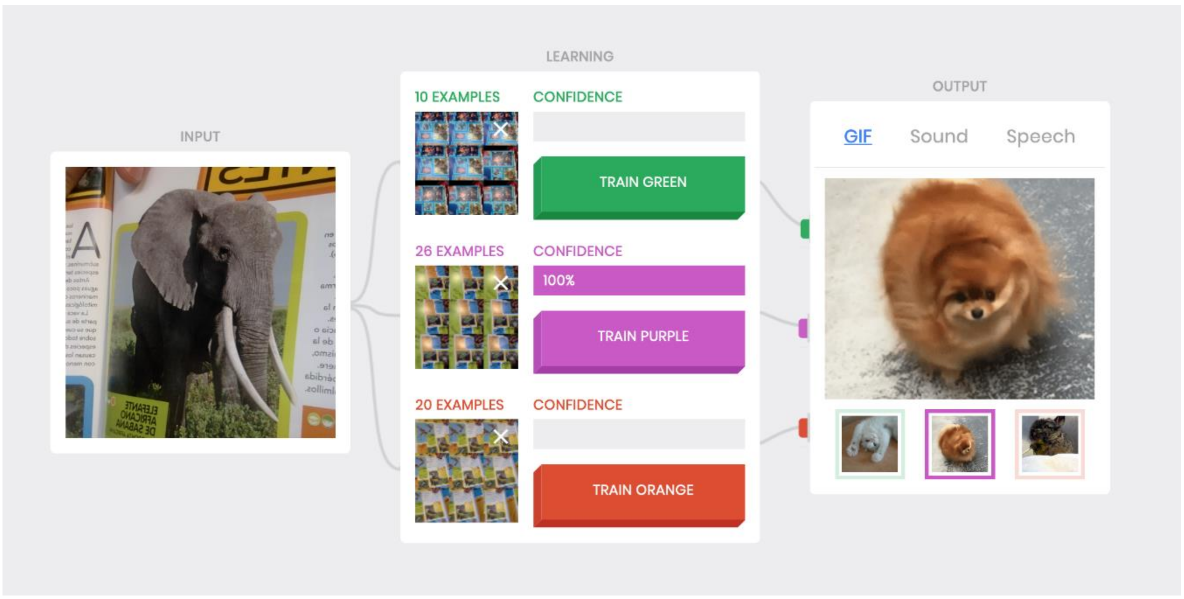


Fig 3. `Teachable Machine` version 1.

The second version (shown in Figure 4) allows to build images, sounds and body postures recognition models. As pointed out in the own application, more types of tasks will be "coming soon". There is no overlap among the ML steps in this second version, which are taking place step by step, clearly revealing the ML operating scheme. In the *training* step the learner can add as many labeled buckets as she wants. Images can be added from the webcam, while the microphone can be used to gather sound. File uploading is also available as an input method, both for images and sounds.

The *learning* step starts when a button labeled as "Train Model" is pressed. Before launching this step, some typical parameters of the underlying algorithm (epoch, batch size and learning rate) can be optionally changed by the learner to explore how they affect the learning process. What these parameters do is clearly explained by means of help boxes. Furthermore, a modal window with some graphic representations of the training process over time is shown when the button labeled "under the hood" is pressed. This is a powerful pedagogical feature to gain insight into the ML process, since some parts of the learning algorithm is revealed to the student. This *uncovering* of the black box is an optional feature which can be exploited by learners once they have a clear understanding of the whole ML operating scheme. Another advantage of this tool is that the learning process takes place locally in the learner web browser -- registration to some AI cloud service is not required. This avoids having to deal with technical concepts such as API key and taking away the registration process. This feature has been possible thanks to the use of `Tensorflow.js`[11] (Simmons, & Holliday 2019), a `javascript` companion library of the popular ML `Python` library `Tensorflow`[12].

After the *training* step has been performed, some test images or sounds can be added using the webcam, the microphone or uploading a file to evaluate the performance of the model. The results are given, as in version 1, by means of a set of bar gauges, one for each label, which indicates the probability that the test data belongs to the class.

Finally, although version 2 of Teachable Machine does not provide a programming platform either, the possibility of exporting the model is available. Once again this has been possible thanks to the use of `Tensorflow.js`. As it is an open source `javascript` library, anyone willing to develop a `javascript` application, could import `Tensorflow.js` to it, and load the model created and exported with `Teachable Machine`. Furthermore, the models can be converted to `Tensorflow` and therefore, can also be used in `Python` projects. Although this is not a suitable task to be done by novice learners, it may allow ML educational tool designers to develop in the future the ML programming platform that `Teachable Machine` lacks.

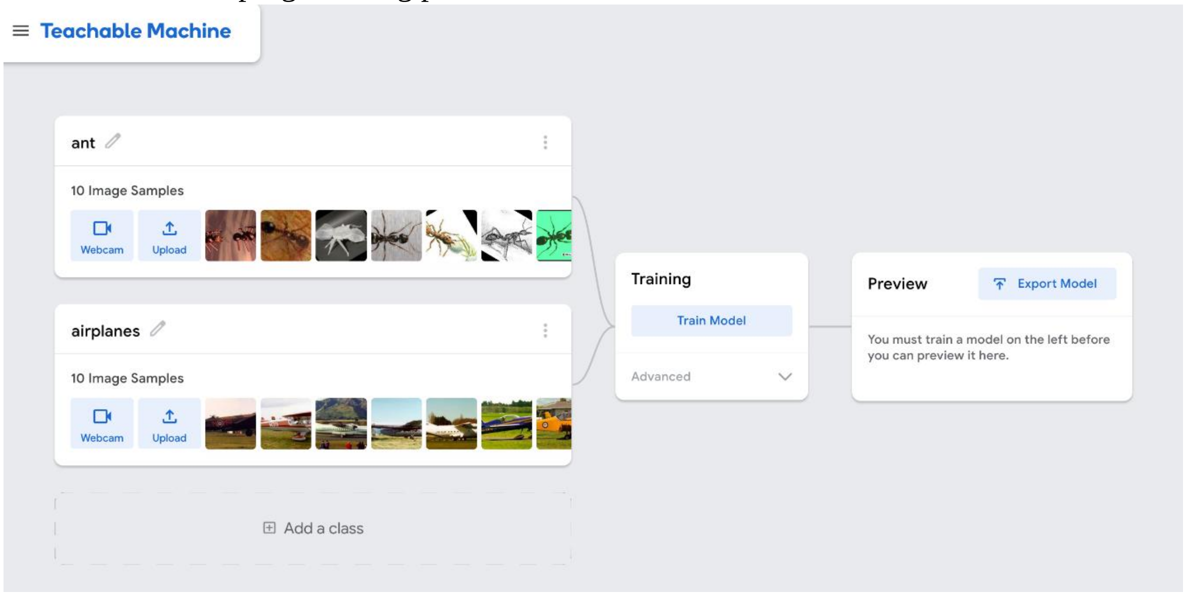


Fig 4. `Teachable Machine` version 2.

In summary, `Teachable Machine` is a perfect election to start with ML, both in the first and the second version. Although it lacks a programming platform to use the generated model in a creative way, it is the most complete and carefully designed ML platform of the whole revision. Since no registration process is required and only a standard web browser is needed to run the tool, students can start using it without any distracting elements and get their first results quickly.

[11] https://www.tensorflow.org/js
[12] https://www.tensorflow.org/

The Personal Image Classifier[13]

The ML tool described in Tang (2019) allows to build image models and provides both the ML platform and the programming platform implemented as an `MIT App Inventor` extension.

The ML platform consists of a web application showing at all times and by means of a timeline every phase of the ML operating scheme (see Figure 5). The current step being executed is pointed out in the timeline and the graphical user interface changes according to the task carried out. As happens with Teachable Machine, this tool also makes use of `Tensorflow.js` as the ML backend, and no registration is required.

The training step is performed, as usual, by adding labeled buckets to which images are gathered from the webcam or by file uploading. Previously exported models can also be uploaded in this step.

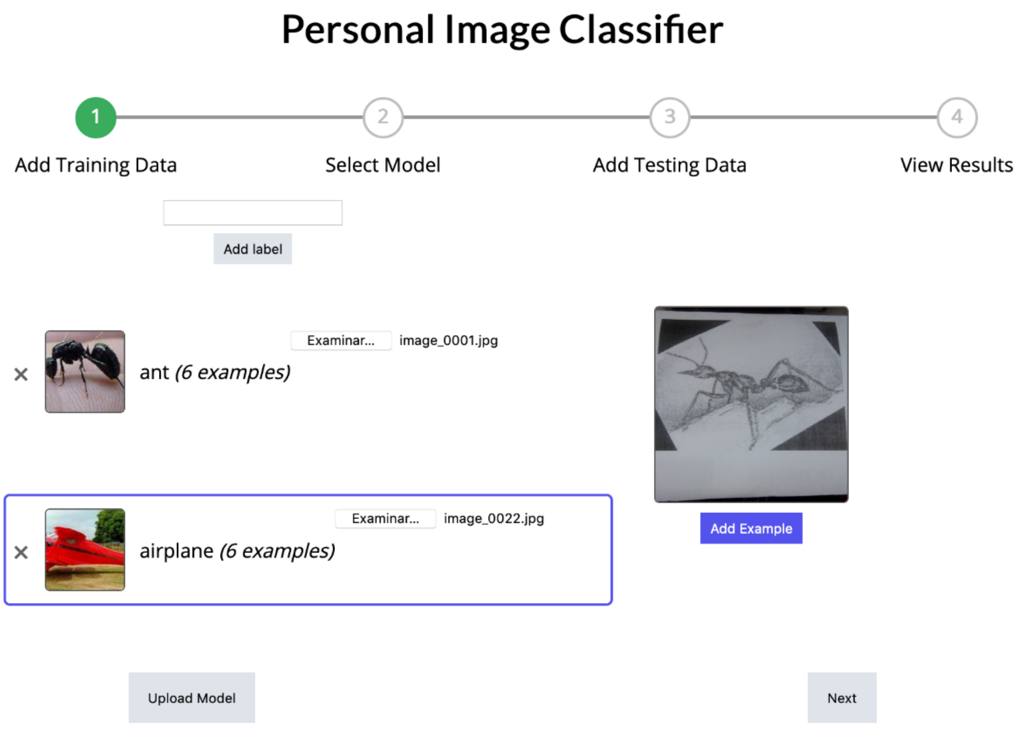


Fig 5. Personal Image Classifier ML Platform *training* step.

Some control over the underlying ANN, which is the ML algorithm that is used, is offered during the learning step. Adding or removing layers of the ANN and changing the parameters as desired can be done. The learner can also change some algorithm parameters, such as the learning rate, epochs, the training data fraction and the type of optimizer used in the learning process. However, the myriad of possibilities comes with too many form input elements, which in addition are labeled in difficult to understand

[13] https://classifier.appinventor.mit.edu/

technical terms. This fact, together with a graphical user interface that is not very friendly, may lead to children and beginners to feel overwhelmed.

The *evaluation* step allows to add at once a bunch of test images to each label. This way the application is able to build a correctness table and a confidence graph as powerful tools to evaluate and analyze the results. The correctness table shows all the testing images and whether they were correctly classified or not. This allows the learner to infer why specific images were correctly classified or not. The confidence graph shows prediction results by label. Three buckets, each one associated with a level of confidence (medium, high and very high) are set, and the predicted images are put into one of these buckets according to the confidence given by the prediction.

Once the model is performant enough, it can be exported as a local downloaded file. Afterwards, it can be imported by uploading the file to `MIT App Inventor` through the `PersonalImageClassifier.aix`[14] extension, which has been developed to create mobile apps using the image generated models. This constitutes the programming platform of the `Personal Image Classifier` tool.

In brief, this is also a very interesting tool to introduce children to ML. However, it is better oriented for high school students, since the graphical user interface is not very user friendly, the *evaluating* step requires some analysis capabilities, the programming platform is more suitable for high school students, and the process of exporting and importing the model requires some training with the file system handling.

[14] https://mit-cml.github.io/extensions/data/extensions/PersonalImageClassifier.aix

Comparison of the tools

Table 1 summarizes and compares the features of the reviewed tools in terms of features of the ML platforms.

	ML4K	Cognim.	Snap!	T.M. 1	T. M. 2	PIC
User Registration	Yes	No	No	No	No	No
ML Backend	Watson IBM	uClassify / Clarifai	Pretrained cloud models	Tensorflow.js	Tensorflow.js	Tensorflow.js
API Keys	Yes	Yes	No	No	No	No
Save data	Yes in cloud	Yes in cloud	No	Yes locally	Yes locally	Yes locally
Save model	Yes in cloud (limited)	Yes in cloud	No	Yes locally	Yes locally	Yes locally
ML Algorithm control	No	No	No	No	Yes	Yes
Evaluation analysis tool	No	No	No	No	Yes	Yes
Types of Problems	Text, images, sounds, numbers	Text, images	Images, sounds	Images	Images, sounds, poses	Images

Table 1. Comparison among features of ML Platforms.

Table 2 summarizes the ML programming platforms allowed by each studied ML tool.

	ML4K	Cognim.	Snap!	T.M. 1	T. M. 2	PIC
Scratch	Yes	Yes	No	No	No	No
MIT App Inventor	Yes	No	No	No	No	Yes
Python	Yes	No	No	No	Yes	No
Snap!	No	No	Yes	No	No	No
Javascript	No	No	No	No	Yes (custom)	No

Table 2. Allowed Programming Platforms.

Introducing `LearningML`

`LearningML` is a free/open source web-based platform aimed to bring ML fundamentals in a practical way to children or people interested in the growing field of AI. This tool provides a *ML Platform* intended to build ML models from example data, and a *Programming Platform* allowing to code creative applications using those models.

This project started as a proof of concept to explore whether ML algorithms could be executed locally in the browser in order to avoid the use of an AI cloud service when designing a tool to learn ML. The development took place after designing at INTEF (the Spanish National Institute for Educational Technologies and Teacher Training) an educational resource aimed to learn ML with `ML4K` (Rodríguez-García, Moreno-León, Román-González, & Robles, 2019).

During the design of the education resource, some drawbacks were detected: firstly, creating an account in whatever AI cloud service was not exempt of difficulties, at least from a classroom context point of view. Secondly, this service is outside the control of the application creators; for instance, today there is a free (as in gratis) plan that can be enough for the purposes of the tool, but conditions may change in the future. Also, even though the service account has been successfully created, password and API keys have to be maintained by the learner, and this is not the best scenario to engage children nor teachers. Last but not least, free plans in AI accounts have some limitations such as the maximum number of projects that can be saved.

Once the proof of concept was finished, a research looking for similar tools was carried out. The results have been summarized in previous sections of this work. Taking into account the *pros* and *cons* of those tools, a new tool, described in this section, is proposed. This tool is a work in progress, therefore many of the features described are not yet available. The final part of this section is devoted to indicating the current status of the tool, pointing out the list of implemented features at current time.

When designing `LearningML`, "*low floor, high ceiling and wide walls*" principles (Resnick et al., 2009) has been taken as a guideline. That is, the tool should be very easy to start with (low floor) but also provide opportunities to create increasingly complex projects over time (high ceiling). Moreover, it should offer the possibility to support different types of projects in order to engage a wide range of interests and learning styles (wide walls). These principles have served as a guideline for the design of the most known block programming platforms such as `Scratch`, `Snap!` and `MIT App Inventor`, and constitute one of the keys for their success. Therefore, we argue that they should be also applied when designing a tool to teach and learn ML.

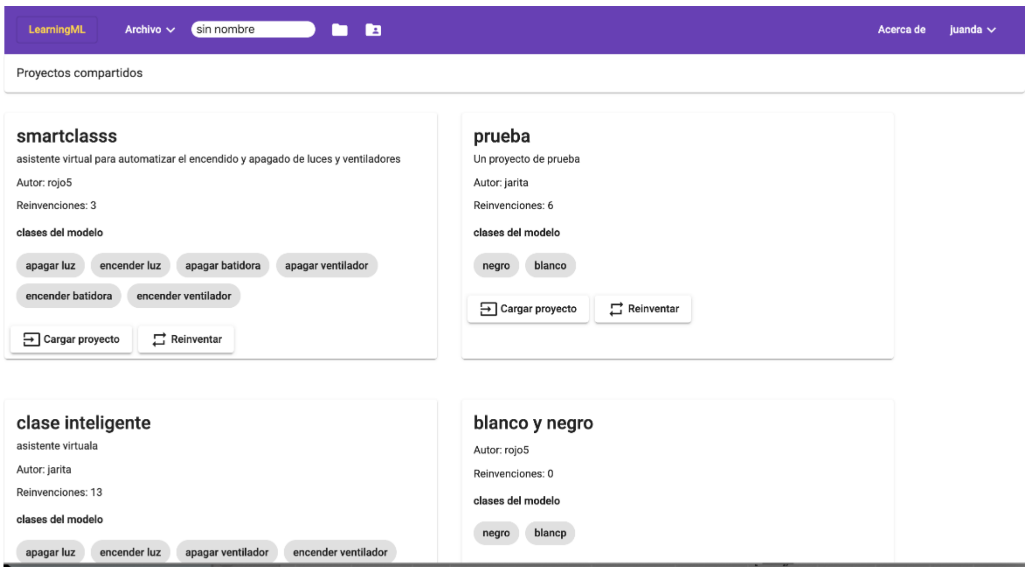
The Machine Learning Platform

The ML Platform[15] is a web-based application developed with `Angular`[16], a powerful `javascript/typescript` framework. The tool allows to build text, image, sound

[15] https://learningml.org/editor
[16] https://angular.io/

and number recognition models. Two operation modes, beginner and advanced mode, can be chosen.

User registration is not required, as one of the desired goals is to allow a quick start (low floor). Nonetheless, the platform allows the learner to register if desired. Registering provides some extra functionalities, but it is not needed to have a complete learning experience. Cloud project storage and sharing are the additional features registration offers (Figure 6). In the future, we want to foster the creation of a community where learners can share their projects. In case the learner decides to register, the only required data are *username*, *email address*, *gender* and *birthdate* as shown in Figure 7. When registering, learners are recommended to choose a *username* that has nothing to do with their actual name. *Email address* is required because the account must be activated and to have some form of communication with the learner in case of problems. Finally, *gender* and *birthdate* are asked for research purposes. If a learner decides not to register, she still can download the project to her disk (and in the future load it back to the application).


LearningML Archivo sin nombre Acerca de juanda

Proyectos compartidos

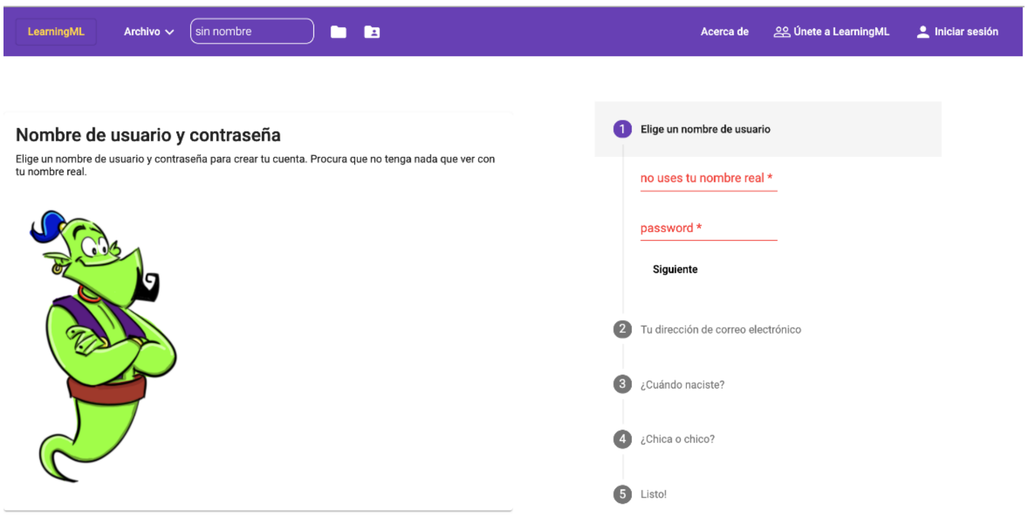
smartclasss
asistente virtual para automatizar el encendido y apagado de luces y ventiladores
Autor: rojo5
Reinvenciones: 3
clases del modelo
apagar luz · encender luz · apagar batidora · apagar ventilador · encender batidora · encender ventilador
Cargar proyecto · Reinventar

prueba
Un proyecto de prueba
Autor: jarita
Reinvenciones: 6
clases del modelo
negro · blanco
Cargar proyecto · Reinventar

clase inteligente
asistente virtuala
Autor: jarita
Reinvenciones: 13
clases del modelo
apagar luz · encender luz · apagar ventilador · encender ventilador

blanco y negro
Autor: rojo5
Reinvenciones: 0
clases del modelo
negro · blancp

Fig. 6 `LearningML` shared projects window.


LearningML Archivo sin nombre Acerca de Únete a LearningML Iniciar sesión

Nombre de usuario y contraseña
Elige un nombre de usuario y contraseña para crear tu cuenta. Procura que no tenga nada que ver con tu nombre real.

1 Elige un nombre de usuario
no uses tu nombre real *
password *
Siguiente
2 Tu dirección de correo electrónico
3 ¿Cuándo naciste?
4 ¿Chica o chico?
5 Listo!

Fig. 7. `LearningML` user registration.

Each of the four tasks allowed by the ML platform --*training*, *learning*, *evaluating* and *exporting*-- are carried out step by step, and pointed out in a timeline that is always visible and that serves both as a reminder of the ML operating scheme and as an indicator of the current step. The learner can navigate this timeline back and forth to make as many iterations as needed to obtain a satisfactory model.

Training step

The *training* step allows the learner to create labeled buckets where data, gathered as examples to feed the ML algorithm, can be added according to the class or label they represent (see Figure 8). Both the buckets and the examples added to them can be removed in case of error. Texts are provided through the keyboard or by means of the microphone using the speech-to-text feature of modern web browsers. Images are gathered from the webcam or by file uploading. In this last case, a bunch of images can be added at once to speed up the process. Sounds can be recorded from the microphone or by file uploading. Once again, several files can be uploaded at once. Finally, number datasets are entered from the keyboard or by uploading a CSV (comma separated values) file.

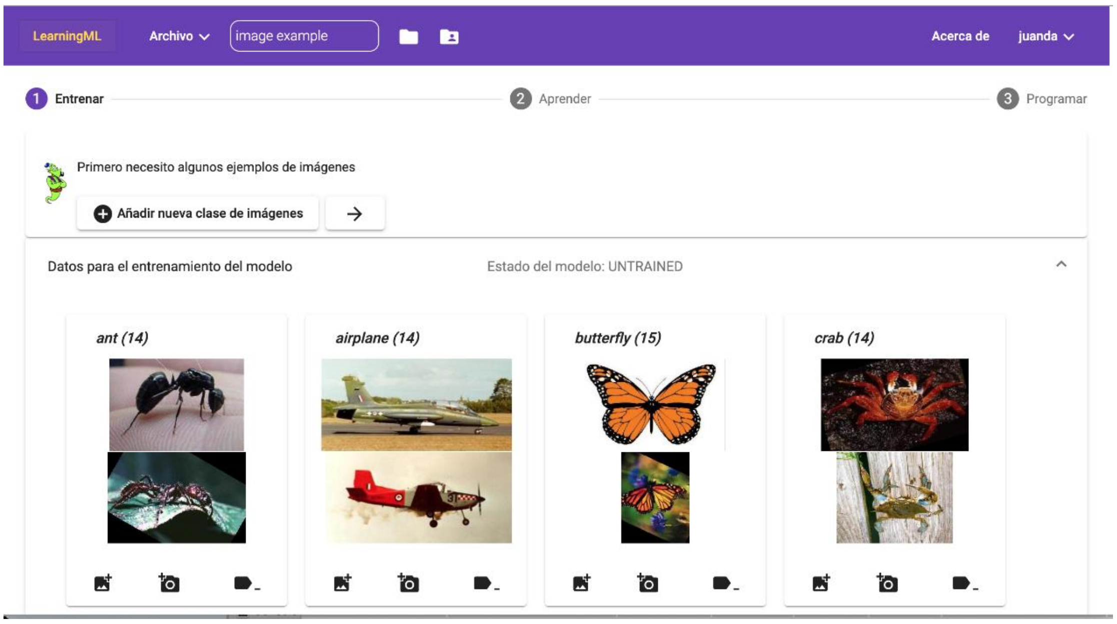

LearningML
Archivo
image example
Acerca de
juanda
1 Entrenar
2 Aprender
3 Programar
Primero necesito algunos ejemplos de imágenes
Añadir nueva clase de imágenes
Datos para el entrenamiento del modelo
Estado del modelo: UNTRAINED
ant (14)
airplane (14)
butterfly (15)
crab (14)

Fig. 8. `LearningML` *training* step with image problem.

Learning step

The *learning* step can be started once some example data have been labeled (see Figure 9). The ML algorithm to build a model from example data is executed locally in the learner's browser. The `Tensorflow.js` library is used for this purpose. An ANN has been chosen as the underlying ML technique. Besides, for text, image and sound recognition, a technique known as *transfer learning* is used to obtain better results empowering the classification abilities of the model being constructed. In brief, with transfer learning a model developed for a task is reused as the starting point for a model on a second task. There exist open accessible models, such as *mobilenet* (Howard et al., 2017) for image recognition, which have been pre trained with thousands or even millions of labeled data. While these models probably will not fit well with the problem being solved, some of their internal layers can be used as the input of a second model. As this filtered input has been somehow discriminated by the first powerful model, the second model can be designed to give as output the labels chosen by the learner and also have a much simpler architecture. If the *advanced* mode has been chosen, the parameters of this second simpler model are revealed to the learner. The learner can try to modify them to see their effect on the performance of the model. The following parameters can be changed: number of internal layers, number of neurons in those layers, epoch, batch size and learning rate. In the *beginner* mode, the learning process is achieved transparently.

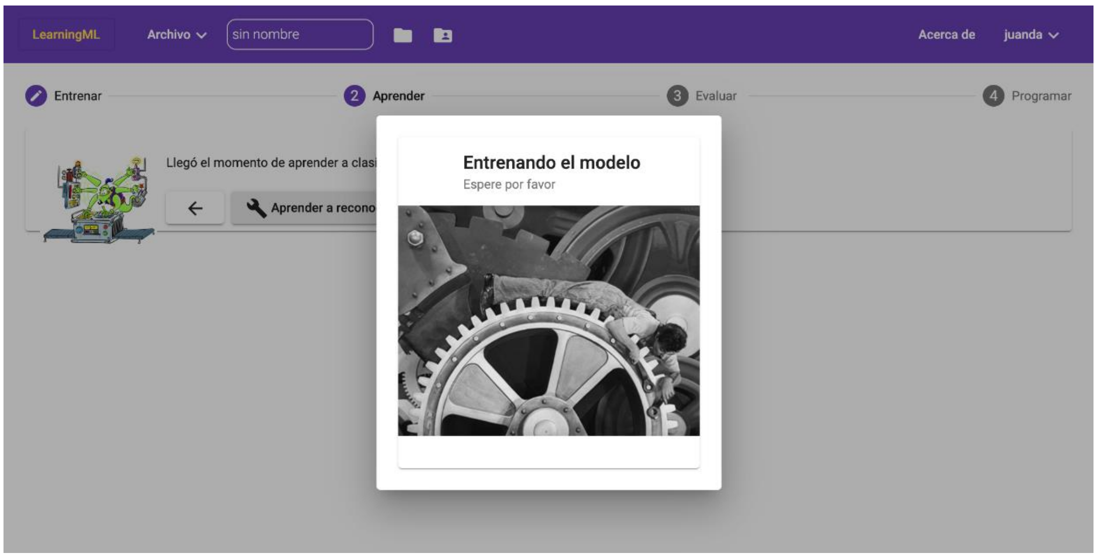
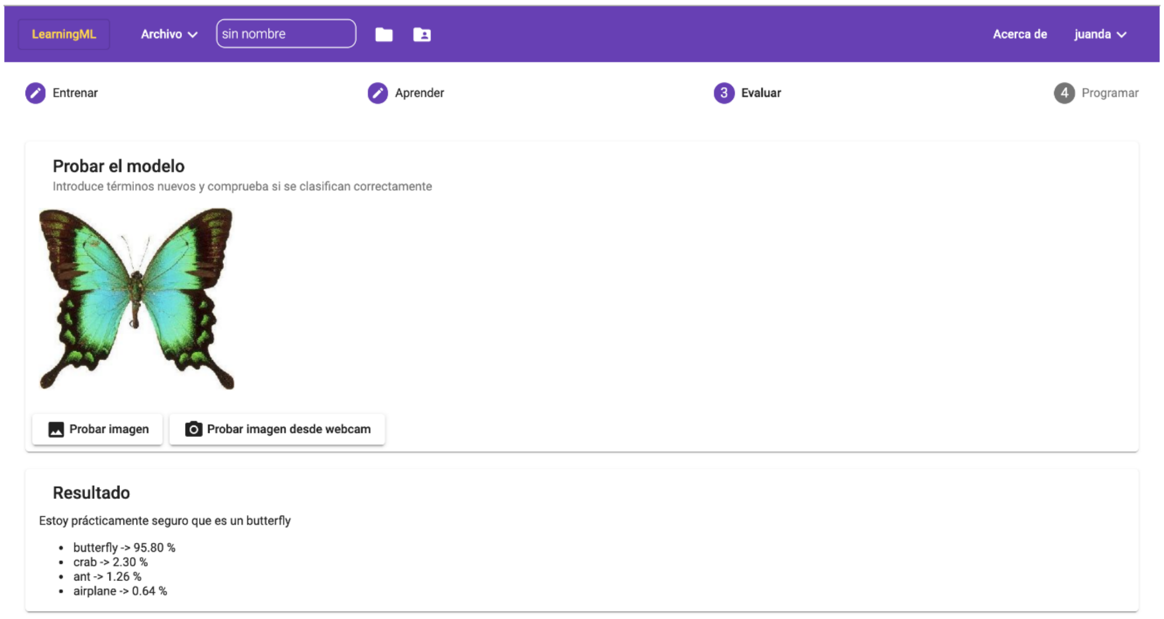


Fig. 9. `LearningML` *learning* step.

Evaluation step

In the *evaluation* step the learner can check how well the model is performing by providing it with test data (see Figure 10). These data are provided by means of the same input devices used in the *training* step: keyboard, microphone, webcam or file uploading according to the type of task. When the *beginner* mode has been chosen, the classification result is provided by means of a phrase expressing what label has been selected by the model together with its degree of confidence. For instance, in a cats and dogs image classification problem, possible output results could be: “I’m not sure at all, but it could be a dog”, “I think it is a dog, isn’t it?”, “it is probably a dog” or “I am almost sure it is a dog”. The goal is to make the probabilistic nature of ML explicit, even for children who have not yet learned probability concepts formally. Indeed, this could be a good starting point to introduce probability to children. On the other hand, when the *advanced* mode is activated, a set of gauge bars are added to the previous sentences indicating the numerical probability, expressed as a percentage, given by the model that the entry belongs to. This numerical expression of the confidence can be also used to deepen the concept of probability and to reinforce the idea that ML solutions are not infallible.

A confusion matrix is also shown in the *advanced* mode. It offers a clear, easy and intuitive way to visualize the performance of a supervised learning algorithm. The confusion matrix consists of a table where columns represent the predicted labels on a test dataset and the rows the real labels. Non-negative values in non-diagonal places of the matrix indicate the presence of prediction errors. In order to build a confusion matrix, a bunch of test data are needed, therefore multiple file uploading can be used to provide them all at once. Evaluation is a very important step to gain insight about how ML works, since the probabilistic nature of inference methods is revealed, and to reinforce the idea that ML models are as good as the example data used to train them. This step also promotes iterations as a means to improve the results, which according to constructivist school (Hein, 2016), has a profound effect on the child learning process.



Probar el modelo

Introduce términos nuevos y comprueba si se clasifican correctamente

Resultado

Estoy prácticamente seguro que es un butterfly

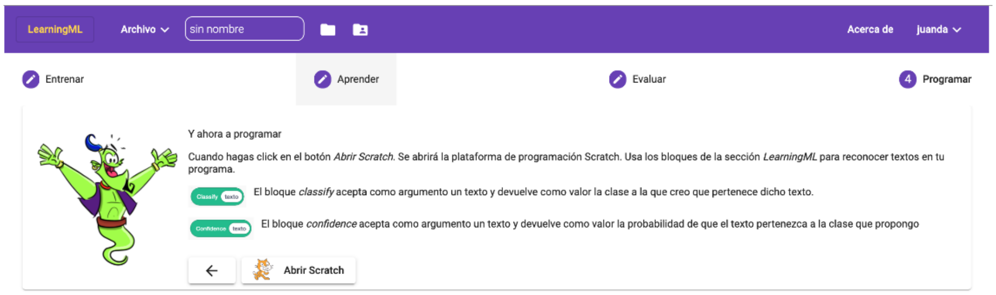
- butterfly -> 95.80 %
- crab -> 2.30 %
- ant -> 1.26 %
- airplane -> 0.64 %

Fig. 10. `LearningML` *evaluating* step.

Exporting step

`LearningML` projects can be locally downloaded to disk and also, when the learner is logged-in, stored in the cloud for later retrieval. A project is composed of the dataset gathered by the learner during the training step and, if the *learning* step has been executed at least once, by the built model. `Tensorflow.js` is used to export the model. This will allow any programming platform capable of loading and managing `Tensorflow.js` models, such as `MIT App Inventor` with the `Personal Image Classifier` extension or a custom `javascript` application, to import models that have been built with `LearningML`, being in line with the "wide walls" principle.

However, as described in the next subsection, `LearningML` has its own Programming Platform to code AI applications. The normal working flow with `LearningML` is to export the model directly to that platform (Figure 11). At the press of a button, the programming platform is launched with the model automatically loaded, and with specific programming blocks aimed to manage and make use of that model. This process is transparent to the learner who can quickly start to code the ML application. This way, the file export/import procedure is avoided and “the floor of the tool is lowered”.



Y ahora a programar

Cuando hagas click en el botón *Abrir Scratch*. Se abrirá la plataforma de programación Scratch. Usa los bloques de la sección *LearningML* para reconocer textos en tu programa.

El bloque *classify* acepta como argumento un texto y devuelve como valor la clase a la que creo que pertenece dicho texto.

El bloque *confidence* acepta como argumento un texto y devuelve como valor la probabilidad de que el texto pertenezca a la clase que propongo

Fig. 11. `LearningML` *exporting* step.

The Programming Platform

`Scratch 3.0` has been developed as a `javascript` web application made with the `React`[17] framework. Furthermore, it has been released with a free (as in libre) BSD license and the code can be publicly downloaded from a GitHub repository[18]. There is no doubt that `Scratch` is nowadays the most-used block programming platform. Therefore, it has been chosen as the programming platform for `LearningML`.

In any case, the official instance of `Scratch` does not provide programming blocks aimed to work with `Tensorflow.js` models, or to perform any kind of ML coding. Hence, a copy (fork) of Scratch has been made[19] and the required ML features have been added to it. This section is devoted to describing these new programming blocks.

Text recognition blocks

Table 3 offers a list of blocks aimed for code texts recognition applications.

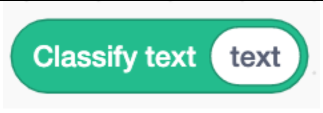
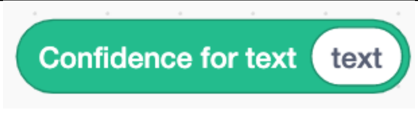
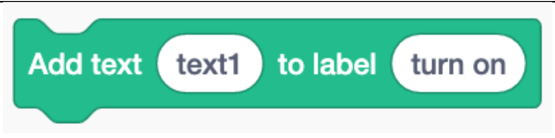
Block	Description
	Reporter block that returns the label assigned by the ML model to the argument `text`.
	Reporter block that returns the confidence, expressed as a percentage, associated with the classification of argument `text`.
	Command block that adds the first argument as an example belonging to the label given by the second argument. This block allows gathering more texts to the dataset.

Table 3. Texts recognition blocks

[17] https://es.reactjs.org/
[18] https://github.com/LLK
[19] https://learningml.org/scratch

Image recognition blocks

Table 4 offers a list of blocks aimed for code images recognition applications.

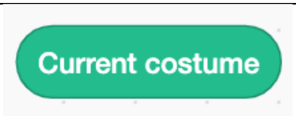
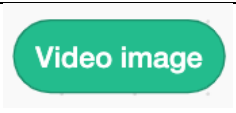
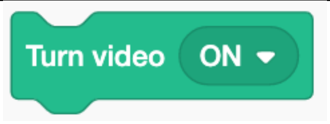
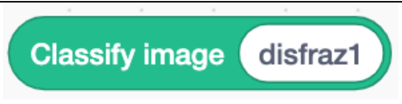
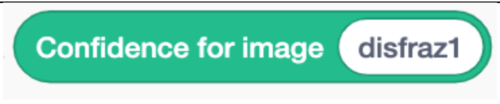
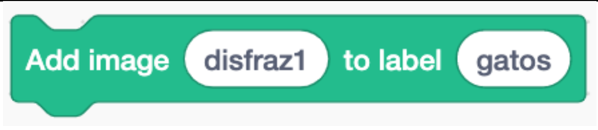
Block	Description
 Current costume	Reporter block that returns the image corresponding to the current costume object.
 Video image	Reporter block that returns the last image taken by the webcam. It only works when the webcam is on.
 Turn video ON	Command block to turn on/off the webcam.
 Classify image disfraz1	Reporter block that returns the label assigned by the ML model to its argument, which can be the current costume or the last frame taken by the webcam. Therefore, the argument of this reporter must be one of the two first blocks of this table.
 Confidence for image disfraz1	Reporter block that returns the confidence, expressed as a percentage, associated with the classification of the image given by its argument. As in the previous block the argument must be one of the two first blocks of this table.
 Add image disfraz1 to label gatos	Command block that adds the image given by the first argument as an example belonging to the label given by the second argument. This block allows gathering more images to the dataset.

Table 4. Images recognition blocks.

Sound recognition blocks

Table 5 offers the blocks aimed to code sounds recognition applications.

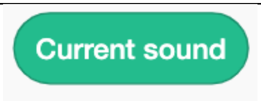
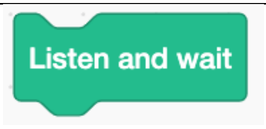
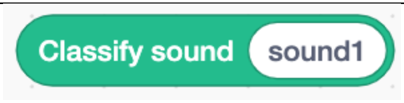
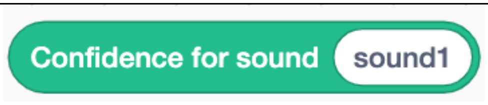
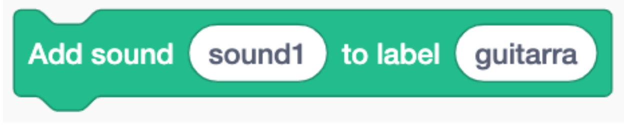
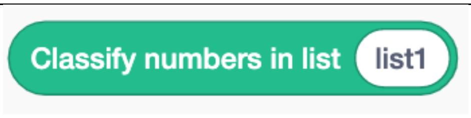
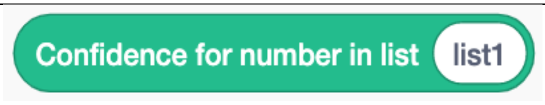
Block	Description
 Current sound	Reporter block that returns the last played sound.
 Listen and wait	Reporter block that turns on the microphone and records a sound. The recorded sound is added to the object list of sound and referenced as “current sound”.
 Classify sound sound1	Reporter block that returns the label assigned by the ML model to the sound referenced by its argument.
 Confidence for sound sound1	Reporter block that returns the confidence, expressed as a percentage, associated with the classification of the sound referenced by its argument.
 Add sound sound1 to label guitarra	Command block that adds the sound referenced by the first argument as an example belonging to the label given by the second argument. This block allows gathering more images to the dataset.

Table 5. Sounds recognition blocks.

Number recognition blocks

Table 6 shows the blocks aimed to code numbers recognition applications.

Block	Description
 Classify numbers in list list1	Reporter block that returns the label assigned by the ML model to the list of numbers referenced by its argument.
 Confidence for number in list list1	Reporter block that returns the confidence, expressed as a percentage, associated with the classification of the list of numbers referenced by its

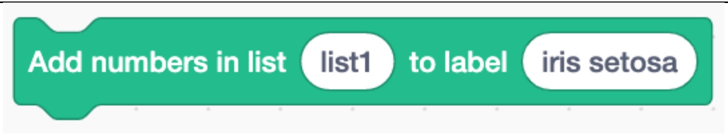
	argument.
 Add numbers in list list1 to label iris setosa	Command block that adds the list of numbers referenced by the first argument as an example belonging to the label given by the second argument. This block allows gathering more lists of numbers to the dataset.

Table 6. Numbers recognition blocks.

General ML blocks

Table 7 lists the blocks aimed to control the ML algorithm.

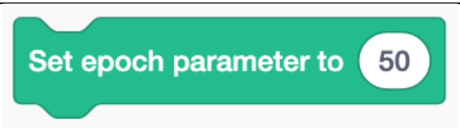
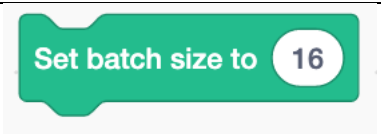
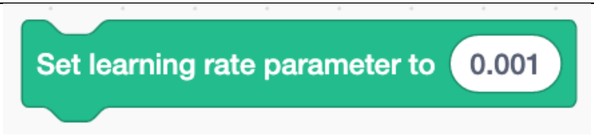
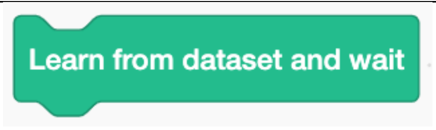
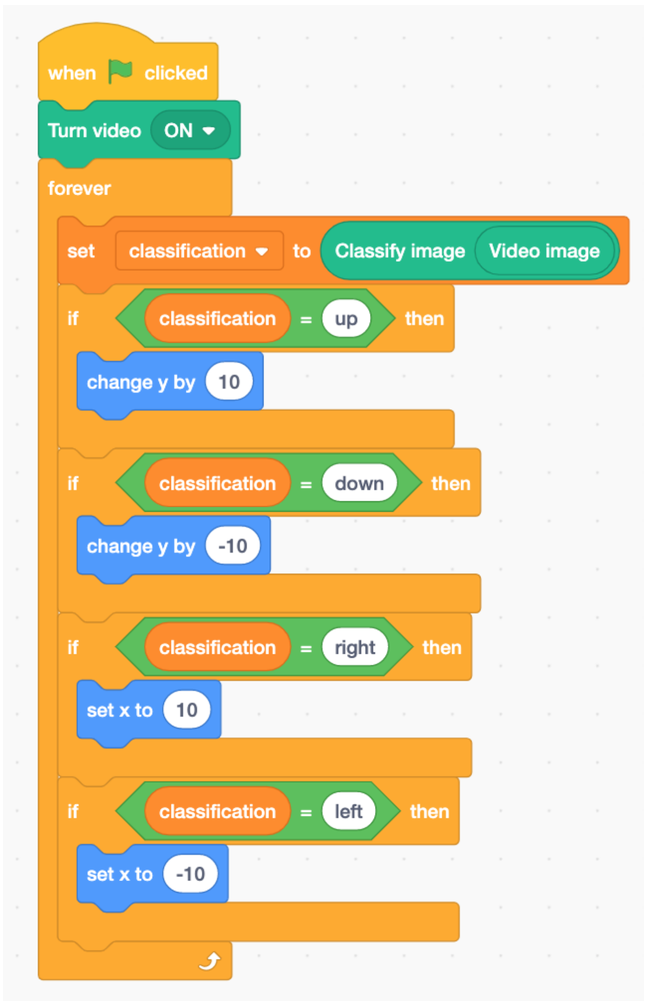
Block	Description
 Set epoch parameter to 50	Command block to set the *epoch* parameter of an ML algorithm.
 Set batch size to 16	Command block to set the *batch size* parameter of an ML algorithm.
 Set learning rate parameter to 0.001	Command block to set the learning *rate* parameter of an ML algorithm.
 Learn from dataset and wait	Command block that executes the learning algorithm providing a new model.

Table 7. General ML blocks.

Combining these new blocks with the ones Scratch provides by default, *awesome* AI applications and video games can be coded. The code presented in Figure 12 shows a very easy and direct application in which the object movement is controlled by the images coming from the webcam. The model should have been trained with four types of images, each type representing one of the four directions: up, down, left and right.

Current status of `LearningML`

At the time of writing this paper, only texts and image recognition features have been implemented, which is enough to start testing the tool in school settings. Also, *beginner*/*advanced* modes are not yet available. A web page[20] devoted to promoting the tool, which provides help tutorials and AI activities with the intention of encouraging teachers to introduce practical AI content in their classrooms, has also been developed.



```
when 🏳 clicked
Turn video ON ▾
forever
    set classification ▾ to Classify image Video image
    if classification = up then
        change y by 10
    if classification = down then
        change y by -10
    if classification = right then
        set x to 10
    if classification = left then
        set x to -10
```

Fig. 12. Scratch program to control an object by means of images captured by the webcam.

A preliminary intervention: testing a pilot workshop with students

In order to test whether `LearningML` can be a useful tool to teach AI, ML and its societal implications, a preliminary intervention was carried out. This was organized in two small workshops, with a total of 14 participating students. The workshops were held at two Computer Science technical colleges in Seville (Spain) between December 2019 and February 2020. Participating students were adults about 20 years old with prior programming experience but no previous training on AI.

[20] https://learningml.org

The structure of the workshops was the same in both cases: first, a pre-test questionnaire on AI knowledge was filled out by students; second, an introduction about AI was presented by one of the researchers; third, students programmed a project with `LearningML` to solve a challenge; fourth and finally, the post-test was filled by the students. The total length of the workshops was 2 hours.

The results of the pre and post-tests are discussed in the following subsections. Both the questionnaire and CSV files with the anonymized pre and post students' answers are provided in the replication package of the paper[21].

First set of questions

Questions 2, 3, 4 and 5 in the questionnaire are Likert-style. Therefore, two new variables -`Total pre 1` and `Total post 1`- were computed as the sum of the scores in those four questions in the pre and post-test, respectively. Since each question was scored with 1-5 points, in terms of its proximity to the right answer, the variation range for both new variables is thus, between 4 and 20.

The count, mean, standard deviation, minimum, first quartile, median, third quartile, and maximum values are shown in Table 8, and illustrated in the boxplot in Figure 13. As can be seen, there was an increase in the results, as the mean in the pre-test is 12.79, while the mean in the post-test is 14.50.

	Before LML (`Total pre 1`)	**After LML (`Total post 1`)**
count	14	14
mean	12.786	14.500
std	2.723	2.345
min	9.000	11.000
25%	10.250	12.500
50%	13.000	14.500
75%	14.750	16.000
max	17.000	18.000

Table 8. Results in the first set of questions before and after the intervention with `LearningML`.

[21] https://github.com/kgblll/kgblll-ReplicationPackage-2020-RED

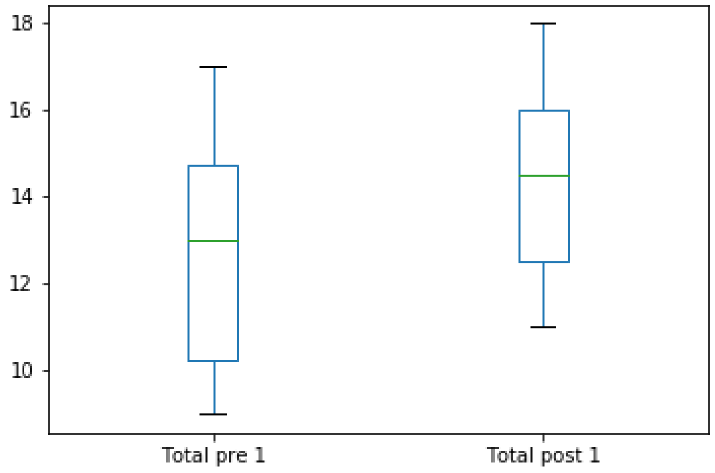


Fig. 13. Boxplot of the results in the first set of questions before and after the intervention with `LearningML`

Since the difference between the two conditions is normally distributed, according to the Shapiro-Wilk test (W = 0.956, p-value = 0.672), we performed a t-test for paired samples, establishing a 95% confidence level (α = 0.05) for our statistical decisions. The results of the t-test for paired data (statistic=-3.618, p-value=0.003) indicate that the null hypothesis of equality of means is rejected and we can therefore state that there are significant differences between the pre-test and post-test.

This shows that the use of `LearningML` helped the participating students to enhance their knowledge on AI. But in order to assess the real impact of using `LearningML`, the effect size of the experiment must also be calculated (Cohen, 1990). For this set of questions, the effect size was 0.675, which is considered a moderate effect (Ellis, 2010) that according to the "influence barometer" (Hattie, 2009) indicates that the educational intervention has fulfilled the desired goals.

Although this analysis involves a small number of students, the results are promising and provide a strong stimulus to continue working on the validation and improvement of the tool.

Second set of questions

Questions 7, 8 and 9 in the questionnaire are multiple choice with only one correct answer. Consequently, we computed two new variables -`Total pre 2` and `Total post 2`- as the sum of the scores in these three questions in the pre and post-test, respectively. Since each question was scored with 0 or 1 points, the variation range for both new variables is therefore between 0 and 3.

Table 9 shows the count, mean, standard deviation, minimum, first quartile, median, third quartile, and maximum values. As can be seen, there was an increase in the results, as the mean of the pre-test is 1.57, while the mean of the post-test is 2.29. This is illustrated in Figure 14, which presents the boxplot of the statistics.

	Before LML (`Total pre 2`)	**After LML (`Total post 2`)**
count	14	14
mean	1.571	2.286
std	1.016	0.611
min	0.000	1.000
25%	1.000	2.000
50%	1.500	2.000
75%	2.000	3.000
max	3.000	3.000

Table 9. Results in the second set of questions before and after the intervention with `LearningML`.

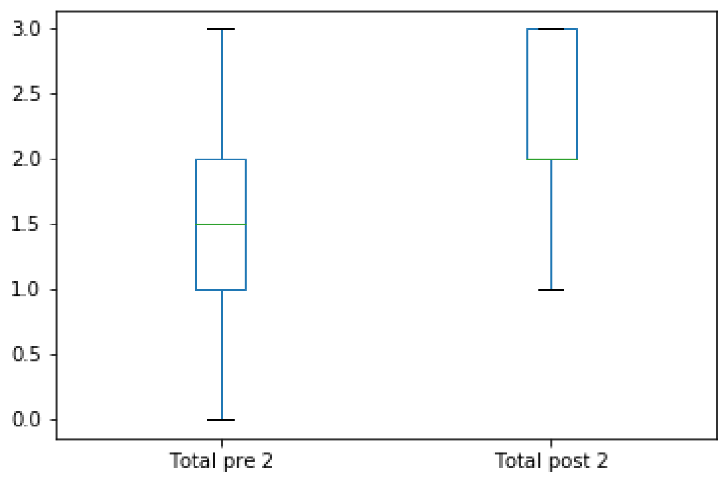


Fig. 14. Boxplot of the results in the first set of questions before and after the intervention with `LearningML`.

In this case the Shapiro-Wilk test for normality (W = 0.796, p-value = 0.004) shows that the difference between the two conditions is not normally distributed. As a consequence, we used the Wilcoxon signed-rank Test (p-value=0.008), which indicates that the null hypothesis of equality of means is rejected and proves significant differences between pre and post-test.

The effect size for the second set of questions was 0.852, which is considered a big effect (Ellis, 2010) and, consequently, indicates that the educational intervention has fulfilled the desired goals (Hattie, 2009).

Open questions

The questionnaire also included two open questions, asking participants to propose a definition of AI and discuss its potential dangers. We planned to perform a text-network analysis (Paranyushkin, 2019) with students' answers, but the small number of cases stopped us from doing so.

In any case, the results of previous subsections can be illustrated by studying the evolution in the definitions provided by two of the subjects:

Student 1
Pre-test: *Artificial intelligence is a type of intelligence that is carried out with robots*.
Post-test: *Artificial Intelligence is a technology that tries to get machines to learn through experience and trial and error.*

Student 2
Pre-test: *The ability to emulate the human mind in a machine*
Post-test: *When a machine has the capacity to perform actions for which we consider it necessary to reason and think like a human.*

Conclusions and future work

In order to ensure that the citizens of the future understand the implications of AI and maximize its potential opportunities, it is critical to provide them with the necessary skills to understand AI and its different areas of action. Moreover, given the rising demand in STEM and AI related jobs, the inclusion of AI contents in education might be an effective movement to awaken vocation in young people.

As AI is closely related to programming, the development of hands-on AI projects seems to be a powerful way to foster CT skills as well as a means to learn AI and ML fundamentals. However, existing programming platforms, as `Scratch`, are not enough to develop complete projects revealing AI techniques. Consequently, researchers and educators have started to develop new tools during the last years, which are mainly focused on teaching and learning of ML and, specifically, on supervised learning.

The nucleus of this paper is constituted by the introduction of `LearningML,` a new tool, designed and developed by the authors, aimed to teach and learn the supervised learning technique. The tool is composed of an ML platform where ML models from data gathered and labeled by the learner are built, and a programming platform where the learner can develop creative applications able to recognize texts, images, sounds or numbers by means of these ML models. All pros and cons found in previous tools along with the "low floor, high ceiling and wide wall" principles have been taken as design guidelines.

Tools aimed to develop CT skills, as `Scratch`, help to engage the student with all of the commonly accepted dimensions of CT: *concepts*, *practices* and *perspectives*. Among these dimensions, seven concepts (*sequences*, *loops*, *events*, *parallelism*, *conditionals*, *operators* and *data*), four sets of practices (*being incremental and iterative*, *testing and debugging*, *reusing and remixing*, and *abstracting and modularizing*) and three perspectives (*expressing*, *connecting* and *questioning*) have been identified as essential elements (Brennan & Resnick, 2012). Since developing practical AI projects implies a coding phase, all of them can be exercised with `LearningML`.

Moreover, the ML model generation performed before coding through the ML platform of `LearningML`, enables students to engage with some new elements that should be added to this CT framework. *Classification*, *prediction* and *generation* appear as concepts that mean different ways in which results are presented by AI models. *Training*, *validating* and *testing* emerge as new practices, since they represent the necessary tasks to be carried out in order to build ML models while they are being generated. Lastly, *evaluating* the final product that integrates the AI model, aims at discovering and questioning to what extent such product achieves its goal, thus providing a new perspective (Brummelen, Shen, & Patton 2019).

	Concepts	Practices	Perspectives
Brennan & Resnick CT Framework dimensions	Sequences Loops Events Parallelism Conditionals Operators Data	Being incremental and iterative Testing and debugging Reusing and remixing Abstracting and modularizing	Expressing Connecting Questioning
AI extension of Brennan & Resnick CT Framework dimensions	Classification Prediction Generation	Training Validating Testing	Evaluating

Table 10. Brennan & Resnick's CT framework (2012) extended for AI (Brummelen, Shen, & Patton 2019).

The results of the intervention seem very promising, especially taking into account that this effect was experienced after just 2 hours of treatment, which highlights the impact that `LearningML` had on the learners and draws attention to its potential as a tool to foster AI and ML knowledge. However, different results may be achieved with younger students and with learners with no prior programming experience. Therefore, these results may be understood just as a validation of the pilot intervention, but more interventions with a more diverse type of students must be carried out in order to fully validate the tool.

Indeed, at the time of writing this paper, the research group behind `LearningML` is working on a strategy to validate whether the tool is an appropriate and effective educational resource. First, some practical AI activities ready to be used in the classroom are being designed and made available to all those who wish to test the tool on the `LearningML` website. These activities will cover all levels of primary and secondary education and will be implemented in the near future through a series of workshops with the collaboration of some schools. The goals of these workshops are to assess both its *face validity*, testing whether the tool can be fluently used by children, and its *instructional validity*, testing if some knowledge about AI and ML contents are acquired by them after the intervention, and if so, to what extent. Therefore, a psychometric test specifically designed to this end has to be elaborated.

If these experimental interventions show positive results, the roadmap for the development of the tool is as follows. We plan to include the possibility of working with both unsupervised and reinforcement learning, the other two members of the ML family. Learners should also be able to choose which learning algorithm, among some of the best known, will be used to build the model, and to tune their main parameters, so learners could study the effect of the changes on the final model.

Presentación del manuscrito: 15 de enero de 2020
Fecha de aprobación: 27 de marzo de 2020
Fecha de publicación: 30 de abril de 2020

Rodríguez-García, J.D., Moreno-León, J., Román-González, M., and Robles, G. (2020). LearningML: A Tool to Foster Computational Thinking Skills Through Practical Artificial Intelligence Projects. *RED. Revista Educación a Distancia, 20*(63). http://dx.doi.org/10.6018/red.410121

Financing

This work has been co-funded by the Madrid Regional Government, through the project e-Madrid-CM (P2018/TCS-4307). The e-Madrid-CM project is also co-financed by the Structural Funds (FSE and FEDER).

References

Ali, S., Payne, B. H., Williams, R., Park, H. W., & Breazeal, C. (2019). *Constructionism, Ethics, and Creativity: Developing Primary and Middle School Artificial Intelligence Education*. 4. *International Workshop on Education in Artificial Intelligence K-12 (EDUAI '19)*. Palo Alto, CA, USA. *Proceedings of IJCAI 2019*.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association. Vancouver: American Educational Research Association., 1*, 25.

Brummelen, J. V., Shen, J. H., & Patton, E. W. (2019). The Popstar, the Poet, and the Grinch: Relating Artificial Intelligence to the Computational Thinking Framework with Block-based Coding. *Proceedings of International Conference* *on Computational Thinking Education 2019. Hong Kong: The Education University of Hong Kong*, 2.

Burton, E., Goldsmith, J., Koenig, S., Kuipers, B., Mattei, N., & Walsh, T. (2017). Ethical Considerations in Artificial Intelligence Courses. *AI Magazine*, *38*(2), 22–34. https://doi.org/10.1609/aimag.v38i2.2731

Cohen, J. (1990). Things I have learned (so far). American Psychologist, 45(12), 1304-1312. doi:10.1037/0003-066x.45.12.1304

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of Oz studies—Why and how. *Knowledge-Based Systems*, *6*(4), 258–266. https://doi.org/10.1016/0950-7051(93)90017-N

Druga, S. (2018). Growing up with AI *Cognimates:* From coding to teaching machines. Massachusetts Institute of Technology (Doctoral dissertation).

Ellis, P. (2010). The essential guide to effect sizes. Cambridge: Cambridge University Press.

Estevez, J., Garate, G., Guede, J. L., & Graña, M. (2019). Using Scratch to Teach Undergraduate Students' Skills on Artificial Intelligence. *ArXiv:1904.00296 [Cs]*. http://arxiv.org/abs/1904.00296

Estevez, J., Garate, G., & Graña, M. (2019). Gentle Introduction to Artificial Intelligence for High-School Students Using Scratch. *IEEE Access*, *7*, 179027–179036. https://doi.org/10.1109/ACCESS.2019.2956136

Hattie, J. (2009). Visible learning. New York: Routledge.

Hein, G. (2016, January 6). *Constructivist Learning Theory*. Exploratorium. https://www.exploratorium.edu/education/ifi/constructivist-learning

Hitron, T., Orlev, Y., Wald, I., Shamir, A., Erel, H., & Zuckerman, O. (2019). Can Children Understand Machine Learning Concepts?: The Effect of Uncovering Black Boxes. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*, 1–11. https://doi.org/10.1145/3290605.3300645

Hitron, T., Wald, I., Erel, H., & Zuckerman, O. (2018). Introducing Children to Machine Learning Concepts Through Hands-on Experience. *Proceedings of the 17th ACM Conference on Interaction Design and Children*, 563–568. https://doi.org/10.1145/3202185.3210776

Howard, Andrew G and Zhu, Menglong and Chen, Bo and Kalenichenko, Dmitry and Wang, Weijun and Weyand, Tobias and Andreetto, Marco and Adam, Hartwig. (2017).Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Kahn, K., & Winters, N. (2017). Child-Friendly Programming Interfaces to AI Cloud Services. In É. Lavoué, H. Drachsler, K. Verbert, J. Broisin, & M. Pérez-Sanagustín (Eds.), *Data Driven Approaches in Digital Education* (pp. 566–570). Springer International Publishing. https://doi.org/10.1007/978-3-319-66610-5_64

Lane, D. (2018). Explaining Artificial Intelligence. *Hello World*, *4*, 44–46.

Malyn-Smith, J., Lee, I. A., Martin, F., Grover, S., Evans, M. A., & Pillai, S. (2018). Developing a Framework for Computational Thinking from a Disciplinary

Perspective. *Proceedings of the International Conference on Computational Thinking Education 2018. Hong Kong: The Education University of Hong Kong*, 5.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence August 31,1955.* AI Magazine, Nº 4. https://doi.org/10.1609/aimag.v27i4.1904.

Ministerio de Ciencia, Innovación y Universidades. (2019). *Estrategia Española de I+D+I En Inteligencia Artificial*. Ministerio de Ciencia, Innovación y Universidades, 2019. Gobierno de España. http://www.ciencia.gob.es/stfls/MICINN/Ciencia/Ficheros/Estrategia_Inteligencia_Artificial_IDI.pdf

Moreno-León, J., Robles, G., Román-González, M., & Rodríguez-García, J. D. (2019). No es lo mismo: Un análisis de red de texto sobre definiciones de pensamiento computacional para estudiar su relación con la programación informática: Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming. *Revista Interuniversitaria de Investigación En Tecnología Educativa, Nº 7*. https://doi.org/10.6018/riite.397151

Paranyushkin, D. (2019, May). Infranodus: generating insight using text network analysis. In The World Wide Web Conference (pp. 3584-3589).

Pedró F., Subosa M., Rivas A., & Valverde P. (2019). Artificial intelligence in education: Challenges and opportunities for sustainable development—*UNESCO Biblioteca Digital* (UNESCO Working Papers on Education Policy). UNESCO. https://unesdoc.unesco.org/ark:/48223/pf0000366994?locale=es

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., & Silver, J. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60. https://doi.org/10.1145/1592761.1592779

Rodríguez-García, J. D., Moreno-León, J., Román-González, M., & Robles, G. (2019). Developing Computational Thinking at School with Machine Learning: An exploration. In 2019 International Symposium on Computers in Education (SIIE) (pp. 1-6). IEEE. https://doi.org/10.1109/SIIE48397.2019.8970124

Simmons, C., & Holliday, M. A. (2019). A Comparison of Two Popular Machine Learning Frameworks. *The Journal of Computing Sciences in Colleges, 20*.

Tang, D. (2019). *Empowering novices to understand and use machine learning with personalized image classification models, intuitive analysis tools, and MIT App Inventor* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/123130

The Design-Based Research Collective. (2003). Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher*, *32*(1), 5–8. https://doi.org/10.3102/0013189X032001005

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (2019). Envisioning AI for K-12: What Should Every Child Know about AI? *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 9795–9799. https://doi.org/10.1609/aaai.v33i01.33019795

Touretzkyds/ai4k12. (n.d.). GitHub. Retrieved 7 January 2020, from https://github.com/touretzkyds/ai4k12

Tuomi, I. (2019). *The Impact of Artificial Intelligence on Learning, Teaching, and Education*. Joint Research Centre (JRC). European Union. http://publications.jrc.ec.europa.eu/repository/bitstream/JRC113226/jrc113226_jrcb4_the_impact_of_artificial_intelligence_on_learning_final_2.pdf