A Classification and Review of Tools for Developing and Interacting with Machine Learning Systems

Eduardo Mosqueira-Rey
Universidade da Coruña (CITIC)
A Coruña, Spain
eduardo@udc.es

Elena Hernández Pereira
Universidade da Coruña (CITIC)
A Coruña, Spain
elena.hernandez@udc.es

David Alonso-Ríos
Universidade da Coruña (CITIC)
A Coruña, Spain
dalonso@udc.es

José Bobes-Bascarán
Universidade da Coruña (CITIC)
A Coruña, Spain
jose.bobes@udc.es

ABSTRACT

In this paper we aim to bring some order to the myriad of tools that have emerged in the field of Artificial Intelligence (AI), focusing on the field of Machine Learning (ML). For this purpose, we suggest a classification of the tools in which the categories are organized following the development lifecycle of an ML system and we make a review of the existing tools within each section of the classification. We believe this will help to better understand the ecosystem of tools currently available and will also allow us to identify niches in which to develop new tools to aid in the development of AI and ML systems. After reviewing the state-of-the-art of the tools, we have identified three trends in them: the incorporation of humans into the loop of the machine learning process, the movement from ad-hoc and experimental approaches to a more engineering perspective and the ability to make it easier to develop intelligent systems for people without an educational background in the area, in order to move the focus from the technical environment to the domain-specific problem.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**;

KEYWORDS

Artificial Intelligence, Machine Learning, Tools

ACM Reference Format:
Eduardo Mosqueira-Rey, Elena Hernández Pereira, David Alonso-Ríos, and José Bobes-Bascarán. 2022. A Classification and Review of Tools for Developing and Interacting with Machine Learning Systems. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event,* . ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3477314.3507310

1 INTRODUCTION

To design and to interact with Artificial Intelligence (AI) systems in general, and Machine Learning (ML) systems in particular, many tools have been provided by industry leaders and free-software organizations that help in the different steps of the AI pipeline: gather data, extract features, label the examples, select the model, tune the hyper-parameters, and evaluate the model performance. Moreover, the tool-set allows users to test and discard certain approaches at an early stage, and move the focus to the domain-specific problem.

This paper aims to offer a classification and review of such tools. In order to achieve this goal, the authors consulted the literature, searching for documents by title focusing on each particular type of tool separately and filtering by relevance based on the abstract, recentness and the number of citations. Research papers were obtained from online academic databases such as Google Scholar, ACM Digital Library, IEEE Xplore, ScienceDirect, Springer Link or Scopus. On the other hand, specific tools that are available for download can be aimed at academic or non-academic audiences, and can be found through their own websites or even the app stores of the major platforms.

These tools can be roughly organized in categories that are not mutually exclusive. These categories are organized more or less following the development lifecycle of an AI system, paying special attention to ML systems, which is the most dynamic branch of AI today and where most developments are carried out.

Therefore, we will start from an initial classification focused on data management and rapid prototyping. Subsequently, we can include the popular frameworks that exist today for the development of AI and ML solutions. These frameworks were initially used to develop traditional ML, also called passive or classical ML in which we have cycles of supplying data to the system, observing the outputs and drawing conclusions that serve to improve the results of the next cycle.

Subsequently, these passive systems opened up to dynamic collaboration with users, first in the form of active learning [70], where humans were used as oracles to call upon when there was a need to label a new datum. In these systems, control rested with the learning algorithms, which called on humans when deemed necessary. Later these systems were extended to models where control was more shared between algorithms and humans, giving rise to interactive machine learning [23], where interactions between algorithms and machines were more focused, frequent, and incremental. Since the interaction between users and learning systems is tighter, Human-Computer Interaction (HCI) techniques are regaining importance in this approach.

The next step are tools that try to help us develop AI-based systems without having much experience in the field, i.e., we may be experts in our application domain but know very little about the different AI and ML models available [47]. They are tools that try to automate slow and iterative tasks allowing to create models with high scaling, efficiency and productivity, while maintaining their quality. These tools are often based on cloud developments, offered through a "Software as a service" (SaaS) model. Here we can also contemplate Machine Teaching tools [78, 87], where the control of learning falls on human experts in the domain that delimit the knowledge that they intend to transfer to the learning model.

Once the model is built, we need to ask the model to explain its conclusions. In this way, explainable AI [27] tools attempt to make the decisions of AI models explainable and justifiable to humans. In other words, it is not enough for a system to work well, it has to be able to explain why it has worked well, since nowadays the decisions taken by AI-powered systems may affect humans' lives (as in e.g. medicine, law or defense).

We end the classification with tools that help build AI models by users who, in many cases, are not even aware that they are helping to develop them. These tools are based on the power of large numbers, i.e., many people doing small tasks, but which together offer synergies that are difficult to obtain in any other way. The final classification obtained can be seen in the following list:

- **Data related tools**: the existence of processes and tools for generating training data, previously a matter mostly for one-off research projects, now have a large impact on the success of machine learning projects [64].
- **Prototyping tools**: that help designers to build working prototypes of artificial intelligence systems.
- **Frameworks**: libraries that make use of AI solutions, in a more user-friendly and configurable way.
- **Active learning tools**: tools to develop systems that use an entity with extensive knowledge of the domain (typically a human expert) as an oracle for label unlabeled examples.
- **Interactive learning tools**: in which there is a closer interaction between users and learning systems, with people interactively supplying information in a more focused, frequent, and incremental way compared to traditional ML.
- **AutoML tools**: provides methods and processes to ease the tasks of creating an evaluating an ML model.
- **Cloud tools**: platforms under the umbrella of Machine Learning as a Service (MLaaS) that cover, in between others, data pre-processing, model training and evaluation.
- **Machine teaching tools**: applications that allow a final user with no technical expertise in AI (acting as a teacher) to build classifiers and entity extractors.
- **Explainable tools**: help on interpreting model predictions, making the results understandable for humans.
- **Crowd-sourcing labeling tools**: allow a huge amount of users to provide labels for the examples to be used for training and validation of ML models.

The contribution of this work is, therefore, a classification of the different tools that exist for developing and interacting with AI and ML systems, and a review of the existing tools within each section of the classification. We believe this will help to better understand the ecosystem of tools currently available and will also allow us to identify niches in which to develop new tools to aid in the development of AI systems.

In Table 1 we can consult a summary of all the categories, subcategories and tools mentioned in the text. It is important to explain that the table is not exhaustive; the most representative or relevant tools in each section have been included, but there are many others that could not be mentioned for lack of space. Within each section we have cited, on occasion, other papers with more exhaustive reviews of tools within that category, and we encourage the reader to refer to them for more detailed information. It is also necessary to remember that the categories are not mutually exclusive and that certain tools can be perfectly classified within several.

2 DATA RELATED TOOLS

From the product development field and the agile world we obtain the idea of a "Minimum Viable Product (MVP)". An MVP is a product with just enough features to satisfy early customers, and to provide feedback for future product development [69]. From the world of machine learning we can now coin the term "*Minimum Viable Data (MVD)*" that refers to the minimum data needed to train the machine learning models [93].

The process of data collecting, labeling, and model training is expensive and time-consuming. Defining an MVD strategy in the prototyping phase (perhaps in collaboration with a data scientist [99]) could lead to significant reductions in cost and time to market, or new insights about the characteristics of the data needed. The idea is to increase speed and reduce complexity defining a minimum viable data set. Once successful, researchers can provide additional data in the future to be integrated in the system.

An example of tools for MVD are rapid interactive refinement tools such as the image segmentation tools. Images are typically partitioned based on edges and regions, focusing on uniformity and discontinuities (a taxonomy of possible approaches can be found in [28]). Depending on the degree of automation, this technique can be classified as:

- **Manual**: Offers total control and optimal results, but it can be too time-consuming, error-prone and tedious (e.g. assigning each pixel to its corresponding class) for intensive professional environments. Examples include RectLabel [17] and VGG Image Annotator (VIA) [20].
- **Fully-automated**: Requires no human involvement but fails to achieve optimal results. Great advances have been made in recent years, however, thanks to new techniques such as Deep Learning (DL). A notable DL tool is U-Net [82], which has generated many extensions [25] (see also [79] for an overview of DL in medical image processing). But so far, this approach has found little success in professional settings.
- **Semi-automated**: The user typically provides partial information (e.g. boundaries or labels), which is then automatically completed by algorithms. This combines the expertise

Table 1: Summary of categories and tools

Category	Sub-category	Tool
Data img. tools	Manual	RectLabel [17], VIA [20]
	Semi-automated	3D Slicer [2], ITK-SNAP [11]
	Fully-automated	U-Net [82]
Prototyping		Wizard of Oz (WOZ) [65] Delft AI Toolkit [93] Lobe.ai [66], ml5.js [68]
Frameworks	Core libraries	Compilers: IPython [75], Cython [30] Vector: NumPy [46]
	Data preparation	Pandas [97]
	Data visualization	Matplotlib [53] Seaborn [96]
	Machine learning	Python: Scikit-learn [74] C++: Shogun [88] Java: Weka [45]
	Deep learning	TensorFlow [22] Keras [35], PyTorch [73] GPT-3 [32]
	Big data	Spark [101] Hadoop [98]
AL tools		ALiPy [91], Libact [100] modAL [36], NEXT [83]
IML tools	General purpose	AnchorViz [90]
	Unstructured data	Crayons [38], CueFlik [41] Ilastik [31], AIDE [58] JAABA [57], AIML [95], InteractML [49] Wekinator [39], BeatBox [50] SLU toolset [29], CLBCI [61].
	Explanatory	CAIPI [92]
AutoML tools	ML developers	H2O AutoML [63] AutoKeras [56]
	Non-ML developers	VDS [62], Akkio [3], DataRobot [8], Levity [13], Obviously.ai [14]
	Technical tasks	AMC [48]
Cloud tools		Google Cloud AutoML [9] IBM Watson ML Studio [10] Microsoft Azure ML Std [7] Amazon SageMaker Std [5]
MT tools		Teachable Machine tool [18] Bonsai [16] LUIS [12], PICL [15]
XAI tools		LIME [80], Skater [21] MS InterpretML [71] Google Explanable AI [1] AI Explainability 360 [24]
	Python packages	ELI5 [60] Alibi [59] Dalex [26]
Crowdsourcing		Amazon Mechanical Turk [4] Appen [6] Toloka.ai [19]

of humans with the effectiveness and repeatability of automation, representing an attractive middle ground between the two former approaches and offering a good trade-off between effort and results. These tools are widely used in medicine, for example, 3D Slicer [2] and ITK-SNAP [11].

3 PROTOTYPING TOOLS

Other tools focus not on data but on helping designers build working prototypes of artificial intelligence systems. For example, we can cite the *Wizard of Oz* (WOZ) approach [65] that allows the designer to easily sketch the AI for themselves, for collaborators, and for usability testing, before investing in a functional AI system. Once the design matures, the toolkit allows the designer to replace the WOZ version of AI with appropriate functional implementations of algorithmic AI.

A relevant example is the Delft AI Toolkit [93], a tool that provides a drag-and-drop, visual programming environment to build the interaction and behavior of an AI system. In addition to building the logic of the application, the tool allows the designer to incrementally prototype the AI, form and behavior.

Some other tools provide the users with a specific use case as image recognition so that they can experiment directly in the application without expert knowledge. An example is Microsoft's *Lobe.ai* [66]. This tool divides the process of machine learning into three steps: (1) collect and label images, (2) train and understand the results and (3) play with your model and improve it. This way, Lobe.ai simplifies the data collection and labeling process, starts training a machine learning model without any setup or configuration and gives live feedback on model's performance allowing to understand model's strengths and weaknesses more quickly.

Another tool that aims to make machine learning approachable for non-expert users is *ml5.js* [68]. This tool is more a library that provides access to machine learning algorithms and models in the browser, providing immediate access to to pre-trained models for multiple situations (detecting human poses, generating text, styling an image with another, composing music, pitch detection, and common English language word relationships, etc.). Currently, many of these developments have been oriented towards AutoML, which we discuss in section 7.

4 FRAMEWORKS

The tools mentioned in the past sections normally are based on AI frameworks. These AI frameworks are created with the aim of making the technology more accessible for end-users. The approaches can vary depending on the types of users that are being considered (e.g. developers, data scientists, designers, etc.). For example, ML model creation frameworks are very popular nowadays, and they are aimed at users that are comfortable writing code [78]. In [43] and [89] we can consult comparisons of the different frameworks. Most of these frameworks are written using the Python [42] programming language. The main reasons for this fact are:

- It is a language with an easy learning curve.
- It is a powerful but flexible language: multi-platform, multi-paradigm, garbage-collected, dynamically-typed, etc.
- It has a strong community, specially in scientific research.
- It has a powerful ecosystem that produces synergies.

Some other reasons about how Python has become a high-level language suited for science and engineering can be found in [72] and a brief history of scientific computing in Python in [67].

However, Python also has disadvantages. The main one is usually its low performance due to the fact that it is an interpreted language. The typical way to solve this is to divide the libraries into two parts, a *backend* with the core functionalities, written in an efficient language such as C or C++, and a *frontend*, written in Python, which acts as an interface with the user or programmer.

Nevertheless, probably the main reason for the success of Python and the frameworks and libraries written with it, is their openness. Starting with the language itself, released with the permissive Python Software Foundation License (PSFL) and controlled by a nonprofit organization, the Python Software Foundation (PSF).

The core scientific libraries of Python (NumPy, Pandas, Matplotlib, etc.) form an ecosystem called ScyPy [94] that is also open source (with a permissive BSD compatible license) and that is sponsored by NumFOCUS, a public charity in the United States. This fact has promoted that libraries built on this basis also follow the same philosophy, even when they are developed by large corporations and not so much by foundations or universities. This scenario favors innovation, constant updating and the growth of the community.

The frameworks and libraries of AI can be organized in a hierarchy, starting from the low level and going upwards in complexity and abstraction. An overview of them can be consulted in [89]:

- **Core**: Core libraries and tools that are the basis of the ecosystem. Here we can highlight compilers like IPython [75] or Cython [30] and libraries for performing efficient vectorized computing such as NumPy [46].
- **Data preparation**: Nowadays, data management is an important part of the AI workflow. Within this area, the Pandas [97] library stands out, allowing us to manage, filter, query and perform operations with tables and data series.
- **Data visualization**: Visualizing data is often almost as important as managing it. Within this field, libraries such as Matplotlib [53] or Seaborn [96] stand out.
- **Machine learning**: There are several machine-learning libraries, some of the most popular are Scikit-learn [74] for the Python language, Shogun [88] for the C++ language or Weka [45] for the Java language.
- **Deep learning**: DL is a subfield of ML concerned with deep neural nets, that is, with a high number of layers. In this category we can highlight general purpose libraries such as TensorFlow [22], developed by Google Brain; Keras [35] built on top of TensorFlow and, nowadays integrated into it as a high level layer; and Pytorch [73] developed by Facebook. Also we can find libraries for specific purposes such as GPT-3 (Generative Pre-trained Transformer) [32] a model developed by OpenAI for natural language processing (NLP).
- **Big data**: Big data tools analyze extremely large data sets to reveal patterns, trends, and associations, especially relating to human behavior and interactions. Within this field we can highlight tools like Spark [101] or Hadoop [98] used mainly with the Java language.

5 ACTIVE LEARNING TOOLS

Active Learning (AL) is a machine learning approach in which the learner (typically a machine) requests an oracle (typically a human, who acts as a teacher) to label examples that are not clear or that will provide relevant information in the learning process. As a result, the learner improves its learning performance (i.e., maximizing accuracy).

The idea behind is to use fewer training examples than other supervised ML techniques in order to achieve higher accuracy. This is of special relevance where many unlabeled examples are available but the acquisition of labels is a costly task, or in scenarios with limited labeled data.

The learner is in control of the learning process, and iteratively selects unlabeled examples to be labeled by the oracle. The selection of specific examples is performed using queries. To explore further on the different query strategies we refer to [85] and [70].

Humans could be involved executing one or more of the following tasks:

- At preparation: collecting and preparing the initial data, to generate the first version of the model.
- At execution: as oracles, labeling the data when requested by the learning algorithm.
- At validation: as model evaluators, checking the performance updating the model when needed.

The great majority of tools available correspond to the Python ecosystem. ALiPy [91], which is a toolbox that provides a module-based implementation, offering support for novel settings as multi-labeled data examples, noisy oracle, and cost-sensitive annotation. Libact [100], a package that provides an easy-to-use environment for solving AL problems, that implements several AL algorithms and leaves interfaces so that it can be extended by the user. modAL [36], a modular framework, fully compatible with scikit-learn (it is built on top of it) and suitable for rapid prototyping. NEXT [54, 83], a system that runs in the cloud and integrates with a crowdsourcing tool to collect labels, and exposes a REST Web API as an integration mechanism.

Even tough Active Learning is a mature technique, there are still some issues to be solved as several assumptions where made from the very beginning that do not hold today [86]. When performing this review, we lack more integration of AL approaches in modern and popular AI platforms.

6 INTERACTIVE LEARNING TOOLS

In Active Learning (AL) the system remains in control of the learning model and humans only interact with it when requested. In Interactive Machine Learning (IML) [23], on the other hand, there is a closer interaction between users and learning systems, with people interactively supplying information in a more focused, frequent, and incremental way. The idea is to have humans and computers working together on the same task doing what each of them does best at any specific moment [77].

Humans can be included in different parts of the ML workflow. They can be used first, performing identification and annotation tasks that are simple for them but complicated for machines; in the middle, helping the model to perform inference tasks, providing reinforcement information and insights about the learning process; and, finally, at the end, validating, interpreting and correcting the results of a machine learning system. This interaction is specially interesting in complex domains, such as health informatics, where data sets are full of uncertainty and incompleteness, and the problems they try to solve are hard [51].

As there is greater interactivity with humans, the system interface now becomes more important, as it is responsible for the bidirectional feedback between users and model, making interface design critical to the success of the IML process [37].

As IML is a recent research field within ML and it encompasses many different aspects of the ML workflow, there are not many specific tools developed to support it. The most notable is AnchorViz [90], an interactive visualization tool that facilitates the discovery of prediction errors and previously unseen concepts through human-driven semantic data exploration. Apart from this, what we found were *ad-hoc* developments aimed at solving specific problems.

For example, IML has been successfully applied to systems that analyze unstructured data—data that has no identifiable structure, does not have a predefined model, or does not fit into relational databases, such as images, video and audio files, and certain types of text documents—. That is, IML seems to be especially useful in giving additional structure to something that does not have it.

For example, for images we can cite tools such as Crayons [38], a system that uses IML to create image classifiers or CueFlik [41] a web image search application in which users create their own rules for classifying images giving examples and counterexamples. But one application with images that has been very successful is using IML for interactive image segmentation (already mentioned when talking about data related tools). Among the recent developments within this technique we can name the ilastik tool [31] that contains pre-defined workflows for image segmentation, object classification, counting and tracking and that allows non-expert users to interactively provide annotations to steep the learning curve; or AIDE [58], an image annotation framework for ecological surveys that integrates closely users and machine learning models. Jiang et al. [55] made a thorough survey of existing IML works in the visual analytics community.

Other tools were developed to deal with unstructured data different from images. For videos we have the JAABA tool [57] used for adding labels to frames of videos to identify certain animals' behaviors; Assisted Interactive Machine Learning (AIML) [95] used for the recognition of musical gestures; or InteractML [49] a tool for making machine learning accessible for creative practitioners working with movement interaction in immersive media. In the music field we can find the Wekinator tool [39, 40], a software system that enables the application of music information retrieval techniques (based on machine learning) to real-time musical performance; or the BeatBox tool [50], a system that enables end-user creation of custom beatbox recognizers. For time series we can cite the SLU toolset [29] used for applying IML techniques to improve SLU (Spoken Language Understanding) models; and developments in the field of electromyography (EMG) analysis [102] or in the development of brain-computer interfaces like CLBCI (Co-Learning for Brain–Computer Interfaces) [61].

Finally we can use IML as a way to address the problem of the black-box nature of many of the ML models. For example the tool called CAIPI [92] for explanatory interactive learning allows the model to query users but also explains that query to them. Users then answer the query but also correct the explanation. The idea is to enhance the explanatory power of ML algorithms and, consequently, the trust that the users put in them.

7 AUTOML TOOLS

Automated Machine Learning tools (AutoML) provide methods and processes to make ML model creation and evaluation easier, and ultimately, available for non-Machine Learning experts. This set of tools seek to automate the decision on what learning algorithms to use, what hyper-parameters to select or which features are more relevant for a certain model. Furthermore, they provide a means of model evaluation and optimization.

On the one hand, we found several AutoML tools suited for developers or data scientist (they require software knowledge) such as H2O AutoML [63] or AutoKeras [56], which have lots of documentation providing examples of running code.

On the other hand, there are tools which goal is to democratize these technologies so users (non-experts) can perform complex analytics, and create ML models. Here we can cite Virtual Data Scientist (VDS) which is a component of the Northstar tool [62] that allows users to feed data sets, from which they can extract features to get insights on the data. It provides a user-friendly interface supporting touch screens and digital pens. Moreover, it trains ML models to run prediction tasks on the data sets.

We include also several online tools under the term "*no-code AI*" which provide an easy-to-use environment where the user could create fully functional AI prototypes and soluctions. Some examples in this category are Akkio [3], DataRobot [8], levity.ai [13], obviously.ai [14].

Other group of tools are focused on technical tasks such as model compression, which allow models to run efficiently on devices which have limited computation resources (e.g., mobile phones). Among them we can highlight AMC [48], which leverages reinforcement learning to automatically search the design space, improving the model compression quality.

For a more in-depth review of the field we can cite the recent work of He et al.[47]. Also, the main cloud platforms (eg., AWS, Azure, Google) are including great AutoML tools as part of their ML stacks, but we have decided to classify this type of tools as a separate category in the following section.

8 CLOUD TOOLS

The main advantage of the cloud is that it provides a platform for the users allowing them to focus in the problem itself, without having to worry about the infrastructure. Particularly, when related to Machine Learning the term Machine Learning as a Service (MLaaS) involves various cloud-based platforms that cover, in between others data pre-processing, model training and evaluation.

In this category we can found products from the biggest tech companies including Google Cloud AutoML [9], IBM Watson ML Studio [10], Microsoft Azure ML Services [7], and Amazon SageMaker [5]. These are the oldest and most mature platforms that provide various products for Machine Learning ranging from natural language processing, service bots, and even deep learning.

The core element of Google's machine learning services in the cloud is the Cloud AutoML suite [9] which suggests a no-code approach to building data-driven solutions with integrated predictions. AutoML was designed to build custom models for both newcomers and experienced machine learning engineers, but also suggests a set of pre-built models available via a set of APIs. The elements that help to differentiate this tool from the rest, are the video and audio data tools, specifically created to support these data types. In fact, the most versatile toolkit for image analysis is currently available at Google Cloud.

IBM Machine Learning platform is organized like other providers. The system offers three options at the moment: (1) the IBM Cloud for data, a suite of integrated solutions that automate the deployment of AI use cases in production; (2) the Watson Machine Learning Cloud service, a set of REST APIs to develop applications that make smarter decisions, solve tough problems, and improve user outcomes; and, (3) the Watson Machine Learning Server, a single-node server that is part of the IBM Watson Studio and that offers analytical assets and machine learning model notebooks. They all offer Auto-AI lifecycle management for models. With its array of open source tools and techniques, IBM Machine Learning gives flexibility over model deployment and model retraining at scale to data scientists.

Microsoft Azure Machine Learning Studio is a development environment that creates a resourceful playground both for entry-level and experienced data scientists. It has tools that range from data analysis, data visualization, data labeling, to deep learning. Approaching machine learning with Azure entails some learning curve, but it eventually leads to a deeper understanding of all major techniques in the field. The Azure ML graphical interface visualizes each step within the workflow and supports newcomers. One of the main benefits of using Azure is the variety of algorithms available to play with and the data transformation tools that are helpful during data analysis. Recently, Azure opened a preview for Azure Percept. The main idea behind Percept is to provide an SDK for creating models that can be integrated with Microsoft-partnered hardware devices. This entails an easy way of building and integrating computer vision or tools for speech recognition.

Amazon has two major products dedicated to machine learning. The earlier platform called Amazon Machine Learning and SageMaker, the actual machine learning platform. Amazon launched SageMaker Studio in 2021 as the first IDE for machine learning. This tool provides a web based interface that allows to perform quick model building and deployment within a single environment. All development methods and tools, including notebooks, debugging instruments, data modeling, and its automatic creation is available via SageMaker Studio for both experienced data scientists and those who just need things done without digging deeper into dataset preparations and modeling.

To wrap up with machine learning as a service (MLaaS) platforms, it seems that Azure has currently the most versatile toolset on the MLaaS market. It covers the majority of ML-related tasks, provides two distinct products for building custom models, and has a solid set of APIs for those who don't want to attack data science with their bare hands. Related to ML in cloud, there exists a survey which investigates the effects using the concepts of ML on cloud environments in terms of automation of resource management and scheduling [52]. Another survey investigates the impact of the cloud computing paradigm into the ML field [76]. This work analyzes statistics tools and libraries deployed in the cloud, existing tools augmented to run jobs on the cloud and ML as SaaS.

Deploying a Machine Learning model in production can be challenging. In fact, that 87% of data science projects do not make it to production. These platforms, which combine Machine Learning, DevOps, and Data Engineering practices to deploy and maintain ML systems reliably and efficiently, help address these issues. DevOps is an approach to software development that suggests merging devs and ops teams to optimize software development processes by focusing on short and fast releases. This is achieved by applying a high level of automation to routine tasks. The popularity of DevOps among the software development community gave birth to the term *MLOps*, which applies the same principles of DevOp to machine learning, which led to the emergence of automated data management, model training/deployment, and monitoring. That's why in 2021, MLaaS providers offer tools for MLOps practitioners to manage these machine learning pipelines.

As ML in cloud computing has numerous benefits, yet they have drawbacks as well. Here we can highlight the need of consistent connectivity, the need of privacy policies and security for data, and the challenge of integration into organizations.

9 MACHINE TEACHING TOOLS

Machine Teaching (MT) is a new AI paradigm in which a teacher (typically a human) transfers knowledge to a learner (typically an AI system).

The MLaaS and MLOps tools in the previous section aim to implement and maintain ML systems in production reliably and efficiently. These tools can be understood as early attempts at offering MT-related capabilities, but they require ML expertise. That is, if you are offering ML-specific information (e.g., learning rate, network architecture, etc.) you are not really acting as a teacher, because you are not transferring knowledge about the topic but about the technique [78]. A teacher gives information about labels but also semantic information about why these labels are used, as well as assessing performance. That is, the goal is to take advantage of the abilities we humans have when it comes to sharing knowledge among us, and using them to transfer knowledge to a machine.

Cloud tools are typically aimed at facilitating the creation and maintenance of traditional AI systems. They focus on engineering tasks and working efficiently, and are not intended for more MT-relevant aspects such as "concept evolution", although they are slowly evolving to become more useful to people who are not necessarily proficient in AI. In terms of MT tools we could say that they have evolved in parallel to the MT paradigm itself, progressing from a batch setting [103] to a "never-ending loop" [87].

Some specific examples we could mention include the Teachable Machine tool [18], in which a user selects examples for the system—or provides them live—and then trains it, checking its correctness at the same time. Another example is Project Bonsai [16], whose motto is "build AI without data science".

Recent tools, then, tend to be more interactive, with examples like Language Understanding (LUIS) [12] and Platform for Interactive Concept Learning (PICL) [15], both from Microsoft (a discussion of interactive versus non-interactive tools can be found in [78]). LUIS is an ML-based service for building natural language capabilities into apps, bots, and IoT devices. It learns continuously via Active Learning, letting the user update and improve the model. PICL, on the other hand, is probably the tool that is most explicitly connected to the current MT paradigm. In this tool "the human teacher guides the machine towards accomplishing the task of interest. The system leverages big data to find examples that maximize the training value of its interaction with the teacher" [87].

The interesting aspect of these systems is that the purely technical matters—such as the training, scoring, regularization, splitting the data set—happen under the hood, allowing the user to focus in providing the best labels to the examples, resulting in a highly accurate model. It is also remarkable that the interface has been carefully designed for a non-expert user.

10 EXPLAINABLE TOOLS

The goal of explainable tools is to help understanding the AI model outputs and build trust. In this section, we focus on explainability by means of external techniques. They are referred as to post-modeling explainability techniques and have been included in tools as LIME (Local Interpretable Model-Agnostic Explanations) [80] that builds locally linear models around the predictions of an opaque model to explain it. This tool generates an explanation by approximating the underlying model by an interpretable one, learned on perturbations of the original instance. The key intuition behind LIME is that it is much easier to approximate a black-box model by a simple model locally (in the neighborhood of the prediction we want to explain), as opposed to trying to approximate a model globally. This is done by weighting the perturbed instances by their similarity to the instance to be explained. LIME was used to explain a myriad of classifiers (such as random forests, support vector machines (SVM), and neural networks) in the text and image domains. In Ribeiro et al. [81] *anchors* is introduced, a novel model-agnostic system that explains individual predictions of a model by using high-precision IF-THEN rules — "anchors" — that support (anchor) the predictions well enough. To find anchors, the authors use reinforcement techniques in combination with a graph search algorithm to explore the sets of perturbations around the data and their effect on the predictions. An empirical evaluation of LIME is performed in [44] which reveals its main advantages and limitations.

Even if it is in beta phase, Skater [21] is an open-source, model-agnostic unified Python framework for model explainability and interpretability. Skater originally started off as a fork of LIME but then broke out as an independent framework of it's own. This tool approaches explainability both globally (inference based on a complete dataset) and locally (inference individual predictions) and it supports deep neural networks, tree algorithms, and scalable Bayes. Skater can also be used to discover hidden feature interactions and then to inform future analyses with that information. Skater enables consistency in the ability to interpret predictive models when in-memory as well as when operationalized, giving practitioners the opportunity to measure how feature interactions change across different model versions. Such form of interpretation is also useful for enabling trust when using off-the-shelf predictive models from an ML marketplace.

Another example is Microsoft InterpretML [71] an open-source package that incorporates state-of-the-art machine learning interpretability techniques under one roof. With this package, the user can train interpretable glassbox models and explain blackbox systems. InterpretML helps the user understand its model's global behavior, or understand the reasons behind individual predictions. Along the same line, Google Explainable AI [1] is a framework that helps to understand and interpret what led machine learning models to a particular prediction. It is centered around an instance-level feature attributions, which provide a signed per-feature attribution score proportional to the feature's contribution to the model's prediction. The framework includes a *What-If* tool, an interactive visual interface designed to help visualize data and enable probing of ML models. Recently, in collaboration with OpenAI, Google came up with Activation Atlases [34], an approach to visualize how neural networks interact with each other and how they mature with information along with the depth of layers. This approach was developed for looking at the inner workings of convolutional vision networks and getting a human-interpretable overview of concepts within the hidden network layers. It started with feature visualization on individual neurons but has since moved to visualize neurons jointly.

AI Explainability 360 [24] is a toolkit from IBM developed as a open source framework that supports the interpretability and explainability of datasets and machine learning models. This tool includes a collection of algorithms that cover different dimensions of explanations along with proxy explainability metrics.

Among the explainable tools, there are several examples in terms of python packages. One of them is ELI5 – Explain Like I'm Five – [60], which has built-in support for several ML frameworks and provides a way to explain black-box models. It offers visualizations and debugging to these models through its unified API and it can be used for both inspect model parameters and try to figure out how the model works globally; and inspect an individual prediction of a model, try to figure out why the model makes the decision it makes. Another example is Alibi [59] that provides high-quality implementations of black-box, white-box, local and global explanation methods for classification and regression models. Finally, we can cite Dalex [26] a set of tools that examines any given model, simple or complex, and explains the behavior of that model. Dalex creates a level of abstraction around each model that makes it easier to explore and explain.

All these different tools allow us to analyze the black-box models, and each of them looks from a slightly different angle. Some of these tools focus on a global explanation — a holistic view of the main factors that influence the predictions of the model. Other tools focus on generating a local explanation, which means focusing on a specific prediction made by the model. What is essential to understand is each model explainability tool is different. Using two distinct tools for computing global explanations lead to two diverse outcomes, and thus, a different understanding of the model.

11 CROWDSOURCING LABELING TOOLS

As part of the AI pipeline, and particularly when using a supervised approach to build an AI model, the labeling task is crucial, as without it no matter what happens in the following phases of the process, the model would just not be efficient at all [70].

A crowdsourcing AI tool uses a huge amount of users that individually, label examples to be used by AI algorithms. These small tasks, where the human outperforms the machine, are delegated to hundreds or users that complete the tasks in exchange for a monetary reward.

The most relevant example is Amazon Mechanical Turk [4]. It is a crowdsourcing platform (marketplace) enabling individuals and businesses to engage a distributed workforce to perform a task. The tasks are called Human Intelligence Tasks (HIT) which are a single, self-contained tasks a human will tackle.

Other examples of this type of tools are: Appen [6], which deals with several data types including speech and natural language data, image and video data, text and alphanumeric data, and relevance data to improve search and social media engines. Toloka.ai [19], which is a platform that manages large-scale data collection and annotation projects for ML.

It is not a surprise that the best accuracy we get in the labeling process, the better performance we will obtain from our model. Nevertheless, when using a crowdsourcing approach it is not possible to be behind each user checking their work, so we are exposed to labeling errors. In some platforms these errors can be tackled and eliminated before going into the model [33].

12 DISCUSSION AND CONCLUSIONS

The history of AI can be told as a succession of "springs" (times of great developments and expectations) followed by "winters" (times when this enthusiasm cools down when the anticipated expectations are not met). Thus, the initial enthusiasm of the 1960s, which saw artificial general intelligence as something feasible, led us to a first winter at the end of the 1970s, when the true complexity of the problem was revealed. A later spring in the 1980s came with the development of expert systems focused on solving concrete problems in complex domains. The limitations of the symbolic model, the lack of computational power and the inability of these systems to be conveniently updated led to a new winter in the 1990s.

Today we can say that we are living in a new springtime, sustained in this case by the following pillars: an increase in computational capacity, based on the use of GPU-based technology; development of new techniques and models, especially those based on machine learning in general and deep learning in particular and, finally, the availability of massive amounts of data and the ease of handling them in a distributed manner. This spring has also been noticed in the tools for the development of AI systems. We can say that today, with the development of AI frameworks and libraries, most of them published under free licenses, developing an intelligent model is easier than ever.

These tools were initially designed for the development of a classical scheme in which all the all labeled data is provided in advance, and the ML algorithm is modeled, built, tested and then offered to the public without further changes, but models that are developed under this scenario might run the risk of not scaling well, becoming static, being hard to evaluate, and degrading their performance due to changes in the context they are deployed into.

Therefore, we can say that a first trend observed in the development of AI tools is to incorporate, to a greater or lesser extent, humans into the loop of the machine learning process. This can be done by following several strategies, one is to use only humans as labelers of new data with the model retaining control of learning (Active Learning). Another strategy is for humans and machines to work together, interactively and incrementally, sharing control of the learning process (Interactive Machine Learning). Finally, the Machine Teaching approach involves humans teaching machines following learning schemes similar to those that humans employ to teach other humans.

Another trend we have observed in this work is that the tools, as they mature, are moving away from the *ad-hoc* and experimental approach to a more engineering perspective. In this way, we see how tools such as AutoML or cloud platforms for developing machine learning as a service provide users with a basic infrastructure that has never been available before in the history of AI.

A final trend detected in this work is that the tools developed not only try to make it easier to develop intelligent systems, or to offer us a working infrastructure to carry it out, but also try to focus our attention, as far as possible, on the domain-specific problem and not so much on the technique needed to implement it. In other words, it wants to offer the domain expert, who is not an expert in AI, the possibility of developing an intelligent system. This would democratize AI and offer it to people without an educational background in the area.

A final though is that, while developing and deploying ML systems is relatively quick and inexpensive today, maintaining them over time is likely to be difficult and expensive. This is due that ML systems have a special capacity for incurring technical debt—the long term costs incurred by moving quickly in software engineering—as pointed by Sculley et al. [84], not only because they have all of the maintenance problems of traditional code but also because the have an additional set of ML-specific issues. Only time will tell if this spring leads to a summer, or a new winter.

ACKNOWLEDGMENTS

This work has been supported by the State Research Agency of the Spanish Government, grant (PID2019-107194GB-I00 / AEI / 10.13039/501100011033) and by the Xunta de Galicia, grant (ED431C 2018/34) with the European Union ERDF funds. We wish to acknowledge the support received from the Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (European Regional Development Fund- Galicia 2014-2020 Program), by grant ED431G 2019/01.

REFERENCES

[1] 2020. AI Explanations whitepaper. https://cerre.eu/wp-content/uploads/2020/07/ai_explainability_whitepaper_google.pdf
[2] 2021. 3D Slicer image computing platform. https://www.slicer.org/
[3] 2021. akkio. https://www.akkio.com/
[4] 2021. Amazon Mechanical Turk. https://www.mturk.com/
[5] 2021. Amazon SageMaker. https://aws.amazon.com/sagemaker/
[6] 2021. Appen. https://appen.com/
[7] 2021. Azure Machine Learning Studio. https://studio.azureml.net/
[8] 2021. datarobot. https://www.datarobot.com/
[9] 2021. Google's AI Platform. https://cloud.google.com/automl/
[10] 2021. IBM Watson Studio. https://www.ibm.com/cloud/watson-studio
[11] 2021. ITK-SNAP. http://www.itksnap.org/
[12] 2021. Language Understanding (LUIS). https://www.luis.ai/

[13] 2021. levity. https://levity.ai/
[14] 2021. obviously. https://www.obviously.ai/
[15] 2021. Platform for Interactive Concept Learning (PICL). https://www.microsoft.com/research/project/platform-for-interactive-concept-learning-picl/
[16] 2021. Project Bonsai. https://bons.ai
[17] 2021. RectLabel. https://rectlabel.com/
[18] 2021. Teachable Machine. https://teachablemachine.withgoogle.com/
[19] 2021. Toloka.ai framework. https://toloka.ai/
[20] 2021. VGG Image Annotator. https://www.robots.ox.ac.uk/~vgg/software/via/
[21] Pramit Choudhary et al Aaron Kramer. 2017. Skater. https://oracle.github.io/Skater/index.html
[22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.
[23] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine* 35, 4 (Dec. 2014), 105–120. https://doi.org/10.1609/aimag.v35i4.2513
[24] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. 2019. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. arXiv:cs.AI/1909.03012
[25] Reza Azad, Maryam Asadi-Aghbolaghi, Mahmood Fathy, and Sergio Escalera. 2019. Bi-directional ConvLSTM U-Net with densley connected convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.
[26] Hubert Baniecki, Wojciech Kretowicz, Piotr Piatyszek, Jakub Wisniewski, and Przemyslaw Biecek. 2021. dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python. *Journal of Machine Learning Research* 22, 214 (2021), 1–7. http://jmlr.org/papers/v22/20-1473.html
[27] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82 – 115. https://doi.org/10.1016/j.inffus.2019.12.012
[28] D Baswaraj, A Govardhan, and P Premchand. 2012. Active Contours and Image Segmentation: The Current State Of the Art. *Global Journal of Computer Science and Technology* (2012). https://computerresearch.org/index.php/computer/article/view/568
[29] Lee Begeja, Bernard Renger, David Gibbon, Zhu Liu, and Behzad Shahraray. 2004. Interactive machine learning techniques for improving SLU models. In *Proceedings of the HLT-NAACL 2004 Workshop on Spoken Language Understanding for Conversational Systems and Higher Level Linguistic Information for Speech Processing*. 10–16.
[30] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. 2011. Cython: The Best of Both Worlds. *Computing in Science Engineering* 13, 2 (2011), 31–39. https://doi.org/10.1109/MCSE.2010.118
[31] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N. Straehle, Bernhard X. Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, et al. 2019. Ilastik: interactive machine learning for (bio)image analysis. *Nature Methods* 16, 12 (2019), 1226–1232. https://doi.org/10.1038/s41592-019-0582-9
[32] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, et al. 2020. Language Models are Few-Shot Learners. arXiv:cs.CL/2005.14165
[33] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. 2011. Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data? *Perspectives on Psychological Science* 6, 1 (2011), 3–5. https://doi.org/10.1177/1745691610393980 arXiv:https://doi.org/10.1177/1745691610393980 PMID: 26162106.
[34] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. Activation Atlas. *Distill* (2019). https://doi.org/10.23915/distill.00015
[35] François Chollet et al. 2015. Keras. https://keras.io.
[36] Tivadar Danka and Peter Horvath. 2018. modAL: A modular active learning framework for Python. *arXiv preprint arXiv:1805.00979* (2018).
[37] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 8 (June 2018), 37 pages. https://doi.org/10.1145/3185517
[38] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive Machine Learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03)*. Association for Computing Machinery, New York, NY, USA, 39–45. https://doi.org/10.1145/604045.604056
[39] Rebecca Fiebrink and Perry R Cook. 2010. The Wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, Vol. 3.
[40] Rebecca Anne Fiebrink. 2011. *Real-time human interaction with supervised learning algorithms for music composition and performance.* Ph.D. Dissertation. Princeton University.
[41] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/1357054.1357061
[42] Python Software Foundation. 2021. The Python Tutorial. https://docs.python.org/3/tutorial/.
[43] Migran N. Gevorkyan, Anastasia V. Demidova, Tatiana S. Demidova, and Anton A. Sobolev. 2019. Review and comparative analysis of machine learning libraries for machine learning. *Discrete and Continuous Models and Applied Computational Science* 27, 4 (2019), 305–315. http://journals.rudn.ru/miph/article/view/22913
[44] Yoseph Hailemariam, Abbas Yazdinejad, Reza M. Parizi, Gautam Srivastava, and Ali Dehghantanha. 2020. An Empirical Evaluation of AI Deep Explainable Tools. In *2020 IEEE Globecom Workshops (GC Wkshps*. 1–6. https://doi.org/10.1109/GCWkshps50303.2020.9367541
[45] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 10–18. https://doi.org/10.1145/1656274.1656278
[46] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, et al. 2020. Array programming with NumPy. *Nature* 585 (2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2
[47] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622
[48] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 784–800.
[49] Clarice Hilton, Nicola Plant, Carlos González Díaz, Phoenix Perry, Ruth Gibson, Bruno Martelli, Michael Zbyszynski, Rebecca Fiebrink, and Marco Gillies. 2021. InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*. 1–10.
[50] Kyle Hipke, Michael Toomim, Rebecca Fiebrink, and James Fogarty. 2014. BeatBox: End-User Interactive Definition and Training of Recognizers for Percussive Vocalizations. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*. Association for Computing Machinery, New York, NY, USA, 121–124. https://doi.org/10.1145/2598153.2598189
[51] Andreas Holzinger, Markus Plass, Michael Kickmeier-Rust, Katharina Holzinger, Gloria Cerasela % Crişan, Camelia-M Pintea, and Vasile Palade. 2019. Interactive machine learning: experimental evidence for the human in the algorithmic loop. *Applied Intelligence* 49, 7 (2019), 2401–2414. https://doi.org/10.1007/s10489-018-1361-5
[52] Elham Hormozi, Hadi Hormozi, Mohammad Kazem Akbari, and Morteza Sargolzai Javan. 2012. Using of Machine Learning into Cloud Environment (A Survey): Managing and Scheduling of Resources in Cloud Systems. In *2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. 363–368. https://doi.org/10.1109/3PGCIC.2012.69
[53] J. D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55
[54] Kevin G Jamieson, Lalit Jain, Chris Fernandez, Nicholas J. Glattard, and Rob Nowak. 2015. NEXT: A System for Real-World Development, Evaluation, and Application of Active Learning. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/89ae0fe22c47d374bc9350ef99e01685-Paper.pdf
[55] Liu Jiang, Shixia Liu, and Changjian Chen. 2019. Recent research advances on interactive machine learning. *Journal of Visualization* 22, 2 (2019), 401–417. https://doi.org/10.1007/s12650-018-0531-1
[56] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1946–1956.
[57] Mayank Kabra, Alice A Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. 2013. JAABA: interactive machine learning for automatic annotation of animal behavior. *Nature Methods* 10, 1 (2013), 64–67. https://doi.org/10.1038/nmeth.2281
[58] Benjamin Kellenberger, Devis Tuia, and Dan Morris. 2020. AIDE: Accelerating image-based ecological surveys with interactive machine learning. *Methods in Ecology and Evolution* 11, 12 (2020), 1716–1727. https://doi.org/10.1111/2041-210X.13489 arXiv:https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13489
[59] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. 2021. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal*

of Machine Learning Research 22, 181 (2021), 1–7. http://jmlr.org/papers/v22/21-0017.html

[60] Mikhail Korobov and Konstantin Lopuhin. 2017. Explain Like I'm Five (ELI5). https://eli5.readthedocs.io/en/latest/index.html

[61] Nataliya Kosmyna, Franck Tarpin-Bernard, and Bertrand Rivet. 2015. Adding Human Learning in Brain–Computer Interfaces (BCIs): Towards a Practical Control Modality. *ACM Trans. Comput.-Hum. Interact.* 22, 3, Article 12 (May 2015), 37 pages. https://doi.org/10.1145/2723162

[62] Tim Kraska. 2018. Northstar: An Interactive Data Science System. *PVLDB* 11, 12 (2018), 2150–2164.

[63] Erin LeDell and Sebastien Poirier. 2020. H2O AutoML: Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning (AutoML)* (July 2020). https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf

[64] Martin Lindvall, Jesper Molin, and Jonas Löwgren. 2018. From Machine Learning to Machine Teaching: The Importance of UX. *Interactions* 25, 6 (Oct. 2018), 52–57. https://doi.org/10.1145/3282860

[65] David Maulsby, Saul Greenberg, and Richard Mander. 1993. Prototyping an Intelligent Agent Through Wizard of Oz. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 277–284. https://doi.org/10.1145/169059.169215

[66] Microsoft. 2020. Lobe.ai. https://www.lobe.ai.

[67] K. J. Millman and M. Aivazis. 2011. Python for Scientists and Engineers. *Computing in Science Engineering* 13, 2 (2011), 9–12. https://doi.org/10.1109/MCSE.2011.36

[68] ml5. 2020. ml5.js - Friendly Machine Learning for the Web. https://learn.ml5js.org/.

[69] Dobrila Rancic Moogk. 2012. Minimum viable product and the importance of experimentation in technology startups. *Technology Innovation Management Review* 2, 3 (2012), 23–26.

[70] Robert Munro. 2020. *Human-in-the-Loop Machine Learning*. Manning Publications.

[71] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv:cs.LG/1909.09223

[72] T. E. Oliphant. 2007. Python for Scientific Computing. *Computing in Science Engineering* 9, 3 (2007), 10–20. https://doi.org/10.1109/MCSE.2007.58

[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[74] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[75] F. Perez and B. E. Granger. 2007. IPython: A System for Interactive Scientific Computing. *Computing in Science Engineering* 9, 3 (2007), 21–29. https://doi.org/10.1109/MCSE.2007.53

[76] Daniel Pop. 2016. Machine Learning and Cloud Computing: Survey of Distributed and SaaS Solutions. arXiv:cs.DC/1603.08767

[77] Reid Porter, James Theiler, and Don Hush. 2013. Interactive Machine Learning in Data Exploitation. *Computing in Science Engineering* 15, 5 (2013), 12–20. https://doi.org/10.1109/MCSE.2013.74

[78] Gonzalo Ramos, Christopher Meek, Patrice Simard, Jina Suh, and Soroush Ghorashi. 2020. Interactive machine teaching: a human-centered approach to building machine-learned models. *Human–Computer Interaction* (2020), 1–39.

[79] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. 2018. Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps* (2018), 323–350.

[80] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:cs.LG/1602.04938

[81] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://ojs.aaai.org/index.php/AAAI/article/view/11491

[82] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[83] Scott Sievert, Daniel Ross, Lalit Jain, Kevin Jamieson, Rob Nowak, and Robert Mankoff. 2017. NEXT: A system to easily connect crowdsourcing and adaptive data collection. In *Proceedings of the 16th Python in Science Conference*, Katy Huff, David Lippa, Dillon Niederhut, and M Pacer (Eds.). 113–119. https://doi.org/10.25080/shinma-7f4c6e7-010

[84] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, et al. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

[85] Burr Settles. 2009. *Active learning literature survey.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences. https://minds.wisconsin.edu/handle/1793/60660

[86] Burr Settles. 2011. From Theories to Queries: Active Learning in Practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010 (Proceedings of Machine Learning Research)*, Isabelle Guyon, Gavin Cawley, Gideon Dror, Vincent Lemaire, and Alexander Statnikov (Eds.), Vol. 16. JMLR Workshop and Conference Proceedings, Sardinia, Italy, 1–18. http://proceedings.mlr.press/v16/settles11a.html

[87] Patrice Y. Simard, Saleema Amershi, David Maxwell Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. 2017. Machine Teaching: A New Paradigm for Building Machine Learning Systems. *arXiv preprint arXiv:1707.06742* (2017). http://arxiv.org/abs/1707.06742

[88] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. 2010. The SHOGUN Machine Learning Toolbox. *Journal of Machine Learning Research* 11, 60 (2010), 1799–1802. http://jmlr.org/papers/v11/sonnenburg10a.html

[89] I. Stančin and A. Jović. 2019. An overview and comparison of free Python libraries for data mining and big data analysis. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 977–982. https://doi.org/10.23919/MIPRO.2019.8757088

[90] Jina Suh, Soroush Ghorashi, Gonzalo Ramos, Nan-Chen Chen, Steven Drucker, Johan Verwey, and Patrice Simard. 2019. AnchorViz: Facilitating Semantic Data Exploration and Concept Discovery for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 10, 1, Article 7 (Aug. 2019), 38 pages. https://doi.org/10.1145/3241379

[91] Ying-Peng Tang, Guo-Xiang Li, and Sheng-Jun Huang. 2019. *ALiPy: Active Learning in Python.* Technical Report. Nanjing University of Aeronautics and Astronautics. https://github.com/NUAA-AL/ALiPy available as arXiv preprint https://arxiv.org/abs/1901.03802.

[92] Stefano Teso and Kristian Kersting. 2019. Explanatory Interactive Machine Learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES '19)*. Association for Computing Machinery, New York, NY, USA, 239–245. https://doi.org/10.1145/3306618.3314293

[93] Philip van Allen. 2018. Prototyping Ways of Prototyping AI. *Interactions* 25, 6 (Oct. 2018), 46–51. https://doi.org/10.1145/3274566

[94] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[95] Federico Ghelli Visi and Atau Tanaka. 2020. Interactive Machine Learning of Musical Gesture. arXiv:cs.LG/2011.13487

[96] Michael L. Waskom. 2021. seaborn: statistical data visualization. *Journal of Open Source Software* 6, 60 (2021), 3021. https://doi.org/10.21105/joss.03021

[97] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a

[98] Tom White. 2009. *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

[99] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. 2018. Investigating How Experienced UX Designers Effectively Work with Machine Learning. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. ACM, New York, NY, USA, 585–596. https://doi.org/10.1145/3196709.3196730

[100] Yao-Yuan Yang, Shao-Chuan Lee, Yu-An Chung, Tung-En Wu, Si-An Chen, and Hsuan-Tien Lin. 2017. *libact: Pool-based Active Learning in Python.* Technical Report. National Taiwan University. https://github.com/ntucllab/libact available as arXiv preprint https://arxiv.org/abs/1710.00379.

[101] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* 59, 11 (Oct. 2016), 56–65. https://doi.org/10.1145/2934664

[102] Michael Zbyszynski, Atau Tanaka, and Federico Visi. 2020. Interactive machine learning: Strategies for live performance using electromyography. In *Open Source Biomedical Engineering*, Hugo Silva (Ed.). Springer. http://research.gold.ac.uk/id/eprint/28215/

[103] Xiaojin Zhu. 2015. Machine Teaching: An Inverse Problem to Machine Learning and an Approach Toward Optimal Education. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 4083–4087. http://dl.acm.org/citation.cfm?id=2888116.2888288