Experiment-driven system for machine learning based research management

Ivan Radosavljević*, Đorđe Obradović*, Zora Konjović*, Aleksandra Mitrović*, Stanko Gavrić*, Mladen Vidović*, Nebojša Nešić*

* Singidunum University, Belgrade, Serbia

iradosavljevic@singidunum.ac.rs, djobradovic@singidunum.ac.rs, zkonjovic@singidunum.ac.rs, amitrovic@singidunum.ac.rs, sgavric@singidunum.ac.rs, mvidovic@singidunum.ac.rs, nnesic@singidunum.ac.rs

***Abstract*—Due to the recent rapid expansion of machine learning and its more frequent use as a valuable tool in scientific research the lack of tools that incorporate research project management and tools for machine learning is becoming apparent. Although there are many tools that can be used for project management or machine learning experiments, most of them are intended for commercial use and usually do not support features necessary for scientific research such as experiment versioning, reviewing and publishing of the experiment results. In this paper the authors propose a software solution that provides a unified environment for running machine learning experiments and research project management. The solution is a microservice oriented web platform that can be used to manage research projects, catalogue datasets, define, validate, and execute machine learning experiments. Some of the advantages of the presented solution are its ability to integrate experiments directly into the research project management process, and to automatically version and validate those experiments with respect to the constraints imposed by the authors or reviewers of the experiments. An additional advantage of the solution is the use of the graphical DSL to define experiments, thus allowing researchers who are not skilled in programming to use the platform.**

I. INTRODUCTION

With the advancement of hardware, machine learning is becoming more relevant to research fields other than computer science such as medicine [1, 2, 3], environmental science [4, 5, 6, 7], engineering [8, 9], astronomy [10, 11], etc. With the expansion of users of machine learning such as the experts from the mentioned domains, the lack of software tools designed for research project management as well as the lack of tools for setup, validation, and execution of machine learning experiments is becoming apparent. Although general purpose project management software solutions are commonly available and can be used to manage research projects, they are usually not suitable for this purpose. This is mostly due to this type of software being designed for commercial applications and lacking tools necessary to track the lifecycle of research projects or experiments defined within these projects. On the other hand, software designed to facilitate machine learning experiments is available, but it usually does not integrate well into the existing project management software tools. Also, tools used for machine learning experiments commonly lack features such as experiment versioning, validation and reviewing, thus greatly limiting their use in research environments.

Some of the commonly used project management tools include simple Kanban board implementations such as Trello[1], or more complex, fully featured, tools such as Jira[2]. Simple tools like Trello can be suitable for management of small research projects, but they lack features such as budget planning, timetables, Gantt charts, hierarchical organization of teams and automatic progress tracking. Tools such as Jira, on the other hand, provide all of the above, but due to their complexity can be overwhelming for an average user. They are also primarily designed to facilitate project management of software development projects and as such adhere to methodologies used for organizing software development teams, thus limiting their viability in use for research project management in other fields.

Notable commercial platforms used to facilitate machine learning are usually cloud-based solutions that provide tools and execution environments to train, evaluate, and exploit machine learning algorithms. Although these platforms are market as user friendly and simple solutions, they may require software development knowledge not available to all researchers interested in using these platforms. Another notable example of tools used for machine learning are the RapidMiner Studio[3], and Orange[4]. One of the main advantages of these tools is that they provide a graphical DSL that can be used to implement machine learning algorithms. However, they do not support executing implemented machine learning algorithms in cloud environments thus limiting their capability to handle complex machine learning algorithms or large quantities of data.

Additionally, a common drawback of most of the commercial machine learning platforms is their lack of support for experiment versioning, setup validation and lack of integration with project management software tools.

In this paper we present a software architecture and implementation that provides research project management tools and tools necessary for defining, validating, executing, versioning and reviewing of machine learning experiments and their results.

In the second chapter, we review some of the existing commercial and non-commercial machine learning platforms as well as popular libraries and tools for machine learning. In III, we present the architecture of our solution. In IV we discuss major implementation details of the proposed solution. Finally in V, we conclude with a comparison of our solution and some of the more popular existing solutions and we discuss possible future improvements and research.

[1] https://trello.com/

[2] https://www.atlassian.com/software/jira

[3] https://rapidminer.com/get-started/

[4] https://orangedatamining.com/

II. RELATED WORK

One of the most commonly used platforms for machine learning is the Google Colaboratory[5]. This platform allows researchers to define research experiments using Python notebooks and to deploy them on Googles infrastructure, albeit within certain limits. The platform also leverages other Googles services such as Google Drive which is supported out of the box as a storage for notebooks, datasets and results obtained from experiments. This approach allows users to easily share their findings and data [12]. Some of the drawbacks of this platform are that the users must be proficient in Python to use it and the existence of limits that are imposed on users by the platform. For example, the availability of GPU-s for training more complex machine learning models is not guaranteed, also there is a time limit, of up to 12 hours, imposed on the execution of notebooks. Although in general these limits may not interfere with research relaying on less demanding machine learning algorithms, in case of the more complex ones they can be too obstructive. There are two ways in which these limits can be removed, either users can host the Google Colaboratory on their own hardware, or they can subscribe to Colab Pro, a less restrictive premium service.

OpenML, an open-source platform with a similar goal, is presented in papers [13, 14]. This platform provides researchers with tools necessary to create and share experiments, datasets, and results. Each experiment is described using, what the authors refer to as, tasks which are supposed to provide well-defined problems for the researchers to solve. Each potential solution is referred to as a run. A run is composed of a task and an ML algorithm used to solve the task. Tasks, runs and datasets can be published on the platform using a REST API provided by the platform. Each submitted task, run or dataset is evaluated by the platform and versioned allowing other participants to build upon existing works published on the platform. Although the platform is strongly oriented towards machine learning it does not provide tools for machine learning, instead it relies on external tools to create machine learning algorithms and datasets and to communicate via the provided web API to store the results they produce.

Another example of a non-commercial solution is the NSML platform. One of the goals of this platform is to alleviate problems associated with resource management in machine learning environments. They achieve this by introducing a centralized approach for scheduling distributed machine learning tasks. Each task is supplied as single container that can be executed by a global scheduler. Besides containers designated as the machine learning containers, authors introduce storage containers that can be used to store datasets, intermediate results, final results and source code. Another notable feature of this platform is its web user interface which enables users to setup and monitor training and exploitation of machine learning algorithms through a graphical user interface. One of the major open questions noted by the authors of the platform is the platforms inability to train reinforcement learning models due to its design being tailored to suit the needs of data-driven machine learning tasks [15].

[5] https://colab.research.google.com/

In [16] authors introduce a platform that utilizes a specially designed declarative language to define machine learning tasks. The core of the platform is the so-called master node which parses the declarative definition of the task and constructs a model based on the specified task. The master node then pulls the data necessary for training the constructed model and spawns slave nodes which execute the training process. The master node also monitors the processes executed on slave nodes. A noteworthy feature of this platform is that it provides an explain function, similar to one found in database systems, which allows users to get insight into how the model was created based on the user specification.

The authors of [17] describe a production scale machine learning platform based on TensorFlow library. The platform is composed of components for data analysis, data transformation, data validation, model training, model evaluation and validation, and serving trained models. The data analysis component is designed to provide users with statistical overview of the data, the data transformation component transforms data in a format applicable to the model that is trained. The data validation component alerts users in case of unsuitable or corrupt data. The trainer component is used to optimize the parameters of the model, whilst the model evaluation and validation component provides mechanisms to evaluate and validate trained models. Finally, the serving component allows users to exploit multiple trained models at the same time and to deploy them for production purposes. Although the platform is designed with TensorFlow as its backbone, the authors note that any machine learning library could be used. The authors also note that the components in their system are tightly coupled and that any changes to the machine learning algorithms provided by the platform could induce changes in multiple components of the platform.

III. PROPOSED ARCHITECTURE

The architecture of the machine learning platform presented in this paper is conceived as to fulfill needs of machine learning experts as well as domain specialists who utilize machine learning tasks to solve problems in their domains. Thus, to support a broad specter of use cases, the platform is designed to be modular and scalable. This is achieved by utilizing a microservice architecture for core components of the platform. A global architecture of the Platform is shown in Fig 1.

The first component of the global architecture is the Project management platform. This component is a common interface through which the users of the platform can manage high level tasks such as organizing teams, assigning tasks to team members, budget planning, defining experiments, as well as low level experiment

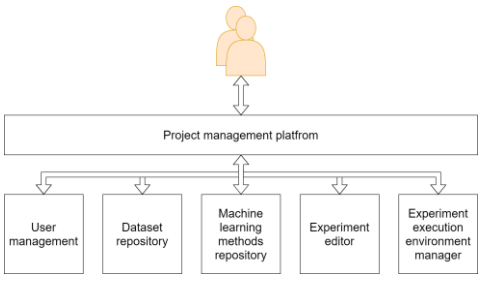

Figure 1. Overview of the global architecture

related tasks such as accessing and manipulating datasets, scheduling machine learning task executions, and collecting experiment results. Due to the nature of its designated use this component is the only component that directly communicates with all other components of the Platform.

The first of the fully decoupled components is the User management component. This component provides user authentication and authorization services, where user authorities are specified through groups, roles, and resource permissions. This approach to defining user authorities allows for fine grained authority checks which may be required in other components such as the Project management platform where users can have specific roles and access only specific resources within a single research team.

The Dataset repository component is tasked with providing and managing datasets and their metadata. This component can manage both locally and remotely stored datasets and is a common source for datasets on the Platform. In order for this component to be used within the platform it must provide a REST API that other components can use to access the datasets. Besides being used by the Project management component which provides a web UI that users of the platform can use to store new datasets or to search for existing datasets, this component is also used by the Experiment editor and Experiment execution component. These two components use the Dataset repository as a source for all the datasets that they can manipulate. The architecture of the Dataset repository component is shown in Fig 2.

The Machine learning methods repository is the main source of all implementations and documentation of machine learning methods provided by the platform. Besides providing the source code of the methods, the repository also keeps track of versions of available implementations and their dependencies. It can be accessed from the Project management component to search for available methods or to add new methods to the repository. Additionally, the Experiment editor and the Experiment execution environment component can also access this repository. The Experiment editor can use the methods stored therein to create new experiments or new machine learning methods that can be stored in the repository. The experiment execution environment component uses this repository in order to prepare the execution environment by installing necessary machine learning methods and their dependencies. The architecture of this component is shown in Fig 3.

The Experiment editor is tasked with providing a compiler and a graphical editor for a custom designed high-level language that can be used to specify machine learning experiments. The compiler provided by this component is modular and is capable to translate the high-

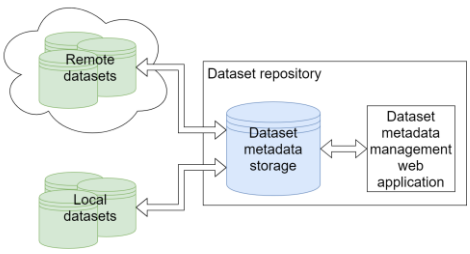

Figure 2. Architecture of the Dataset repository

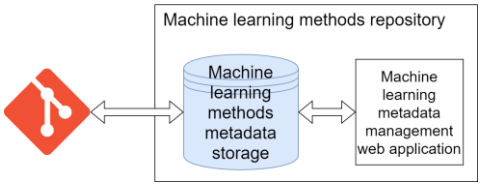

Figure 3. Architecture of the Machine learning methods repository

level declarative input code into a concrete programming language such as Python, C++ or any other language for which an appropriate module is provided. Besides providing means to specify machine learning experiments, this component can be used to specify new machine learning methods, and searchable metadata for machine learning methods and experiments. Also, it automatically tracks dependencies of specified experiments and methods, and versions experiments and methods according to any change. Figure 4 shows the architecture of the Experiment editor component.

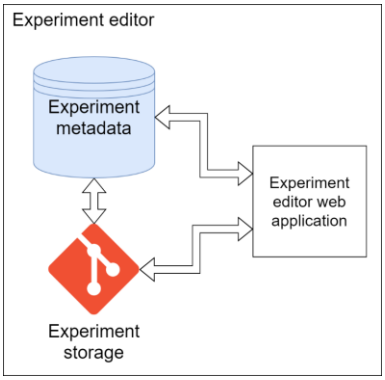

Figure 4. Architecture of the Experiment editor component

Finally, the Experiment execution environment manager is a component that manages containers that are used to execute machine learning tasks. This component can create containers on demand, track their status, and schedule machine learning tasks with respect to their resource demand and current platform load. It has access to the Machine learning repository and Dataset repository component. Figure 5 shows the architecture of the Experiment execution environment component.

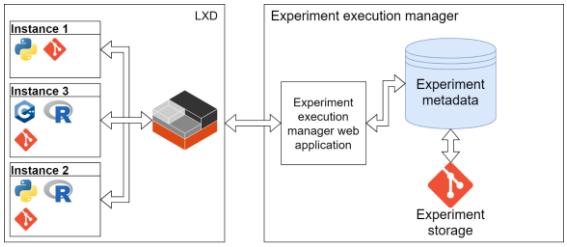

Figure 5. Architecture of the Experiment execution environment component.

IV. IMPLEMENTATION DETAILS

All of the components described in III, except the User management component, are implemented using Spring Boot framework [18]. The orchestration of the aforementioned components is implemented using the Spring Cloud Netflix project [19]. The UI used for the Project management component is implemented using the Angular framework [20] and a custom-built library of generic UI components based on Angular Material library [21].

The User management component is implemented using Keycloak [22], an open-source off-the-shelf identity and access management software. Keycloak is in such a manner as to suit the needs of all the components involved in the implementation of the platform.

The Dataset repository is implemented as a Spring Boot application that enables uploading datasets and managing their metadata. The metadata is stored as DCAT2 [23] compatible RDF triples.

The Machine learning methods repository is another Spring boot application that exposes a REST API through which it is possible to add access and add new machine learning methods and their metadata. This repository utilizes Git [24] repositories to store and version local implementations of machine learning methods. Installation instructions or links to remotely stored machine learning methods are stored in metadata only. The metadata provided by this component is also DCAT2 compatible.

The Experiment execution environment is implemented as Spring boot application wrapping the existing REST API provided by LXD [25]. LXD is a system container manager for Linux containers. Its REST API allow for full control over the process of configuring and creating containers, however, it lacks tools necessary to facilitate advanced user authorization and authentication and its API can be too cumbersome to use directly. In order to alleviate these problems a wrapper that relies on authorization and authentication provided by the User management component is implemented. This component exposes simplified REST API that allows users to create, run, and shutdown containers, upload and download files from containers, and to execute tasks and track progress of the executed tasks. The configuration of the containers can be provided by users through the Project management platform in form of RDF compliant files. This choice of configuration system allows the Experiment execution environment to suggest to users the best containers for each task or to automatically deploy tasks on containers that fulfil requirements of a given task.

An example of a simple use case of uploading a file, then processing it and retrieving the results is given in Fig 6. Steps flow from top to bottom. Step 3, Execute a task, can either be an asynchronous task or synchronous task. In a case of the asynchronous task the Experiment execution environment returns an URL to the websocket that a client can subscribe to in order to receive any messages from the task. In a case of a synchronous task the returned URL can be used to pull the task progress data from the experiment environment on client's demand.

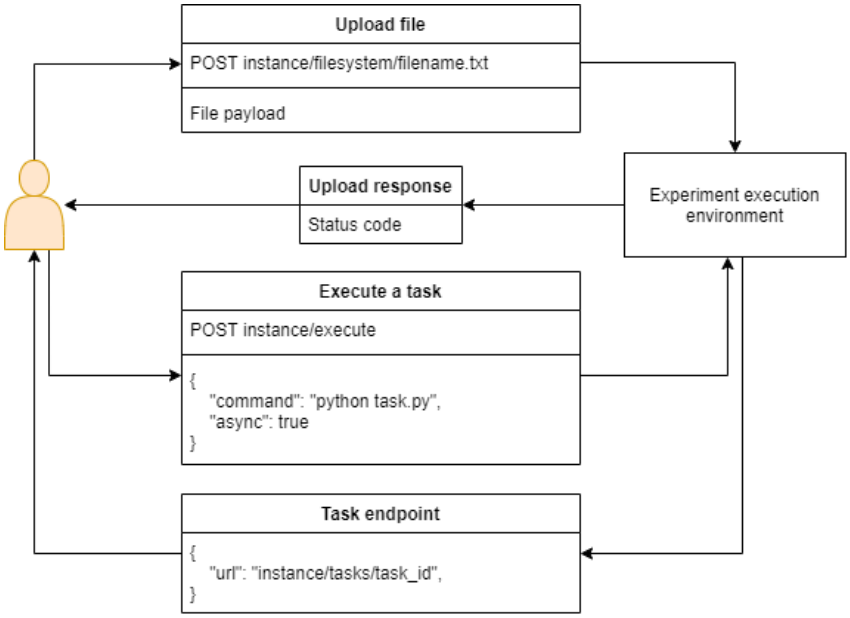

Figure 6. A process of uploading and processing a file using the Experiment execution environment component.

V. CONCLUSION

In this paper we review existing machine learning platforms and propose an architecture for implementing one such platform. We also show current implementation details of platform components. Compared to existing solutions our platform is easily scalable due to its reliance on microservice architecture design principles and emphasis on decoupled major components and supports both high-level research project management and low-level operations such as experiment and dataset creation. An important feature of our solution is its built-in support for versioning of all the research related data and its reliance on DCAT2 standard for specifying metadata. The use of DCAT2 allows for advanced queries to be executed on the platform in order to search for datasets, machine learning methods or any other data available on the

platform. One notable drawback of the presented solution its complexity that makes it less adequate for small research projects.

Further development of this platform is directed towards streamlining the UI for the Project management component, implementing custom user defined filters for dataset and machine learning repository components search, and implementing the Experiment editor component.

ACKNOWLEDGEMENT

The authors acknowledge funding provided by the Science Fund of the 604 Republic of Serbia #GRANT No. 6524105, AI—ATLAS.

REFERENCES

[1] A. Rajkomar, J. Dean, and I. Kohane, "Machine Learning in Medicine," New England Journal of Medicine, vol. 380, no. 14, pp. 1347–1358, Apr. 2019, doi: 10.1056/NEJMra1814259.

[2] H. Seo et al., "Machine learning techniques for biomedical image segmentation: An overview of technical aspects and introduction to state-of-art applications," Medical Physics, vol. 47, no. 5, pp. e148–e167, 2020, doi: https://doi.org/10.1002/mp.13649.

[3] Amisha, P. Malik, M. Pathania, and V. K. Rathaur, "Overview of artificial intelligence in medicine," J Family Med Prim Care, vol. 8, no. 7, pp. 2328–2331, Jul. 2019, doi: 10.4103/jfmpc.jfmpc_440_19.

[4] A. Masih, "Machine learning algorithms in air quality modeling," Global Journal of Environmental Science and Management, vol. 5, no. 4, pp. 515–534, Oct. 2019, doi: 10.22034/GJESM.2019.04.10.

[5] G. Kaur, J. Gao, S. Chiao, S. Lu, and G. Xie, "Air Quality Prediction: Big Data and Machine Learning Approaches," International Journal of Environmental Science and Development, vol. 9, pp. 8–16, Jan. 2018, doi: 10.18178/ijesd.2018.9.1.1066.

[6] M. Hino, E. Benami, and N. Brooks, "Machine learning for environmental monitoring," Nat Sustain, vol. 1, no. 10, Art. no. 10, Oct. 2018, doi: 10.1038/s41893-018-0142-9.

[7] J. Li, A. D. Heap, A. Potter, and J. J. Daniell, "Application of machine learning methods to spatial interpolation of environmental variables," Environmental Modelling & Software, vol. 26, no. 12, pp. 1647–1659, Dec. 2011, doi: 10.1016/j.envsoft.2011.07.004.

[8] J. Morgenroth, U. T. Khan, and M. A. Perras, "An Overview of Opportunities for Machine Learning Methods in Underground Rock Engineering Design," Geosciences, vol. 9, no. 12, Art. no. 12, Dec. 2019, doi: 10.3390/geosciences9120504.

[9] K. Dimililer, H. Dindar, and F. Al-Turjman, "Deep learning, machine learning and internet of things in geophysical engineering applications: An overview," Microprocessors and Microsystems, vol. 80, p. 103613, Feb. 2021, doi: 10.1016/j.micpro.2020.103613.

[10] D. Baron, "Machine Learning in Astronomy: a practical overview," arXiv:1904.07248 [astro-ph], Apr. 2019, Accessed: Jun. 04, 2021. [Online]. Available: http://arxiv.org/abs/1904.07248

[11] N. M. Ball and R. J. Brunner, "Data mining and machine learning in astronomy," Int. J. Mod. Phys. D, vol. 19, no. 07, pp. 1049–1106, Jul. 2010, doi: 10.1142/S0218271810017160.

[12] T. Carneiro, R. V. M. D. NóBrega, T. Nepomuceno, G. Bian, V. H. C. D. Albuquerque, and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," IEEE Access, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[13] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning," SIGKDD Explor. Newsl., vol. 15, no. 2, pp. 49–60, Jun. 2014, doi: 10.1145/2641190.2641198.

[14] J. N. van Rijn et al., "OpenML: A Collaborative Science Platform," in Machine Learning and Knowledge Discovery in Databases, Berlin, Heidelberg, 2013, pp. 645–649. doi: 10.1007/978-3-642-40994-3_46.

[15] N. Sung et al., "NSML: A Machine Learning Platform That Enables You to Focus on Your Models," arXiv:1712.05902 [cs], Dec. 2017, Accessed: Mar. 22, 2021. [Online]. Available: http://arxiv.org/abs/1712.05902

[16] T. Kraska, A. Talwalkar, and J. Duchi, "MLbase: A Distributed Machine-learning System," p. 7.

[17] D. Baylor et al., "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax NS Canada, Aug. 2017, pp. 1387–1395. doi: 10.1145/3097983.3098021.

[18] "Spring Boot." https://spring.io/projects/spring-boot/ (accessed Jun. 04, 2021).

[19] "Spring Cloud Netflix." https://spring.io/projects/spring-cloud-netflix (accessed Jun. 04, 2021).

[20] "Angular." https://angular.io/ (accessed Jun. 04, 2021).

[21] "Angular Material," Angular Material. https://material.angular.io/ (accessed Jun. 04, 2021).

[22] "Keycloak." https://www.keycloak.org/ (accessed Jun. 04, 2021).

[23] "Data Catalog Vocabulary (DCAT) - Version 2." https://www.w3.org/TR/vocab-dcat-2/ (accessed Jun. 04, 2021).

[24] "Git." https://git-scm.com/ (accessed Jun. 04, 2021).

[25] "Linux Containers - LXD - Introduction." https://linuxcontainers.org/lxd/introduction/ (accessed Jun. 04, 2021).