MLaaS: Decoupling Application Intelligence from Application Logic

John Byabazaire*, Dimitris Chatzopoulos*
*School of Computer Science, University College Dublin, Ireland
{ john.byabazaire, dimitris.chatzopoulos}@ucd.ie

***Abstract*—Existing research recognises the critical role played by machine leaning in various business applications to improve decision making. While there has been an increase in the application of machine learning to various business domains, most businesses lack the experience, knowledge, or infrastructure to develop, deploy, monitor and retrain the machine learning systems. This paper proposes a micro-services based architecture that delivers machine learning solutions as a service. Unlike existing solutions that focus only on deployment, our solution offers end-to-end integration including model drift monitoring and automatic retraining.**

***Index Terms*—Machine Learning, Models, Micro-services**

I. Introduction

Technological advancements in computing have led to an increase in the amount of data generated today. This is partly from the wide spread of sensor networks and connected mobile devices [1]. As a results, this has given companies and organisations the power to harness such data through complex algorithms and machine learning to improve decision making. Machine learning (ML) allows a computer system to learning from data without explicit instructions and make new discoveries [2]. Today, ML has been widely adapted in many business and research domains due its ability to learn from data and provide invaluable insights.

While large companies continue to develop and deploy their on ML tools, smaller and medium sized ones struggle in comparison [3], [4]. This is due to the lack of the infrastructure and technical expertise to develop, deploy and maintain such solutions. As a result, Machine Leaning as a Service (MLaaS), a new paradigm that allows users to utilize ML without the need to focus on the infrastructure has been on the raise [3]. This way, the user can focus on the application logic.

Most of the solutions focus on the 3 stages of the ML cycle used in [3]. These include data acquisition and understanding, modeling/training and finally deployment. While this strategy works in the initial stages, ML models have been show to drift as they are introduced to new data [5], requiring retraining. In this paper, we proposed a MLaaS solution that is based on microservice architecture that extends current solutions to include model monitoring for drift and retraining through a feedback mechanism.

II. System design

This section describes the proposed architecture . It is based on FASTAPI open source python library. As shown in the figure 1 the design of the architecture is broken down into two main stages: the API for the initialization phase, this includes model registration and discovery, and the API for the deployment phase, this includes endpoints for deployment, inference and drift monitoring. The following section describe processes in each stage.

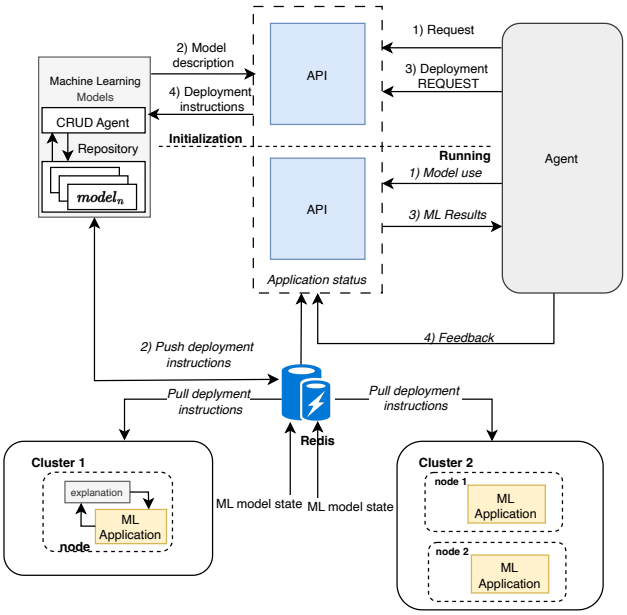

Fig. 1. The overall data and system flow.

A. Model description

The process start with model registration. We use a declarative description to define the parameters of each model. Each ML model is defined by a set of properties. These include model type, its internal hyperparameters, the minimum resources required to train and run that model, and its explainability characteristics. A user, or agent queries the model repository with such parameters whenever they need to identify matching models for a task. Figure 2 show some of the properties of the models.

B. Invocation

The invocation stage is broken down into a series of REST API calls:

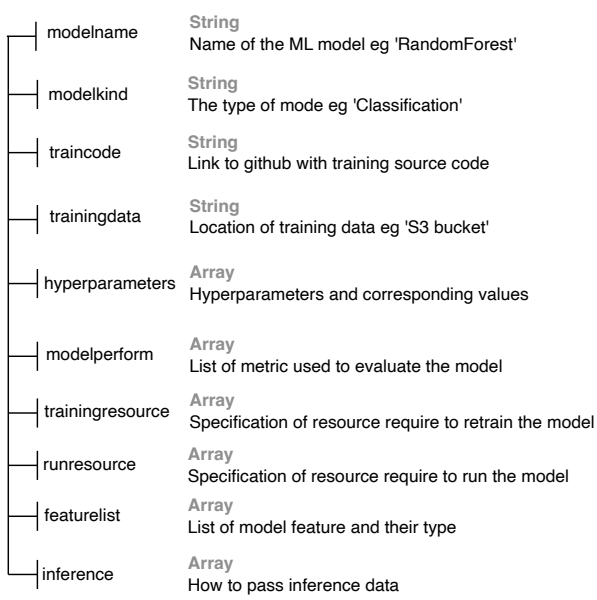

Fig. 2. Parameters used to define a model

1) Request. Here, the user/agent specifies the type of the ML model it needs using the ML model description defined above.
2) Model description. Once a matching model is found, the API returns a list of all the ML models that match the agent's requirements.
3) Deployment REQUEST. At this stage, the agent picks one model and sends back a request to API to deploy the selected ML model.
4) Deployment instructions. The API pushes the request to the queue for deployment and returns a deployment ID to the agent to be used in the next stage. This is a continuation of step 3 above.

C. Deployment

The second stage of the cycle is the deployment phase. After the user acknowledges the deployment of an ML model, the request is pushed to a redis queue. Based on the resource requirements of the ML model to be deployed, a worker node(s) will be initialized using kubernetes. The systems supports two modes of model deployment; 1) open deployment, that is, the model can be deployed on any node, 2) strict deployment, the model has to be deployed on a specific node. The second options supports privacy preserving applications.

D. Inference

All ML models are package and deployed as docker containers. This ensure faster setup and tire down, but also guarantees they can easily be deployed in any environment that has a docker engine. Each of the deployment exposes an endpoint that is used for inference. The system supports two modes of inference; 1) The user can pass the data directly to the endpoint, or pass a reference to the data. The data can be in an AWS S3 bucket for example.

E. Monitoring and retraining

For every inference made, the application send feedback which is used to monitor model performance for drift. This feedback mechanism is only currently possible in the open deployment mode.

F. Decommission

Once a user no longer needs the ML model, they send a decommission command, and the application will be destroyed. It important to note that this is a 2 step process, first, the ML application is cached, and if no other users/agents requires the same model, it is then completely destroyed.

III. Related work

The increasing amount of data generated and used has led to wide spread integration of ML solutions to extract value. Many organizations, however, lack the technical expertise and the necessary infrastructure. Due to this, several solutions have been proposed that offer MLaaS. This section highlight some of those works.

Ribeiro et. al. [4] proposed an open source flexible, and non-blocking architecture that delivers MLaaS based on service component architecture and focusing on predictive modeling. The proposed solution was evaluated in a case study based on forecasting electricity. Zhao et. al. [6] presented a platform based on micro-services that lets business integrate ML functionality into other business applications. In a similar way, Rao et. al. [7] proposed a containerized solution, which gives users the ability to develop, improve and deploy ML models with ease. The solution also features a GUI.

Several other solutions have been proposed [8]–[10], however, the focus of most of them is model training, storage and deployment. They neglect model drift monitoring and automatic retraining. One might argue that such functions are present in commercial tool like; SageMaker by Amazon, Google AI, or even Azure machine by Microsoft. However, it is important to note that the implementation details of these products are not publicly available.

IV. Discussion, conclusion and open challenges

In this paper we introduce a micro-service based architecture that decouples ML solutions from application logic. This is aimed to extend current solutions which focus on only data acquisition, model training and deployment and neglect drift monitoring and retraining as necessary steps for MLaaS solution. By incorporating these step, our proposed solution offer an end-to-end ML solution.

While the proposed and existing solution offer means for smaller and medium sized companies to leverage ML solutions without worrying about infrastructure and technical challenges, other open challenges still exist. The main challenge is privacy. Since most ML solutions need access to data to train and retrain, and there is a increase on reliance on personal data to develop custom models, it is challenging to develop MLaaS solution because regulatory bodies do not allow organization to share such data.

ACKNOWLEDGMENT

This work has received funding from the Horizon Europe research and innovation programme of the European Union, under grant agreement no 101092912, project MLSysOps.

REFERENCES

[1] Moussa Aboubakar, Mounir Kellil, and Pierre Roux. A review of iot network management: Current status and perspectives. *Journal of King Saud University-Computer and Information Sciences*, 34(7):4163–4176, 2022.

[2] Ethem Alpaydin. *Machine learning*. MIT press, 2021.

[3] Robert Philipp, Andreas Mladenow, Christine Strauss, and Alexander Völz. Machine learning as a service: Challenges in research and applications. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, pages 396–406, 2020.

[4] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 896–902. IEEE, 2015.

[5] Berkman Sahiner, Weijie Chen, Ravi K Samala, and Nicholas Petrick. Data drift in medical machine learning: implications and potential remedies. *The British Journal of Radiology*, 96(1150):20220878, 2023.

[6] Heyang Qin, Syed Zawad, Yanqi Zhou, Lei Yang, Dongfang Zhao, and Feng Yan. Swift machine learning model serving scheduling: a region based reinforcement learning approach. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 384–397, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018.

[9] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.

[10] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proceedings of the 2017 Internet Measurement Conference*, pages 384–397, 2017.