

CSE484 (Cloud Computing)

Machine Learning as a Service (MLaaS)

by

Abir Ahammed Bhuiyan

20101197

Ahnaf Tahmid

20101555

Saadman Omar Siddique

19301180

Mohammad Tamim

20301428

Department of Computer Science and Engineering

School of Data and Sciences

Brac University

Spring 2024

© 2024. Brac University
All rights reserved.

Abstract

The deployment and accessibility of machine learning models and algorithms have undergone a paradigm shift with the introduction of Machine Learning as a Service (MLaaS). With this service-oriented strategy, businesses may leverage machine learning's potential without requiring a lot of infrastructure or specialized knowledge. Through APIs or user-friendly interfaces, MLaaS platforms offer a variety of activities, such as model training, deployment, and management. MLaaS democratizes machine learning by utilizing cloud computing resources, making it available to companies of all sizes and industries. In this paper, we proposed an MLaaS platform that aims to break down barriers to artificial intelligence (AI). With a user-friendly interface, this cloud-based platform will enable users to train, deploy, and manage machine learning models without the need for a lot of expertise in being able to code AI or ML models.

Keywords: MLaaS, ML, API, AI, Cloud Computing, SaaS, Docker

Table of Contents

Abstract	i
Table of Contents	ii
List of Figures	iii
List of Tables	iv
Nomenclature	v
1 Introduction	1
1.1 Paper Organization	1
2 Literature Review	2
2.1 Related Works	2
2.2 Strengths and Shortcomings of Related Works	3
2.2.1 Strengths	3
2.2.2 Shortcomings	3
3 Demonstration	5
3.1 How to run	5
3.2 Create	7
3.3 Train	9
3.4 Predict	10
4 Methodology	11
4.1 Toolsets	11
4.2 System Analysis	12
4.2.1 Create	12
4.2.2 Train	12
4.2.3 Predict	13
4.3 API endpoints	13
5 Conclusion	14
Bibliography	15
Appendix: Thesis Github Repository	16

List of Figures

3.1	Cloning the git repository	6
3.2	Building the Docker image name 'heart'	6
3.3	Installing the dependencies and running the server	7
3.4	Example of a csv	7
3.5	UI for creating a new instance	8
3.6	Database entry	8
3.7	Table of the created instances	9
3.8	Predictable label in the UI	9
3.9	Log of an instance	9
3.10	Prediction UI of an instanace	10
3.11	Space separated value to fetch prediction	10
4.1	Creation of a new instance	12
4.2	Training of an instance	12
4.3	Predicting from an instance	13

List of Tables

4.1	API Endpoints	13
-----	-------------------------	----

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AI Artificial Intelligence

API Application Programming Interface

ML Machine Learning

MLaaS Machine Learning as a Service

SaaS Software as a Service

SOC Separation Of Concerns

Chapter 1

Introduction

The field of machine learning (ML) has seen remarkable growth and progress in recent years, bringing about revolutionary changes in various industries and domains. However, despite its immense potential, the implementation of ML technologies has been hindered by the complexity of the process and the requirement for specialized programming expertise. This has resulted in a significant gap, limiting access to the benefits of ML only to a few individuals with technical skills. In response to this challenge, our project endeavors to democratize access to machine learning by developing a user-friendly platform tailored for individuals with no programming experience. By eliminating the barriers posed by technical complexity, our platform aims to empower users from diverse backgrounds to harness the capabilities of ML algorithms for their specific needs and applications. The overarching goal of this project is to bridge the gap between the potential of machine learning and its accessibility, thereby fostering innovation and collaboration across industries. Through intuitive interfaces and guided workflows, users can seamlessly navigate the ML process, from data preprocessing to model deployment, without the need for coding expertise. This project is significant because it allows more people, even those with little experience in machine learning, to use powerful AI tools. This democratization of AI makes it possible for companies and individuals to use machine learning for better automation, creativity, and decision-making. The MLaaS project proposes a cloud-based platform with user-friendly tools that allow users to train, deploy, and manage machine learning models without needing internal ML specialists. This makes it easy for users to take advantage of the potential of ML.

1.1 Paper Organization

The remainder of the study is organized as follows: In Chapter 2, a thorough overview of the existing literature is presented along with their strength and shortcomings which are related to our project. The next chapter, Chapter 3, a small demonstration of our project has been given. Chapter 4 outlines the methodology or implementation of our project. Finally, Chapter 5 provides a conclusion and discusses scopes of future works.

Chapter 2

Literature Review

In this chapter, the research studies done by other authors are discussed.

2.1 Related Works

A Dockerized ML Model Transfer Learning Platform refers to a sophisticated system designed for efficiently training and deploying machine learning models within a Docker container environment. Docker, acting as a containerization platform, provides a consistent and isolated environment for ML model development and deployment, ensuring seamless portability across various computing environments. By adopting Docker containerization, the platform facilitates rapid prototyping, experimentation, and collaboration among data scientists and ML practitioners. Overall, a Dockerized ML Model Transfer Learning Platform offers a robust infrastructure for organizations to harness the power of machine learning, enabling accelerated model development and deployment while maintaining consistency, reliability, and scalability across the ML lifecycle.

In recent years, the advent of cloud computing has revolutionized the landscape of machine learning deployment and accessibility. Cloud-based machine-learning platforms offer scalable infrastructure and services that facilitate the development, deployment, and management of machine-learning models without the need for significant upfront investment in hardware or specialized expertise [5]. These platforms typically provide a range of tools and services, including data storage, model training, inference, and monitoring, thereby streamlining the end-to-end machine-learning workflow [3]. By leveraging cloud-based machine learning platforms, organizations can accelerate innovation, reduce time to market, and lower operational costs, thereby gaining a competitive edge in the rapidly evolving digital landscape. Moreover, federated learning has emerged as a promising approach to training machine learning models across decentralized edge devices while preserving data privacy and security [2]. In federated learning, model training occurs locally on edge devices, and only model updates are transmitted to a central server, thereby minimizing the need for raw data exchange [1]. This distributed learning paradigm enables organizations to harness the collective knowledge from diverse data sources while respecting user privacy and regulatory constraints [4]. Federated learning holds great potential for various applications, including healthcare, IoT, and personalized recommenda-

tion systems, where data privacy and locality are paramount concerns [6].

Furthermore, the integration of machine learning with DevOps practices has gained traction in recent years, giving rise to the concept of MLOps (machine learning operations). MLOps aims to apply DevOps principles and best practices to the machine learning lifecycle, encompassing model development, training, deployment, and monitoring [7]. By adopting MLOps practices, organizations can establish robust pipelines for automating and orchestrating the end-to-end machine learning workflow, thereby improving collaboration, reproducibility, and scalability.

2.2 Strengths and Shortcomings of Related Works

In the next part, we discuss some of the strengths and weaknesses of the papers we discussed above.

2.2.1 Strengths

Comprehensive Coverage: The papers by [5] and [3] offer comprehensive coverage of their respective topics. They provide in-depth analyses of machine learning as a service (MLaaS) and cloud computing, offering insights into the evolution, challenges, and future directions of these domains. This comprehensive coverage ensures that readers gain a holistic understanding of the subject matter, making these papers valuable resources for researchers, practitioners, and industry professionals.

Cutting-edge Research: [2], [1], [4], and [6] present cutting-edge research on federated learning, a novel approach to training machine learning models across decentralized edge devices. These papers contribute significant advancements to the field by addressing key challenges such as data privacy, communication efficiency, and model aggregation in federated learning settings. Their rigorous methodologies, theoretical frameworks, and experimental results demonstrate the feasibility and effectiveness of federated learning for various real-world applications.

2.2.2 Shortcomings

Limited Practical Guidance: While the papers by [2], [1], [4], and [6] offer valuable insights into federated learning algorithms and methodologies, they may lack practical guidance for implementing federated learning systems in real-world scenarios. These papers often focus on theoretical aspects and experimental evaluations, leaving readers with limited guidance on practical issues such as system architecture, deployment considerations, and integration with existing infrastructure. As a result, practitioners may face challenges in translating theoretical concepts into actionable solutions.

Lack of Real-world Case Studies: While the papers reviewed provide theoretical frameworks and experimental results, they often lack real-world case studies or practical applications demonstrating the efficacy of the proposed approaches in real-world settings. Practical case studies can provide valuable insights into the challenges, best practices, and lessons learned from deploying machine learning systems in diverse application domains. The absence of real-world case studies limits the generalizability and applicability of the research findings, hindering the adoption of federated learning and other machine-learning techniques in practical settings.

In summary, while the reviewed papers offer valuable contributions to their respective fields, they exhibit strengths in terms of comprehensive coverage and cutting-edge research but also weaknesses related to limited practical guidance and the absence of real-world case studies. Addressing these weaknesses can enhance the relevance, applicability, and impact of future research in machine learning and related domains.

Chapter 3

Demonstration

To create this project, we utilized the widely used NodeJS framework along with ExpressJS for the backend. For the frontend, we chose Handlebars to dynamically generate HTML based on the data received from the backend. To containerize our application, we opted for Docker and utilized Dockerode to interact with the Docker Engine API via NodeJS. Additionally, Pycaret, a Python library, was used for machine learning (ML) purposes.

In this project, we have implemented Machine Learning as a Service (MLaaS) using various technologies mentioned earlier. The project includes a user interface (UI) where the user can upload a CSV file of their dataset, along with the name of the target column that needs to be predicted. They can also specify the type of prediction, such as classification or regression, the ML model they want to train, and a preferred project name. Once the details are inputted, the user can create a container to train their ML model using the uploaded dataset. The creation of the container is kept abstract from the user. After the training is complete, they can predict the target value for the dataset. While predicting, different labels should be separated by a space instead of a comma.

This project supports the creation and training of multiple ML models simultaneously by running them in individual containers to ensure that the models do not interfere with one another in any way. Lastly, users can view the information of their models and predict using the trained models.

3.1 How to run

To run the project first we have to clone the project from '<https://github.com/eniac00/MLaaS>' or we can download the zip.

To clone the repository we can use this command:

```
git clone https://github.com/eniac00/MLaaS
```

After cloning we have to go in the directory 'MLaaS' and build the docker image from the provided docker file in the directory. One thing to remember while building

```
[ab1r@ahammed-20101197] [~]
git clone https://github.com/eniac00/MLaaS
Cloning into 'MLaaS'...
remote: Enumerating objects: 165, done.
remote: Counting objects: 100% (165/165), done.
remote: Compressing objects: 100% (112/112), done.
remote: Total 165 (delta 67), reused 133 (delta 38), pack-reused 0
Receiving objects: 100% (165/165), 181.60 KiB | 281.00 KiB/s, done.
Resolving deltas: 100% (67/67), done.
[ab1r@ahammed-20101197] [~]
```

Figure 3.1: Cloning the git repository

the docker image is that, the docker image name must be ‘heart’.

This command can be used to build the docker image name ‘heart’:

```
docker build -t heart .
```

```
[ab1r@ahammed-20101197] [~/MLaaS] [main]
docker build -t heart .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  512.5kB
Step 1/5 : FROM ubuntu:latest
----> 3db8720ecbf5
Step 2/5 : RUN apt-get update && apt-get install -y python3 python3-pip
----> Running in 3f953075e746
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1078 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2308 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1789 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
```

Figure 3.2: Building the Docker image name ‘heart’

Next, we have to make sure ‘nodejs’ and ‘npm’ are installed in our machine, without those we will not be able to install the required dependencies and run the server.

Use this command to install the dependencies:

```
npm install
```

Use this command to run the server:

```
npm run dev
```

As we can see in the figure 3.3 after executing the commands the server is running on port ‘3000’. Now we can visit the UI using <IP>:3000 (e.g. 127.0.0.1:3000).

```
[abir@ahammed-20101197] [~/MLaaS] [main]
o npm install

added 147 packages, and audited 148 packages in 6s

21 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
[abir@ahammed-20101197] [~/MLaaS] [main]
o npm run dev

> mlaas@1.0.0 dev
> nodemon server -e js,hbs

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,hbs
[nodemon] starting `node server.js`
Server running on port 3000
```

Figure 3.3: Installing the dependencies and running the server

3.2 Create

For our demonstration purpose we will be using a CSV file about penguins as shown in the figure 3.4.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
1							
2	Adelie	Torgersen	39.1	18.7	181	3750	MALE
3	Adelie	Torgersen	39.5	17.4	186	3800	FEMALE
4	Adelie	Torgersen	40.3	18	195	3250	FEMALE
5	Adelie	Torgersen					
6	Adelie	Torgersen	36.7	19.3	193	3450	FEMALE
7	Adelie	Torgersen	39.3	20.6	190	3650	MALE
8	Adelie	Torgersen	38.9	17.8	181	3625	FEMALE
9	Adelie	Torgersen	39.2	19.6	195	4675	MALE
10	Adelie	Torgersen	34.1	18.1	193	3475	
11	Adelie	Torgersen	42	20.2	190	4250	
12	Adelie	Torgersen	37.8	17.1	186	3300	

Figure 3.4: Example of a csv

To create a new instance we can open the URL in any browser as discussed before, in our case it is 192.168.0.108:3000. Visiting that URL we will see a UI shown in figure 3.5. There we can provided all the necessary information along with the CSV dataset. After pressing the create button we will be redirected to a new page where can see our newly created instance as shown in figure 3.7.

The screenshot shows a web browser window with the address bar displaying "Not secure | 192.168.0.108:3000". The page has a header with the title "MLaaS" and a button labeled "Show Instances". Below the header, there is a sub-header that reads "Run Machine Learning models without code" and "create train & predict". The main content area is a form with the following fields:

- Work Name:** A text input field containing "penguin_classification".
- Learning:** A dropdown menu with "MachineLearning" selected.
- Task:** A dropdown menu with "Classification" selected.
- Model Name:** A dropdown menu with "SVM" selected.
- Target:** A text input field containing "species".
- CSV/ZIP:** A button labeled "Choose File" followed by the text "penguins.csv".

At the bottom of the form is a "Create" button.

Figure 3.5: UI for creating a new instance

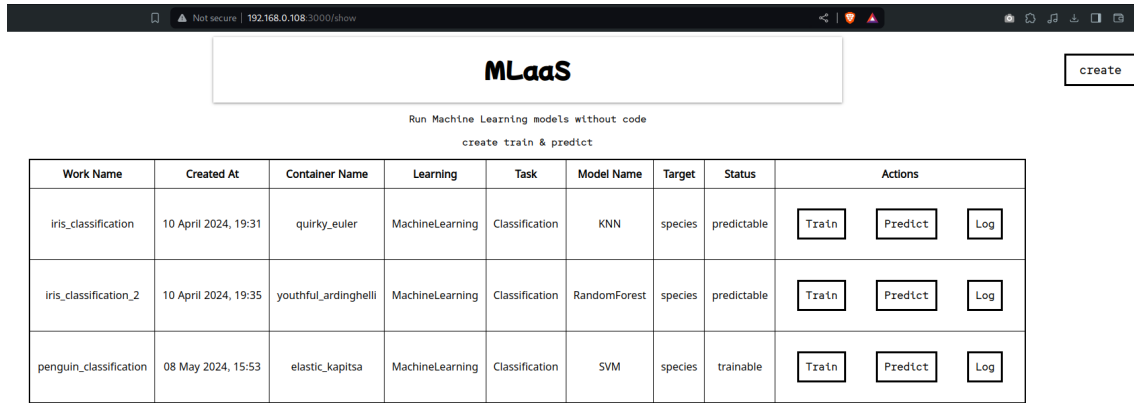
```

1    },
40   {
1     "id": "1486d4f841",
2     "workName": "penguin_classification",
3     "createdAt": "08 May 2024, 15:53",
4     "containerName": "elastic_kapitsa",
5     "status": "predictable",
6     "learning": "MachineLearning",
7     "task": "Classification",
8     "modelName": "SVM",
9     "target": "species",
10    "csvFile": "./resources/1486d4f841/data.csv",
11    "features": [
12      "island",
13      "bill_length_mm",
14      "bill_depth_mm",
15      "flipper_length_mm",
16      "body_mass_g",
17      "sex"
18    ]
19  }
20 ]
21 }

```

NORMAL db.json

Figure 3.6: Database entry



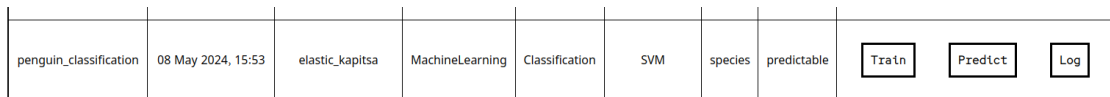
The screenshot shows the MLaaS web interface. At the top, there's a header with the MLaaS logo and a 'create' button. Below the header, a subtitle reads 'Run Machine Learning models without code' and 'create train & predict'. The main content is a table with columns: Work Name, Created At, Container Name, Learning, Task, Model Name, Target, Status, and Actions. The table lists three instances: 'iris_classification', 'iris_classification_2', and 'penguin_classification'. Each instance has buttons for 'Train', 'Predict', and 'Log' in the Actions column.

Work Name	Created At	Container Name	Learning	Task	Model Name	Target	Status	Actions
iris_classification	10 April 2024, 19:31	quirky_euler	MachineLearning	Classification	KNN	species	predictable	<button>Train</button> <button>Predict</button> <button>Log</button>
iris_classification_2	10 April 2024, 19:35	youthful_ardinghelli	MachineLearning	Classification	RandomForest	species	predictable	<button>Train</button> <button>Predict</button> <button>Log</button>
penguin_classification	08 May 2024, 15:53	elastic_kapitsa	MachineLearning	Classification	SVM	species	trainable	<button>Train</button> <button>Predict</button> <button>Log</button>

Figure 3.7: Table of the created instances

3.3 Train

In the table we can see three buttons appearing for every instances viz Train, Predict and Log. We can press on the Train button to train our instance using the provided information. If the training is successful then we will see that Status column entry for that instance will change from ‘trainable’ to ‘predictable’ as shown in figure 3.8.

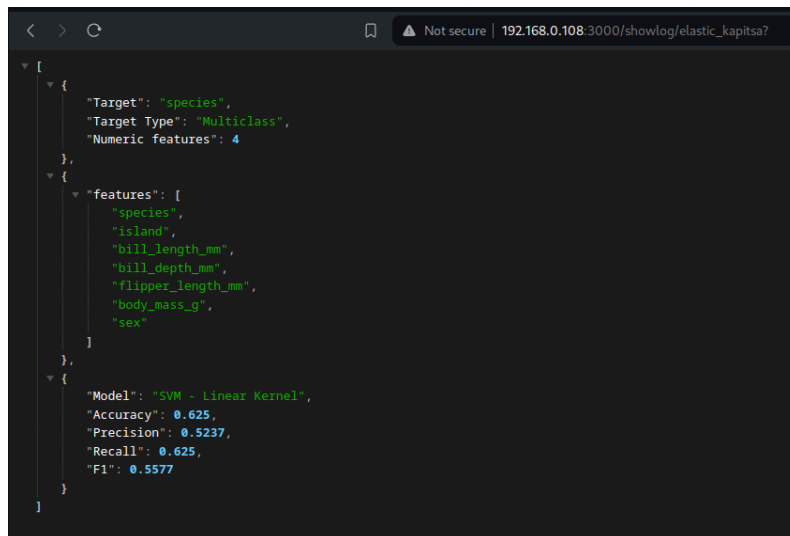


The screenshot shows a single row from the table in Figure 3.7. The 'Status' column now shows 'predictable' instead of 'trainable'. The 'Actions' column still contains the 'Train', 'Predict', and 'Log' buttons.

penguin_classification	08 May 2024, 15:53	elastic_kapitsa	MachineLearning	Classification	SVM	species	predictable	<button>Train</button> <button>Predict</button> <button>Log</button>
------------------------	--------------------	-----------------	-----------------	----------------	-----	---------	-------------	--

Figure 3.8: Predictable label in the UI

In the Status label is ‘predictable’ that means the training was successful and we can press on the Log button to see the log of that instance as shown in figure 3.9.



The screenshot shows the log output for the 'penguin_classification' instance. The log is displayed in a dark-themed code editor. It shows the target, target type, numeric features, features list, model name, and performance metrics (Accuracy, Precision, Recall, F1).

```

{
  "Target": "species",
  "Target Type": "Multiclass",
  "Numeric features": 4,
  "features": [
    "species",
    "island",
    "bill_length_mm",
    "bill_depth_mm",
    "flipper_length_mm",
    "body_mass_g",
    "sex"
  ],
  "Model": "SVM - Linear Kernel",
  "Accuracy": 0.625,
  "Precision": 0.5237,
  "Recall": 0.625,
  "F1": 0.5577
}

```

Figure 3.9: Log of an instance

3.4 Predict

To predict from an instance we can press on the Predict button and a new page will appear displaying the necessary information with an input box as shown in figure 3.10.

The screenshot shows a web browser window with the URL `192.168.0.108:3000/showpredict/elastic_kapitsa?`. The page has a header with the **MLaaS** logo and a 'Show Instances' button. Below the header, it says 'Run Machine Learning models without code' and 'create train & predict'. The main content area displays the following information:

```
Name: penguin_classification
Learning: MachineLearning
Task: Classification
Model: SVM
Target: species
Features: island,bill_length_mm,bill_depth_mm,flipper_length_mm,body_mass_g,sex
```

Below this information is a text input box with the placeholder text 'Only space separated values'. A 'Send Data' button is located below the input box. The result of the prediction is displayed as 'Result: hello'.

Figure 3.10: Prediction UI of an instance

We can provide the space separated values and press on the Send Data to get the prediction like figure 3.11.

This screenshot shows the same MLaaS prediction UI as Figure 3.10, but with different input and output. The text input box now contains the space-separated values 'Biscoe 48.8 16.2 222 6000 MALE'. After clicking the 'Send Data' button, the result displayed is 'Result: Gentoo'.

Figure 3.11: Space separated value to fetch prediction

Chapter 4

Methodology

In this project, we aim to develop a SaaS that will provide a user interface where users can upload their datasets and choose their desired ML models. Also, they will be able to run those models in multiple dockerized containers without delving into the technical nitty gritty.

For this purpose, we developed a web application where in the frontend users can interact with the system via a GUI while in the backend server, machine learning models, argument parsing and docker containerization were communicating with each other to achieve the goal.

4.1 Toolsets

- **NodeJS** : Runtime environment for our backend.
- **PyCaret** : Low-code machine learning library in Python that automates machine learning workflows.
- **ExpressJS** : Web application framework for building RESTful APIs with Node.js.
- **Docker** : Virtualization platform used for SOC (i.e. separation of concerns) between instances.
- **Dockerode** : Docker Remote API module for managing docker containers or images programmatically.
- **LowDB** : Simple and fast JSON database.
- **Handlebars** : Templating engine for dynamic frontend.

4.2 System Analysis

‘MLaaS’ backend system can be divided into 3 major core parts create, train and predict. The flow diagram for each of the process is given below.

4.2.1 Create

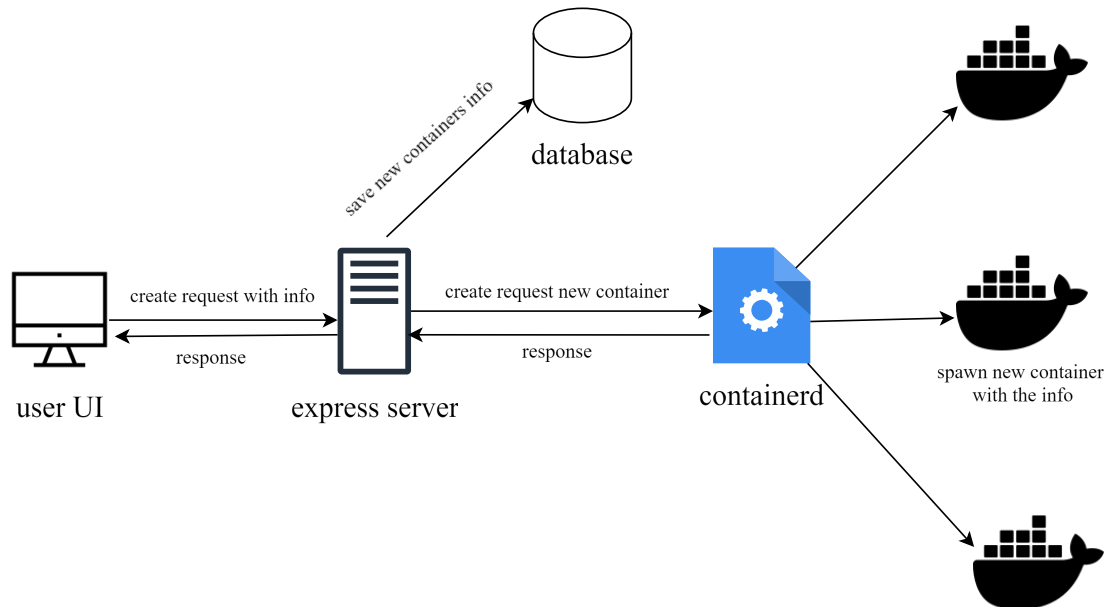


Figure 4.1: Creation of a new instance

4.2.2 Train

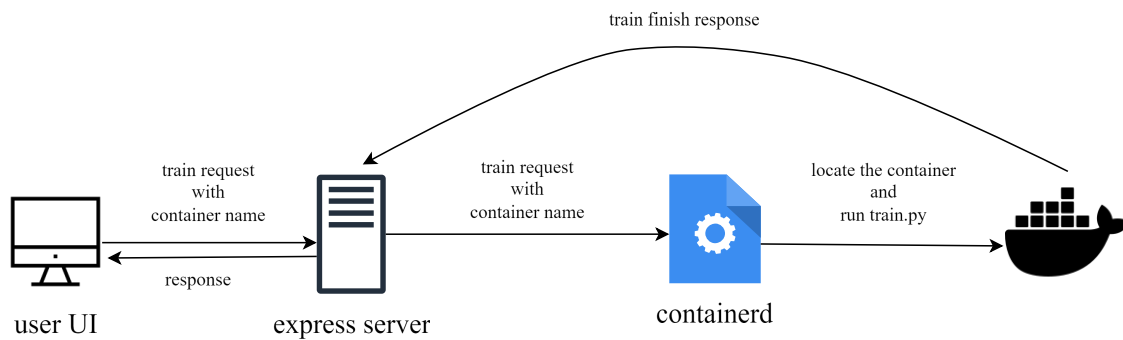


Figure 4.2: Training of an instance

4.2.3 Predict

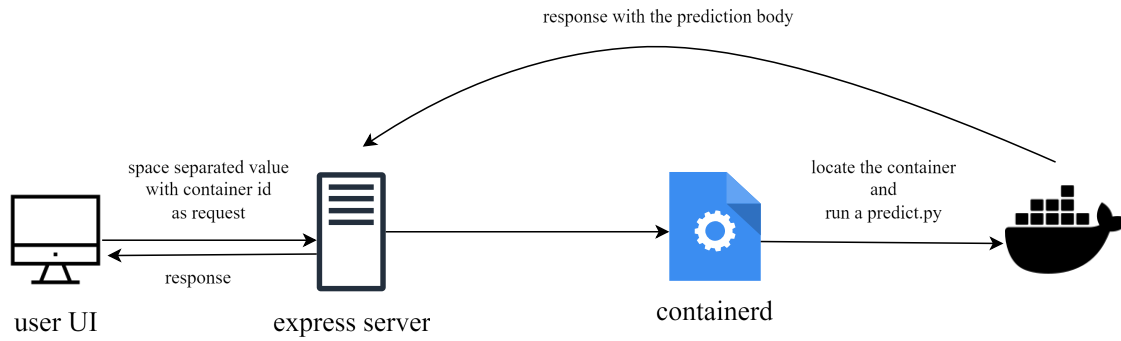


Figure 4.3: Predicting from an instance

4.3 API endpoints

Endpoint	Verb	Description
/	GET	Show the creation UI
/create	POST	Create a new instance by providing info
/show	GET	Show the table UI of instances
/train	POST	Train an instance by providing id
/predict	POST	Predict from an instance by providing id and values to be predicted
/showlog	GET	Show log of an instance by providing container name
/showpredict	GET	Show the prediction UI
/signal	GET	Containers (train.py) use this route to let the server know if training is finished or not

Table 4.1: API Endpoints

Chapter 5

Conclusion

In this project, we made a prototype of a Machine Learning as a Service whose main focus is to enable people with limited knowledge about ML and very little to no knowledge about coding to be able to use and train ML models for their needs. It is an open-source project that is available to everyone. The project also encompasses a user-friendly UI that can be understood and used by non-technical people or people from fields other than computer science. At the moment, only a handful of models are available to train for classification and regression. In the future, we look forward to making a lot of upgrades to this project which includes the addition of robust and complex preprocessing capabilities and increasing the number of models that can be trained and used.

Bibliography

- [1] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv preprint arXiv:1610.02527*, 2016.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.
- [3] G. S. Puri, R. Tiwary, and S. Shukla, “A review on cloud computing,” in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE, 2019, pp. 63–68.
- [4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
- [5] J. Kaur and R. Rani, “A survey on machine learning as a service,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, pp. 177–183, 2020.
- [6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [7] <https://www.gartner.com/en/information-technology/glossary/mlops>., Accessed: 2024-5-2.

Project Github Repository

All the source code along with an instruction manual can be found at
<https://github.com/eniac00/MLaaS>.