



MLOps Project: Scrabble Player Rating

Deep Learning Final Project

Written By:

- El belhadji Soumaya
- Faham Hassan

Class:

2nd year master MBD
2022/2023

Supervised By:

- Pr. EL AACHAK Lotfi

Table of Contents

O- General Introduction	4
O.1 Purpose of work:	4
I- Used Technologies and Platforms	5
I.1 Kaggle.....	5
I.2 Angular	5
I.3 Docker	6
I.4 Kubernetes.....	6
I.5 Jupyter:	7
I.6 Python libraries:.....	7
II- Kaggle Competition	9
II.1 Description	9
II.2 Deep Learning Model	10
II.3 Competition Result	10
III- MLOps Project.....	12
III.1 Fast API development	12
III.2 API deployment	13
III.3 Consumption via web SPA	15
Conclusion.....	16

Table of Figures

<i>Figure 1: Architecture of the MLOps Project</i>	<i>5</i>
<i>Figure 2: Kaggle Competition Overview</i>	<i>10</i>
<i>Figure 3: Kaggle competition private score</i>	<i>11</i>
<i>Figure 4: FastAPI request body</i>	<i>12</i>
<i>Figure 5: FastAPI response body</i>	<i>12</i>
<i>Figure 6: The Dockerfile</i>	<i>13</i>
<i>Figure 7: Running the docker container</i>	<i>13</i>
<i>Figure 8: Applying the deployment.yaml file</i>	<i>14</i>
<i>Figure 9: The Kubernetes pods created</i>	<i>14</i>
<i>Figure 10: Web Platform of type SPA</i>	<i>15</i>

O- General Introduction

Machine Learning Operations, or MLOps, an engineering term that was built around this need is becoming, a new trend, it is a practice that combines software engineering and data engineering principles with machine learning (ML) to enable organizations to effectively build, deploy, and manage ML systems at scale. The goal of an MLOps project is to streamline the development and deployment of ML models, while also ensuring that they are reliable, scalable, and maintainable.

This report presents an overview of an MLOps project based on a deep learning model developed during a Kaggle competition. The project involved the implementation of a number of tools and practices, in order to create, deploy and consume the model,

In the following sections, we will describe the background and motivation for the project, the specific tools and practices that were implemented, and the results of the project. We will also discuss any challenges and lessons learned during the project.

O.1 Purpose of work:

The purpose of this MLOPs project is to compete in a Kaggle competition using deep learning techniques. The goal of the chosen competition is to develop a deep learning model that can accurately predict the ratings of players based on Scrabble gameplay. The project will utilize Docker to containerize the model and its dependencies, allowing it to be easily deployed and run on any system with Docker installed. Kubernetes will be used to manage the containers and ensure that the model is highly available and can scale up or down as needed. And FastAPI to build the API for the model, that will be consumed in a convenient and user-friendly web platform of SPA (Single Page Application) type to perform the predictions.

Overall, the purpose of this work is to create a robust and scalable MLOPs solution that makes it easy to deploy, manage, and use deep learning models in a production environment, following the architecture below :

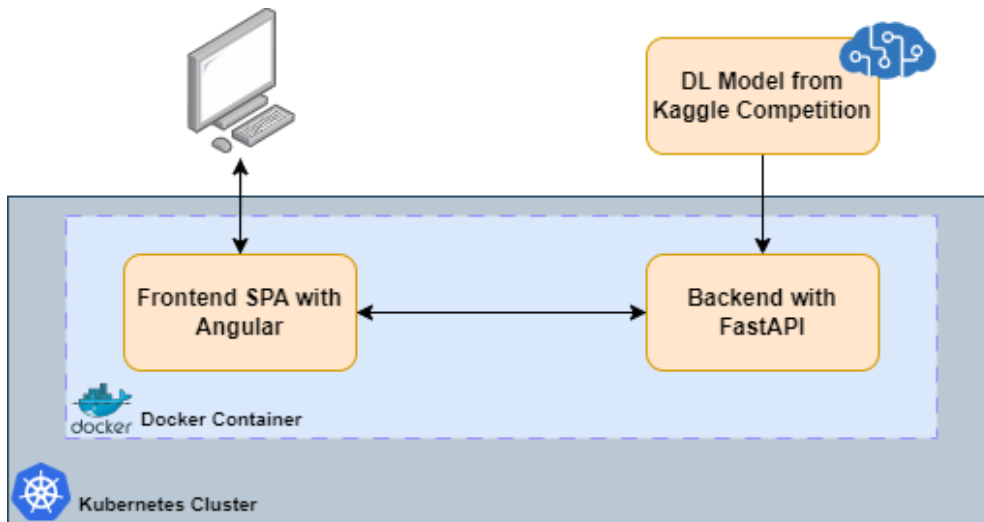


Figure 1: Architecture of the MLOps Project

I- Used Technologies and Platforms

I.1 Kaggle



Kaggle is a website that hosts data science competitions, as well as provides a cloud-based workbench for developing and running data science code.

Kaggle competitions are online challenges in which data scientists and machine learning developers compete to build the best models for a given task. Competitions are often sponsored by organizations that want to solve a specific problem or advance the state of the art in a particular field. Participants in a competition are given access to a dataset, and they compete to build the best model based on the metrics defined in the competition.

I.2 Angular



Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

Angular is used as the frontend of the MEAN stack, consisting of MongoDB database, Express.js web application server framework, Angular itself (or AngularJS), and Node.js server runtime environment.

1.3 Docker



Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Using Docker, you can develop an application locally, then package it into a container that can be run in any other environment with the same libraries and dependencies. This makes it easy to deploy applications to different environments, such as development, staging, and production, without worrying about differences in the underlying infrastructure.

Docker uses containerization technology to create lightweight, standalone, and portable containers for applications. A container is a standardized unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

1.4 Kubernetes



Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).

Kubernetes is designed to work with containerization technologies such as Docker and provides a number of features for managing containerized applications at scale, including:

- Deployment and roll-out of applications: Kubernetes can deploy and update applications quickly and safely, rolling out changes in a controlled way and rolling back if necessary.
- Scaling applications: Kubernetes can automatically scale applications up or down based on demand, ensuring that resources are used efficiently.
- Load balancing: Kubernetes can distribute incoming traffic across multiple replicas of an application to ensure that it can handle high levels of traffic.
- Self-healing: Kubernetes can detect and recover from failures, ensuring that applications are always available.

I.5 Jupyter:



Jupyter Notebook can connect to many kernels to allow programming in different languages. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once.

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

I.6 Python libraries:

FastAPI:



FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints. It is developed and maintained by Sebastián Ramírez.

It uses Pydantic, a library for data validation and parsing, to provide request validation, input parsing, and request/response modeling. This makes it easy to create robust and reliable APIs that are easy to use and maintain.

FastAPI is designed to be easy to use and learn, with a simple and intuitive API, and has a number of features that make it well-suited for building APIs.

Pandas:



Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

Numpy:



NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project. In Python we have lists that serve the

purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

TensorFlow:



TensorFlow is an open-source software library for machine learning and artificial intelligence. It was developed by Google and is used by researchers and developers all over the world to build and deploy machine learning models.

TensorFlow allows you to define, train, and deploy machine learning models using a variety of architectures, including deep neural networks, convolutional neural networks, and recurrent neural networks. It provides a flexible and efficient ecosystem for training and deploying machine learning models at scale.

Uvicorn:



Uvicorn is a lightning-fast ASGI (Asynchronous Server Gateway Interface) server, built on top of uvloop and httptools, which makes it one of the fastest web servers available for Python.

Uvicorn is often used as a production-grade web server for applications built with ASGI frameworks such as FastAPI, Starlette, and Quart. It is designed to be easy to use and provide excellent performance, with a simple and intuitive API.

II- Kaggle Competition

II.1 Description

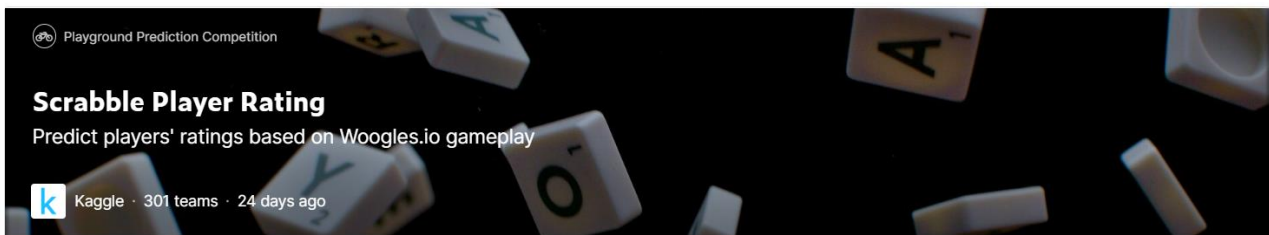


Figure 2: Kaggle Competition Overview

In this Kaggle competition, participants are challenged to build a machine learning model that can accurately predict the rating of Scrabble players based on their past games and other relevant data.

The dataset includes information about ~73,000 Scrabble games played by three bots on Woogles.io: BetterBot (beginner), STEEBot (intermediate), and HastyBot (advanced). The games are between the bots and their opponents who are regular registered users. There are metadata about the games as well as turns in each game (i.e., players' racks and where and what they played, AKA gameplay) to predict the rating of the human opponents in the test set (test.csv). We need to train the model on gameplay data from one set of human opponents to make predictions about a different set of human opponents in the test set.

This dataset with a size of 122.18Mb contains 4 csv files:

- games.csv: Metadata for each game (e.g., who went first, time controls).
- turns.csv: Gameplay data about turns played by each player from start to finish of each game.
- train.csv and test.csv: Final scores and ratings from before a given game was played for each player in each game.

II.2 Deep Learning Model

After the preprocessing phase that consisted of:

- one hot encoding the categorical feature *turn_type* and *location* so the model can understand it.
- Aggregate the turns of each game together by calculating the mean of all the numerical features.
- Label encode *rating_mode*, *lexicon*, *game_end_reason* and *time_control_name* in the train and test sets.

- Remove the bots' data in test set, so we have only the human records to predict their rating.

We build our deep learning model, with the following hyperparameters:

Hyperparameters	Value
Number of hidden layers	4
Activation function	ReLU
Loss function	Mean Squared Error
Optimizer	Adam
Learning Rate	0.001
Evaluation metric	Root Mean Squared Error
Epochs	50

- The ReLU activation function is a simple mathematical function that takes in a real-valued input and outputs the input if it is positive, and zero if it is negative. It is defined as:

$$f(x) = \max(0, x)$$

where x is the input to the activation function.

- The Adam (Adaptive Moment Estimation) optimizer is a variant of stochastic gradient descent that uses moving averages of the parameters to provide a running estimate of the second raw moments of the gradients; the term adaptive in the name refers to the fact that the algorithm "adapts" the learning rates of each parameter based on the historical gradient information.

II.3 Competition Result

The model will be evaluated based on its ability to accurately predict the ratings of players in a holdout test set. The evaluation algorithm in Kaggle competition is RMSE score:

III- MLOps Project

III.1 FastAPI development

To start the MLOps project we will use the saved deep learning model created in the Kaggle competition, our API will use the following features:

`N_games_played: int`
`mean_points_per_round: float`
`mean_move_length: float`
`first_to_play: bool`
`winner: bool`

In the request body we specify the value of each feature:



Figure 4: FastAPI request body

The result of the prediction is shown in the response body:



Figure 5: FastAPI response body

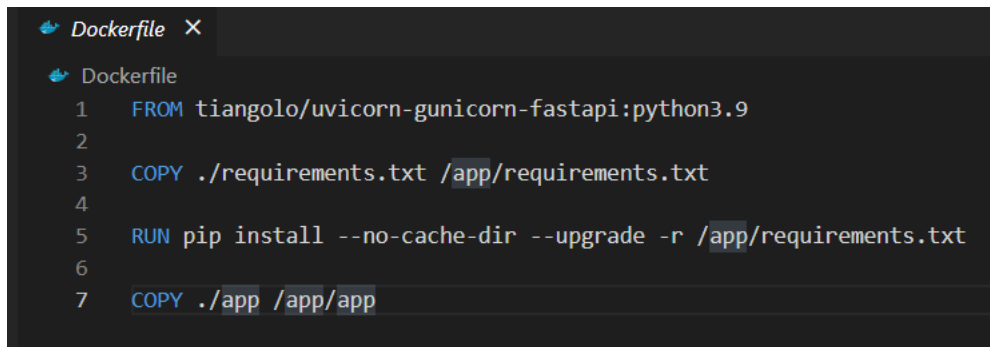
III.2 API deployment

– Dockerizing the API:

Dockerizing the FastAPI involves packaging the API and its dependencies into a Docker container, which can then be easily deployed and run in any environment that supports Docker. There are several benefits to dockerizing an API:

- **Portability:** Docker containers can be easily moved from one environment to another, making it easy to deploy the API on different platforms or in different environments, such as staging, testing, and production.
- **Isolation:** Docker containers provide isolation between the API and the host system, which can help to prevent conflicts between the API and other software running on the same system.
- **Ease of deployment:** Docker containers can be easily deployed on any machine that has Docker installed, which can save time and effort compared to manually setting up and configuring the API in each environment.

To dockerize our API, we created a *Dockerfile* that specifies the steps to build the Docker image for the API. This *Dockerfile* include instructions to install any dependencies required by the API and to copy the API code into the image:



```
1 FROM tiangolo/uvicorn-gunicorn-fastapi:python3.9
2
3 COPY ./requirements.txt /app/requirements.txt
4
5 RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt
6
7 COPY ./app /app/app
```

Figure 6: The Dockerfile

Once we have created the *Dockerfile*, we can use the `docker build` command to build the image, then the `docker run` command to start a container based on the image and run the API:






	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	 stoic_blackwell 249cebd481fc 	ranking-prediction-app:latest	Running	80:80 	4 minutes ago 	

Figure 7: Running the docker container

– Deploy the API with Kubernetes:

Overall, deploying an API with Kubernetes allows for easy management and scaling of the API, as well as robust monitoring and logging capabilities. It helps make a reliable and scalable API deployment that can handle a high volume of traffic.

To set up a Kubernetes cluster, we created a new file called *deployment.yaml* , this file is connected to the Docker image created earlier.

```
C:\Users\Pc\Documents\Study\MST\S3\Big Data Analytics\DL-FAstAPI>kubect1 apply -f deployment.yaml
service/ranking-prediction-service created
deployment.apps/ranking-prediction-app created
C:\Users\Pc\Documents\Study\MST\S3\Big Data Analytics\DL-FAstAPI>_
```

Figure 8: Applying the deployment.yaml file

The purpose of this deployment is to make the API available online to clients, for that we need to expose it using a Kubernetes service. Which is a logical abstraction over a set of pods, and it allows clients to access the API through a stable IP address and DNS name.


```
C:\Users\Pc\Documents\Study\MST\S3\Big Data Analytics\DL-FAstAPI>kubect1 get pods
NAME                                READY   STATUS    RESTARTS   AGE
ranking-prediction-app-bb9866bb-4dq2w 1/1     Running   0           108s
ranking-prediction-app-bb9866bb-4lhgf 1/1     Running   0           108s
ranking-prediction-app-bb9866bb-9w9fg 1/1     Running   0           108s
ranking-prediction-app-bb9866bb-jq2hc 1/1     Running   0           108s
ranking-prediction-app-bb9866bb-kwtpt 1/1     Running   0           108s
```

Figure 9: The Kubernetes pods created

III.3 Consumption via web SPA

In order to consume this model, we will create a web platform of type SPA using Angular that will take inputs of the features from the user and post them to fastAPI, in its part it will predict the rating of the player based on these features, and then pass the result to client side:

Predict The Rating of Scrabble Players



Enter Scrabble Game Player Data and check his rating.

Number of Games Played:

Average Points Per Round:

Average Move Length:

First To Play ?

Yes ☒ No ☐

Winner ?

Yes ☐ No ☒

Predict

Predicted Rating : **1538**

Figure 10: Web Platform of type SPA

Conclusion

In conclusion, this MLOps project was a success. By using Angular for the frontend, we were able to create a user-friendly and intuitive interface for interacting with the DL model. The FastAPI server allowed us to easily expose a REST API for the frontend to use, and provided excellent performance and reliability.

Docker allowed us to package the DL model server and FastAPI server into a container image, which made it easy to deploy and run the servers in different environments. Using Kubernetes, we were able to set up and manage a cluster of machines that ran the containerized servers, ensuring high availability and scalability.

Overall, the combination of Angular, FastAPI, Docker, and Kubernetes proved to be a powerful and effective stack for building and deploying an MLOps project.