**Deploying a Machine Learning Model: Part 2 — Local Deployment**

A quick demo using the MLEM package

[[Tejas Ekawade](https://medium.com/@TejasEkawade?source=post_page-----1c36d029ce7a--------------------------------)](https://medium.com/@TejasEkawade?source=post_page-----1c36d029ce7a--------------------------------)

[Tejas Ekawade](https://medium.com/@TejasEkawade?source=post_page-----1c36d029ce7a--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F46a3e6c6898d&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40TejasEkawade%2Fdeploying-a-machine-learning-model-part-2-local-deployment-1c36d029ce7a&user=Tejas+Ekawade&userId=46a3e6c6898d&source=post_page-46a3e6c6898d----1c36d029ce7a---------------------post_header-----------)

4 min read

·

Aug 24, 2023

18

I wrote about the basic concepts of machine learning deployment in an article — <https://medium.com/@ekawade1394/deploying-a-machine-learning-model-getting-started-2644392c8953>  
In this article, we are going to focus on the practical aspect — deploying a model locally.

I came across this library called ‘***MLEM***’ by iterative.ai that simplifies the process of serving models locally — using servers like FastAPI, Streamlit — and in production — docker, kubernetes. The website states the following:

*MLEM is a tool to easily package, deploy, and serve Machine Learning models. It seamlessly supports a variety of scenarios like real-time serving and batch processing.*

This library allows us to deploy models locally with ease as many of the details are abstracted such as creating a docker file among other things.  
However, the user should ensure that the Docker Desktop is installed.

Here are the following steps for deploying a model locally:

1. Set up a virtual environment and install the required packages
2. Build and save the model
3. Serve the model locally
4. Create a folder structure for files
5. Build a docker image
6. Spin up a container to run the application
7. Test the model endpoint

As the focus is on deployment, we are going to build a very simple Random Forest model. Let’s get started

1. ***Set up a virtual environment and install the packages*:**The steps are mentioned in this article — [https://medium.com/me/stats/post/cb865c3147a7](https://medium.com/@ekawade1394/installing-python-virtual-environments-cb865c3147a7)

pip install pandas scikit-learn mlem[fastapi]

***2. Build and save a model:***The script below saves a Random Forest model in the model's directory. However, there is an additional file created(rf.mlem) that saves the metadata for the model.

from sklearn.datasets import load\_iris  
from sklearn.ensemble import RandomForestClassifier  
import joblib  
import pandas as pd  
from mlem.api import save  
  
# load data  
data, y = load\_iris(return\_X\_y=True, as\_frame=True)  
# instantiate and train model  
rf = RandomForestClassifier(  
 n\_jobs=2,  
 random\_state=42,  
)  
rf.fit(data, y)  
# save to file  
# joblib.dump(rf, "models/rf")  
save(rf, "models/rf", sample\_data=data)

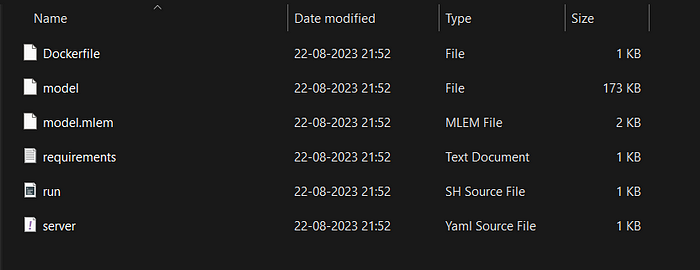
from mlem.api import load  
import pandas as pd  
  
model = load("models/rf") # RandomForestClassifier  
features = [  
 "sepal length (cm)",  
 "sepal width (cm)",  
 "petal length (cm)",  
 "petal width (cm)",  
]  
df = pd.DataFrame([[0, 1, 2, 3]], columns=features)  
y\_pred = model.predict\_proba(df)  
print(y\_pred)

***3. Serve the model locally:*** Generally, you would like to start a server locally and need a framework like FastAPI to implement an endpoint. However, this intermediate step is skipped and we can use our model as an endpoint  
to serve the model predictions locally with the following command:

mlem serve fastapi --model models/rf

***4. Create a folder structure:***This will consist of the dockerfile, requirements.txt, the model, and mlem files that are required to create a docker image.  
However, we just have to type this command and all the files will be created.

mlem build docker\_dir --model models/rf --server fastapi --target build/docker



The directory with the created files

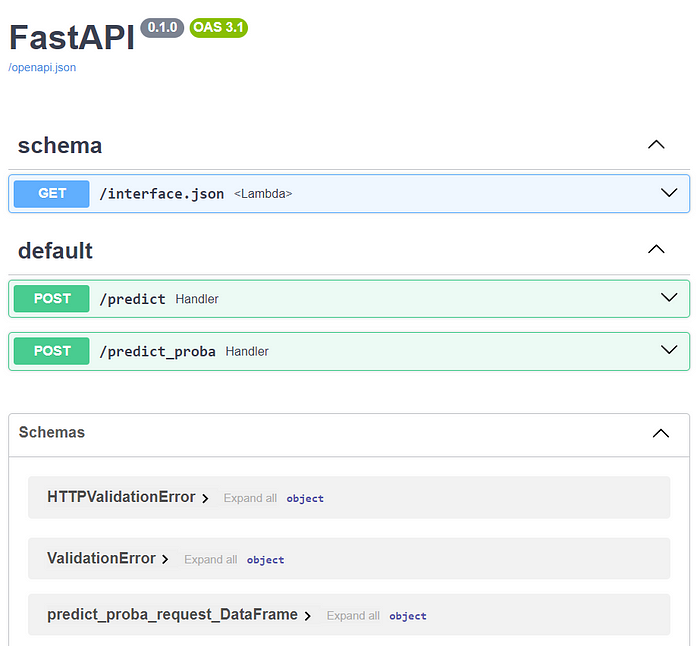
***5. Build a docker image:***This is the docker syntax for building an image.

docker build .\build\docker -t mlem-model:latest

***6. Spin up a container to run the docker image:***

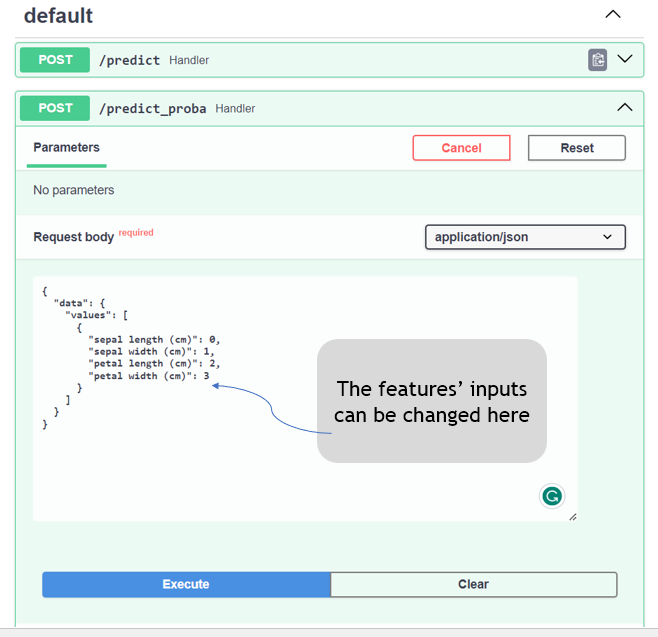
docker run -p 8080:8080 mlem-model:latest

***7. Testing the endpoint:***We haven’t created a UI here for our application but FastAPI provides us with a UI (Swagger UI)that lets us test if the API is working correctly. The UI can be accessed with the link — <http://localhost:8080/docs> (Note that the port will change depending on the user-specified port)

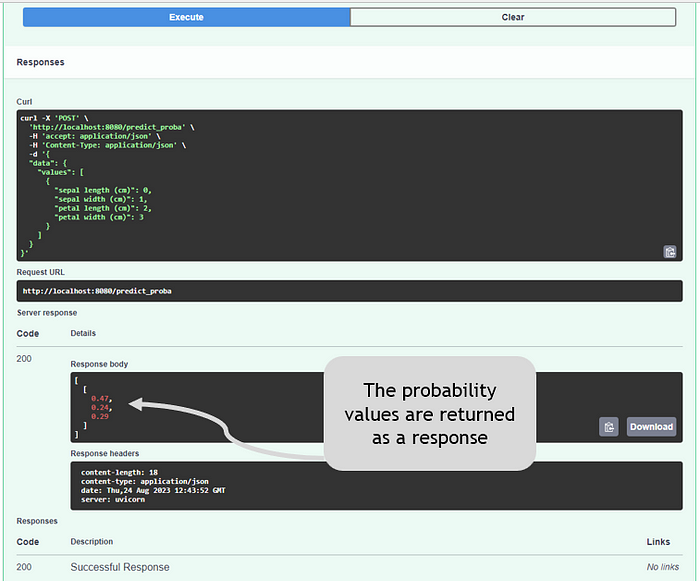


The Swagger UI

We will test the predict\_proba function using the endpoint and check if it returns the model output as the response.



Let’s look at what the response looks like:



Woohoo! Our deployed machine learning model is sending back the model response using the API endpoint.

**What’s next?**

This was a quick way to deploy a Machine Learning model locally. While this library provides an easy way to do that (at the cost of creating MLEM files), it is good to learn commands for creating a docker file, and writing FastAPI code so that we can customize the Machine Learning application based on our use case.

Now that we have deployed a model locally, the logical next step is to deploy it on a cloud platform. The sky is the limit 😆