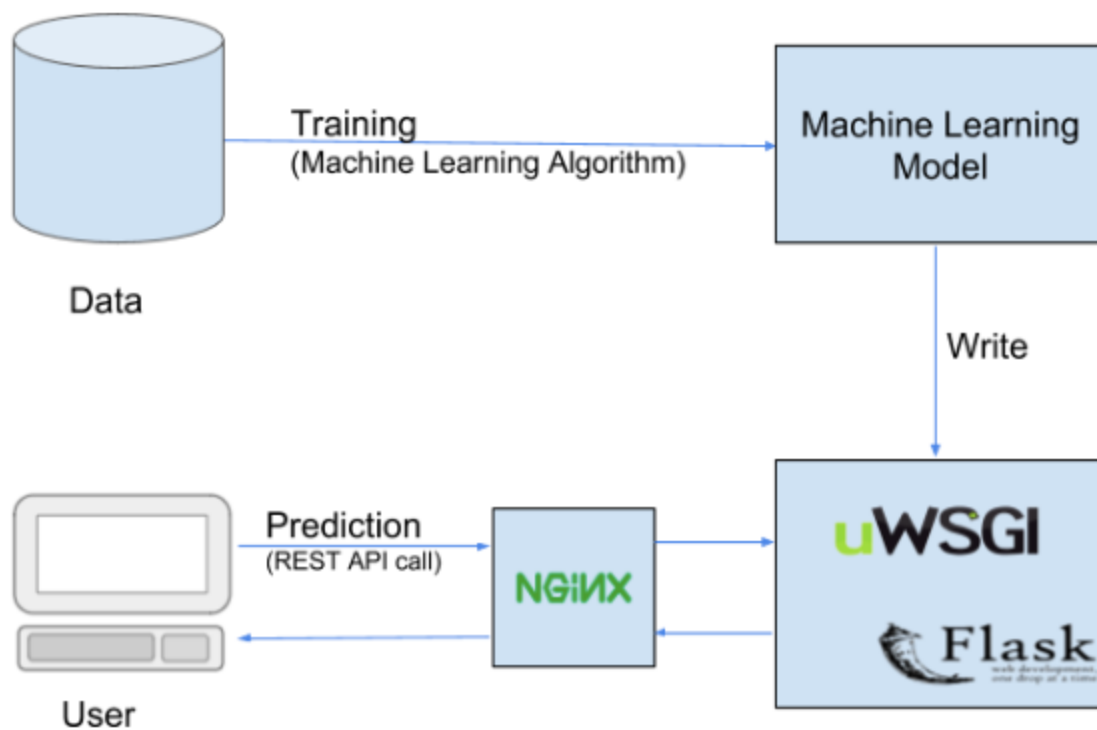


1- Deployment of Machine Learning Model

By: eng. Esraa Madhi

Why Deploy a Machine Learning Model?

Deploying allows the model to be **integrated into real-world applications** like websites or apps, where it can perform tasks like making predictions or recommendations based on new data.



How can we make that happen? **Using API**

<https://youtu.be/s7wmiS2mSXY>

<https://youtu.be/s7wmiS2mSXY>

Role of an API in Machine Learning

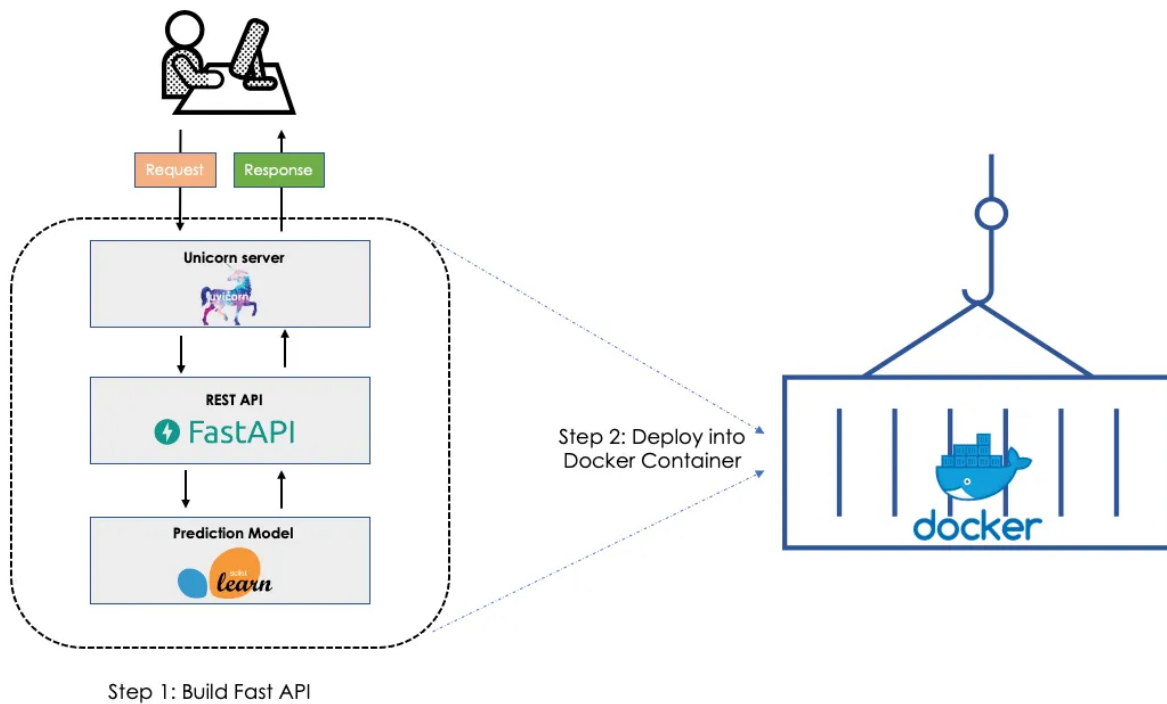
- **Abstraction:** An API lets users interact with the model by sending inputs and receiving outputs without needing to understand the underlying complexities.
- **Security:** Manages authentication, ensuring only authorized users can access the model.
- **Integration:** Enables easy integration of the machine learning model with other applications, allowing developers to incorporate model predictions into their software.
- **Consistency and Monitoring:** Ensures consistent model performance and allows for usage tracking and performance monitoring.

There are many different frameworks for building APIs in Python. Some of the most popular frameworks for creating APIs in Python are Django, Flask, and FastAPI.

What is FastAPI

FastAPI is a high-performing web framework for building APIs with Python.

FastAPI



Note:

Uvicorn is typically used as the server to run Python web applications that are built with an ASGI framework. For instance, if you develop an application using FastAPI, you can use Uvicorn to serve that application.

How to build API?

Let's create an API for our prediction model of used cars prices in usecase 4 in the next lesson

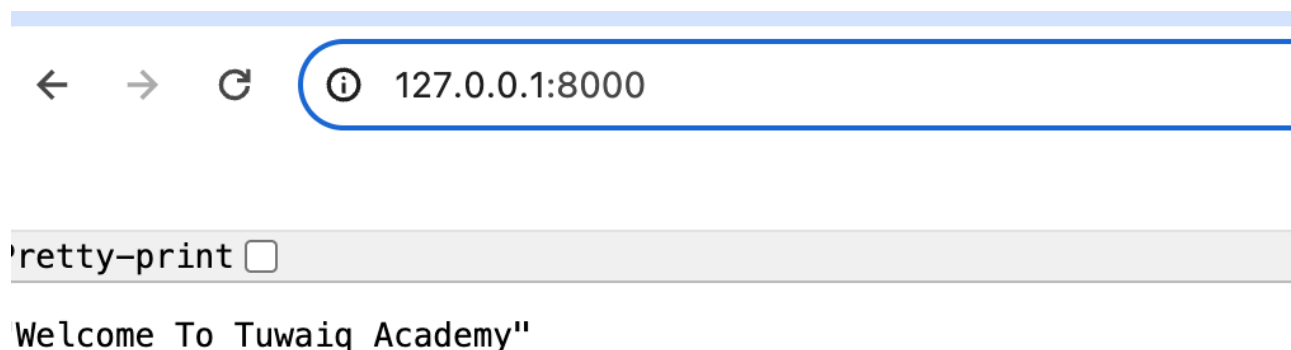
In steps:

Check '2- FastAPI Reference.pdf'

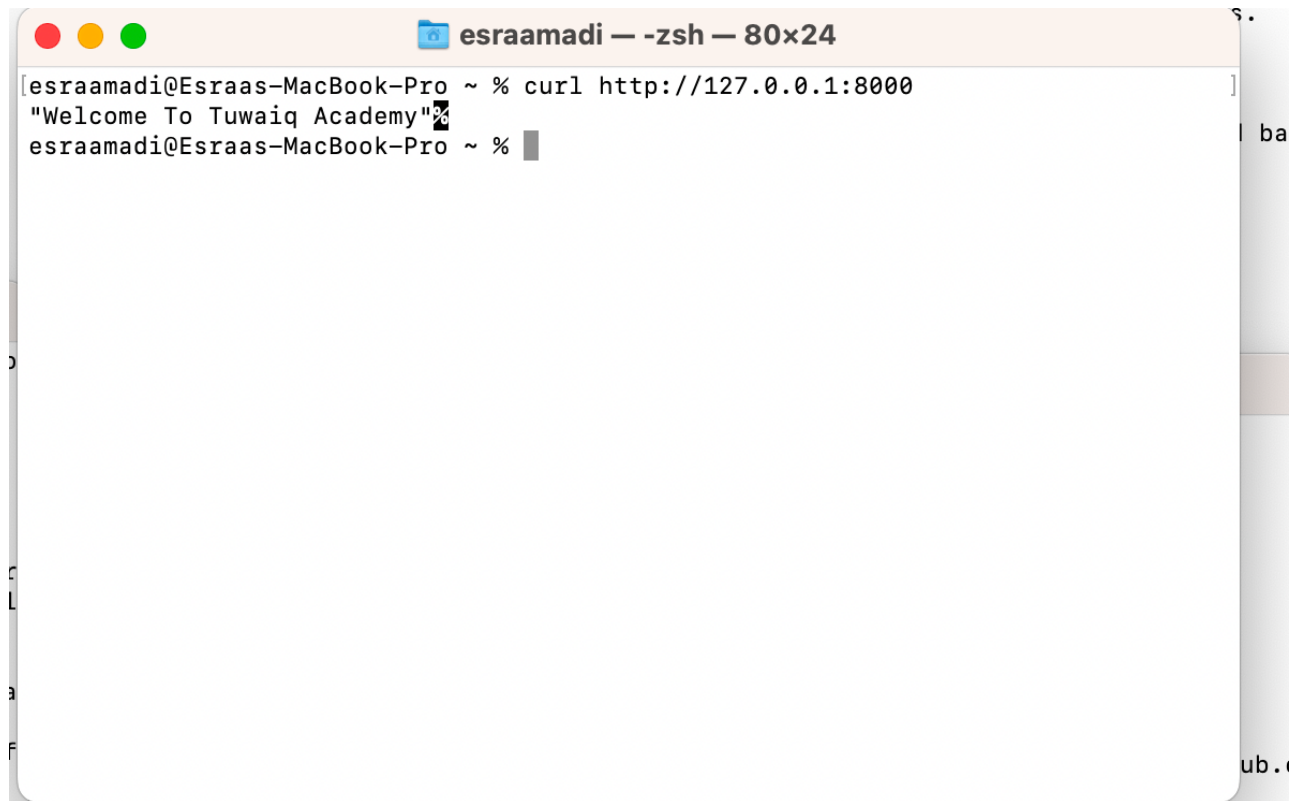
How to use it?

There are three common methods to send requests to an API endpoint:

- **Browser:** Quick and easy for GET requests.

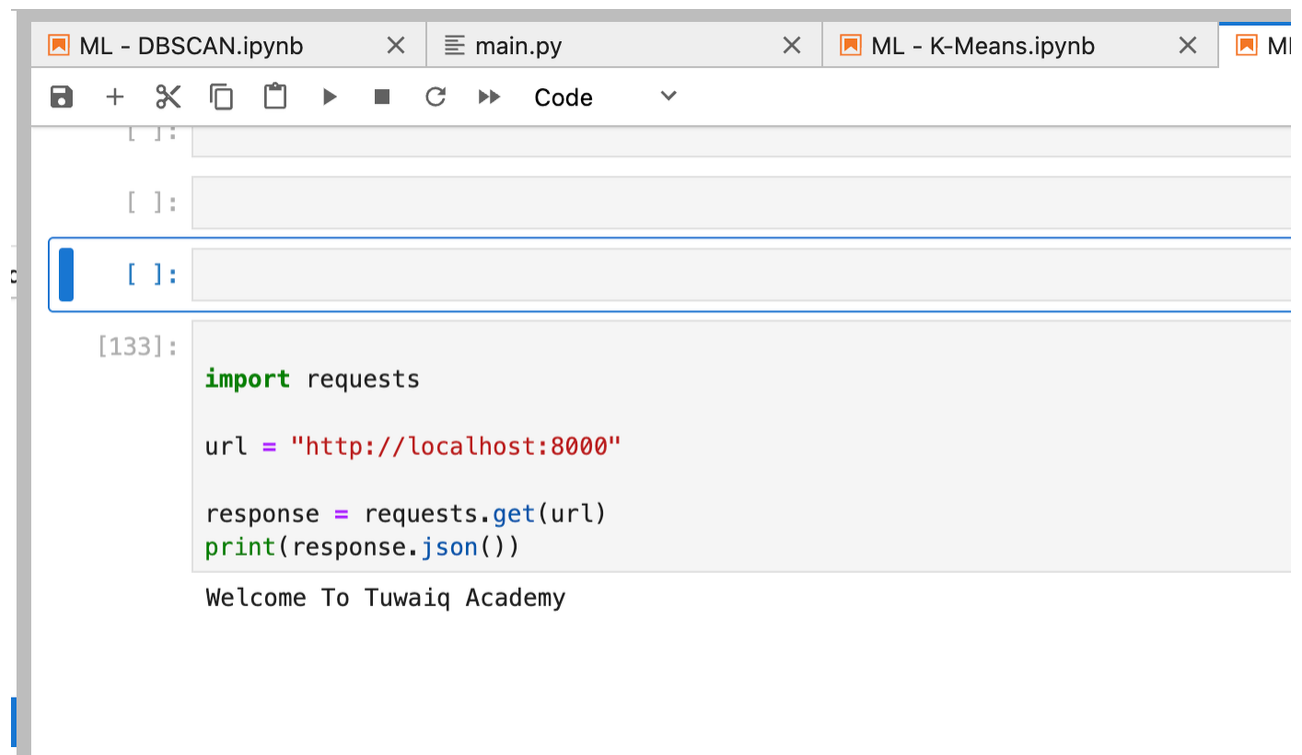


- **Curl:** Versatile for both GET and POST requests, useful for testing from command line.

A screenshot of a macOS terminal window titled "esraamadi — -zsh — 80x24". The terminal shows a user prompt "esraamadi@Esraas-MacBook-Pro ~ %" followed by the command "curl http://127.0.0.1:8000". The output of the command is "Welcome To Tuwaiq Academy" followed by a carriage return character. The prompt "esraamadi@Esraas-MacBook-Pro ~ %" is shown again on the next line.

```
esraamadi@Esraas-MacBook-Pro ~ % curl http://127.0.0.1:8000
"Welcome To Tuwaiq Academy"
esraamadi@Esraas-MacBook-Pro ~ %
```

- **Python `requests`:** Best for integrating HTTP requests into larger Python applications or scripts.



The screenshot shows a Jupyter Notebook with three tabs: 'ML - DBSCAN.ipynb', 'main.py', and 'ML - K-Means.ipynb'. The 'main.py' tab is active. The interface includes a toolbar with icons for saving, adding, deleting, copying, pasting, running, and code completion. Below the toolbar, there are three code cells. The first two are empty. The third cell, which is selected, contains the following Python code:

```
[133]:  
  
import requests  
  
url = "http://localhost:8000"  
  
response = requests.get(url)  
print(response.json())  
  
Welcome To Tuwaiq Academy
```

The output of the code cell is 'Welcome To Tuwaiq Academy'.

In steps:

Check '3- How to Request API?.pdf'