Supercharge Your AI with Microservices : A Fun-Filled Journey with Python and Flask

Nandana Sreeraj

Ecole Polytechnique

November 3, 2024

1. Flask is a lightweight, micro web framework written in Python.
2. It allows developers to build web applications quickly and with minimal code.

Key Features of Flask

1. Micro framework
2. Flexibility
3. Extensibility
4. Minimalistic

Feature	Flask	Django
Framework Type	Micro (Minimal)	Full-Stack
Learning Curve	Easy (Beginner-Friendly)	Steeper (More Components)
Flexibility	Highly Flexible	Predefined Structure
Best For	Small apps, Microservices	Large, Scalable Projects

Table: Comparison of Flask and Django Frameworks

When to Use Flask?

1. Microservices: Flask's small footprint and simplicity make it perfect for creating isolated, single-function services.
2. Prototyping: Ideal for quickly spinning up MVPs (Minimum Viable Products) or prototypes.
3. Small to Medium Apps: Great for personal projects or small to medium-sized applications.

Strengths of Flask

1. Microservices
2. Customizable
3. Scalable for Microservices
4. Active Community

Understanding Routing in Flask

1. Routing is the mechanism that maps URLs to specific functions in your web app.
2. Flask routes are defined using decorators (functions starting with @).
3. Each URL path is associated with a view function.
4. Routes handle both static and dynamic URLs.

Defining Routes with Decorators (Defining Routes with @app.route)

1. Flask uses @app.route() to define a route for a specific URL.
2. Syntax: @app.route('/path').

Dynamic URLs in Flask

1. Flask can handle dynamic URLs using path parameters.
2. Define placeholders in the URL using <variable> in @app.route.

What are HTTP Methods?

1. HTTP methods define the action to be performed on a specified resource (like a web page or an API endpoint).
2. They are part of the HTTP protocol, which is the foundation of data communication on the web.

GET Method

1. Purpose:
   The GET method requests data from a specified resource.
2. Use Cases:
   Fetching data to display on a web page (e.g., retrieving user profiles or a list of products).

POST Method

1. Purpose:
   The POST method submits data to be processed to a specified resource, often resulting in the creation of a new resource.
2. Use Cases:
   Submitting form data, uploading files, creating new user accounts, etc.

PUT Method

1. Purpose:
   The PUT method updates a resource or creates it if it doesn't exist.
2. Use Cases:
   Updating user information, modifying product details, etc.

DELETE Method

1. Purpose:
   The DELETE method removes a specified resource from the server.
2. Use Cases:
   Deleting user accounts, removing products from a catalog, etc.

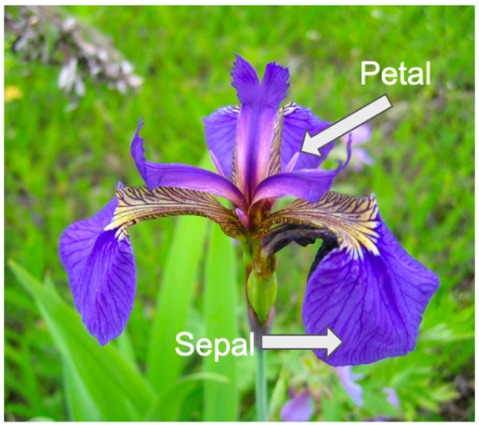
Introduction to AI Models as Microservices

1. Instead of embedding a machine learning model directly into a monolithic application, each model is deployed as an independent microservice that can be accessed over HTTP.
2. This makes scaling, updating, and maintaining individual services much easier.

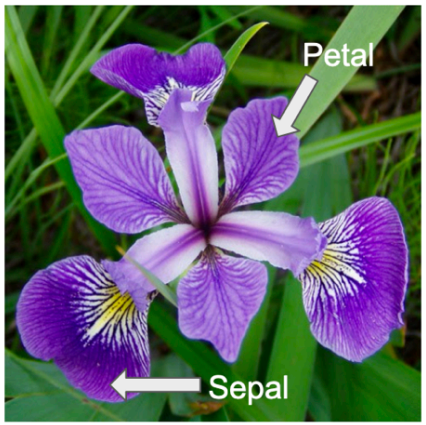
IRIS DATASET

1. The Iris dataset is a classic dataset in machine learning and statistics, often used for testing algorithms.
2. It contains data about three species of Iris flowers (Setosa, Versicolor, and Virginica) based on four features:
    1. Sepal length
    2. Sepal width
    3. Petal length
    4. Petal width

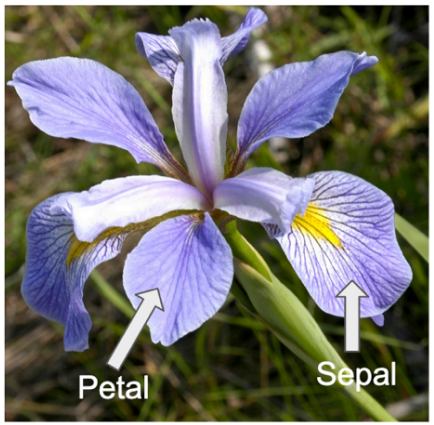
Iris setosa

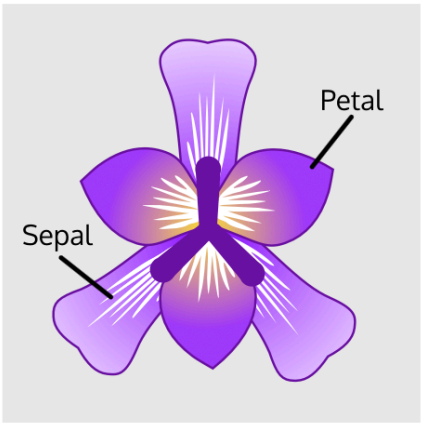


Iris versicolor

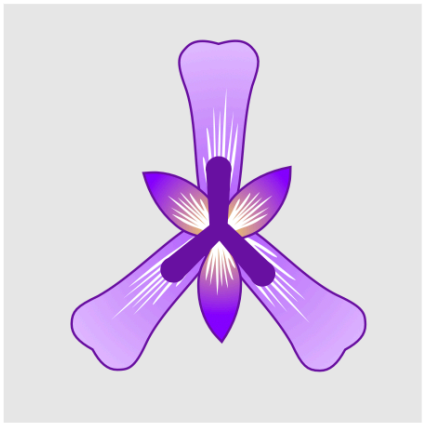


Iris virginica

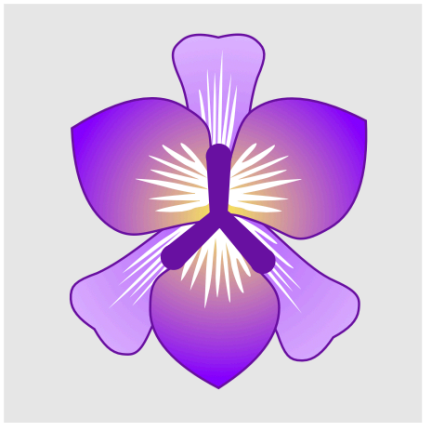




Iris Versicolor



Iris Setosa



Iris Virginica

KNN Classifier

- Choose the Value of k
- Calculate Distances
- Identify Nearest Neighbors
- Vote for Class Labels
- Assign Class Label
- Repeat for All Test Instances

KNN Classifier - Decision Matrix

- Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Manhattan Distance:

$$d(x, y) = \sum_{i=1}^{n}|x_i - y_i|$$

Example Table

Point (x, y)	Label
(1, 1)	A
(2, 1)	A
(4, 3)	B
(5, 4)	B
(3, 2)	? (Classified as A)

Step 1: Choose the Value of k

- Select the number of nearest neighbors, k.
- k defines how many data points will be used to determine the class of a new point.
- Example: Here we choose $k = 3$.

Step 2: Calculate Distances

- Calculate the distance between the test point and each training point.
- Common distance metric: Euclidean distance

$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Step 3: Identify Nearest Neighbors

- Find the k data points that are closest to the test point.
- Example: For $k = 3$, we select the three closest points.

Step 4: Vote for Class Labels

- Each of the k neighbors "votes" for its own class.
- The class with the majority of votes is chosen for the test point.

Step 5: Assign Class Label

- Based on the voting, assign the most frequent class label to the test point.
- Example: If the majority of the closest points are class A, classify the new point as class A.

CORS -Flask

CORS is a security feature in web browsers that controls whether web applications can request resources from a different origin (domain, protocol, or port).

Purpose of CORS

- **Security:** Protects users by preventing malicious websites from making unauthorized requests to a different domain.
- **Flexible Configuration:** Servers can define specific methods (GET, POST, etc.) and headers that are permitted in cross-origin requests.

Testing and Monitoring in Flask (What is Unit Testing?)

1. Unit Testing is a software testing method where individual components or functions of an application are tested in isolation to verify their correctness.
2. The goal is to ensure that each unit of the application behaves as expected, allowing for easier debugging and maintenance

Why Unit Testing is Important?

1. Catch Bugs Early
2. Refactoring Safety
3. Documentation

Unit Testing Frameworks in Flask

1. unittest
2. pytest

Scaling AI Microservices with Flask

1. Scalability requirements for AI microservices (e.g., handling high traffic, latency, efficient resource management).
2. The three techniques we'll cover:
   1. load balancing
   2. caching
   3. asynchronous task management

Load Balancing with Flask

1. Distribute traffic across multiple instances of your Flask app to handle increased loads.
2. Load Balancer Role: Routes requests across multiple servers to ensure no single server is overwhelmed.
3. Common Load Balancers: NGINX, HAProxy, AWS Elastic Load Balancer.

Caching with Redis or Memcached

1. Reduce response times and server load by caching frequently requested data or responses.
2. Stores data temporarily so that the same data doesn't need to be recalculated or retrieved from the database.
3. Tools: Redis and Memcached are popular choices

Asynchronous Task Processing with Celery

1. Offload long-running tasks (e.g., model training or batch predictions) from the main request cycle to prevent blocking.
2. Celery: A distributed task queue that allows asynchronous execution of tasks outside the request-response cycle.
3. Handling large, time-consuming jobs in the background.

THANK YOU