Documentation

Py-Measurement

It aims to estimate waist size based on input data. While it may not be entirely accurate initially, with consistent input of correct information over time, it becomes increasingly intelligent in predicting the correct waist size.

V1.0

System Requirements

CPU: 1
RAM: 2GB
Platform: Ubuntu 20.04, Docker

Download & Installation (Docker - Linux)

1. Open terminal and clone the project

   git clone https://github.com/agentcdp/py-measurements.git

2. Enter to project

   cd ./py-measurements

3. Change permission of “entrypoint.sh”

   sudo chmod +x entrypoint.sh

4. Build docker image and run

   docker-compose up -d

Getting started

If everything works fine, open http://localhost:8000/ to access the app.

Fill height, age & weight to see the waist size.

If you are not satisfied with given results, click on “Not your waist” to update your original waist.

Stacks

1. Python - Python is a versatile and widely-used programming language known for its simplicity, readability, and ease of use. It is an interpreted, high-level language with dynamic typing, making it suitable for a wide range of applications, from web development and scientific computing to data analysis, artificial intelligence, and more.
    - Django - Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.
    - ML Libraries - scikit-learn is a free software machine learning library for the Python programming language
    - Data Libraries – pandas is a software library written for the Python programming language for data manipulation and analysis.
2. HTML/CSS/JavaScript – Following are used for developing the frontend.
3. jQuery - jQuery is a JavaScript framework designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax.
4. Bootstrap - Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.
5. Animate.css - A cross-browser library of CSS animations. As easy to use as an easy thing.
6. Docker - Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

Production Deployment

1. Create local_settings.py file inside the pyApp folder.
2. Create variable DEBUG=True
3. Create variable ALLOWED_HOST=[‘you_host_id_or_domain’]
4. Serve static files as mentioned on Django documentation. https://docs.djangoproject.com/en/4.2/howto/static-files/
5. Setup database mysql or any based-on Django documentation.
6. Setup nginx, SSL and deploy.

MVC

Data Model: Measurement

- height
- weight
- age
- waist
- user_created
- created_at
- updated_at

home/views.py

- get_waist
  To get waist size
- update_waist
  To update waist size
- Index
  To serve home page

home/url.py

- / - index page
- /get_waist/?param
  API
- /update_waist
  API

ML Predictor - Linear regression

/home/utils.py

- WaistSizePredictor(data:list)

  Accept list of objects

  - *Example: payload = [{'height': object.height, 'weight': object.weight, 'age': object.age, 'waist': object.waist},]*

- __init__ :

  Initialize the class with data

- set_prediction_data():

  {'height': [new_height], 'weight': [new_weight], 'age': [new_age]}

- predict():

  provides predicted result

Scaling the application

1. To elevate the user experience, we can create supplementary front-end (FE) and API services using Angular, React, or Vue. Django REST framework, a versatile and robust toolkit, can be utilized for constructing efficient web APIs.

2. To ensure optimal concurrency and request handling, a load balancer can be created using tools such as Nginx and Gunicorn. These tools can effectively distribute incoming requests across multiple servers, facilitating efficient load balancing for the application.

3. In order to optimize database performance, a separate database instance can be created. This involves setting up a dedicated database server or using a separate database service to store and manage the application's data. Separating the database from other components of the application can enhance scalability, performance, and data management capabilities.

4. A worker service can be established to handle prediction tasks using Celery or any other queue management system. This worker service can efficiently handle time-consuming or resource-intensive prediction tasks in the background, asynchronously processing them without blocking the main application. Celery, a popular distributed task queue system, can be used to manage and distribute tasks across multiple workers, allowing for efficient processing of predictor tasks in the application.

5. To optimize database caching, a service similar to Elasticsearch can be set up. This involves implementing a caching mechanism that stores frequently accessed data in a fast and efficient manner, improving the responsiveness and performance of the application. Elasticsearch, or similar caching technologies, can be utilized to create a dedicated caching service that stores and retrieves database query results, reducing the need for repetitive database queries and enhancing overall application performance.

6. Depending on the specific requirements of the application, Kubernetes can be leveraged to scale both the application and workers. Kubernetes is a popular container orchestration platform that allows for efficient scaling of applications and services. By deploying application components as containers within Kubernetes clusters, the application can be automatically scaled up or down based on demand, ensuring optimal resource utilization and performance. This can include scaling the main application components as well as the worker services, ensuring that the application can handle increased traffic, workload, and processing requirements effectively.

Thank You ☺