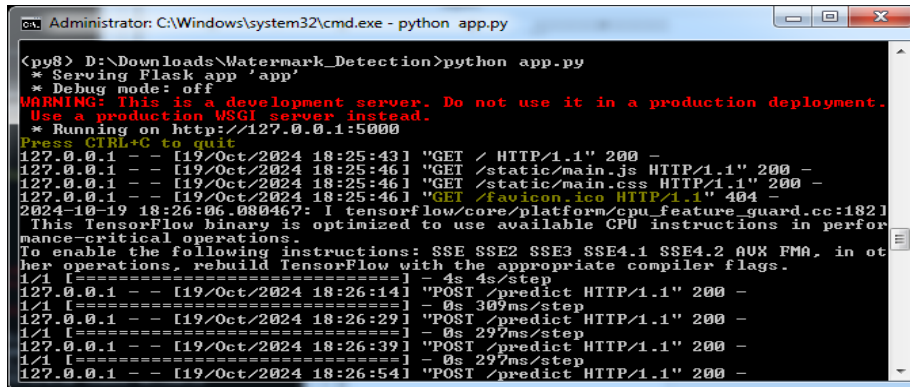


Documentation:

- How to running the code locally

Steps to run the model locally are:

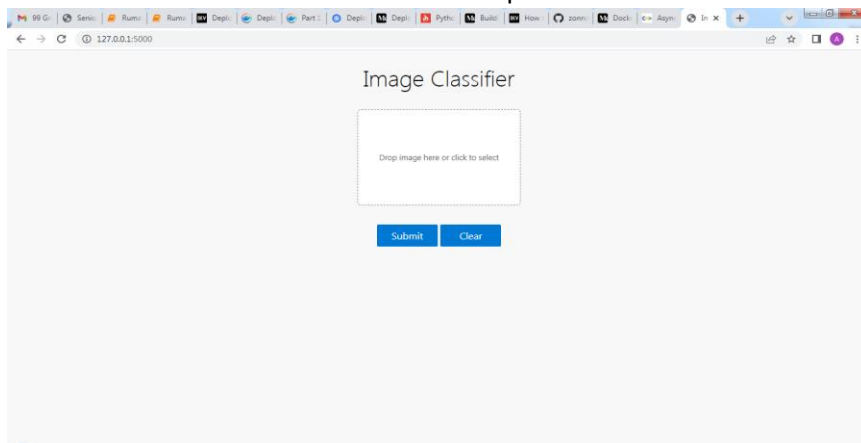
- Create Python environment contains all required libraries in "requirement.txt" file.
- Execute command "python app.py" as shown below



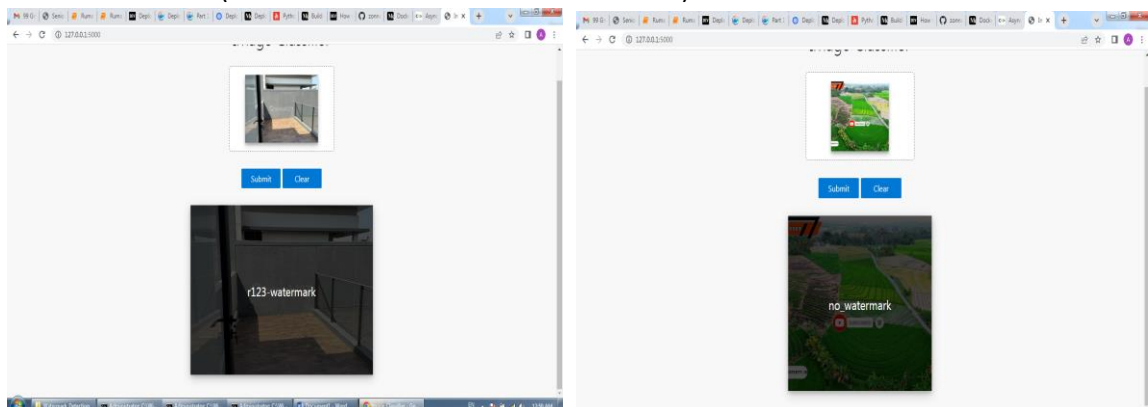
```
C:\Windows\system32\cmd.exe - python app.py

(py8) D:\Downloads\Watermark_Detection>python app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [19/Oct/2024 18:25:43] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Oct/2024 18:25:46] "GET /static/main.js HTTP/1.1" 200 -
127.0.0.1 - - [19/Oct/2024 18:25:46] "GET /static/main.css HTTP/1.1" 200 -
127.0.0.1 - - [19/Oct/2024 18:25:46] "GET /favicon.ico HTTP/1.1" 404 -
2024-10-19 18:26:06.080467: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in perfor-
mance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX FMA, in ot-
her operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [=====] - 4s 4s/step
127.0.0.1 - - [19/Oct/2024 18:26:14] "POST /predict HTTP/1.1" 200 -
1/1 [=====] - 0s 309ms/step
127.0.0.1 - - [19/Oct/2024 18:26:29] "POST /predict HTTP/1.1" 200 -
1/1 [=====] - 0s 297ms/step
127.0.0.1 - - [19/Oct/2024 18:26:39] "POST /predict HTTP/1.1" 200 -
1/1 [=====] - 0s 297ms/step
127.0.0.1 - - [19/Oct/2024 18:26:54] "POST /predict HTTP/1.1" 200 -
```

- For using the application, open site <http://127.0.0.1:5000/> as shown in the command prompt above in a browser. The result will be like picture below.



- Add or drop picture to box above and click button submit to see the prediction result. The result will be like below ("r123-watermark" or "no-watermark")



- Automating Machine Learning Pipelines with Apache Airflow

For automating machine learning to retrain the model, we decided to use Airflow with steps below:

1. Download the Docker Compose file for Airflow

```
curl -LfO 'https://airflow.apache.org/docs/apache-airflow/2.6.1/docker-compose.yaml'
```

2. Create directories for DAGs, logs, and plugins

3. Initialize the database

```
docker compose up airflow-init
```

4. Start the Airflow service

```
docker compose up
```

5. Once the containers are running, access the Airflow web interface by navigating to

<http://localhost:8080> in web browser. The default login credentials are:

Username: airflow

Password: airflow

6. Execute `"python app_airflow.py"` on command prompt with appropriate python environment.

- Deploy using Docker & Kubernetes

Steps to containerize the Machine Learning Models with Docker:

1. Install Docker: Download and install Docker from the official website.

2. Prepare a Dockerfile to define the environment and instructions for building the image. The dockerfile is already setup in the folder with name `"dockerfile.txt"` or `"my-ml-model"`.

3. Build the Docker Image by running the following command in the directory containing Dockerfile.

```
docker build -t my-ml-model
```

4. Run the Docker Container: Start a container from the image.

```
docker run -p 4000:80 my-ml-model
```

Deploying Containers with Kubernetes

1. Install Kubernetes: Set up a Kubernetes cluster using a service like Google Kubernetes Engine (GKE), Amazon EKS, or locally with Minikube.

2. Create a Deployment YAML

Define the Kubernetes deployment configuration is already saved as `"deployment.yaml"`.

3. Apply the Deployment: Deploy your container to the Kubernetes cluster.

```
kubectl apply -f deployment.yaml
```

4. Expose the Deployment: Create a service to expose the deployment

```
kubectl expose deployment my-ml-model-deployment --type=LoadBalancer --port=80 --target-port=80
```