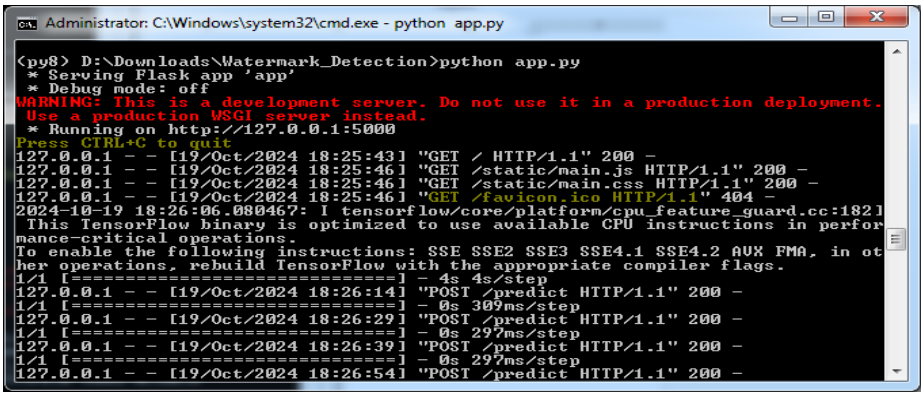
Documentation:

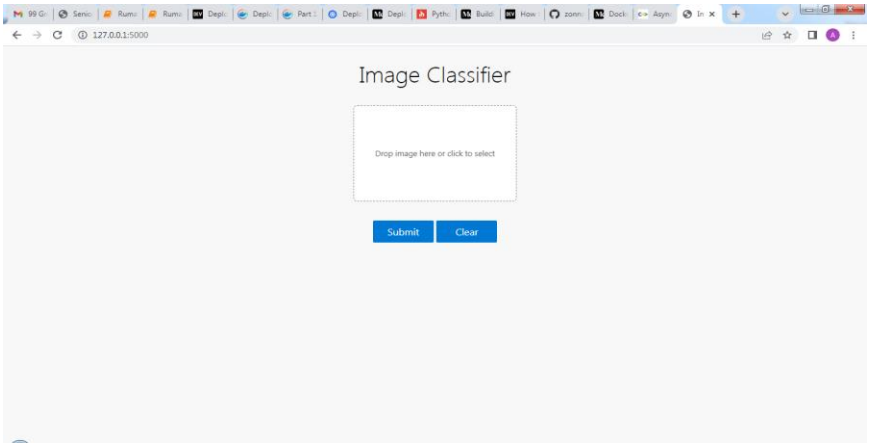
- How to running the code locally

  Steps to run the model locally are:

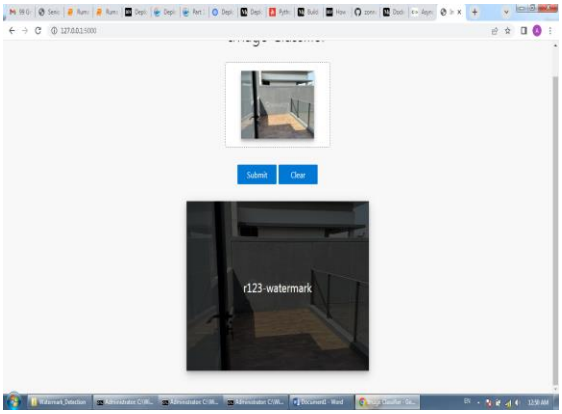
  1. Create Python environment contains all required libraries in “requirement.txt” file.
  2. Execute command “python app.py” as shown below

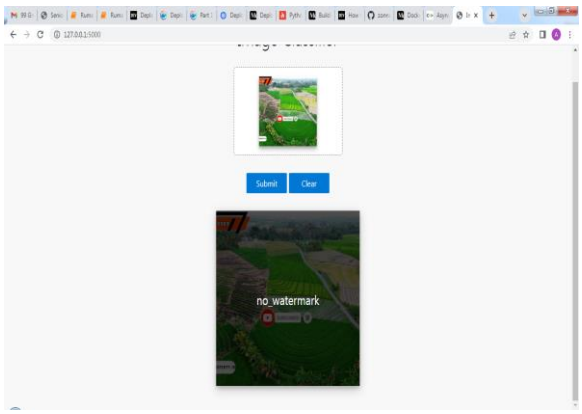


  3. For using the application, open site http://127.0.0.1:5000/ as shown in the command prompt above in a browser. The result will be like picture below.



  4. 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 & Kubernates
  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`