Day-3

Prerequisite:-

Steps

Download the .py file
Link
-https://scikit-learn.org/stable/_downloads/1a55101a8e49ab5d3213dadb31332045/plot_digits_classification.py

git clone "put ssh repo link" // for cloning//

cd "goto current directory" //for moving current directory //

mv ~/Downloads/plot_digits_classification.py //(move the file, Use when you are in the current directory)

nano requirements.txt //creating and opening the file//

Put all the required library

matplotlib
scikit-learn
pandas
numpy

pip install -r requirements.txt //install all the required library //

python3 plot_digits_classification.py //running the py file//

git add .

git commit -m "new changes"

git push

We will be covering today

GitHub Actions, hyperparameter finetuning for ml models

Github Actions

- Manually checking if any new update is working correctly is time-consuming and prone to errors.
- Alternative option: Automate the checking of any new updates based on a fixed set of instructions and tests.
- A CI/CD pipeline is a series of automated steps that streamline the software development process, integrating code changes and delivering them to production environments efficiently and reliably.
- It combines Continuous Integration (CI) and Continuous Delivery/Deployment (CD) to automate building, testing, and deploying software, reducing manual intervention and improving release frequency.

https://docs.github.com/en/actions/get-started/understanding-github-actions

- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.
- You can create **workflows** that build and test every pull request to your repository or deploy merged pull requests to production.
- You can configure a GitHub Actions workflow to be triggered when an **event** occurs in your repository, such as a pull request being opened or an issue being created.
- Your workflow contains one or more **jobs** that can run in sequential order or in parallel.
- Each job will run inside its own virtual machine **runner**, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.

Workflows

- A workflow is a configurable automated process that runs one or more **jobs**.
- Workflows are defined by a **YAML** file checked into your repository and will run when triggered by an **event** in your repository, or they can be triggered manually, or at a defined schedule.
- Workflows are defined in the **.github/workflows** directory in a repository.
- A repository can have multiple workflows, each of which can perform a different set of tasks, such as:
    - Building and testing pull requests
    - Deploying your application every time a release is created
    - Adding a label whenever a new issue is opened

Events

- An event is a specific activity in a repository that triggers a workflow run.
- For example, an activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository.
- You can also trigger a workflow to run on a schedule, by posting to a REST API, or manually.

Jobs

- A job is a set of steps in a workflow that is executed on the same runner.
- Each step is either a shell script that will be executed or an action that will be run.
- Steps are executed in order and are dependent on each other.
- Since each step is executed on the same runner, you can share data from one step to another.
- For example, you can have a step that builds your application, followed by a step that tests the application that was built.
- You can configure a job's dependencies with other jobs; by default, jobs have no dependencies and run in parallel.
- When a job takes a dependency on another job, it waits for the dependent job to complete before running.

Actions

- An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task.
- Use an action to help reduce the amount of repetitive code that you write in your workflow files.
- An action can pull your Git repository from GitHub, set up the correct toolchain for your build environment, or set up the authentication to your cloud provider.

Runners

- A runner is a server that runs your workflows when they're triggered. Each runner can run a single job at a time.
- GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows.
- Each workflow run executes in a fresh, newly-provisioned virtual machine.

Steps

- Create a new branch “actionbranch”
- Within the local github repository, create a new folder “.github”
    - Don’t miss the “.”
- Create a folder “workflows” inside this folder
- Create a new file with .yml extension inside workflows
- Paste the following into this file

```
# Step 1: Create and switch to a new branch named "actionbranch"
git checkout -b actionbranch
# (-b means "create new branch" and immediately switch to it)

# Step 2: Create the hidden .github directory and workflows folder inside it
mkdir -p .github/workflows
# (-p ensures parent folders are created if they don't exist)

# Step 3: Create a new empty YAML file for the GitHub Actions workflow
touch .github/workflows/testing.yml
# (touch simply makes a new file; here our workflow file is testing.yml)

# Step 4: Open the workflow file in nano editor to add YAML content
nano .github/workflows/testing.yml
# (nano is a terminal text editor; here we paste our workflow code)
```

YAML file link
ml_ops.ipynb

```
# Step 5: Stage all changes (new files and edits) for commit
git add .
# (. means "add everything in the current directory to staging area")

# Step 6: Commit the changes with a message
git commit -m "anything"
# (-m lets you write a short message describing the commit)

# Step 7: Push the new branch to the remote GitHub repository
git push -u origin actionbranch
# (-u sets this branch to track the remote branch, so future pushes are simpler)

#  Step 8: -Merge the branch with the main branch
pull
Merge
Confirm
```

===

YAML file link
https://colab.research.googl
e.com/drive/1FwIPHuItHdnL
jZ_a7ttX3Qe8PxWJRvpf?us
p=sharing

```
name: GitHub Actions Demo
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version:
    [3.13.5] steps:
      - run: echo "The job was automatically triggered by a ${{ github.event_name }} event."
      - run: echo "This job is now running on a ${{ runner.os }} server hosted by GitHub!"
      -    run: echo "The name of your branch is ${{ github.ref }} and your repository is
${{ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v3
      - run: echo "The ${{ github.repository }} repository has been cloned to the runner."
      - run: echo "The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: ls -r ${{ github.workspace
        }}
      - run: echo "This job's status is ${{ job.status }}."
      - name: Install dep
        run: pip3 install --no-cache-dir -r requirements.txt
      - name: Run experiment
        run: python plot_digits_classification.py
```
==

- Commit and Push to remote repository
- Merge the branch with the main branch

Hyperparameter

What are hyperparameters?

Hyperparameters → chosen by you before training.

Example: learning rate, number of epochs, batch size.

manually tunable parameters

Check for the best

combination

Task

- Create a new branch “hyperparam”
- Add optimal hyperparameter selection to your code for dev_size, gamma, C
- Push to branch
- Check the corresponding GitHub action process on github.com that gets triggered on push
- Merge branch with main

Hints:-
1. Create a new branch
2. Add optimal hyperparameter selection to plot_digits_classification.py
3. Stage and commit changes
4. Push to GitHub
5. Check GitHub Actions
6. Merge branch with main