Task 1: Train a Regression Model and Tune Hyperparameters

Github , LinkedIn, email : mannarmannan02@gmail.com

Assigned project Documentation:Boston_model-hparam tuning and CI/CD Implementation

Model selection: XG-Boost

Parallelism: Kubernetes Jobs (Hyper-parameter-tuning)

Introduction:

I implemented a model to predict house prices based on the provided features. And, I developed an end-to-end CI/CD pipeline for this model using GitHub Actions. Below is the file structure and contents for a detailed evaluation.

File 1: train.py

```
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

column_names = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX",
    "PTRATIO", "B", "LSTAT"
]

boston_df = pd.DataFrame(data, columns=column_names)
boston_df['MEDV'] = target

X = boston_df.drop('MEDV', axis=1)
y = boston_df['MEDV']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Arguments to be passed from Kubernetes into model container
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--n_estimators", type=int, required=True)
parser.add_argument("--max_depth", type=int, required=True)
parser.add_argument("--learning_rate", type=float, required=True)
parser.add_argument("--subsample", type=float, required=True)
args = parser.parse_args()

model = XGBRegressor(
    n_estimators=args.n_estimators,
    max_depth=args.max_depth,
    learning_rate=args.learning_rate,
    subsample=args.subsample
)
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"MSE: {mse}")

model_filename = f"/data/model_{mse}.joblib"
metrics_filename = f"/data/metrics_{mse}.txt"

joblib.dump(model, model_filename)
with open(metrics_filename, "w") as f:
    f.write(str(mse)+"\n")
```

Code Description:

1. **Arguments:**
   The script takes the following hyperparameters as arguments, which are passed via Kubernetes:
   - **n_estimators   * max_depth   *learning_rate   * subsample.**
2. **Training and Splitting Data:**
   The Boston housing dataset is split into **80% training and 20% testing**.
3. **Model and MSE Scores:**
   The trained model is saved as **model_{mse}.joblib**. The mean squared error (MSE) is saved as **metrics_{mse}.txt** for further analysis.

File 2: Docker File (Dockerfile.train)

```
FROM python:3.8-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY train.py .
ENTRYPOINT ["python", "train.py"]
```

File 3: Requirements.txt

```
pandas==2.0.3
scikit-learn==1.3.0
xgboost==1.7.6
joblib==1.3.2
flask==2.0.1
numpy==1.21.2
```

File 4: Kubernetes Job Template (job-hyperparameter.yaml)

```
apiVersion: batch/v1
kind: Job
metadata:
  name: trial-{{N_ESTIMATORS}}-{{MAX_DEPTH}}-{{LEARNING_RATE}}-{{SUBSAMPLE}}
  labels:
```

```yaml
    app: xgb-hyperparameter-tuning
    trial: "true"
spec:
  template:
    metadata:
      labels:
        app: trial
        n_estimators: "{{N_ESTIMATORS}}"
        max_depth: "{{MAX_DEPTH}}"
        learning_rate: "{{LEARNING_RATE}}"
        subsample: "{{SUBSAMPLE}}"
    spec:
      containers:
      - name: xgb-trainer
        image: mannarn/model-train:latest
        args:
        - "--n_estimators={{N_ESTIMATORS}}"
        - "--max_depth={{MAX_DEPTH}}"
        - "--learning_rate={{LEARNING_RATE_RAW}}"
        - "--subsample={{SUBSAMPLE_RAW}}"
        volumeMounts:
        - name: data-volume
          mountPath: /data
      restartPolicy: Never
      volumes:
      - name: data-volume
        hostPath:
          path: /data
          type: DirectoryOrCreate
```

Code Description:

1. **Kind**
   The Job resource in Kubernetes ensures that a pod runs to completion.
2. **Metadata (Job Name and Labels):**
   The job name follows a structured naming convention using hyperparameters:
   trial-{{N_ESTIMATORS}}-{{MAX_DEPTH}}-{{LEARNING_RATE}}-{{SUBSAMPLE}}
   Example: trial-200-5-0-1-0.6
3. **Pod Metadata:**
   Hyperparameters (**n_estimators, max_depth, learning_rate, subsample**) are assigned as pod labels for easy identification.
4. **Passing Hyperparameters as Arguments:**
   These arguments are passed to the container when it starts, allowing the container to configure the XGBoost model dynamically.

File 5: Script to Launch Trials (generate-jobs.ps1)

```powershell
$n_estimators_list = @(100, 200)
$max_depth_list = @(5, 7)
$learning_rate_list = @(0.01, 0.1)
$subsample_list = @(0.6, 0.8)
```

```
# All combination of h-parameters
foreach ($n in $n_estimators_list) {
    foreach ($depth in $max_depth_list) {
        foreach ($lr in $learning_rate_list) {
            foreach ($sub in $subsample_list) {
                # Sanitize values with dots (e.g., 0.01 → 0-01 for Kubernetes naming)
                $sanitized_lr = "$lr".Replace(".", "-")
                $sanitized_sub = "$sub".Replace(".", "-")

                $yamlContent = (Get-Content job-hyperparameter.yaml -Raw) `
                    -replace '\{\{N_ESTIMATORS\}\}', $n `
                    -replace '\{\{MAX_DEPTH\}\}', $depth `
                    -replace '\{\{LEARNING_RATE\}\}', $sanitized_lr `
                    -replace '\{\{SUBSAMPLE\}\}', $sanitized_sub `
                    -replace '\{\{LEARNING_RATE_RAW\}\}', $lr `
                    -replace '\{\{SUBSAMPLE_RAW\}\}', $sub

                Write-Output "Submitting job: trial-$n-$depth-$sanitized_lr-$sanitized_sub"
                $yamlContent | kubectl apply -f -
            }
        }
    }
}
```

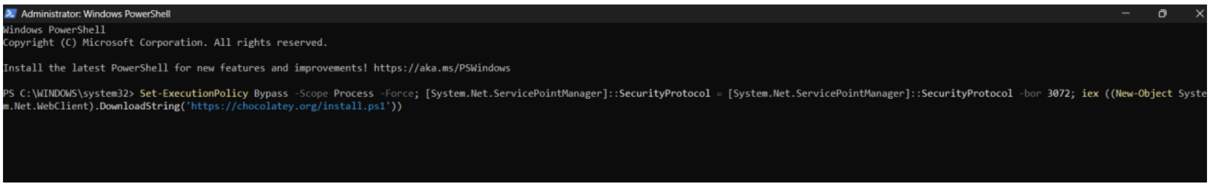
Code Description:

1. **Automated Job Submission:**
   This PowerShell script generates Kubernetes job configurations for all possible hyperparameter combinations.
2. **Sanitizing Values for Kubernetes Naming:**
   Since Kubernetes does not allow dots (.) in resource names, we replace them with hyphens (-):
   $sanitized_lr = "$lr".Replace(".", "-")
   $sanitized_sub = "$sub".Replace(".", "-")
   This ensures compliance with **DNS-1123 Kubernetes naming rules**File_4: Dockerfile.train

STEPS TO RUN Hyperparameter tuning in Local machine

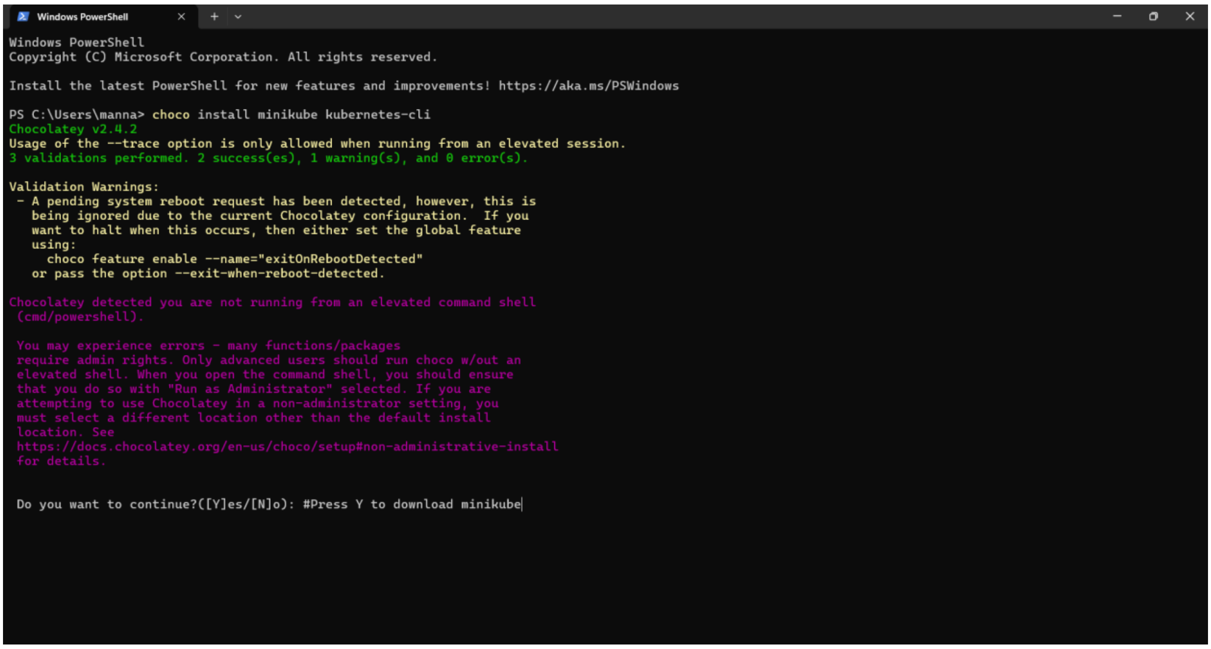
#Install dependencies:

1.Chocoley or Winget to download Mini Kube:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object Syste
m.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

2. Install Mini Kube and Kubernetes CLI tools to create a local development cluster.

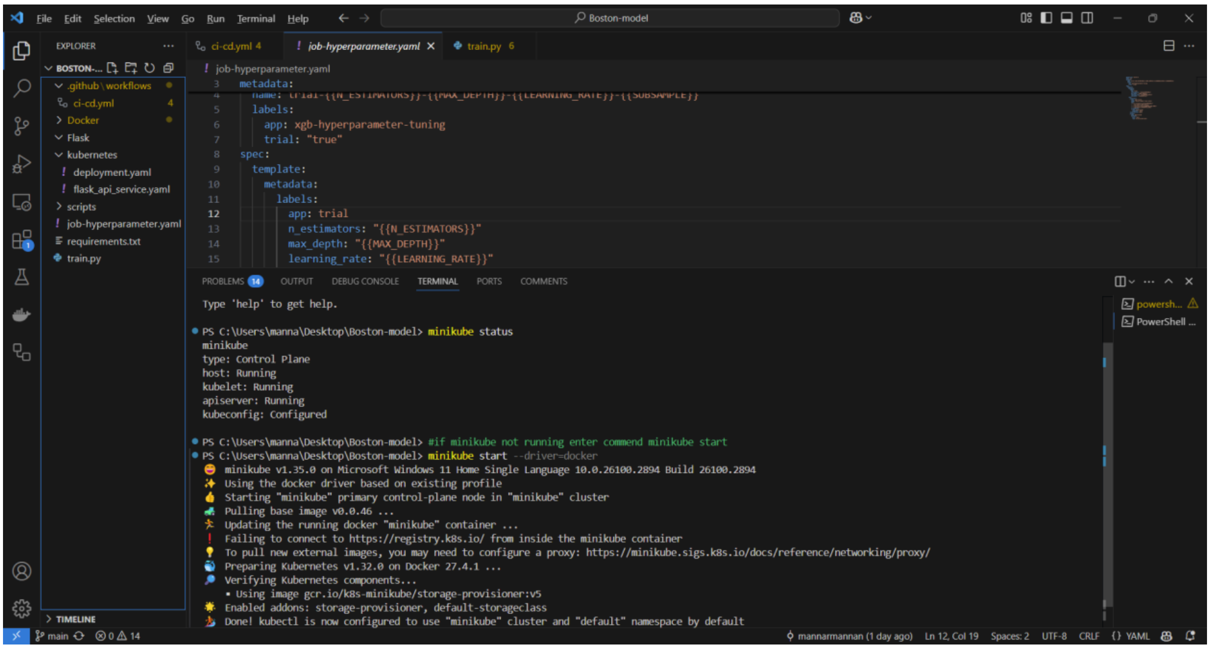


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\manna> choco install minikube kubernetes-cli
Chocolatey v2.4.2
Usage of the --trace option is only allowed when running from an elevated session.
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
 - A pending system reboot request has been detected, however, this is
   being ignored due to the current Chocolatey configuration.  If you
   want to halt when this occurs, then either set the global feature
   using:
     choco feature enable --name="exitOnRebootDetected"
   or pass the option --exit-when-reboot-detected.

Chocolatey detected you are not running from an elevated command shell
 (cmd/powershell).

 You may experience errors - many functions/packages
 require admin rights. Only advanced users should run choco w/out an
 elevated shell. When you open the command shell, you should ensure
 that you do so with "Run as Administrator" selected. If you are
 attempting to use Chocolatey in a non-administrator setting, you
 must select a different location other than the default install
 location. See
 https://docs.chocolatey.org/en-us/choco/setup#non-administrative-install
 for details.


 Do you want to continue?([Y]es/[N]o): #Press Y to download minikube
```

3. Start Mini Kube Cluster

>>minikube start --driver=docker

Initializes a local Kubernetes cluster using Docker driver for container orchestration.



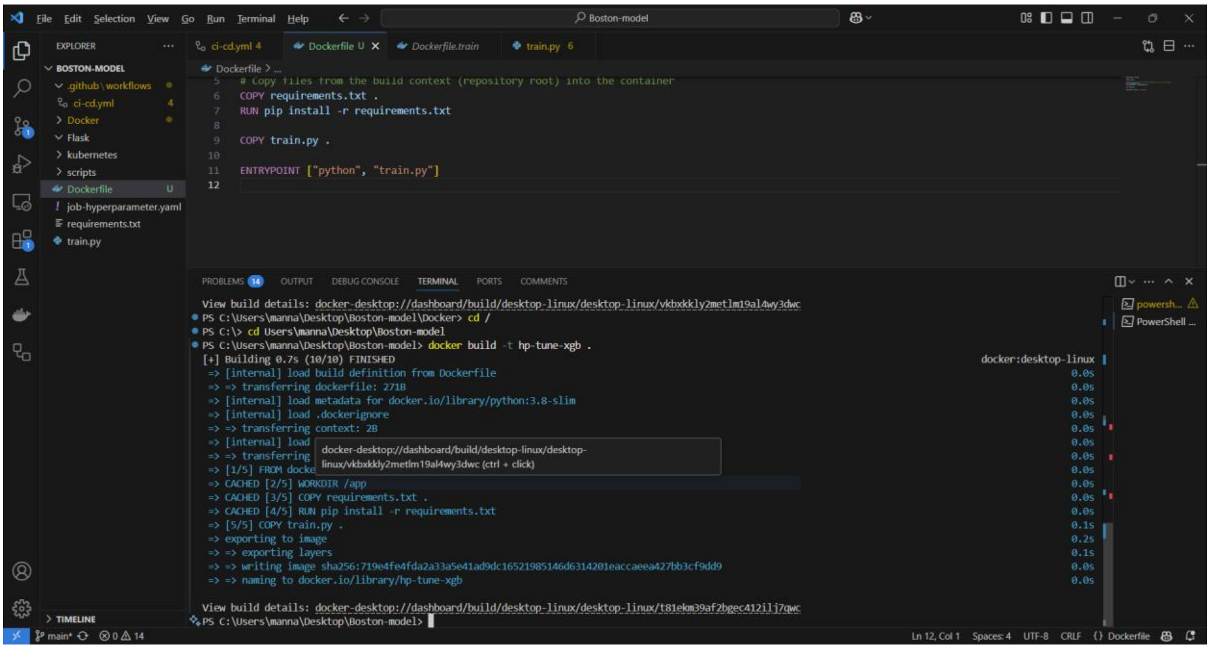
```yaml
metadata:
  name: trial-{{N_ESTIMATORS}}-{{MAX_DEPTH}}-{{LEARNING_RATE}}-{{SUBSAMPLE}}
  labels:
    app: xgb-hyperparameter-tuning
    trial: "true"
spec:
  template:
    metadata:
      labels:
        app: trial
        n_estimators: "{{N_ESTIMATORS}}"
        max_depth: "{{MAX_DEPTH}}"
        learning_rate: "{{LEARNING_RATE}}"
```

```
Type 'help' to get help.

PS C:\Users\manna\Desktop\Boston-model> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Users\manna\Desktop\Boston-model> #if minikube not running enter commend minikube start
PS C:\Users\manna\Desktop\Boston-model> minikube start --driver=docker
😄  minikube v1.35.0 on Microsoft Windows 11 Home Single Language 10.0.26100.2894 Build 26100.2894
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🏃  Updating the running docker "minikube" container ...
❗  Failing to connect to https://registry.k8s.io/ from inside the minikube container
💡  To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

4 Build Training Image

>>docker build –t hp-xgboost .

Creates Docker image with training environment and XGBoost dependencies.



```
# Copy files from the build context (repository root) into the container
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY train.py .

ENTRYPOINT ["python", "train.py"]

```

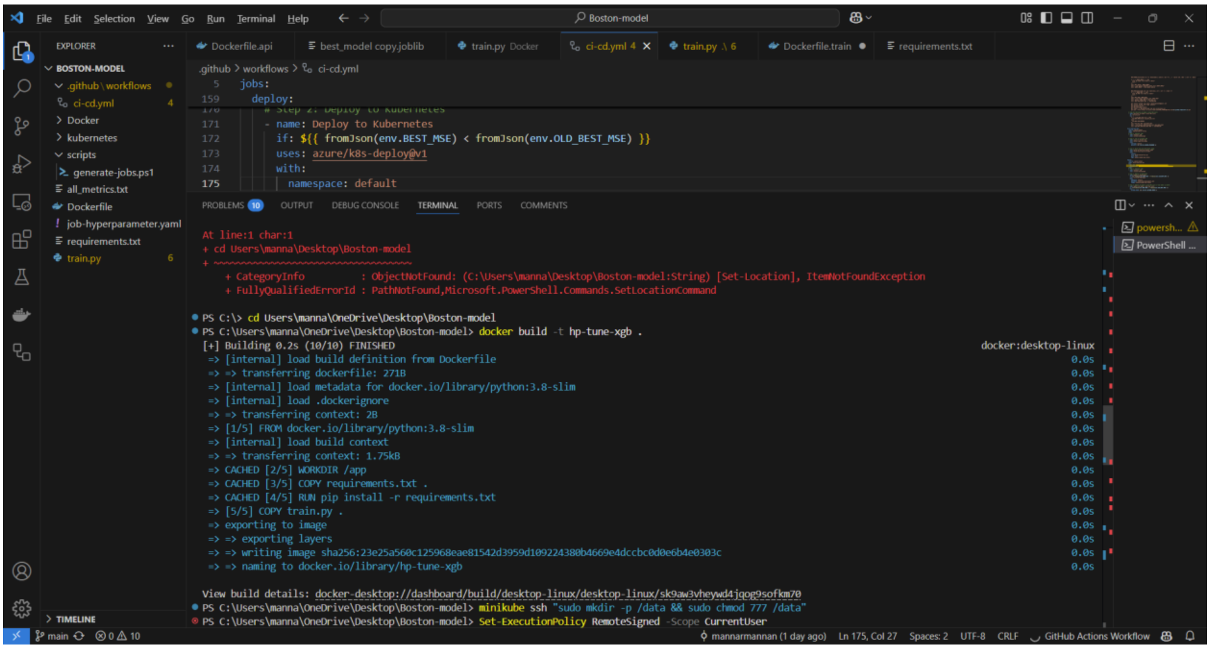
```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/vkbxkkly2metlm19al4wy3dwc
PS C:\Users\manna\Desktop\Boston-model\Docker> cd /
PS C:\> cd Users\manna\Desktop\Boston-model
PS C:\Users\manna\Desktop\Boston-model> docker build -t hp-tune-xgb .
[+] Building 0.7s (10/10) FINISHED                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 271B                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                          0.0s
 => [internal] load .dockerignore                                                           0.0s
 => => transferring context: 2B                                                             0.0s
 => [internal] load                                                                         0.0s
 => => transferring                                                                         0.0s
 => [1/5] FROM docke                                                                        0.0s
 => CACHED [2/5] WORKDIR /app                                                               0.0s
 => CACHED [3/5] COPY requirements.txt .                                                    0.0s
 => CACHED [4/5] RUN pip install -r requirements.txt                                        0.0s
 => [5/5] COPY train.py .                                                                   0.1s
 => exporting to image                                                                      0.2s
 => => exporting layers                                                                     0.1s
 => => writing image sha256:719e4fe4fda2a33a5e41ad9dc16521985146d6314201eaccaeea427bb3cf9dd9   0.0s
 => => naming to docker.io/library/hp-tune-xgb                                              0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/t81ekm39af2bgec412ilj7qwc
PS C:\Users\manna\Desktop\Boston-model>
```

docker-desktop://dashboard/build/desktop-linux/desktop-linux/vkbxkkly2metlm19al4wy3dwc (ctrl + click)

5. Create Shared Volume

>>minikube ssh "sudo mkdir -p /data && sudo chmod 777 /data"

Sets up persistent storage in Minikube VM for model artifacts and metrics.



```
jobs:
  deploy:
    # Step 2: Deploy to Kubernetes
    - name: Deploy to Kubernetes
      if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
      uses: azure/k8s-deploy@v1
      with:
        namespace: default
```

```
At line:1 char:1
+ cd Users\manna\Desktop\Boston-model
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Users\manna\Desktop\Boston-model:String) [Set-Location], ItemNotFoundException
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\> cd Users\manna\OneDrive\Desktop\Boston-model
PS C:\Users\manna\OneDrive\Desktop\Boston-model> docker build -t hp-tune-xgb .
[+] Building 0.2s (10/10) FINISHED                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 271B                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                          0.0s
 => [internal] load .dockerignore                                                           0.0s
 => => transferring context: 2B                                                             0.0s
 => [1/5] FROM docker.io/library/python:3.8-slim                                            0.0s
 => [internal] load build context                                                           0.0s
 => => transferring context: 1.75kB                                                         0.0s
 => CACHED [2/5] WORKDIR /app                                                               0.0s
 => CACHED [3/5] COPY requirements.txt .                                                    0.0s
 => CACHED [4/5] RUN pip install -r requirements.txt                                        0.0s
 => [5/5] COPY train.py .                                                                   0.0s
 => exporting to image                                                                      0.0s
 => => exporting layers                                                                     0.0s
 => => writing image sha256:23e25a560c125968eae81542d3959d109224380b4669e4dccbc0d0e6b4e0303c   0.0s
 => => naming to docker.io/library/hp-tune-xgb                                              0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/sk9aw3vheywd4jqog9sofkm70
PS C:\Users\manna\OneDrive\Desktop\Boston-model> minikube ssh "sudo mkdir -p /data && sudo chmod 777 /data"
PS C:\Users\manna\OneDrive\Desktop\Boston-model> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

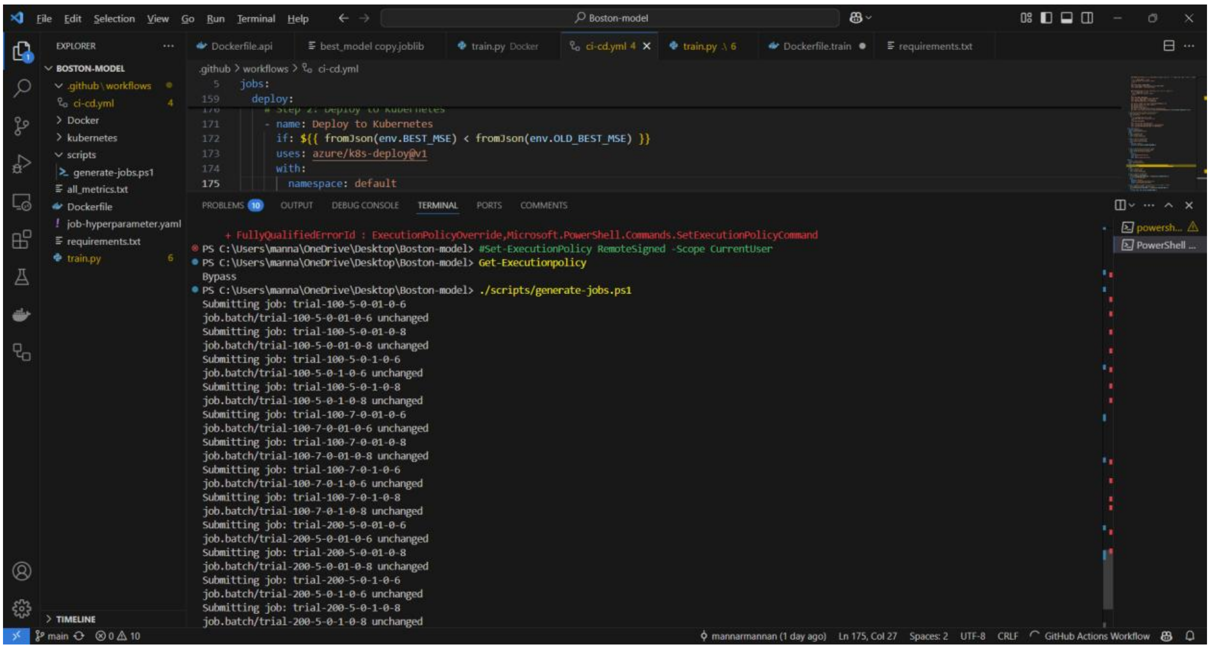
5. Enable PowerShell Script Execution & Launch Parallel Training Jobs

>>Set-ExecutionPolicy Bypass -Scope Process

Temporarily allows PowerShell script execution for hyperparameter job generation.

>>./scripts/generate-jobs.ps1

Submits 16 Kubernetes jobs with different hyperparameter combinations for distributed tuning.



```
jobs:
  deploy:
      # Step 2: Deploy to Kubernetes
      - name: Deploy to Kubernetes
        if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
        uses: azure/k8s-deploy@v1
        with:
          namespace: default
```

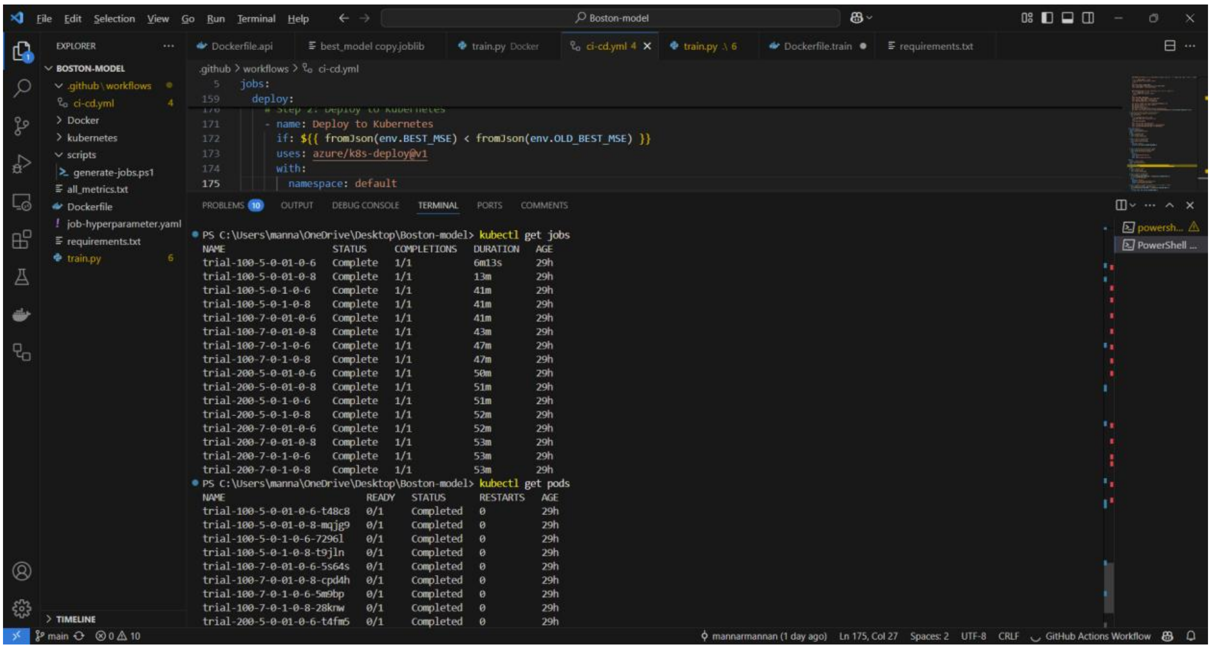
```
    + FullyQualifiedErrorId : ExecutionPolicyOverride,Microsoft.PowerShell.Commands.SetExecutionPolicyCommand
PS C:\Users\manna\OneDrive\Desktop\Boston-model> #Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
PS C:\Users\manna\OneDrive\Desktop\Boston-model> Get-Executionpolicy
Bypass
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ./scripts/generate-jobs.ps1
Submitting job: trial-100-5-0-01-0-6
job.batch/trial-100-5-0-01-0-6 unchanged
Submitting job: trial-100-5-0-01-0-8
job.batch/trial-100-5-0-01-0-8 unchanged
Submitting job: trial-100-5-0-1-0-6
job.batch/trial-100-5-0-1-0-6 unchanged
Submitting job: trial-100-5-0-1-0-8
job.batch/trial-100-5-0-1-0-8 unchanged
Submitting job: trial-100-7-0-01-0-6
job.batch/trial-100-7-0-01-0-6 unchanged
Submitting job: trial-100-7-0-01-0-8
job.batch/trial-100-7-0-01-0-8 unchanged
Submitting job: trial-100-7-0-1-0-6
job.batch/trial-100-7-0-1-0-6 unchanged
Submitting job: trial-100-7-0-1-0-8
job.batch/trial-100-7-0-1-0-8 unchanged
Submitting job: trial-200-5-0-01-0-6
job.batch/trial-200-5-0-01-0-6 unchanged
Submitting job: trial-200-5-0-01-0-8
job.batch/trial-200-5-0-01-0-8 unchanged
Submitting job: trial-200-5-0-1-0-6
job.batch/trial-200-5-0-1-0-6 unchanged
Submitting job: trial-200-5-0-1-0-8
job.batch/trial-200-5-0-1-0-8 unchanged
```

6. Monitor Job Progress

>>kubectl get jobs

>>Kuberctl get pods

Tracks job completion status in real-time through Kubernetes watch mode.



```
jobs:
  deploy:
      # Step 2: Deploy to Kubernetes
      - name: Deploy to Kubernetes
        if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
        uses: azure/k8s-deploy@v1
        with:
          namespace: default
```

```
PS C:\Users\manna\OneDrive\Desktop\Boston-model> kubectl get jobs
NAME                   STATUS     COMPLETIONS   DURATION   AGE
trial-100-5-0-01-0-6   Complete   1/1           6m13s      29h
trial-100-5-0-01-0-8   Complete   1/1           13m        29h
trial-100-5-0-1-0-6    Complete   1/1           41m        29h
trial-100-5-0-1-0-8    Complete   1/1           41m        29h
trial-100-7-0-01-0-6   Complete   1/1           41m        29h
trial-100-7-0-01-0-8   Complete   1/1           43m        29h
trial-100-7-0-1-0-6    Complete   1/1           47m        29h
trial-100-7-0-1-0-8    Complete   1/1           47m        29h
trial-200-5-0-01-0-6   Complete   1/1           50m        29h
trial-200-5-0-01-0-8   Complete   1/1           51m        29h
trial-200-5-0-1-0-6    Complete   1/1           51m        29h
trial-200-5-0-1-0-8    Complete   1/1           52m        29h
trial-200-7-0-01-0-6   Complete   1/1           52m        29h
trial-200-7-0-01-0-8   Complete   1/1           53m        29h
trial-200-7-0-1-0-6    Complete   1/1           53m        29h
trial-200-7-0-1-0-8    Complete   1/1           53m        29h
PS C:\Users\manna\OneDrive\Desktop\Boston-model> kubectl get pods
NAME                         READY   STATUS      RESTARTS   AGE
trial-100-5-0-01-0-6-t48c8   0/1     Completed   0          29h
trial-100-5-0-01-0-8-mqjg9   0/1     Completed   0          29h
trial-100-5-0-1-0-6-7296l    0/1     Completed   0          29h
trial-100-5-0-1-0-8-t9jln    0/1     Completed   0          29h
trial-100-7-0-01-0-6-5s64s   0/1     Completed   0          29h
trial-100-7-0-01-0-8-cpd4h   0/1     Completed   0          29h
trial-100-7-0-1-0-6-5m9bp    0/1     Completed   0          29h
trial-100-7-0-1-0-8-28knw    0/1     Completed   0          29h
trial-200-5-0-01-0-6-t4fm5   0/1     Completed   0          29h
```

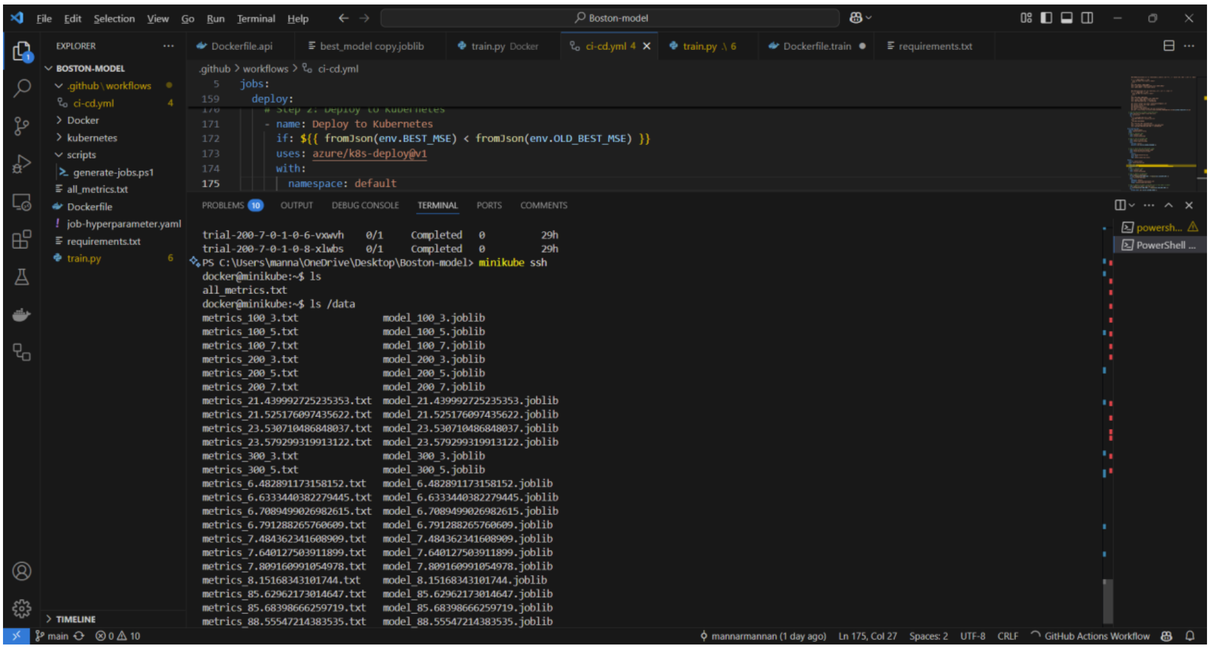
7.Login into shared memory of Virtual machine

>>Minikibe ssh

Inspect Training Results

>>minikube ssh "ls -lh /data"

Verifies artifact creation in shared volume with model files and metrics.



```
jobs:
  deploy:
      # Step 2: Deploy to Kubernetes
      - name: Deploy to Kubernetes
        if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
        uses: azure/k8s-deploy@v1
        with:
          namespace: default
```

```
trial-200-7-0-1-0-6-vxwvh    0/1     Completed   0          29h
trial-200-7-0-1-0-8-xlwbs    0/1     Completed   0          29h
PS C:\Users\manna\OneDrive\Desktop\Boston-model> minikube ssh
docker@minikube:~$ ls
all_metrics.txt
docker@minikube:~$ ls /data
metrics_100_3.txt              model_100_3.joblib
metrics_100_5.txt              model_100_5.joblib
metrics_100_7.txt              model_100_7.joblib
metrics_200_3.txt              model_200_3.joblib
metrics_200_5.txt              model_200_5.joblib
metrics_200_7.txt              model_200_7.joblib
metrics_21.439992725235353.txt  model_21.439992725235353.joblib
metrics_21.525176097435622.txt  model_21.525176097435622.joblib
metrics_23.530710486848037.txt  model_23.530710486848037.joblib
metrics_23.579299319913122.txt  model_23.579299319913122.joblib
metrics_300_3.txt              model_300_3.joblib
metrics_300_5.txt              model_300_5.joblib
metrics_6.482891173158152.txt   model_6.482891173158152.joblib
metrics_6.6333440382279445.txt  model_6.6333440382279445.joblib
metrics_6.7089499026982615.txt  model_6.7089499026982615.joblib
metrics_6.791288265760609.txt   model_6.791288265760609.joblib
metrics_7.484362341608909.txt   model_7.484362341608909.joblib
metrics_7.640127503911899.txt   model_7.640127503911899.joblib
metrics_7.809160991054978.txt   model_7.809160991054978.joblib
metrics_8.15168343101744.txt    model_8.15168343101744.joblib
metrics_85.62962173014647.txt   model_85.62962173014647.joblib
metrics_85.68398666259719.txt   model_85.68398666259719.joblib
metrics_88.55547214383535.txt   model_88.55547214383535.joblib
```

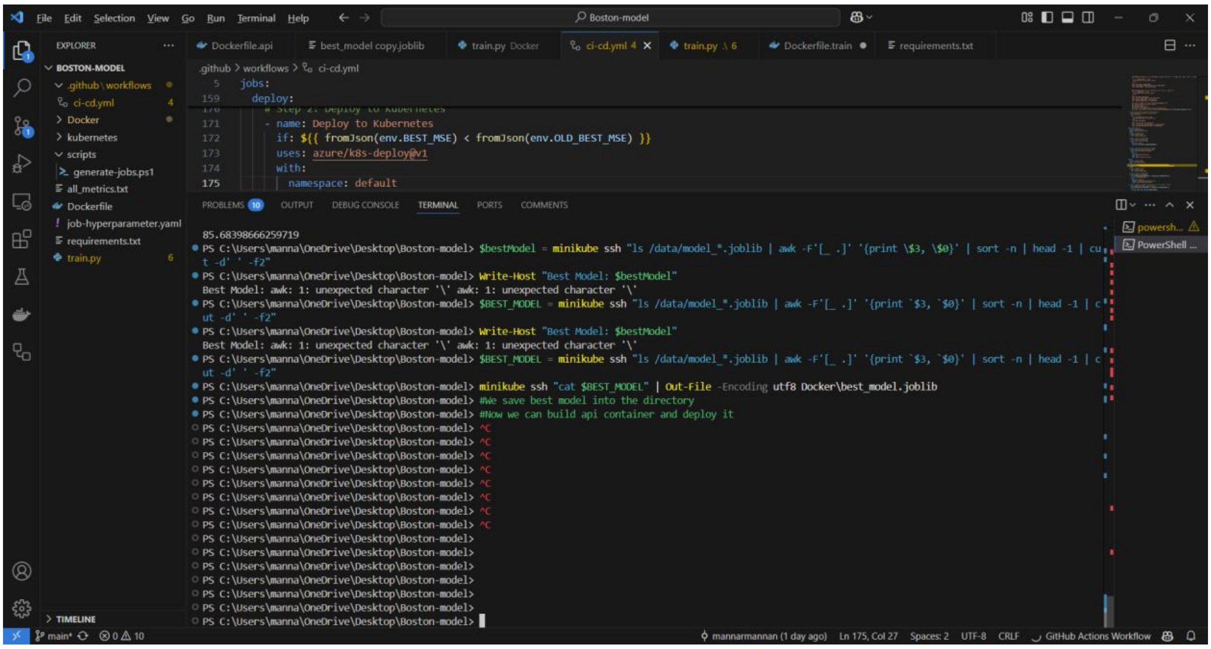
9. Retrieve Best Model

>>minikube ssh "ls /data/model_*.joblib | sort -V | head -1"

>>$BEST_MODEL = minikube ssh "ls /data/model_*.joblib | awk -F'[_ .]' '{print `$3, `$0}' | sort -n | head -1 | cut -d' ' -f2"

>>minikube ssh "cat $BEST_MODEL" | Out-File -Encoding utf8 Docker\best_model.joblib

Uses version-sorting to identify the model with lowest MSE from filename patterns.



```
jobs:
  deploy:
      # Step 2: Deploy to Kubernetes
      - name: Deploy to Kubernetes
        if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
        uses: azure/k8s-deploy@v1
        with:
          namespace: default
```

```
85.68398666259719
PS C:\Users\manna\OneDrive\Desktop\Boston-model> $bestModel = minikube ssh "ls /data/model_*.joblib | awk -F'[_ .]' '{print \$3, \$0}' | sort -n | head -1 | cu
t -d' ' -f2"
PS C:\Users\manna\OneDrive\Desktop\Boston-model> Write-Host "Best Model: $bestModel"
Best Model: awk: 1: unexpected character '\' awk: 1: unexpected character '\'
PS C:\Users\manna\OneDrive\Desktop\Boston-model> $BEST_MODEL = minikube ssh "ls /data/model_*.joblib | awk -F'[_ .]' '{print `$3, `$0}' | sort -n | head -1 | c
ut -d' ' -f2"
PS C:\Users\manna\OneDrive\Desktop\Boston-model> Write-Host "Best Model: $bestModel"
Best Model: awk: 1: unexpected character '\' awk: 1: unexpected character '\'
PS C:\Users\manna\OneDrive\Desktop\Boston-model> $BEST_MODEL = minikube ssh "ls /data/model_*.joblib | awk -F'[_ .]' '{print `$3, `$0}' | sort -n | head -1 | c
ut -d' ' -f2"
PS C:\Users\manna\OneDrive\Desktop\Boston-model> minikube ssh "cat $BEST_MODEL" | Out-File -Encoding utf8 Docker\best_model.joblib
PS C:\Users\manna\OneDrive\Desktop\Boston-model> #We save best model into the directory
PS C:\Users\manna\OneDrive\Desktop\Boston-model> #Now we can build api container and deploy it
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model> ^C
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
PS C:\Users\manna\OneDrive\Desktop\Boston-model>
```

Task 2: Build a CI/CD Pipeline

CI/CD workflow file for all the above instruction to automate the process

CI/CD Workflow Enhancements:

- Added clearer stage separation (build → tune → deploy)
- Fixed environment variable handling for MSE comparisons
- Improved error handling in model retrieval steps
- Added explicit skip-ci tags to prevent pipeline loops

Key Pipeline Improvements:

1. Parallel Build Stages
   Simultaneous image building and dependency installation for faster execution
2. Atomic Artifact Handling
   Models and metrics stored with MSE in filenames for easy version comparison
3. Rolling Update Strategy
   Conditional deployment only when new model outperforms previous best MSE
4. Persistent Metric Tracking
   Commits best MSE to repository for historical comparison across runs

NOTE: Major efforts not taken in deployment stage

```
          name: Model CI/CD Pipeline with Hyperparameter Tuning


on: [push]

jobs:
  build-and-train:
    runs-on: ubuntu-latest
    steps:
    # Step 1: Checkout code
    - name: Checkout code
      uses: actions/checkout@v2

    # Step 2: Set up Docker Buildx
    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v1

    # Step 3: Log in to Docker Hub
    - name: Log in to Docker Hub
      uses: docker/login-action@v1
      with:
```

```
      username: mannarn
      password: ${{ secrets.DOCKER_PASSWORD }}

  # Step 4: Build and push Train Docker image
  - name: Build and push Train Docker image
    uses: docker/build-push-action@v2
    with:
      context: .
      file: Docker/Dockerfile.train
      push: true
      tags: mannarn/model-train:latest

hyperparameter-tuning:
  runs-on: ubuntu-latest
  needs: build-and-train
  outputs:
    best_mse: ${{ steps.collect_results.outputs.best_mse }}
    best_model_file: ${{ steps.collect_results.outputs.best_model_file }}
    old_best_mse: ${{ steps.retrieve_old.outputs.old_best_mse }}
  steps:
  # Step 1: Checkout code
  - name: Checkout code
    uses: actions/checkout@v2

  # Step 2: Set up Python
  - name: Set up Python
    uses: actions/setup-python@v2
    with:
      python-version: '3.8'

  # Step 3: Install dependencies
  - name: Install dependencies
    run: pip install -r requirements.txt

  # Step 4: Set up Minikube
  - name: Set up Minikube
    run: |
      curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
      chmod +x minikube
      sudo mv minikube /usr/local/bin/
      minikube start --driver=docker --cpus=4 --memory=8192mb

  # Step 5: Generate Kubernetes Jobs for hyperparameter tuning
  - name: Generate Kubernetes Jobs
    shell: pwsh
    run: |
      chmod +x scripts/generate-jobs.ps1
      ./scripts/generate-jobs.ps1
  # Step 6: Wait for Jobs to complete
  - name: Wait for Jobs to complete
    run: |
      kubectl wait --for=condition=complete --timeout=1000s job --all || true

  # Step 7: Check Job and Pod Status
  - name: Check Job and Pod Status
    run: |
      kubectl get jobs
      kubectl get pods

  # Step 8: Retrieve logs of failed pods: Debugging
  - name: Retrieve logs of failed pods
    run: |
      for pod in $(kubectl get pods --field-selector=status.phase=Failed -o
jsonpath='{.items[*].metadata.name}'); do
        echo "Logs for pod $pod:"
        kubectl logs $pod
      done

  # Step 9: Get the results of hyperparameter tuning
  - name: Getting results
    id: getting_results
    run: |
      BEST_MODEL=$(minikube ssh "ls /data/model_*.joblib | awk -F'[_ .]' '{print \$3, \$0}' | sort -n |
head -1 | cut -d' ' -f2")

      if [ -z "$BEST_MODEL" ]; then
        echo "No model file found in /data/"
        exit 1
      fi
      echo "Best Model: $BEST_MODEL"
      echo "::set-output name=best_model_file::$BEST_MODEL"
      echo "$BEST_MODEL" > best_model.joblib


      BEST_MSE=$(minikube ssh "cat /data/mse_*.txt | sort -n | head -1")
      if [ -z "$BEST_MSE" ]; then
        echo "No MSE file found in /data/"
        exit 1
      fi
      echo "Best MSE: $BEST_MSE"
      echo "::set-output name=best_mse::$BEST_MSE"
      echo "Output: $BEST_MSE" > best_mse.txt
      echo "BEST_MSE=$BEST_MSE" >> $GITHUB_ENV

      git config --global user.email "mannarmannan02@gmail.com"
      git config --global user.name "mannarn"
```

```
        git add best_mse.txt
        git add best_model.joblib
        git commit -m "Uploading model and mse scores[skip ci]"
        git push --force https://x-access-token:${GITHUB_TOKEN}@github.com/${{ github.repository }}.git

    # Step 10: Retrieve old model's best MSE score
    - name: Retrieve old model's best MSE score
      id: retrieve_old
      run: |
        if [ -f old_best_mse.txt ]; then
          OLD_BEST_MSE=$(cat old_best_mse.txt)
        else
          OLD_BEST_MSE=999999
        fi
        echo "Old Best MSE: $OLD_BEST_MSE"
        echo "::set-output name=old_best_mse::$OLD_BEST_MSE"
        echo "OLD_BEST_MSE=$OLD_BEST_MSE" >> $GITHUB_ENV

  build-and-push-api:
    runs-on: ubuntu-latest
    needs: hyperparameter-tuning
    steps:
    # Step 1: Checkout code
    - name: Checkout code
      uses: actions/checkout@v2

    # Step 2: Log in to Docker Hub
    - name: Log in to Docker Hub
      uses: docker/login-action@v1
      with:
        username: mannarn
        password: ${{ secrets.DOCKER_PASSWORD }}


    # Step 4: Build and push API Docker image
    - name: Build and push API Docker image
      uses: docker/build-push-action@v2
      with:
        context: .
        file: Docker/Dockerfile.api
        push: true
        tags: mannarn/model-api:latest

  deploy:
    runs-on: ubuntu-latest
    needs: build-and-push-api
    env:
      BEST_MSE: ${{ needs.hyperparameter-tuning.outputs.best_mse }}
      OLD_BEST_MSE: ${{ needs.hyperparameter-tuning.outputs.old_best_mse }}
    steps:
    # Step 1: Checkout code
    - name: Checkout code
      uses: actions/checkout@v2

    # Step 2: Deploy to Kubernetes
    - name: Deploy to Kubernetes
      if: ${{ fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
      uses: azure/k8s-deploy@v1
      with:
        namespace: default
        manifests: kubernetes/deployment.yaml
        images: mannarn/model-api:latest

    # Step 3: Update old model's best MSE score if new model is better
    - name: Update old model's best MSE score
      if: ${{fromJson(env.BEST_MSE) < fromJson(env.OLD_BEST_MSE) }}
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      run: |
        echo "$BEST_MSE" > old_best_mse.txt
        git config --global user.email "mannarmannan02@gmail.com"
        git config --global user.name "mannarn"
        git add old_best_mse.txt
        git commit -m "Update old best MSE score[skip ci]"
        git push --force https://x-access-token:${GITHUB_TOKEN}@github.com/${{ github.repository
}}.git
```

Conclusion

In this project, I successfully developed and deployed an end-to-end machine learning pipeline for predicting house prices using the XGBoost regression model. The implementation involved hyperparameter tuning through Kubernetes Jobs, ensuring efficient parallel execution of multiple training trials.

By leveraging Kubernetes for parallelized hyperparameter tuning, I optimized model performance while maintaining scalability and automation. The integration of GitHub Actions enabled a robust CI/CD pipeline, automating the build, tuning, and deployment processes seamlessly.