# MLOps: Major Assignment

Ajinkya Ghodake (G24AI1046)

GitHub Repo: Major Assignment

---

## Step-by-Step Breakdown

- Pre-requisite for Step 1:
    - Create local project structure

```
(base) PS D:\Projects\Study_Assignments\ML_OPS> mkdir mlops-linear-regression

    Directory: D:\Projects\Study_Assignments\ML_OPS


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/30/2025  11:15 PM                mlops-linear-regression


(base) PS D:\Projects\Study_Assignments\ML_OPS> cd .\mlops-linear-regression\
```

```
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> mkdir src, tests, .github, .github/workflows


    Directory: D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression
```

```
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> New-Item -Path README.md -ItemType File
>> New-Item -Path .gitignore -ItemType File
>> New-Item -Path requirements.txt -ItemType File
>> New-Item -Path src/train.py -ItemType File -Force
>> New-Item -Path src/quantize.py -ItemType File -Force
>> New-Item -Path src/predict.py -ItemType File -Force
>> New-Item -Path src/utils.py -ItemType File -Force
>> New-Item -Path tests/test_train.py -ItemType File -Force
>> New-Item -Path .github/workflows/ci.yml -ItemType File -Force
```

## Step 1: Repository Setup

- Initialize repo with:
    - README.md

- .gitignore
- requirements.txt

```
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git init
Initialized empty Git repository in D:/Projects/Study_Assignments/ML_OPS/mlops-linear-regression/.git/
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> echo "# mlops-linear-regression" > README.md
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> echo "__pycache__/\n*.pyc\n*.pkl\n*.joblib\n.env/\n.env" > .gitignore
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> echo "scikit-learn/\njoblib/\n" > requirements.txt
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git add README.md, requirements.txt, .gitignore
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git commit -m "Step 1: Repository setup"
[master (root-commit) e85df81] Step 1: Repository setup
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 requirements.txt
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git remote add origin https://github.com/git-commit-acc/mlops-linear-regression.git
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git branch -M main
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 474 bytes | 474.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/git-commit-acc/mlops-linear-regression.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression>
```

- Pre-requisite for Step 2:
  - Create a conda virtual environment locally
  - Install the dependencies (requirements.txt)

```
(base) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> conda create -n mlops-linear-regression
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\ajink\anaconda3\envs\mlops-linear-regression



Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate mlops-linear-regression
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> pip install -r .\requirements.txt
Requirement already satisfied: scikit-learn in c:\users\ajink\appdata\local\programs\python\python313\lib\site-packages (from -r
Requirement already satisfied: numpy in c:\users\ajink\appdata\local\programs\python\python313\lib\site-packages (from -r .\requ
Requirement already satisfied: joblib in c:\users\ajink\appdata\local\programs\python\python313\lib\site-packages (from -r .\req
Collecting pytest (from -r .\requirements.txt (line 4))
```

## Step 2: Model Training (src/train.py)

- Load dataset.

```python
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

def load_data(test_size=0.2, random_state=45):
    data = fetch_california_housing()
    X_train, X_test, y_train, y_test = train_test_split(
        data.data, data.target, test_size=test_size, random_state=random_state
    )
    return X_train, X_test, y_train, y_test
```

- Train LinearRegression model.
- Print R2 score and loss.
- Save model using joblib.

train.py U ✕ utils.py U

src > train.py > ...

```python
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import joblib
import numpy as np
import sys, os
sys.path.append(os.path.dirname(__file__))
from utils import load_data

def main():
    X_train, X_test, y_train, y_test = load_data()

    model = LinearRegression()
    model.fit(X_train, y_train)

    preds = model.predict(X_test)
    r2 = r2_score(y_test, preds)
    mse = mean_squared_error(y_test, preds)

    print(f"R2 Score: {r2:.4f}")
    print(f"MSE: {mse:.4f}")

    joblib.dump(model, "src/trained_model.joblib")

if __name__ == "__main__":
    main()
```

- Test the code by running locally:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\train.py
R2 Score: 0.5758
MSE: 0.5559
```

- Commit changes to main branch:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git add src/train.py, src/utils.py, requirements.txt
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git commit -m "Step 2: Model Training done"
[main 5c4ff4c] Step 2: Model Training done
 3 files changed, 36 insertions(+)
 create mode 100644 src/train.py
 create mode 100644 src/utils.py
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 20 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.03 KiB | 1.03 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/git-commit-acc/mlops-linear-regression.git
   e85df81..5c4ff4c  main -> main
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression>
```

# Step 3: Testing Pipeline (tests/test train.py)

- Unit test dataset loading.
- Validate model creation (LinearRegression instance).
- Check if model was trained (e.g., coef exists).
- Ensure R2 score exceeds minimum threshold.

test_train.py U

tests > test_train.py > ...

```python
import sys, os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
from src.utils import load_data
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

def test_data_loading():
    X_train, X_test, y_train, y_test = load_data()
    assert X_train.shape[0] > 0
    assert X_test.shape[0] > 0

def test_model_creation():
    model = LinearRegression()
    assert isinstance(model, LinearRegression)

def test_model_training():
    X_train, X_test, y_train, y_test = load_data()
    model = LinearRegression()
    model.fit(X_train, y_train)
    assert hasattr(model, 'coef_')
    preds = model.predict(X_test)
    r2 = r2_score(y_test, preds)
    assert r2 > 0.5
```

- Test locally if these testcases work:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> pytest tests/
========================================================================== test session starts ==========================================================================
platform win32 -- Python 3.13.0, pytest-8.4.1, pluggy-1.6.0
rootdir: D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression
plugins: dash-3.0.3
collected 3 items

tests\test_train.py ...                                                                                                                                                  [100%]

========================================================================== 3 passed in 0.98s ==========================================================================
```

## Step 4: Manual Quantization (src/quantize.py)

- Load trained model.
- Extract coef and intercept .
- Save raw parameters (unquant params.joblib).
- Manually quantize them to unsigned 8-bit integers.
- Save quantized parameters (quant params.joblib).
- Perform inference with the de-quantized weights.

```
rc > quantize.py > dequantize
import joblib
import numpy as np
import sys, os
sys.path.append(os.path.dirname(__file__))
from utils import load_data
from sklearn.metrics import r2_score, mean_squared_error

def min_max_quantize(arr):
    arr_min, arr_max = arr.min(), arr.max()

    if arr_max == arr_min:
        quantized = np.full(arr.shape, 127, dtype=np.uint8)
        return quantized, arr_min, arr_max

    # Normal quantization
    quantized = ((arr - arr_min) / (arr_max - arr_min) * 255).round().astype(np.uint8)
    return quantized, arr_min, arr_max

def dequantize(quantized, arr_min, arr_max):

    if arr_max == arr_min:
        return np.full(quantized.shape, arr_min, dtype=np.float32)

    # Normal dequantization
    return quantized.astype(np.float32) / 255 * (arr_max - arr_min) + arr_min
```

```python
def main():
    model = joblib.load("src/trained_model.joblib")
    coef = model.coef_
    intercept = np.atleast_1d(model.intercept_)

    joblib.dump({'coef_': coef, 'intercept_': intercept}, "src/unquant_params.joblib")

    # Calculate original model performance
    X_train, X_test, y_train, y_test = load_data()
    orig_preds = np.dot(X_test, coef) + intercept[0]
    orig_r2 = r2_score(y_test, orig_preds)
    orig_mse = mean_squared_error(y_test, orig_preds)

    # Quantize weights
    q_coef, coef_min, coef_max = min_max_quantize(coef)
    q_intercept, int_min, int_max = min_max_quantize(intercept)
    joblib.dump({
        'q_coef': q_coef,
        'coef_min': coef_min,
        'coef_max': coef_max,
        'q_intercept': q_intercept,
        'int_min': int_min,
        'int_max': int_max,
    }, "src/quant_params.joblib")

    dq_coef = dequantize(q_coef, coef_min, coef_max)
    dq_intercept = dequantize(q_intercept, int_min, int_max)[0]
    preds = np.dot(X_test, dq_coef) + dq_intercept
    r2 = r2_score(y_test, preds)
    mse = mean_squared_error(y_test, preds)

    print(f"\n" + "="*50)
    print(f"RESULTS:")
    print(f"Original Model — R2: {orig_r2:.6f}, MSE: {orig_mse:.6f}")
    print(f"Quantized Model — R2: {r2:.4f}, MSE: {mse:.4f}")
    print(f"\n" + "="*50)

if __name__ == "__main__":
    main()
```

- Test locally to check the code:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\quantize.py

==================================================
RESULTS:
Original Model — R2: 0.575788, MSE: 0.555892
Quantized Model — R2: -0.1799, MSE: 1.5462

==================================================
```

## Step 5: Dockerization

Create a Dockerfile that:

• Installs dependencies

• Includes predict.py for model verification

| Job Name | Description | Depends On |
|---|---|---|
| test_suite | Runs pytest. Must pass before others execute. | None |
| train_and_quantize | Trains model, runs quantization, uploads artifacts | test_suite |
| build_and_test_container | Builds Docker image, runs container (must execute predict.py successfully) | train_and_quantize |

Dockerfile > ...

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ src/
COPY tests/ tests/
ENTRYPOINT ["python", "src/predict.py"]
```

- Test locally by building Docker image:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> docker build -t mlops-lr-demo:latest .
[+] Building 30.5s (11/11) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                   0.0s
 => => transferring dockerfile: 228B                                                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                                                    0.5s
 => [internal] load .dockerignore                                                                                                      0.0s
 => => transferring context: 2B                                                                                                        0.0s
 => [1/6] FROM docker.io/library/python:3.10-slim@sha256:81f1cdb3770d54ecfdbddcc52c2125fce674c14a1d976dfd8f65dc0734f9c3c5              0.0s
 => => resolve docker.io/library/python:3.10-slim@sha256:81f1cdb3770d54ecfdbddcc52c2125fce674c14a1d976dfd8f65dc0734f9c3c5              0.0s
 => [internal] load build context                                                                                                      0.0s
 => => transferring context: 12.41kB                                                                                                   0.0s
 => CACHED [2/6] WORKDIR /app                                                                                                          0.0s
 => [3/6] COPY requirements.txt .                                                                                                      0.0s
 => [4/6] RUN pip install --no-cache-dir -r requirements.txt                                                                          20.2s
 => [5/6] COPY src/ src/                                                                                                               0.0s
 => [6/6] COPY tests/ tests/                                                                                                           0.0s
 => exporting to image                                                                                                                 9.4s
 => => exporting layers                                                                                                                8.0s
 => => exporting manifest sha256:b183e328fcd9bdaa6b2f356fb2438c4846b1ef2e9e4e92db89107148beafcd4e                                      0.0s
 => => exporting config sha256:988919e81aa4f07a29881ad7ec8ea7c75750b020926d6faef4853a7e5a07aa45                                        0.0s
 => => exporting attestation manifest sha256:8ca0d430fe45e2a711209b8de2729289289c8428087f42f7ab1685ee045d37e6                          0.0s
 => => exporting manifest list sha256:fbc547650e3aeb1a2c7a0a7b74d8fb29c3bdabeb37d32bed2f8ea12a2150e310                                 0.0s
 => => naming to docker.io/library/mlops-lr-demo:latest                                                                                0.0s
 => => unpacking to docker.io/library/mlops-lr-demo:latest                                                                             1.4s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression>
```

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> docker run mlops-lr-demo:latest
/usr/local/lib/python3.10/site-packages/sklearn/base.py:442: InconsistentVersionWarning: Trying to unpickle estimator LinearRegressio
is might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Sample predictions: [1.39867688 1.70268297 2.32887843 0.99256975 2.31608385]
Corresponding ground truths: [0.658 2.284 2.411 1.375 1.93 ]
```

src/predict.py:

- Load trained model
- Run prediction on test set
- Print sample outputs

predict.py U × Dockerfile 1, U

src > predict.py > ...

```python
import joblib
import sys, os
sys.path.append(os.path.dirname(__file__))
from utils import load_data

def main():
    _, X_test, _, y_test = load_data()
    model = joblib.load("src/trained_model.joblib")
    preds = model.predict(X_test)
    print("Sample predictions:", preds[:5])
    print("Corresponding ground truths:", y_test[:5])

if __name__ == "__main__":
    main()
```

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\predict.py
Sample predictions: [0.71912284 1.76401657 2.70965883 2.83892593 2.60465725]
Corresponding ground truths: [0.477   0.458   5.00001 2.186   2.78   ]
```

## Step 6: CI/CD Workflow (.github/workflows/ci.yml)

Run on every push to main.

Define 3 jobs:

predict.py U | ci.yml U | Dockerfile 1, U

github > workflows > ci.yml

```yaml
name: MLOps Workflow

on:
  push:
    branches: [main]

jobs:
  test-suite:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: '3.10'
      - run: pip install -r requirements.txt
      - run: pytest tests/

  train-and-quantize:
    needs: test-suite
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: '3.10'
      - run: pip install -r requirements.txt
      - run: python src/train.py
      - run: python src/quantize.py
      - name: Upload artifacts
        uses: actions/upload-artifact@v4
        with:
          name: model
          path: |
            src/trained_model.joblib
            src/quant_params.joblib

  build-and-test-container:
    needs: train-and-quantize
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: docker build -t mlops-lr-demo .
      - run: docker run mlops-lr-demo
```

# Outputs

- Execution in local environment:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\train.py
R2 Score: 0.5758
MSE: 0.5559
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> pytest tests/
============================================================================= test session starts ==============================
platform win32 -- Python 3.13.0, pytest-8.4.1, pluggy-1.6.0
rootdir: D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression
plugins: dash-3.0.3
collected 3 items

tests\test_train.py ...

============================================================================== 3 passed in 0.97s ===============================
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\quantize.py

==================================================
RESULTS:
Original Model — R2: 0.575788, MSE: 0.555892
Quantized Model — R2: -0.1799, MSE: 1.5462

==================================================
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> python .\src\predict.py
Sample predictions: [0.71912284 1.76401657 2.70965883 2.83892593 2.60465725]
Corresponding ground truths: [0.477   0.458   5.00001 2.186   2.78   ]
```

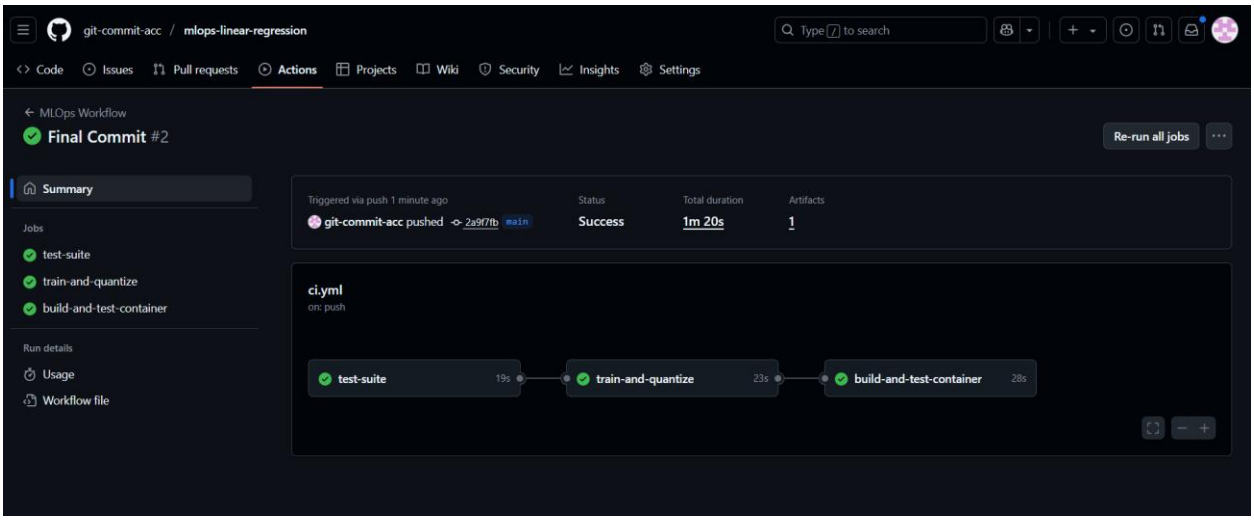
- Docker containerization:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> docker build -t mlops-lr-demo:latest .
[+] Building 2.8s (11/11) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 228B                                                                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                                                 2.4s
 => [internal] load .dockerignore                                                                                                   0.0s
 => => transferring context: 2B                                                                                                     0.0s
 => [1/6] FROM docker.io/library/python:3.10-slim@sha256:81f1cdb3770d54ecfdbddcc52c2125fce674c14a1d976dfd8f65dc0734f9c3c5           0.0s
 => => resolve docker.io/library/python:3.10-slim@sha256:81f1cdb3770d54ecfdbddcc52c2125fce674c14a1d976dfd8f65dc0734f9c3c5           0.0s
 => [internal] load build context                                                                                                   0.0s
 => => transferring context: 5.94kB                                                                                                 0.0s
 => CACHED [2/6] WORKDIR /app                                                                                                       0.0s
 => CACHED [3/6] COPY requirements.txt .                                                                                            0.0s
 => CACHED [4/6] RUN pip install --no-cache-dir -r requirements.txt                                                                 0.0s
 => [5/6] COPY src/ src/                                                                                                            0.0s
 => [6/6] COPY tests/ tests/                                                                                                        0.0s
 => exporting to image                                                                                                              0.2s
 => => exporting layers                                                                                                             0.1s
 => => exporting manifest sha256:9ff89bb2e321f370ead9704a94a60994d20ff6ef1b75fa24395369485697fe09                                   0.0s
 => => exporting config sha256:178f50e36fe6d1bb86486a24bdc7d12514ec1da9a93452c962fb720d4bdcd49d                                     0.0s
 => => exporting attestation manifest sha256:ff80d65ddce0dcf456ac7a5555ea283bcd59bdd94529cd2c4d7726a6a0c6eefe                       0.0s
 => => exporting manifest list sha256:39f88e39b617815df5661060ac0d7d725472d88a219930a48823e7c16e69329e                              0.0s
 => => naming to docker.io/library/mlops-lr-demo:latest                                                                             0.0s
 => => unpacking to docker.io/library/mlops-lr-demo:latest                                                                          0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> docker run mlops-lr-demo:latest
/usr/local/lib/python3.10/site-packages/sklearn/base.py:442: InconsistentVersionWarning: Trying to unpickle estimator LinearRegression from version 1.7.0 when using version 1.7.1. Th
is might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Sample predictions: [0.71912284 1.76401657 2.70965883 2.83892593 2.60465725]
Corresponding ground truths: [0.477   0.458   5.00001 2.186   2.78   ]
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression>
```

- Github Actions:

```
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git add .
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git commit -m "Final Commit"
[main 2a9f7fb] Final Commit
 6 files changed, 37 insertions(+), 44 deletions(-)
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression> git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 20 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 2.09 KiB | 2.09 MiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/git-commit-acc/mlops-linear-regression.git
   d07d6c2..2a9f7fb  main -> main
(mlops-linear-regression) PS D:\Projects\Study_Assignments\ML_OPS\mlops-linear-regression>
```




**Note:** Uploading docker image to Docker hub was not mentioned in the Assignment guidelines. Hence, I have not implemented any logic to upload the Docker image to Docker hub.