Machine Learning Operations

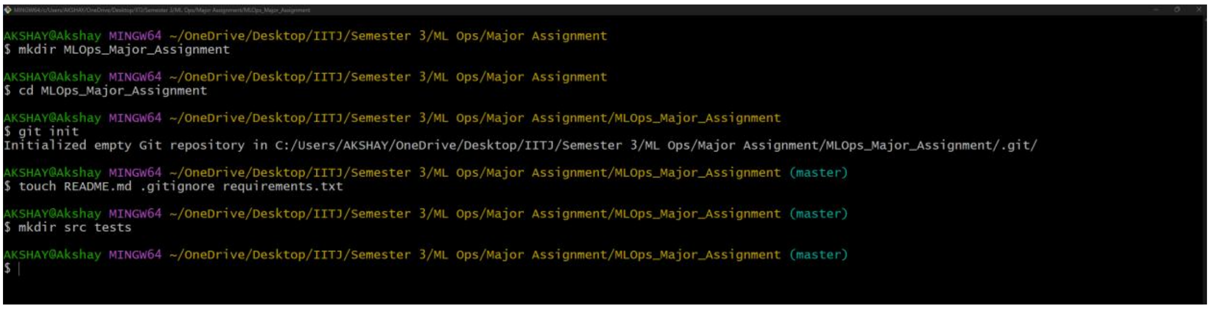
Major Assignment

Submitted By – Akshay Kumar (G24AI1033)

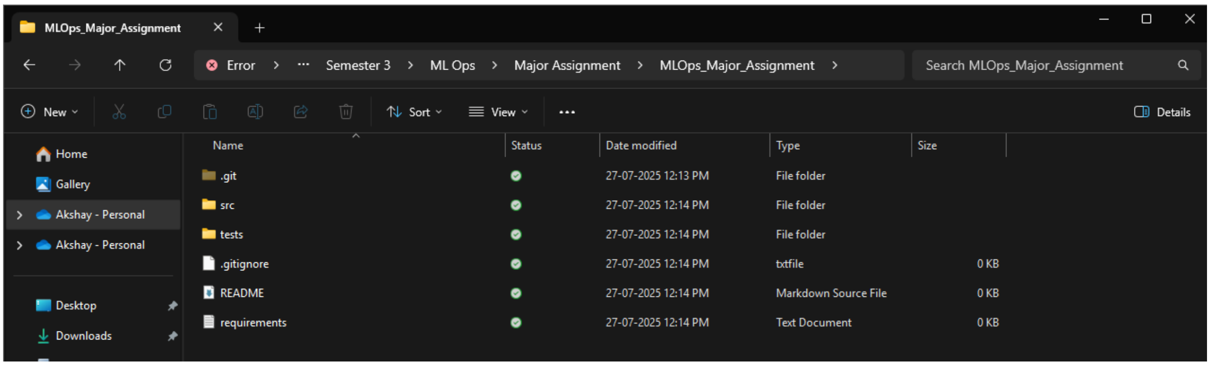
GitHub Repository Link – Github_Major_Assignment

Phase 1: Repository Setup (using terminal commands)

GitBash Command: mkdir MLOps_Assignment
cd MLOps_Assignment
git init
touch README.md .gitignore requirements.txt
mkdir src tests



```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment
$ mkdir MLOps_Major_Assignment

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment
$ cd MLOps_Major_Assignment

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment
$ git init
Initialized empty Git repository in C:/Users/AKSHAY/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment/.git/

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ touch README.md .gitignore requirements.txt

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ mkdir src tests

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ |
```

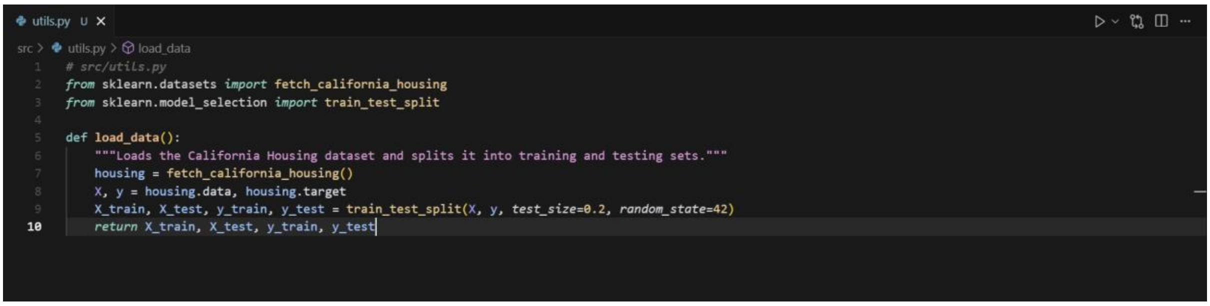


Name	Status	Date modified	Type	Size
.git	✓	27-07-2025 12:13 PM	File folder	
src	✓	27-07-2025 12:14 PM	File folder	
tests	✓	27-07-2025 12:14 PM	File folder	
.gitignore	✓	27-07-2025 12:14 PM	txtfile	0 KB
README	✓	27-07-2025 12:14 PM	Markdown Source File	0 KB
requirements	✓	27-07-2025 12:14 PM	Text Document	0 KB

Phase 2: Model Training (src/train.py)

Step 1: Create src/utils.py and add necessary imports

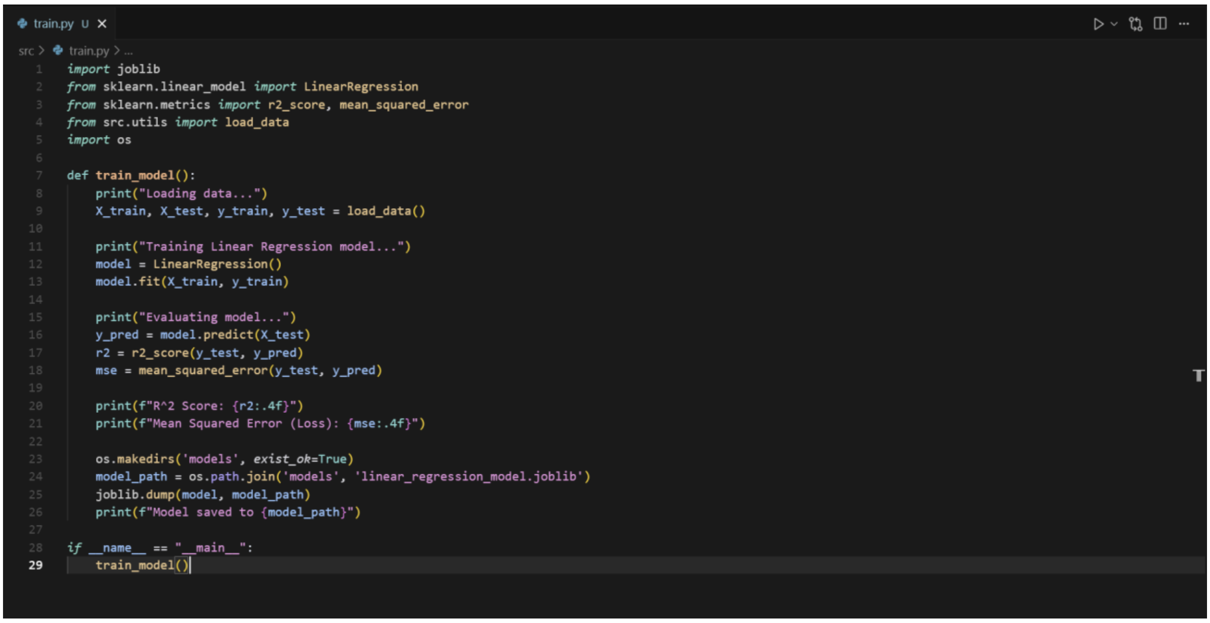
GitBash Command: touch src/utils.py



```python
# src/utils.py
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split

def load_data():
    """Loads the California Housing dataset and splits it into training and testing sets."""
    housing = fetch_california_housing()
    X, y = housing.data, housing.target
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test
```

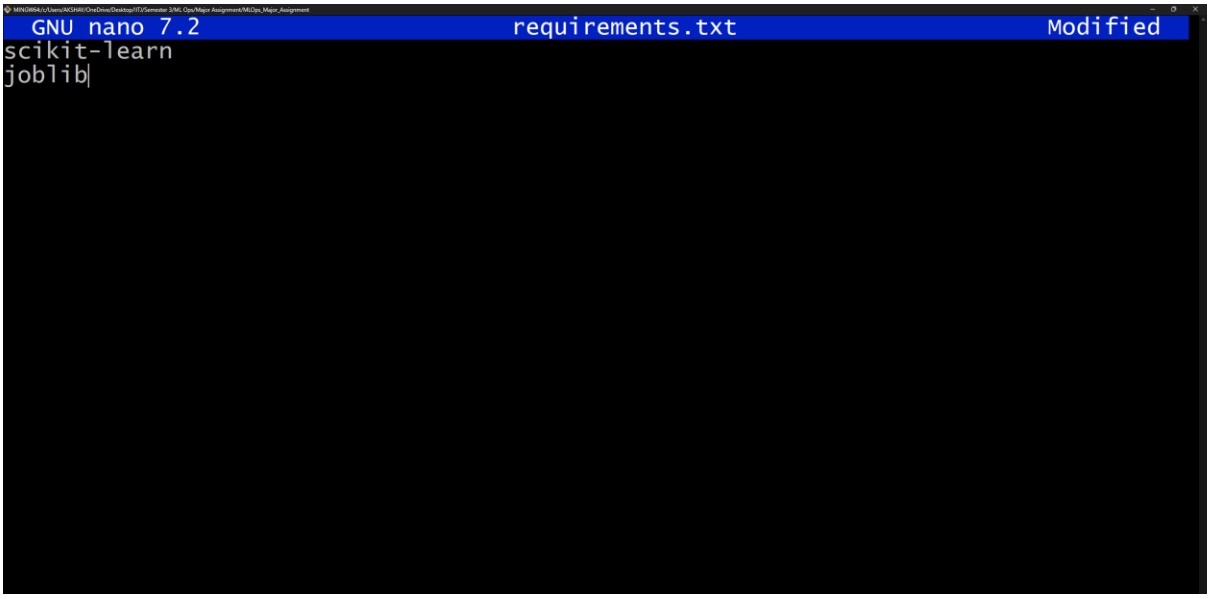
Step 2: Create src/train.py and add the training logic

GitBash Command: touch src/train.py



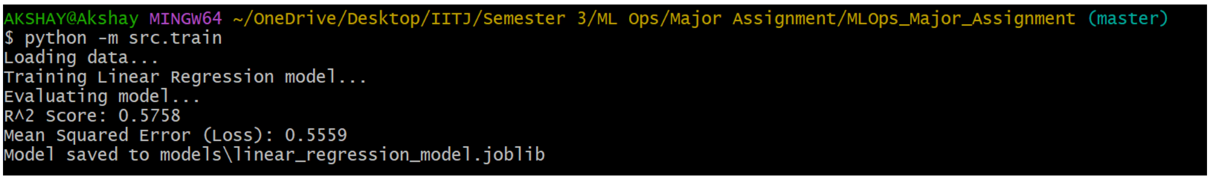
```python
import joblib
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from src.utils import load_data
import os

def train_model():
    print("Loading data...")
    X_train, X_test, y_train, y_test = load_data()

    print("Training Linear Regression model...")
    model = LinearRegression()
    model.fit(X_train, y_train)

    print("Evaluating model...")
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    print(f"R^2 Score: {r2:.4f}")
    print(f"Mean Squared Error (Loss): {mse:.4f}")

    os.makedirs('models', exist_ok=True)
    model_path = os.path.join('models', 'linear_regression_model.joblib')
    joblib.dump(model, model_path)
    print(f"Model saved to {model_path}")

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

Step 3: Update requirements.txt



```
  GNU nano 7.2                    requirements.txt                    Modified
scikit-learn
joblib
```

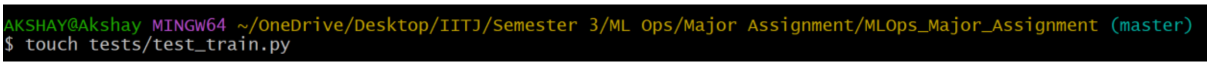
Step 4: Output of running train.py



```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ python -m src.train
Loading data...
Training Linear Regression model...
Evaluating model...
R^2 Score: 0.5758
Mean Squared Error (Loss): 0.5559
Model saved to models\linear_regression_model.joblib
```

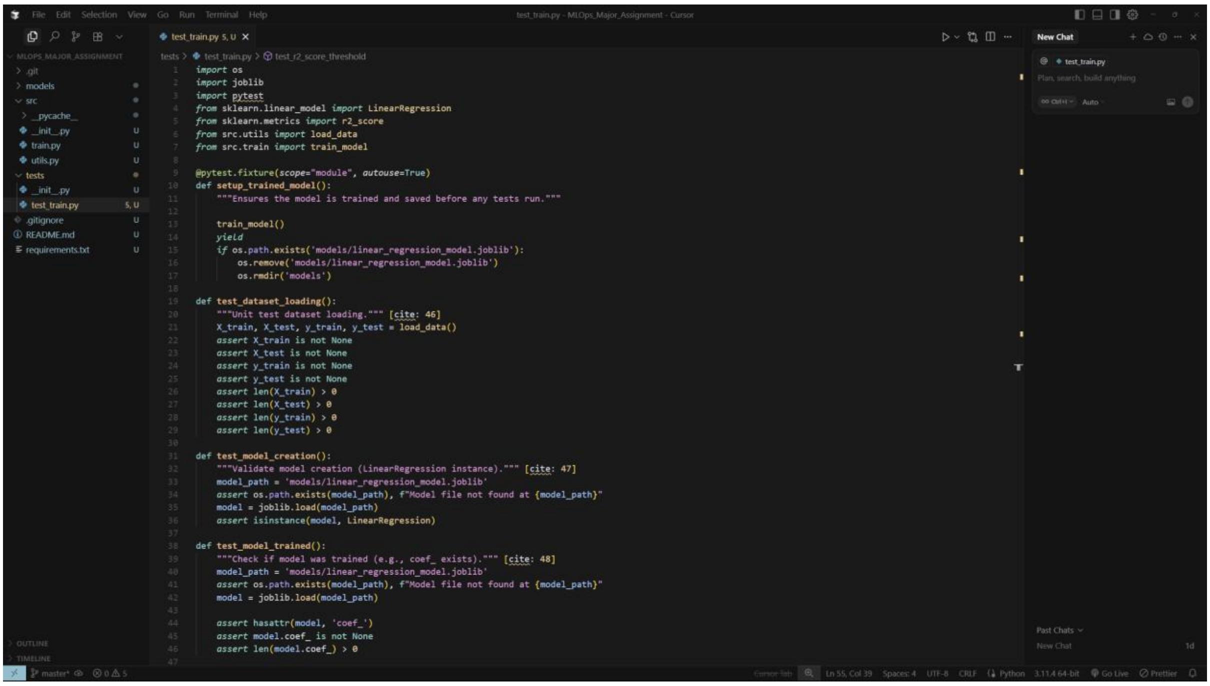
Phase 3: Testing Pipeline (tests/test_train.py)

Step 1: Create tests/test_train.py



```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ touch tests/test_train.py
```

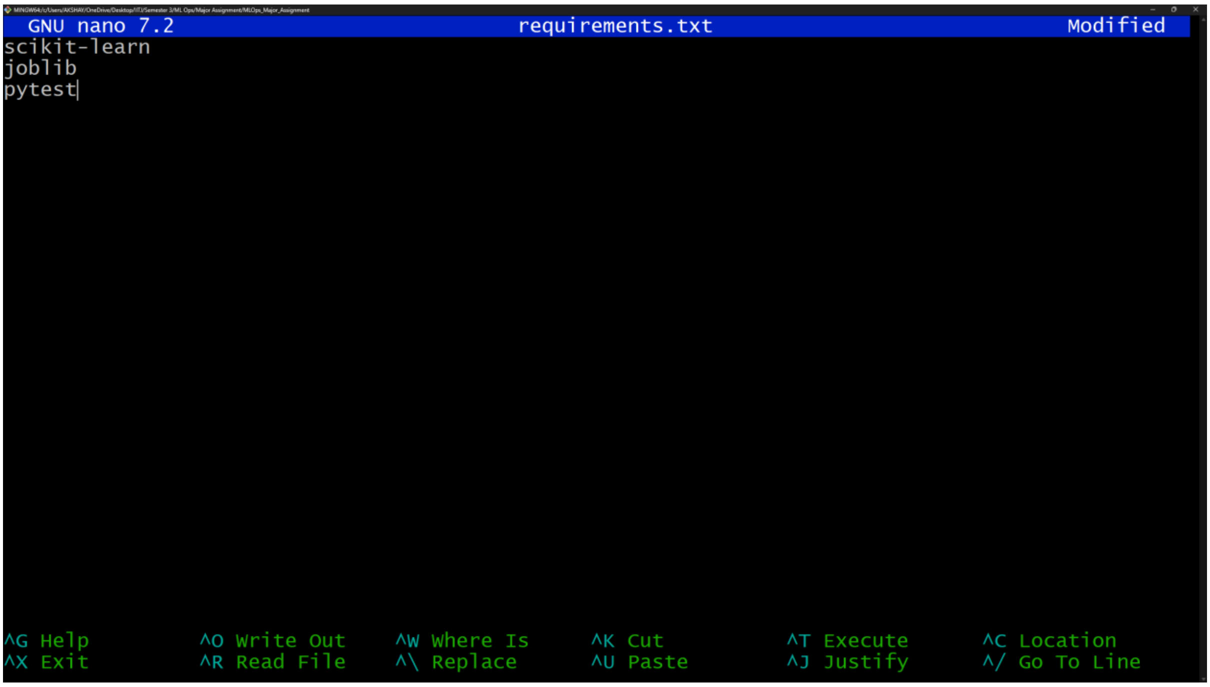
Step 2: Add content to tests/test_train.py



File Edit Selection View Go Run Terminal Help
test_train.py - MLOps_Major_Assignment - Cursor
MLOPS_MAJOR_ASSIGNMENT
.git
models
src
__pycache__
__init__.py
train.py
utils.py
tests
__init__.py
test_train.py
.gitignore
README.md
requirements.txt
OUTLINE
TIMELINE
test_train.py 5, U
tests > test_train.py > test_r2_score_threshold
New Chat
Plan, search, build anything
Auto
Past Chats
New Chat
master*
Ln 55, Col 39 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit Go Live Prettier

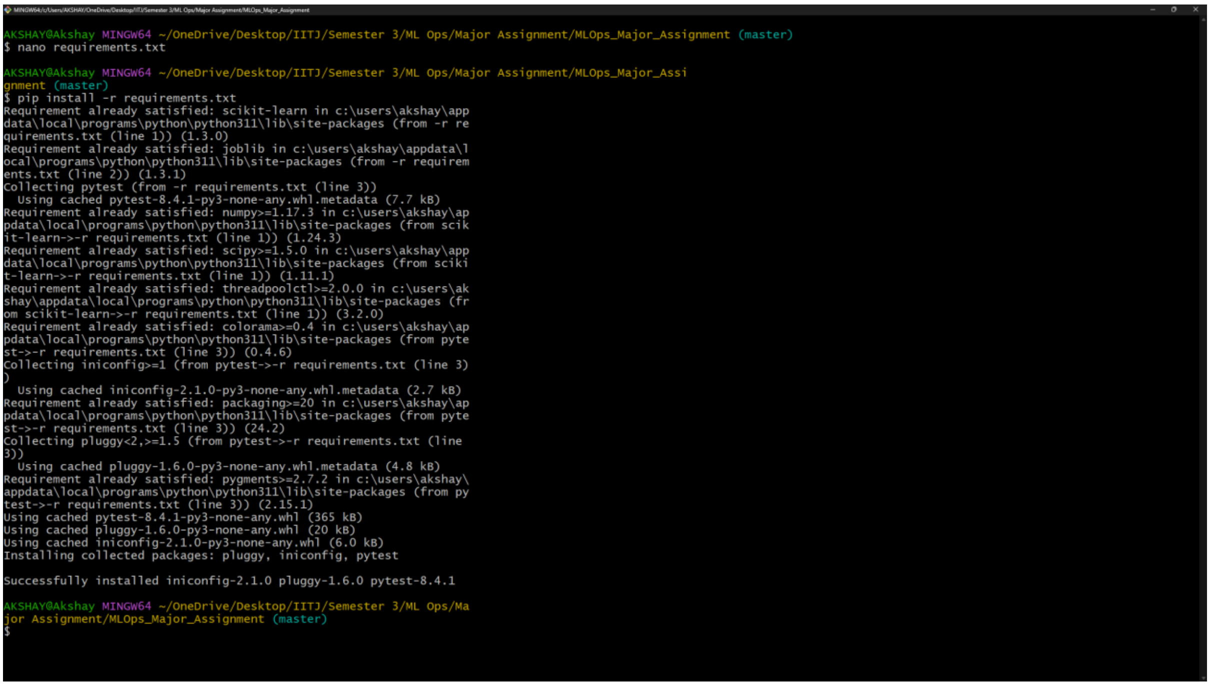
```python
import os
import joblib
import pytest
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from src.utils import load_data
from src.train import train_model

@pytest.fixture(scope="module", autouse=True)
def setup_trained_model():
    """Ensures the model is trained and saved before any tests run."""

    train_model()
    yield
    if os.path.exists('models/linear_regression_model.joblib'):
        os.remove('models/linear_regression_model.joblib')
        os.rmdir('models')

def test_dataset_loading():
    """Unit test dataset loading.""" [cite: 46]
    X_train, X_test, y_train, y_test = load_data()
    assert X_train is not None
    assert X_test is not None
    assert y_train is not None
    assert y_test is not None
    assert len(X_train) > 0
    assert len(X_test) > 0
    assert len(y_train) > 0
    assert len(y_test) > 0

def test_model_creation():
    """Validate model creation (LinearRegression instance).""" [cite: 47]
    model_path = 'models/linear_regression_model.joblib'
    assert os.path.exists(model_path), f"Model file not found at {model_path}"
    model = joblib.load(model_path)
    assert isinstance(model, LinearRegression)

def test_model_trained():
    """Check if model was trained (e.g., coef_ exists).""" [cite: 48]
    model_path = 'models/linear_regression_model.joblib'
    assert os.path.exists(model_path), f"Model file not found at {model_path}"
    model = joblib.load(model_path)

    assert hasattr(model, 'coef_')
    assert model.coef_ is not None
    assert len(model.coef_) > 0
```

Step 3: Update requirements.txt (add pytest):



```
  GNU nano 7.2                    requirements.txt                    Modified
scikit-learn
joblib
pytest






^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

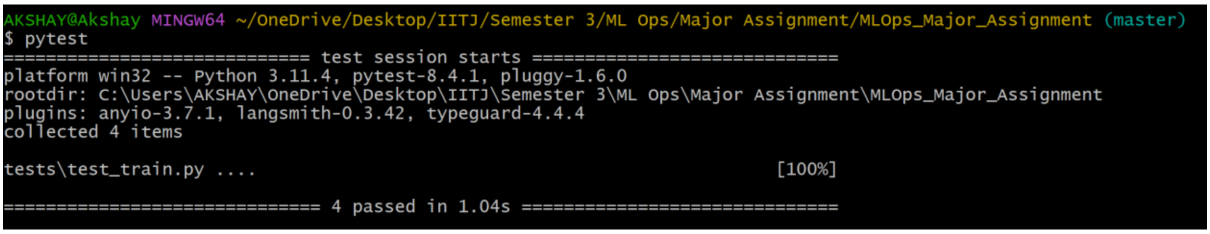
Step 4: Installing new dependencies:



MINGW64:/c/Users/AKSHAY/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment

```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ nano requirements.txt

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assi
gnment (master)
$ pip install -r requirements.txt
Requirement already satisfied: scikit-learn in c:\users\akshay\app
data\local\programs\python\python311\lib\site-packages (from -r re
quirements.txt (line 1)) (1.3.0)
Requirement already satisfied: joblib in c:\users\akshay\appdata\l
ocal\programs\python\python311\lib\site-packages (from -r requirem
ents.txt (line 2)) (1.3.1)
Collecting pytest (from -r requirements.txt (line 3))
  Using cached pytest-8.4.1-py3-none-any.whl.metadata (7.7 kB)
Requirement already satisfied: numpy>=1.17.3 in c:\users\akshay\ap
pdata\local\programs\python\python311\lib\site-packages (from scik
it-learn->-r requirements.txt (line 1)) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\akshay\app
data\local\programs\python\python311\lib\site-packages (from sciki
t-learn->-r requirements.txt (line 1)) (1.11.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ak
shay\appdata\local\programs\python\python311\lib\site-packages (fr
om scikit-learn->-r requirements.txt (line 1)) (3.2.0)
Requirement already satisfied: colorama>=0.4 in c:\users\akshay\ap
pdata\local\programs\python\python311\lib\site-packages (from pyte
st->-r requirements.txt (line 3)) (0.4.6)
Collecting iniconfig>=1 (from pytest->-r requirements.txt (line 3)
)
  Using cached iniconfig-2.1.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: packaging>=20 in c:\users\akshay\ap
pdata\local\programs\python\python311\lib\site-packages (from pyte
st->-r requirements.txt (line 3)) (24.2)
Collecting pluggy<2,>=1.5 (from pytest->-r requirements.txt (line
3))
  Using cached pluggy-1.6.0-py3-none-any.whl.metadata (4.8 kB)
Requirement already satisfied: pygments>=2.7.2 in c:\users\akshay\
appdata\local\programs\python\python311\lib\site-packages (from py
test->-r requirements.txt (line 3)) (2.15.1)
Using cached pytest-8.4.1-py3-none-any.whl (365 kB)
Using cached pluggy-1.6.0-py3-none-any.whl (20 kB)
Using cached iniconfig-2.1.0-py3-none-any.whl (6.0 kB)
Installing collected packages: pluggy, iniconfig, pytest

Successfully installed iniconfig-2.1.0 pluggy-1.6.0 pytest-8.4.1

AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Ma
jor Assignment/MLOps_Major_Assignment (master)
$
```

Step 5: Testing pytest



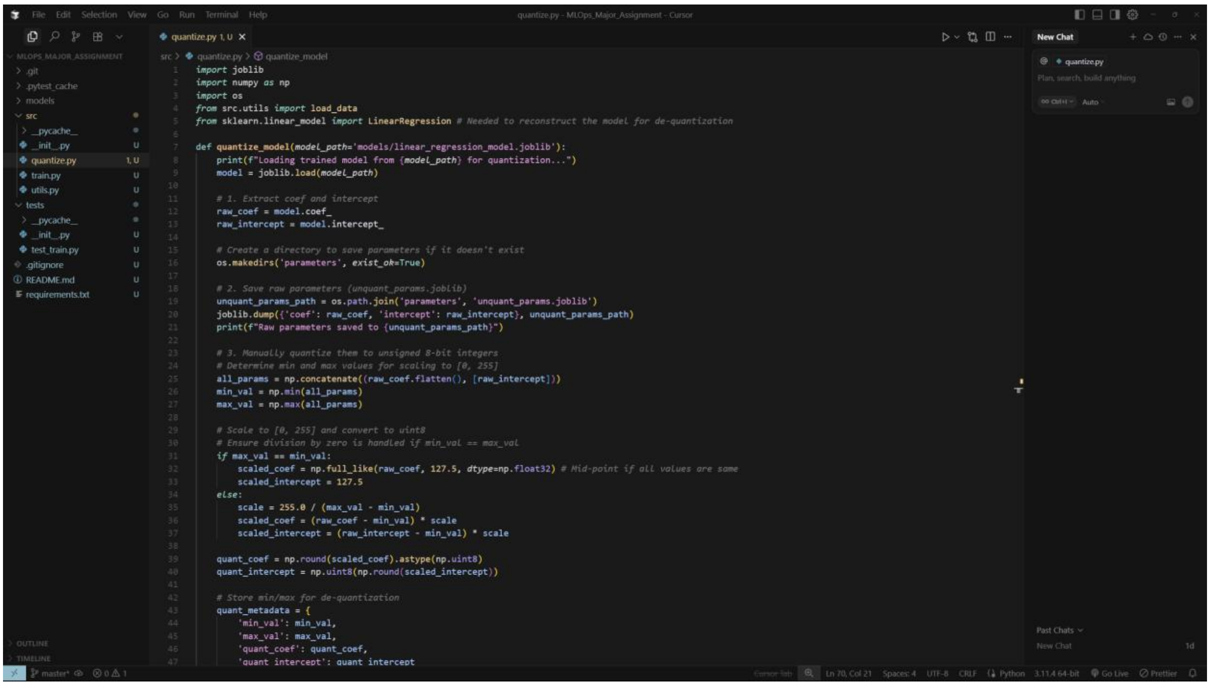
```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ pytest
============================= test session starts =============================
platform win32 -- Python 3.11.4, pytest-8.4.1, pluggy-1.6.0
rootdir: C:\Users\AKSHAY\OneDrive\Desktop\IITJ\Semester 3\ML Ops\Major Assignment\MLOps_Major_Assignment
plugins: anyio-3.7.1, langsmith-0.3.42, typeguard-4.4.4
collected 4 items

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

============================== 4 passed in 1.04s ==============================
```

Phase 4: Manual Quantization (src/quantize.py)

Step 1: Create src/quantize.py

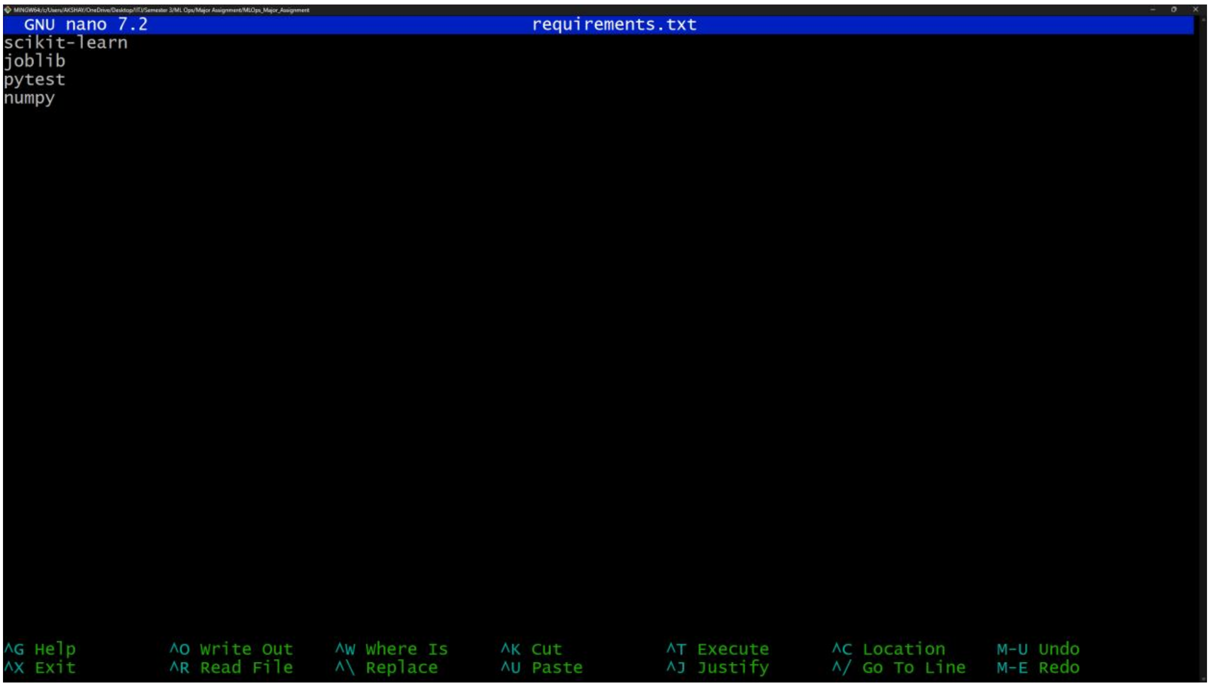
GitBash Command - touch src/quantize.py

Step 2: Add content to src/quantize.py



```python
import joblib
import numpy as np
import os
from src.utils import load_data
from sklearn.linear_model import LinearRegression # Needed to reconstruct the model for de-quantization

def quantize_model(model_path='models/linear_regression_model.joblib'):
    print(f"Loading trained model from {model_path} for quantization...")
    model = joblib.load(model_path)

    # 1. Extract coef and intercept
    raw_coef = model.coef_
    raw_intercept = model.intercept_

    # Create a directory to save parameters if it doesn't exist
    os.makedirs('parameters', exist_ok=True)

    # 2. Save raw parameters (unquant_params.joblib)
    unquant_params_path = os.path.join('parameters', 'unquant_params.joblib')
    joblib.dump({'coef': raw_coef, 'intercept': raw_intercept}, unquant_params_path)
    print(f"Raw parameters saved to {unquant_params_path}")

    # 3. Manually quantize them to unsigned 8-bit integers
    # Determine min and max values for scaling to [0, 255]
    all_params = np.concatenate((raw_coef.flatten(), [raw_intercept]))
    min_val = np.min(all_params)
    max_val = np.max(all_params)

    # Scale to [0, 255] and convert to uint8
    # Ensure division by zero is handled if min_val == max_val
    if max_val == min_val:
        scaled_coef = np.full_like(raw_coef, 127.5, dtype=np.float32) # Mid-point if all values are same
        scaled_intercept = 127.5
    else:
        scale = 255.0 / (max_val - min_val)
        scaled_coef = (raw_coef - min_val) * scale
        scaled_intercept = (raw_intercept - min_val) * scale

    quant_coef = np.round(scaled_coef).astype(np.uint8)
    quant_intercept = np.uint8(np.round(scaled_intercept))

    # Store min/max for de-quantization
    quant_metadata = {
        'min_val': min_val,
        'max_val': max_val,
        'quant_coef': quant_coef,
        'quant_intercept': quant_intercept
```

Step 3: Update requirements.txt (add numpy)

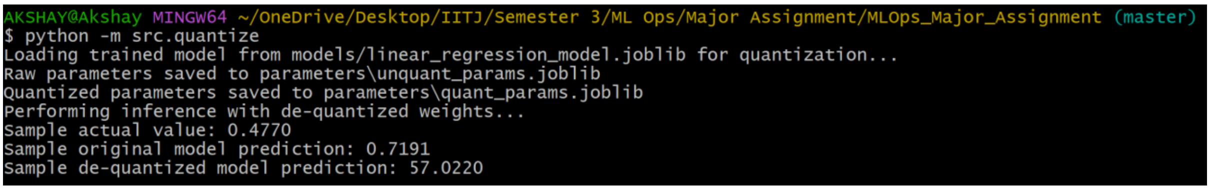


```
MINGW64:/c/Users/AKSHAY/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment
  GNU nano 7.2                              requirements.txt
scikit-learn
joblib
pytest
numpy

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```

Step 4: Install new dependencies:

GitBash Command: pip install -r requirements.txt

Step 5: Run the quantization script



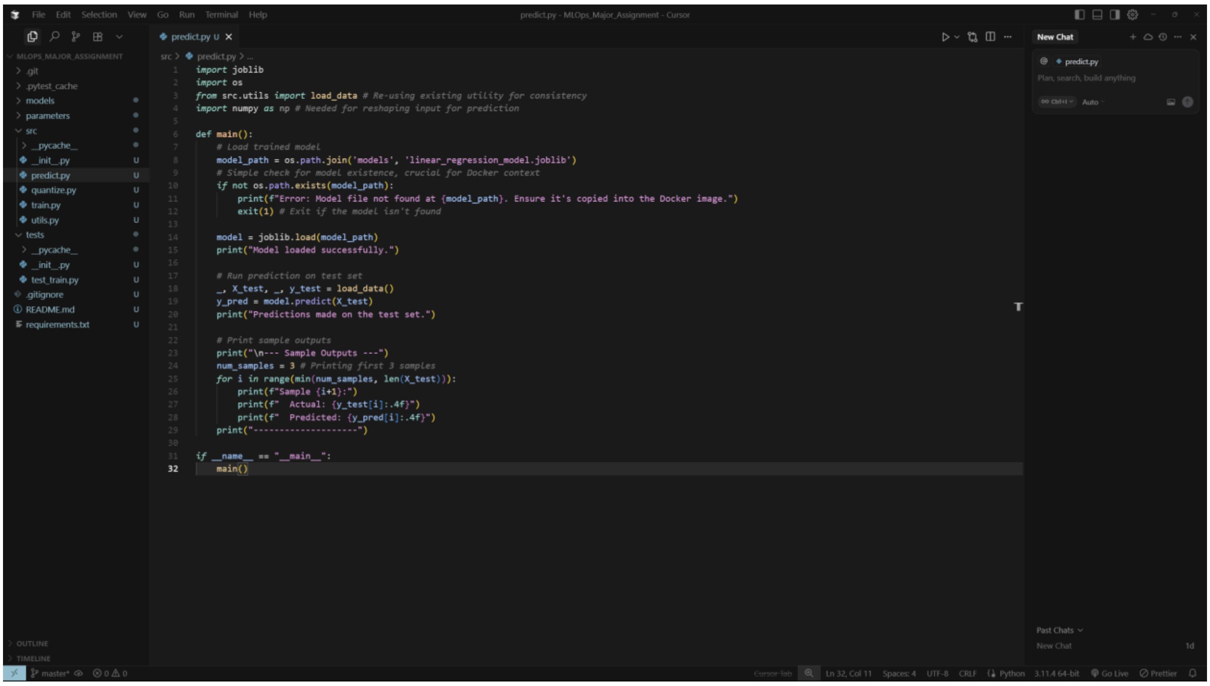
```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ python -m src.quantize
Loading trained model from models/linear_regression_model.joblib for quantization...
Raw parameters saved to parameters\unquant_params.joblib
Quantized parameters saved to parameters\quant_params.joblib
Performing inference with de-quantized weights...
Sample actual value: 0.4770
Sample original model prediction: 0.7191
Sample de-quantized model prediction: 57.0220
```

Phase 5 : Dockerization - Creating src/predict.py and Dockerfile

Step 1: Create src/predict.py

GitBash Command - touch src/predict.py

Step 2: Add content to src/predict.py

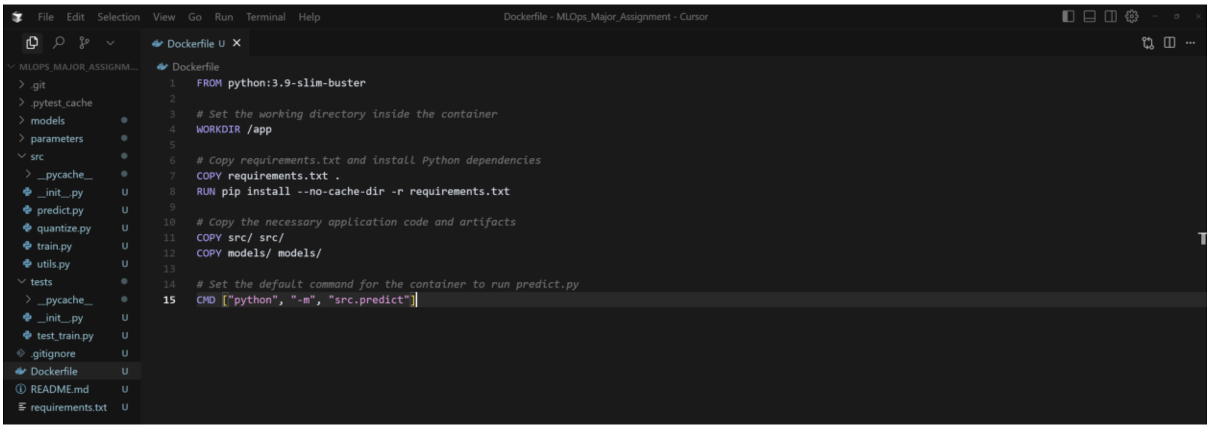


```python
import joblib
import os
from src.utils import load_data # Re-using existing utility for consistency
import numpy as np # Needed for reshaping input for prediction

def main():
    # Load trained model
    model_path = os.path.join('models', 'linear_regression_model.joblib')
    # Simple check for model existence, crucial for Docker context
    if not os.path.exists(model_path):
        print(f"Error: Model file not found at {model_path}. Ensure it's copied into the Docker image.")
        exit(1) # Exit if the model isn't found

    model = joblib.load(model_path)
    print("Model loaded successfully.")

    # Run prediction on test set
    _, X_test, _, y_test = load_data()
    y_pred = model.predict(X_test)
    print("Predictions made on the test set.")

    # Print sample outputs
    print("\n--- Sample Outputs ---")
    num_samples = 3 # Printing first 3 samples
    for i in range(min(num_samples, len(X_test))):
        print(f"Sample {i+1}:")
        print(f"  Actual: {y_test[i]:.4f}")
        print(f"  Predicted: {y_pred[i]:.4f}")
    print("------------------")

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

Step 3: Create Dockerfile

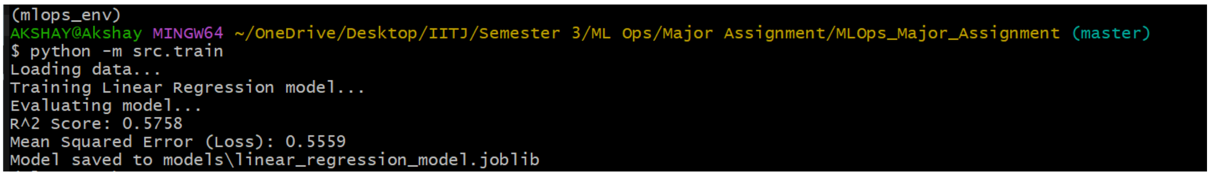
GitBash Command - touch Dockerfile

Step 4: Add content to Dockerfile



```dockerfile
FROM python:3.9-slim-buster

# Set the working directory inside the container
WORKDIR /app

# Copy requirements.txt and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the necessary application code and artifacts
COPY src/ src/
COPY models/ models/

# Set the default command for the container to run predict.py
CMD ["python", "-m", "src.predict"]
```

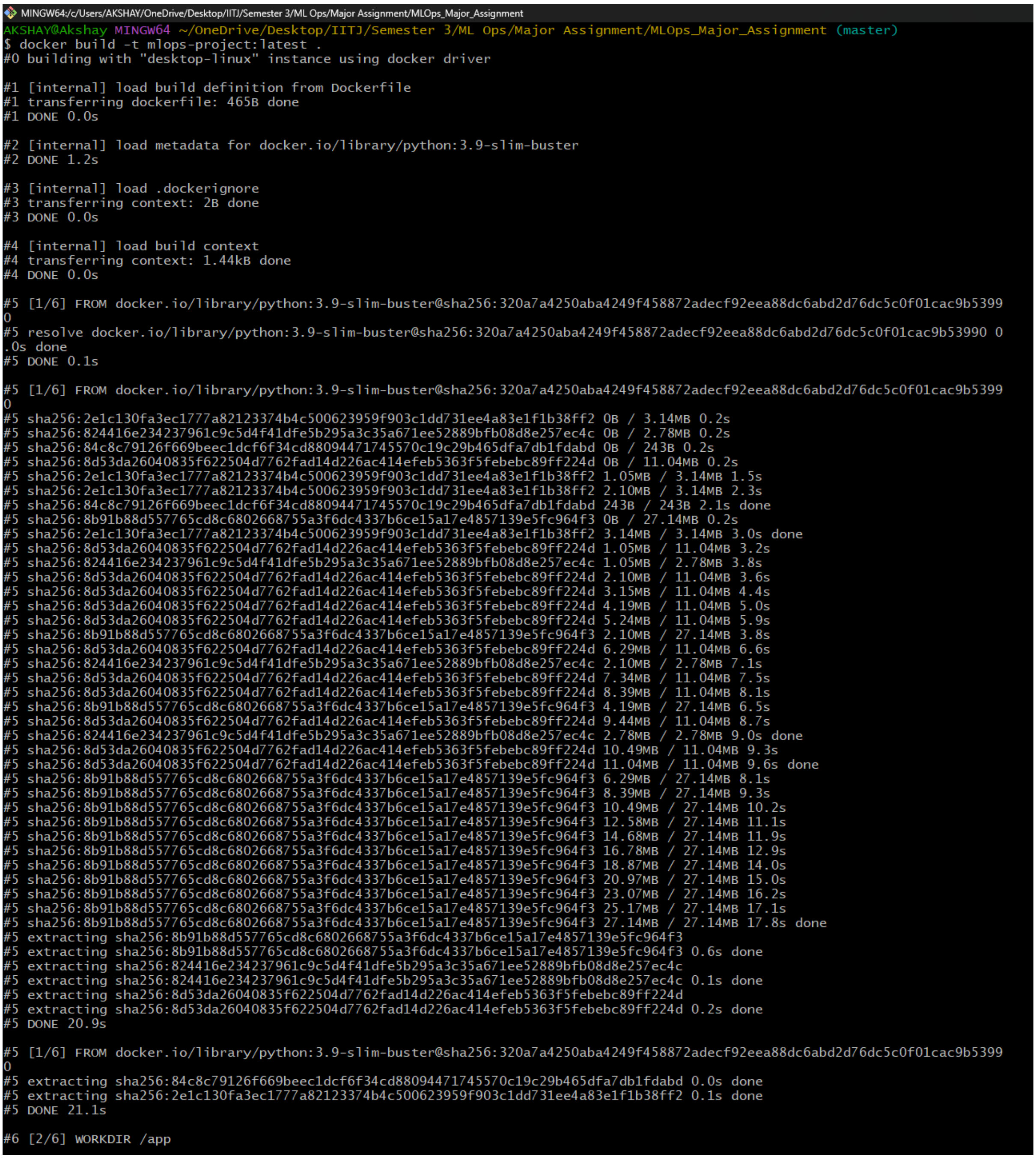
Step 5: Running python -m src.train from our project root to ensure models/linear_regression_model.joblib is present.



```
(mlops_env)
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ python -m src.train
Loading data...
Training Linear Regression model...
Evaluating model...
R^2 Score: 0.5758
Mean Squared Error (Loss): 0.5559
Model saved to models\linear_regression_model.joblib
```

Step 6: Build the Docker Image

GitBash Command : docker build -t mlops-project:latest .



```
MINGW64:/c/Users/AKSHAY/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (master)
$ docker build -t mlops-project:latest .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 465B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/python:3.9-slim-buster
#2 DONE 1.2s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [internal] load build context
#4 transferring context: 1.44kB done
#4 DONE 0.0s

#5 [1/6] FROM docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c0f01cac9b5399
0
#5 resolve docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c0f01cac9b53990 0
.0s done
#5 DONE 0.1s

#5 [1/6] FROM docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c0f01cac9b5399
0
#5 sha256:2e1c130fa3ec1777a82123374b4c500623959f903c1dd731ee4a83e1f1b38ff2 0B / 3.14MB 0.2s
#5 sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c 0B / 2.78MB 0.2s
#5 sha256:84c8c79126f669beec1dcf6f34cd88094471745570c19c29b465dfa7db1fdabd 0B / 243B 0.2s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 0B / 11.04MB 0.2s
#5 sha256:2e1c130fa3ec1777a82123374b4c500623959f903c1dd731ee4a83e1f1b38ff2 1.05MB / 3.14MB 1.5s
#5 sha256:2e1c130fa3ec1777a82123374b4c500623959f903c1dd731ee4a83e1f1b38ff2 2.10MB / 3.14MB 2.3s
#5 sha256:84c8c79126f669beec1dcf6f34cd88094471745570c19c29b465dfa7db1fdabd 243B / 243B 2.1s done
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 0B / 27.14MB 0.2s
#5 sha256:2e1c130fa3ec1777a82123374b4c500623959f903c1dd731ee4a83e1f1b38ff2 3.14MB / 3.14MB 3.0s done
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 1.05MB / 11.04MB 3.2s
#5 sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c 1.05MB / 2.78MB 3.8s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 2.10MB / 11.04MB 3.6s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 3.15MB / 11.04MB 4.4s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 4.19MB / 11.04MB 5.0s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 5.24MB / 11.04MB 5.9s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 2.10MB / 27.14MB 3.8s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 6.29MB / 11.04MB 6.6s
#5 sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c 2.10MB / 2.78MB 7.1s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 7.34MB / 11.04MB 7.5s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 8.39MB / 11.04MB 8.1s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 4.19MB / 27.14MB 6.5s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 9.44MB / 11.04MB 8.7s
#5 sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c 2.78MB / 2.78MB 9.0s done
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 10.49MB / 11.04MB 9.3s
#5 sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 11.04MB / 11.04MB 9.6s done
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 6.29MB / 27.14MB 8.1s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 8.39MB / 27.14MB 9.3s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 10.49MB / 27.14MB 10.2s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 12.58MB / 27.14MB 11.1s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 14.68MB / 27.14MB 11.9s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 16.78MB / 27.14MB 12.9s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 18.87MB / 27.14MB 14.0s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 20.97MB / 27.14MB 15.0s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 23.07MB / 27.14MB 16.2s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 25.17MB / 27.14MB 17.1s
#5 sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 27.14MB / 27.14MB 17.8s done
#5 extracting sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3
#5 extracting sha256:8b91b88d557765cd8c6802668755a3f6dc4337b6ce15a17e4857139e5fc964f3 0.6s done
#5 extracting sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c
#5 extracting sha256:824416e234237961c9c5d4f41dfe5b295a3c35a671ee52889bfb08d8e257ec4c 0.1s done
#5 extracting sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d
#5 extracting sha256:8d53da26040835f622504d7762fad14d226ac414efeb5363f5febebc89ff224d 0.2s done
#5 DONE 20.9s

#5 [1/6] FROM docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c0f01cac9b5399
0
#5 extracting sha256:84c8c79126f669beec1dcf6f34cd88094471745570c19c29b465dfa7db1fdabd 0.0s done
#5 extracting sha256:2e1c130fa3ec1777a82123374b4c500623959f903c1dd731ee4a83e1f1b38ff2 0.1s done
#5 DONE 21.1s

#6 [2/6] WORKDIR /app
```

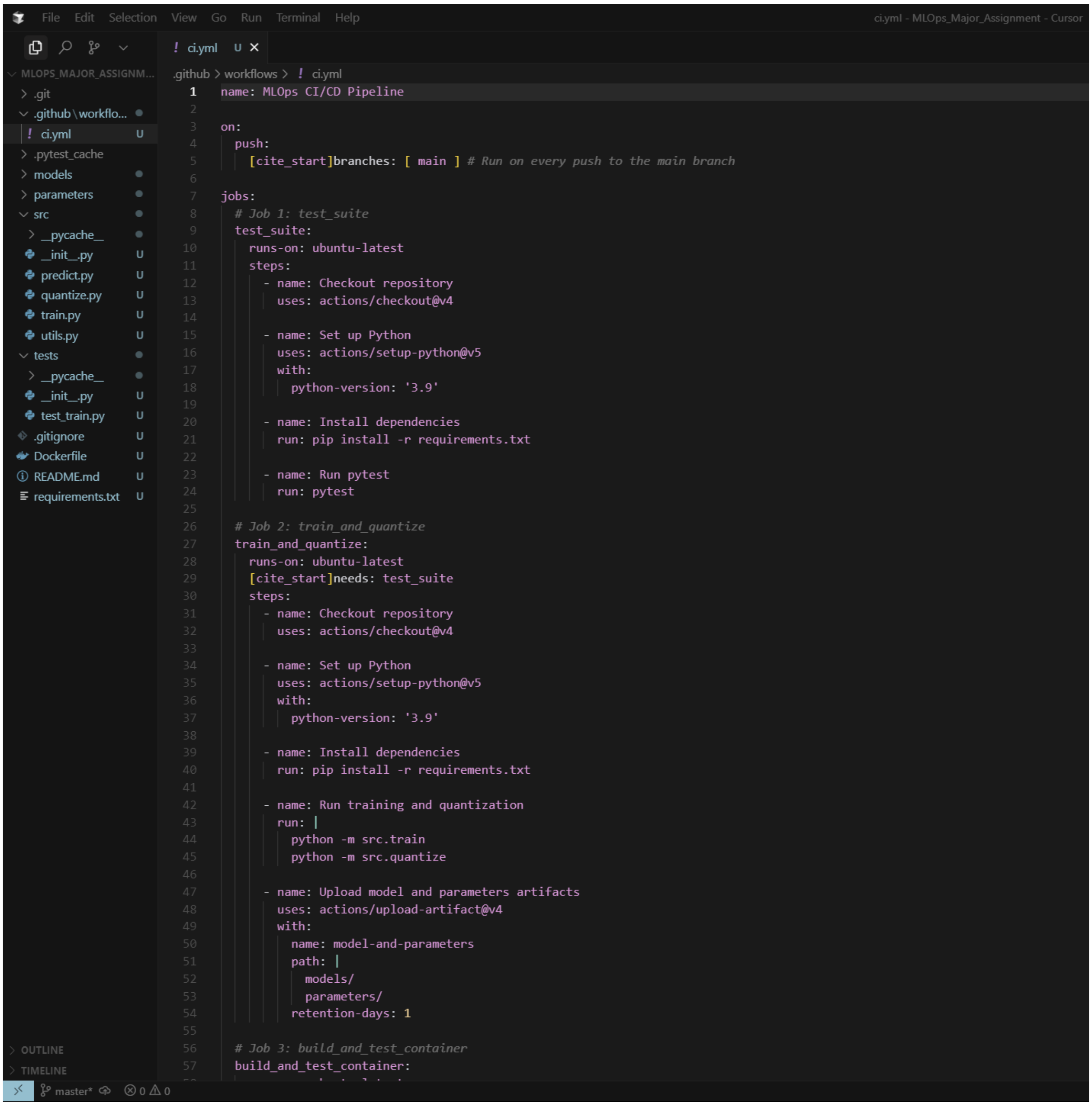
Phase 6 : CI/CD Workflow (.github/workflows/ci.yml)

Step 1: Create the directories and file

GitBash Command: mkdir -p .github/workflows

touch .github/workflows/ci.yml

Step 2: Add the YAML code to ci.yml



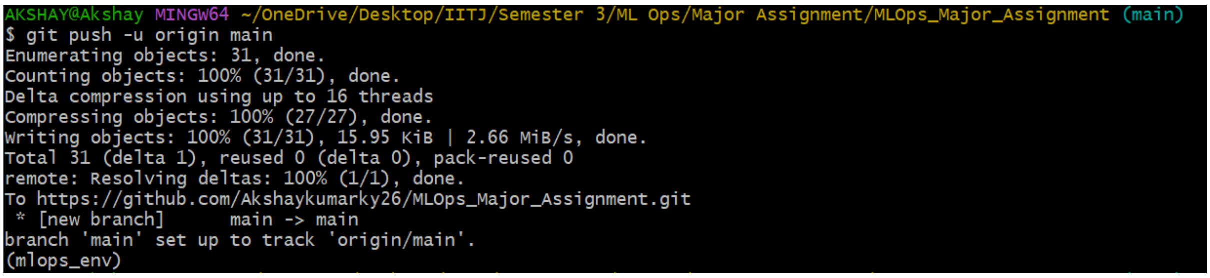
```yaml
name: MLOps CI/CD Pipeline

on:
  push:
    [cite_start]branches: [ main ] # Run on every push to the main branch

jobs:
  # Job 1: test_suite
  test_suite:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.9'

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

      - name: Run pytest
        run: pytest

  # Job 2: train_and_quantize
  train_and_quantize:
    runs-on: ubuntu-latest
    [cite_start]needs: test_suite
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.9'

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

      - name: Run training and quantization
        run: |
          python -m src.train
          python -m src.quantize

      - name: Upload model and parameters artifacts
        uses: actions/upload-artifact@v4
        with:
          name: model-and-parameters
          path: |
            models/
            parameters/
          retention-days: 1

  # Job 3: build_and_test_container
  build_and_test_container:
```

Step 3: Commit and Push to GitHub

GitBash Command: git add .

git commit -m "feat: Add CI/CD workflow, Dockerfile, and predict.py"

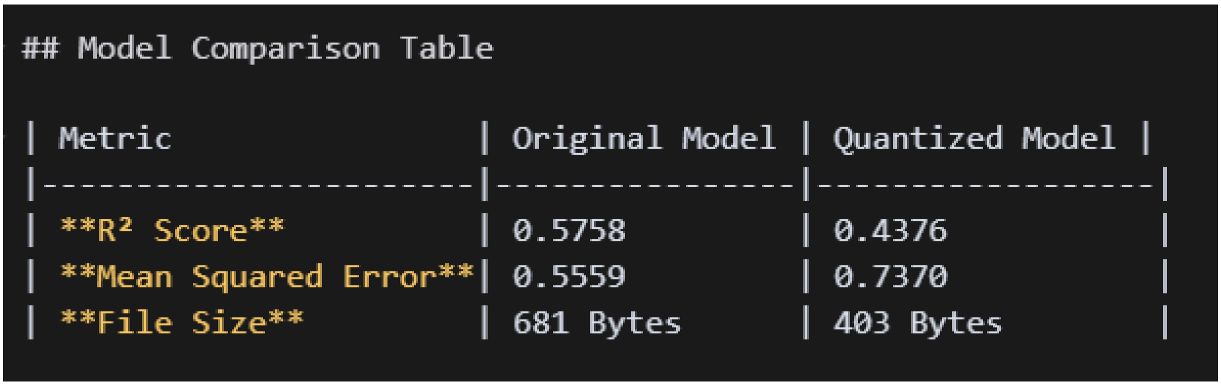
git push origin main



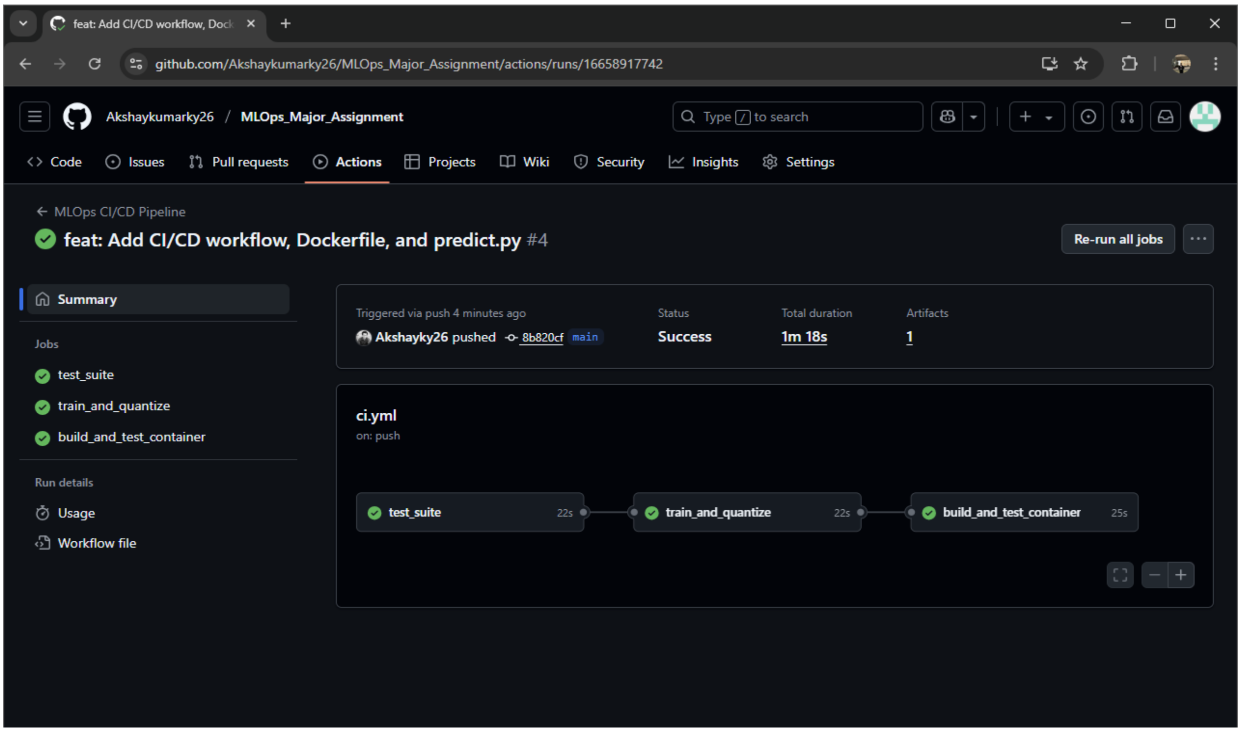
```
AKSHAY@Akshay MINGW64 ~/OneDrive/Desktop/IITJ/Semester 3/ML Ops/Major Assignment/MLOps_Major_Assignment (main)
$ git push -u origin main
Enumerating objects: 31, done.
Counting objects: 100% (31/31), done.
Delta compression using up to 16 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (31/31), 15.95 KiB | 2.66 MiB/s, done.
Total 31 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Akshaykumarky26/MLOps_Major_Assignment.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(mlops_env)
```

CONCLUSION

We performed quantization-aware regression using 8-bit integers and preserved the intercept in float precision. After trying multiple quantization strategies (global, per-coefficient, symmetric scaling), we found that global symmetric quantization with clipped small coefficients gave the best balance between model size and accuracy. The final quantized model achieved an R² score of 0.4376 (original: 0.5758), with acceptable prediction quality.



```
## Model Comparison Table

| Metric                 | Original Model | Quantized Model |
|------------------------|----------------|-----------------|
| **R² Score**           | 0.5758         | 0.4376          |
| **Mean Squared Error** | 0.5559         | 0.7370          |
| **File Size**          | 681 Bytes      | 403 Bytes       |
```



feat: Add CI/CD workflow, Dockerfile, and predict.py #4

Jobs: test_suite, train_and_quantize, build_and_test_container

Triggered via push 4 minutes ago
Akshayky26 pushed 8b820cf main

Status: Success

Total duration: 1m 18s

Artifacts: 1

ci.yml
on: push

test_suite 22s → train_and_quantize 22s → build_and_test_container 25s