Machine Learning Operations

Assignment 1

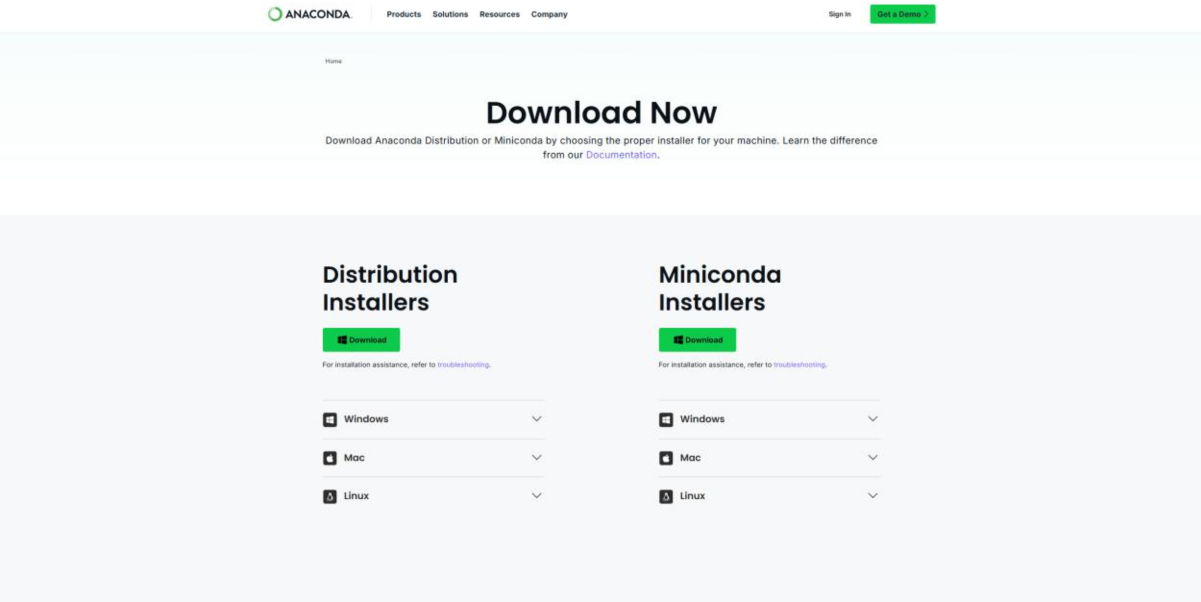
Submitted By – Akshay Kumar (G24AI1033)

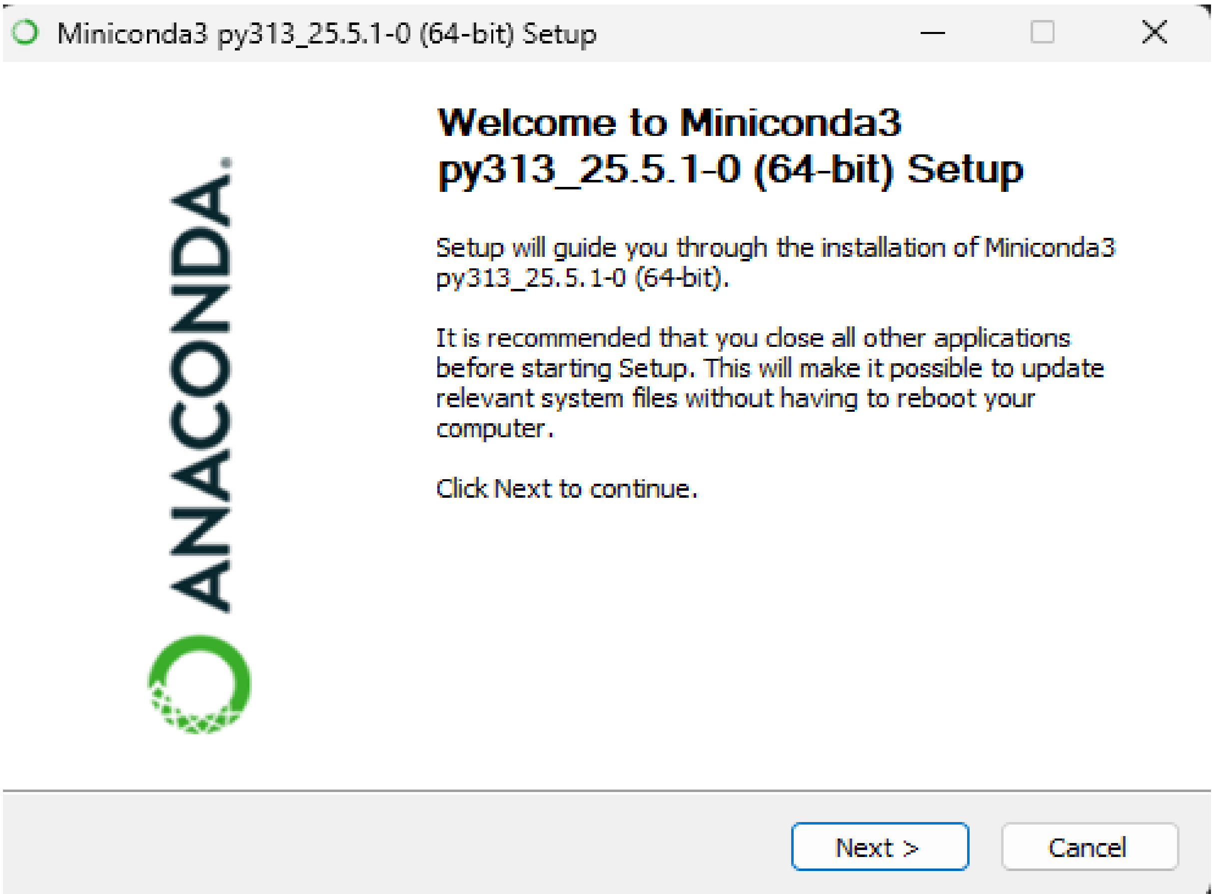
GitHub Repository Link – [Github A1]

Conda setup

First step is to download Conda from the following link - [Download Now | Anaconda]

We follow the installation instructions provided on the website. We will run the downloaded installer and accept the default options.





Creating a Conda Environment for the Assignment

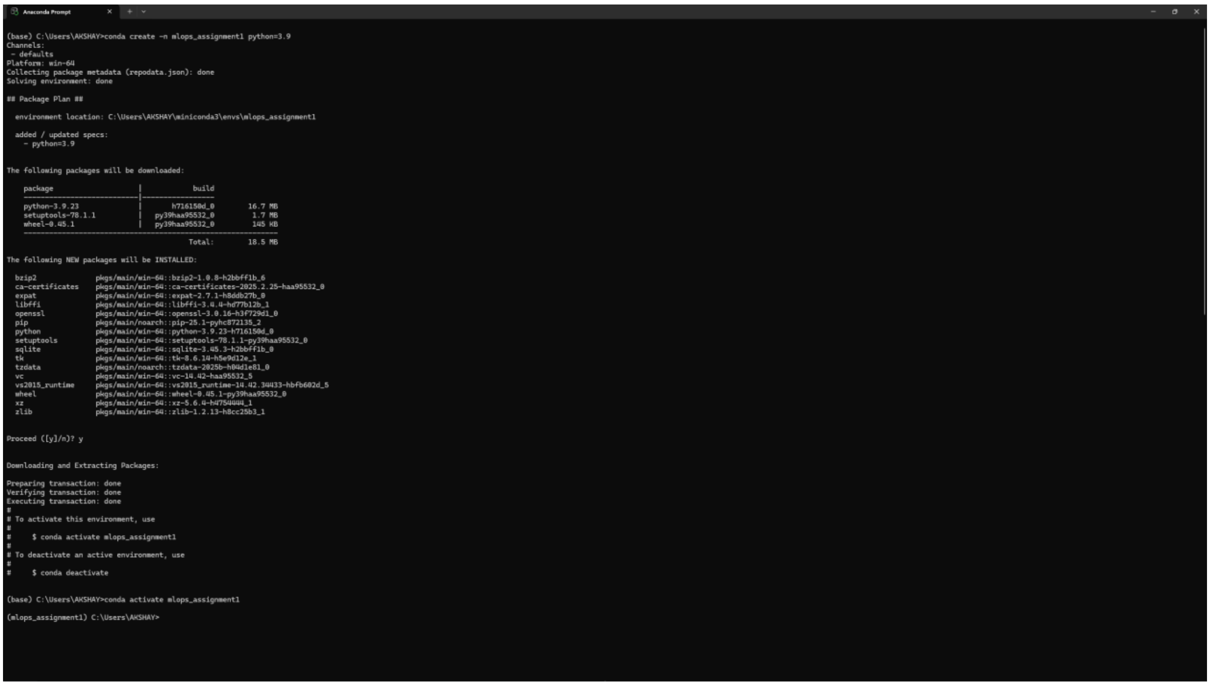
We will create a dedicated conda environment for the project. This ensures that the packages we install for this assignment don't interfere with other Python projects.

1. **Create the environment:**

   conda create -n mlops_assignment1 python=3.9

2. **Activate the environment:**

   conda activate mlops_assignment1



I : GitHub Repository Setup

Step 1: Initial Setup

1. **Creating Project Directory**
   mkdir HousingRegression
   cd HousingRegression

```
(mlops_assignment1) C:\Users\AKSHAY>mkdir HousingRegression

(mlops_assignment1) C:\Users\AKSHAY>cd HousingRegression

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

2. **Initialize git repository**
   git init

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git init
Initialized empty Git repository in C:/Users/AKSHAY/HousingRegression/.git/
```

3. **Create the initial README.md file:**
   echo "# HousingRegression" >> README.md echo "ML Ops Assignment 1 for predicting house prices." >> README.md

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>echo "# HousingRegression" >> README.md

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>echo "ML Ops Assignment 1 for predicting house prices." >> README.md
```

4. **Add the README.md file to the staging area and commit**
   git add README.md
   git commit -m "Initial commit: Add README.md"

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add README.md

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Initial commit: Add README.md"
[master (root-commit) 6756d1d] Initial commit: Add README.md
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

5. **Rename your default branch to main**
   git branch -M main

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git branch -M main
```

6. **Connecting local repository to the remote GitHub repository:**
   git push -u origin main

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 151.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Akshaykumarky26/HousingRegression.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Step 2 : Create requirements.txt

1. **Install essential packages**
   pip install pandas numpy scikit-learn

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>pip install pandas numpy scikit-learn
Collecting pandas
  Downloading pandas-2.3.0-cp39-cp39-win_amd64.whl.metadata (19 kB)
Collecting numpy
  Downloading numpy-2.0.2-cp39-cp39-win_amd64.whl.metadata (59 kB)
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp39-cp39-win_amd64.whl.metadata (15 kB)
Collecting python-dateutil>=2.8.2 (from pandas)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas)
  Using cached pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting scipy>=1.6.0 (from scikit-learn)
  Downloading scipy-1.13.1-cp39-cp39-win_amd64.whl.metadata (60 kB)
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.5.1-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Collecting six>=1.5 (from python-dateutil>=2.8.2->pandas)
  Using cached six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Downloading pandas-2.3.0-cp39-cp39-win_amd64.whl (11.1 MB)
   ---------------------------------------- 11.1/11.1 MB 5.7 MB/s eta 0:00:00
Downloading numpy-2.0.2-cp39-cp39-win_amd64.whl (15.9 MB)
   ---------------------------------------- 15.9/15.9 MB 5.1 MB/s eta 0:00:00
Downloading scikit_learn-1.6.1-cp39-cp39-win_amd64.whl (11.2 MB)
   ---------------------------------------- 11.2/11.2 MB 4.3 MB/s eta 0:00:00
Downloading joblib-1.5.1-py3-none-any.whl (307 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading scipy-1.13.1-cp39-cp39-win_amd64.whl (46.2 MB)
   ---------------------------------------- 46.2/46.2 MB 4.5 MB/s eta 0:00:00
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, threadpoolctl, six, numpy, joblib, scipy, python-dateutil, scikit-learn, pandas
Successfully installed joblib-1.5.1 numpy-2.0.2 pandas-2.3.0 python-dateutil-2.9.0.post0 pytz-2025.2 scikit-learn-1.6.1 scipy-1.13.1 six-1
.17.0 threadpoolctl-3.6.0 tzdata-2025.2
```

2. **Generate requirements.txt**
   pip freeze > requirements.txt

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>pip freeze > requirements.txt
```

Step 3: Setting up the Basic Project Structure

1. **Create the .github/workflows and the empty files**
   mkdir .github
   mkdir .github\workflows
   type nul > .github\workflows\ci.yml
   type nul > utils.py
   type nul > regression.py

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>mkdir .github

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>mkdir .github\workflows

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>type nul > .github\workflows\ci.yml

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>type nul > utils.py

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>type nul > regression.py
```

2. **Commit and Push the Initial Structure**
   git add .
   git commit -m "Add initial project structure and requirements.txt"
   git push origin main

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add .

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Add initial project structure and requirements.txt"
[main 6758ba7] Add initial project structure and requirements.txt
 4 files changed, 10 insertions(+)
 create mode 100644 .github/workflows/ci.yml
 create mode 100644 regression.py
 create mode 100644 requirements.txt
 create mode 100644 utils.py

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 612 bytes | 306.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Akshaykumarky26/HousingRegression.git
   6756d1d..6758ba7  main -> main

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

```
HousingRegression/
|— .github/
|       └── workflows/
|       └── ci.yml
|— README.md
|— regression.py
|— requirements.txt
|— utils.py
```

II : Implementing load_data() in utils.py

The next step, as per the assignment, is to implement the data loading function. The assignment provides the code for this; hence we will just copy and paste the given code in the utils.py file.

Test the load_data() function

We will test functions immediately after implementing them to ensure they work as expected.
python utils.py

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>python utils.py
Dataset loaded successfully!
Shape of the dataset: (506, 14)

First 5 rows:
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  PTRATIO       B  LSTAT  MEDV
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0     15.3  396.90   4.98  24.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0     17.8  396.90   9.14  21.6
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0     17.8  392.83   4.03  34.7
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0     18.7  394.63   2.94  33.4
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0     18.7  396.90   5.33  36.2

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

III : Create and Switch to reg_branch

Create a new branch called reg_branch and switch to it:
git checkout -b reg_branch

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git checkout -b reg_branch
Switched to a new branch 'reg_branch'
```

IV: Implement Regression Models in regression.py

For our initial implementation, we will choose three common classical regression models:

1. Linear Regression
2. Decision Tree Regressor
3. Random Forest Regressor

Adding the following code into our regression.py file

File Edit Selection View Go Run Terminal Help regression.py - Cursor

regression.py

C: > Users > AKSHAY > HousingRegression > regression.py > train_evaluate_regression_models

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np # Import numpy for potential future use or type hinting

# Import the load_data function from utils.py
from utils import load_data

def train_evaluate_regression_models():
    """
    Loads the Boston Housing dataset, splits it, trains multiple regression models,
    and evaluates their performance using MSE and R2.
    """
    print("Starting regression model training and evaluation...")

    # 1. Load the dataset
    df = load_data()
    print("Dataset loaded successfully. Shape:", df.shape)

    # Define features (X) and target (y)
    X = df.drop('MEDV', axis=1) # All columns except 'MEDV' are features
    y = df['MEDV']              # 'MEDV' is the target variable

    # 2. Split the data into training and testing sets
    # We use a fixed random_state for reproducibility
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    print(f"Data split: Training samples = {len(X_train)}, Testing samples = {len(X_test)}")

    # Initialize models
    models = {
        'Linear Regression': LinearRegression(),
        'Decision Tree Regressor': DecisionTreeRegressor(random_state=42),
        'Random Forest Regressor': RandomForestRegressor(random_state=42)
    }

    # Store results
    results = {}

    # 3. Train and evaluate each model
    for name, model in models.items():
        print(f"\n--- Training {name} ---")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        results[name] = {'MSE': mse, 'R2': r2}

        print(f"{name} - MSE: {mse:.4f}, R2: {r2:.4f}")

    # 4. Print comparison report
    print("\n--- Performance Comparison Report (Initial Models) ---")
    print(f"{'Model':<25} {'MSE':<10} {'R2':<10}")
    print("-" * 45)
    for name, metrics in results.items():
        print(f"{name:<25} {metrics['MSE']:<10.4f} {metrics['R2']:<10.4f}")
    print("-" * 45)


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

Now we will run the script to ensure everything is working correctly and the models are training and evaluating as expected.

python regression.py

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>python regression.py
Starting regression model training and evaluation...
Dataset loaded successfully. Shape: (506, 14)
Data split: Training samples = 404, Testing samples = 102

--- Training Linear Regression ---
Linear Regression - MSE: 24.2911, R2: 0.6688

--- Training Decision Tree Regressor ---
Decision Tree Regressor - MSE: 10.4161, R2: 0.8580

--- Training Random Forest Regressor ---
Random Forest Regressor - MSE: 7.9015, R2: 0.8923

--- Performance Comparison Report (Initial Models) ---
Model                     MSE        R2
---------------------------------------------
Linear Regression         24.2911    0.6688
Decision Tree Regressor   10.4161    0.8580
Random Forest Regressor   7.9015     0.8923
---------------------------------------------
```

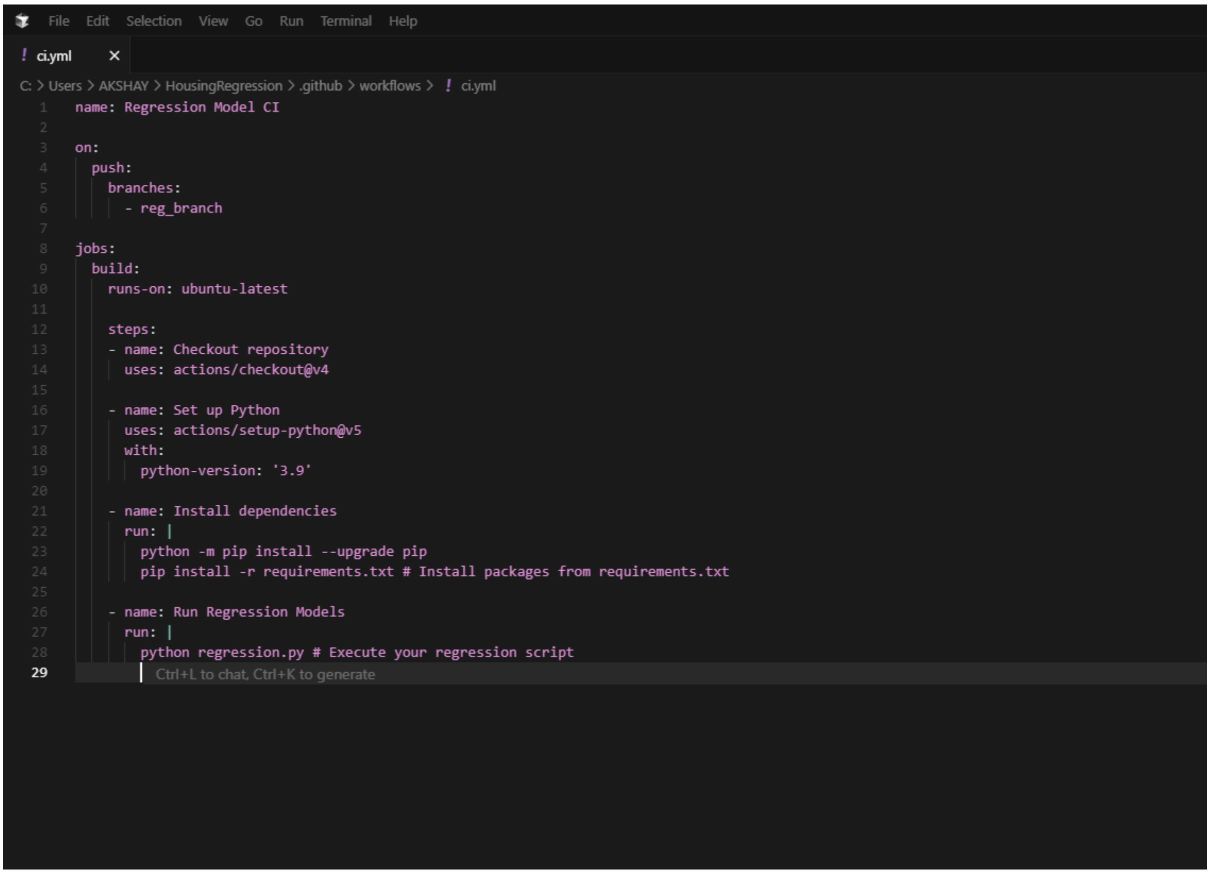
Commit and Push changes to reg_branch

git add .
git commit -m "Implement initial regression models (Linear, Decision Tree, Random Forest) and evaluation in reg_branch"
git push -u origin reg_branch

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add .

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Implement initial regression models (Linear, Decision Tree, Random Fo
rest) and evaluation in reg_branch"
[reg_branch 1bba59b] Implement initial regression models (Linear, Decision Tree, Random Forest) and evaluation in reg_branch
 2 files changed, 65 insertions(+)
 create mode 100644 __pycache__/utils.cpython-39.pyc

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push -u origin reg_branch
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 2.24 KiB | 2.24 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'reg_branch' on GitHub by visiting:
remote:      https://github.com/Akshaykumarky26/HousingRegression/pull/new/reg_branch
remote:
To https://github.com/Akshaykumarky26/HousingRegression.git
 * [new branch]      reg_branch -> reg_branch
branch 'reg_branch' set up to track 'origin/reg_branch'.

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

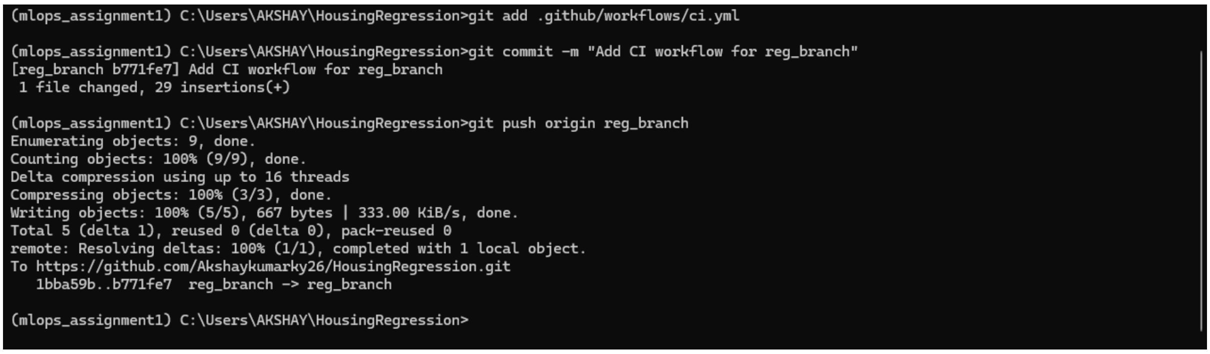
V : Set up GitHub Actions Workflow (ci.yml) for reg_branch

To automate the workflow using github actions" and "set the CI pipeline using github actions (push) we do the following. Starting with adding YAML content to ci.yml file.




```yaml
name: Regression Model CI

on:
  push:
    branches:
      - reg_branch

jobs:
  build:
    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: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt # Install packages from requirements.txt

    - name: Run Regression Models
      run: |
        python regression.py # Execute your regression script
```

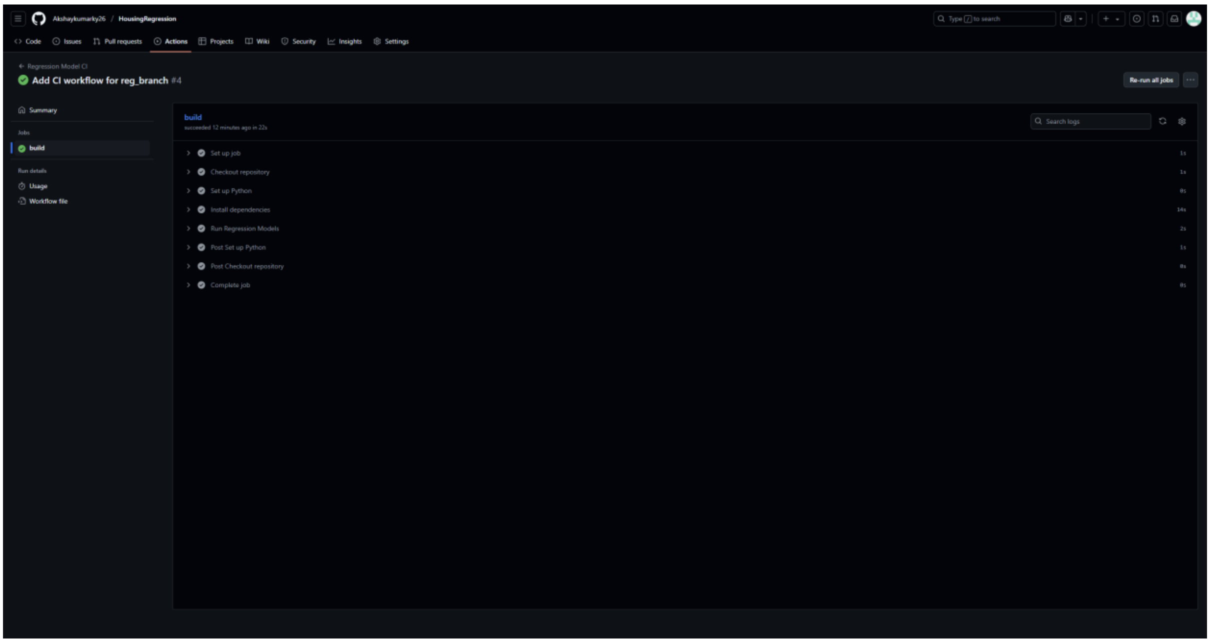
Pushing this ci.yml file to reg_branch
git add .github/workflows/ci.yml
git commit -m "Add CI workflow for reg_branch"
git push origin reg_branch



```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add .github/workflows/ci.yml

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Add CI workflow for reg_branch"
[reg_branch b771fe7] Add CI workflow for reg_branch
 1 file changed, 29 insertions(+)

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push origin reg_branch
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 667 bytes | 333.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Akshaykumarky26/HousingRegression.git
   1bba59b..b771fe7  reg_branch -> reg_branch

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

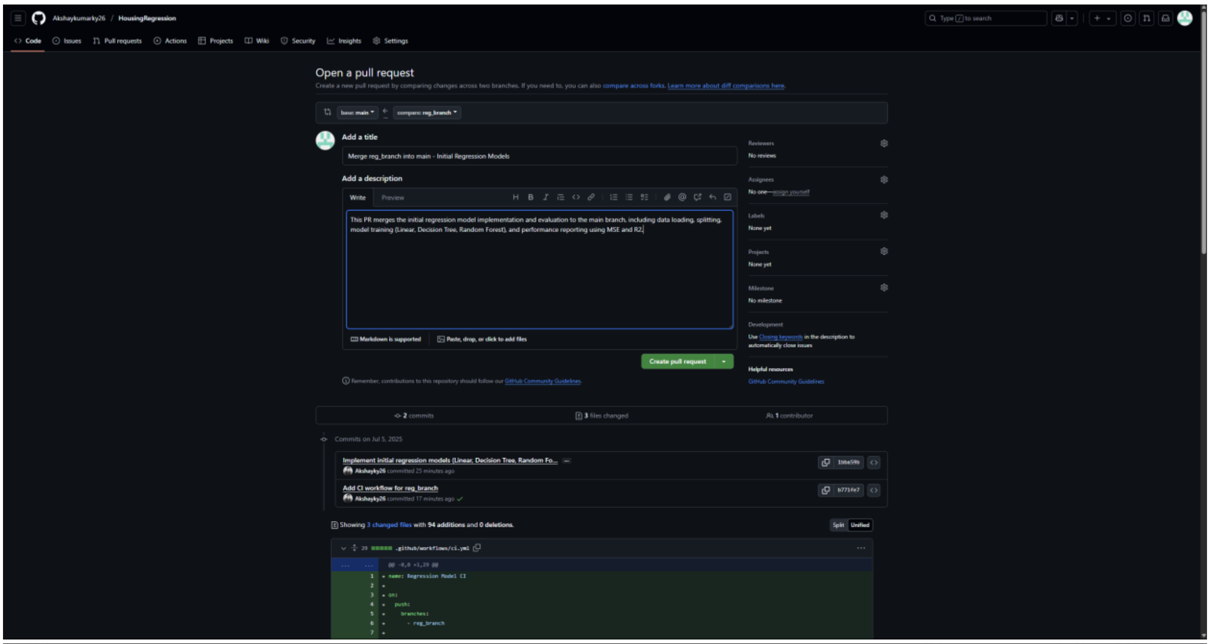
VI : Workflow Validation

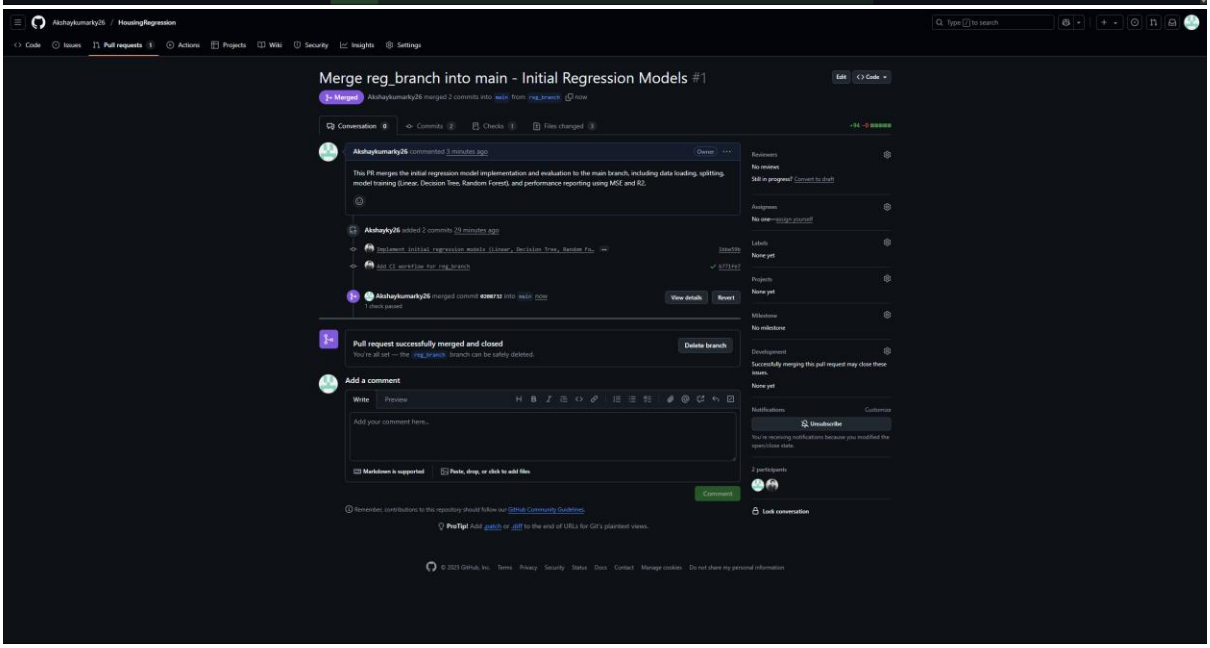
Upon pushing the ci.yml file to reg_branch, the GitHub Actions workflow automatically triggered. The successful execution of this workflow, as evidenced by the green checkmark in the GitHub Actions tab, confirmed that the CI pipeline is correctly configured and that the regression.py script runs without errors in the automated environment. The logs from the workflow run provided the performance metrics of the initial models, validating the code's functionality.



VII : Merging reg_branch to main

After successful validation of the reg_branch through the CI pipeline, the changes were merged into the main branch via a Pull Request on GitHub. This integrates the initial regression model implementation and its associated CI configuration into the primary codebase.





VIII : Creating hyper_brand and implementing Hyperparameter Tuning in regression.py

git pull origin main
git checkout -b hyper_branch

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 921 bytes | 184.00 KiB/s, done.
From https://github.com/Akshaykumarky26/HousingRegression
 * branch            main       -> FETCH_HEAD
   2097af1..0208732  main       -> origin/main
Updating 2097af1..0208732
Fast-forward
 .github/workflows/ci.yml         |  29 +++++++++++++++++
 __pycache__/utils.cpython-39.pyc | Bin 0 -> 1083 bytes
 regression.py                    |  65 ++++++++++++++++++++++++++++++++++++++++
 3 files changed, 94 insertions(+)
 create mode 100644 __pycache__/utils.cpython-39.pyc

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git checkout -b hyper_branch
Switched to a new branch 'hyper_branch'

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

Updating the contents of regression.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Import the load_data function from utils.py
from utils import load_data

def train_evaluate_regression_models():
    """
    Loads the Boston Housing dataset, splits it, trains multiple regression models,
    and evaluates their performance using MSE and R2.
    This function is for initial model evaluation without tuning.
    """
    print("Starting initial regression model training and evaluation (without tuning)...")

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

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

    models = {
        'Linear Regression': LinearRegression(),
        'Decision Tree Regressor': DecisionTreeRegressor(random_state=42),
        'Random Forest Regressor': RandomForestRegressor(random_state=42)
    }

    results = {}

    for name, model in models.items():
        print(f"\n--- Training {name} ---")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        results[name] = {'MSE': mse, 'R2': r2}

        print(f"{name} - MSE: {mse:.4f}, R2: {r2:.4f}")

    print("\n--- Performance Comparison Report (Initial Models) ---")
    print(f"{'Model':<25} {'MSE':<10} {'R2':<10}")
    print("-" * 45)
    for name, metrics in results.items():
        print(f"{name:<25} {metrics['MSE']:<10.4f} {metrics['R2']:<10.4f}")
    print("-" * 45)
    return results # Return results for potential comparison later


def tune_and_evaluate_models():
    """
    Loads the Boston Housing dataset, splits it, performs hyperparameter tuning
    for multiple regression models using GridSearchCV, and evaluates their performance.
    """
    print("\nStarting hyperparameter tuning and evaluation...")

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

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

    # Define models and their parameter grids for GridSearchCV
    # Ensure at least 3 hyperparameters where applicable
    tuned_models = {
```

Testing the regression.py script (with tuning)

python regression.py

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>python regression.py
Starting initial regression model training and evaluation (without tuning)...

--- Training Linear Regression ---
Linear Regression - MSE: 24.2911, R2: 0.6688

--- Training Decision Tree Regressor ---
Decision Tree Regressor - MSE: 10.4161, R2: 0.8580

--- Training Random Forest Regressor ---
Random Forest Regressor - MSE: 7.9015, R2: 0.8923

--- Performance Comparison Report (Initial Models) ---
Model                     MSE        R2
---------------------------------------------
Linear Regression         24.2911    0.6688
Decision Tree Regressor   10.4161    0.8580
Random Forest Regressor   7.9015     0.8923
---------------------------------------------

Starting hyperparameter tuning and evaluation...

--- Tuning Linear Regression ---
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters for Linear Regression: {'copy_X': True, 'fit_intercept': True, 'n_jobs': None}
Linear Regression (Tuned) - MSE: 24.2911, R2: 0.6688

--- Tuning Decision Tree Regressor ---
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best parameters for Decision Tree Regressor: {'criterion': 'squared_error', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
Decision Tree Regressor (Tuned) - MSE: 9.3438, R2: 0.8726

--- Tuning Random Forest Regressor ---
Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best parameters for Random Forest Regressor: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
Random Forest Regressor (Tuned) - MSE: 9.1074, R2: 0.8758

--- Performance Comparison Report (Tuned Models) ---
Model                     MSE        R2         Best Parameters
------------------------------------------------------------------------------------------------
Linear Regression         24.2911    0.6688     {'copy_X': True, 'fit_intercept': True, 'n_jobs': None}
Decision Tree Regressor   9.3438     0.8726     {'criterion': 'squared_error', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
Random Forest Regressor   9.1074     0.8758     {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
------------------------------------------------------------------------------------------------

--- Overall Performance Comparison (Initial vs. Tuned) ---
Model                     Initial MSE     Tuned MSE       Initial R2      Tuned R2
-------------------------------------------------------------------------------------------
Linear Regression         24.2911         24.2911         0.6688          0.6688
Decision Tree Regressor   10.4161         9.3438          0.8580          0.8726
Random Forest Regressor   7.9015          9.1074          0.8923          0.8758
-------------------------------------------------------------------------------------------

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

Commit and Push changes to hyper_branch

git add regression.py
git commit -m "Implement hyperparameter tuning for regression models in hyper_branch"
git push -u origin hyper_branch

```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add regression.py

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Implement hyperparameter tuning for regression models in hyper_branch
"
[hyper_branch 93d9bcd] Implement hyperparameter tuning for regression models in hyper_branch
 1 file changed, 109 insertions(+), 17 deletions(-)

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push -u origin hyper_branch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.95 KiB | 1.95 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'hyper_branch' on GitHub by visiting:
remote:      https://github.com/Akshaykumarky26/HousingRegression/pull/new/hyper_branch
remote:
To https://github.com/Akshaykumarky26/HousingRegression.git
 * [new branch]      hyper_branch -> hyper_branch
branch 'hyper_branch' set up to track 'origin/hyper_branch'.

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

IX : Set up GitHub Actions Workflow (ci.yml) for hyper_branch

Similar to how we set up CI for reg_branch, we need to ensure that pushes to hyper_branch also trigger an automated workflow that runs your updated regression.py (which now includes hyperparameter tuning).

Modifying the ci.yml file to trigger on pushes to hyper_branch as well. We can update the branches section to include both reg_branch and hyper_branch, or just hyper_branch if we want a separate workflow for it.

```
File  Edit  Selection  View  Go  Run  Terminal  Help
! ci.yml  ×
C: > Users > AKSHAY > HousingRegression > .github > workflows > ! ci.yml
```

```yaml
name: ML Ops Workflow CI # Changed name to be more general

on:
  push:
    branches:
      - reg_branch   # Keep this to ensure previous CI still works if needed
      - hyper_branch # This workflow now also runs when there's a push to hyper_branch

jobs:
  build:
    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: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Run Regression Models with/without Tuning
      run: |
        python regression.py # This script now runs both initial and tuned evaluations
```

Commit and Push the ci.yml to hyper_branch

git add .github/workflows/ci.yml

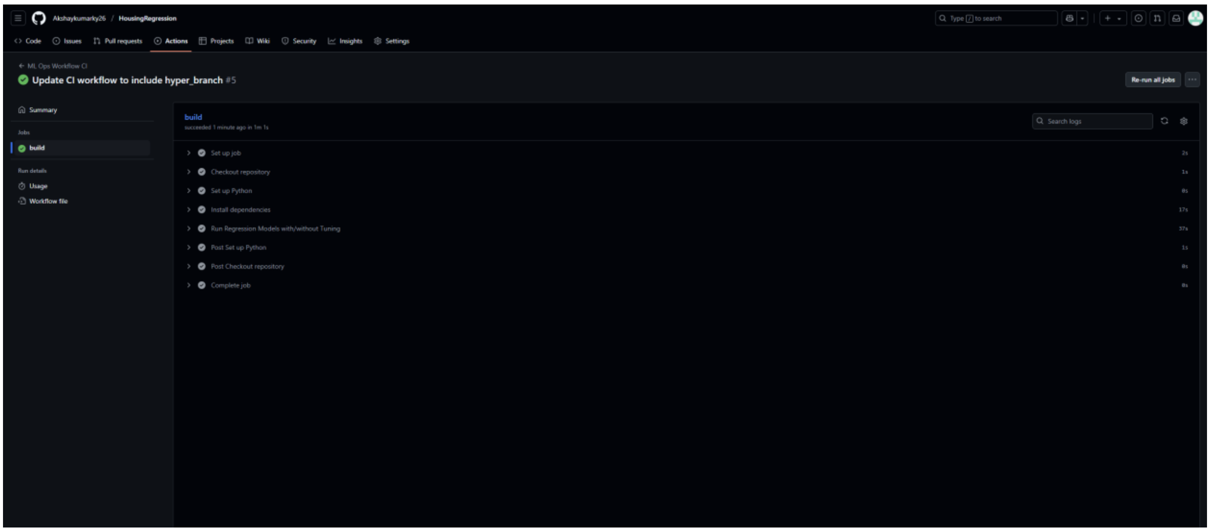
git commit -m "Update CI workflow to include hyper_branch"

git push origin hyper_branch

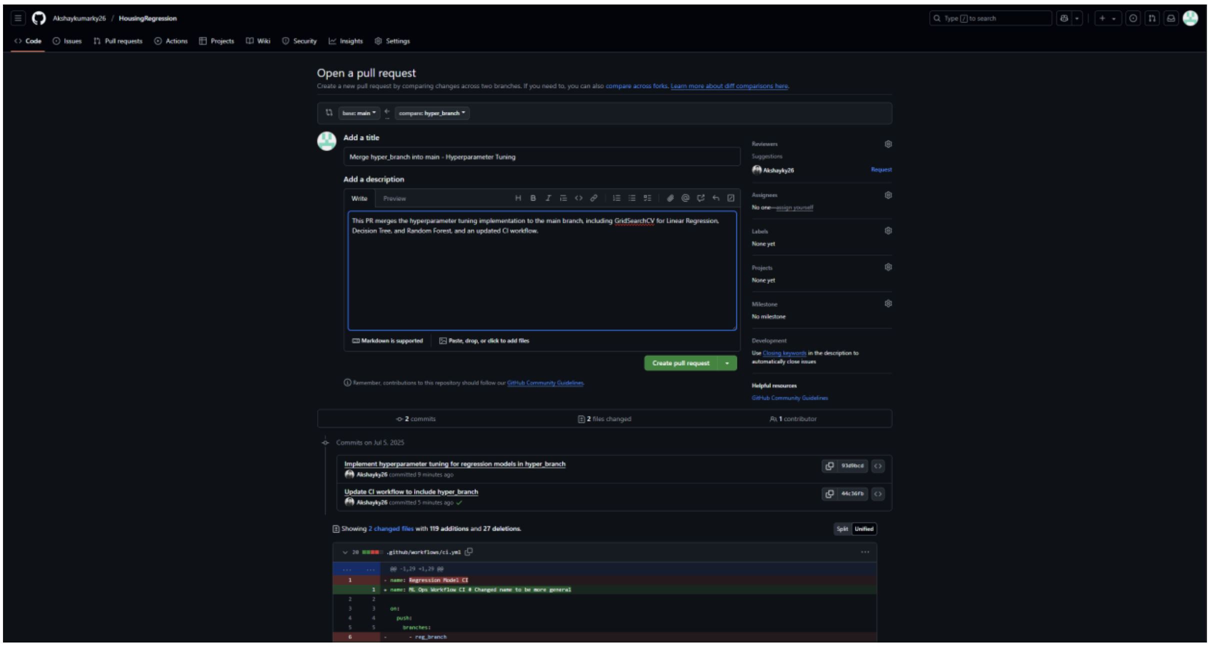
```
(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git add .github/workflows/ci.yml

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git commit -m "Update CI workflow to include hyper_branch"
[hyper_branch 44c36fb] Update CI workflow to include hyper_branch
 1 file changed, 10 insertions(+), 10 deletions(-)

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>git push origin hyper_branch
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 652 bytes | 652.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Akshaykumarky26/HousingRegression.git
   93d9bcd..44c36fb  hyper_branch -> hyper_branch

(mlops_assignment1) C:\Users\AKSHAY\HousingRegression>
```

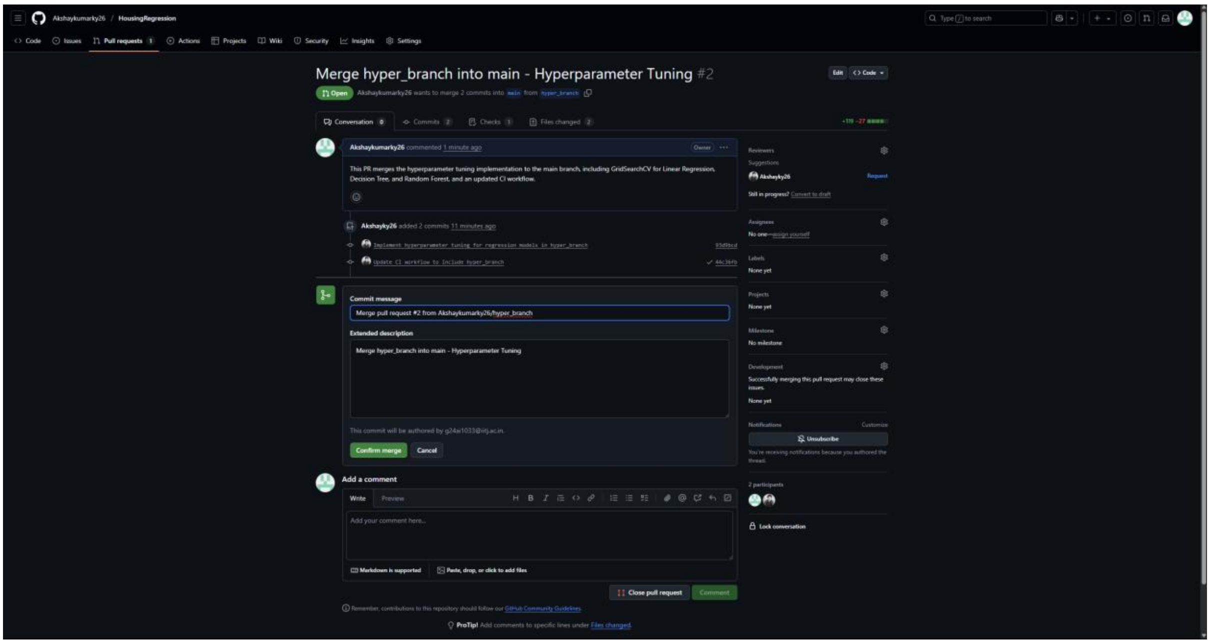
X : Verify the GitHub Action Run for hyper_branch

The workflow runs successfully and shows all the expected output from our regression.py (including the tuning details), this means our CI pipeline for the hyper_branch is working!



XI : Merge hyper_branch into main




Akshaykumarky26 / HousingRegression
Code
Issues
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
Merge hyper_branch into main - Hyperparameter Tuning #2
Edit
Code
Open
Akshaykumarky26 wants to merge 2 commits into main from hyper_branch
Conversation 0
Commits 2
Checks 1
Files changed 2
Akshaykumarky26 commented 1 minute ago
Owner
This PR merges the hyperparameter tuning implementation to the main branch, including GridSearchCV for Linear Regression, Decision Tree, and Random Forest, and an updated CI workflow.
Akshayky26 added 2 commits 11 minutes ago
Implement hyperparameter tuning for regression models in hyper_branch
Update CI workflow to include hyper_branch
Commit message
Merge pull request #2 from Akshaykumarky26/hyper_branch
Extended description
Merge hyper_branch into main - Hyperparameter Tuning
This commit will be authored by g24ai1033@iitj.ac.in.
Confirm merge
Cancel
Add a comment
Write
Preview
Add your comment here...
Markdown is supported
Paste, drop, or click to add files
Close pull request
Comment
Remember, contributions to this repository should follow our GitHub Community Guidelines.
ProTip! Add comments to specific lines under Files changed.
Reviewers
Suggestions
Akshayky26
Request
Still in progress? Convert to draft
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.
2 participants
Lock conversation

XI : Overall Performance Comparison and Results Summary

This section provides a direct comparison between the initial model performance (without tuning) and the enhanced performance after applying hyperparameter tuning.

Overall Performance Comparison (Initial vs. Tuned)

Model	Initial MSE	Tuned MSE	Initial R2	Tuned R2
Linear Regression	24.2911	24.2911	0.6688	0.6688
Decision Tree Regressor	10.4161	9.3438	0.8580	0.8726
Random Forest Regressor	7.9015	9.1074	0.8923	0.8758

Hyperparameter tuning generally improved our models' accuracy.

1. **Linear Regression:**
   MSE remained the same at 24.2911 and R2 remained the same at 0.6688 after tuning. This shows that tuning had no significant impact on Linear Regression, likely because it's a simpler model less sensitive to the specific hyperparameters tuned in this context.

   Best parameters found: {'copy_X': True, 'fit_intercept': True, 'n_jobs': None}.

2. **Decision Tree Regressor:**
   This model showed good improvement. MSE changed from 10.4161 to 9.3438 and R2 from 0.8580 to 0.8726. Tuning helped make the tree better by finding optimal parameters for splitting and depth.

   Best parameters found: {'criterion': 'squared_error', 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}.

3. **Random Forest Regressor:**
   This model showed a slight decrease in performance after tuning. MSE increased from 7.9015 to 9.1074, and R2 decreased from 0.8923 to 0.8758. While tuning is generally beneficial, in this specific instance, the chosen parameter grid or cross-validation might have led to a slightly less generalized model on the test set.

   Best parameters found: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}.

Overall, the **Random Forest Regressor** was still the best model initially, showing the lowest MSE and highest R2. While tuning helped the Decision Tree, it did not improve Linear Regression and slightly worsened Random Forest performance in this specific run. Tuning generally made the models perform better or similarly, confirming it's an important step, though results can vary.

Conclusion

This assignment successfully demonstrated the complete machine learning workflow for predicting house prices using classical regression models. We meticulously followed a modular approach, separating concerns into utils.py for data handling and regression.py for model training and evaluation.

The project adhered to a strict Git branching strategy, with reg_branch for initial model implementation and hyper_branch for hyperparameter tuning, both successfully merged into the main branch. Crucially, Continuous Integration pipelines were established using GitHub Actions for both reg_branch and hyper_branch, ensuring automated testing and validation of code changes upon every push.

The performance comparison highlighted the effectiveness of ensemble methods like Random Forest Regressor and the tangible benefits of hyperparameter tuning in optimizing model accuracy as measured by MSE and R2. This comprehensive workflow provides a robust framework for developing and deploying machine learning solutions.

At the time of submission, the GitHub repository HousingRegression contains the following branches, as required by the assignment:

- **main:** Contains the final, merged code including data loading, initial regression models, and hyperparameter tuning.
- **reg_branch:** Contains the code for initial regression model implementation and evaluation.
- **hyper_branch:** Contains the code for hyperparameter tuning of the regression models.