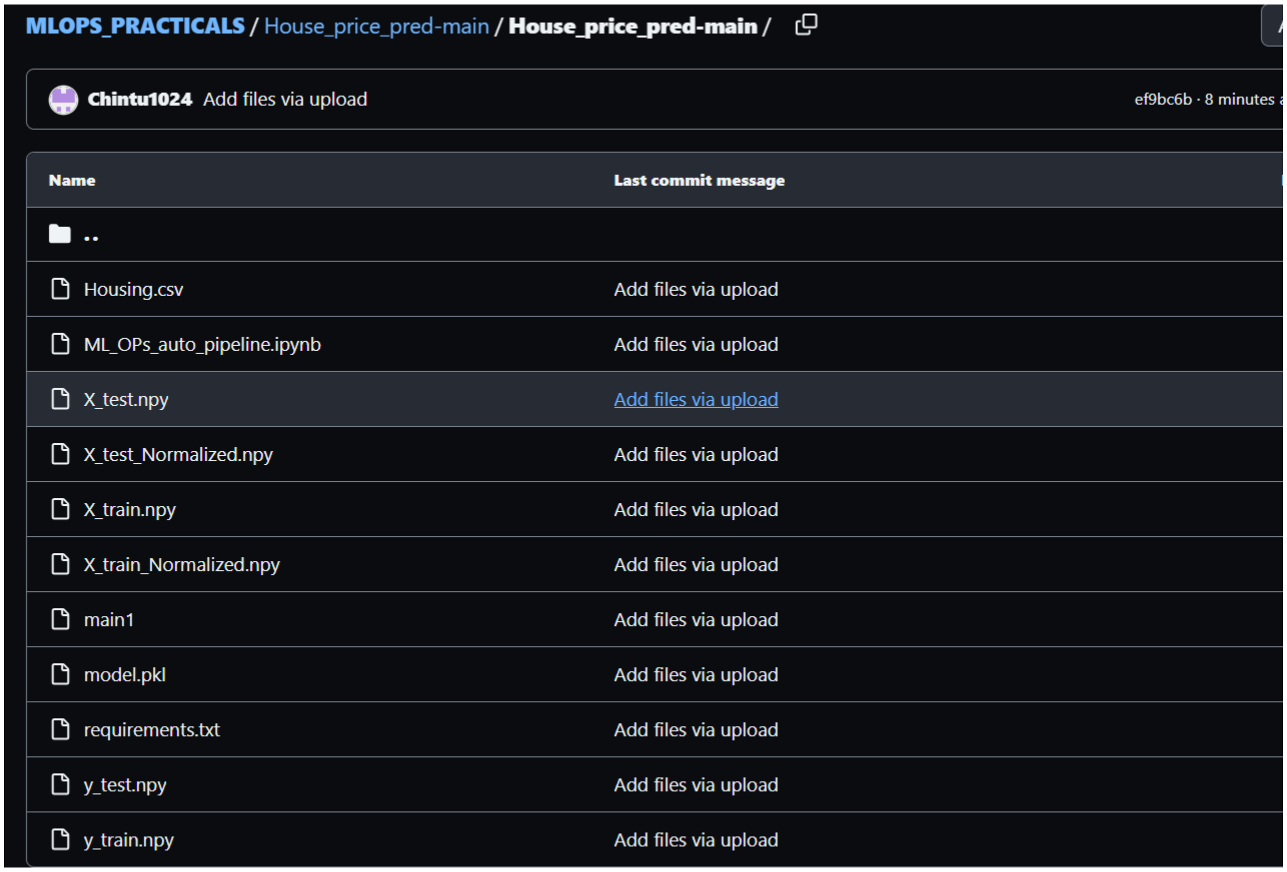
Practical-3

Generation of Reproducible and Interactive ML Project.

Task 1: Create the Github repository for the house rate prediction project created in practical 2.

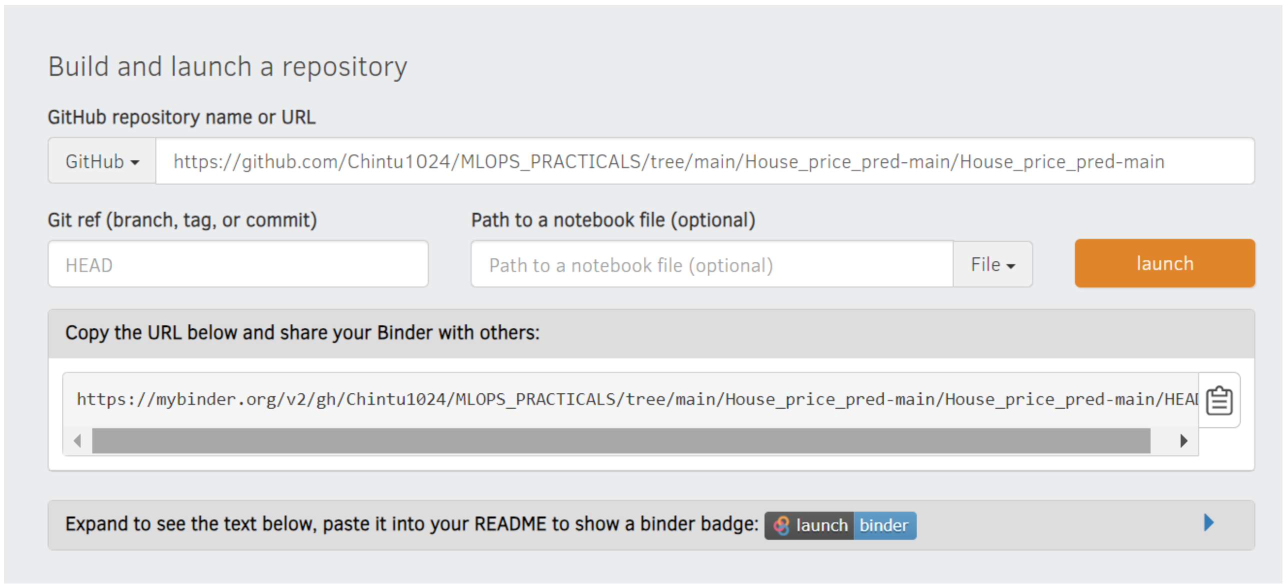


MLOPS_PRACTICALS / House_price_pred-main / House_price_pred-main /

Chintu1024 Add files via upload ef9bc6b · 8 minutes

Name	Last commit message
..	
Housing.csv	Add files via upload
ML_OPs_auto_pipeline.ipynb	Add files via upload
X_test.npy	Add files via upload
X_test_Normalized.npy	Add files via upload
X_train.npy	Add files via upload
X_train_Normalized.npy	Add files via upload
main1	Add files via upload
model.pkl	Add files via upload
requirements.txt	Add files via upload
y_test.npy	Add files via upload
y_train.npy	Add files via upload

Task 2: Integrate your repository with the binder to make your project interactive. (Hint: refer to the following link for the steps: (https://mybinder.org/)



Build and launch a repository

GitHub repository name or URL

GitHub https://github.com/Chintu1024/MLOPS_PRACTICALS/tree/main/House_price_pred-main/House_price_pred-main

Git ref (branch, tag, or commit): HEAD

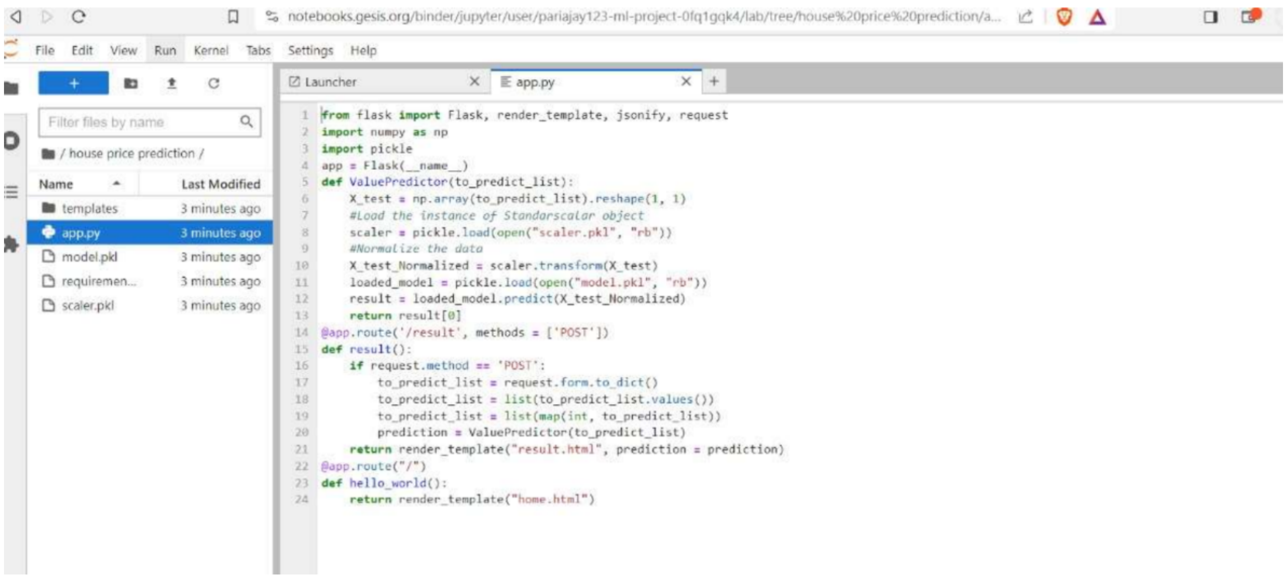
Path to a notebook file (optional): Path to a notebook file (optional) File

launch

Copy the URL below and share your Binder with others:

```
https://mybinder.org/v2/gh/Chintu1024/MLOPS_PRACTICALS/tree/main/House_price_pred-main/House_price_pred-main/HEAI
```

Expand to see the text below, paste it into your README to show a binder badge: launch binder



notebooks.gesis.org/binder/jupyter/user/pariajay123-ml-project-0fq1gqk4/lab/tree/house%20price%20prediction/a...

/ house price prediction /

```
Name           Last Modified
templates      3 minutes ago
app.py         3 minutes ago
model.pkl      3 minutes ago
requiremen...  3 minutes ago
scaler.pkl     3 minutes ago
```

app.py

```python
from flask import Flask, render_template, jsonify, request
import numpy as np
import pickle
app = Flask(__name__)
def ValuePredictor(to_predict_list):
    X_test = np.array(to_predict_list).reshape(1, 1)
    #Load the instance of Standarscalar object
    scaler = pickle.load(open("scaler.pkl", "rb"))
    #Normalize the data
    X_test_Normalized = scaler.transform(X_test)
    loaded_model = pickle.load(open("model.pkl", "rb"))
    result = loaded_model.predict(X_test_Normalized)
    return result[0]
@app.route('/result', methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        prediction = ValuePredictor(to_predict_list)
    return render_template("result.html", prediction = prediction)
@app.route("/")
def hello_world():
    return render_template("home.html")
```