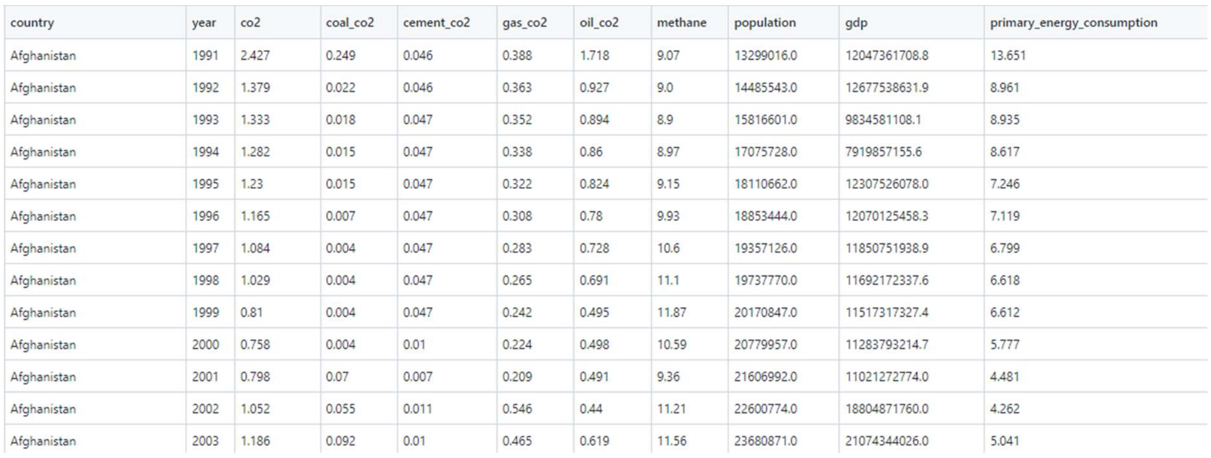
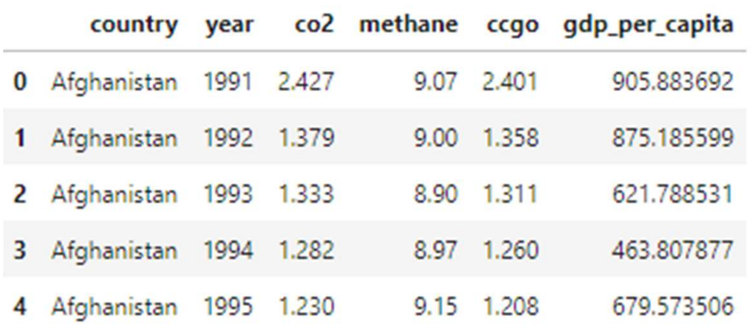
Data Overview before pre-processing:



country	year	co2	coal_co2	cement_co2	gas_co2	oil_co2	methane	population	gdp	primary_energy_consumption
Afghanistan	1991	2.427	0.249	0.046	0.388	1.718	9.07	13299016.0	12047361708.8	13.651
Afghanistan	1992	1.379	0.022	0.046	0.363	0.927	9.0	14485543.0	12677538631.9	8.961
Afghanistan	1993	1.333	0.018	0.047	0.352	0.894	8.9	15816601.0	9834581108.1	8.935
Afghanistan	1994	1.282	0.015	0.047	0.338	0.86	8.97	17075728.0	7919857155.6	8.617
Afghanistan	1995	1.23	0.015	0.047	0.322	0.824	9.15	18110662.0	12307526078.0	7.246
Afghanistan	1996	1.165	0.007	0.047	0.308	0.78	9.93	18853444.0	12070125458.3	7.119
Afghanistan	1997	1.084	0.004	0.047	0.283	0.728	10.6	19357126.0	11850751938.9	6.799
Afghanistan	1998	1.029	0.004	0.047	0.265	0.691	11.1	19737770.0	11692172337.6	6.618
Afghanistan	1999	0.81	0.004	0.047	0.242	0.495	11.87	20170847.0	11517317327.4	6.612
Afghanistan	2000	0.758	0.004	0.01	0.224	0.498	10.59	20779957.0	11283793214.7	5.777
Afghanistan	2001	0.798	0.07	0.007	0.209	0.491	9.36	21606992.0	11021272774.0	4.481
Afghanistan	2002	1.052	0.055	0.011	0.546	0.44	11.21	22600774.0	18804871760.0	4.262
Afghanistan	2003	1.186	0.092	0.01	0.465	0.619	11.56	23680871.0	21074344026.0	5.041

Data Overview after pre-processing:

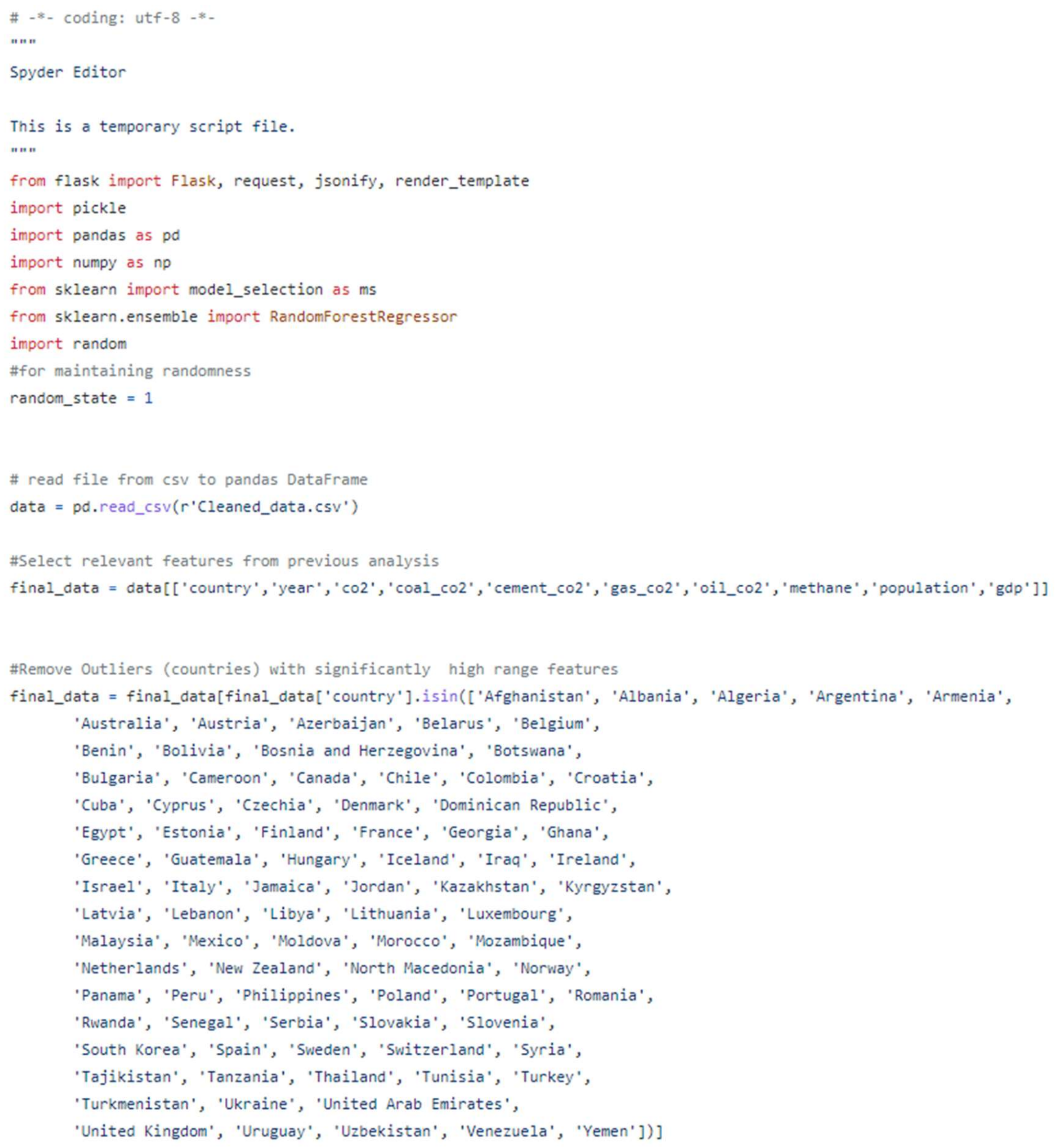


	country	year	co2	methane	ccgo	gdp_per_capita
0	Afghanistan	1991	2.427	9.07	2.401	905.883692
1	Afghanistan	1992	1.379	9.00	1.358	875.185599
2	Afghanistan	1993	1.333	8.90	1.311	621.788531
3	Afghanistan	1994	1.282	8.97	1.260	463.807877
4	Afghanistan	1995	1.230	9.15	1.208	679.573506

Model creation

The model was then created in the model.py file and was saved as a pickle file.

Libraries Import and Outlier removal:



```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
from flask import Flask, request, jsonify, render_template
import pickle
import pandas as pd
import numpy as np
from sklearn import model_selection as ms
from sklearn.ensemble import RandomForestRegressor
import random
#for maintaining randomness
random_state = 1


# read file from csv to pandas DataFrame
data = pd.read_csv(r'Cleaned_data.csv')

#Select relevant features from previous analysis
final_data = data[['country','year','co2','coal_co2','cement_co2','gas_co2','oil_co2','methane','population','gdp']]


#Remove Outliers (countries) with significantly  high range features
final_data = final_data[final_data['country'].isin(['Afghanistan', 'Albania', 'Algeria', 'Argentina', 'Armenia',
       'Australia', 'Austria', 'Azerbaijan', 'Belarus', 'Belgium',
       'Benin', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana',
       'Bulgaria', 'Cameroon', 'Canada', 'Chile', 'Colombia', 'Croatia',
       'Cuba', 'Cyprus', 'Czechia', 'Denmark', 'Dominican Republic',
       'Egypt', 'Estonia', 'Finland', 'France', 'Georgia', 'Ghana',
       'Greece', 'Guatemala', 'Hungary', 'Iceland', 'Iraq', 'Ireland',
       'Israel', 'Italy', 'Jamaica', 'Jordan', 'Kazakhstan', 'Kyrgyzstan',
       'Latvia', 'Lebanon', 'Libya', 'Lithuania', 'Luxembourg',
       'Malaysia', 'Mexico', 'Moldova', 'Morocco', 'Mozambique',
       'Netherlands', 'New Zealand', 'North Macedonia', 'Norway',
       'Panama', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Romania',
       'Rwanda', 'Senegal', 'Serbia', 'Slovakia', 'Slovenia',
       'South Korea', 'Spain', 'Sweden', 'Switzerland', 'Syria',
       'Tajikistan', 'Tanzania', 'Thailand', 'Tunisia', 'Turkey',
       'Turkmenistan', 'Ukraine', 'United Arab Emirates',
       'United Kingdom', 'Uruguay', 'Uzbekistan', 'Venezuela', 'Yemen'])]
```

Model Overview:

```
#dimensionality reduction
final_data['ccgo'] = final_data['cement_co2'] + final_data['gas_co2'] + final_data['oil_co2'] + final_data['coal_co2']
final_data['gdp_per_capita'] = final_data['gdp'] / final_data['population']
final_data.head()


data = final_data.drop(['cement_co2','gas_co2','oil_co2','coal_co2','gdp','population'],axis=1)
```

```
#splitting dataset
ft_cols = ['year','methane','ccgo','gdp_per_capita']
lb_col = ['co2']

features = np.array(data[ft_cols])
label = np.array(data[lb_col]).ravel()

#Data splitting using sklearn train_test_split function
ft_train,ft_test,lb_train,lb_test = ms.train_test_split(features,label,test_size=0.3
                                                        ,shuffle = True, random_state= random_state)
```

```
RFR = RandomForestRegressor(max_depth = 9, max_features = 3, n_estimators = 40, random_state = random_state)
RFR.fit(ft_train, lb_train)

# Saving model to disk
pickle.dump(RFR, open('model.pkl','wb'))

model = pickle.load(open('model.pkl','rb'))
```

Flask deployment:

The flask deployment procedure includes three sections; the first is the Importing of libraries and initialization of flask app.

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
```

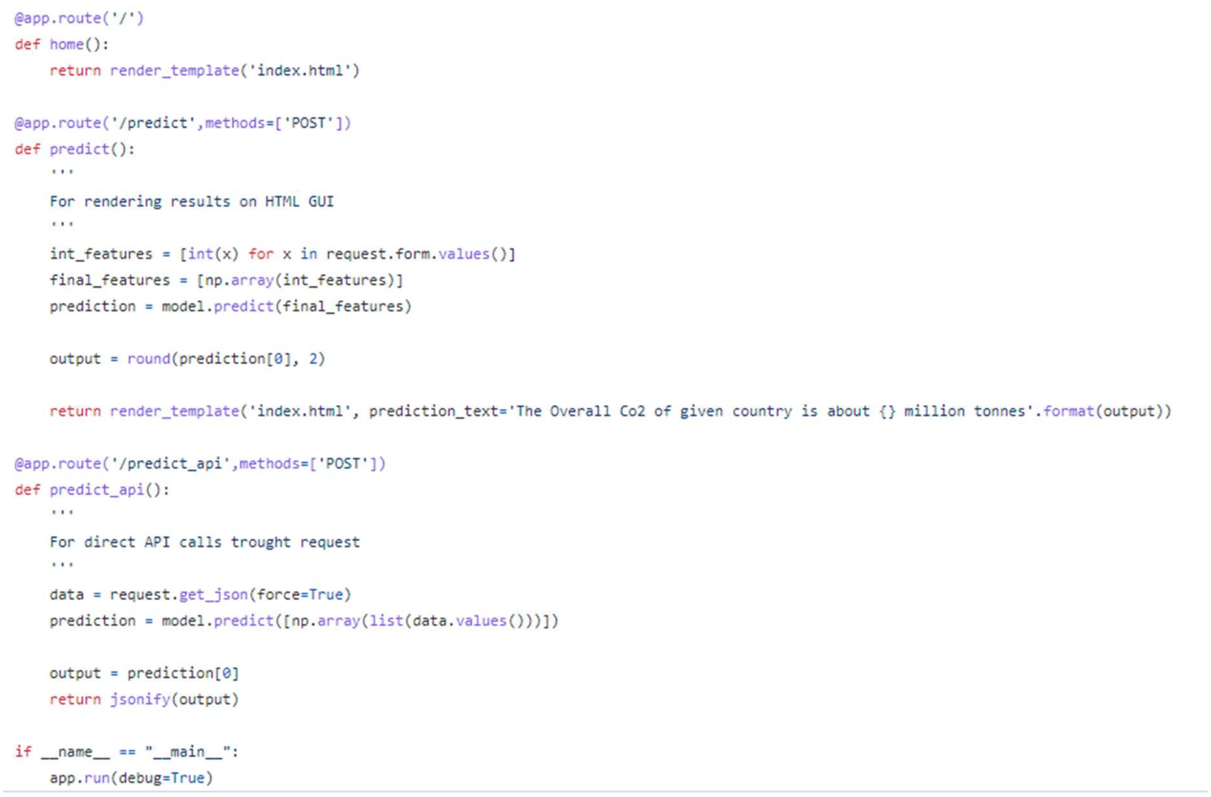
Second, loading the model

```
model = pickle.load(open('model.pkl', 'rb'))
```

Finally, The default or main page of our web application will be its lone page. The / route is the home URL. This should open the index.html file, which serves as the homepage by default.

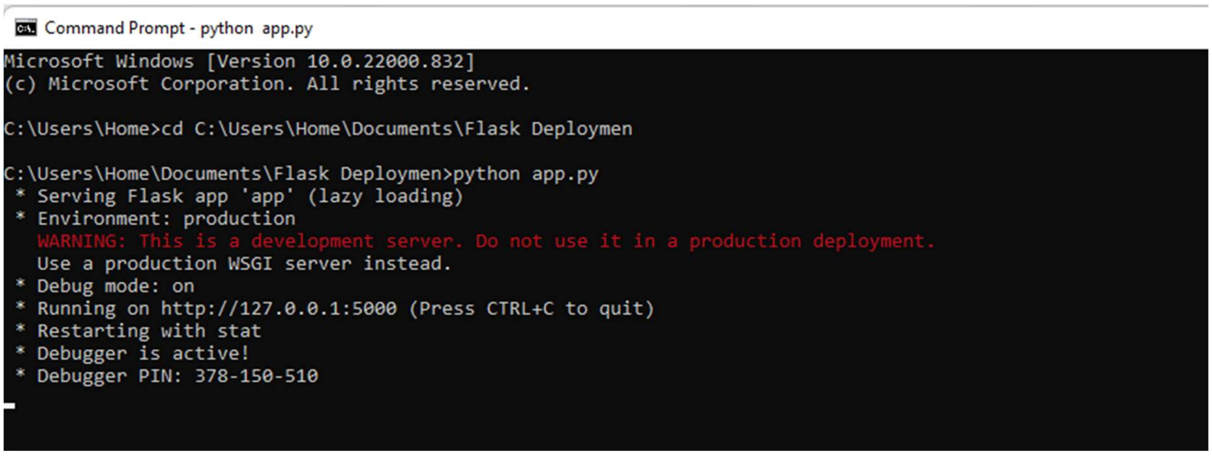
We want to carry out a specific action when a user accesses the home page. Based on the POST requests, we created a function that would respond to user queries.

When the data is returned to the webserver, a user submits a POST request. The user can enter the Year, Methane, CCGO(Coal, Cement, Oil and Gas Co2), and GDP per Capita of the country. The request.form.values() function is used to get this data from an HTML file. The features list is now converted to a NumPy array and stored in the final_features.



```python
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='The Overall Co2 of given country is about {} million tonnes'.format(output))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
```

Testing app

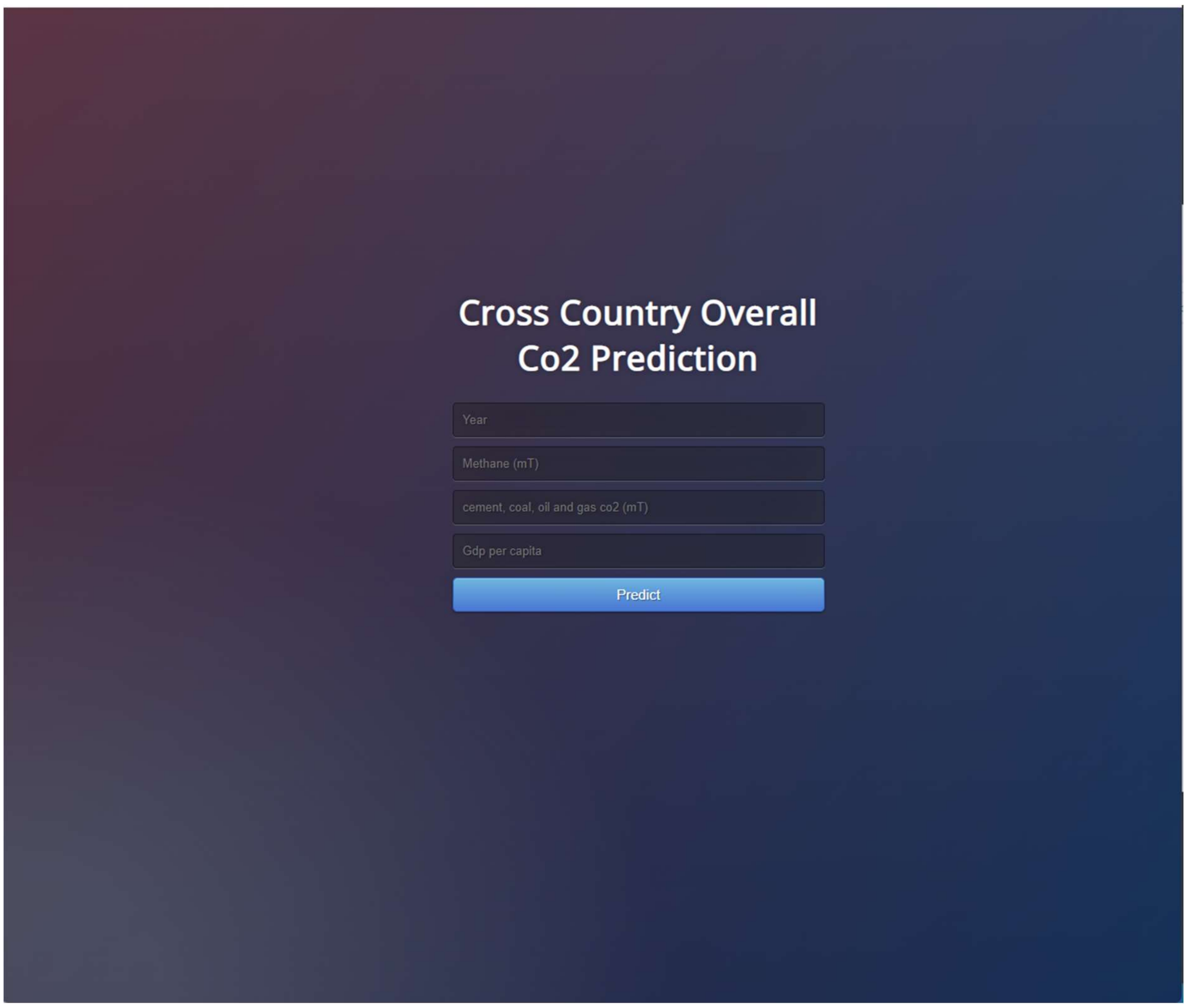


Command Prompt - python app.py

```
Microsoft Windows [Version 10.0.22000.832]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Home>cd C:\Users\Home\Documents\Flask Deploymen

C:\Users\Home\Documents\Flask Deploymen>python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 378-150-510
```

The http://127.0.0.1.5000 can be used to access the app.

Home page of app:

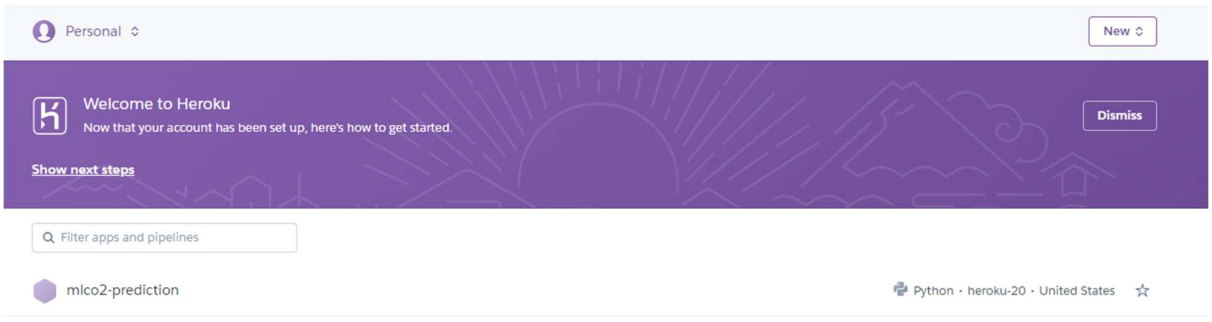

Cross Country Overall Co2 Prediction
Year
Methane (mT)
cement, coal, oil and gas co2 (mT)
Gdp per capita
Predict

Heroku deployment

Things to do before deployment

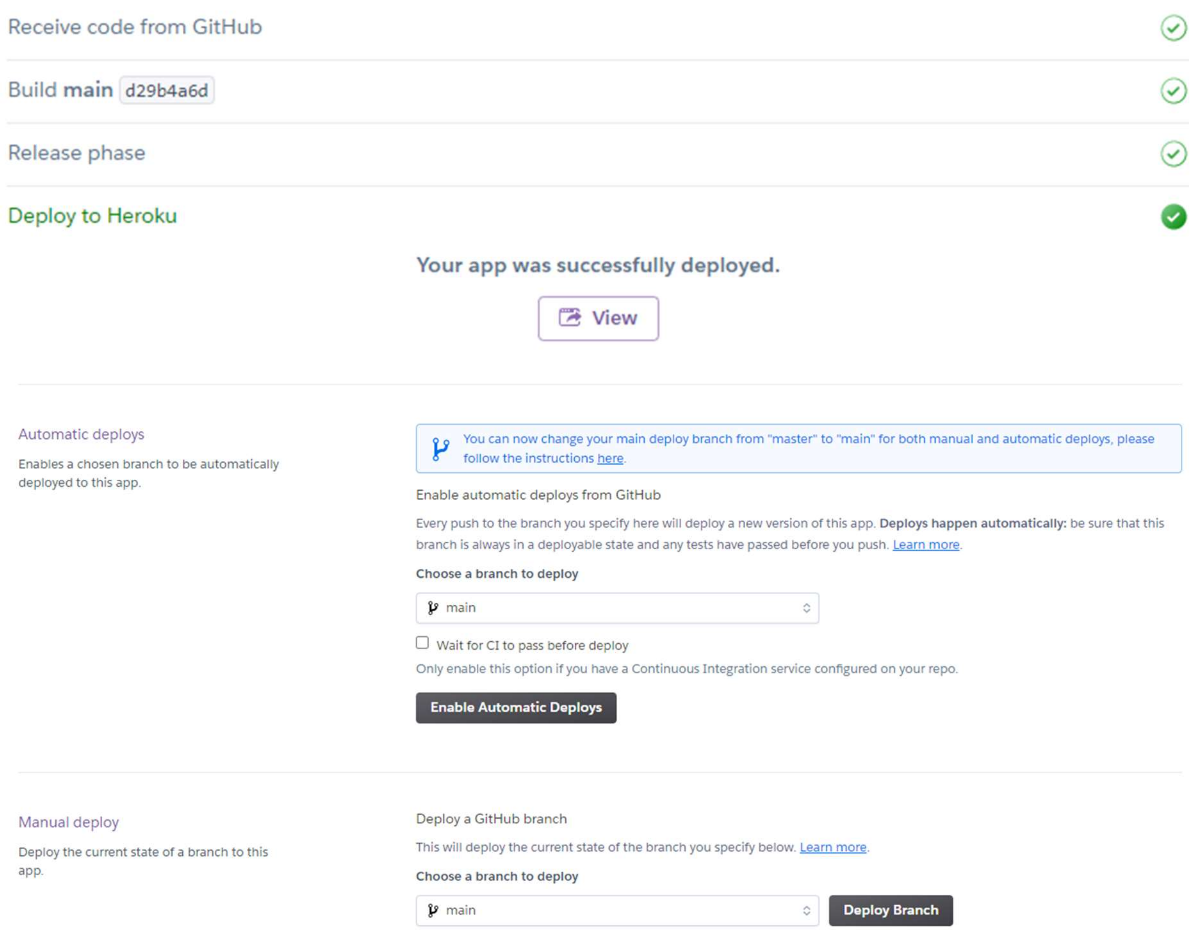
Use ‘pip freeze’ keyword and ‘> requirements’ to help in the installation of dependencies on Heroku.

The next stage involves signing up, creating an app, and specifying the name.



Welcome to Heroku
Now that your account has been set up, here's how to get started.

mlco2-prediction — Python · heroku-20 · United States



Receive code from GitHub ✓

Build **main** `d29b4a6d` ✓

Release phase ✓

Deploy to Heroku ✓

Your app was successfully deployed.

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

Deploy the current state of a branch to this app.

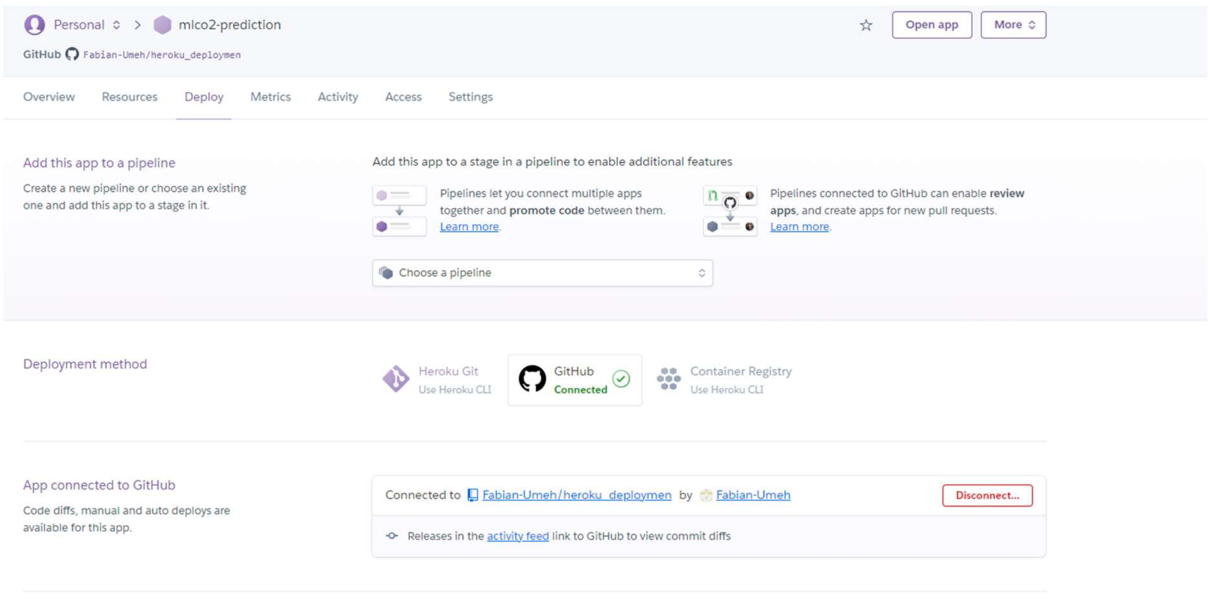
Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main

Connect to GitHub



Personal > mlco2-prediction

GitHub Fabian-Umeh/heroku_deploymen

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

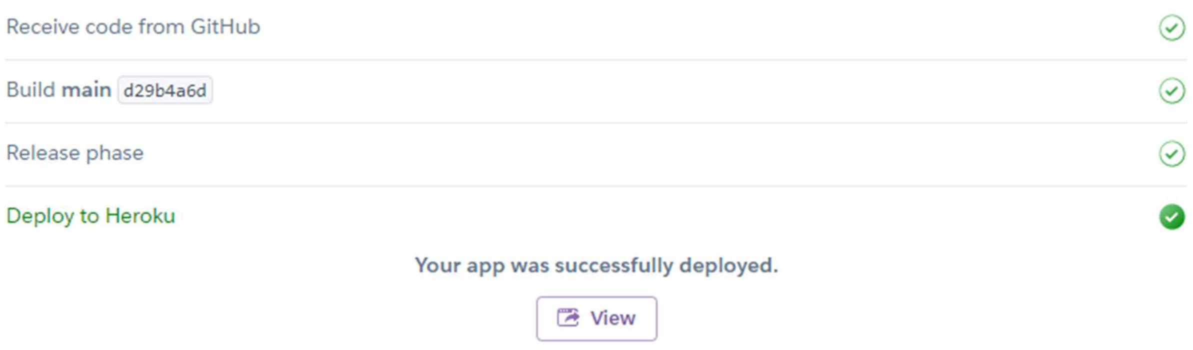
App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to Fabian-Umeh/heroku_deploymen by Fabian-Umeh

Releases in the activity feed link to GitHub to view commit diffs

Deployment


Receive code from GitHub ✓

Build **main** `d29b4a6d` ✓

Release phase ✓

Deploy to Heroku ✓

Your app was successfully deployed.