Data Science Intern at Data Glacier

Week 5: Cloud and API deployment

Name: Tejeswar Reddy Nalijeni

Batch Code: LISUM35

Date: 2 August 2024

Submitted to: Data Glacier

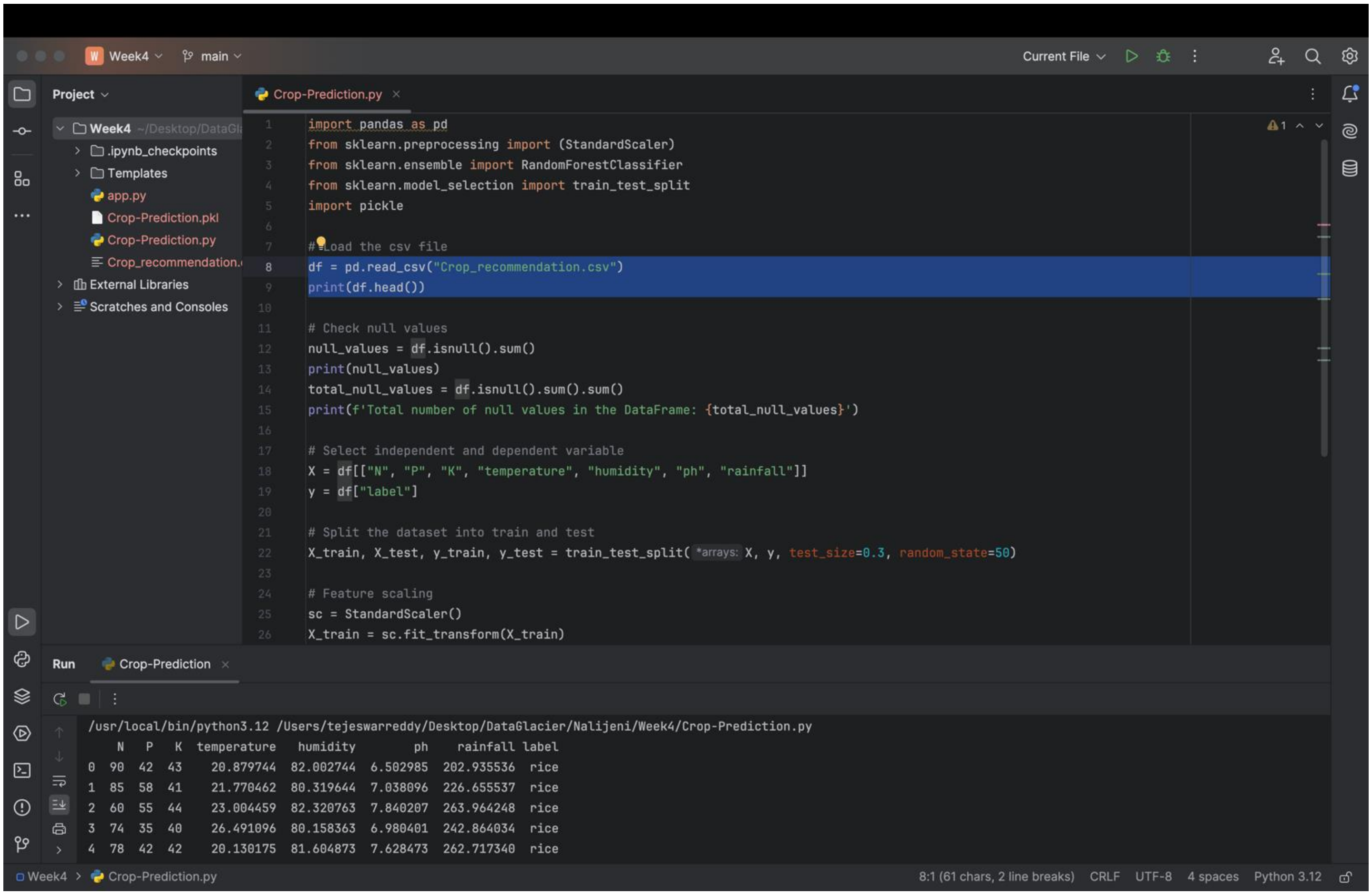
Dataset Collection:

The Data is collected from Kaggle which contains the data of crops based on certain conditions.

Attributes:

This contains 8 attributes which are N, P, K, temperature, humidity, pH, rainfall, label.

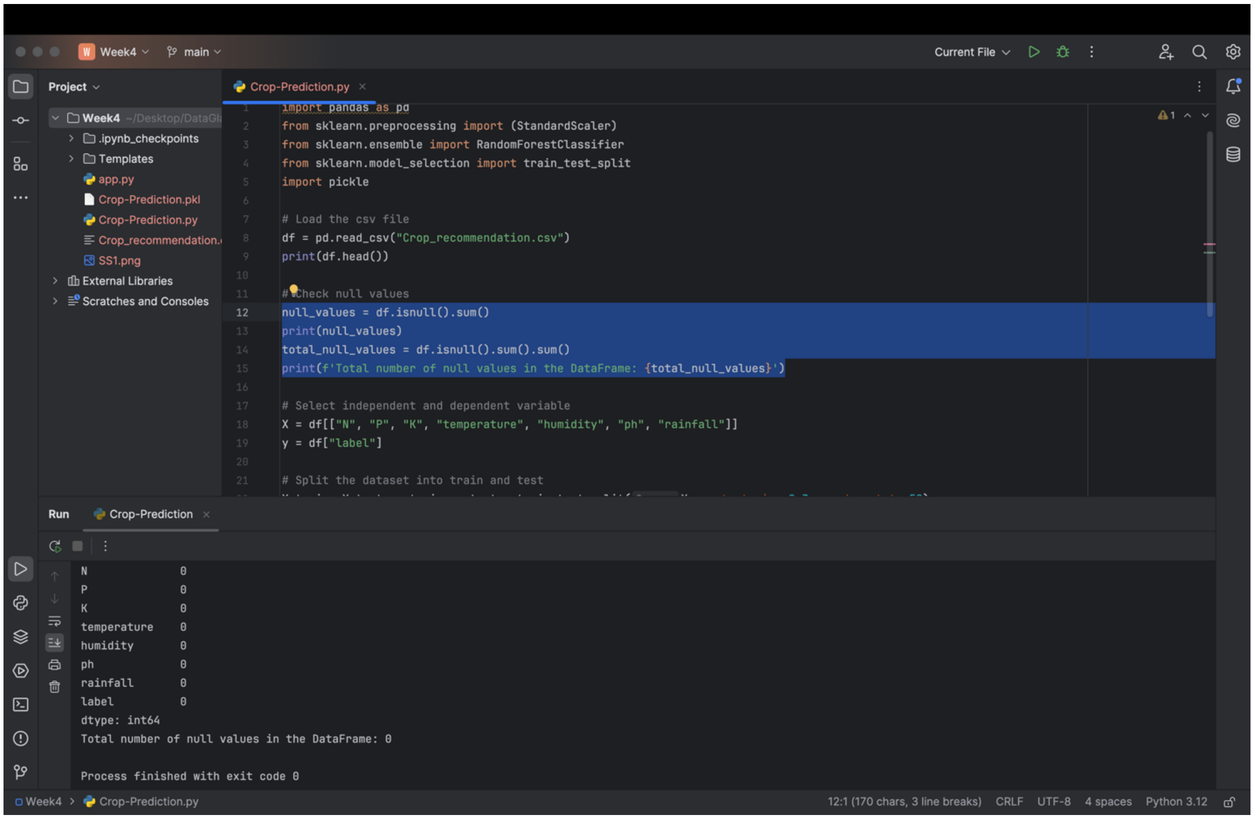
Building a Model:



```
import pandas as pd
from sklearn.preprocessing import (StandardScaler)
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle

#Load the csv file
df = pd.read_csv("Crop_recommendation.csv")
print(df.head())

# Check null values
null_values = df.isnull().sum()
print(null_values)
total_null_values = df.isnull().sum().sum()
print(f'Total number of null values in the DataFrame: {total_null_values}')

# Select independent and dependent variable
X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
y = df["label"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

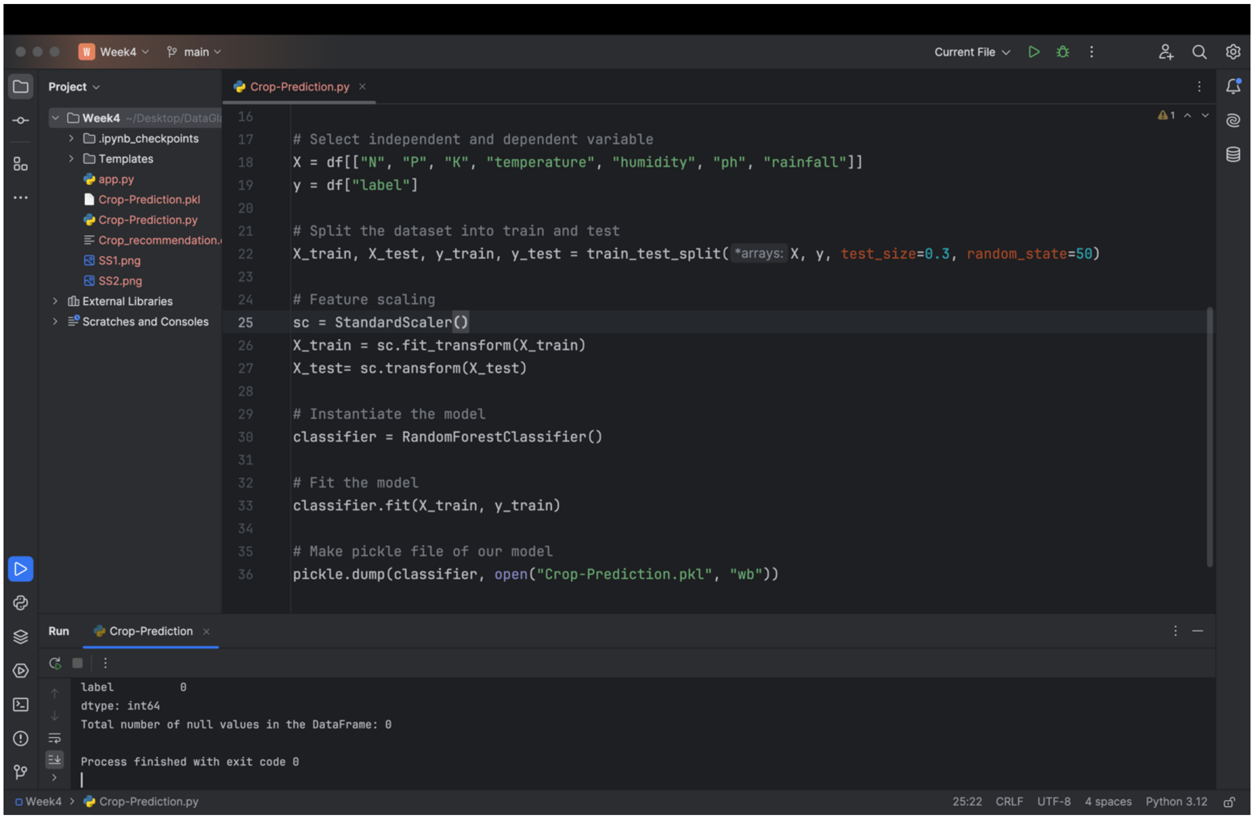
```
/usr/local/bin/python3.12 /Users/tejeswarreddy/Desktop/DataGlacier/Nalijeni/Week4/Crop-Prediction.py
    N   P   K  temperature   humidity        ph    rainfall label
0  90  42  43    20.879744  82.002744  6.502985  202.935536  rice
1  85  58  41    21.770462  80.319644  7.038096  226.655537  rice
2  60  55  44    23.004459  82.320763  7.840207  263.964248  rice
3  74  35  40    26.491096  80.158363  6.980401  242.864034  rice
4  78  42  42    20.130175  81.604873  7.628473  262.717340  rice
```

Data Preprocessing and Building the Model:



```python
import pandas as pd
from sklearn.preprocessing import (StandardScaler)
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle

# Load the csv file
df = pd.read_csv("Crop_recommendation.csv")
print(df.head())

# Check null values
null_values = df.isnull().sum()
print(null_values)
total_null_values = df.isnull().sum().sum()
print(f'Total number of null values in the DataFrame: {total_null_values}')

# Select independent and dependent variable
X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
y = df["label"]

# Split the dataset into train and test
```

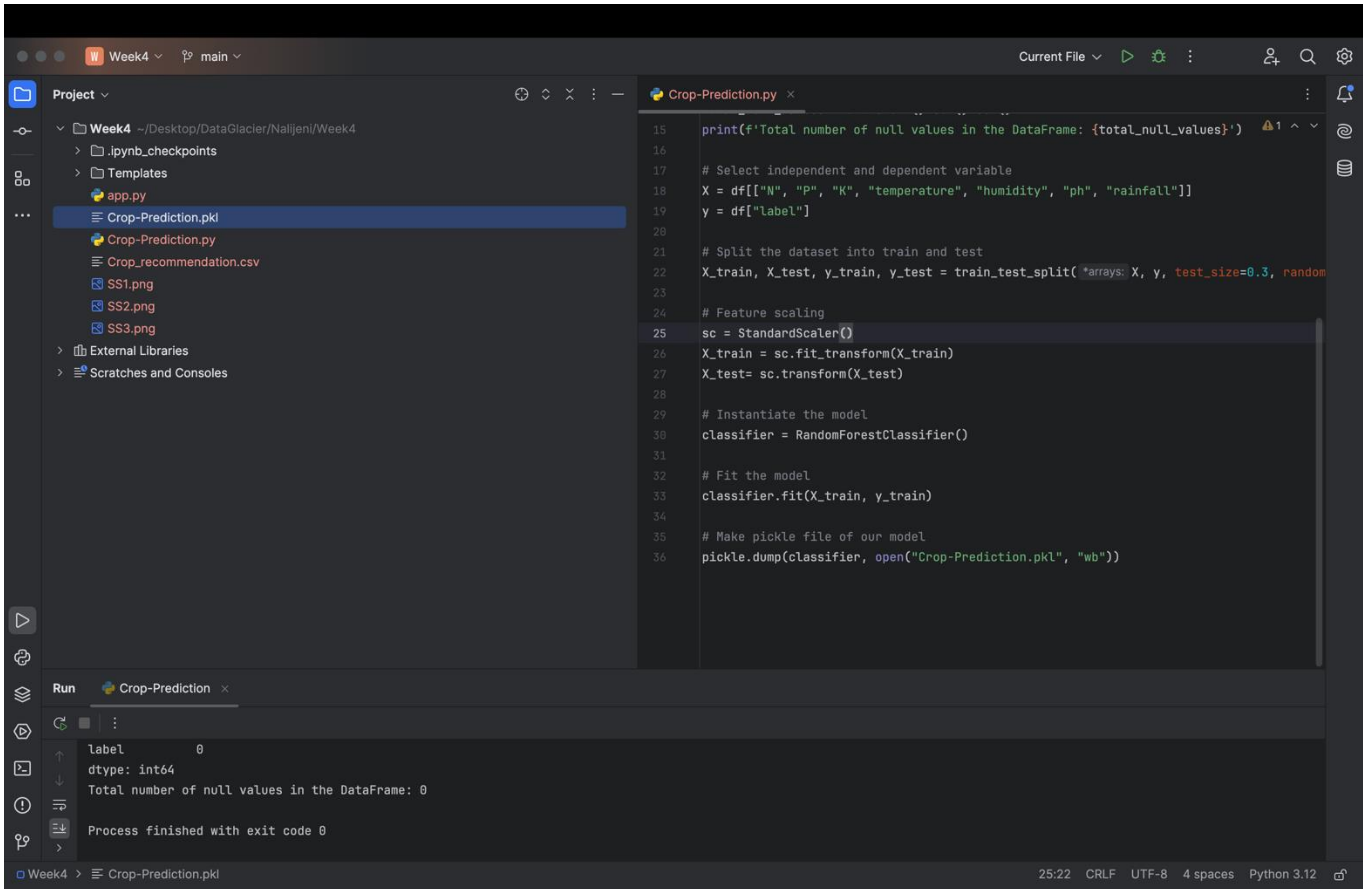
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
Total number of null values in the DataFrame: 0

Process finished with exit code 0
```



```python

# Select independent and dependent variable
X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
y = df["label"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

# Make pickle file of our model
pickle.dump(classifier, open("Crop-Prediction.pkl", "wb"))
```

```
label          0
dtype: int64
Total number of null values in the DataFrame: 0

Process finished with exit code 0
```

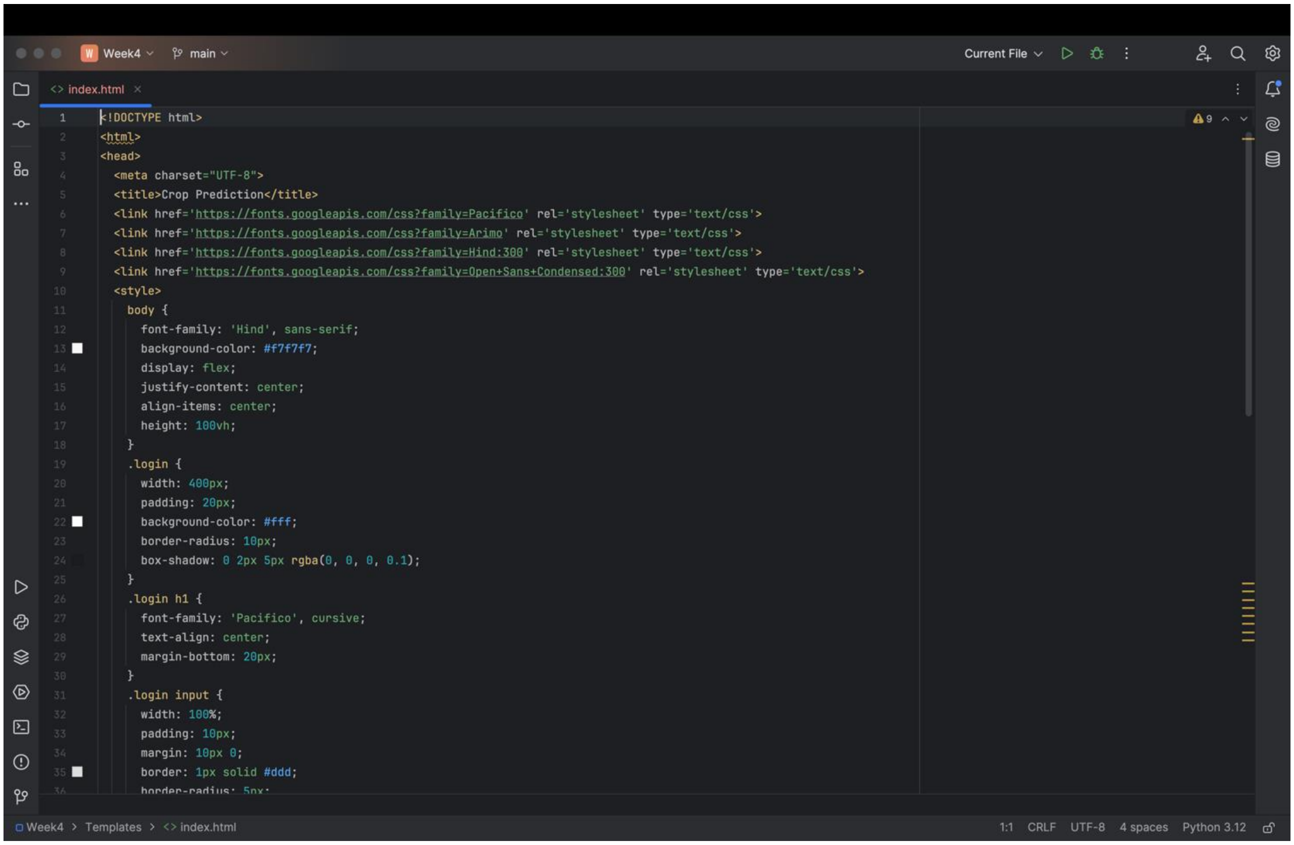
Saving the model and converting to pickle model:



```python
print(f'Total number of null values in the DataFrame: {total_null_values}')

# Select independent and dependent variable
X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
y = df["label"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

# Make pickle file of our model
pickle.dump(classifier, open("Crop-Prediction.pkl", "wb"))
```

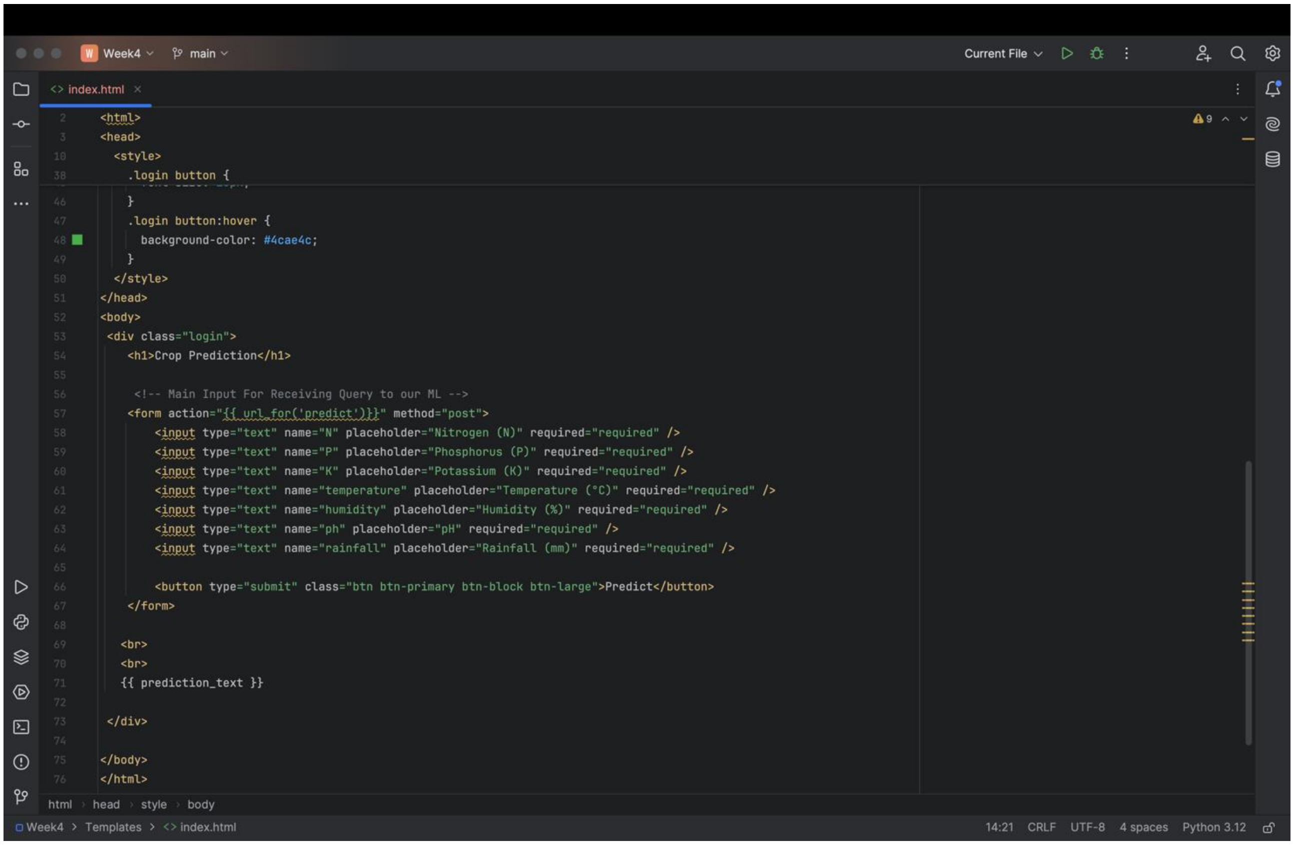
```
label          0
dtype: int64
Total number of null values in the DataFrame: 0

Process finished with exit code 0
```

Created HTML file:



Week4
main
Current File
index.html
9
Week4 > Templates > index.html
1:1 CRLF UTF-8 4 spaces Python 3.12

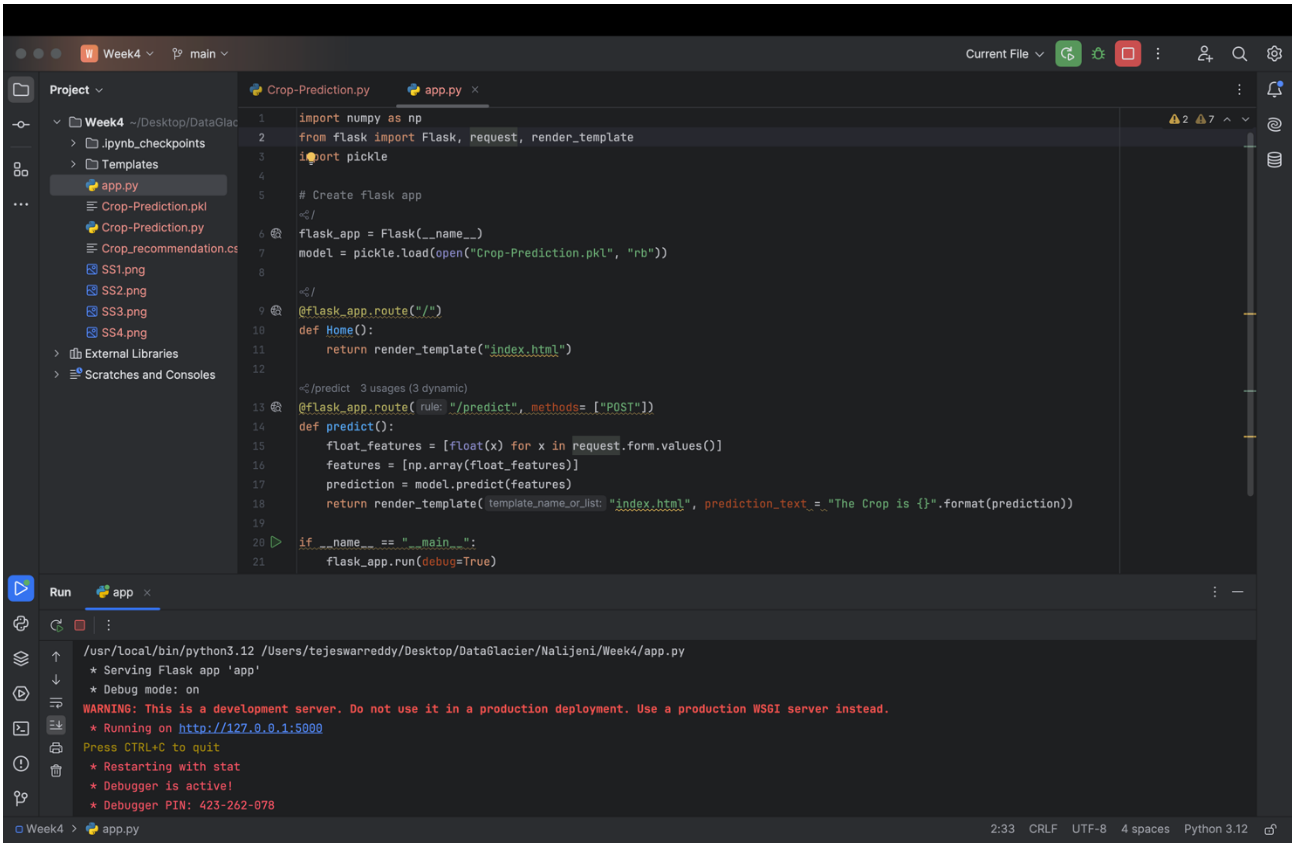
```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Crop Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
  <style>
    body {
      font-family: 'Hind', sans-serif;
      background-color: #f7f7f7;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .login {
      width: 400px;
      padding: 20px;
      background-color: #fff;
      border-radius: 10px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }
    .login h1 {
      font-family: 'Pacifico', cursive;
      text-align: center;
      margin-bottom: 20px;
    }
    .login input {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
      border: 1px solid #ddd;
      border-radius: 5px;
```



Week4
main
Current File
index.html
9
html › head › style › body
Week4 > Templates > index.html
14:21 CRLF UTF-8 4 spaces Python 3.12

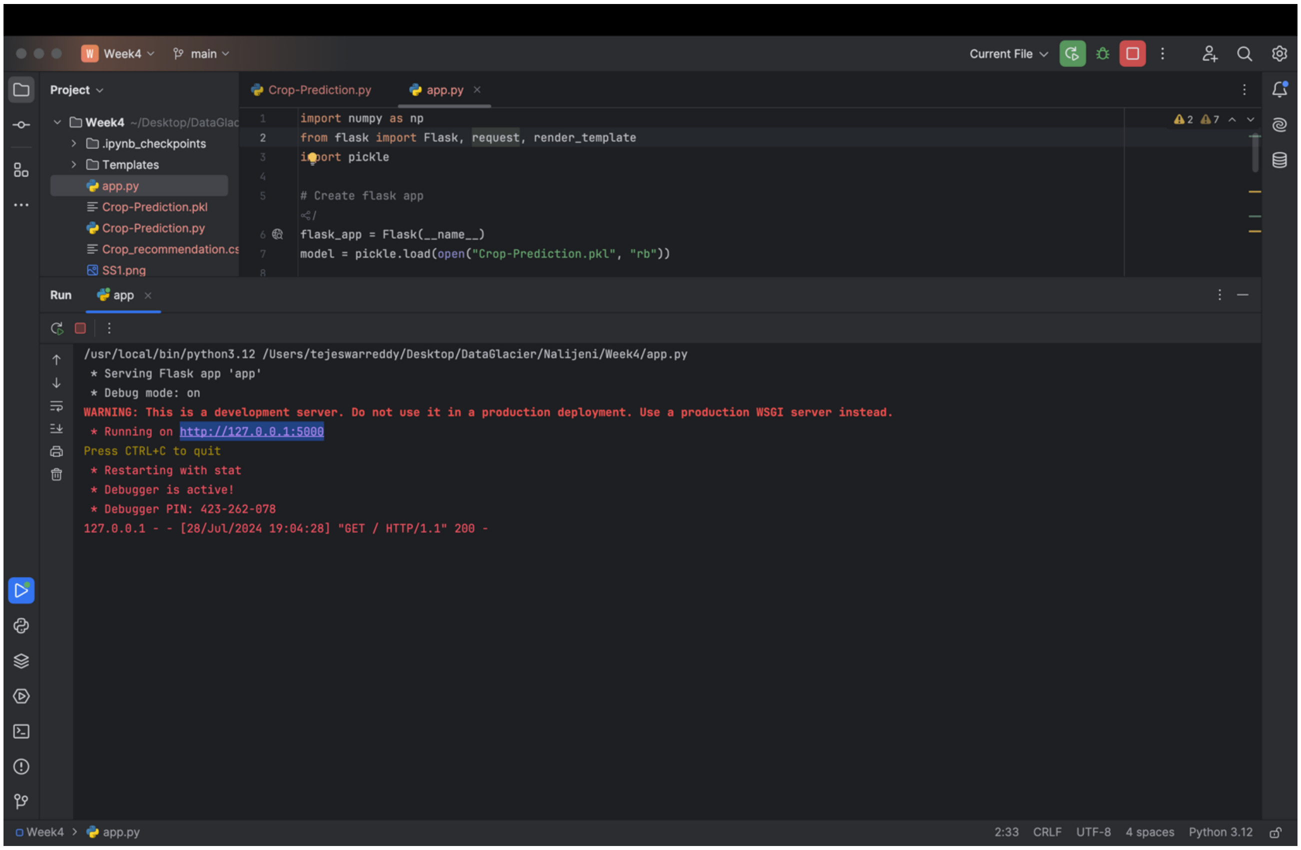
```html
<html>
<head>
  <style>
    .login button {
    }
    .login button:hover {
      background-color: #4cae4c;
    }
  </style>
</head>
<body>
 <div class="login">
    <h1>Crop Prediction</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
        <input type="text" name="N" placeholder="Nitrogen (N)" required="required" />
        <input type="text" name="P" placeholder="Phosphorus (P)" required="required" />
        <input type="text" name="K" placeholder="Potassium (K)" required="required" />
        <input type="text" name="temperature" placeholder="Temperature (°C)" required="required" />
        <input type="text" name="humidity" placeholder="Humidity (%)" required="required" />
        <input type="text" name="ph" placeholder="pH" required="required" />
        <input type="text" name="rainfall" placeholder="Rainfall (mm)" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>

</body>
</html>
```

Creating flask web application:



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

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("Crop-Prediction.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods= ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The Crop is {}".format(prediction))

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

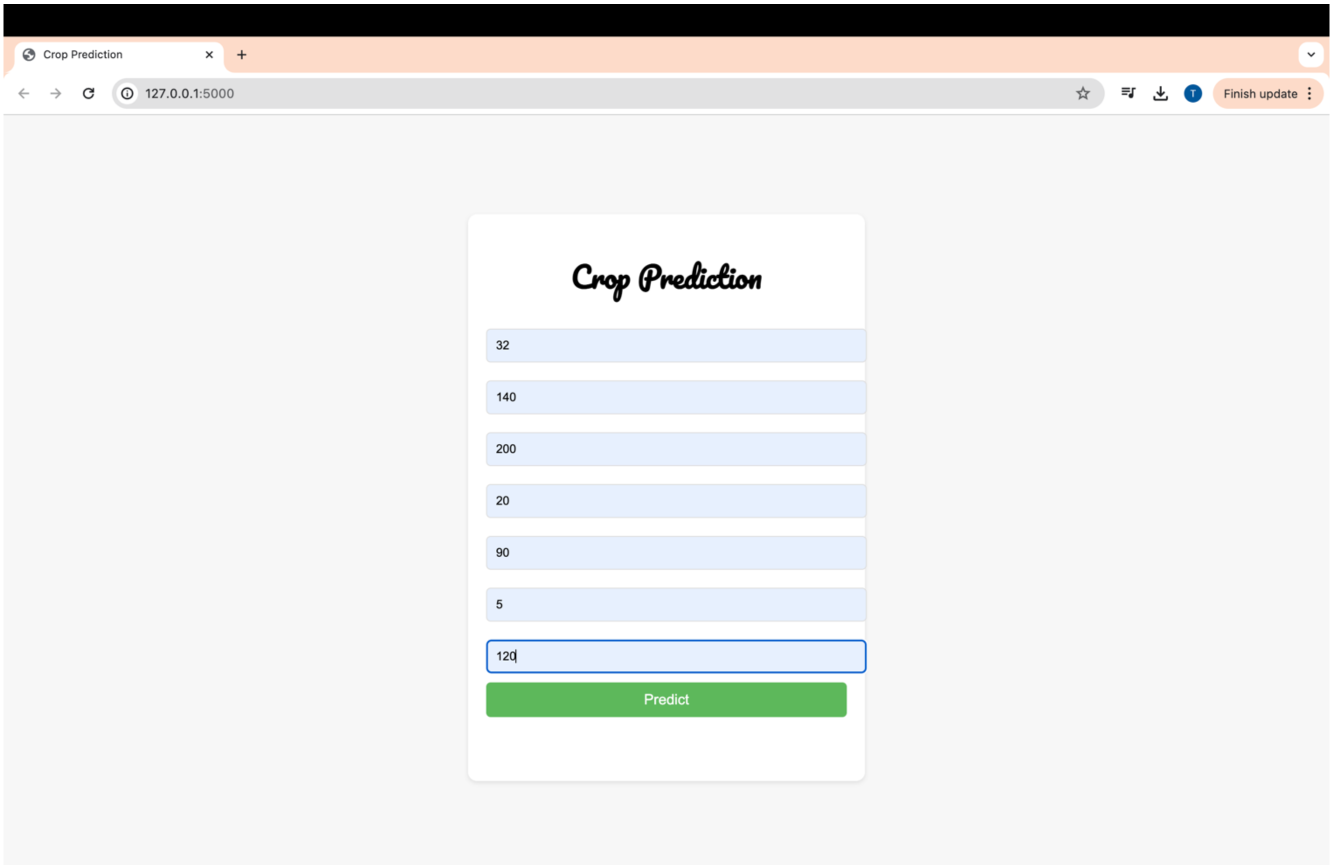
```
/usr/local/bin/python3.12 /Users/tejeswarreddy/Desktop/DataGlacier/Nalijeni/Week4/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 423-262-078
```



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

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("Crop-Prediction.pkl", "rb"))
```

```
/usr/local/bin/python3.12 /Users/tejeswarreddy/Desktop/DataGlacier/Nalijeni/Week4/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 423-262-078
127.0.0.1 - - [28/Jul/2024 19:04:28] "GET / HTTP/1.1" 200 -
```

Result of Web App:



Crop Prediction

32

140

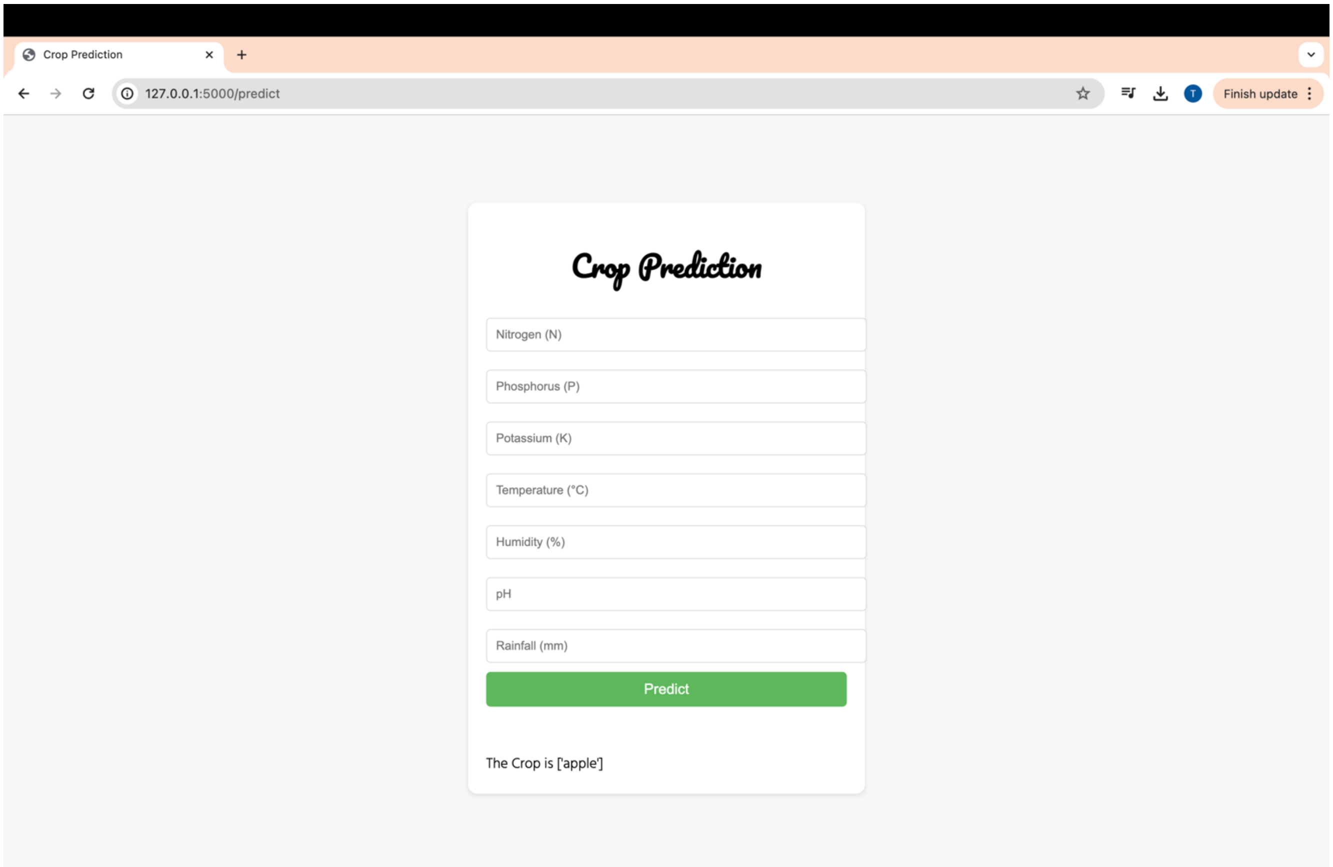
200

20

90

5

120



Crop Prediction

Nitrogen (N)

Phosphorus (P)

Potassium (K)

Temperature (°C)

Humidity (%)

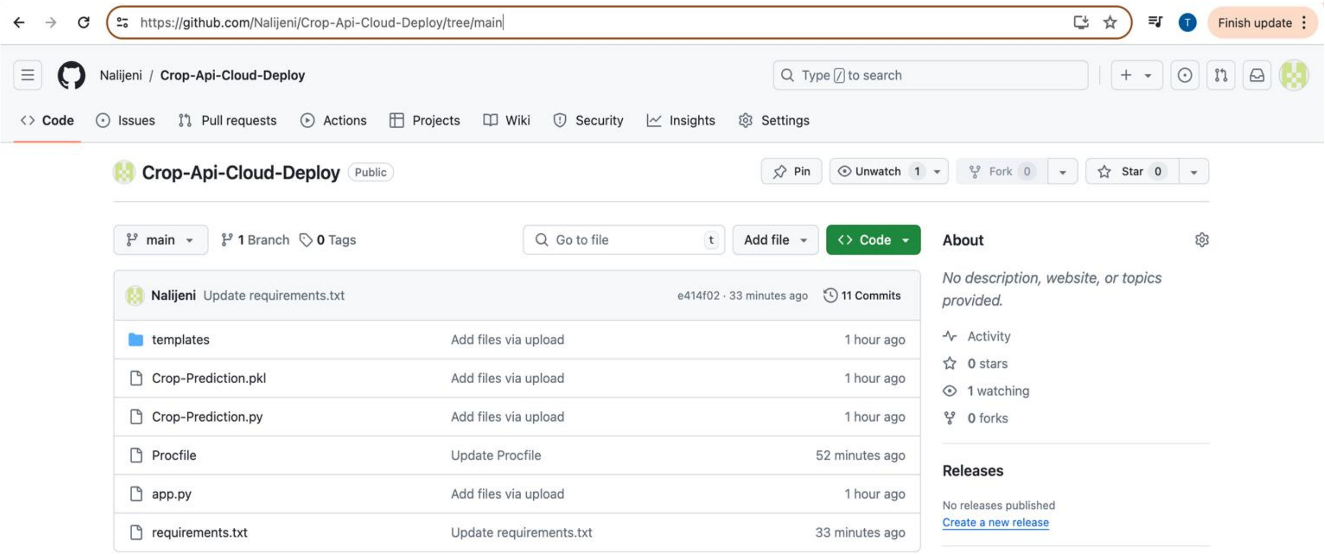
pH

Rainfall (mm)

The Crop is ['apple']

Cloud Api Deployment in Heroku:

Note: Before deployment on Heroku we need to create new repo and upload all the files required along with two files named **"Procfile"** and **"requirements.txt"** as shown in this repo - https://github.com/Nalijeni/Crop-Api-Cloud-Deploy/tree/main.



https://github.com/Nalijeni/Crop-Api-Cloud-Deploy/tree/main

Nalijeni / Crop-Api-Cloud-Deploy

Crop-Api-Cloud-Deploy Public

main · 1 Branch · 0 Tags

Nalijeni Update requirements.txt · e414f02 · 33 minutes ago · 11 Commits

templates	Add files via upload	1 hour ago
Crop-Prediction.pkl	Add files via upload	1 hour ago
Crop-Prediction.py	Add files via upload	1 hour ago
Procfile	Update Procfile	52 minutes ago
app.py	Add files via upload	1 hour ago
requirements.txt	Update requirements.txt	33 minutes ago

About

No description, website, or topics provided.

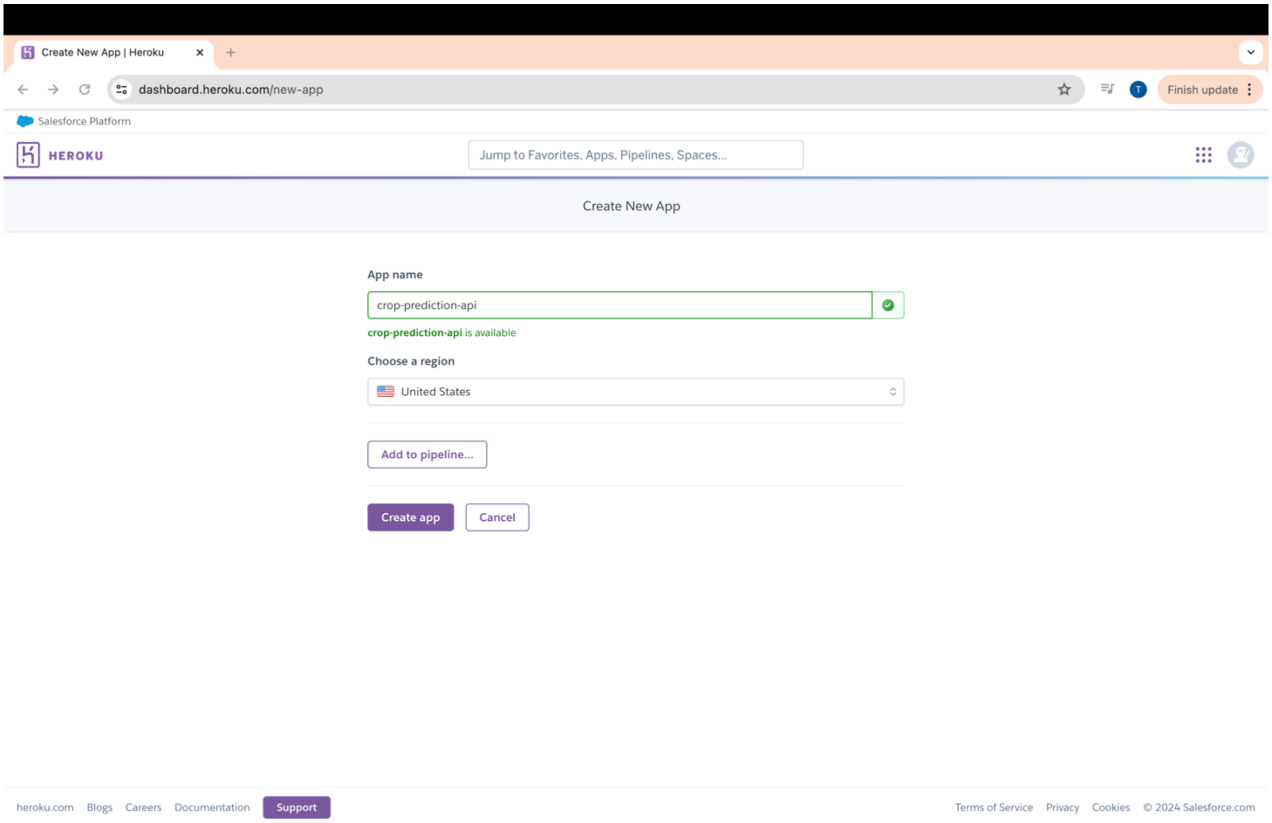
Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published

Create a new release

Step 1: Create an account in Heroku and give app name and click on create app.



Create New App | Heroku

dashboard.heroku.com/new-app

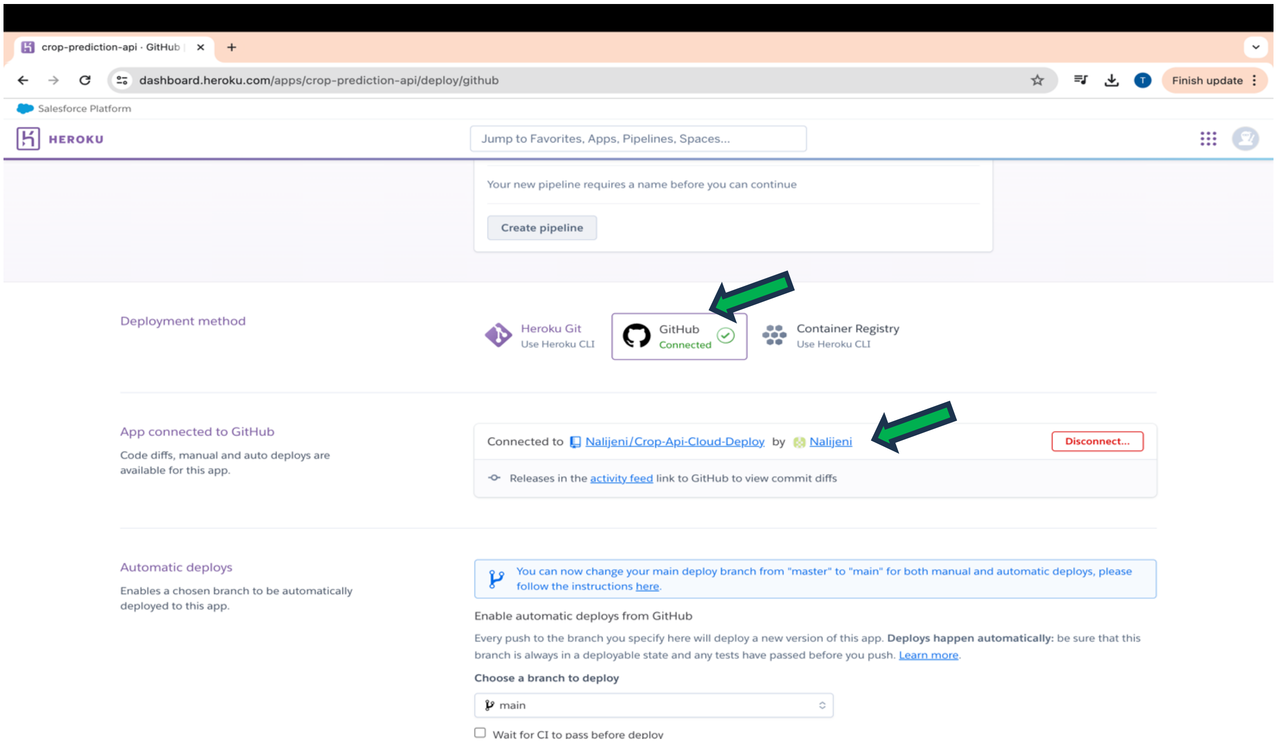
Create New App

App name: crop-prediction-api

crop-prediction-api is available

Choose a region: United States

Step 2: Select GitHub and give repo name to connect with Git repo. After successfully connecting it appears as shown in below.


crop-prediction-api · GitHub
dashboard.heroku.com/apps/crop-prediction-api/deploy/github
Finish update
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...

Your new pipeline requires a name before you can continue

Create pipeline

Deployment method

Heroku Git
Use Heroku CLI
GitHub
Connected
Container Registry
Use Heroku CLI

App connected to GitHub

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

Connected to Nalijeni/Crop-Api-Cloud-Deploy by Nalijeni

Disconnect...

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

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

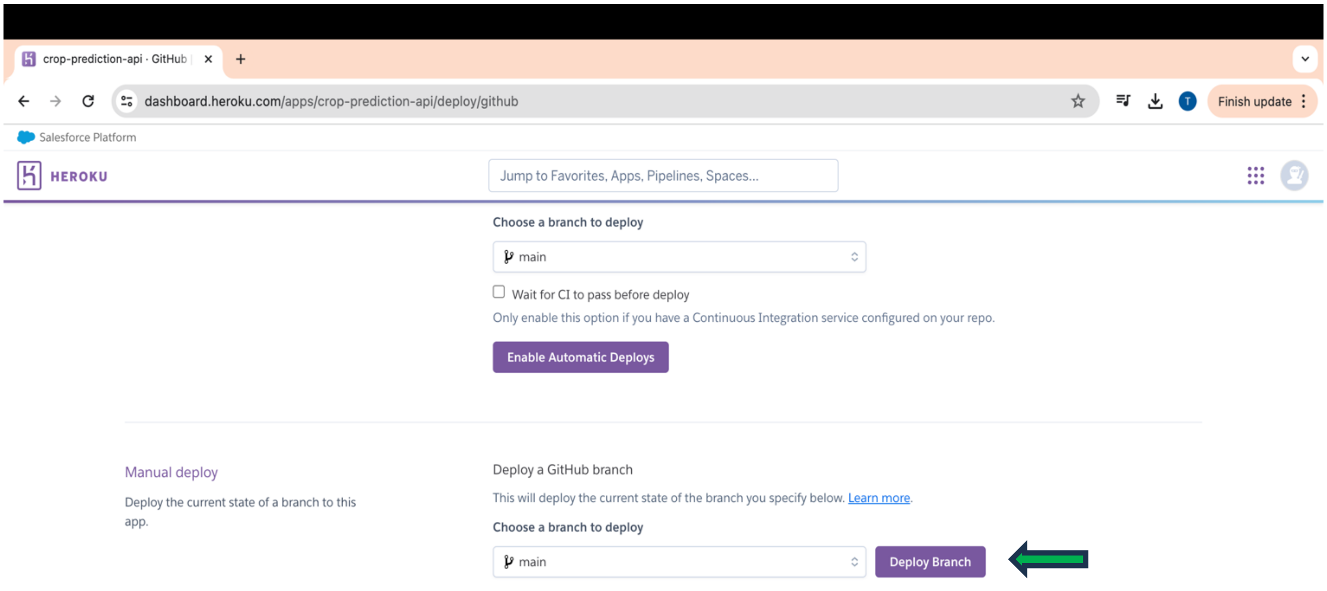
Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

□ Wait for CI to pass before deploy

Step 3: Select manual deploy and if the branch is main then proceed with deploy branch as shown below.


crop-prediction-api · GitHub
dashboard.heroku.com/apps/crop-prediction-api/deploy/github
Finish update
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...

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.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

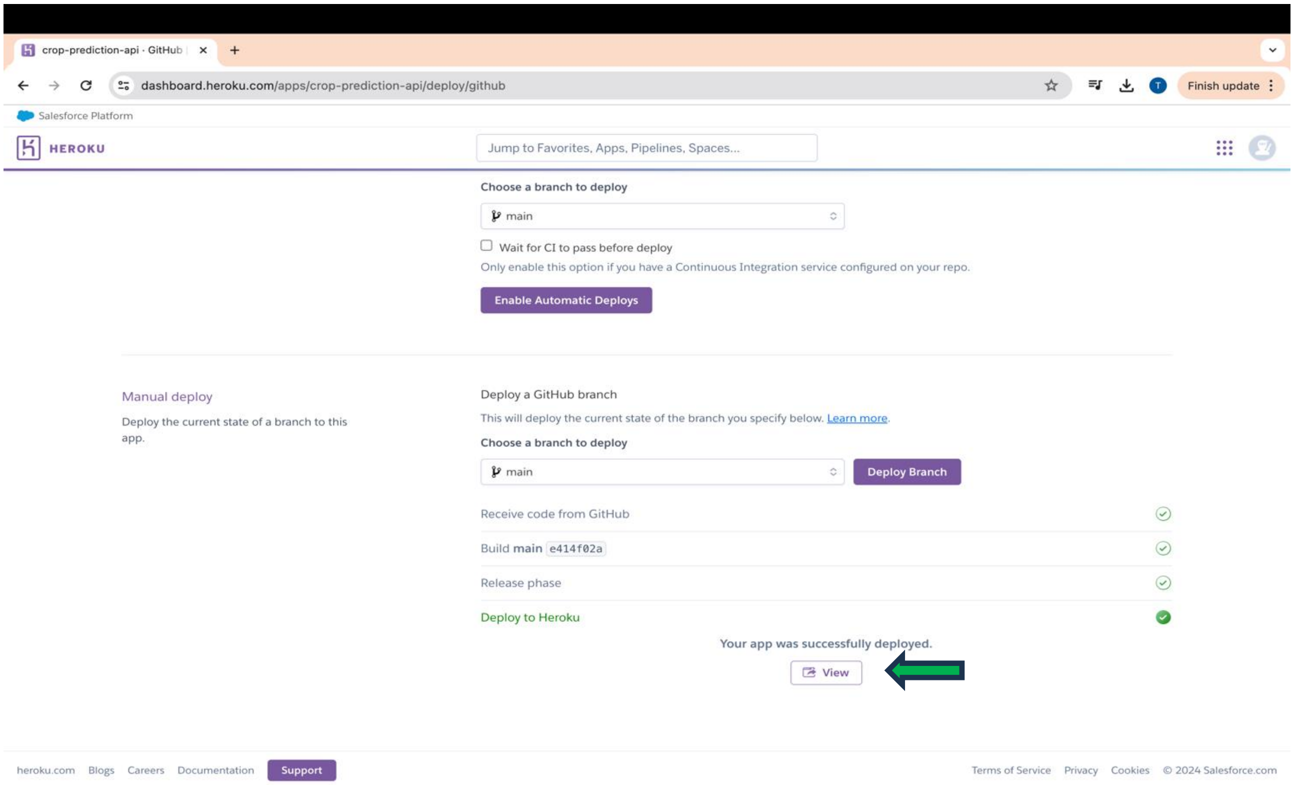
This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main

Deploy Branch

Step 4: After successful deployment to Heroku, we can now view the app by clicking on view which directs to its website.



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

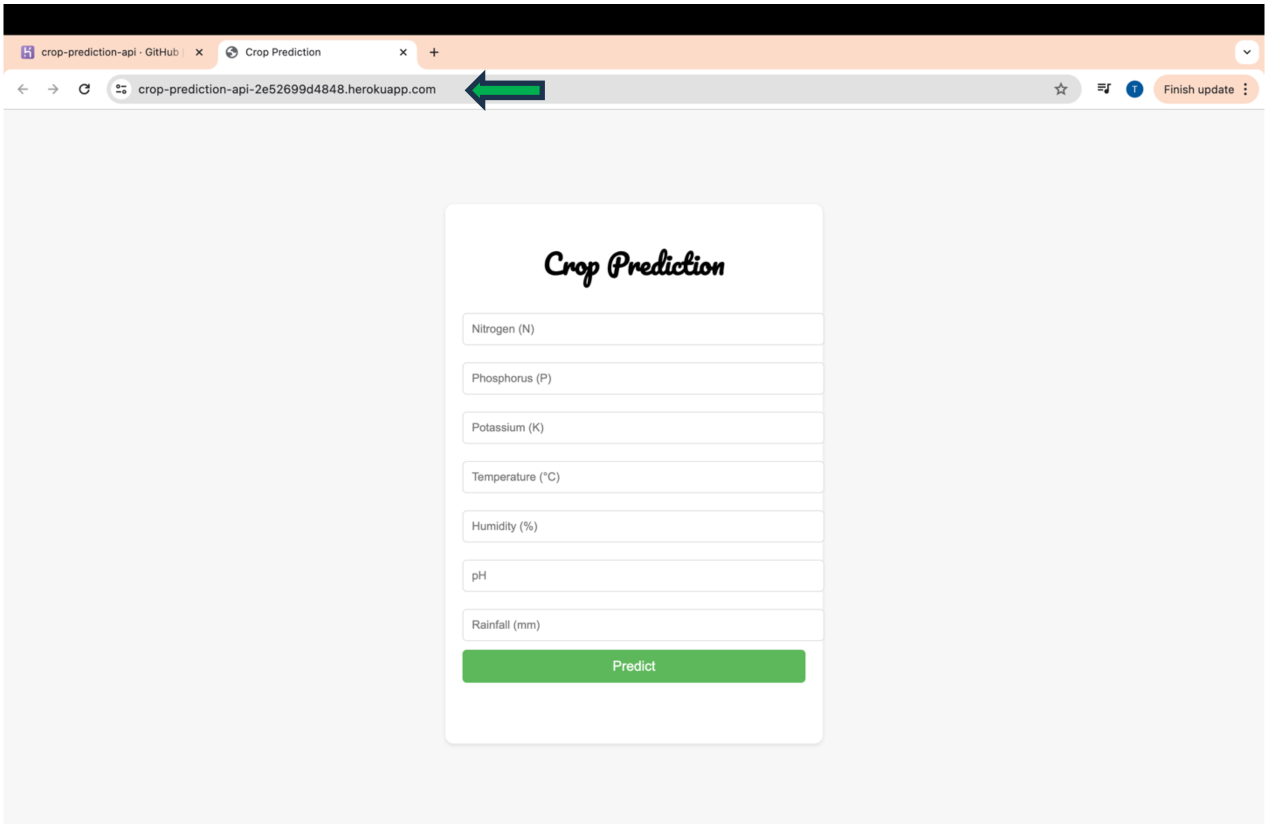
Receive code from GitHub ✓

Build main `e414f02a` ✓

Release phase ✓

Deploy to Heroku ✓

Your app was successfully deployed.



The website for my cloud deployed app - https://crop-prediction-api-2e52699d4848.herokuapp.com/