

Data Science Intern at Data Glacier

Week 4: Deployment on Flask

Name: Tejeswar Reddy
Nalijeni

Batch Code: LISUM35

Date: 28 July 2024

Submitted to: Data Glacier

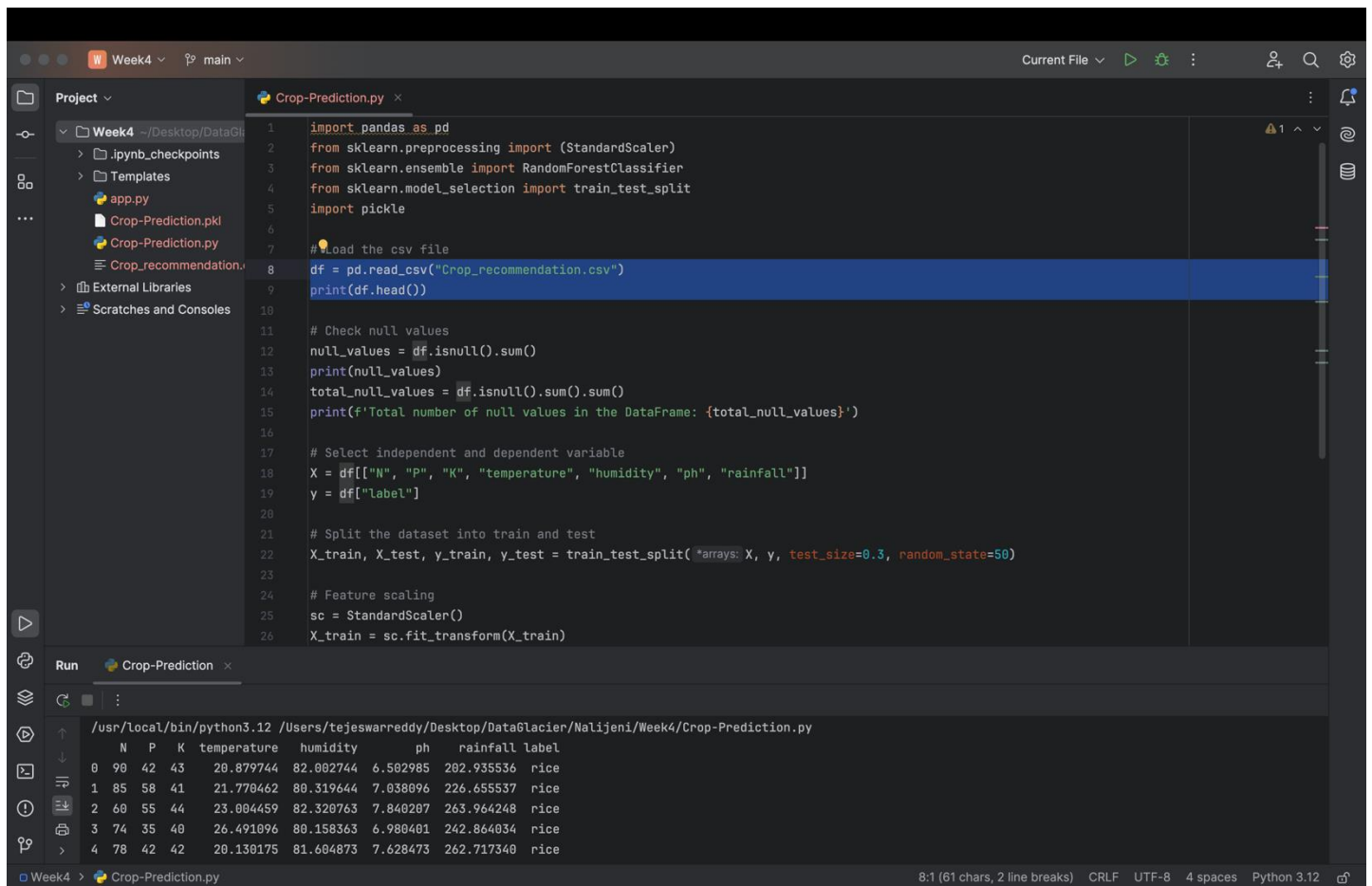
Dataset Collection:

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:



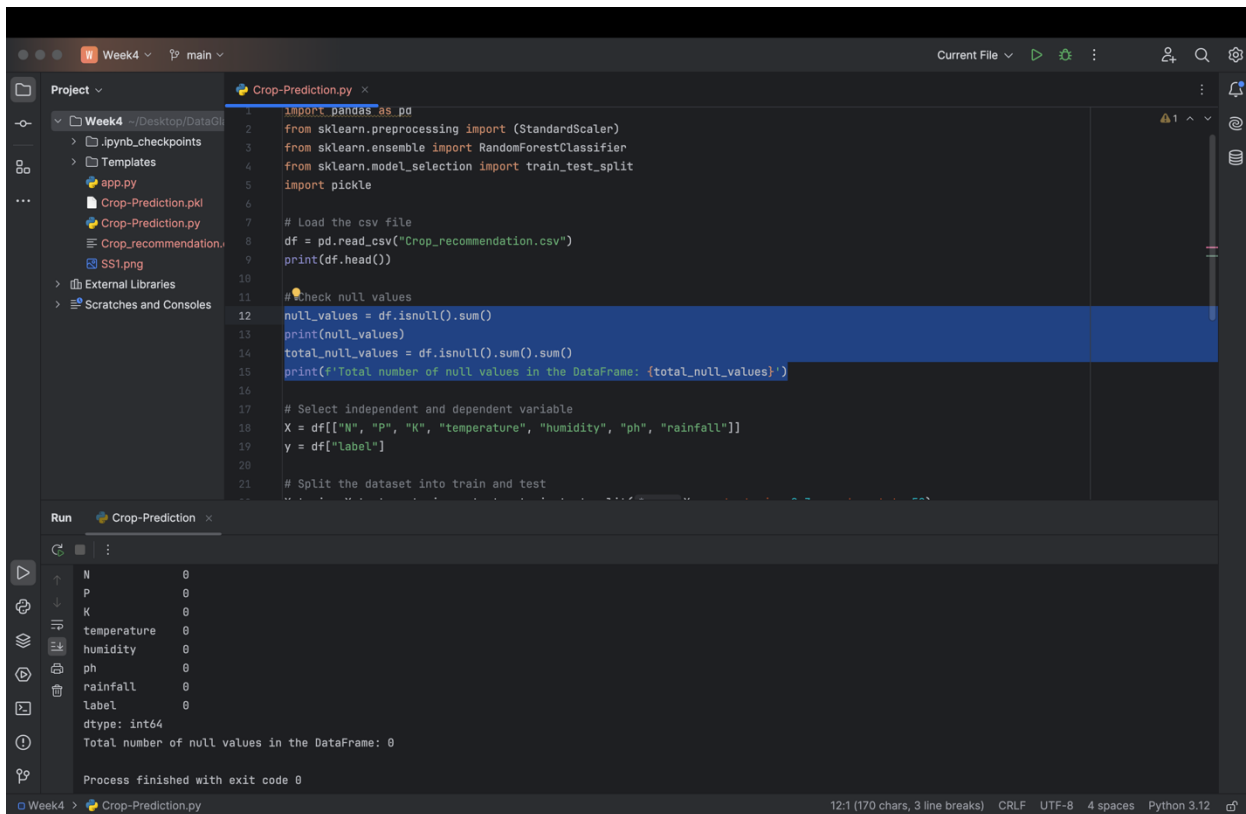
```
1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.model_selection import train_test_split
5 import pickle
6
7 # Load the csv file
8 df = pd.read_csv("Crop_recommendation.csv")
9 print(df.head())
10
11 # Check null values
12 null_values = df.isnull().sum()
13 print(null_values)
14 total_null_values = df.isnull().sum().sum()
15 print(f'Total number of null values in the DataFrame: {total_null_values}')
16
17 # Select independent and dependent variable
18 X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
19 y = df["label"]
20
21 # Split the dataset into train and test
22 X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.3, random_state=50)
23
24 # Feature scaling
25 sc = StandardScaler()
26 X_train = sc.fit_transform(X_train)
```

Run Crop-Prediction

```
/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  50  41    21.770462    80.319644    7.030096    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
```

8:1 (61 chars, 2 line breaks) CRLF UTF-8 4 spaces Python 3.12

Data Preprocessing and Building the Model:

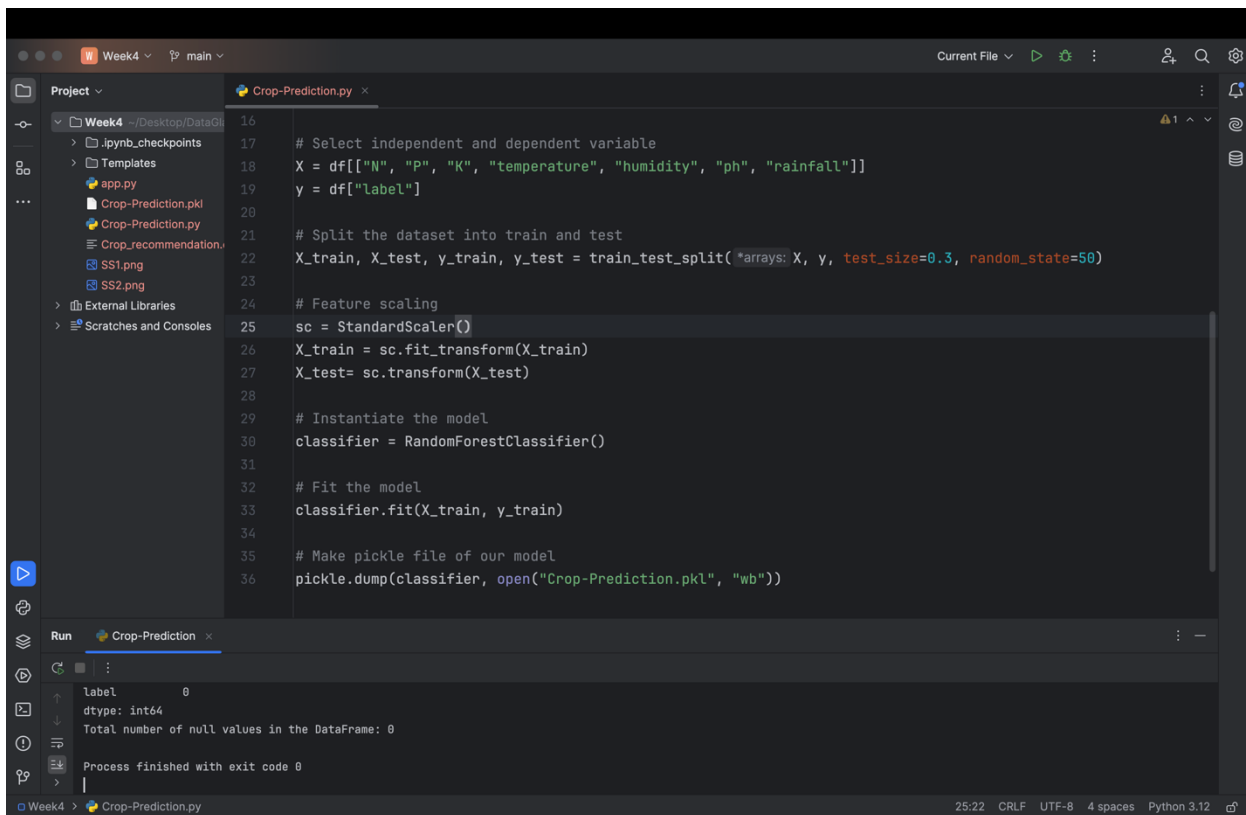


```
1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.model_selection import train_test_split
5 import pickle
6
7 # Load the csv file
8 df = pd.read_csv("Crop_recommendation.csv")
9 print(df.head())
10
11 # Check null values
12 null_values = df.isnull().sum()
13 print(null_values)
14 total_null_values = df.isnull().sum().sum()
15 print(f'Total number of null values in the DataFrame: {total_null_values}')
16
17 # Select independent and dependent variable
18 X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
19 y = df["label"]
20
21 # Split the dataset into train and test
```

Run Crop-Prediction

```
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
```



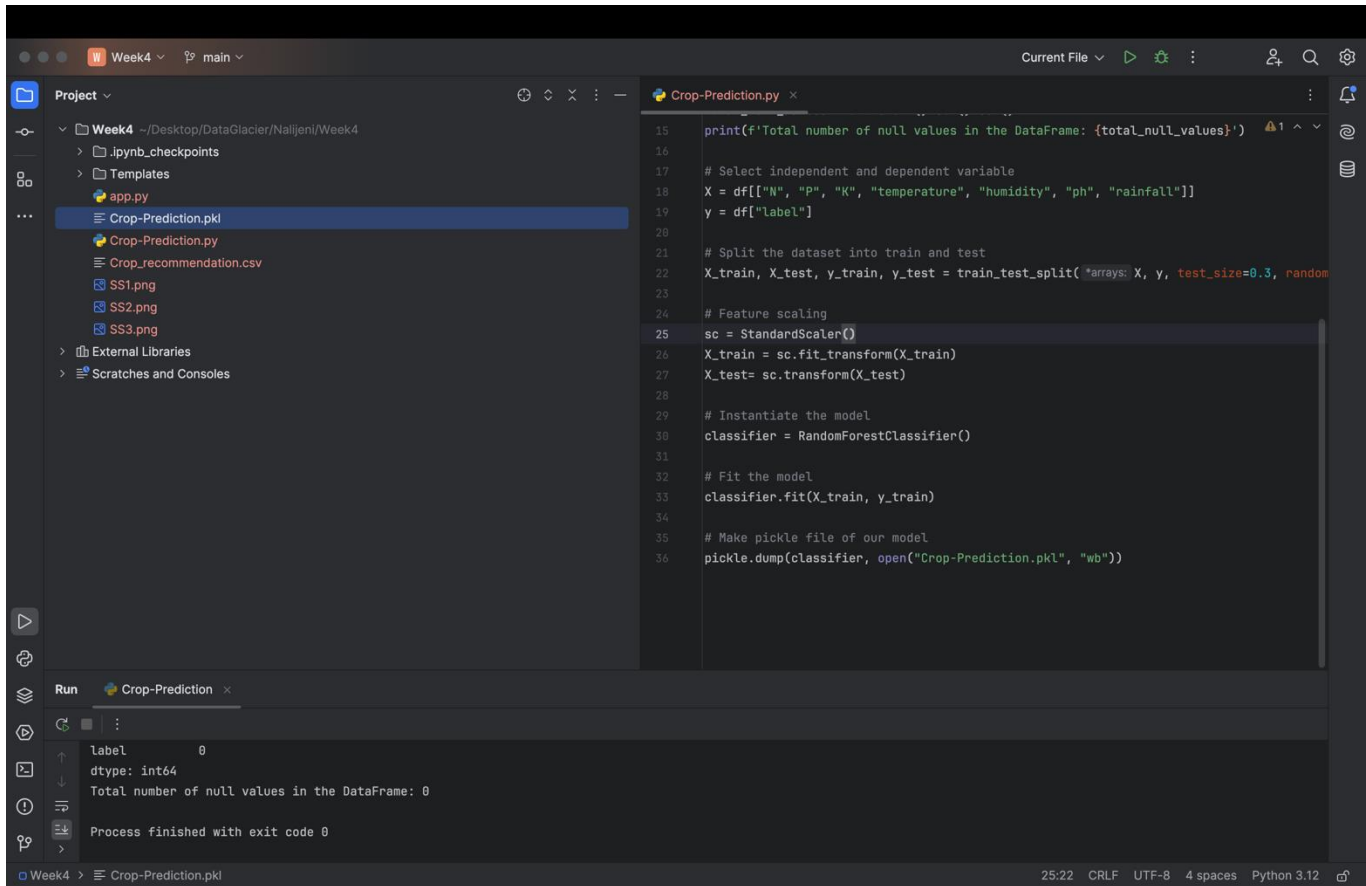
```
16
17 # Select independent and dependent variable
18 X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
19 y = df["label"]
20
21 # Split the dataset into train and test
22 X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.3, random_state=50)
23
24 # Feature scaling
25 sc = StandardScaler()
26 X_train = sc.fit_transform(X_train)
27 X_test = sc.transform(X_test)
28
29 # Instantiate the model
30 classifier = RandomForestClassifier()
31
32 # Fit the model
33 classifier.fit(X_train, y_train)
34
35 # Make pickle file of our model
36 pickle.dump(classifier, open("Crop-Prediction.pkl", "wb"))
```

Run Crop-Prediction

```
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:



The screenshot displays a Jupyter Notebook environment with a dark theme. The left sidebar shows a project structure for 'Week4' located at '~/Desktop/DataGlacier/Nalijeni/Week4'. The files listed include '.ipynb_checkpoints', 'Templates', 'app.py', 'Crop-Prediction.pkl', 'Crop-Prediction.py', 'Crop_recommendation.csv', and three image files (SS1.png, SS2.png, SS3.png). Below these are 'External Libraries' and 'Scratches and Consoles'.

The main editor window shows the 'Crop-Prediction.py' file with the following Python code:

```
15 print(f'Total number of null values in the DataFrame: {total_null_values}')
16
17 # Select independent and dependent variable
18 X = df[["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]]
19 y = df["label"]
20
21 # Split the dataset into train and test
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
23
24 # Feature scaling
25 sc = StandardScaler()
26 X_train = sc.fit_transform(X_train)
27 X_test = sc.transform(X_test)
28
29 # Instantiate the model
30 classifier = RandomForestClassifier()
31
32 # Fit the model
33 classifier.fit(X_train, y_train)
34
35 # Make pickle file of our model
36 pickle.dump(classifier, open("Crop-Prediction.pkl", "wb"))
```

The bottom panel shows the 'Run' output for 'Crop-Prediction'. The output text is:

```
label      0
dtype: int64
Total number of null values in the DataFrame: 0
Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'Crop-Prediction.pkl', the time is 25:22, and the encoding is UTF-8 with 4 spaces and Python 3.12.

Created HTML file:

```
<!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=Acimo" 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;
}
```

```

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

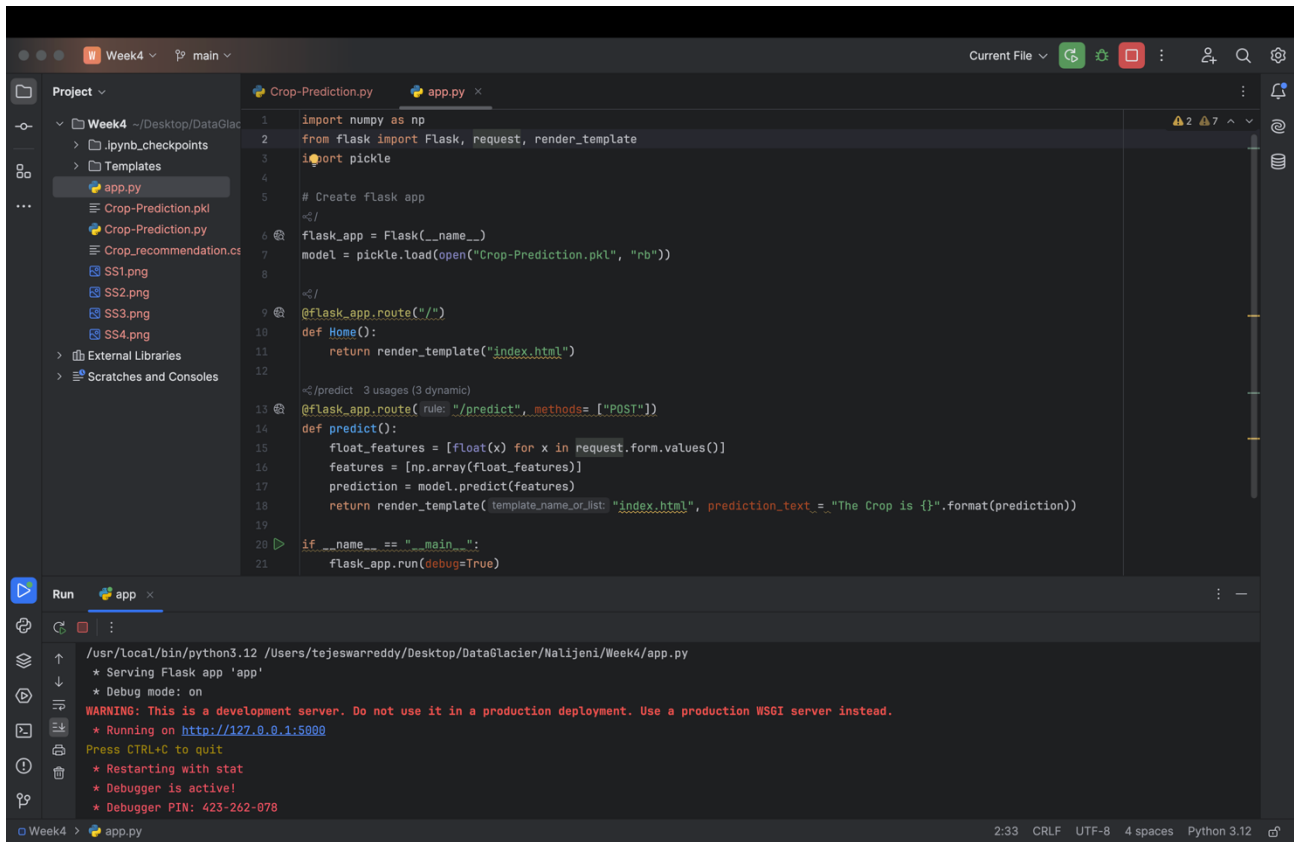
<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}" method="post">
<input type="text" name="N" placeholder="Nitrogen (N)" required="required" />
<input type="text" name="P" placeholder="Phosphorus (P)" required="required" />
<input type="text" name="K" placeholder="Potassium (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:

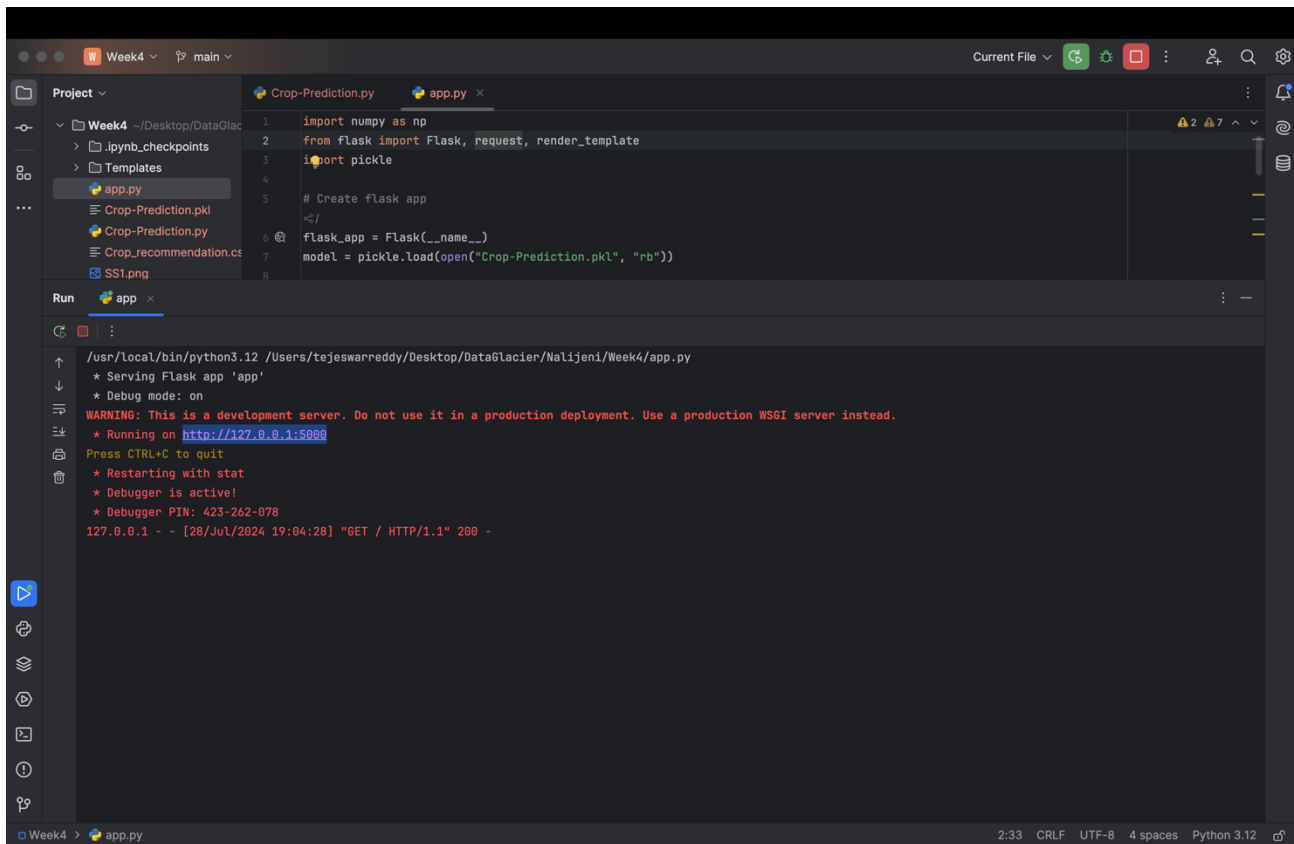


```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 # Create flask app
6 <!--
7 flask_app = Flask(__name__)
8 model = pickle.load(open("Crop-Prediction.pkl", "rb"))
9
10 <!--
11 @flask_app.route("/")
12 def Home():
13     return render_template("index.html")
14
15 <!--/predict 3 usages (3 dynamic)
16 @flask_app.route("/predict", methods= ["POST"])
17 def predict():
18     float_features = [float(x) for x in request.form.values()]
19     features = [np.array(float_features)]
20     prediction = model.predict(features)
21     return render_template(template_name_or_list: "index.html", prediction_text= "The Crop is {}".format(prediction))
22
23 if __name__ == "__main__":
24     flask_app.run(debug=True)
```

Run app x

/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



```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 # Create flask app
6 <!--
7 flask_app = Flask(__name__)
8 model = pickle.load(open("Crop-Prediction.pkl", "rb"))
9
10 <!--
11 @flask_app.route("/")
12 def Home():
13     return render_template("index.html")
14
15 <!--/predict 3 usages (3 dynamic)
16 @flask_app.route("/predict", methods= ["POST"])
17 def predict():
18     float_features = [float(x) for x in request.form.values()]
19     features = [np.array(float_features)]
20     prediction = model.predict(features)
21     return render_template(template_name_or_list: "index.html", prediction_text= "The Crop is {}".format(prediction))
22
23 if __name__ == "__main__":
24     flask_app.run(debug=True)
```

Run app x

/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:

The screenshot shows a web browser window with the title "Crop Prediction". The address bar displays "127.0.0.1:5000". The main content area features a white card with the heading "Crop Prediction" in a cursive font. Below the heading are eight light blue input fields containing the values: 32, 140, 200, 20, 90, 5, and 120. A green "Predict" button is positioned at the bottom of the card.

The screenshot shows the same web browser window, but the address bar now displays "127.0.0.1:5000/predict". The input fields on the white card are labeled: Nitrogen (N), Phosphorus (P), Potassium (K), Temperature (°C), Humidity (%), pH, and Rainfall (mm). A green "Predict" button is at the bottom of the card. Below the button, the text "The Crop is [apple]" is displayed.