**CODE FOR INTERFACE**

**index.html**

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF-8">

<title>Crop Reccomendation</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/bootstrap.min.css') }}">

<script src="{{ url\_for('static', filename='js/bootstrap.min.js')}}"></script>

</head>

<body>

<div class="container" style="margin-top: 120px;">

<div class="card col-md-8 mx-auto shadow-lg form">

<div class="card-body">

<h4 align="center">Crop Recommendation</h4>

<form method="post" action="{{ url\_for('predict')}}">

<div class="card col-md-12 mx-auto py-4" style="background-color: bisque;">

<input type="text" name="nitro" placeholder="Enter Nitrogen Value" class="form-control my-2">

<input type="text" name="phosporous" placeholder="Enter Phosporous Value" class="form-control my-2">

<input type="text" name="potassium" placeholder="Enter Potassium Value" class="form-control my-2">

<input type="text" name="temp" placeholder="Enter Temperature" class="form-control my-2">

<input type="text" name="humid" placeholder="Enter Humidity in %" class="form-control my-2">

<input type="text" name="ph" placeholder="Enter pH value" class="form-control my-2">

<input type="text" name="rainfall" placeholder="Enter rainfall value" class="form-control my-2">

<div class="row">

<div class="col">

<select name="soiltype" id="soiltype">

<option value="">Select Soil Type</option>

<option value="0">Black</option>

<option value="1">Clayey</option>

<option value="2">Loamy</option>

<option value="3">Red</option>

<option value="4">Sandy</option>

</select>

</div>

<div class="col">

<select name="weather" id="weather">

<option value="">Select Weather</option>

<option value="0">Cloudy</option>

<option value="1">Rainy</option>

<option value="2">Sunny</option>

</select>

</div>

<div class="col">

<select name="location" id="location">

<option value="">Select District</option>

<option value="0">Ananthapur</option>

<option value="1">Chittoor</option>

<option value="2">East Godavari</option>

<option value="3">Guntur</option>

<option value="4">Krishna</option>

<option value="5">Kurnool</option>

<option value="6">Prakasam</option>

<option value="7">Nellore</option>

<option value="8">Visakhapatnam</option>

<option value="9">West Godavari</option>

</select>

</div>

</div>

<div class="row mt-4">

<div class="col">

<input type="submit" value="Submit" class="btn btn-outline-success form-control">

</div>

</div>

</form>

{{ prediction\_text }}

</div>

</div>

</div>

</body>

</html>

**flask\_api.py**

from flask import Flask, request, render\_template

import numpy as np

from sklearn.preprocessing import StandardScaler

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

app = Flask(\_name\_)

df = pd.read\_csv('Crop\_recommendation.csv')

encode\_soil = LabelEncoder()

#fitting the label encoder

df.soiltype = encode\_soil.fit\_transform(df.soiltype)

#creating the DataFrame

soiltype=pd.DataFrame(zip(encode\_soil.classes\_,encode\_soil.transform(encode\_soil.classes\_)),columns=['Original','Encoded'])

soiltype = soiltype.set\_index('Original')

encode\_weather = LabelEncoder()

#fitting the label encoder

df.weather = encode\_weather.fit\_transform(df.weather)

#creating the DataFrame

weather=pd.DataFrame(zip(encode\_weather.classes\_,encode\_weather.transform(encode\_weather.classes\_)),columns=['Original','Encoded'])

weather = weather.set\_index('Original')

encode\_location = LabelEncoder()

#fitting the label encoder

df.location = encode\_location.fit\_transform(df.location)

#creating the DataFrame

location=pd.DataFrame(zip(encode\_location.classes\_,encode\_location.transform(encode\_location.classes\_)),columns=['Original','Encoded'])

location = location.set\_index('Original')

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

scaler = StandardScaler()

X = scaler.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42,shuffle=True,stratify=y)

model=RandomForestClassifier(n\_estimators=1000,random\_state=123)

model.fit(X\_train,y\_train)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict',methods=['POST'])

def predict():

int\_features = [float(x) for x in request.form.values()]

final\_features = np.array([int\_features])

print(final\_features)

data = scaler.transform(final\_features)

prediction = model.predict\_proba(data)

print(prediction)

top\_5\_crops\_indices = np.argsort(prediction[0])[::-1][:5]

print(top\_5\_crops\_indices)

top\_5\_crops = [model.classes\_[idx] for idx in top\_5\_crops\_indices]

print(top\_5\_crops)

return render\_template('index.html', prediction\_text='Top 5 ideal crops to grow are: {}'.format(', '.join(top\_5\_crops)))

if \_name\_ == "\_main\_":

app.run(debug=True)

**OUTPUT:**