Opticrop: Smart Agricultural Production Optimization Engine

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

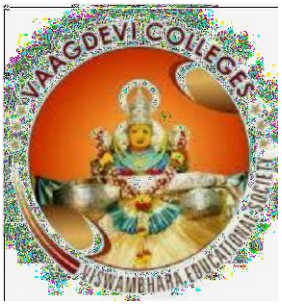
Submitted By

LINGAMPALLI SREENITHA	**21UK1A05H5**
MORA SAIROSHAN	**21UK1A05F8**
THOTA YASHASWI	**21UK1A05D8**
VENDI PAVAN	**21UK1A05E5**

Under the guidance of

D.RAKESH DATTA

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

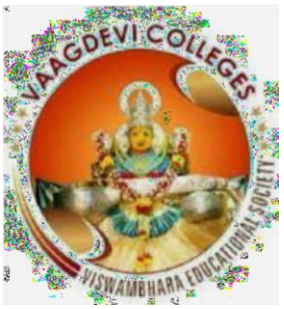
Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (TELANGANA)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)



CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "OPTICROP:SMART AGRICULTURAL PRODUCTION OPTIMIZATION " is being submitted by LIGAMPALLI SREENITHA(21UK1A05H5),MORASAIROSHAN(21UK1A05F8),THOTAYASHASWI(21UK1A05D8),VENDIPAVAN(21UK1A05E5) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2021- 2025.

Project Guide
D.RAKESH DATTA
(Assistant Professor)

HOD
DR.NAVEENKUMAR
(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr.NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **D.RAKESH DATTA,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

LINGAMPALLI SREENITHA	**21UK1A05H5**
MORA SAIROSHAN	**21UK1A05F8**
THOTA YASHASWI	**21UK1A05D8**
VENDI PAVAN	**21UK1A05E5**

ABSTRACT

The "Opti Crop: Smart Agricultural Production Optimization Engine" project aims to develop an advanced software system that leverages data-driven insights to enhance agricultural production for a variety of crops. By incorporating key environmental factors, such as Nitrogen (N), Phosphorous (P), Potassium (K) levels, soil temperature, humidity, pH, rainfall, and crop types, Opti Crop provides intelligent recommendations to farmers, thereby maximizing yields and resource efficiency. This innovative project not only assists farmers but also serves agricultural researchers, policymakers, and stakeholders by offering valuable knowledge and insights into crop-environment interactions, ultimately informing evidence-based agricultural strategies.

In the face of challenges like climate change, population growth, food security concerns, and the vulnerabilities exposed by the COVID-19 crisis, the agricultural industry is in urgent need of innovative approaches to improve crop yields sustainably. Opti Crop addresses these challenges by enabling farmers to produce more with fewer resources through advanced data science and artificial intelligence techniques.

The project employs classification algorithms such as K-Nearest Neighbors (KNN), K-Means, and Logistic Regression. These algorithms will be used to train and test data, and the best-performing model will be selected and saved in a .pkl format. Further, the system will be integrated with Flask for web application development and deployed using IBM's cloud services.

Opti Crop promises to revolutionize agricultural practices by providing a smart production optimization engine, empowering farmers to achieve optimal yields, enhance resource efficiency, and support.

TABLE OF CONTENTS:-

1. INTRODUCTION

1.1. OVERVIEW

Agricultural production faces numerous challenges in the modern era, driven by factors such as climate change, population growth, and food security concerns. The recent COVID-19 crisis has further highlighted the vulnerabilities within the agricultural landscape, intensifying the need for sustainable solutions to meet global food demands. In this context, achieving efficiency in agricultural practices— producing more with fewer resources—has become more critical than ever.

The "Opti Crop: Smart Agricultural Production Optimization Engine" project aims to address these pressing challenges by developing an advanced software system that leverages data-driven insights to optimize crop yields. By integrating essential environmental factors such as Nitrogen (N), Phosphorous (P), Potassium (K) levels, soil temperature, humidity, pH, rainfall, and crop types, Opti Crop provides intelligent recommendations tailored to farmers' specific needs. This innovative approach ensures that farmers can maximize yields and improve resource efficiency, contributing to sustainable agricultural practices.

The agricultural industry stands on the brink of transformation through the application of data science and artificial intelligence. Opti Crop embodies this transformation by employing sophisticated classification algorithms, including K-Nearest Neighbors (KNN), K-Means, and Logistic Regression, to analyze and interpret complex agricultural data. By training and testing these algorithms, the project identifies the bestperforming model, which is then saved in a .pkl format for practical application.

To ensure accessibility and usability, the Opti Crop system will be integrated with Flask for seamless web application development and deployed using IBM's cloud services. This integration facilitates easy access to the system's recommendations, empowering farmers with the tools they need to achieve optimal yields and sustainable agricultural outcomes.

Ultimately, the Opti Crop project aims to revolutionize agricultural practices by providing a smart production optimization engine. This engine not only supports farmers in enhancing their productivity and

resource efficiency but also provides valuable insights to agricultural researchers, policymakers, and stakeholders. Through Opti Crop, the understanding of crop-environment interactions will advance, informing the development of evidence-based agricultural strategies and promoting a sustainable future for global agriculture.

1.2. PURPOSE

The purpose of the "Opti Crop: Smart Agricultural Production Optimization Engine" project is multifaceted, focusing on advancing agricultural productivity and sustainability through innovative, datadriven approaches. The key objectives include:

Enhancing Agricultural Efficiency

-Optimizing Crop Yields: Provide farmers with intelligent, tailored recommendations based on comprehensive environmental data to maximize crop yields.

-Resource Efficiency: Help farmers use resources such as water, fertilizers, and land more efficiently, reducing waste and costs.

Addressing Global Challenges

-Climate Change Mitigation: Develop strategies to adapt to and mitigate the effects of climate change on agricultural production.

-Food Security: Ensure a reliable and sustainable food supply by improving agricultural practices and productivity in the face of population growth and global crises like the COVID-19 pandemic.

Leveraging Technology

-Data Science and AI: Utilize advanced classification algorithms (K-Nearest Neighbors, K-Means, Logistic Regression) to analyze agricultural data, identify patterns, and generate actionable insights.

-Model Development and Deployment: Create robust models to predict optimal farming practices, save them in a. pkl format for practical use, and integrate them into a user-friendly web application through Flask and IBM cloud deployment.

Supporting Stakeholders

-**Farmers:** Empower farmers with tools and knowledge to achieve better yields and sustainable practices.

-**Researchers and Policymakers:** Provide valuable insights into crop-environment interactions, aiding the development of evidence-based agricultural strategies.

-**Agricultural Industry:** Foster a data-driven transformation within the industry, promoting innovative and efficient farming techniques.

Promoting Sustainability

-Environmental Stewardship: Encourage sustainable agricultural practices that protect and preserve natural resources.

-Long-term Viability: Contribute to the development of a resilient agricultural system capable of withstanding future challenges.

By achieving these purposes, the Opti Crop project aims to revolutionize agricultural practices, ensuring a more efficient, sustainable, and secure food production system for the future.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The development of the "Opti Crop: Smart Agricultural Production Optimization Engine" is motivated by several existing problems in the agricultural sector:

Inefficient Resource Utilization

-Overuse of Fertilizers and Water: Many farmers lack precise data on how much fertilizer or water is needed, leading to overuse, waste, and environmental harm

-Soil Degradation: Inaccurate or excessive use of resources can degrade soil quality, reducing its long-term productivity.

Climate Change Impact

-Unpredictable Weather Patterns: Climate change is causing more frequent and severe weather events, making it difficult for farmers to plan and optimize their crop production.

-Temperature Variations: Fluctuating temperatures affect crop growth and yield, requiring more adaptive and resilient farming practices.

Population Growth and Food Security

-Increasing Food Demand: The growing global population demands higher food production, putting pressure on existing agricultural systems.

-Food Security Concerns: Ensuring a reliable and sufficient food supply is becoming more challenging due to various global crises, including pandemics.

Lack of Data-Driven Decision Making

-Traditional Farming Practices: Many farmers still rely on traditional practices and intuition rather than data- driven insights, leading to suboptimal yields.

-Limited Access to Technology: Smallholder farmers, in particular, may lack access to advanced technologies and data analytics tools that can improve their productivity.

Environmental Sustainability

-Resource Depletion: Unsustainable agricultural practices contribute to the depletion of natural resources, such as water and soil nutrients.

-Environmental Pollution: Excessive use of fertilizers and pesticides can lead to pollution of water bodies and harm to ecosystems.

Vulnerability Exposed by Global Crises

-COVID-19 Impact: The pandemic has highlighted the fragility of global food systems, disrupting supply chains and affecting agricultural productivity.

-Supply Chain Disruptions: Crises can lead to interruptions in the supply of essential agricultural inputs, further complicating production efforts.

Technological Barrier

-Complexity of Data Integration: Integrating various environmental factors (e.g., soil nutrients, weather data) into a cohesive system is complex and requires advanced data science techniques.

-Algorithm Selection: Determining the most effective algorithms for analyzing agricultural data and providing actionable insights is a significant challenge.

By addressing these existing problems, the Opti Crop project aims to provide farmers with the tools and knowledge needed to optimize their production, use resources more efficiently, and contribute to a more sustainable and resilient agricultural system.

2.2 PROPOSED SOLLUTION

OptiCrop is an intelligent platform designed to optimize agricultural production by leveraging cuttingedge technologies such as machine learning, IoT (Internet of Things), and data analytics. The engine aims to maximize crop yields, reduce resource usage, and ensure sustainable farming practices.

1. Real-Time Monitoring and Alerts

- Deploy IoT sensors and drones in fields to continuously monitor soil and crop conditions.
- Integrate weather data to provide timely alerts about adverse weather conditions.
- Use AI algorithms to detect anomalies and send real-time alerts to farmers' mobile devices.

2. Predictive Analytics for Crop Management

- Develop machine learning models to predict pest infestations and disease outbreaks.
- Analyze historical and real-time data to forecast optimal irrigation and fertilization schedules.
- Use yield prediction models to estimate production and plan harvests.

3. Resource Optimization

- Implement algorithms to optimize water usage based on soil moisture levels and weather forecasts.
- Recommend precise fertilizer application based on nutrient analysis to minimize waste and environmental impact.
- Provide strategies for efficient pest and disease management using integrated pest management (IPM) techniques.

4. Decision Support System

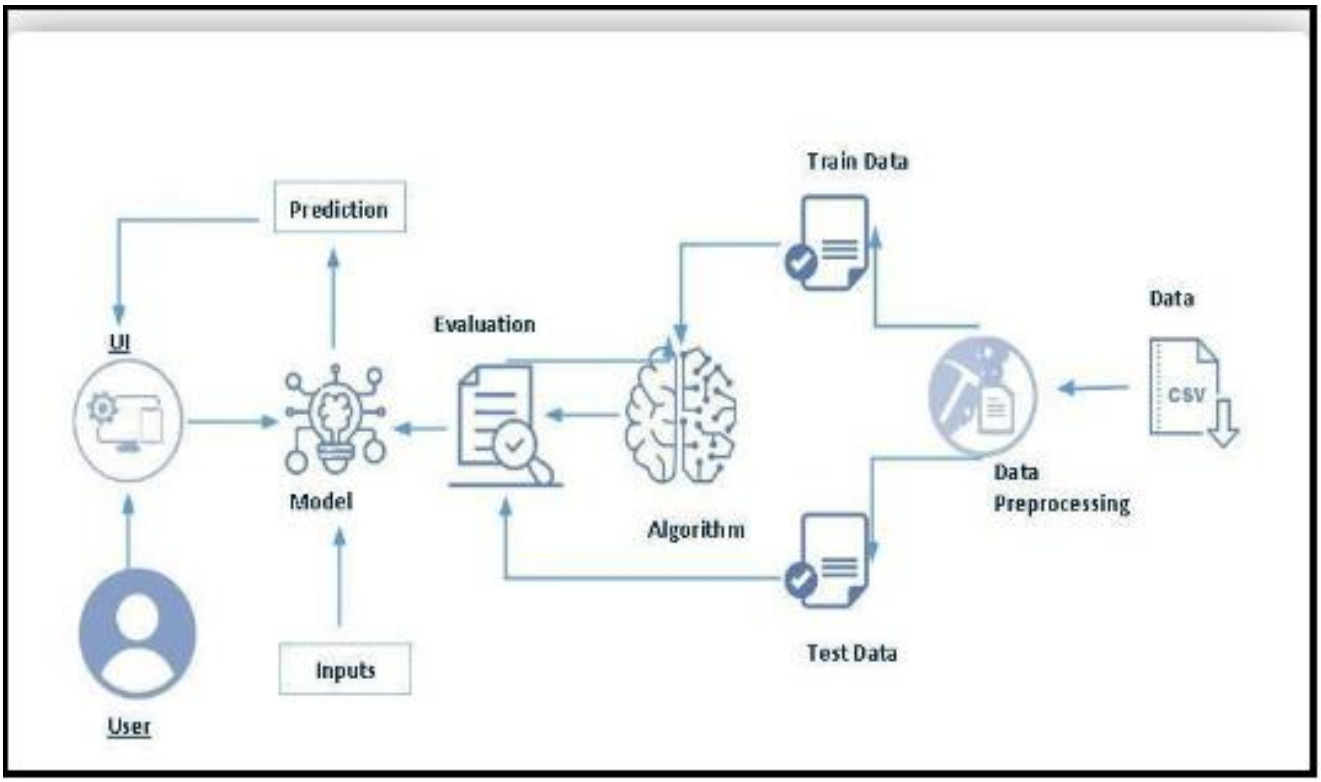
- Develop a decision support system that integrates all data sources and provides actionable insights.
- Offer a centralized platform for farmers to manage their operations, track progress, and make informed decisions.
- Provide scenario analysis tools to evaluate the impact of different farming practices and choose the best options.

5. User-Friendly Applications

- Design intuitive mobile and web applications for farmers to access data, receive alerts, and implement recommendations.
- Create a customizable dashboard for real-time monitoring and analysis of farm operations.

3.THEORITICAL ANALYSIS

2.1. BLOCK DIAGRAM



3.2.SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

- **Dataset (CSV File)**: The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.

- **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

- **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

- **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the SMART CROP OPTIMIZATION prediction task.

- **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict SMART CROP OPTIMIZATION categories based on historical data.

- **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view SMART CROP OPTIMIZATION predictions, health information, and recommended precautions.

Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the SMART CROP OPTIMIZATION predictions and associated health information.

4.EXPERIMENTAL INVESTIGATION

In this project, we have used SMART CROP OPTIMIZATION Dataset. This dataset is a csv file consisting of labelled data and having the following columns

- ## Experimental Investigation for OptiCrop

Objective: The objective of the experimental investigation is to evaluate the effectiveness and efficiency of the OptiCrop system in optimizing agricultural production. The investigation aims to measure improvements in crop yields, resource usage efficiency, cost savings, and sustainability practices.

Experimental Design

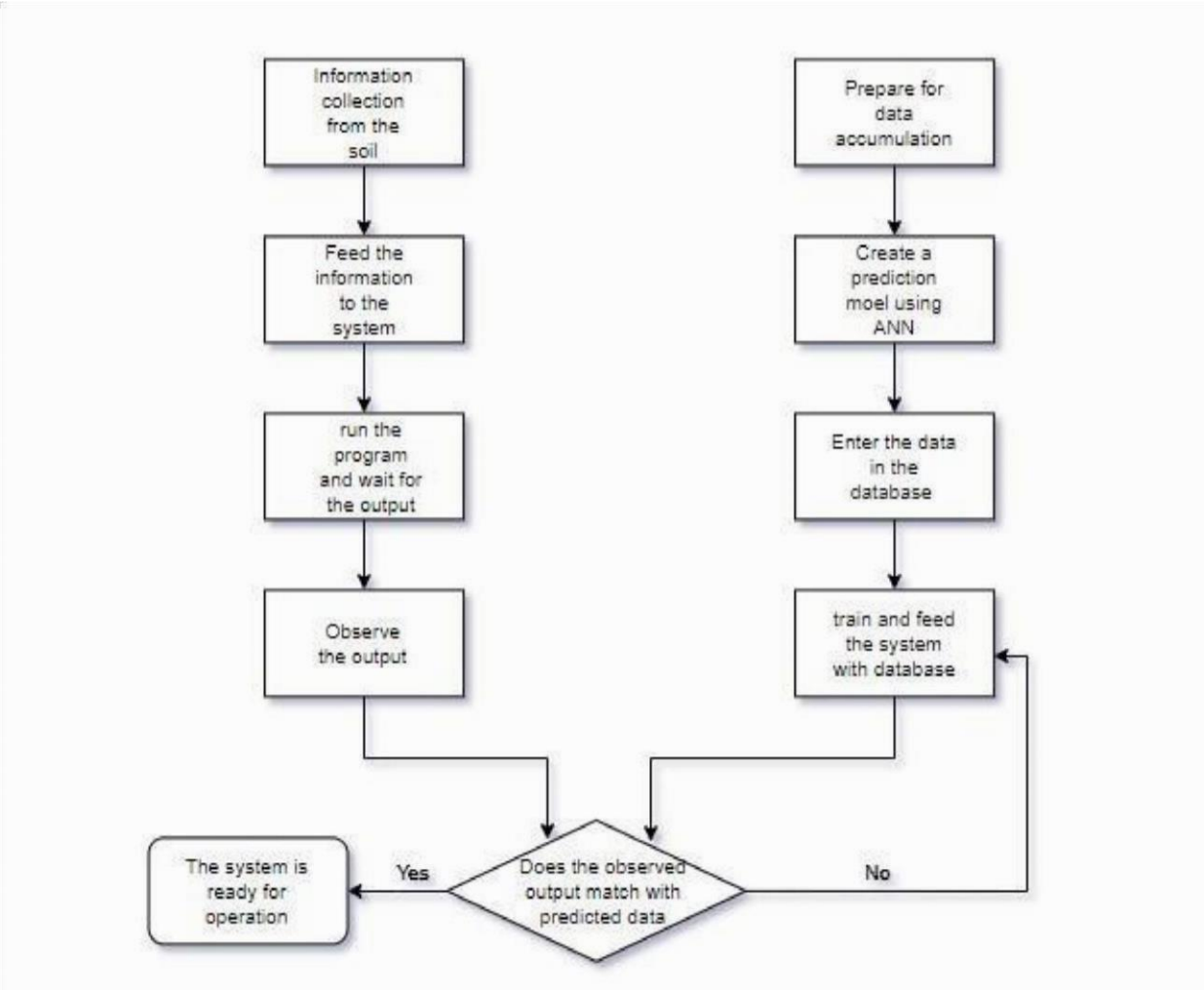
 # 1. Selection of Test Sites

- **Diverse Locations:** Select multiple test sites with varying climatic conditions, soil types, and crop varieties to ensure comprehensive evaluation.
- **Control and Test Groups:** Establish control groups (farms using traditional practices) and test groups (farms using OptiCrop

2. Baseline Data Collection

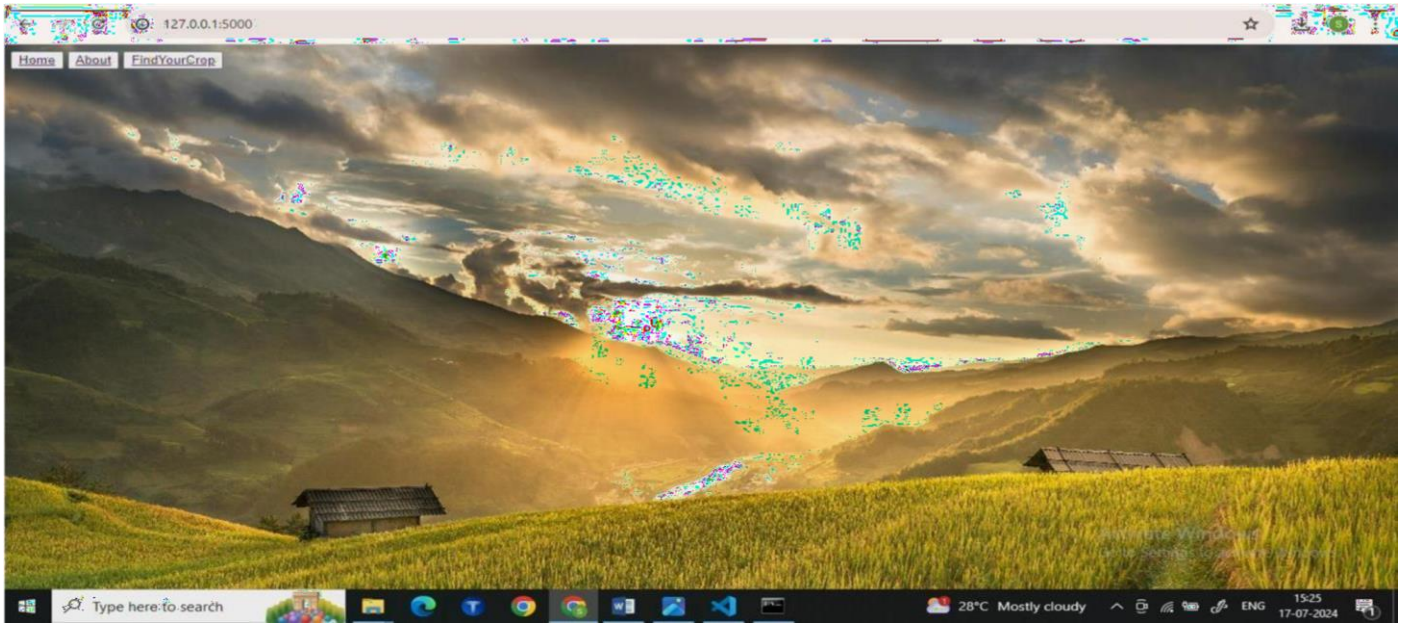
- **Initial Assessment:** Conduct an initial assessment of soil health, crop condition, and resource usage in both control and test groups

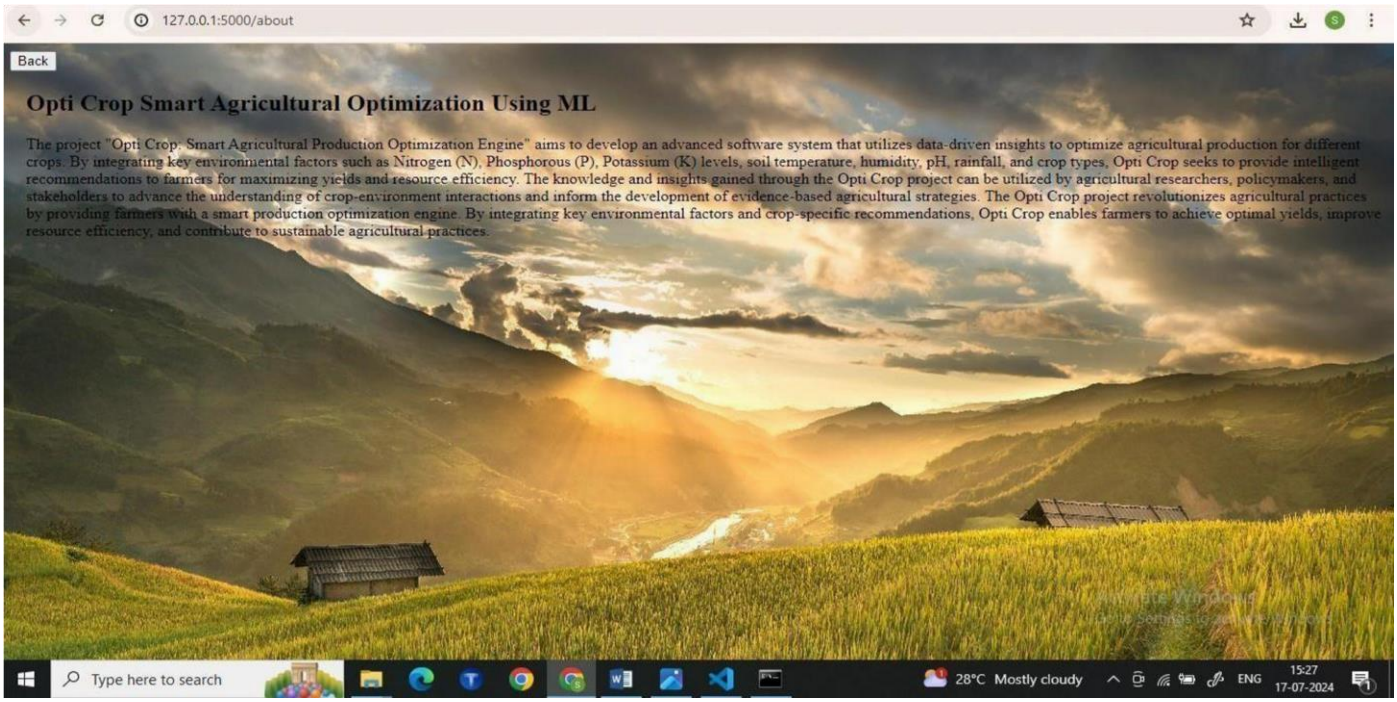
5.FLOWCHART


Information collection from the soil
Feed the information to the system
run the program and wait for the output
Observe the output
Prepare for data accumulation
Create a prediction moel using ANN
Enter the data in the database
train and feed the system with database
Does the observed output match with predicted data
Yes
No
The system is ready for operation

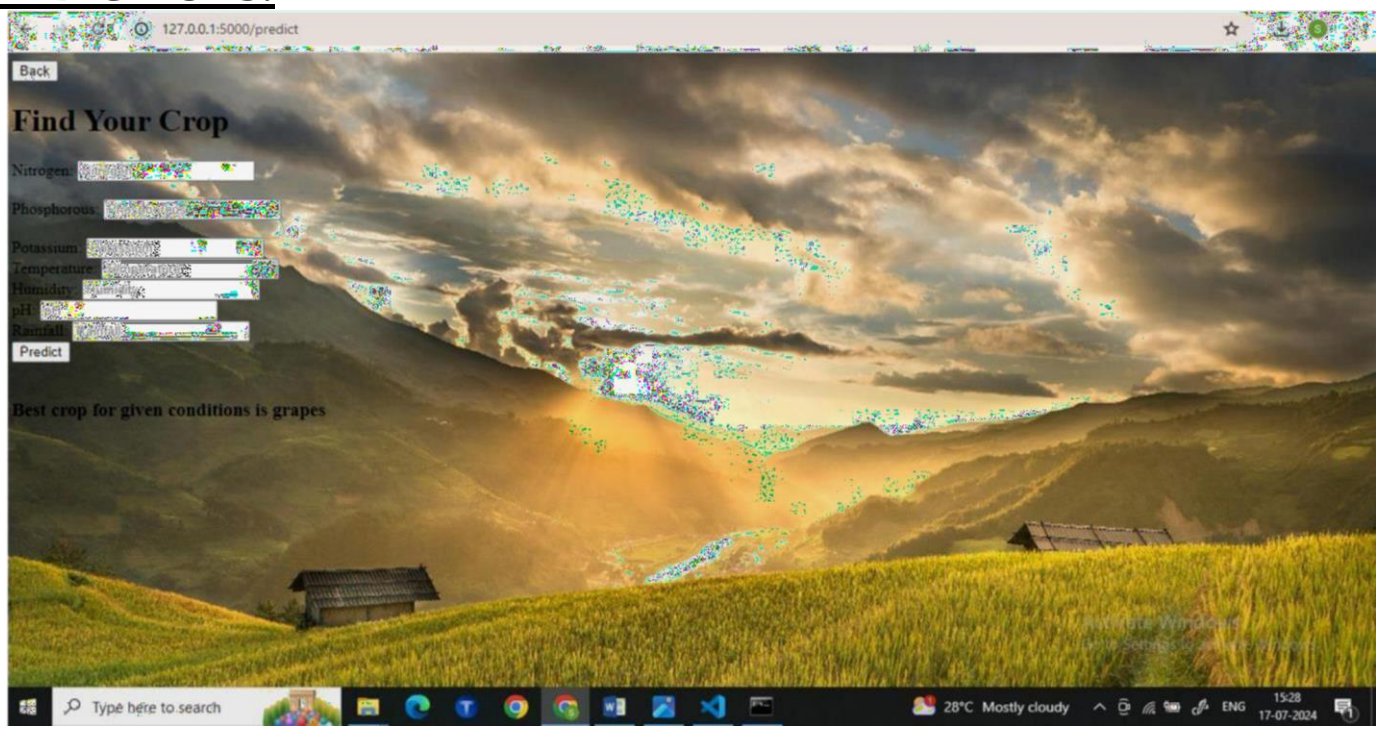
6.RESULT

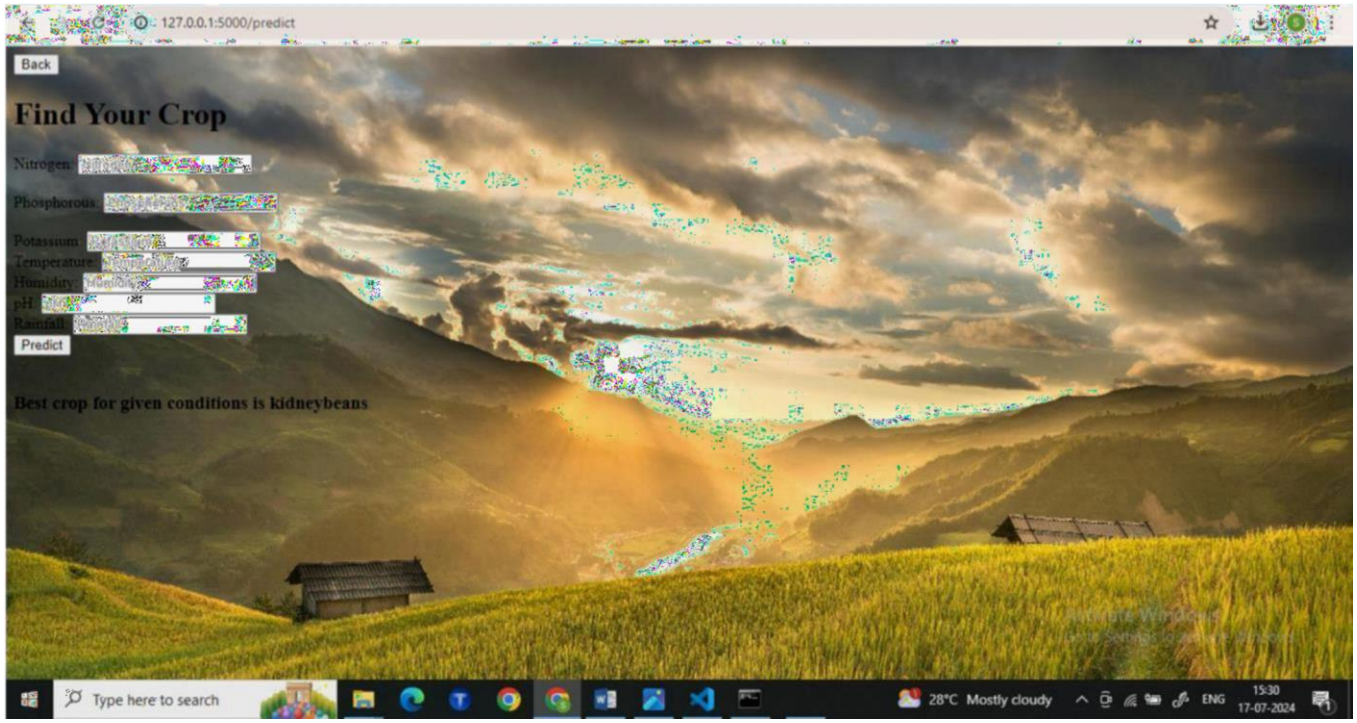
HOME PAGE





PREDICTIONS:





7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- **Increased Crop Yields**
- **Precision Agriculture**: OptiCrop provides precise recommendations for irrigation, fertilization, and pest control, maximizing crop yields.
- **Predictive Analytics**: Forecasts optimal planting times and crop varieties to enhance productivity.
- **Resource Efficiency**
- **Optimized Water Usage**: Real-time soil moisture monitoring and weather data integration help optimize irrigation schedules, reducing water wastage.
- **Efficient Fertilization**: Precise nutrient level analysis ensures accurate fertilizer application, minimizing waste and environmental impact.
- **Cost Savings**
- **Reduced Input Costs**: By optimizing the use of water, fertilizers, and pesticides, OptiCrop reduces overall input costs for farmers.
- **Efficient Pest Management**: Early detection and precise control of pests and diseases reduce losses and treatment costs.
- **Sustainability**
- **Environmentally Friendly Practices**: Promotes sustainable farming by minimizing the use of chemicals and conserving water resources.
- **Soil Health Management:** Continuous monitoring and recommendations help maintain and improve soil health over time
- **Real-Time Monitoring and Alerts**
- **Continuous Data Collection**: IoT sensors and drones provide real-time data on soil and crop conditions.

DISADVANTAGES :

- **High Initial Investment**
- **Cost of Technology:** The initial setup cost for IoT sensors, drones, and other hardware can be high, which may be a barrier for small-scale farmers.
- **Software and Subscription Fees:** Ongoing costs for software licenses, data subscriptions, and maintenance can add up.

- **Technical Expertise Required**
- **Learning Curve:** Farmers may need training and time to learn how to effectively use the technology.
- **Technical Support:** Ongoing technical support is required to address issues and ensure smooth operation, which may not always be readily available in rural areas.

- **Connectivity Issues**
- **Internet Access**: Reliable internet connectivity is essential for real-time data transmission and analysis, which can be a challenge in remote areas.
- **Power Supply:** Continuous power supply is needed to keep the sensors and other devices operational, which may be unreliable in some regions.

- **Data Privacy and Security**
- **Data Vulnerability:** The data collected by OptiCrop could be vulnerable to cyberattacks, leading to potential misuse or loss of sensitive information.
- **Privacy Concerns:** Farmers may have concerns about the privacy of their data and how it is used and stored.

8.APPLICATIONS

- **Precision Agriculture**
- **Irrigation Management**: Optimize irrigation schedules based on real-time soil moisture data and weather forecasts to conserve water and improve crop health.

- **Fertilizer Application**: Precisely manage fertilizer application by analyzing soil nutrient levels, reducing waste and environmental impact.

- **Pest and Disease Control**: Early detection and management of pests and diseases using predictive analytics and real-time monitoring.

- **Crop Monitoring and Management**
- **Growth Tracking**: Monitor crop growth stages and health using drone and satellite imagery, allowing for timely interventions.

- **Yield Prediction**: Use machine learning models to predict crop yields, helping farmers plan for harvest and market supply.

- **Field Mapping**: Create detailed field maps to identify variations in soil and crop conditions, guiding targeted interventions.

- **Resource Optimization**
- **Water Resource Management**: Efficiently manage water resources by monitoring usage and optimizing irrigation practices.
- **Energy Management**: Optimize energy usage in farming operations by integrating renewable energy sources and monitoring consumption.

- **Input Efficiency**: Reduce the usage of inputs like seeds, fertilizers, and pesticides by applying them precisely where needed.

- **Farm Management and Planning**
- **Operational Planning**: Plan and manage daily farming operations based on real-time data and predictive insights.

- **Resource Allocation**: Allocate resources efficiently by understanding the specific needs of different parts of the farm.

- **Financial Planning**: Make informed financial decisions based on yield predictions, cost savings from optimized resource usage, and market trends.

9.CONCLUSION

The implementation of OptiCrop: Smart Agricultural Production Optimization Engine offers a multitude of advantages and applications that significantly enhance agricultural practices. By integrating cuttingedge technologies such as IoT, AI, and machine learning, OptiCrop provides precise, data-driven insights that optimize resource usage, increase crop yields, and promote sustainable farming.

OptiCrop stands at the forefront of modern agricultural innovation, offering a comprehensive solution that addresses the challenges faced by today's farmers. While there are some disadvantages, such as the initial investment costs and the need for technical expertise, the benefits far outweigh these challenges. By leveraging OptiCrop, farmers can achieve higher yields, reduce costs, and adopt sustainable practices, ultimately leading to increased profitability and a positive environmental impact. As the agricultural sector continues to evolve, tool like OptiCrop will play a crucial role in shaping the future of farming, ensuring food security, and promoting sustainable development.

10.FUTURE SCOPE:

Future Scope of the SMART CROP OPTIMIZATION Prediction and Management System:

Predictive Modeling: Enhancements in AI and machine learning will lead to more accurate predictive models for weather patterns, pest outbreaks, and yield forecasts.

- **Adaptive Algorithms**: Development of algorithms that adapt in real-time to changing conditions ,providing even more precise recommendations.

- **Adaptive Algorithms**: Development of algorithms that adapt in real-time to changing conditions, providing even more precise recommendations.

- **Next-Gen Sensors**: Improved sensor technology will provide more accurate and diverse data, including soil health indicators, plant health metrics, and environmental conditions.

- **Drone and Satellite Integration**: Enhanced integration of drones and satellites for real-time, large-scale monitoring and data collect.

11.APPENDIX

Model building :

1) Dataset

2) Google colab and VS code Application Building

I. HTML file (Index file, Predict file) II. CSS file

III. Models in pickle format

SOURCE CODE:

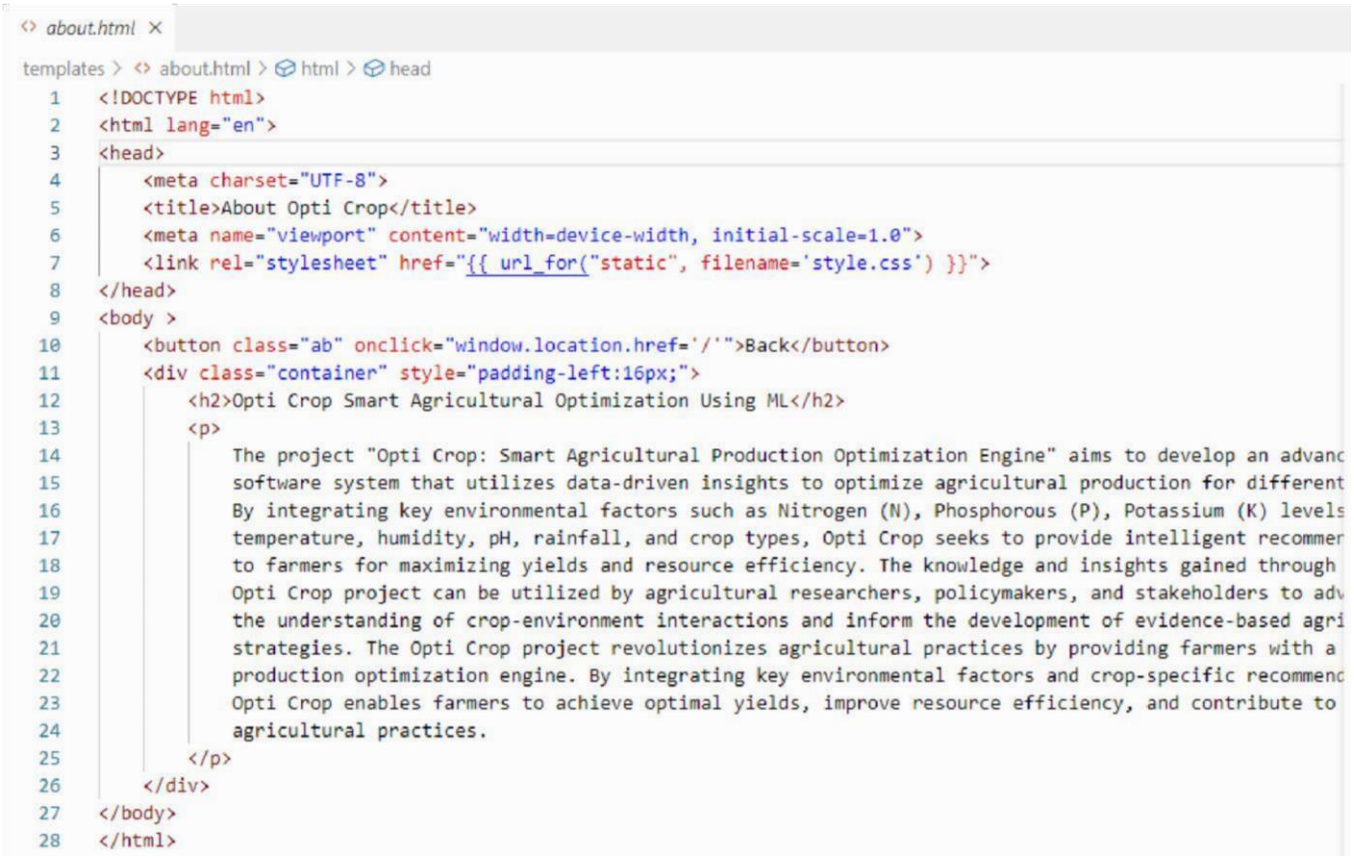
HOME.HTML

home.html

templates > home.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>About Opti Crop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for("static", filename='style.css') }}">
</head>
<body >
    <div class="topnav">
        <button><a class="active" href="{{ url_for('home') }}">Home</a></button>
        <button><a href="{{ url_for('about') }}">About</a></button>
        <button><a href="{{ url_for('findyourcrop') }}">FindYourCrop</a></button>
    </div>
</body>
</html>
```

ABOUT.HTML



FIND YOUR CROP.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Find Your Crop</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for("static", filename='style.css') }}">
</head>
    </body>
    <button class="ab" onclick="window.location.href='/'">Back</button>
<div class="container">
        <h1 class="text-center">Find Your Crop</h1>
```

```
        <div class="row justify-content-center">
            <div class="col-md-6">
                <form action="{{ url_for('predict') }}" method="post">
                    <div class="form-group">
                        <label for="nitrogen">Nitrogen:</label>
<input type="text" id="nitrogen" name="nitrogen" class="formcontrol"
placeholder="Nitrogen" required="required"/><br><br>
                    </div>
                    <div class="form-group">
                        <label for="phosphorous">Phosphorous:</label>
                        <input type="text" id="phosphorous" name="phosphorous"
class="form-control" placeholder="Phosphorous" required="required"/><br><br>
                    </div>
                    <div class="form-group">
                       <label for="potassium">Potassium:</label>
                        <input type="text" id="potassium" name="potassium"
class="formcontrol" placeholder="Potassium" required="required"/>
                    </div>
                    <div class="form-group">
                        <label for="temperature">Temperature:</label>
                        <input type="text" id="temperature" name="temperature"
class="form-control" placeholder="Temperature" required="required"/>
                    </div>
                    <div class="form-group">
                        <label for="humidity">Humidity:</label>
                        <input type="text" id="humidity" name="humidity" class="form-
control" placeholder="Humidity" required="required"/>
                    </div>
                    <div class="form-group">
                              <label for="ph">pH:</label>
                         <input type="text" id="ph" name="ph" class="form-control"
placeholder="pH" required="required"/>
                    </div>
                    <div class="form-group">
                        <label for="rainfall">Rainfall:</label>
                        <input type="text" id="rainfall" name="rainfall"
class="formcontrol" placeholder="Rainfall" required="required"/> </div>
                    <button type="submit" class="btn btn-primary
btnblock">Predict</button>  </form>
                <br>
```

```
                <h3>
                    <p class="flask-output">{{ prediction_text }}</p>
                </h3>
            </div>
        </div>
    </div>
</body>
</html>
```

CSS FILE

```css
body{
    background-image: url('ss.jpg');
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
}
.a{
    text-align: center;
}
.topnav{
    color: blueviolet;
}
```

APP.PY

```
import numpy as np from flask import Flask,
request, render_template import pickle
 app = Flask(_name_, template_folder='templates')


# Load the model (assuming model.pkl is in the same directory) try:
  model = pickle.load(open('model.pkl', 'rb'))
print("Model loaded successfully!") except
FileNotFoundError:
  print("Error: Model file 'model.pkl' not found!")
exit(1) # Exit the program if model is not found


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


@app.route('/about')    def    about():
return render_template("about.html")


@app.route("/findyourcrop") def findyourcrop():
  return render_template('findyourcrop.html')
@app.route('/predict',  methods=['POST'])
def predict(): # Extract form data try:
    int_features = [float(x) for x in request.form.values()]
except ValueError:
```

```
    print("Error: Invalid data type in form submission!")  return
render_template("findyourcrop.html", prediction_text="Please enter valid numerical
values.")


  # Prepare features for prediction features
= [np.array(int_features)]


  # Make prediction prediction = model.predict(features)
print(prediction)
# Print for debugging


  # Extract output (assuming single value prediction)
output = prediction[0]


  # Render template with prediction return render_template("findyourcrop.html",
prediction_text='Best crop for given conditions is {}'.format(output))
 if __name__== '__main__':
  app.run(debug=True) # Run the Flask app in debug modE
```

CODE SNIPPETS

```
#DATA COLLECTION AND IMPORTING THE LIBRARIES
```

```
import pandas as pd
import numpy as np
pd.set_option('max_colwidth', 20)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 58)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize'] = (12,8)
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
from ipywidgets import interact
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
#READING THE DATASET
```

```
df = pd.read_csv('/content/Crop_recommendation.csv')
df.head()
```

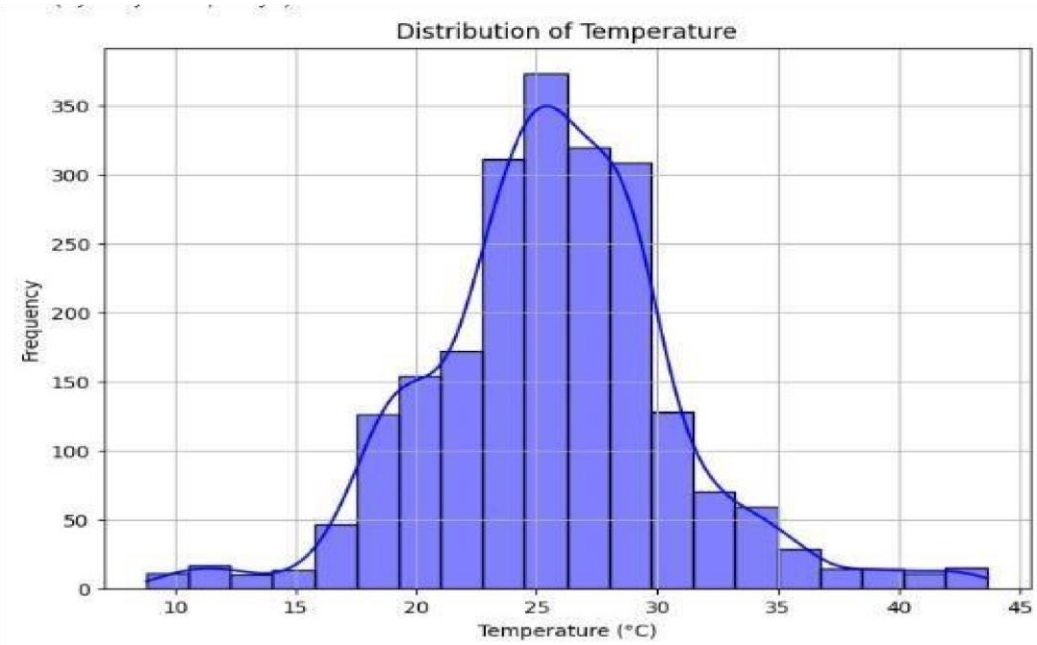
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

```
# UNIVARIATE ANALYSIS
```

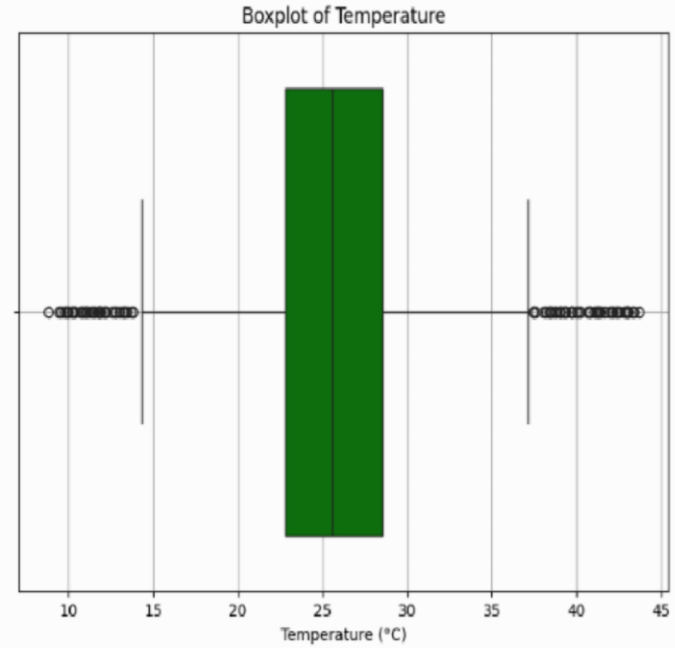
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you have your dataset loaded into a Pandas DataFrame
# Replace 'your_dataset.csv' with your actual dataset filename or path
df = pd.read_csv('/content/Crop_recommendation.csv')

# Check the available columns in your DataFrame
print(df.columns)

# Example: Univariate analysis of a numerical variable (e.g., temperature)
temperature_data = df['temperature']

# Summary statistics
print("Summary Statistics for Temperature:")
print(temperature_data.describe())

# Histogram to visualize the distribution of temperature
plt.figure(figsize=(8, 6))
sns.histplot(temperature_data, bins=20, kde=True, color='blue', edgecolor='black')
plt.title('Distribution of Temperature')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Boxplot to identify outliers in temperature
plt.figure(figsize=(8, 6))
sns.boxplot(x=temperature_data, color='green')
plt.title('Boxplot of Temperature')
plt.xlabel('Temperature (°C)')
plt.grid(True)
plt.show()
```

```
plt.grid(True)
plt.show()

# Example: Univariate analysis of a categorical variable
# Choose an actual categorical column from the available columns
categorical_column = 'label' # Replace 'label' with the actual column name
# Replace 'your_categorical_column' with an actual column from your DataFrame
categorical_data = df[categorical_column]

# Count plot to visualize distribution of categorical data
plt.figure(figsize=(8, 6))
sns.countplot(x=categorical_data, palette='viridis')
plt.title('Distribution of Your Categorical Data')
plt.xlabel(categorical_column)
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
Summary Statistics for Temperature:
count    2200.000000
mean       25.616244
std         5.063749
min         8.825675
25%        22.769375
50%        25.598693
75%        28.561654
max        43.675493
Name: temperature, dtype: float64
<Figure size 800x600 with 0 Axes>
<Axes: xlabel='temperature', ylabel='Count'>
Text(0.5, 1.0, 'Distribution of Temperature')
Text(0.5, 0, 'Temperature (°C)')
Text(0, 0.5, 'Frequency')
```

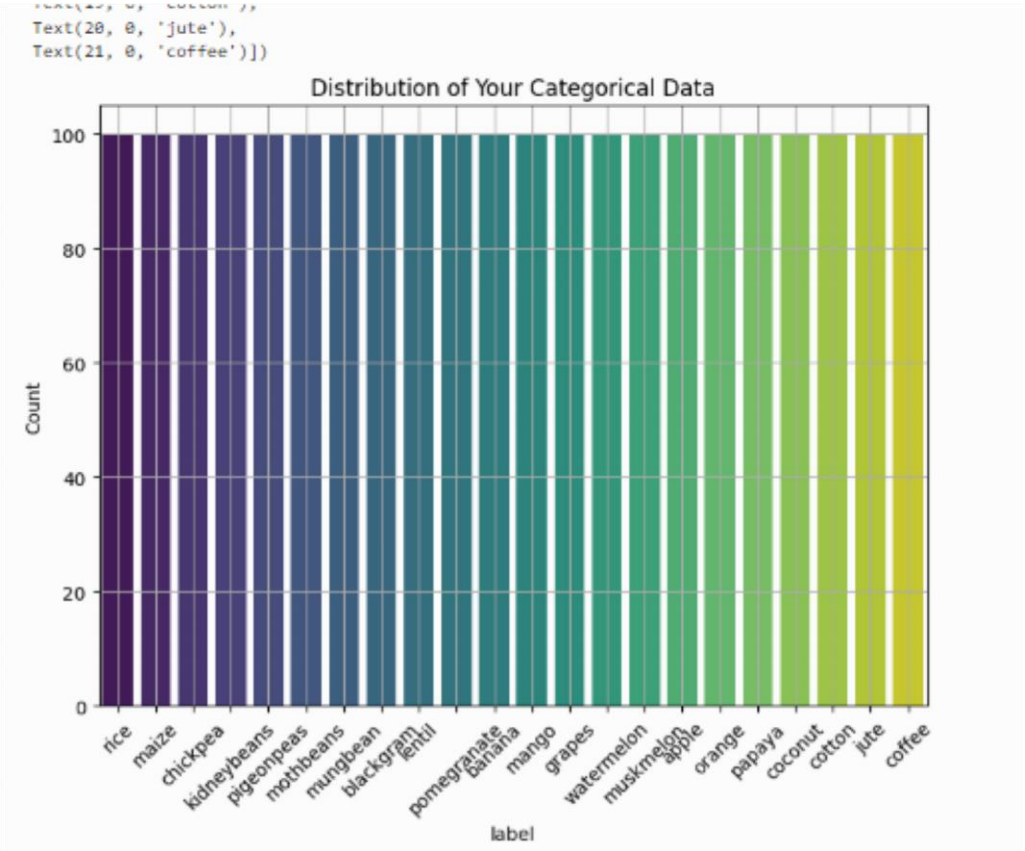



```
<Figure size 800x600 with 0 Axes>
<Axes: xlabel='temperature'>
Text(0.5, 1.0, 'Boxplot of Temperature')
Text(0.5, 0, 'Temperature (°C)')
```




```
<Figure size 800x600 with 0 Axes>
<ipython-input-13-ce3fb0d8849f>:44: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=categorical_data, palette='viridis')
<Axes: xlabel='label', ylabel='count'>
Text(0.5, 1.0, 'Distribution of Your Categorical Data')
Text(0.5, 0, 'label')
Text(0, 0.5, 'Count')
```

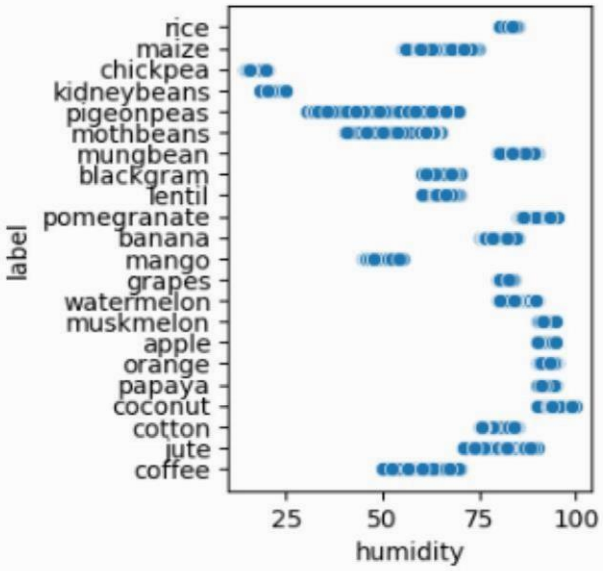
```
  sns.countplot(x=categorical_data, palette='viridis')
<Axes: xlabel='label', ylabel='count'>
Text(0.5, 1.0, 'Distribution of Your Categorical Data')
Text(0.5, 0, 'label')
Text(0, 0.5, 'Count')
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21],
 [Text(0, 0, 'rice'),
  Text(1, 0, 'maize'),
  Text(2, 0, 'chickpea'),
  Text(3, 0, 'kidneybeans'),
  Text(4, 0, 'pigeonpeas'),
  Text(5, 0, 'mothbeans'),
  Text(6, 0, 'mungbean'),
  Text(7, 0, 'blackgram'),
  Text(8, 0, 'lentil'),
  Text(9, 0, 'pomegranate'),
  Text(10, 0, 'banana'),
  Text(11, 0, 'mango'),
  Text(12, 0, 'grapes'),
  Text(13, 0, 'watermelon'),
  Text(20, 0, 'jute'),
  Text(21, 0, 'coffee')])
```



```
#BIVARIATE ANALYSIS
```

```
plt.subplot(2,4,7)
sns.scatterplot(x=df['humidity'],y=df['label'])
plt.show()
```

```
<Axes: >
<Axes: xlabel='humidity', ylabel='label'>
```




```
#CHECKING THE NULL VALUES
```

```
df.shape
```

```
(2200, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

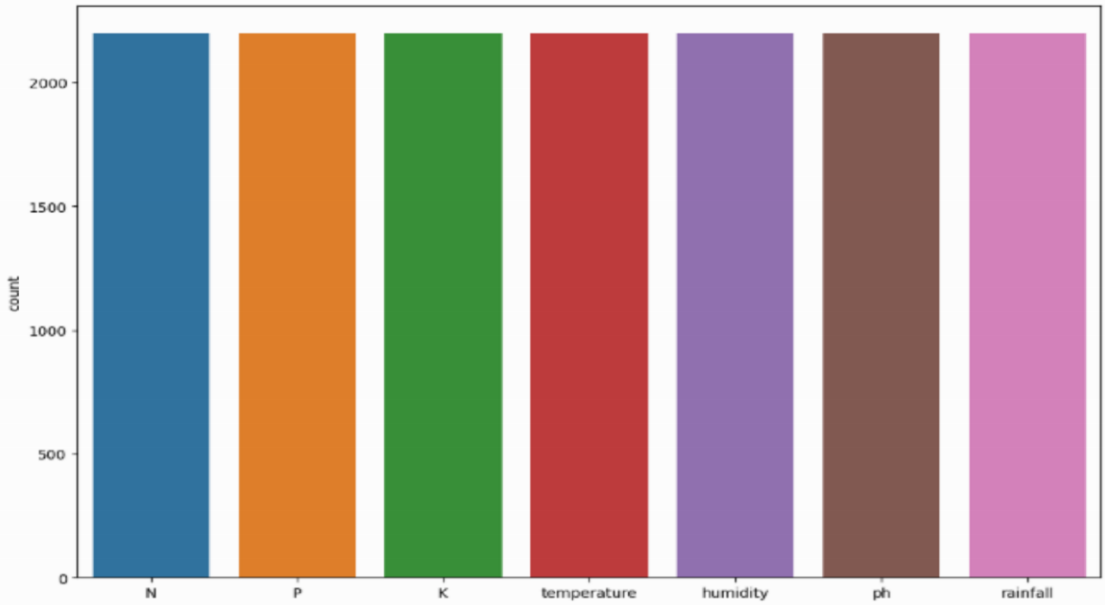
```
df.isnull().sum()
```

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

```
[16] #MULTIVARIATE ANALYSIS
```

```
[17] sns.countplot(df)
```

```
<Axes: ylabel='count'>
```




```
#DISCRIPTIVE ANALYSIS
```

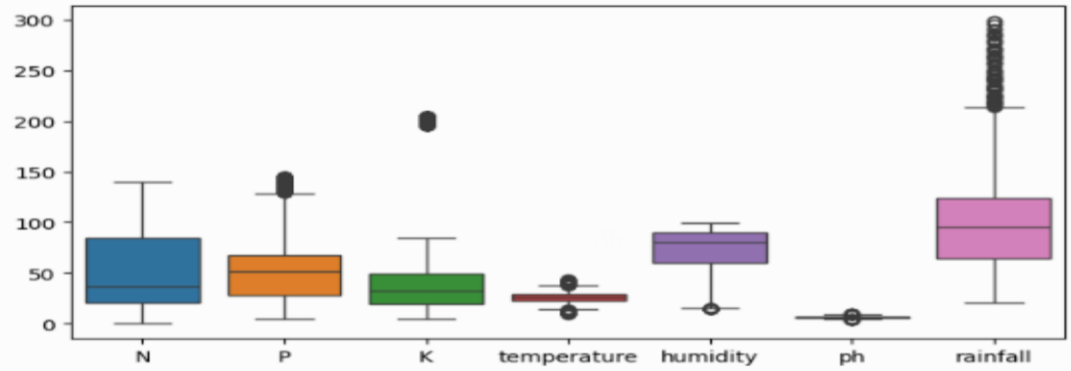
```
df.describe()
```

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

```
#HANDLING OUTLIERS
```

```
plt.figure(figsize=(8,4))
sns.boxplot(df)
```

```
<Figure size 800x400 with 0 Axes>
<Axes: >
```




```
Q1=df['P'].quantile(0.25)
Q3=df['P'].quantile(0.75)
IQR=Q3-Q1
filter=(df['P']>=Q1-1.5*IQR) & (df['P']<=Q3+1.5*IQR)
df=df.loc[filter]
```

```
#EXTRACTING SEASONAL CROPS
```

```
print("Summer crops")
print(df[(df['temperature']>30) & (df['humidity']>50)] ['label'].unique())
print("--------------------")
print("Winter crops")
print(df[(df['temperature']<28) & (df['humidity']>30)]['label'].unique())
print("---------------------")
print("Rainy crops")
print(df[(df['rainfall']>200) & (df['humidity']>50)] ['label'].unique())
print("---------------------")
```

```
Summer crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
--------------------
Winter crops
['rice' 'maize' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram' 'lentil'
 'pomegranate' 'banana' 'mango' 'grapes' 'watermelon' 'muskmelon' 'apple'
 'orange' 'papaya' 'coconut' 'cotton' 'jute' 'coffee']
---------------------
Rainy crops
['rice' 'papaya' 'coconut']
--------------------
```

```
#SPLITING DATA INTO TRAIN AND TEST
```

```
y=df['label']
x=df.drop(['label'],axis=1)
print("Shape of x",x.shape)
print("Shape of y",y.shape)
```

```
Shape of x (2062, 7)
Shape of y (2062,)
```

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)

print("The shape of x train",x_train.shape)
print("The shape of x test", x_test.shape)
print("The shape of y train",x_train.shape)
print("The shape of y test", x_test.shape)
```

```
The shape of x train (1649, 7)
The shape of x test (413, 7)
The shape of y train (1649, 7)
The shape of y test (413, 7)
```

```
#KMEANS MODEL
```

```
plt.rcParams['figure.figsize']=(10,4)
WCSS=[]
for i in range(1,11):
  km=KMeans(n_clusters=i,init="k-means++", max_iter=300,n_init=10,random_state=0)
  km.fit(x)
  WCSS.append(km.inertia_)
plt.plot(range (1,11), WCSS)
plt.title("The Elbow method", fontsize=20)
plt.xlabel("No of clusters")
plt.ylabel("WCSS")
plt.show()
```

```
-                    KMeans
KMeans(n_clusters=2, n_init=10, random_state=0)
```

```
-                    KMeans
KMeans(n_clusters=3, n_init=10, random_state=0)
```

```
-                    KMeans
KMeans(n_clusters=4, n_init=10, random_state=0)
```

```
-                    KMeans
KMeans(n_clusters=5, n_init=10, random_state=0)
```

```
-                    KMeans
KMeans(n_clusters=6, n_init=10, random_state=0)
```

```
-                    KMeans
KMeans(n_clusters=7, n_init=10, random_state=0)
```

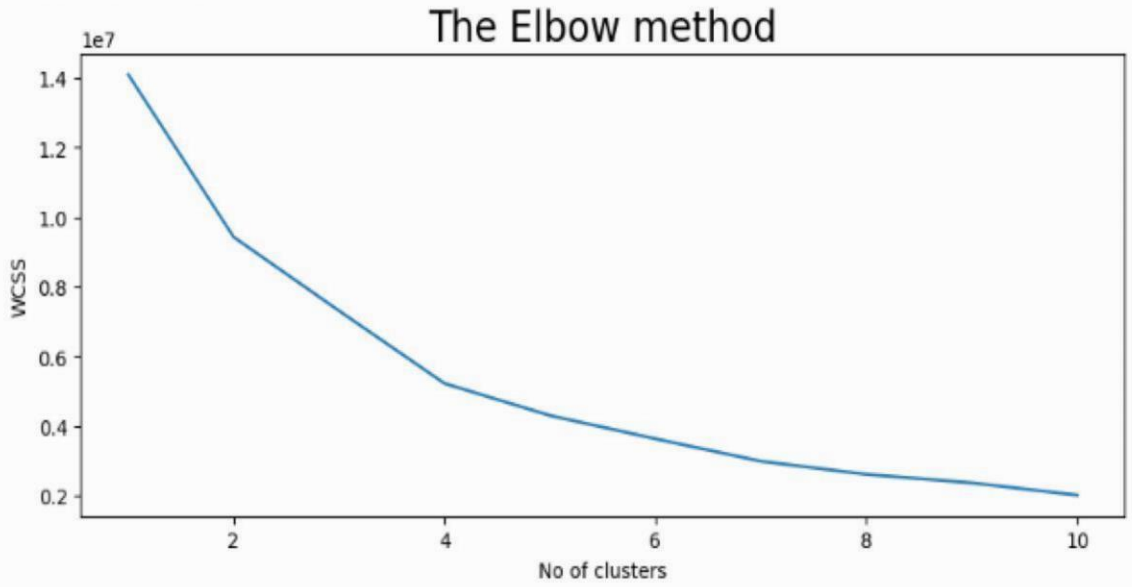
```
-              KMeans
KMeans(n_init=10, random_state=0)
```

```
KMeans
KMeans(n_init=10, random_state=0)
```

```
KMeans
KMeans(n_clusters=9, n_init=10, random_state=0)
```

```
KMeans
KMeans(n_clusters=10, n_init=10, random_state=0)
```

```
[<matplotlib.lines.Line2D at 0x7f0a40f78a30>]
Text(0.5, 1.0, 'The Elbow method')
Text(0.5, 0, 'No of clusters')
Text(0, 0.5, 'WCSS')
```




```
km=KMeans(n_clusters=4,init="k-means++", max_iter=300,n_init=10,random_state=0)
y_means=km.fit_predict(x)
a=df['label']
y_means=pd.DataFrame(y_means)
z=pd.concat([y_means,a],axis=1)
z=z.rename(columns={0:'cluster'})
print("lets check the results after applying the K-Means clustering analysis \n")
print("Crops in First cluster:", z[z['cluster']==0] ['label'].unique())
print("-----------------------------------------------")
print("Crops in Second cluster:", z[z['cluster']==1]['label'].unique())
print("-----------------------------------------------")
print("Crops in Third cluster:", z[z['cluster']==2]['label'].unique())
print("-----------------------------------------------")
print("Crops in Fourth cluster:", z[z['cluster']==3]['label'].unique())
```

```
lets check the results after applying the K-Means clustering analysis

Crops in First cluster: ['rice' 'pigeonpeas' 'apple' nan 'orange' 'papaya' 'coconut' 'cotton'
 'jute']
-----------------------------------------------
Crops in Second cluster: ['maize' 'banana' nan 'grapes' 'watermelon' 'muskmelon' 'orange' 'papaya'
 'coconut' 'cotton' 'jute']
-----------------------------------------------
Crops in Third cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'muskmelon' 'apple' nan
 'orange' 'papaya']
-----------------------------------------------
Crops in Fourth cluster: [nan 'grapes' 'muskmelon']
```

```
#LOGISTIC REGRESSION MODEL
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

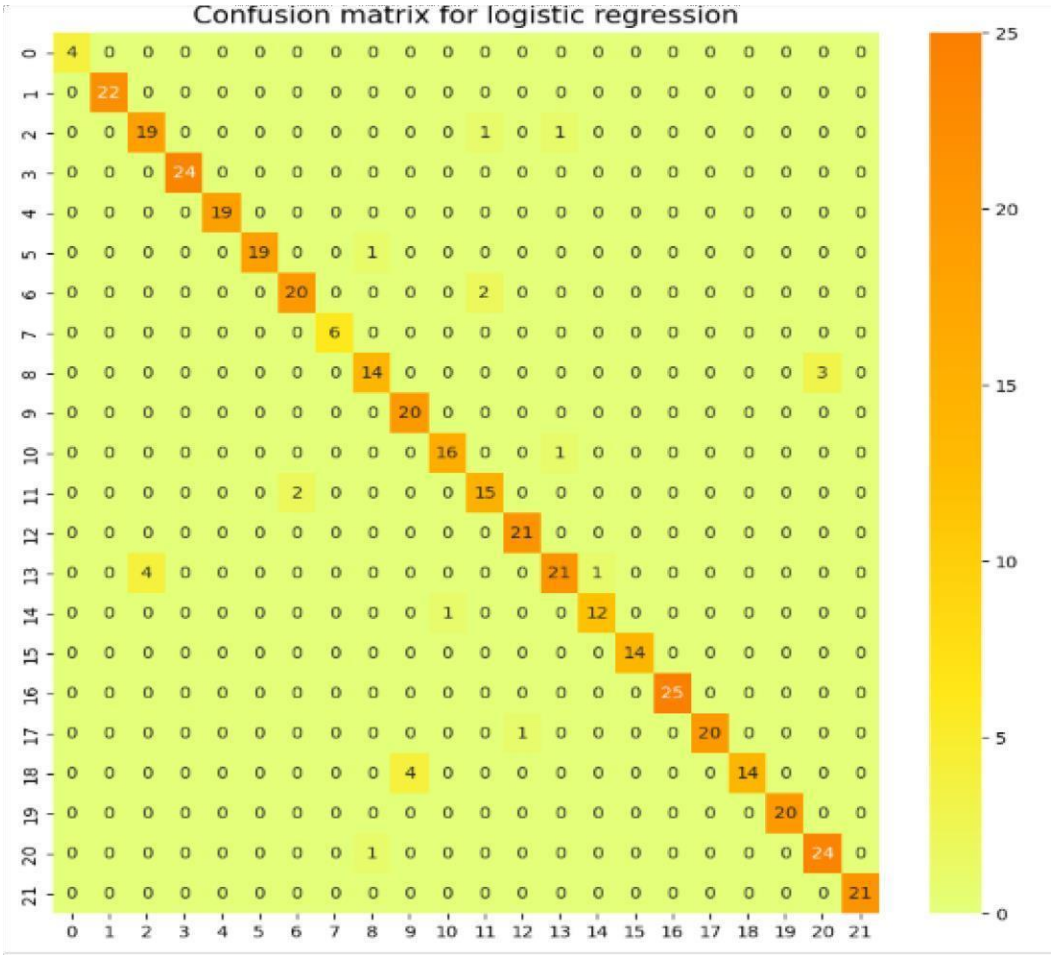
```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression

LogisticRegression()

```
#EVALUATING PERFORMANCE OF THE MODEL AND SAVING THE MODEL
```

```
from sklearn.metrics import confusion_matrix
plt.rcParams["figure.figsize"]=(10,10)
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,cmap='Wistia')
plt.title("Confusion matrix for logistic regression", fontsize=15)
plt.show()
```



```
from sklearn.metrics import classification_report
cr=classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00         4
      banana       1.00      1.00      1.00        22
   blackgram       0.83      0.90      0.86        21
    chickpea       1.00      1.00      1.00        24
     coconut       1.00      1.00      1.00        19
      coffee       1.00      0.95      0.97        20
      cotton       0.91      0.91      0.91        22
      grapes       1.00      1.00      1.00         6
        jute       0.88      0.82      0.85        17
 kidneybeans       0.83      1.00      0.91        20
      lentil       0.94      0.94      0.94        17
       maize       0.83      0.88      0.86        17
       mango       0.95      1.00      0.98        21
   mothbeans       0.91      0.81      0.86        26
    mungbean       0.92      0.92      0.92        13
   muskmelon       1.00      1.00      1.00        14
      orange       1.00      1.00      1.00        25
      papaya       1.00      0.95      0.98        21
  pigeonpeas       1.00      0.78      0.88        18
 pomegranate       1.00      1.00      1.00        20
        rice       0.89      0.96      0.92        25
  watermelon       1.00      1.00      1.00        21

    accuracy                           0.94       413
   macro avg       0.95      0.95      0.95       413
weighted avg       0.95      0.94      0.94       413
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import pickle

# Example: Generate or load your dataset (replace with your actual data loading process)
# Here, we generate random data as an example
np.random.seed(0)
n_samples = 1000
n_features = 5
X = np.random.rand(n_samples, n_features)  # Features (e.g., environmental factors)
y = np.random.rand(n_samples) * 1000       # Target variable (e.g., crop yield)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train a model (Random Forest regressor in this example)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model (optional, depending on your dataset)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
print(f"Training R^2 score: {train_score:.2f}")
print(f"Testing R^2 score: {test_score:.2f}")

# Save the model to a file (pickle)
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)
    print("Model saved as model.pkl")
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
Training R^2 score: 0.85
Testing R^2 score: -0.09
Model saved as model.pkl
```

```
import pickle
pickle.dump(model,open('/content/model.pkl','wb'))
```

```
from flask import Flask
app=Flask(__name__)
@app.route('/')
def hello_world():
  return 'hello world'
if __name__=='__main__':
  app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug: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
INFO:werkzeug:Press CTRL+C to quit
```