SoilSense ,an End-to-End Flutter Application: Revolutionizing Modern Agriculture with AI and ML Technologies

-Bhemal Sasi Sandeep Singh

Abstract :

Modern farming is an essential industry that has been grappling with numerous challenges in recent times, such as unpredictable weather patterns, soil degradation, pest infestations, and changing market trends, that reduce yield and profitability. The development of SoilSense, a one-stop app for modern farmers that leverages AI and ML technologies, is a significant step towards overcoming these obstacles. SoilSense is a comprehensive platform that offers all necessary functionalities such as Analyzing soil health with the help of testing kit, crop selection, farm equipment control, weather forecasting, pest control, and market analysis in one app. The global smart agriculture market, expected to reach $15.3 billion by 2025, with a CAGR of 9.8%, indicates a high level of potential adoption of the product. Additionally, the app's development is planned to be compliant with relevant laws and regulations and open-source software and frameworks to increase farmers' trust. By filling existing market gaps and providing premium users with advanced features, SoilSense ensure that tech-savvy modern farmers can optimize their operations and increase yield and profitability. Overall, the development of SoilSense is a promising step towards revolutionizing the agricultural sector.

Problem Statement :

Modern farming has become increasingly complex, with farmers needing to keep up with changing environmental conditions, market demands, and technological advancements. As the global population continues to grow, the demand for food production also increases, making it necessary for farmers to maximize their yields while minimizing costs and risks.

The need for a comprehensive app that can help modern farmers to maximize their yields and profits while minimizing costs and risks

In response to these challenges, there is a need for a comprehensive app that can help farmers to optimize their operations and make informed decisions. Such an app would provide farmers with access to a range of features, including soil testing, crop selection, weather forecasting, pest control, farm equipment control, and market analysis.

Market/Customer/Business Need Assessment for SoilSense:

The global smart agriculture market has seen tremendous growth in recent years, with a compound annual growth rate (CAGR) of 13.8% from 2019 to 2025. This growth is driven by the increasing demand for food due to the growing global population, coupled with the need for sustainable and efficient agricultural practices.

Modern farmers face numerous challenges in optimizing their operations, such as ensuring soil quality, selecting the best crops, monitoring weather conditions, controlling pests, and managing equipment efficiently. These challenges require extensive knowledge, expertise, and time, which may not always be available to farmers.

SoilSense addresses these challenges by providing a comprehensive platform that helps farmers optimize their operations using AI and ML technologies. With SoilSense, farmers can use sample test kits to test the soil conditions, find the best crops and combinations of crops, get real-time insights into weather conditions and crop conditions, monitor and control farm equipment, and instantly control pests, all through their phone.

A.Overview of the global smart agriculture market.

The global smart agriculture market is expected to grow significantly over the next few years, driven by the increasing demand for food production and the adoption of advanced technologies. According to a report by MarketsandMarkets, the global smart agriculture market size is projected to reach USD 22.0 billion by 2025, growing at a CAGR of 13.8% from 2020 to 2025.

B. Compound Annual Growth Rate (CAGR) of the smart agriculture market.

The CAGR of the smart agriculture market is significant, indicating a high potential for growth and innovation in the industry. This growth is driven by factors such as increasing demand for food production, rising adoption of precision farming, and the need for sustainable agriculture practices.

The need for a comprehensive app like SoilSense is significant in the agriculture industry, where technological advancements are necessary to maximize yields and profits while minimizing costs and risks. SoilSense is designed to meet the unique needs of modern farmers, who are tech-savvy and interested in optimizing their operations using AI and ML technologies.

SoilSense's business model includes charging farmers for premium features and partnering with agribusinesses to provide targeted marketing and advertising to users, creating potential revenue streams. SoilSense's potential for expansion is vast, as the agriculture industry continues to grow and evolve with technological advancements.

Target Specifications and Characterization:

SoilSense is designed for tech-savvy modern farmers who are interested in optimizing their operations using AI and ML technologies. The app targets farmers who want to maximize their yields and profits while minimizing costs and risks. These farmers understand the unique needs of modern agriculture in terms of technology and expertise. The app is designed to be user-friendly and accessible to farmers of all ages and experience levels.

The target specifications for SoilSense include features such as soil testing, crop selection, weather forecasting, pest control, farm equipment control, and market analysis. The app uses AI and ML technologies such as Natural Language Processing (NLP), Computer Vision, and Predictive Analytics to provide real-time insights into soil quality and crop conditions. The app also utilizes sample test kits for soil testing, making it easy and convenient for farmers to obtain accurate soil analysis.

In terms of characterization, the target audience for SoilSense is diverse and encompasses a broad range of farmers. The app is designed to meet the needs of farmers who are seeking to improve their operations using advanced technologies. This includes both large-scale commercial farmers as well as small-scale and hobbyist farmers. The app is ideal for farmers who are looking for a comprehensive platform that provides actionable insights into their soil conditions, crop selection, and market trends.

Bench marking alternate products :

When it comes to smart farming technology, there are a number of products and services currently available in the market. Some of the main competitors of

SoilSense include:

1. AgroPulse: This is a platform that provides farmers with real-time data on soil health, weather, and crop growth. It uses sensors and remote monitoring to collect data, which is then analyzed using machine learning algorithms to generate insights and recommendations.
2. Granular: This is a cloud-based platform that provides farmers with tools for managing their operations, including crop planning, field mapping, and financial analysis. It also offers features such as soil mapping and yield monitoring, but does not use AI or ML technologies.
3. Climate FieldView: This platform provides farmers with real-time data on weather and crop growth, as well as tools for managing their operations. It uses sensors and remote monitoring to collect data, which is then analyzed using machine learning algorithms to generate insights and recommendations.

SoilSense is different from these competitors in several ways. Firstly, it is a comprehensive platform that combines soil testing, crop selection, weather forecasting, pest control, farm equipment control, and market analysis in one platform, providing farmers with a one-stop-shop for managing their operations. Secondly, SoilSense utilizes AI and ML technologies to provide farmers with real-time insights and recommendations that are tailored to their specific needs and conditions. Finally, SoilSense offers premium features that allow farmers to optimize their operations even further, such as personalized marketing and advertising from agribusinesses.

Business Model :

Revenue Streams:

Freemium Model: Basic features are free, with premium features available for purchase on a subscription basis.

1. Hardware Sales: Partner with IoT hardware companies to sell soil sensors and other connected devices to farmers through the SoilSense platform.
2. Targeted Advertising: Partner with agribusinesses to provide targeted advertising and promotional opportunities to users of the SoilSense platform.
3. Data Sales: Aggregate and anonymize user data to sell to third-party organizations for research purposes.

Cost Structure:

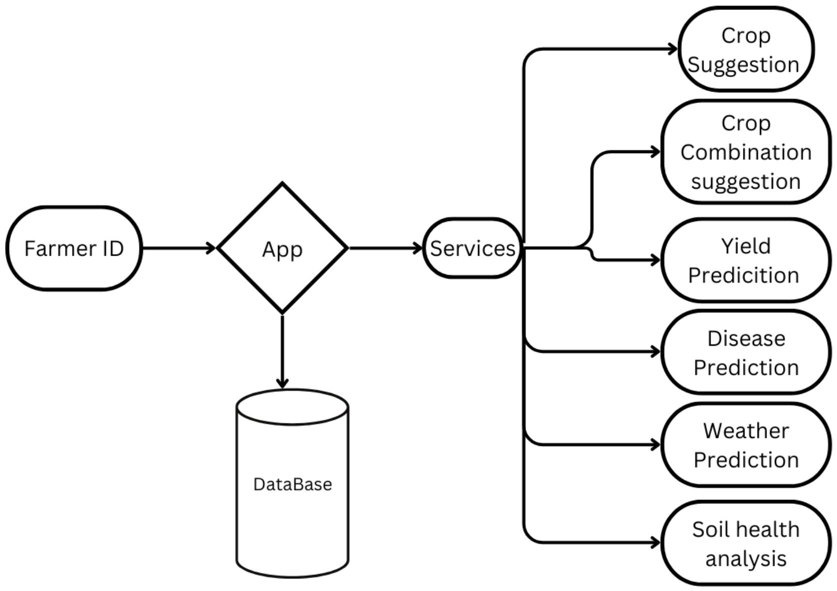
1. Research and Development: Ongoing investment in AI and ML technologies to improve the accuracy and functionality of the SoilSense platform.
2. Marketing and Sales: Advertising and promotional campaigns to attract and retain users.

3. Customer Support: Dedicated staff to assist users with technical issues and questions.
4. Hardware Partnerships: Negotiating deals with IoT hardware manufacturers to sell their devices through the SoilSense platform

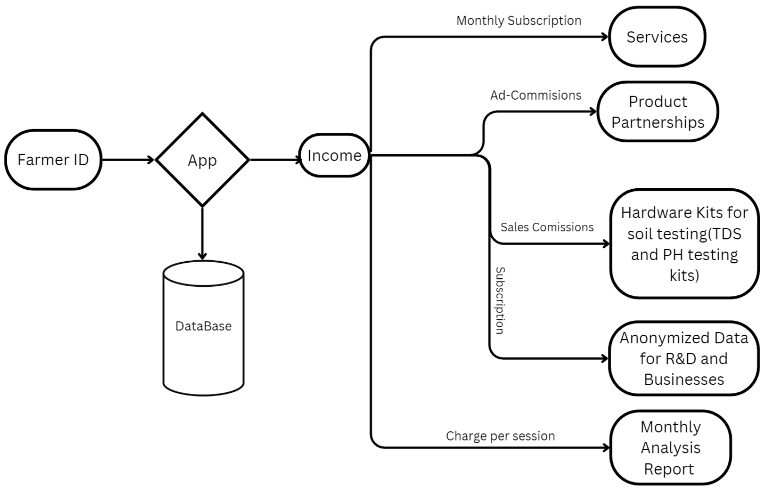
Key Partnerships:

1. IoT Hardware Companies: Partner with hardware companies to sell soil sensors, irrigation systems, and other connected devices through the SoilSense platform.
2. Agribusinesses: Partner with agribusinesses to provide targeted advertising and promotional opportunities to users of the SoilSense platform.
3. Research Institutions: Collaborate with research institutions to improve the accuracy and functionality of the SoilSense platform, as well as to leverage the data generated by the platform for research purposes.

Below is the Business Model for services Provided in the application :



Business model for the various streams of income generated from application .



Final Product Prototype (Abstract):

The final product prototype of SoilSense is a comprehensive platform designed to help modern farmers optimize their operations using AI and ML technologies. It combines soil testing, crop selection, weather forecasting, pest control, farm equipment control, and market analysis in one easy-to-use platform.

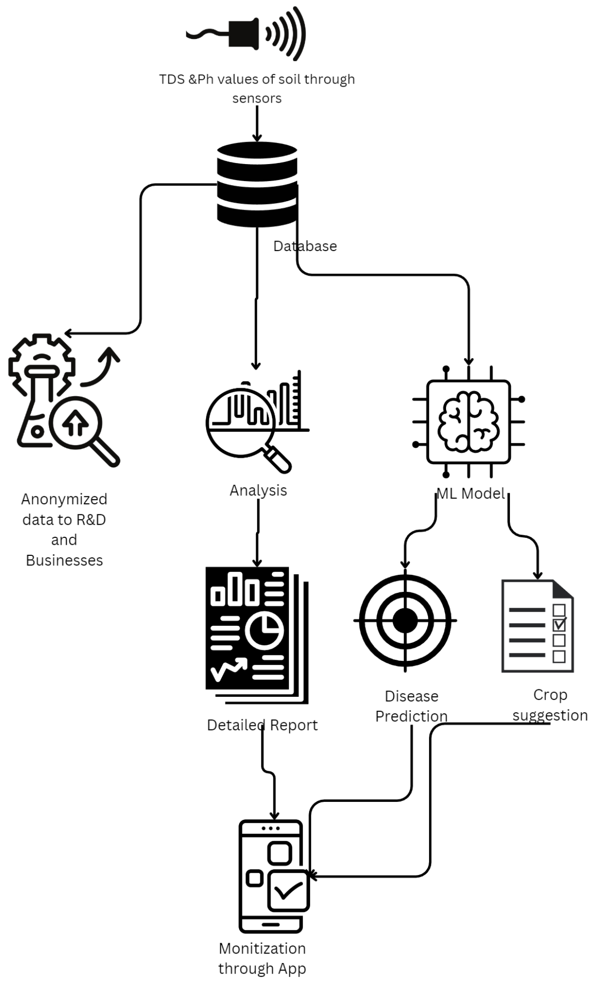
Farmers can use SoilSense to obtain real-time insights into their soil quality and make data-driven decisions about crop selection and pest control. They can also monitor weather patterns and adjust their farming practices accordingly. SoilSense also allows farmers to control their farm equipment remotely, saving time and reducing the risk of accidents.

The platform utilizes advanced AI and ML technologies such as natural language processing, computer vision, and predictive analytics to provide farmers with accurate and personalized recommendations. It also offers premium features that farmers can pay for, such as customized crop plans and access to exclusive market data.

SoilSense's business model involves charging farmers for premium features and partnering with agribusinesses to provide targeted marketing and advertising to users. The platform also plans to partner with hardware companies to allow IoT devices to perform farm equipment through the app.

Overall, SoilSense is a unique and innovative product that addresses the needs of modern farmers and offers them an all-in-one solution to optimize their farming operations.

Schematic diagram:



Product Details :

1.how does it work ?

The ML model uses inputs like TDS(Total Dissolved salts) & Ph in the soil to determine the best suitable crop for that soil type .It also uses ML to determine the best combination of crops to produce higher yields.

Here TDS Value and Ph can be obtained by dissolving a sample of soil in distilled water and placing the TDS and Ph sensors in that water . These inputs are entered into the app .

The images of leaves and roots can be used to determine the crop diseases .Users data will be stored at regular intervals and will be used to produce detailed report of crop and soil condition.

Data Sources:

- SoilSense will require a range of data sources to function effectively, including weather data, soil data, crop data, pest data, and market data. These data sources can be obtained from various sources, such as government agencies, weather services, crop research institutions, and market data providers.

Algorithms & Frame Works:

- SoilSense will utilize various AI and ML algorithms and frameworks to process and analyze the data, such as computer vision, predictive analytics, and machine learning algorithms. The platform will also require a range of software tools, such as databases, cloud computing platforms, and data visualization tools.

Team Required:

- Developing SoilSense will require a team with diverse skills and expertise, including software developers, data scientists, agronomists, and business development professionals. The team will need to work together to design, develop, test, and market the platform.

Cost:

- The cost of developing SoilSense will depend on various factors, such as the size of the development team, the complexity of the algorithms and frameworks used, the amount of data needed, and the duration of the development process. Developing such a platform can be costly, and it may require a significant investment in terms of time, money, and resources. However, the potential benefits of the platform, such as increased yields and profits for farmers, make it a worthwhile investment.

Code Implementation :

Importing libraries

```
# for manipulations

import numpy as np

import pandas as pd


# for data visualizations

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('fivethirtyeight')

# for interactivity

import ipywidgets

from ipywidgets import interact

data = pd.read_csv("/content/Crop_recommendation.csv")

print("Shape of the Dataset :", data.shape)
```

```
Shape of the Dataset : (2200, 8)
```

```
data.head()
```

```
                                        N P K temperature humidity ph rainfall label

2195 107 34 32 26.774637 66.413269 6.780064 177.774507 coffee

2196 99 15 27 27.417112 56.636362 6.086922 127.924610 coffee

2197 118 33 30 24.131797 67.225123 6.362608 173.322839 coffee

2198 117 32 34 26.272418 52.127394 6.758793 127.175293 coffee

2199 104 18 30 23.603016 60.396475 6.779833 140.937041 coffee
```

```
data.tail()
```

```
                                        N P K temperature humidity ph rainfall label

2195 107 34 32 26.774637 66.413269 6.780064 177.774507 coffee

2196 99 15 27 27.417112 56.636362 6.086922 127.924610 coffee

2197 118 33 30 24.131797 67.225123 6.362608 173.322839 coffee
```

```
2198 117 32 34 26.272418 52.127394 6.758793 127.175293 coffee
2199 104 18 30 23.603016 60.396475 6.779833 140.937041 coffee
```

```
data.isnull().sum()
```

```
N 0
P 0
K 0
temperature 0
humidity 0
ph 0
rainfall 0


label 0
dtype: int64
```

```
data['label'].value_counts()
```

```
rice 100
maize 100
jute 100
cotton 100
coconut 100
papaya 100
orange 100
apple 100
muskmelon 100
watermelon 100
grapes 100
mango 100
banana 100
pomegranate 100
lentil 100
```

```
blackgram 100

mungbean 100

mothbeans 100

pigeonpeas 100

kidneybeans 100

chickpea 100

coffee 100

Name: label, dtype: int64
```

```
print("Average Ratio of Nitrogen in the Soil :

{0:.2f}".format(data['N'].mean()))

print("Average Ratio of Phosphorous in the Soil :

{0:.2f}".format(data['P'].mean()))

print("Average Ratio of Potassium in the Soil :

{0:.2f}".format(data['K'].mean()))

print("Average Tempature in Celsius :

{0:.2f}".format(data['temperature'].mean()))

print("Average Relative Humidity in % :

{0:.2f}".format(data['humidity'].mean()))

print("Average PH Value of the soil : {0:.2f}".format(data['ph'].mean()))

print("Average Rainfall in mm : {0:.2f}".format(data['rainfall'].mean()))
```

```
Average Ratio of Nitrogen in the Soil : 50.55

Average Ratio of Phosphorous in the Soil : 53.36

Average Ratio of Potassium in the Soil : 48.15

Average Tempature in Celsius : 25.62

Average Relative Humidity in % : 71.48

Average PH Value of the soil : 6.47

Average Rainfall in mm : 103.46
```

```
# lets check the Summary Statistics for each of the Crops

@interact
```

```
def summary(crops = list(data['label'].value_counts().index)):

x = data[data['label'] == crops]

print("---------------------------------------------")

print("Statistics for Nitrogen")

print("Minimum Nitrigen required :", x['N'].min())

print("Average Nitrogen required :", x['N'].mean())

print("Maximum Nitrogen required :", x['N'].max())

print("---------------------------------------------")

print("Statistics for Phosphorous")

print("Minimum Phosphorous required :", x['P'].min())

print("Average Phosphorous required :", x['P'].mean())

print("Maximum Phosphorous required :", x['P'].max())

print("---------------------------------------------")

print("Statistics for Potassium")

print("Minimum Potassium required :", x['K'].min())

print("Average Potassium required :", x['K'].mean())

print("Maximum Potassium required :", x['K'].max())

print("---------------------------------------------")

print("Statistics for Temperature")

print("Minimum Temperature required :

{0:.2f}".format(x['temperature'].min()))

print("Average Temperature required :

{0:.2f}".format(x['temperature'].mean()))

print("Maximum Temperature required :

{0:.2f}".format(x['temperature'].max()))

print("---------------------------------------------")

print("Statistics for Humidity")

print("Minimum Humidity required : {0:.2f}".format(x['humidity'].min()))

                              print("Average Humidity required : {0:.2f}".format(x['humidity'].mean()))

print("Maximum Humidity required : {0:.2f}".format(x['humidity'].max()))

print("---------------------------------------------")
print("Statistics for PH")

print("Minimum PH required : {0:.2f}".format(x['ph'].min()))

print("Average PH required : {0:.2f}".format(x['ph'].mean()))

print("Maximum PH required : {0:.2f}".format(x['ph'].max()))

print("---------------------------------------------")

print("Statistics for Rainfall")

print("Minimum Rainfall required : {0:.2f}".format(x['rainfall'].min()))

                                        print("Average Rainfall required : {0:.2f}".format(x['rainfall'].mean()))

print("Maximum Rainfall required : {0:.2f}".format(x['rainfall'].max()))
```

{"model _id": "ac1b 7979c1 7445a 7a88f c88c5 a7 d1640" , "vers ion_m ajor" : 2,"ve rsion _min

or":0}

```
@interact

def compare(conditions =

['N','P','K','temperature','ph','humidity','rainfall']):

 print("Average Value for", conditions,"is {0:.2f}".format(data[conditions].mean()))

print("----------------------------------------------")

print("Rice : {0:.2f}".format(data[(data['label'] ==

'rice')][conditions].mean()))

print("Black Grams : {0:.2f}".format(data[data['label'] ==

'blackgram'][conditions].mean()))

print("Banana : {0:.2f}".format(data[(data['label'] ==

'banana')][conditions].mean()))

print("Jute : {0:.2f}".format(data[data['label'] ==

'jute'][conditions].mean()))

print("Coconut : {0:.2f}".format(data[(data['label'] ==

'coconut')][conditions].mean()))

print("Apple : {0:.2f}".format(data[data['label'] ==

'apple'][conditions].mean()))

print("Papaya : {0:.2f}".format(data[(data['label'] ==

'papaya')][conditions].mean()))
```

```
print("Muskmelon : {0:.2f}".format(data[data['label'] ==
'muskmelon'][conditions].mean()))
print("Grapes : {0:.2f}".format(data[(data['label'] ==
'grapes')][conditions].mean()))
print("Watermelon : {0:.2f}".format(data[data['label'] ==
'watermelon'][conditions].mean()))
print("Kidney Beans: {0:.2f}".format(data[(data['label'] ==
'kidneybeans')][conditions].mean()))
print("Mung Beans : {0:.2f}".format(data[data['label'] ==
'mungbean'][conditions].mean()))
print("Oranges : {0:.2f}".format(data[(data['label'] ==
'orange')][conditions].mean()))
print("Chick Peas : {0:.2f}".format(data[data['label'] ==
'chickpea'][conditions].mean()))
print("Lentils : {0:.2f}".format(data[(data['label'] ==
'lentil')][conditions].mean()))
print("Cotton : {0:.2f}".format(data[data['label'] ==
'cotton'][conditions].mean()))
print("Maize : {0:.2f}".format(data[(data['label'] ==
'maize')][conditions].mean()))
print("Moth Beans : {0:.2f}".format(data[data['label'] ==
'mothbeans'][conditions].mean()))
print("Pigeon Peas : {0:.2f}".format(data[(data['label'] ==
'pigeonpeas')][conditions].mean()))
print("Mango : {0:.2f}".format(data[data['label'] ==
'mango'][conditions].mean()))
print("Pomegranate : {0:.2f}".format(data[(data['label'] ==
'pomegranate')][conditions].mean()))
print("Coffee : {0:.2f}".format(data[data['label'] ==
'coffee'][conditions].mean()))
```

{"model _id": "2b4f 66426c a54ad b9da8 93eea b3 c5981" , "vers ion_m ajor" : 2,"ve rsion _min

or":0}

```
@interact

def compare(conditions = ['N','P','K','temperature','ph','humidity','rainfall']):

print("Crops which require greater than average", conditions,'\n')

                    print(data[data[conditions] > data[conditions].mean()]['label'].unique())

print("----------------------------------------------")

print("Crops which require less than average", conditions,'\n')

print(data[data[conditions] <=

data[conditions].mean()]['label'].unique())
```

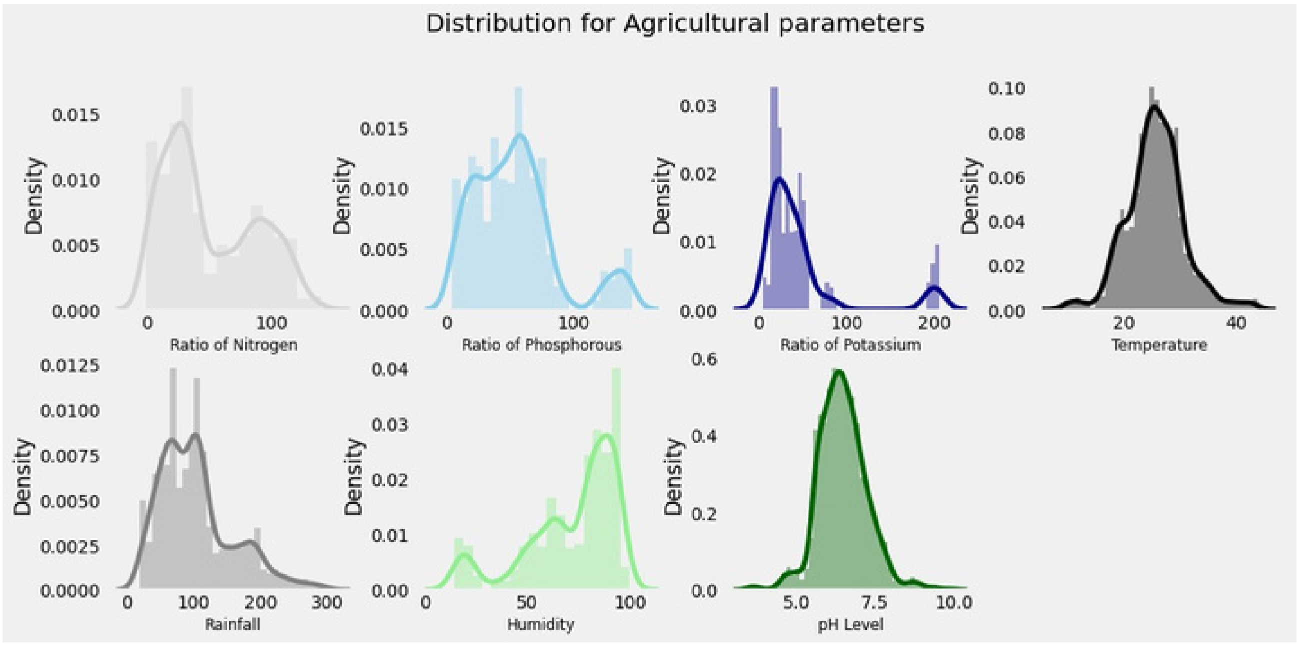
{"model _id": "e429 a90063 9c448 19500 55f66 9b a5e95" , "vers ion_m ajor" : 2,"ve rsion _min

or":0}

Analyzing agro-parameters

```
### distribution for agricultural parameters plt.rcParams['figure.figsize'] = (15, 7)

plt.subplot(2, 4, 1)

sns.distplot(data['N'], color = 'lightgrey')

plt.xlabel('Ratio of Nitrogen', fontsize = 12)

plt.grid()

plt.subplot(2, 4, 2)

sns.distplot(data['P'], color = 'skyblue')

plt.xlabel('Ratio of Phosphorous', fontsize = 12)

plt.grid()

plt.subplot(2, 4, 3)

sns.distplot(data['K'], color ='darkblue')

plt.xlabel('Ratio of Potassium', fontsize = 12)

plt.grid()

plt.subplot(2, 4, 4)

sns.distplot(data['temperature'], color = 'black')
```

```
plt.xlabel('Temperature', fontsize = 12)

plt.grid()

plt.subplot(2, 4, 5)

sns.distplot(data['rainfall'], color = 'grey')

plt.xlabel('Rainfall', fontsize = 12)

plt.grid()

plt.subplot(2, 4, 6)

sns.distplot(data['humidity'], color = 'lightgreen')

plt.xlabel('Humidity', fontsize = 12)

plt.grid()
plt.subplot(2, 4, 7)

sns.distplot(data['ph'], color = 'darkgreen')

plt.xlabel('pH Level', fontsize = 12)

plt.grid()


plt.suptitle('Distribution for Agricultural parameters', fontsize = 20)

plt.show()
```



Finding patterns

```
print("Some Interesting Patterns")
print("---------------------------------")

print("Crops which requires very High Ratio of Nitrogen Content in Soil:", data[data['N'] >
120]['label'].unique())

print("Crops which requires very High Ratio of Phosphorous Content in Soil:", data[data['P'] >
100]['label'].unique())

print("Crops which requires very High Ratio of Potassium Content in Soil:", data[data['K'] >
200]['label'].unique())

print("Crops which requires very High Rainfall:", data[data['rainfall'] > 200]['label'].unique())

print("Crops which requires very Low Temperature :", data[data['temperature'] < 10]
['label'].unique())

print("Crops which requires very High Temperature :",

data[data['temperature'] > 40]['label'].unique())

print("Crops which requires very Low Humidity:", data[data['humidity'] < 20]['label'].unique())

print("Crops which requires very Low pH:", data[data['ph'] < 4]['label'].unique())

print("Crops which requires very High pH:", data[data['ph'] > 9]['label'].unique())
```

```
Some Interesting Patterns

---------------------------------

Crops which requires very High Ratio of Nitrogen Content in Soil: ['cotton'] Crops which
requires very High Ratio of Phosphorous Content in Soil: ['grapes' 'apple']

Crops which requires very High Ratio of Potassium Content in Soil: ['grapes' 'apple']

Crops which requires very High Rainfall: ['rice' 'papaya' 'coconut']

Crops which requires very Low Temperature : ['grapes']

Crops which requires very High Temperature : ['grapes' 'papaya']

Crops which requires very Low Humidity: ['chickpea' 'kidneybeans']

Crops which requires very Low pH: ['mothbeans']

Crops which requires very High pH: ['mothbeans']
```

Seasonal growth

```
print("Summer Crops")

print(data[(data['temperature'] > 30) & (data['humidity'] > 50)]['label'].unique())

print("-----------------------------------")

print("Winter Crops")

print(data[(data['temperature'] < 20) & (data['humidity'] > 30)]['label'].unique())

print("-----------------------------------")
print("Rainy Crops")

print(data[(data['rainfall'] > 200) & (data['humidity'] > 30)]['label'].unique())
```

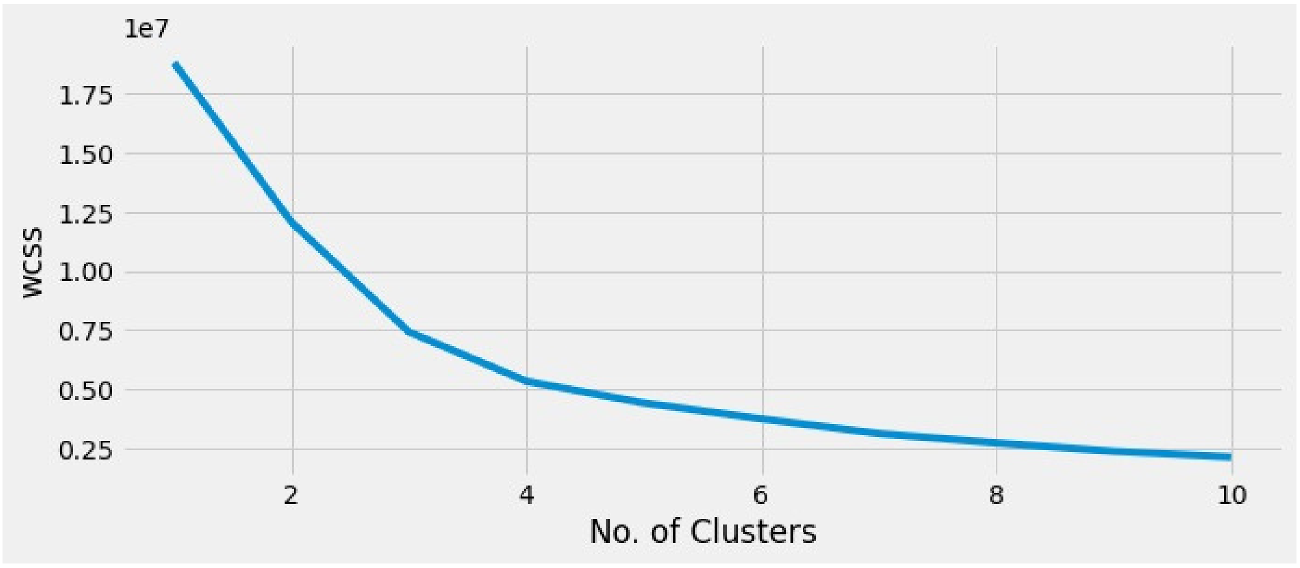
```
Summer Crops

['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']

----------------------------------

Winter Crops

['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']

----------------------------------

Rainy Crops

['rice' 'papaya' 'coconut']
```

Clustering similar crops

```
import warnings

warnings.filterwarnings('ignore')

# Lets select the Spending score, and Annual Income Columns from the Data

x = data.loc[:,

['N','P','K','temperature','ph','humidity','rainfall']].values

# let's check the shape of x

print(x.shape)

# lets convert this data into a dataframe

x_data = pd.DataFrame(x)

x_data.head()
```

```
(2200, 7)

0 1 2 3 4 5 6

0 90.0 42.0 43.0 20.879744 6.502985 82.002744 202.935536

1 85.0 58.0 41.0 21.770462 7.038096 80.319644 226.655537

2 60.0 55.0 44.0 23.004459 7.840207 82.320763 263.964248

3 74.0 35.0 40.0 26.491096 6.980401 80.158363 242.864034

4 78.0 42.0 42.0 20.130175 7.628473 81.604873 262.717340
```

```
from sklearn.cluster import KMeans

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_)

# lets plot the results

plt.plot(range(1, 11), wcss)

plt.xlabel('No. of Clusters')

plt.ylabel('wcss')

plt.show()
```



lets implement the K Means algorithm to perform Clustering analysis

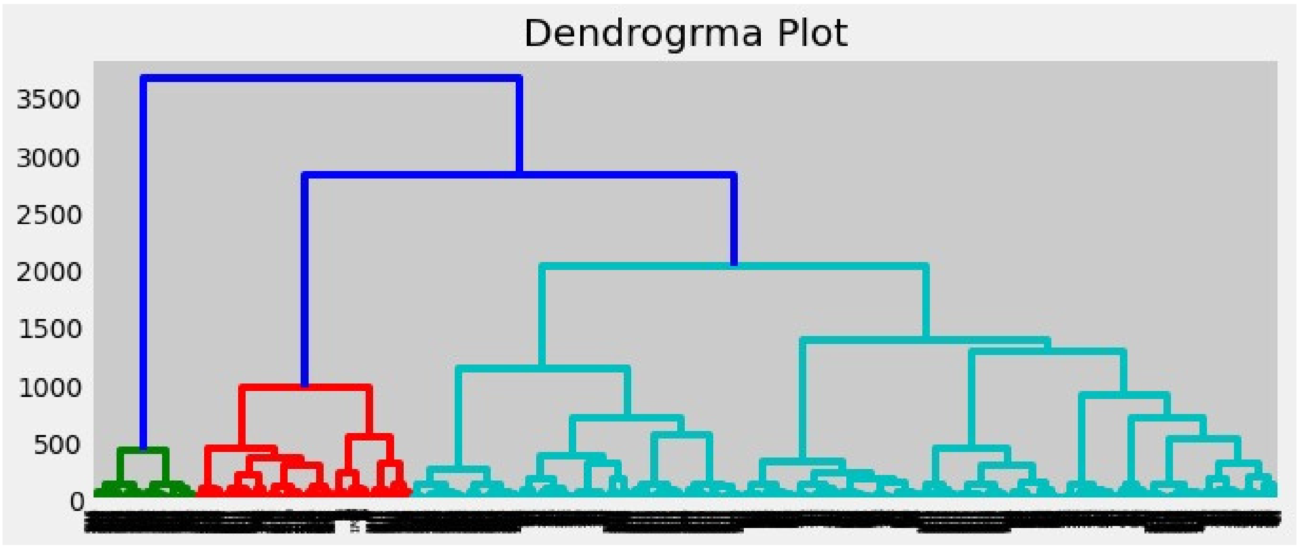
```
# lets implement the K Means algorithm to perform Clustering analysis

km = KMeans(n_clusters = 4, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)

y_means = km.fit_predict(x)


# lets find out the Results


a = data['label']
```

```
y_means = pd.DataFrame(y_means)

z = pd.concat([y_means, a], axis = 1)

z = z.rename(columns = {0: 'cluster'})


# lets check the Clusters of each Crops

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: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas'

'mothbeans' 'mungbean'

'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']

---------------------------------------------------------------

Crops in Second Cluster: ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya'

'cotton' 'coffee']

---------------------------------------------------------------

Crops in Third Cluster: ['grapes' 'apple']

---------------------------------------------------------------

Crops in Fourth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute'

'coffee']
```

```
#Dendrogrma Plot

import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x, method="ward"))
```

```
plt.title("Dendrogrma Plot")

plt.show()
```

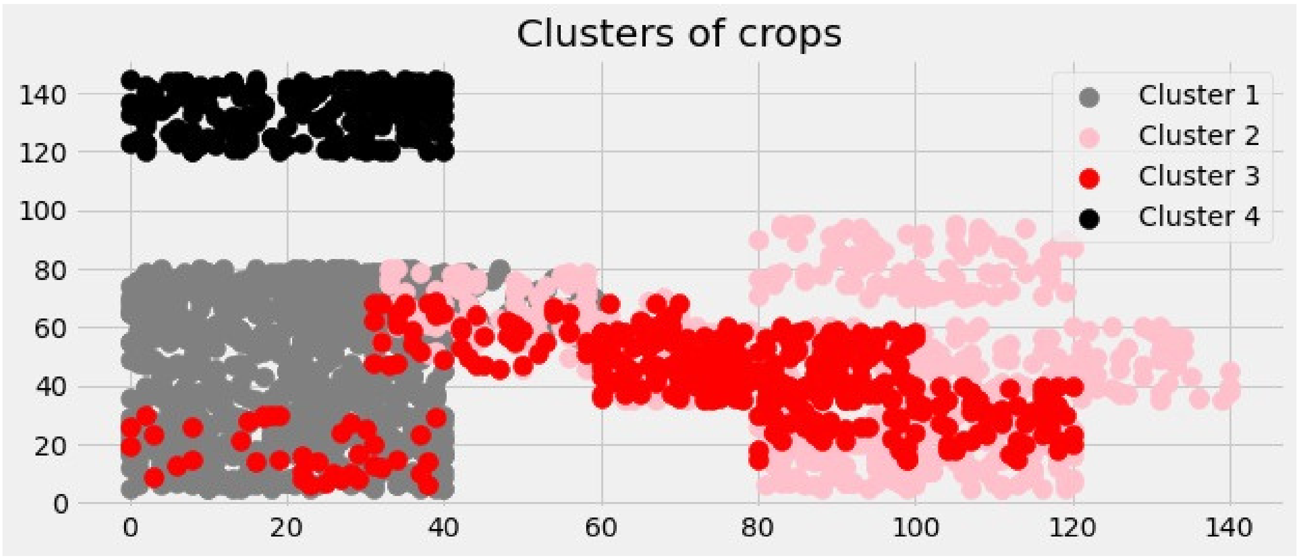


Training the hierarchical model on dataset

```
#Training the hierarchical model on dataset

from sklearn.cluster import AgglomerativeClustering

hc= AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')

y_her= hc.fit_predict(x)

b = data['label']

y_herr = pd.DataFrame(y_her)

w = pd.concat([y_herr, b], axis = 1)

w= w.rename(columns = {0: 'cluster'})


#the Clusters of each Crops

print("Hierachical Clustering Analysis \n")

print("Crops in Zero Cluster:", w[w['cluster'] == 0]['label'].unique())

print("---------------------------------------------------------------")

print("Crops in First Cluster:", w[w['cluster'] == 1]['label'].unique())

print("---------------------------------------------------------------")

print("Crops in Second Cluster:", w[w['cluster'] == 2]['label'].unique())

print("---------------------------------------------------------------")

print("Crops in Third Cluster:", w[w['cluster'] == 3]['label'].unique())
```

```
Hierachical Clustering Analysis

Crops in Zero Cluster: ['chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'

'mungbean' 'blackgram'

'lentil' 'pomegranate' 'mango' 'orange' 'coconut']

---------------------------------------------------------------

Crops in First Cluster: ['maize' 'blackgram' 'banana' 'watermelon'

'muskmelon' 'papaya' 'cotton']

---------------------------------------------------------------

Crops in Second Cluster: ['rice' 'papaya' 'coconut' 'jute' 'coffee']

---------------------------------------------------------------

Crops in Third Cluster: ['grapes' 'apple']
```

```python
##visulaizing the clusters

plt.scatter(x[y_her == 0, 0], x[y_her == 0, 1], s = 100, c = 'gray', label = 'Cluster 1')

plt.scatter(x[y_her == 1, 0], x[y_her == 1, 1], s = 100, c = 'pink', label = 'Cluster 2')

plt.scatter(x[y_her== 2, 0], x[y_her == 2, 1], s = 100, c = 'red', label = 'Cluster 3')

plt.scatter(x[y_her == 3, 0], x[y_her == 3, 1], s = 100, c = 'black', label = 'Cluster 4')

plt.title('Clusters of crops')

plt.legend()

plt.show()
```



Hard Clustering

```python
print("Results for Hard Clustering\n")
```

```
counts = z[z['cluster'] == 0]['label'].value_counts()

d = z.loc[z['label'].isin(counts.index[counts >= 50])]

d = d['label'].value_counts()

print("Crops in Cluster 1:", list(d.index)) print("---------------------------------------------------")

counts = z[z['cluster'] == 1]['label'].value_counts()

d = z.loc[z['label'].isin(counts.index[counts >= 50])]

d = d['label'].value_counts()

print("Crops in Cluster 2:", list(d.index)) print("---------------------------------------------------")

counts = z[z['cluster'] == 2]['label'].value_counts()

d = z.loc[z['label'].isin(counts.index[counts >= 50])]

d = d['label'].value_counts()

print("Crops in Cluster 3:", list(d.index)) print("---------------------------------------------------")

counts = z[z['cluster'] == 3]['label'].value_counts()

d = z.loc[z['label'].isin(counts.index[counts >= 50])]

d = d['label'].value_counts()

print("Crops in Cluster 4:", list(d.index))
```

Results for Hard Clustering

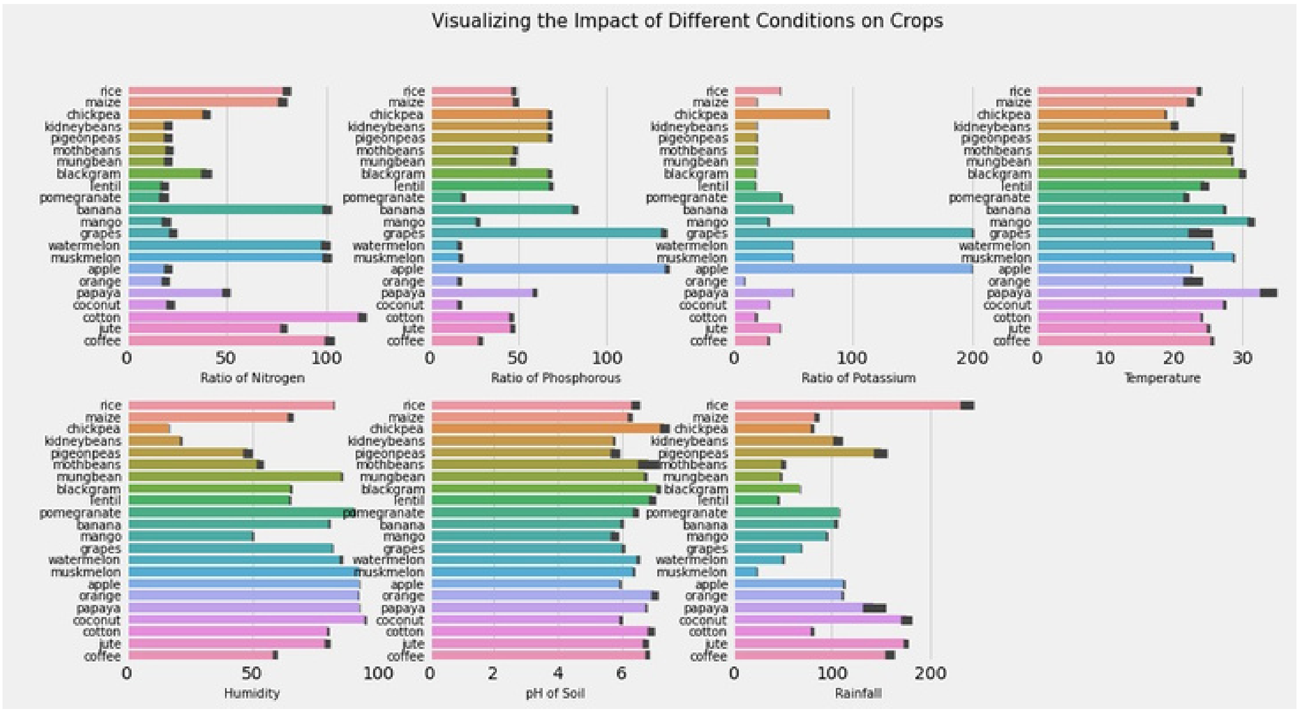
```
Crops in Cluster 1: ['chickpea', 'kidneybeans', 'mothbeans', 'mungbean',

'blackgram', 'lentil', 'pomegranate', 'mango', 'orange']

---------------------------------------------------

Crops in Cluster 2: ['maize', 'banana', 'watermelon', 'muskmelon', 'cotton']

---------------------------------------------------

Crops in Cluster 3: ['grapes', 'apple']

---------------------------------------------------

Crops in Cluster 4: ['rice', 'pigeonpeas', 'papaya', 'coconut', 'jute',

'coffee']
```

Visualizing the Hidden Patterns

### Data Visualizations

```
plt.rcParams['figure.figsize'] = (15, 8)


plt.subplot(2, 4, 1)

sns.barplot(data['N'], data['label'])

plt.ylabel(' ')

plt.xlabel('Ratio of Nitrogen', fontsize = 10)

plt.yticks(fontsize = 10)

plt.subplot(2, 4, 2)

sns.barplot(data['P'], data['label'])

plt.ylabel(' ')

plt.xlabel('Ratio of Phosphorous', fontsize = 10)

plt.yticks(fontsize = 10)

plt.subplot(2, 4, 3)

sns.barplot(data['K'], data['label'])

plt.ylabel(' ')

plt.xlabel('Ratio of Potassium', fontsize = 10)

plt.yticks(fontsize = 10)

plt.subplot(2, 4, 4)

sns.barplot(data['temperature'], data['label'])

plt.ylabel(' ')

plt.xlabel('Temperature', fontsize = 10)

plt.yticks(fontsize = 10)

plt.subplot(2, 4, 5)

sns.barplot(data['humidity'], data['label'])

plt.ylabel(' ')

plt.xlabel('Humidity', fontsize = 10)

plt.yticks(fontsize = 10)

plt.subplot(2, 4, 6)
```

```
sns.barplot(data['ph'], data['label'])

plt.ylabel(' ')

plt.xlabel('pH of Soil', fontsize = 10)

plt.yticks(fontsize = 10)


plt.subplot(2, 4, 7)

sns.barplot(data['rainfall'], data['label'])

plt.ylabel(' ')

plt.xlabel('Rainfall', fontsize = 10)

plt.yticks(fontsize = 10)


plt.suptitle('Visualizing the Impact of Different Conditions on Crops',

fontsize = 15) plt.show()
```



Predictive Modelling

```
# lets split the Dataset for Predictive Modelling


y = data['label']

x = data.drop(['label'], axis = 1)

print("Shape of x:", x.shape)
print("Shape of y:", y.shape)

Shape of x: (2200, 7)

Shape of y: (2200,)

# lets create Training and Testing Sets for Validation of Results

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:", y_train.shape)

print("The Shape of y test:", y_test.shape)

The Shape of x train: (1760, 7)

The Shape of x test: (440, 7)

The Shape of y train: (1760,)

The Shape of y test: (440,)
from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import AdaBoostClassifier

from xgboost import XGBClassifier

import xgboost as xgb

from sklearn.metrics import classification_report

from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score from
mlxtend.plotting import plot_confusion_matrix


def evaluator(y_test, y_pred):


# Accuracy:

print('Accuracy is: ', accuracy_score(y_test,y_pred))
print('')

# Classification Report:

                                        print('Classification Report: \n',classification_report(y_test,y_pred))

print('Confusion Matrix: \n\n')

plt.style.use("ggplot")

cm = confusion_matrix(y_test,y_pred)

plot_confusion_matrix(conf_mat = cm,figsize=(10,10),show_normed=True)

                      plt.title('Confusion Matrix for Logistic Regression', fontsize = 15)

plt.show()

model_accuracy = pd.DataFrame(columns=['Model','Accuracy'])

models = {

"KNN" : KNeighborsClassifier(),

"DT" : DecisionTreeClassifier(),

'RFC' : RandomForestClassifier(),

'GBC' : GradientBoostingClassifier(),

'XGB' : XGBClassifier()

}

for test, clf in models.items():

clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)

acc = accuracy_score(y_test,y_pred)

train_pred = clf.predict(x_train)

train_acc = accuracy_score(y_train, train_pred)

print("\n", test + ' scores')

print(acc)

print(classification_report(y_test,y_pred))

print(confusion_matrix(y_test,y_pred))

print('*' * 100,"\n")

                              model_accuracy = model_accuracy.append({'Model': test, 'Accuracy': acc,
'Train_acc': train_acc}, ignore_index=True)
```

```
KNN scores

0.9772727272727273

precision recall f1-score support


apple 1.00 1.00 1.00 18

banana 1.00 1.00 1.00 18

blackgram 1.00 1.00 1.00 22

chickpea 1.00 1.00 1.00 23

coconut 1.00 1.00 1.00 15

coffee 1.00 1.00 1.00 17

cotton 1.00 0.94 0.97 16

grapes 1.00 1.00 1.00 18

jute 0.79 0.90 0.84 21

kidneybeans 0.91 1.00 0.95 20

lentil 1.00 1.00 1.00 17

maize 0.95 1.00 0.97 18

mango 1.00 1.00 1.00 21

mothbeans 1.00 1.00 1.00 25

mungbean 1.00 1.00 1.00 17

muskmelon 1.00 1.00 1.00 23

orange 1.00 1.00 1.00 23

papaya 1.00 1.00 1.00 21

pigeonpeas 1.00 0.91 0.95 22

pomegranate 1.00 1.00 1.00 23

rice 0.91 0.80 0.85 25

watermelon 1.00 1.00 1.00 17

accuracy 0.98 440

macro avg 0.98 0.98 0.98 440
```

```
weighted avg 0.98 0.98 0.98 440

[[18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 15 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 2 0]

[ 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0]

 [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 20 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0]

[ 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 20 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17]]

******* ***** ***** ****** ***** ***** **** * ** ******* ***** ***** ****** ***** ***** **** **********************

 DT scores

0.9886363636363636

precision recall f1-score support


apple 1.00 1.00 1.00 18

banana 1.00 1.00 1.00 18

blackgram 1.00 0.95 0.98 22

chickpea 0.92 1.00 0.96 23

coconut 1.00 1.00 1.00 15

coffee 1.00 1.00 1.00 17

cotton 1.00 1.00 1.00 16

grapes 1.00 1.00 1.00 18

jute 0.95 0.95 0.95 21

kidneybeans 1.00 0.90 0.95 20

lentil 1.00 1.00 1.00 17

maize 1.00 1.00 1.00 18

mango 1.00 1.00 1.00 21

mothbeans 0.96 1.00 0.98 25

mungbean 1.00 1.00 1.00 17

muskmelon 1.00 1.00 1.00 23

orange 1.00 1.00 1.00 23

papaya 1.00 1.00 1.00 21

pigeonpeas 1.00 1.00 1.00 22

pomegranate 1.00 1.00 1.00 23

rice 0.96 0.96 0.96 25

watermelon 1.00 1.00 1.00 17
```

```
accuracy 0.99 440

macro avg 0.99 0.99 0.99 440

weighted avg 0.99 0.99 0.99 440

[[18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 21 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]

[ 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 1 0]

 [ 0 0 0 2 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0]
[ 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 24 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17]]
```

```
******* ***** ***** ****** ***** ***** **** * ** ******* ***** ***** ****** ***** ***** **** **********************

 RFC scores

0.9977272727272727

precision recall f1-score support


apple 1.00 1.00 1.00 18

banana 1.00 1.00 1.00 18

blackgram 1.00 1.00 1.00 22

chickpea 1.00 1.00 1.00 23

coconut 1.00 1.00 1.00 15

coffee 1.00 1.00 1.00 17

cotton 1.00 1.00 1.00 16

grapes 1.00 1.00 1.00 18

jute 0.95 1.00 0.98 21

kidneybeans 1.00 1.00 1.00 20

lentil 1.00 1.00 1.00 17

maize 1.00 1.00 1.00 18

mango 1.00 1.00 1.00 21

mothbeans 1.00 1.00 1.00 25

mungbean 1.00 1.00 1.00 17

muskmelon 1.00 1.00 1.00 23

orange 1.00 1.00 1.00 23

papaya 1.00 1.00 1.00 21

pigeonpeas 1.00 1.00 1.00 22

pomegranate 1.00 1.00 1.00 23

rice 1.00 0.96 0.98 25

watermelon 1.00 1.00 1.00 17

accuracy 1.00 440

macro avg 1.00 1.00 1.00 440

weighted avg 1.00 1.00 1.00 440
```

```
[[18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

 [ 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0]

[ 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 24 0]

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17]]
******* ***** ***** ****** ***** ***** ***** ** ******* ***** ***** ****** ***** ***** **** *********************

 GBC scores

0.9954545454545455

                                   precision recall f1-score support


                                            apple 1.00 1.00 1.00 18

                                           banana 1.00 1.00 1.00 18

                                        blackgram 1.00 1.00 1.00 22
```

```
chickpea 1.00 1.00 1.00 23
coconut 1.00 1.00 1.00 15
coffee 1.00 1.00 1.00 17
cotton 1.00 1.00 1.00 16
grapes 1.00 1.00 1.00 18
jute 0.95 0.95 0.95 21
kidneybeans 1.00 1.00 1.00 20
lentil 1.00 1.00 1.00 17
maize 1.00 1.00 1.00 18
mango 1.00 1.00 1.00 21
mothbeans 1.00 1.00 1.00 25
mungbean 1.00 1.00 1.00 17
muskmelon 1.00 1.00 1.00 23
orange 1.00 1.00 1.00 23
papaya 1.00 0.95 0.98 21
pigeonpeas 1.00 1.00 1.00 22
pomegranate 1.00 1.00 1.00 23
rice 0.96 1.00 0.98 25

watermelon 1.00 1.00 1.00 17

accuracy 1.00 440
macro avg 1.00 1.00 1.00 440
weighted avg 1.00 1.00 1.00 440

[[18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 1 0]
[ 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 20 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17]]
******* ***** ***** ****** ***** ***** ***** ** ******* ***** ***** ****** ***** ***** ****
***********************
 XGB scores
0.9954545454545455
                                   precision recall f1-score support


                                        apple 1.00 1.00 1.00 18
                                       banana 1.00 1.00 1.00 18
                                    blackgram 1.00 1.00 1.00 22
                                     chickpea 1.00 1.00 1.00 23
                                      coconut 1.00 1.00 1.00 15
                                       coffee 1.00 1.00 1.00 17
                                       cotton 1.00 1.00 1.00 16
                                       grapes 1.00 1.00 1.00 18
                                         jute 0.95 1.00 0.98 21
                                  kidneybeans 1.00 1.00 1.00 20
```

```
lentil 1.00 0.94 0.97 17

maize 1.00 1.00 1.00 18

mango 1.00 1.00 1.00 21


 mothbeans 0.96 1.00 0.98 25

mungbean 1.00 1.00 1.00 17

muskmelon 1.00 1.00 1.00 23

orange 1.00 1.00 1.00 23

papaya 1.00 0.95 0.98 21

pigeonpeas 1.00 1.00 1.00 22

pomegranate 1.00 1.00 1.00 23

rice 1.00 1.00 1.00 25

watermelon 1.00 1.00 1.00 17


accuracy 1.00 440

macro avg 1.00 1.00 1.00 440

weighted avg 1.00 1.00 1.00 440

[[18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 16 0 0 1 0 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 20 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17]]
******* ***** ***** ****** ***** ***** ***** ** ******* ***** ***** ****** ***** ***** ****
**********************
```

```
model_accuracy.sort_values(ascending=False, by = 'Accuracy')
```

```
Model Accuracy Train_acc
2 RFC 0.997727 1.000000
3 GBC 0.995455 1.000000
4 XGB 0.995455 1.000000
1 DT 0.988636 1.000000
0 KNN 0.977273 0.988636
```

```
from sklearn.neighbors import KNeighborsClassifier kn_classifier = KNeighborsClassifier()
kn_classifier.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
pred_kn = kn_classifier.predict(x_test) evaluator(y_test, pred_kn)
```

```
Accuracy is: 0.9772727272727273

Classification Report:

                                                   precision recall f1-score support

apple 1.00 1.00 1.00 18

banana 1.00 1.00 1.00 18
```

blackgram 1.00 1.00 1.00 22

chickpea 1.00 1.00 1.00 23

coconut 1.00 1.00 1.00 15

coffee 1.00 1.00 1.00 17

cotton 1.00 0.94 0.97 16

grapes 1.00 1.00 1.00 18

jute 0.79 0.90 0.84 21

kidneybeans 0.91 1.00 0.95 20

lentil 1.00 1.00 1.00 17

maize 0.95 1.00 0.97 18

mango 1.00 1.00 1.00 21

mothbeans 1.00 1.00 1.00 25

mungbean 1.00 1.00 1.00 17

muskmelon 1.00 1.00 1.00 23

orange 1.00 1.00 1.00 23

papaya 1.00 1.00 1.00 21

pigeonpeas 1.00 0.91 0.95 22

pomegranate 1.00 1.00 1.00 23

rice 0.91 0.80 0.85 25

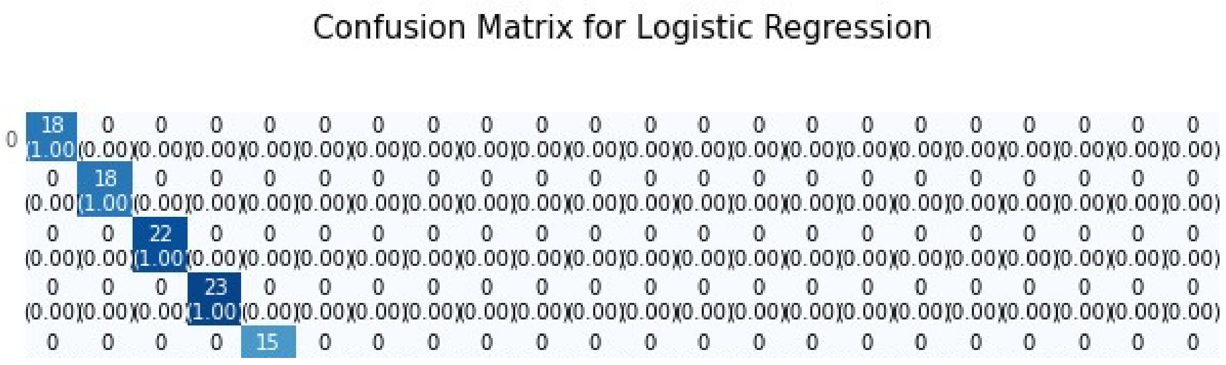
watermelon 1.00 1.00 1.00 17

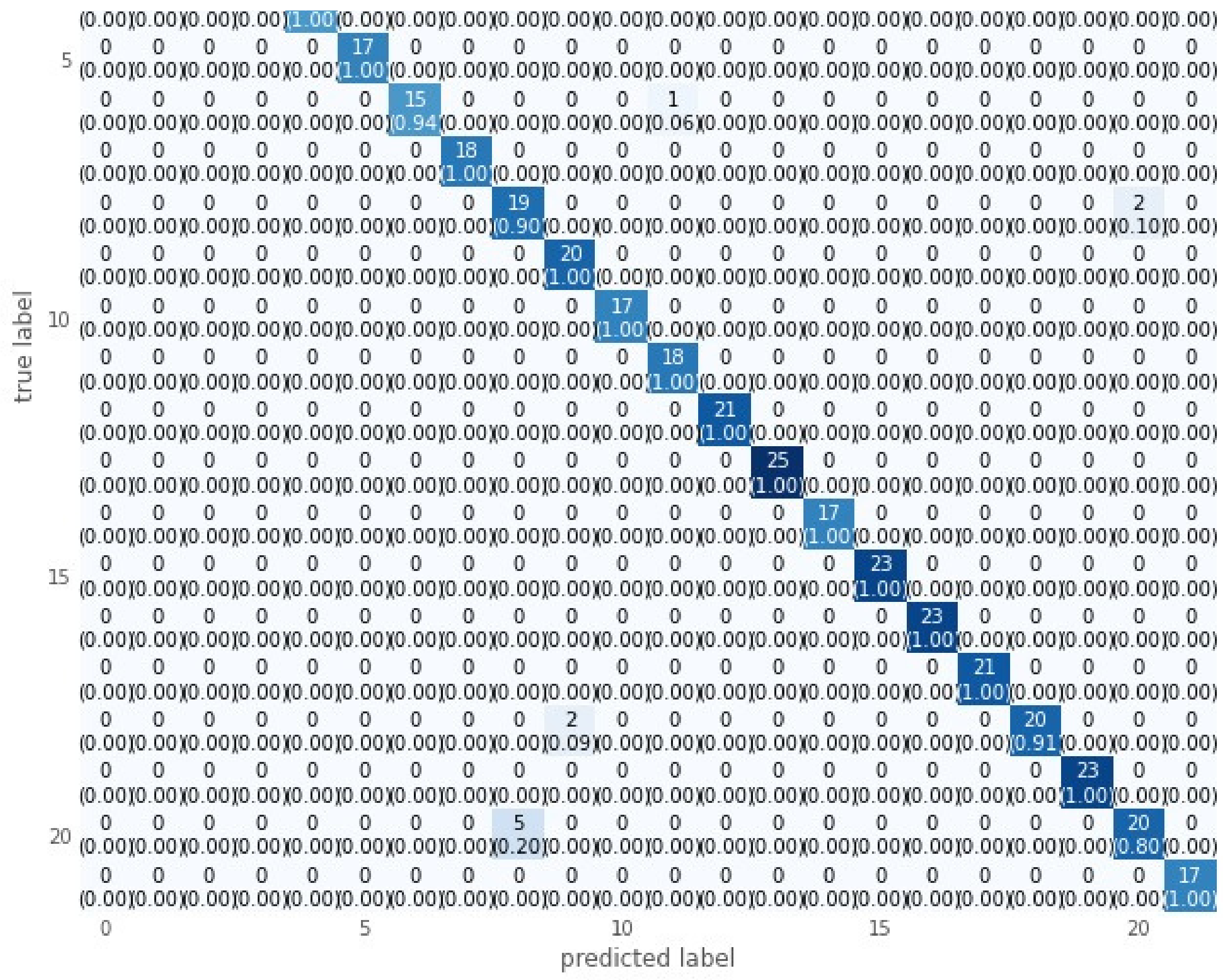
accuracy 0.98 440

macro avg 0.98 0.98 0.98 440

weighted avg 0.98 0.98 0.98 440

Confusion Matrix:





Real time Predictions

```
data.he ad()
```

```
 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
```

predicting the suitable crop for given parameters

```
prediction = kn_classifier.predict((np.array([[90,

40,

 40,

20,
80,
7,
200]])))
print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

```
The Suggested Crop for Given Climatic Condition is : ['jute']
```

checking data for oranges

```
data[data['label'] == 'orange'].head()
```

```
N P K temperature humidity ph rainfall label
1600 22 30 12 15.781442 92.510777 6.354007 119.035002 orange
1601 37 6 13 26.030973 91.508193 7.511755 101.284774 orange
1602 27 13 6 13.360506 91.356082 7.335158 111.226688 orange
1603 7 16 9 18.879577 92.043045 7.813917 114.665951 orange
1604 20 7 9 29.477417 91.578029 7.129137 111.172750 orange
```

```
# lets do some Real time Predictions
prediction = kn_classifier.predict((np.array([[20,
30,
10,
15,
90,
7.5,
100]])))
print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

```
The Suggested Crop for Given Climatic Condition is : ['orange']
```

Conclusion :

Modern farming is a complex and challenging industry, and farmers must overcome many obstacles to increase yield and profitability. By leveraging AI and ML technologies, farmers can gain real-time insights into their crops, soil, weather, and market conditions, making informed decisions that optimize yield and profit. The development of SoilSense, a one-stop app, is a significant step towards achieving this goal. SoilSense is a comprehensive platform that provides farmers with all necessary functionalities such as soil testing, crop selection, farm equipment control, weather forecasting, pest control, and market analysis all in one handy suite. The app's ability to fill the existing gaps in the market and providing premium customers with advanced features ensures that the interest of tech-savvy modern farmers is maintained. The implementation of open-source software, frameworks, and data privacy regulations would enable farmers to trust the app further. The development of SoilSense is a significant breakthrough in the industry and highly promising.