In [1]:
```
#for data minipulations
import numpy as np
import pandas as pd

#For data visualization
import matplotlib.pyplot as plt
import seaborn as sns

#For interactivity
from ipywidgets import interact
```

In [8]:
```
#Lets read the dataset
data=pd.read_csv('Agriculture.csv')
```

In [9]:
```
#Lets checkc the shape of the data set
print('Shape of the dataset is:',data.shape)
```

```
Shape of the dataset is: (2200, 8)
```

In [10]:
```
#lets check sample data first 5 rows
#Head of the dataset
data.head()
```

Out[10]:

	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

In [11]:
```
#check missing values or null.
#To deal with Null values
# 1. Fill-NA function is used to replace missings with mean, median and mode. Numerica
data.isnull().sum()
```

Out[11]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [14]:
```
#Lets check type of crops in dataset. value_counts() function gives total unique count
data['label'].value_counts()
```

Out[14]:
```
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
```

In [18]:
```
#Lets check the summary for all the crops
print("Average Ratio of Nitrogen in the soil:{0:.2f}".format(data['N'].mean()))
print("Average Ratio of Phospourus in the soil:{0:.2f} ".format(data['P'].mean()))
print("Average Ratio of Potasium in the soil:{0:.2f} ".format(data['K'].mean()))
print("Average Temperature Celcius:{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 Phospourus in the soil:53.36
Average Ratio of Potasium in the soil:48.15
Average Temperature Celcius:25.62
Average Relative humidity in %:71.48
Average ph value of the soil:6.47
Average rainfall in mm:103.46
```

In [23]:
```
#check each crop soil and climate requirement. @interact is the magic keyword it provi
#it can be used with only ipywidget
@interact
def summary(crops=list(data['label'].value_counts().index)):
    x=data[data['label']==crops]
    print("................................................")
    print("Statistics for Nitrogen")
    print("Minimum Nitrogen required:",x['N'].min())
    print("Average Nitrogen required:",x['N'].mean())
    print("Maximum Nitrogen required:",x['N'].max())
    print("................................................")
    print("Statistics for Phospourus")
    print("Minimum Phospourus required:",x['P'].min())
    print("Average Phospourus required:",x['P'].mean())
    print("Maximum Phospourus 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()))
    print("...................................................")
```

```
interactive(children=(Dropdown(description='crops', options=('rice', 'maize', 'jute',
'cotton', 'coconut', 'pa…
```

In [26]:

```
#Lets check average requirement of each crop with average conditions
@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')][conditior
    print("banana : {0:.2f}".format(data[(data['label']== 'banana')][conditions].mean(
    print("chickpea : {0:.2f}".format(data[(data['label']== 'chickpea')][conditions].m
    print("coconut : {0:.2f}".format(data[(data['label']== 'coconut')][conditions].mea
    print("coffee : {0:.2f}".format(data[(data['label']== 'coffee')][conditions].mean(
    print("cotton : {0:.2f}".format(data[(data['label']== 'cotton')][conditions].mean(
    print("grapes : {0:.2f}".format(data[(data['label']== 'grapes')][conditions].mean(
    print("jute : {0:.2f}".format(data[(data['label']== 'jute')][conditions].mean()))
    print("kidneybeans : {0:.2f}".format(data[(data['label']== 'kidneybeans')][conditi
    print("lentil : {0:.2f}".format(data[(data['label']== 'lentil')][conditions].mean(
    print("maize : {0:.2f}".format(data[(data['label']== 'maize')][conditions].mean())
    print("mango : {0:.2f}".format(data[(data['label']== 'mango')][conditions].mean())
    print("mothbeans : {0:.2f}".format(data[(data['label']== 'mothbeans')][conditions]
    print("mungbean : {0:.2f}".format(data[(data['label']== 'mungbean')][conditions].m
    print("muskmelon : {0:.2f}".format(data[(data['label']== 'muskmelon')][conditions]
    print("orange : {0:.2f}".format(data[(data['label']== 'orange')][conditions].mean(
    print("papaya : {0:.2f}".format(data[(data['label']== 'papaya')][conditions].mean(
    print("pigeonpeas : {0:.2f}".format(data[(data['label']== 'pigeonpeas')][conditior
    print("pomegranate : {0:.2f}".format(data[(data['label']== 'pomegranate')][conditi
    print("watermelon : {0:.2f}".format(data[(data['label']== 'watermelon')][conditior
```

```
interactive(children=(Dropdown(description='conditions', options=('N', 'P', 'K', 'tem
perature', 'ph', 'humidit…
```

In [27]:

```
#check paterns, which grows below avg and which grows above avg. Helps to identify unu
@interact
```

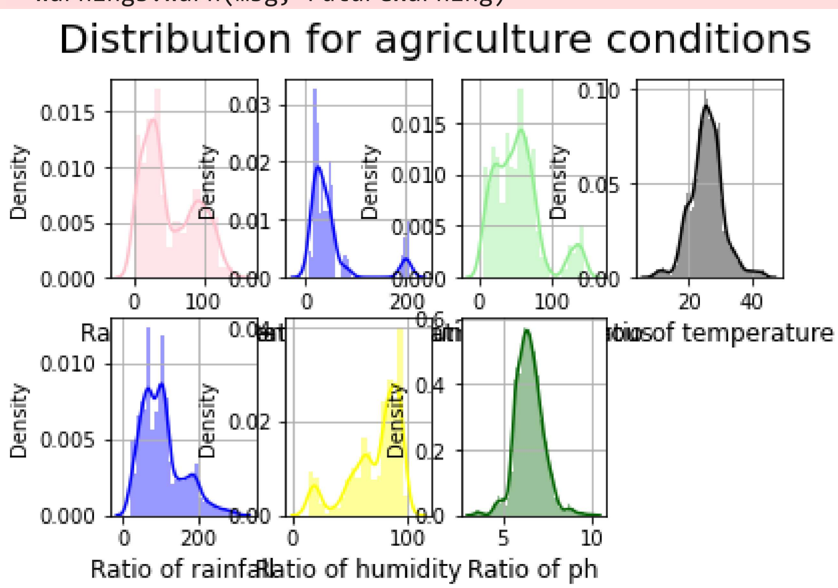
```
def compare(conditions=['N','P','K','temperature','ph','humidity','rainfall']):
    print("Crops which required greater than average",conditions,'\n')
    print(data[data[conditions] > data[conditions].mean()]['label'].unique())
    print("..........................................................")
    print("Crops which required less than average",conditions,'\n')
    print(data[data[conditions] <= data[conditions].mean()]['label'].unique())
```

```
interactive(children=(Dropdown(description='conditions', options=('N', 'P', 'K', 'tem
perature', 'ph', 'humidit…
```

In [33]:

```
plt.subplot(2, 4, 1)
sns.distplot(data['N'],color='pink')
plt.xlabel('Ratio of Nitrogen',fontsize=12)
plt.grid()

plt.subplot(2, 4, 2)
sns.distplot(data['K'],color='blue')
plt.xlabel('Ratio of Potassium',fontsize=12)
plt.grid()

plt.subplot(2,4,3)
sns.distplot(data['P'],color='lightgreen')
plt.xlabel('Ratio of Phospurous',fontsize=12)
plt.grid()

plt.subplot(2,4,4)
sns.distplot(data['temperature'],color='black')
plt.xlabel('Ratio of temperature',fontsize=12)
plt.grid()

plt.subplot(2,4,5)
sns.distplot(data['rainfall'],color='blue')
plt.xlabel('Ratio of rainfall',fontsize=12)
plt.grid()

plt.subplot(2,4,6)
sns.distplot(data['humidity'],color='yellow')
plt.xlabel('Ratio of humidity',fontsize=12)
plt.grid()

plt.subplot(2,4,7)
sns.distplot(data['ph'],color='darkgreen')
plt.xlabel('Ratio of ph',fontsize=12)
plt.grid()

plt.suptitle('Distribution for agriculture conditions',fontsize=20)
plt.show()
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [34]:

```python
#Lets find interesting facts
print("Some interesting patterns")
print("...........................................................")
print("Crop that require very high ratio of Nitrogen content in soil:",data[data['N']
print("Crop that require very high ratio of Phospurous content in soil:",data[data['P'
```

```
print("Crop that require very high ratio of Potassium content in soil:",data[data['K']
print("Crop that require very high rainfall:",data[data['rainfall'] > 200]['label'].ur
print("Crop that require very high temperature:",data[data['temperature'] > 40]['label
print("Crop that require very low temperature:",data[data['temperature'] < 10]['label'
print("Crop that require very high pH:",data[data['ph'] > 9]['label'].unique())
print("Crop that require very low pH:",data[data['ph'] < 4]['label'].unique())
print("Crop that require very high humidity:",data[data['humidity'] > 40]['label'].uni
print("Crop that require very low humidiy:",data[data['humidity'] < 20]['label'].uniqu
```

```
Some interesting patterns
.....................................................
Crop that require very high ratio of Nitrogen content in soil: ['cotton']
Crop that require very high ratio of Phospurous content in soil: ['grapes' 'apple']
Crop that require very high ratio of Potassium content in soil: ['grapes' 'apple']
Crop that require very high rainfall: ['rice' 'papaya' 'coconut']
Crop that require very high temperature: ['grapes' 'papaya']
Crop that require very low temperature: ['grapes']
Crop that require very high pH: ['mothbeans']
Crop that require very low pH: ['mothbeans']
Crop that require very high humidity: ['rice' 'maize' 'pigeonpeas' 'mothbeans' 'mungb
ean' 'blackgram' 'lentil'
 'pomegranate' 'banana' 'mango' 'grapes' 'watermelon' 'muskmelon' 'apple'
 'orange' 'papaya' 'coconut' 'cotton' 'jute' 'coffee']
Crop that require very low humidiy: ['chickpea' 'kidneybeans']
```

In [36]:
```
#Lets find out which crops can be grown only in Summer, Winter and rainy season
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())
```

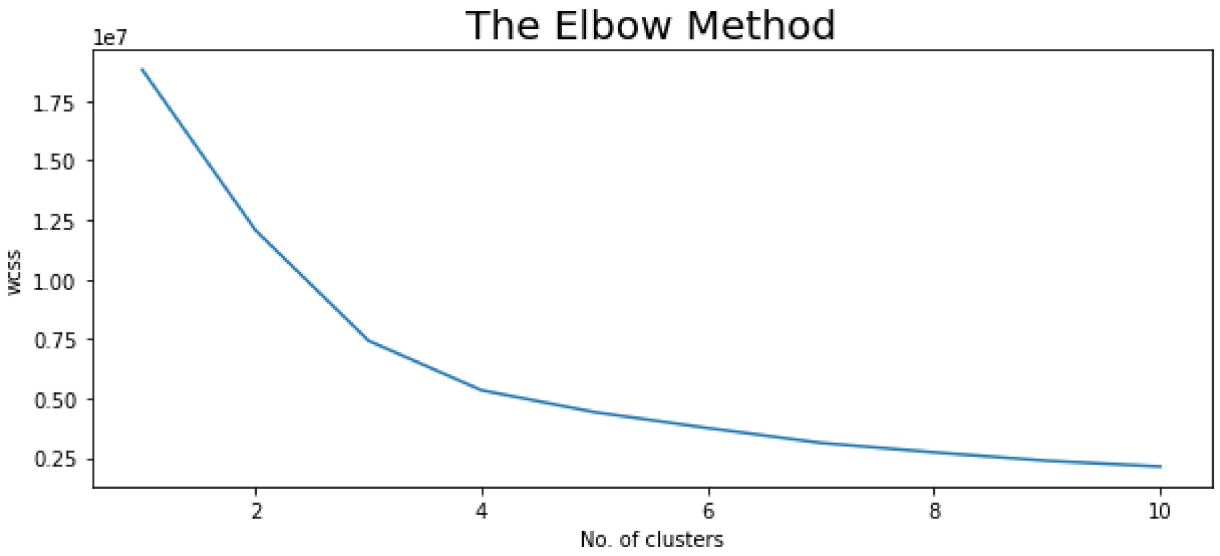
```
Summer Crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
....................................................
Winter Crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
....................................................
Rainy Crops
['rice' 'papaya' 'coconut']
```

In [45]:
```
#Lets find out which crops can be grown with similar soil and climate conditions using
#First lets immport sklearn cluser
from sklearn.cluster import KMeans

#Removing labels columns
x=data.drop(['label'],axis=1)

#Selecting all values of the data
x=x.values

#Checking the shaope
print(x.shape)
```

```
(2200, 7)
```

In [51]:
```
#plot elbow chart to identify number of clusters
#Lets find the optimum number of clusters in data
```

```
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 plat the results
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method',fontsize=20)
plt.xlabel('No. of clusters')
plt.ylabel('wcss')
plt.show()
```



In [55]:
```
#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 cluster of each crops
print("Lets check the result after applying K means clustering algorythm 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 result after applying K means clustering algorythm analysis

Crops in first cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'm
ungbean'
 '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']
```

In [56]:
```
#Evaluation matrix - to find model accuracy
#Lets split the data for predictive modelling. Remove labels from data set so that mod
y=data['label']
x=data.drop(['label'],axis=1)
print("Shape of x:",x.shape)
print("Shape of y:",y.shape)
```

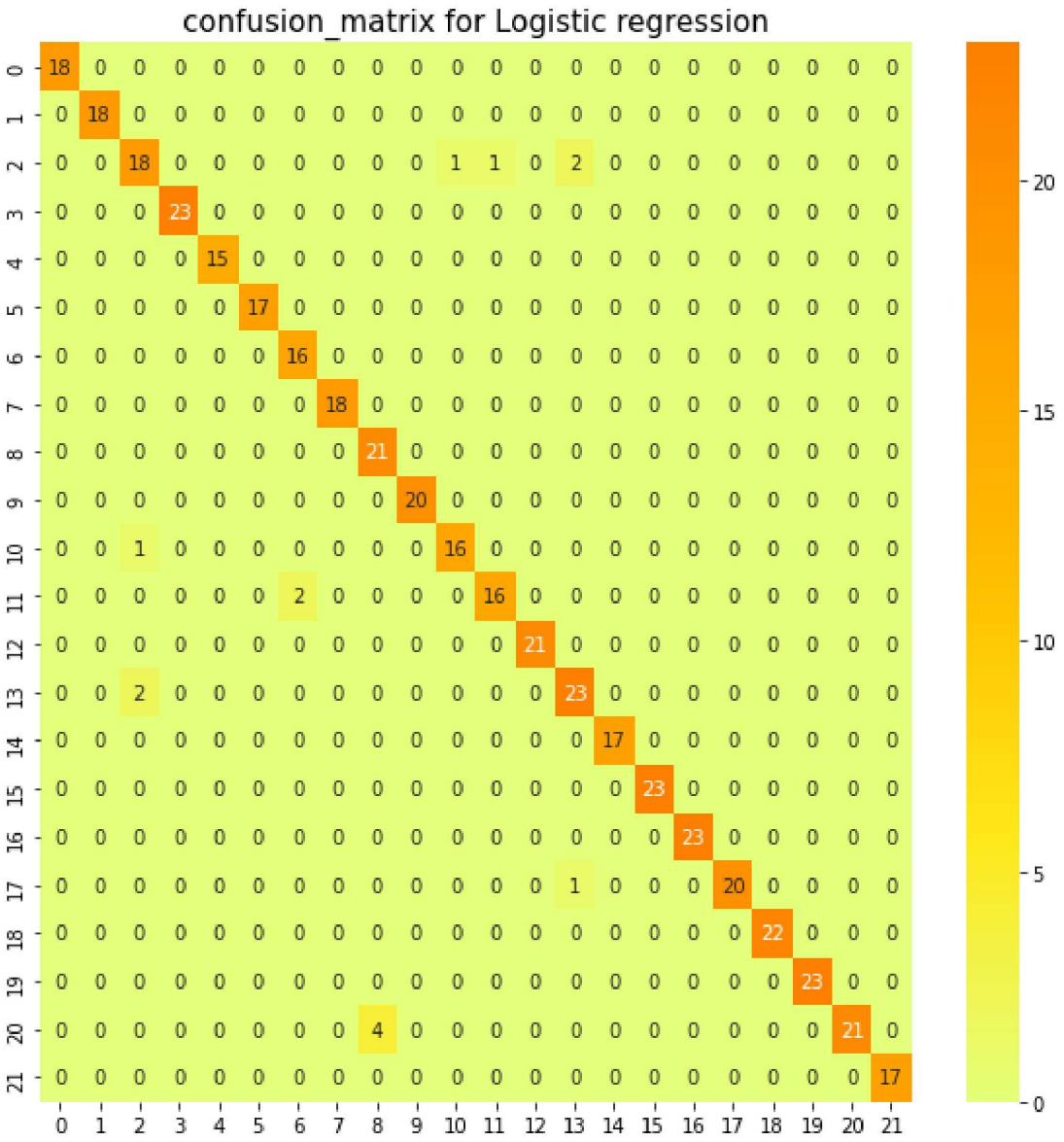
```
Shape of x: (2200, 7)
Shape of y: (2200,)
```

In [57]:
```
#Lets divide data into 2 sets, Training set and Testing set. We do it to check the acc
# we split 80% training:20% testing
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("Shape of the x train:",x_train.shape)
print("Shape of the x test:",x_test.shape)
print("Shape of the y train:",y_train.shape)
print("Shape of the y test:",y_test.shape)
```

```
Shape of the x train: (1760, 7)
Shape of the x test: (440, 7)
Shape of the y train: (1760,)
Shape of the y test: (440,)
```

In [59]:
```
#Lets to ML predictive model
#Logistic regression
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train, y_train)
y_pred=model.predict(x_test)
```

In [61]:
```
#using training data we will train Logistic regression model.
#using predict function
#we compare y_test with y_pred by classification report
#Lets evaluate the model performance
from sklearn.metrics import confusion_matrix

#Let print the confusion matrix first
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()
```



In [64]:
```
#Using classification report we can know precision and recall value. If both are good
#lets print classification report also
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	18
banana	1.00	1.00	1.00	18
blackgram	0.86	0.82	0.84	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	0.89	1.00	0.94	16
grapes	1.00	1.00	1.00	18
jute	0.84	1.00	0.91	21
kidneybeans	1.00	1.00	1.00	20
lentil	0.94	0.94	0.94	17
maize	0.94	0.89	0.91	18
mango	1.00	1.00	1.00	21
mothbeans	0.88	0.92	0.90	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	0.84	0.91	25
watermelon	1.00	1.00	1.00	17
accuracy			0.97	440
macro avg	0.97	0.97	0.97	440
weighted avg	0.97	0.97	0.97	440

In [65]:
```
#Check if prediction of the model is correct
#To be sure, check from the some of the datapoints
prediction=model.predict((np.array([[90,
                                     40,
                                     40,
                                     20,
                                     80,
                                     7,
                                    200]])))
print("The suggested crop for the given climatic condition is:", prediction)
```

```
The suggested crop for the given climatic condition is: ['rice']
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does no
t have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [ ]: