In [1]:
```python
#for manipulations
import numpy as np
import pandas as pd


#for data visualizations
import matplotlib.pyplot as plt
import seaborn as sns


#for interactivity
from ipywidgets import interact
```

In [2]:
```python
#lets read the dataset
data = pd.read_csv('data.csv')
```

In [3]:
```python
#lets check the shape of the dataset
print("shape of the dataset:",data.shape)
```

```
shape of the dataset: (2200, 8)
```

In [4]:
```python
#lets check the head of the dataset
data.head()
```

Out[4]:

	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 [5]:
```python
#lets check if there is any missing value present in the dataset
data.isnull().sum()
```

Out[5]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [6]:
```python
#lets check the crops present in this dataset
data['label'].value_counts()
```

Out[6]:
```
papaya         100
lentil         100
cotton         100
coconut        100
watermelon     100
blackgram      100
banana         100
kidneybeans    100
jute           100
pigeonpeas     100
chickpea       100
muskmelon      100
maize          100
pomegranate    100
rice           100
orange         100
apple          100
grapes         100
coffee         100
mothbeans      100
mungbean       100
mango          100
Name: label, dtype: int64
```

In [7]:
```python
#lets check the summary statistics for all the crops
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 ratio of temperature 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()))
```

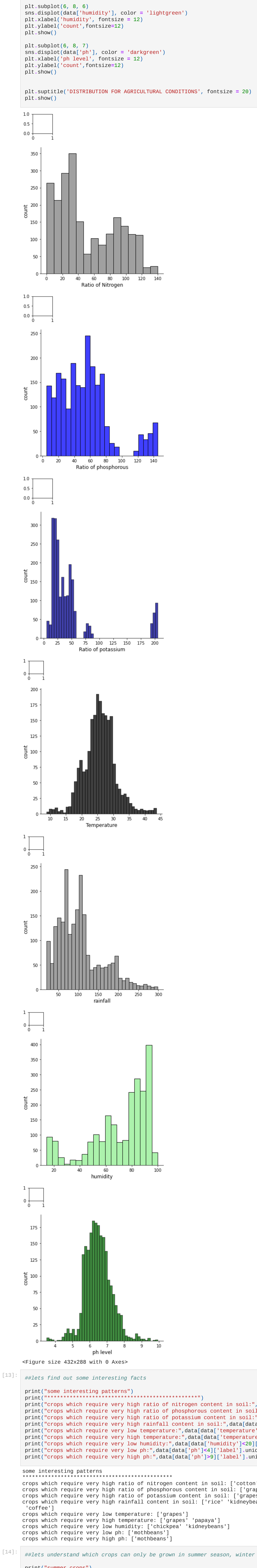
```
Average Ratio of Nitrogen in the Soil : 50.55
Average Ratio of phosphorous in the soil :53.36
Average ratio of potassium in the soil : 48.15
Average ratio of temperature in celsius :25.62
Average Relative humidity in % :71.48
Average PH value of the soil :6.47
Average Rainfall in mm :103.46
```

In [8]:
```python
#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 Nitrogen 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 :", x['temperature'].min())
    print("Average temperature required :", x['temperature'].mean())
    print("Maximum temperature required :", x['temperature'].max())
    print("*****************************************")
    print("statistics for humidity")
    print("Minimum humidity required :", x['humidity'].min())
    print("Average humidity required :", x['humidity'].mean())
    print("Maximum humidity required :", x['humidity'].max())
    print("*****************************************")
    print("statistics for PH")
    print("Minimum PH required :", x['ph'].min())
    print("Average PH required :", x['ph'].mean())
    print("Maximum PH required :", x['ph'].max())
    print("*****************************************")
    print("statistics for rainfall")
    print("Minimum rainfall required :", x['rainfall'].min())
    print("Average rainfall required :", x['rainfall'].mean())
    print("Maximum rainfall required :", x['rainfall'].max())
    print("*****************************************")
```

In [9]:
```python
##lets compare the average requirement for each crops 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'][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("MungBeans : {0:.2f}".format(data[data['label'] == 'mungbeans'][conditions].mean()))
    print("Oranges : {0:.2f}".format(data[(data['label'] == 'orange')][conditions].mean()))
    print("Chickpeas : {0:.2f}".format(data[data['label'] == 'chickpeas'][conditions].mean()))
    print("Lentils : {0:.2f}".format(data[(data['label'] == 'lentils')][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'] == 'pigeon peas')][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()))
```

In [10]:
```python
#lets make this function more intuitive

@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())
```

In [12]:
```python
#distribution for agricultural conditions
plt.subplot(4, 6, 1)
sns.distplot(data['N'], color = 'grey')
plt.xlabel('Ratio of Nitrogen', fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(4, 6, 2)
sns.distplot(data['P'], color = 'blue')
plt.xlabel('Ratio of phosphorous', fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(4,6, 3)
sns.distplot(data['K'], color = 'darkblue')
plt.xlabel('Ratio of potassium',fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(6, 8, 4)
sns.distplot(data['temperature'], color = 'black')
plt.xlabel('Temperature', fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(6, 8, 5)
sns.distplot(data['rainfall'], color = 'grey')
plt.xlabel('rainfall',fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(6, 8, 6)
sns.distplot(data['humidity'], color = 'lightgreen')
plt.xlabel('humidity', fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()

plt.subplot(6, 8, 7)
sns.distplot(data['ph'], color = 'darkgreen')
plt.xlabel('ph level', fontsize = 12)
plt.ylabel('count',fontsize=12)
plt.show()


plt.suptitle('DISTRIBUTION FOR AGRICULTURAL CONDITIONS', fontsize = 20)
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [13]:
```python
##lets find out some interesting facts

print("some interesting patterns")
print("************************************************")
print("crops which require very high ratio of nitrogen content in soil:",data[data['N']>120]['label'].unique())
print("crops which require very high ratio of phosphorous content in soil:",data[data['P']>100]['label'].unique())
print("crops which require very high ratio of potassium content in soil:",data[data['K']>120]['label'].unique())
print("crops which require very high rainfall content in soil:",data[data['rainfall']>120]['label'].unique())
print("crops which require very low temperature:",data[data['temperature']<10]['label'].unique())
print("crops which require very high temperature:",data[data['temperature']>40]['label'].unique())
print("crops which require very low humidity:",data[data['humidity']<20]['label'].unique())
print("crops which require very low ph:",data[data['ph']<4]['label'].unique())
print("crops which require very high ph:",data[data['ph']>9]['label'].unique())
```

```
some interesting patterns
************************************************
crops which require very high ratio of nitrogen content in soil: ['cotton']
crops which require very high ratio of phosphorous content in soil: ['grapes' 'apple']
crops which require very high ratio of potassium content in soil: ['grapes' 'apple']
crops which require very high rainfall content in soil: ['rice' 'kidneybeans' 'pigeonpeas' 'apple' 'papaya' 'coconut' 'jute'
 'coffee']
crops which require very low temperature: ['grapes']
crops which require very high temperature: ['grapes' 'papaya']
crops which require very low humidity: ['chickpea' 'kidneybeans']
crops which require very low ph: ['mothbeans']
crops which require very high ph: ['mothbeans']
```

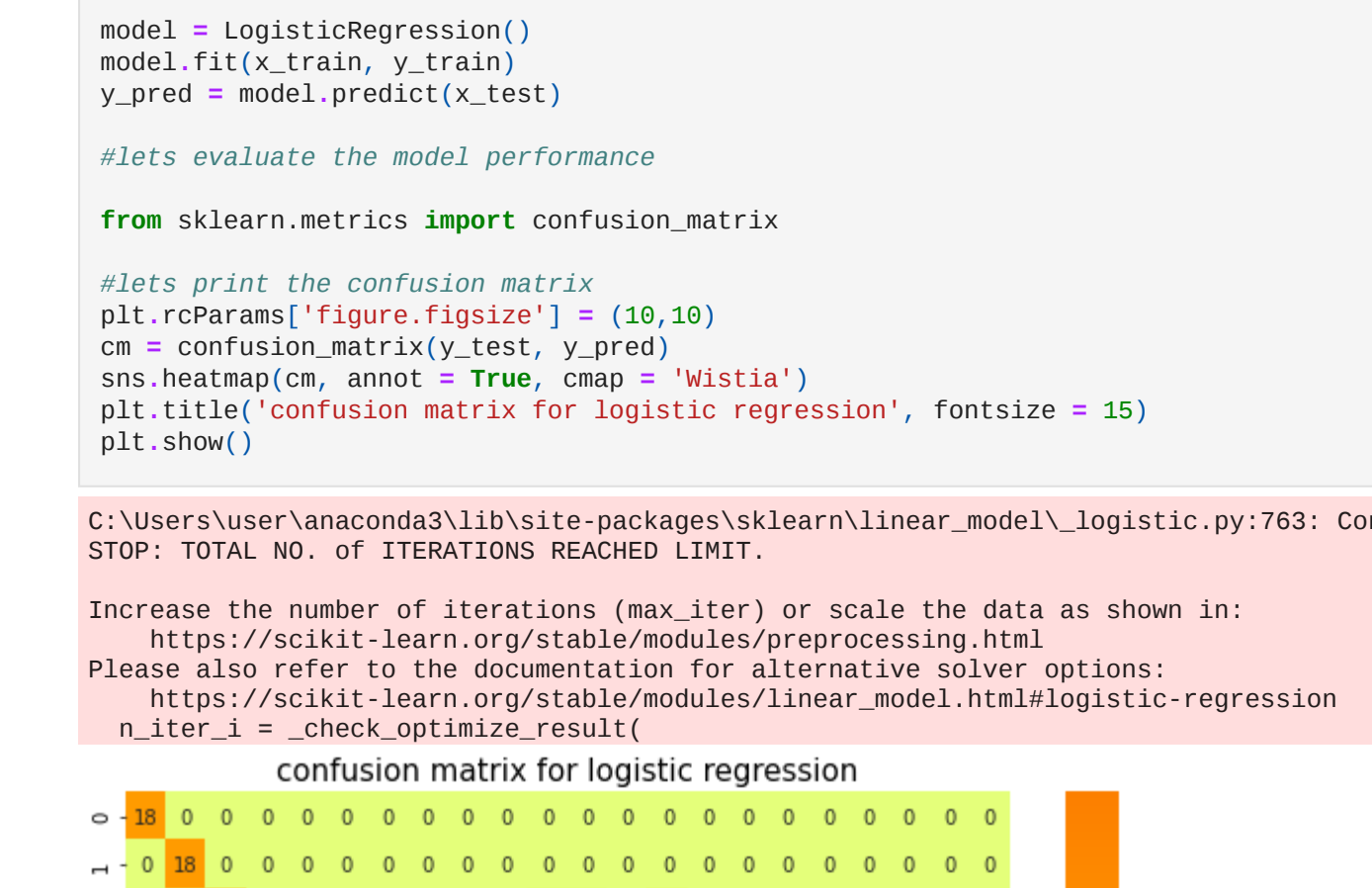
In [14]:
```python
##lets understand which crops can only be grown in summer season, winter season and rainy season

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 [15]:
```python
##lets import KMeans from sKlearn library for making clusters

from sklearn.cluster import KMeans

#removing the labels column
x = data.drop(['label'], axis = 1)

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

#checking the shape
print(x.shape)
```

```
(2200, 7)
```

In [16]:
```python
#lets determine the optimum number of clusters within the dataset

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.title('THE ELBOW METHOD', fontsize = 20)
plt.xlabel('NO. OF CLUSTERS ')
plt.ylabel('wcss')
plt.show()
```



In [17]:
```python
#lets implement the KMeans 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 results after aaplying 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 aaplying 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']
```

In [18]:
```python
#lets split dataset for predictive learning

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

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

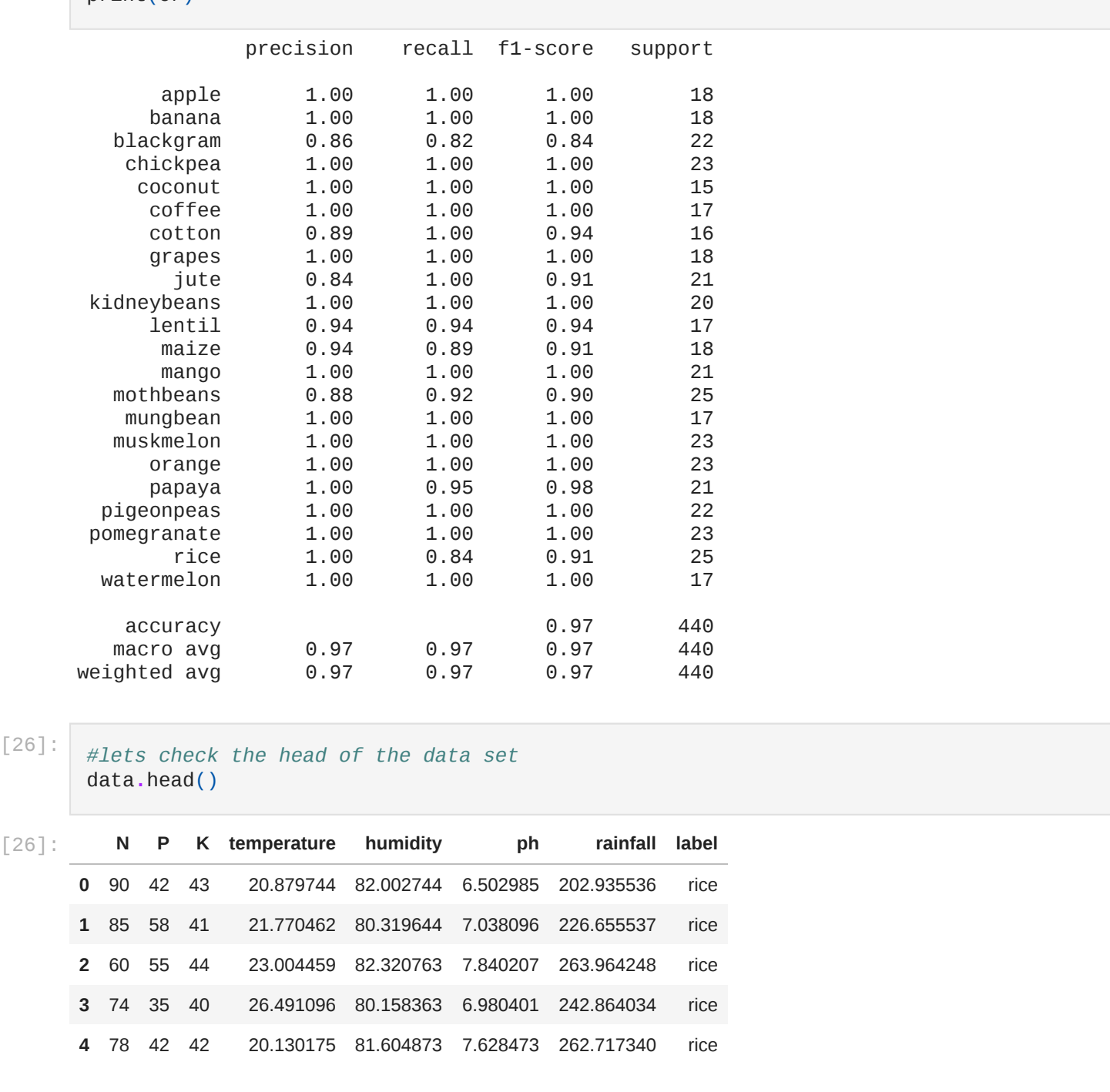
```
shape of x: (2200, 7)
shape of y: (2200,)
```

In [19]:
```python
#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,)
```

In [22]:
```python
#lets create a predictive model

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

#lets evaluate the model performance

from sklearn.metrics import confusion_matrix

#lets print the 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()
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: 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(
```



In [23]:
```python
#lets print the 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 [26]:
```python
#lets check the head of the data set
data.head()
```

Out[26]:

	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 [30]:
```python
prediction = model.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: ['rice']
```

In [ ]:

In [ ]: