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

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

# interactivity
from ipywidgets import interact
```

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

In [3]:
```
data
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...
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

2200 rows × 8 columns

In [4]:
```
print("Shape of the DataSet:", data.shape)
```

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

In [5]:
```
data.head()
```

Out[5]:

	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 [6]:
```
data.isnull().sum()
```

Out[6]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [9]:
```
#Lets check the Crops present in this Dataset
data['label'].value_counts()
```

Out[9]:
```
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 [17]:
```
# Lets check the Sumary 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 Potassiun in the Soil:{0:.2f}".format(data['K'].mean()))
print("Average 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 the m:{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 Potassiun in the Soil:48.15
Average Temperature in Celsius:25.62
Average Relative Humidity in %:71.48
Average PH Value of the Soil:6.47
Average Rainfall in the m:103.46
```

In [34]:
```
#Lets check the Sumary Statistics for each of the Crops

@interact
def sumary (crops = list(data['label'].value_counts().index)):
    x = data[data['label']== crops]
    print('---------------------------------------------------')
    print("Statistics for Nitrogen")
    print("Minimum Nitrogen required :{0:.2f}",x['N'].min())
    print("Avarage Nitrogen required :{0:.2f}",x['N'].mean())
```

```
    print("Maximum Nitrogen required :{0:.2f}",x['N'].max())
    print("--------------------------------------------------")
    print("Statistics for Phosphorous")
    print("Minimum Phosphorous required :{0:.2f}",x['P'].min())
    print("Avarage Phosphorous required :{0:.2f}",x['P'].mean())
    print("Maximum Phosphorous required :{0:.2f}",x['P'].max())
    print('--------------------------------------------------')
    print("Statistics for Potassiun")
    print("Minimum Potassiun required :{0:.2f}",x['K'].min())
    print("Avarage Potassiun required :{0:.2f}",x['K'].mean())
    print("Maximum Potassiun required :{0:.2f}",x['K'].max())
    print('--------------------------------------------------')
    print("Statistics for Temperature")
    print("Minimum Temperature required :{0:.2f}",x['temperature'].min())
    print("Avarage Temperature required :{0:.2f}",x['temperature'].mean())
    print("Maximum Temperature required :{0:.2f}",x['temperature'].max())
    print('--------------------------------------------------')
    print("Statistics for Relative Humidity")
    print("Minimum Relative Humidity required :{0:.2f}",x['humidity'].min())
    print("Avarage Relative Humidity required :{0:.2f}",x['humidity'].mean())
    print("Maximum Relative Humidity required :{0:.2f}",x['humidity'].max())
    print('--------------------------------------------------')
    print("Statistics for PH Value")
    print("Minimum PH Value required :{0:.2f}",x['ph'].min())
    print("Avarage PH Value required :{0:.2f}",x['ph'].mean())
    print("Maximum PH Value required :{0:.2f}",x['ph'].max())
    print('--------------------------------------------------')
    print("Statistics for Rainfall")
    print("Minimum Rainfall required :{0:.2f}",x['rainfall'].min())
    print("Avarage Rainfall required :{0:.2f}",x['rainfall'].mean())
    print("Maximum Rainfall required :{0:.2f}",x['rainfall'].max())
    print('--------------------------------------------------')
```

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

In [69]:

```
@interact
def compare (conditions = ['N','P','K','temperature','ph','humidity','rainfall']):
    print('Avarage Value for',conditions,'is {0:.2f}'.format(data[conditions].mean
    print("-----------------------------------------------------------------------"
    print('1. rice:{0:2f}'.format(data[(data['label'] == 'rice')][conditions].mean
    print('2. maize:{0:2f}'.format(data[(data['label'] == 'maize')][conditions].mea
    print('3. chickpea:{0:2f}'.format(data[(data['label'] == 'chickpea')][conditio
    print('4. kidneybeans:{0:2f}'.format(data[(data['label'] == 'kidneybeans')][co
    print('5. pigeonpeas:{0:2f}'.format(data[(data['label'] == 'pigeonpeas')][cond
    print('6. mothbeans:{0:2f}'.format(data[(data['label'] == 'mothbeans')][condit
    print('7. mungbean:{0:2f}'.format(data[(data['label'] == 'mungbean')][conditio
    print('8. blackgram:{0:2f}'.format(data[(data['label'] == 'blackgram')][condit
    print('9. lentil:{0:2f}'.format(data[(data['label'] == 'lentil')][conditions].
    print('10.pomegranate:{0:2f}'.format(data[(data['label'] == 'pomegranate')][co
    print('11.banana:{0:2f}'.format(data[(data['label'] == 'banana')][conditions].
    print('12.mango:{0:2f}'.format(data[(data['label'] == 'mango')][conditions].mea
    print('13.grapes:{0:2f}'.format(data[(data['label'] == 'grapes')][conditions].
    print('14.watermelon:{0:2f}'.format(data[(data['label'] == 'watermelon')][cond
    print('15.muskmelon:{0:2f}'.format(data[(data['label'] == 'muskmelon')][condit
    print('16.apple:{0:2f}'.format(data[(data['label'] == 'apple')][conditions].mea
    print('17.orange:{0:2f}'.format(data[(data['label'] == 'orange')][conditions].
    print('18.papaya:{0:2f}'.format(data[(data['label'] == 'papaya')][conditions].
    print('19.coconut:{0:2f}'.format(data[(data['label'] == 'coconut')][conditions
    print('20.cotton:{0:2f}'.format(data[(data['label'] == 'cotton')][conditions].
    print('21.jute:{0:2f}'.format(data[(data['label'] == 'jute')][conditions].mean
    print('22.coffee:{0:2f}'.format(data[(data['label'] == 'coffee')][conditions].
```

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

In [65]:
```
data['label'].unique().size
```

Out[65]:
```
22
```

In [66]:
```
data['label'].unique()
```

Out[66]:
```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

In [76]:
```
data['label'].value_counts().sample
```

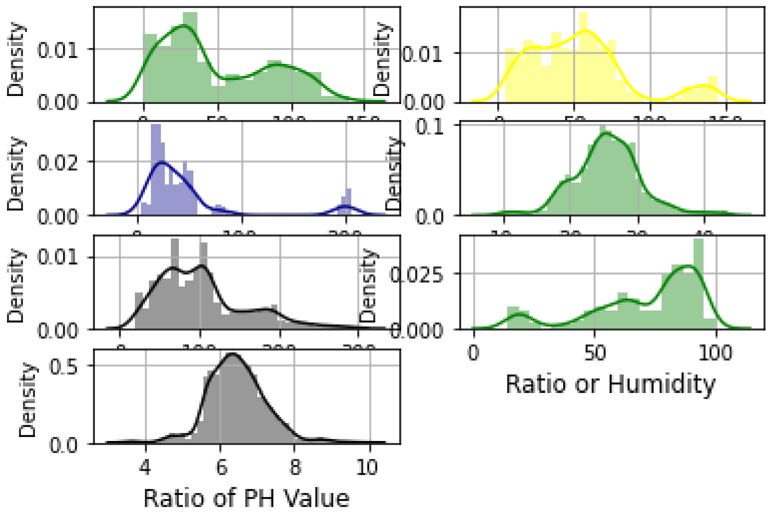
Out[76]:
```
<bound method NDFrame.sample of rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
kidneybeans    100
chickpea       100
coffee         100
Name: label, dtype: int64>
```

In [78]:
```
# lets make this function more Intuitive

@interact
def compare (conditions = ['N','P','K','temperature','ph','humidity','rainfall']):
    print('Crops which require greater than avarage',conditions,'\n')
    print(data[data[conditions]>data[conditions].mean()]['label'].unique())
    print('-----------------------------------------------------------------')
    print('Crops which require greater than avarage',conditions,'\n')
    print(data[data[conditions]<=data[conditions].mean()]['label'].unique())
```

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

In [125…
```
import warnings
warnings.filterwarnings('ignore')

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

```
plt.subplot(4, 2, 2)
sns.distplot(data['P'], color='yellow')
plt.xlabel('Ratio or Phosphorous',fontsize =12)
plt.grid()



plt.subplot(4, 2, 3)
sns.distplot(data['K'], color='darkblue')
plt.xlabel('Ratio or ptassium',fontsize =12)
plt.grid()



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



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



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



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


plt.show()
```



In [127…

```
## lets find out size Interesting Facts

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

```
print("crops which requires very High Ratio of Nitrogen content in Soil:",data[data
print("crops which requires very High Ratio of Phosphorous content in Soil:",data[d
print("crops which requires very High Ratio of ptassium content in Soil:",data[data
print("crops which requires very High Rainfall:",data[data['rainfall']>200]['label
print("crops which requires very Low Temperature:",data[data['temperature']<10]['la
print("crops which requires very High Temperature:",data[data['temperature']>40]['l
print("crops which requires very Low Humidity:",data[data['humidity']<20]['label']
print("crops which requires very High PH Value:",data[data['ph']<4]['label'].unique
print("crops which requires very High PH Value:",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' 'ap
ple']
crops which requires very High Ratio of ptassium content in Soil: ['grapes' 'appl
e']
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 High PH Value: ['mothbeans']
crops which requires very High PH Value: ['mothbeans']
```

In [129…

```
###Lets understan which crops can only be Gron in Summer Season, Winter Season and
print("summer Crops")
print(data[(data['temperature']>30) & (data['humidity']>50)]['label'].unique())
print("-----------------------------------------------------------------")
print("Wrinter 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']
-----------------------------------------------------------------
Wrinter Crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
-----------------------------------------------------------------
Rainy Crops
['rice' 'papaya' 'coconut']
```

In [130…

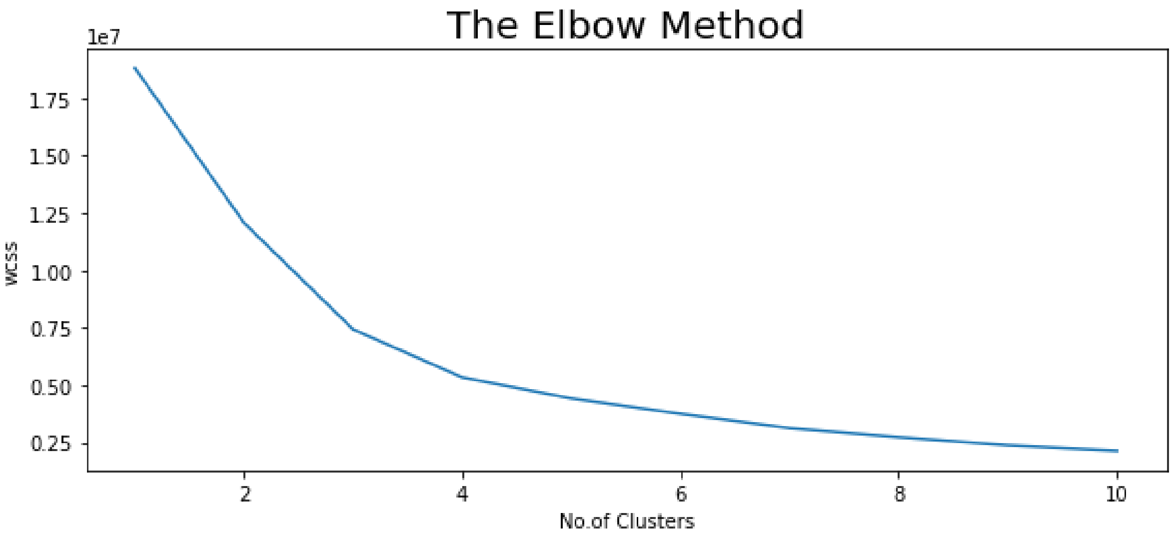
```
from sklearn.cluster import KMeans
# removing the labels column
x= data.drop(['label'],axis=1)

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

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

```
(2200, 7)
```

In [140…

```
# lets determint the Optimum Number of Clusters with in 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_s
    km.fit(x)
    wcss.append(km.inertia_)
```

In [142…
```
#lets plot the reslts

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



In [149…
```
# Lets implement the K Means algorithm to perform Clustering analysis
km =KMeans(n_clusters= 4, init = 'k-means++',max_iter=300, n_init= 30, random_state
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']==0]['label'].unique())
print('---------------------------------------------------------------')
print("Crops in Third Cluster:",z[z['cluster']==0]['label'].unique())
print('---------------------------------------------------------------')
print("Crops in Forth Cluster:",z[z['cluster']==0]['label'].unique())
```

```
Lets check the Results After Applying the K Means Clustering Analysis

Crops in First Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
---------------------------------------------------------------
Crops in Second Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
---------------------------------------------------------------
Crops in Third Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
---------------------------------------------------------------
Crops in Forth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
```

In [150…
```
# Let split the Dataset for Predictive Modeling
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 [151…

```
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_stat

print("The Shape of x trin:",x_train.shape)
print("The Shape of x test:",x_test.shape)
print("The Shape of y trin:",x_train.shape)
print("The Shape of y test:",x_test.shape)
```

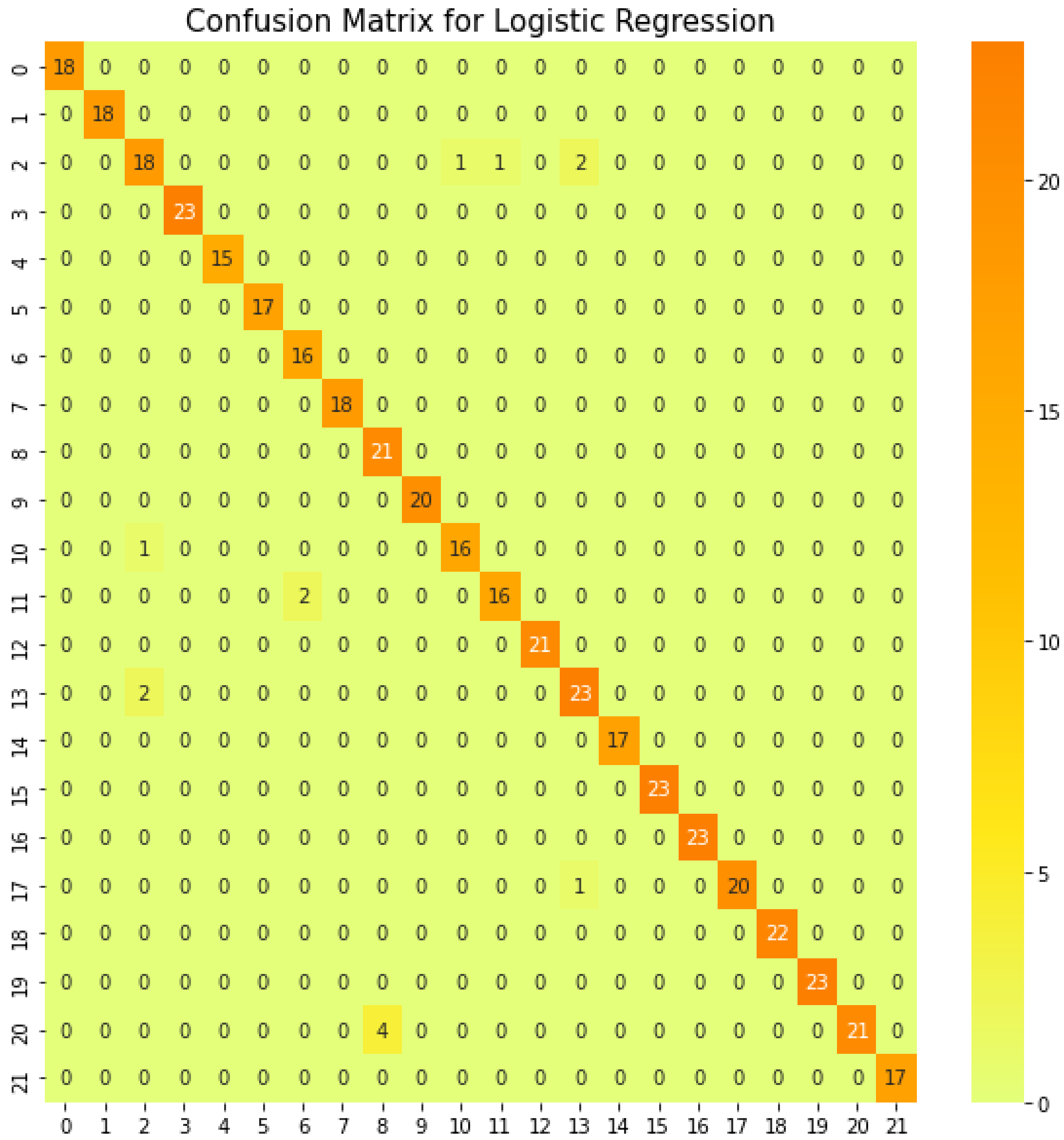
```
The Shape of x trin: (1760, 7)
The Shape of x test: (440, 7)
The Shape of y trin: (1760, 7)
The Shape of y test: (440, 7)
```

In [152…

```
from sklearn.linear_model import LogisticRegression

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

In [155…

```
# lets evaluate the Model Perfromce
from sklearn.metrics import confusion_matrix

#let printthe 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 [158…

```
#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 [159…

```
data.head()
```

Out[159]:

	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 [161…

```
predicition = model.predict((np.array([[90,40,40,20,80,7,200]])))
print("The Suggested Crop for Given Climatic Conditon is :",predicition)
```

```
The Suggested Crop for Given Climatic Conditon is : ['rice']
```

In [ ]: