In [66]:
```python
#importing the libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ipywidgets import interact
```

In [67]:
```python
#to read a dataset we use read_csv function and assign to df variable
df = pd.read_csv('data (2).csv')
```

In [68]:
```python
#to check the rows and column

print('shape of the dataset : ',df.shape)
```

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

In [69]:
```python
df
```

Out[69]:

	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 [70]:
```python
#to display top 5 rows
df.head(100)
```

Out[70]:

	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
...	...	...	...	...	...	...	...	...
95	88	46	42	22.683191	83.463583	6.604993	194.265172	rice
96	93	47	37	21.533463	82.140041	6.500343	295.924880	rice
97	60	55	45	21.408658	83.329319	5.935745	287.576694	rice
98	78	35	44	26.543481	84.673536	7.072656	183.622266	rice
99	65	37	40	23.359054	83.595123	5.333323	188.413665	rice

100 rows × 8 columns

In [71]:
```python
#to check how many missing values are present in the dataset

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

Out[71]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [72]:
```python
#lets check the crop present in the dataset

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

Out[72]:
```
grapes         100
lentil         100
maize          100
kidneybeans    100
coffee         100
banana         100
rice           100
muskmelon      100
mothbeans      100
chickpea       100
coconut        100
pomegranate    100
jute           100
mango          100
cotton         100
blackgram      100
mungbean       100
orange         100
watermelon     100
pigeonpeas     100
apple          100
papaya         100
Name: label, dtype: int64
```

In [73]:
```python
# Average climatic and soil requirements
# On an average distribution of nitrogen, potassium and phosphorous in a soil is
# on an average tempetrature,relative humidity,ph value and rainfall

print('Average ratio of nitrogen in soil : {0:.2f}'.format(df['N'].mean()))
print('Average ratio of phosphorous in soil : {0:.2f}'.format(df['P'].mean()))
print('Average ratio of potassium in soil : {0:.2f}'.format(df['K'].mean()))
print('Average temperature in celsius : {0:.2f}'.format(df['temperature'].mean()))
print('Average relative humidity in % : {0:.2f}'.format(df['humidity'].mean()))
print('Average ph value of the soil : {0:.2f}'.format(df['ph'].mean()))
print('Average rainfall in mm : {0:.2f}'.format(df['rainfall'].mean()))
```

```
Average ratio of nitrogen in soil : 50.55
Average ratio of phosphorous in soil : 53.36
Average ratio of potassium in 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 mm : 103.46
```

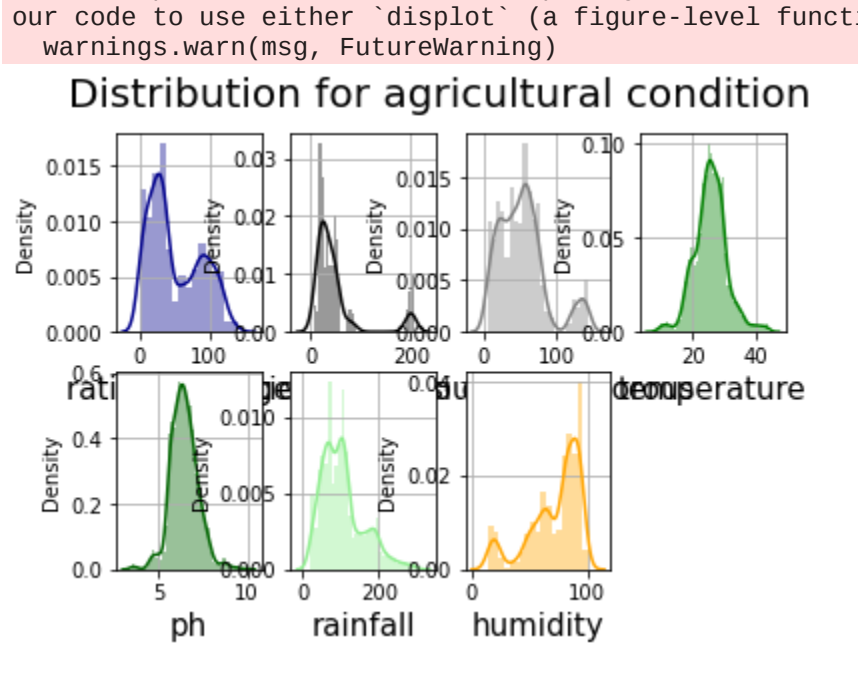
In [74]:
```python
@interact
def summary(crops = list(df['label'].value_counts().index)):
    x = df[df['label'] == crops]
    print('-------------------------')
    print('statistics for nitrogen')
    print('min nitrogen required :',x['N'].min())
    print('mean nitrogen required :',x['N'].mean())
    print('max nitrogen required :',x['N'].max())
    print('-------------------------')
    print('statistics for phosphorous')
    print('min phosphorous required :',x['P'].min())
    print('mean phosphorous required :',x['P'].mean())
    print('max phosphorous required :',x['P'].max())
    print('-------------------------')
    print('statistics for nitrogen')
    print('min potassium required :',x['K'].min())
    print('mean potassium required :',x['K'].mean())
    print('max potassium required :',x['K'].max())
    print('-------------------------')
    print('statistics for temperature')
    print('min temperature required :{0:.2f}'.format(x['temperature'].min()))
    print('mean temperature required :{0:.2f}'.format(x['temperature'].mean()))
    print('max temperature required :{0:.2f}'.format(x['temperature'].max()))
    print('-------------------------')
    print('statistics for ph')
    print('min temperature required :{0:.2f}'.format(x['ph'].min()))
    print('mean temperature required :{0:.2f}'.format(x['ph'].mean()))
    print('max temperature required :{0:.2f}'.format(x['ph'].max()))
    print('-------------------------')
    print('statistics for humidity')
    print('min temperature required :{0:.2f}'.format(x['humidity'].min()))
    print('mean temperature required :{0:.2f}'.format(x['humidity'].mean()))
    print('max temperature required :{0:.2f}'.format(x['humidity'].max()))
    print('-------------------------')
    print('statistics for rainfall')
    print('min temperature required :{0:.2f}'.format(x['rainfall'].min()))
    print('mean temperature required :{0:.2f}'.format(x['rainfall'].mean()))
    print('max temperature required :{0:.2f}'.format(x['rainfall'].max()))
```

In [75]:
```python
@interact
def compare(conditions = ['N','P','K','temperature','ph','humidity','rainfall']):
    print('average value for',conditions,'is {0:.2f}'.format(df[conditions].mean()) )
    print('--------------------------------------------')
    print('Rice :{0:.2f}'.format(df[(df['label'] =='rice')][conditions].mean()) )
    print('mungbean :{0:.2f}'.format(df[(df['label'] =='mungbean')][conditions].mean()) )
    print('grapes :{0:.2f}'.format(df[(df['label'] =='grapes')][conditions].mean()) )
    print('apple :{0:.2f}'.format(df[(df['label'] =='apple')][conditions].mean()) )
    print('watermelon:{0:.2f}'.format(df[(df['label'] =='watermelon')][conditions].mean()) )
    print('papaya :{0:.2f}'.format(df[(df['label'] =='papaya')][conditions].mean()) )
    print('muskmelon :{0:.2f}'.format(df[(df['label'] =='muskmelon')][conditions].mean()) )
    print('jute :{0:.2f}'.format(df[(df['label'] =='jute')][conditions].mean()) )
    print('mango :{0:.2f}'.format(df[(df['label'] =='mango')][conditions].mean()) )
    print('blackgram :{0:.2f}'.format(df[(df['label'] =='blackgram')][conditions].mean()) )
    print('coconut :{0:.2f}'.format(df[(df['label'] =='coconut')][conditions].mean()) )
    print('pomegranate :{0:.2f}'.format(df[(df['label'] =='pomegranate')][conditions].mean()) )
    print('coffee :{0:.2f}'.format(df[(df['label'] =='coffee')][conditions].mean()) )
    print('pigeonpeas :{0:.2f}'.format(df[(df['label'] =='pigeonpeas')][conditions].mean()) )
    print('lentil :{0:.2f}'.format(df[(df['label'] =='lentil')][conditions].mean()) )
    print('orange :{0:.2f}'.format(df[(df['label'] =='orange')][conditions].mean()) )
    print('kidneybeans :{0:.2f}'.format(df[(df['label'] =='kidneybeans')][conditions].mean()) )
    print('banana :{0:.2f}'.format(df[(df['label'] =='banana')][conditions].mean()) )
    print('cotton :{0:.2f}'.format(df[(df['label'] =='cotton')][conditions].mean()) )
    print('mothbeans :{0:.2f}'.format(df[(df['label'] =='mothbeans')][conditions].mean()) )
    print('chickpea :{0:.2f}'.format(df[(df['label'] =='chickpea')][conditions].mean()) )
    print('maize :{0:.2f}'.format(df[(df['label'] =='maize')][conditions].mean()) )
```

In [76]:
```python
@interact
def compare(conditions = ['N','P','K','temperature','ph','humidity','rainfall']):
    print('crops which require greater than average',conditions,'\n')
    print(df[df[conditions] > df[conditions].mean()]['label'].unique())
    print('------------------------------------------')
    print('crops which require less than average',conditions,'\n')
    print(df[df[conditions] <= df[conditions].mean()]['label'].unique())
```

In [77]:
```python
plt.subplot(2,4,1)
sns.distplot(df['N'],color = 'darkblue')
plt.xlabel('ratio of nitrogen',fontsize =15)
plt.grid(True)

plt.subplot(2,4,2)
sns.distplot(df['K'],color = 'black')
plt.xlabel('ratio of potassium',fontsize =15)
plt.grid()

plt.subplot(2,4,3)
sns.distplot(df['P'],color = 'grey')
plt.xlabel('ratio of phosphorous',fontsize =15)
plt.grid()

plt.subplot(2,4,4)
sns.distplot(df['temperature'],color = 'green')
plt.xlabel('temperature',fontsize =15)
plt.grid()

plt.subplot(2,4,5)
sns.distplot(df['ph'],color = 'darkgreen')
plt.xlabel('ph',fontsize =15)
plt.grid()

plt.subplot(2,4,6)
sns.distplot(df['rainfall'],color = 'lightgreen')
plt.xlabel('rainfall',fontsize =15)
plt.grid()

plt.subplot(2,4,7)
sns.distplot(df['humidity'],color = 'orange')
plt.xlabel('humidity',fontsize =15)
plt.grid()



plt.suptitle('Distribution for agricultural condition',fontsize =20)
plt.show()
```

```
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\prash\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [78]:
```python
print('some interesting facts')
print('-----------------------------------------------------')

print('soil which require very high ratio of nitrogen content in soil:',df[df['N'] > 120]['label'].unique())
print('soil which require very high ratio of phosphorous content in soil:',df[df['P'] > 100]['label'].unique())
print('soil which require very high ratio of potassium content in soil:',df[df['K'] > 200]['label'].unique())
print('crops which require very rainfall:',df[df['rainfall'] > 200]['label'].unique())
print('crops which require low temperature:',df[df['temperature'] < 10]['label'].unique())
print('crops which require high temperature:',df[df['temperature'] > 40]['label'].unique())
print('crops which require low humidity:',df[df['humidity'] <20]['label'].unique())
print('crops which require low ph:',df[df['ph'] < 4]['label'].unique())
print('crops which require high ph:',df[df['ph'] > 9]['label'].unique())
```

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

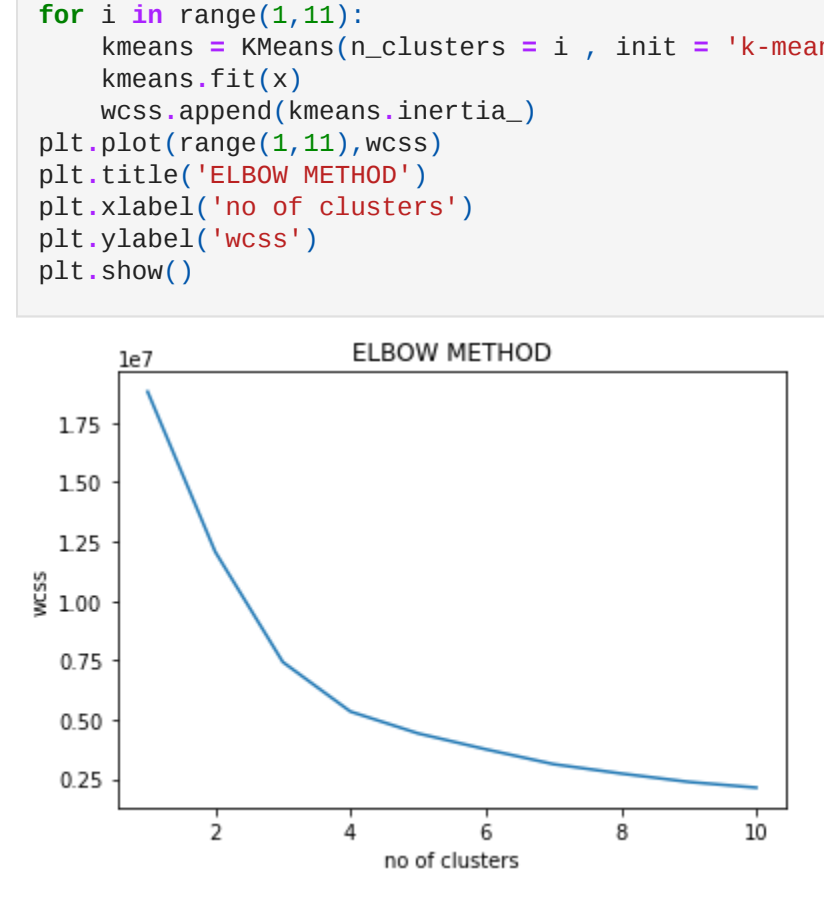
In [79]:
```python
print('summer crops')
print(df[(df['temperature'] > 30) & (df['humidity'] > 50)]['label'].unique())
print('winter crops')
print(df[(df['temperature'] < 20) & (df['humidity'] > 30)]['label'].unique())
print('rainy crops')
print(df[(df['rainfall'] > 200) & (df['humidity'] > 30)]['label'].unique())
```

```
summer crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
winter crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
rainy crops
['rice' 'papaya' 'coconut']
```

In [80]:
```python
from sklearn.cluster import KMeans
x = df.drop(['label'],axis =1)
x = x.values
print(x.shape)
```

```
(2200, 7)
```

In [81]:
```python
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i , init = 'k-means++',random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('ELBOW METHOD')
plt.xlabel('no of clusters')
plt.ylabel('wcss')
plt.show()
```



In [82]:
```python
kmeans = KMeans(n_clusters =  4 , init = 'k-means++',random_state = 0)
y_kmean   = kmeans.fit_predict(x)
```

In [83]:
```python
a = df['label']
y_kmean = pd.DataFrame(y_kmean)
z = pd.concat([y_kmean ,a],axis =1)
```

In [84]:
```python
z = z.rename(columns ={0:'cluster'})
```

In [85]:
```python
z.head(525)
```

Out[85]:

	cluster	label
0	3	rice
1	3	rice
2	3	rice
3	3	rice
4	3	rice
...	...	...
520	0	mothbeans
521	0	mothbeans
522	0	mothbeans
523	0	mothbeans
524	0	mothbeans

525 rows × 2 columns

In [86]:
```python
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())
```

```
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 [87]:
```python
# lets split the data set for predictive modelling
y = df['label']
x = df.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 [88]:
```python
#splitting the dataset
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)
```

In [89]:
```python
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 [90]:
```python
from sklearn.linear_model import LogisticRegression
classifier =  LogisticRegression(random_state = 1)
classifier.fit(x_train,y_train)
```

```
C:\Users\prash\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(
```

Out[90]:
```
LogisticRegression(random_state=1)
```

In [91]:
```python
y_pred = classifier.predict(x_test)
```

In [94]:
```python
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[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 18  0  0  0  0  0  0  0  1  1  0  2  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  1  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  2  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 21  0  0  0  0  0  0  0  0  0]
 [ 0  0  2  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 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  1  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  4  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 17]]
```

Out[94]:
```
0.9681818181818181
```

In [102…
```python
# predict at a given soil and climatic condition which crop we can grow

prediction = 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: ['rice']
```

In [ ]: