In [6]:
```
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
from ipywidgets import interact
```

In [7]:
```
data = pd.read_csv('crop_recommendation.csv')
```

In [8]:
```
print("Shape of the Dataset :", data.shape)
```

```
Shape of the Dataset : (2200, 10)
```

In [9]:
```
data.head()
```

Out[9]:

	Nitrogen	Phosphorus	Pottasium	Temp	Moisture	ph	Rainfall	Crop	So
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	Ye
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice	Ye
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 [10]:
```
data.isnull().sum()
```

Out[10]:
```
Nitrogen      0
Phosphorus    0
Pottasium     0
Temp          0
Moisture      0
ph            0
Rainfall      0
Crop          0
Soilcolor     0
Precip_Sum    0
dtype: int64
```

In [11]:
```
data['Crop'].value_counts()
```

Out[11]:
```
Crop
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: count, dtype: int64
```

In [12]:
```
print("Average Ratio of Nitrogen in the Soil : {0:.2f}".format(data['Nitrogen'].mea
print("Average Ratio of Phosphorous in the Soil : {0:.2f}".format(data['Phosphorus'
print("Average Ratio of Potassium in the Soil : {0:.2f}".format(data['Pottasium'].m
print("Average Rainfall in mm : {0:.2f}".format(data['Rainfall'].mean()))
print("Average Tempature in Celsius : {0:.2f}".format(data['Temp'].mean()))
print("Average Relative Humidity in % : {0:.2f}".format(data['Moisture'].mean()))
print("Average PH Value of the soil : {0:.2f}".format(data['ph'].mean()))
print("Average precipitation Sum  : {0:.2f}".format(data['Precip_Sum'].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 Rainfall in mm : 103.46
Average Tempature in Celsius : 25.62
Average Relative Humidity in % : 71.48
Average PH Value of the soil : 6.47
Average precipitation Sum  : 78.77
```

In [13]:
```
def summary(crops = list(data['Crop'].value_counts().index)):
    x = data[data['Crop'] == crops]
    print("\n-----------------------------------------")
    print("Statistics for Nitrogen")
    print("Minimum Nitrogen required :", x['Nitrogen'].min())
    print("Average Nitrogen required :", x['Nitrogen'].mean())
    print("Maximum Nitrogen required :", x['Nitrogen'].max())
    print("\n-----------------------------------------")
    print("\nStatistics for Phosphorous")
    print("Minimum Phosphorous required :", x['Phosphorus'].min())
    print("Average Phosphorous required :", x['Phosphorus'].mean())
    print("Maximum Phosphorous required :", x['Phosphorus'].max())
    print("\n-----------------------------------------")
```

```
    print("\nStatistics for Potassium")
    print("Minimum Potassium required :", x['Pottasium'].min())
    print("Average Potassium required :", x['Pottasium'].mean())
    print("Maximum Potassium required :", x['Pottasium'].max())
    print("\n-----------------------------------------------")
    print("\nStatistics for Temperature")
    print("Minimum Temperature required : {0:.2f}".format(x['Temp'].min()))
    print("Average Temperature required : {0:.2f}".format(x['Temp'].mean()))
    print("Maximum Temperature required : {0:.2f}".format(x['Temp'].max()))
    print("\n-----------------------------------------------")
    print("\nStatistics for Humidity")
    print("Minimum Humidity required : {0:.2f}".format(x['Moisture'].min()))
    print("Average Humidity required : {0:.2f}".format(x['Moisture'].mean()))
    print("Maximum Humidity required : {0:.2f}".format(x['Moisture'].max()))
    print("\n-----------------------------------------------")
    print("\nStatistics 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("\n-----------------------------------------------")
    print("\nStatistics 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}\n".format(x['Rainfall'].max()))
```

In [14]:
```
print('Crops : ',data.Crop.unique())
fcrop = input("\nName the Crop from the Given options : ")
summary(fcrop)
```

```
Crops :  ['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
```

```
----------------------------------------------
Statistics for Nitrogen
Minimum Nitrogen required : 60
Average Nitrogen required : 77.76
Maximum Nitrogen required : 100

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

Statistics for Phosphorous
Minimum Phosphorous required : 35
Average Phosphorous required : 48.44
Maximum Phosphorous required : 60

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

Statistics for Potassium
Minimum Potassium required : 15
Average Potassium required : 19.79
Maximum Potassium required : 25

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

Statistics for Temperature
Minimum Temperature required : 18.04
Average Temperature required : 22.39
Maximum Temperature required : 26.55

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

Statistics for Humidity
Minimum Humidity required : 55.28
Average Humidity required : 65.09
Maximum Humidity required : 74.83

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

Statistics for PH
Minimum PH required : 5.51
Average PH required : 6.25
Maximum PH required : 7.00

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

Statistics for Rainfall
Minimum Rainfall required : 60.65
Average Rainfall required : 84.77
Maximum Rainfall required : 109.75
```

In [17]:
```
def compare(conditions = ['Nitrogen','Phosphorus','Pottasium','Temp','ph','Moisture
    print("\nAverage Value for", conditions,"is {0:.2f}".format(data[conditions].me
    print("\n----------------------------------------------")
    print("\nApple : {0:.2f}".format(data[data['Crop'] == 'apple'][conditions].mean
    print("Banana : {0:.2f}".format(data[(data['Crop'] == 'banana')][conditions].me
    print("Black Grams : {0:.2f}".format(data[data['Crop'] == 'blackgram'][conditio
    print("Coconut : {0:.2f}".format(data[(data['Crop'] == 'coconut')][conditions].
```

```
    print("Coffee : {0:.2f}".format(data[data['Crop'] == 'coffee'][conditions].mean
    print("Cotton : {0:.2f}".format(data[data['Crop'] == 'cotton'][conditions].mean
    print("Chick Peas : {0:.2f}".format(data[data['Crop'] == 'chickpea'][conditions
    print("Grapes : {0:.2f}".format(data[(data['Crop'] == 'grapes')][conditions].me
    print("Jute : {0:.2f}".format(data[data['Crop'] == 'jute'][conditions].mean()))
    print("Kidney Beans: {0:.2f}".format(data[(data['Crop'] == 'kidneybeans')][cond
    print("Lentils : {0:.2f}".format(data[(data['Crop'] == 'lentil')][conditions].m
    print("Maize : {0:.2f}".format(data[(data['Crop'] == 'maize')][conditions].mean
    print("Mango : {0:.2f}".format(data[data['Crop'] == 'mango'][conditions].mean()
    print("Papaya : {0:.2f}".format(data[(data['Crop'] == 'papaya')][conditions].me
    print("Muskmelon : {0:.2f}".format(data[data['Crop'] == 'muskmelon'][conditions
    print("Moth Beans : {0:.2f}".format(data[data['Crop'] == 'mothbeans'][condition
    print("Mung Beans : {0:.2f}".format(data[data['Crop'] == 'mungbean'][conditions
    print("Oranges : {0:.2f}".format(data[(data['Crop'] == 'orange')][conditions].m
    print("Pigeon Peas : {0:.2f}".format(data[(data['Crop'] == 'pigeonpeas')][condi
    print("Pomegranate : {0:.2f}".format(data[(data['Crop'] == 'pomegranate')][cond
    print("Rice : {0:.2f}".format(data[(data['Crop'] == 'rice')][conditions].mean()
    print("Watermelon : {0:.2f}".format(data[data['Crop'] == 'watermelon\n'][condit
```

In [18]:
```
print('Conditions:',['Nitrogen', 'Phosphorus', 'Pottasium', 'Temp', 'Moisture', 'ph
comparecondi = input("Enter the Condition require from above options: ")
compare(comparecondi)
```

```
Conditions: ['Nitrogen', 'Phosphorus', 'Pottasium', 'Temp', 'Moisture', 'ph', 'Rainf
all', 'Crop']
Average Value for Temp is 25.62

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

Apple : 22.63
Banana : 27.38
Black Grams : 29.97
Coconut : 27.41
Coffee : 25.54
Cotton : 23.99
Chick Peas : 18.87
Grapes : 23.85
Jute : 24.96
Kidney Beans: 20.12
Lentils : 24.51
Maize : 22.39
Mango : 31.21
Papaya : 33.72
Muskmelon : 28.66
Moth Beans : 28.19
Mung Beans : 28.53
Oranges : 22.77
Pigeon Peas : 27.74
Pomegranate : 21.84
Rice : 23.69
Watermelon : nan
```

In [19]:
```
def compare(conditions = ['Nitrogen','P','K','Temp','ph','Moisture','Rainfall']):
    print("\nCrops which require greater than average", conditions,'\n')
    print(data[data[conditions] > data[conditions].mean()]['Crop'].unique())
    print("\n----------------------------------------------")
```

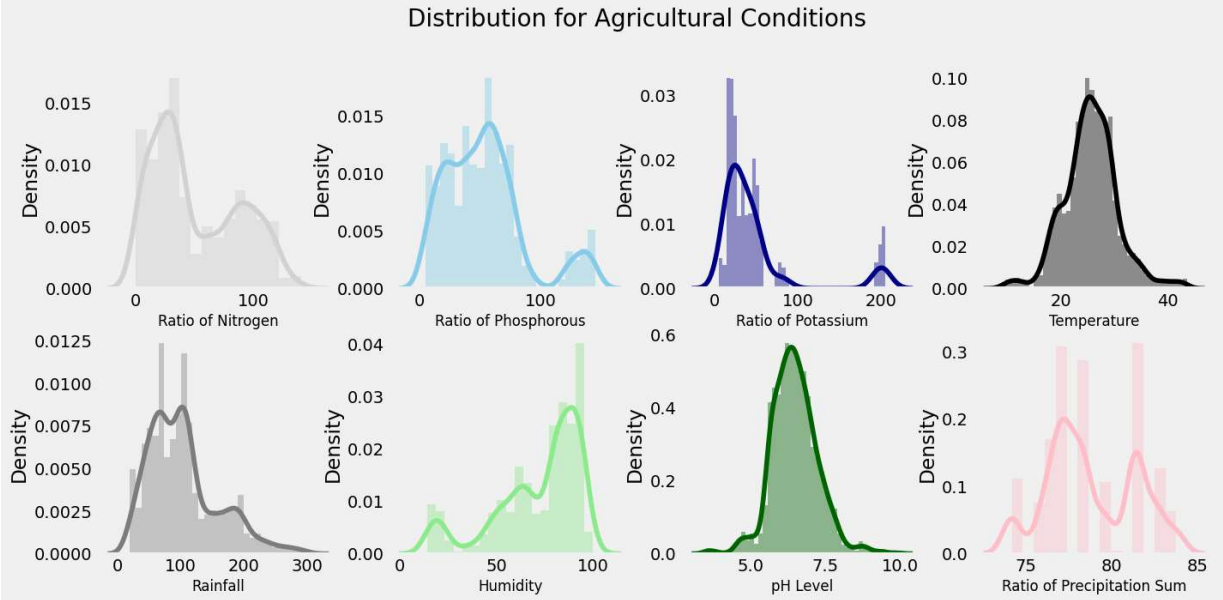
```
    print("\nCrops which require less than average", conditions,'\n')
    print(data[data[conditions] <= data[conditions].mean()]['Crop'].unique())
```

In [20]:
```
print('COnditions:',['Nitrogen', 'Phosphorus', 'Pottasium', 'Temp', 'Moisture', 'ph
rancondi = input("Enter the Feature require from above options: ")
compare(rancondi)
```

```
COnditions: ['Nitrogen', 'Phosphorus', 'Pottasium', 'Temp', 'Moisture', 'ph', 'Rainf
all', 'Crop']
Crops which require greater than average Nitrogen

['rice' 'maize' 'chickpea' 'blackgram' 'banana' 'watermelon' 'muskmelon'
 'papaya' 'cotton' 'jute' 'coffee']

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

Crops which require less than average Nitrogen

['chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram'
 'lentil' 'pomegranate' 'mango' 'grapes' 'apple' 'orange' 'papaya'
 'coconut']
```

In [21]:
```
### Lets check the distribution of Agricultural Conditions

import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (15, 7)

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

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

plt.subplot(2, 4, 3)
sns.distplot(data['Pottasium'], color ='darkblue')
plt.xlabel('Ratio of Potassium', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 4)
sns.distplot(data['Temp'], 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['Moisture'], 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.subplot(2, 4, 8)
sns.distplot(data['Precip_Sum'], color = 'pink')
plt.xlabel('Ratio of Precipitation Sum', fontsize = 12)
plt.grid()


plt.suptitle('Distribution for Agricultural Conditions', fontsize = 20)
plt.show()
```



In [22]:
```
## Lets find out some Interesting Facts

print("Some Interesting Patterns")
print("-------------------------------")
print("Crops which requires very High Ratio of Nitrogen Content in Soil:", data[dat
print("Crops which requires very High Ratio of Phosphorous Content in Soil:", data[
print("Crops which requires very High Ratio of Potassium Content in Soil:", data[da
print("Crops which requires very High Rainfall:", data[data['Rainfall'] > 200]['Cro
print("Crops which requires very Low Temperature :", data[data['Temp'] < 10]['Crop'
print("Crops which requires very High Temperature :", data[data['Temp'] > 40]['Crop
print("Crops which requires very Low Humidity:", data[data['Moisture'] < 20]['Crop'
print("Crops which requires very Low pH:", data[data['ph'] < 4]['Crop'].unique())
print("Crops which requires very High pH:", data[data['ph'] > 9]['Crop'].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' 'appl
e']
Crops which requires very High Ratio of Potassium 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 Low pH: ['mothbeans']
Crops which requires very High pH: ['mothbeans']
```

In [23]:
```
### Lets understand which crops can only be Grown in Summer Season, Winter Season a

print("Summer Crops")
print(data[(data['Temp'] > 30) & (data['Moisture'] > 50)]['Crop'].unique())
print("-----------------------------------")
print("Winter Crops")
print(data[(data['Temp'] < 20) & (data['Moisture'] > 30)]['Crop'].unique())
print("-----------------------------------")
print("Rainy Crops")
print(data[(data['Rainfall'] > 200) & (data['Moisture'] > 30)]['Crop'].unique())
```

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

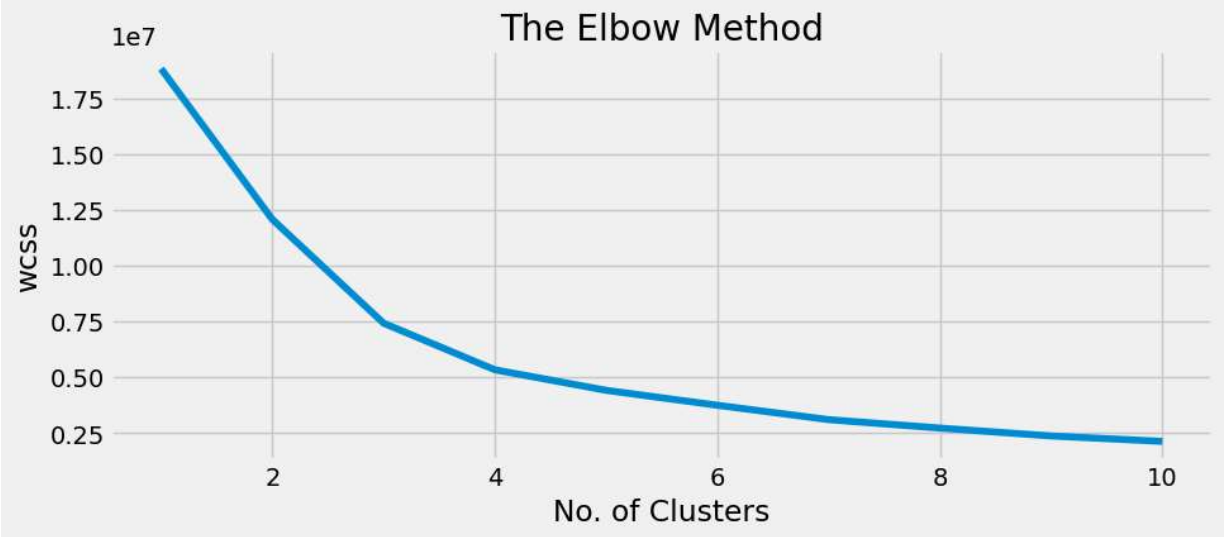
In [24]:
```
### Lets try to Cluster these Crops

# lets import the warnings library so that we can avoid warnings
import warnings
warnings.filterwarnings('ignore')

# Lets select the Spending score, and Annual Income Columns from the Data
x = data.loc[:, ['Nitrogen','Phosphorus','Pottasium','Temp','ph','Moisture','Rainfa

# 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)
```

Out[24]:

	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

In [25]:
```
# lets determine the Optimum Number of Clusters within the Dataset

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, ra
    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 [26]:
```
# 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
y_means = km.fit_predict(x)

# lets find out the Results
a = data['Crop']
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]['Crop'].unique())
print("-----------------------------------------------------------")
print("Crops in Second Cluster:", z[z['cluster'] == 1]['Crop'].unique())
print("-----------------------------------------------------------")
print("Crops in Third Cluster:", z[z['cluster'] == 2]['Crop'].unique())
print("-----------------------------------------------------------")
print("Crops in Forth Cluster:", z[z['cluster'] == 3]['Crop'].unique())
```

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

Crops in First Cluster: ['grapes' 'apple']
-----------------------------------------------------------
Crops in Second Cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
-----------------------------------------------------------
Crops in Third Cluster: ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya' 'cotton'
'coffee']
-----------------------------------------------------------
Crops in Forth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
```

In [27]:

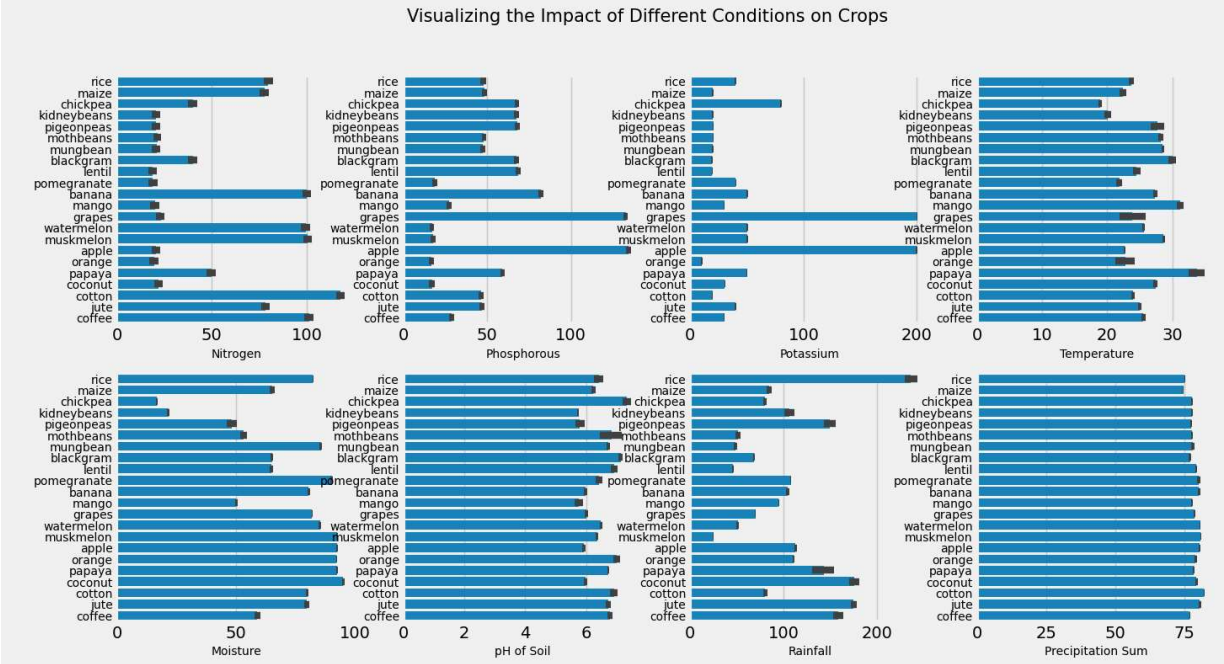
```
# Hard Clustering

print("Results for Hard Clustering\n")
counts = z[z['cluster'] == 0]['Crop'].value_counts()
d = z.loc[z['Crop'].isin(counts.index[counts >= 50])]
d = d['Crop'].value_counts()
print("Crops in Cluster 1:", list(d.index))
print("-------------------------------------------------")
counts = z[z['cluster'] == 1]['Crop'].value_counts()
d = z.loc[z['Crop'].isin(counts.index[counts >= 50])]
d = d['Crop'].value_counts()
print("Crops in Cluster 2:", list(d.index))
print("-------------------------------------------------")
counts = z[z['cluster'] == 2]['Crop'].value_counts()
d = z.loc[z['Crop'].isin(counts.index[counts >= 50])]
d = d['Crop'].value_counts()
print("Crops in Cluster 3:", list(d.index))
print("-------------------------------------------------")
counts = z[z['cluster'] == 3]['Crop'].value_counts()
d = z.loc[z['Crop'].isin(counts.index[counts >= 50])]
d = d['Crop'].value_counts()
print("Crops in Cluster 4:", list(d.index))
```

```
Results for Hard Clustering

Crops in Cluster 1: ['grapes', 'apple']
--------------------------------------------------
Crops in Cluster 2: ['chickpea', 'kidneybeans', 'mothbeans', 'mungbean', 'blackgra
m', 'lentil', 'pomegranate', 'mango', 'orange']
--------------------------------------------------
Crops in Cluster 3: ['maize', 'banana', 'watermelon', 'muskmelon', 'cotton']
--------------------------------------------------
Crops in Cluster 4: ['rice', 'pigeonpeas', 'papaya', 'coconut', 'jute', 'coffee']
```

In [29]:

```
### Data Visualizations

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

plt.subplot(2, 4, 1)
sns.barplot(x='Nitrogen', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Nitrogen', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 2)
sns.barplot(x='Phosphorus', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Phosphorous', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 3)
sns.barplot(x='Pottasium', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Potassium', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 4)
sns.barplot(x='Temp', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Temperature', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 5)
sns.barplot(x='Moisture', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Moisture', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 6)
sns.barplot(x='ph', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('pH of Soil', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 7)
sns.barplot(x='Rainfall', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Rainfall', fontsize = 10)
plt.yticks(fontsize = 10)
```

```
plt.subplot(2, 4, 8)
sns.barplot(x='Precip_Sum', y='Crop', data=data)
plt.ylabel(' ')
plt.xlabel('Precipitation Sum', fontsize = 10)
plt.yticks(fontsize = 10)

plt.suptitle('Visualizing the Impact of Different Conditions on Crops', fontsize =
plt.show()
```



In [36]:
```
# lets split the Dataset for Predictive Modelling

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

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

```
Shape of x: (2200, 9)
Shape of y: (2200,)
```

In [37]:
```
# 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_s

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, 9)
The Shape of x test: (440, 9)
The Shape of y train: (1760,)
The Shape of y test: (440,)
```

In [32]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

data['Soilcolor']=le.fit_transform(data['Soilcolor'])
data.head()
```

Out[32]:

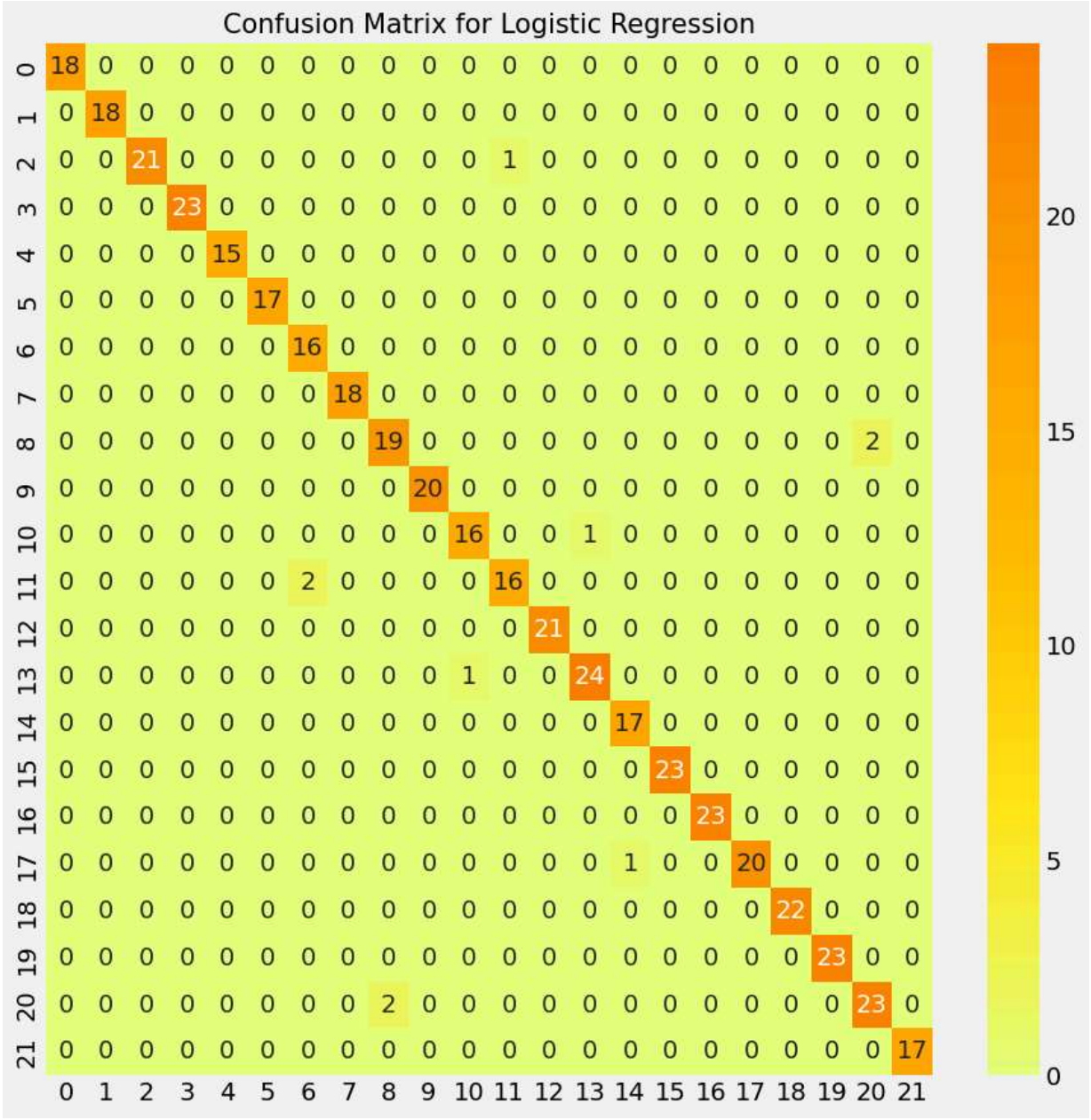
	Nitrogen	Phosphorus	Pottasium	Temp	Moisture	ph	Rainfall	Crop	So
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice	
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice	
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice	
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice	

In [38]:
```
# lets create a Predictive Model

from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression()
log_model.fit(x_train, y_train)
y_pred_log = log_model.predict(x_test)
```

In [39]:
```
score = log_model.score(x_test,y_test)
print("Accuracy for Logistic Regression is "+str(score))
```

```
Accuracy for Logistic Regression is 0.9772727272727273
```

In [40]:
```
# lets evaluate the Model Performance
from sklearn.metrics import classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')


# lets print the Confusion matrix first
plt.rcParams['figure.figsize'] = (10, 10)
cm = confusion_matrix(y_test, y_pred_log)
sns.heatmap(cm, annot = True, cmap = 'Wistia')
plt.title('Confusion Matrix for Logistic Regression', fontsize = 15)
plt.show()

# lets print the Classification Report also
cr = classification_report(y_test, y_pred_log)
print(cr)
```


Confusion Matrix for Logistic Regression
0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 0 0 21 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
3 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 2 0
9 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0 16 0 0 1 0 0 0 0 0 0 0 0
11 0 0 0 0 0 0 2 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0
12 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0
13 0 0 0 0 0 0 0 0 0 0 1 0 0 24 0 0 0 0 0 0 0 0
14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0
15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0
16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0
17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 20 0 0 0 0
18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0
19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0
20 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 23 0
21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
20
15
10
5
0

	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	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	0.90	0.90	0.90	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	0.96	0.96	0.96	25
mungbean	0.94	1.00	0.97	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	0.92	0.92	0.92	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

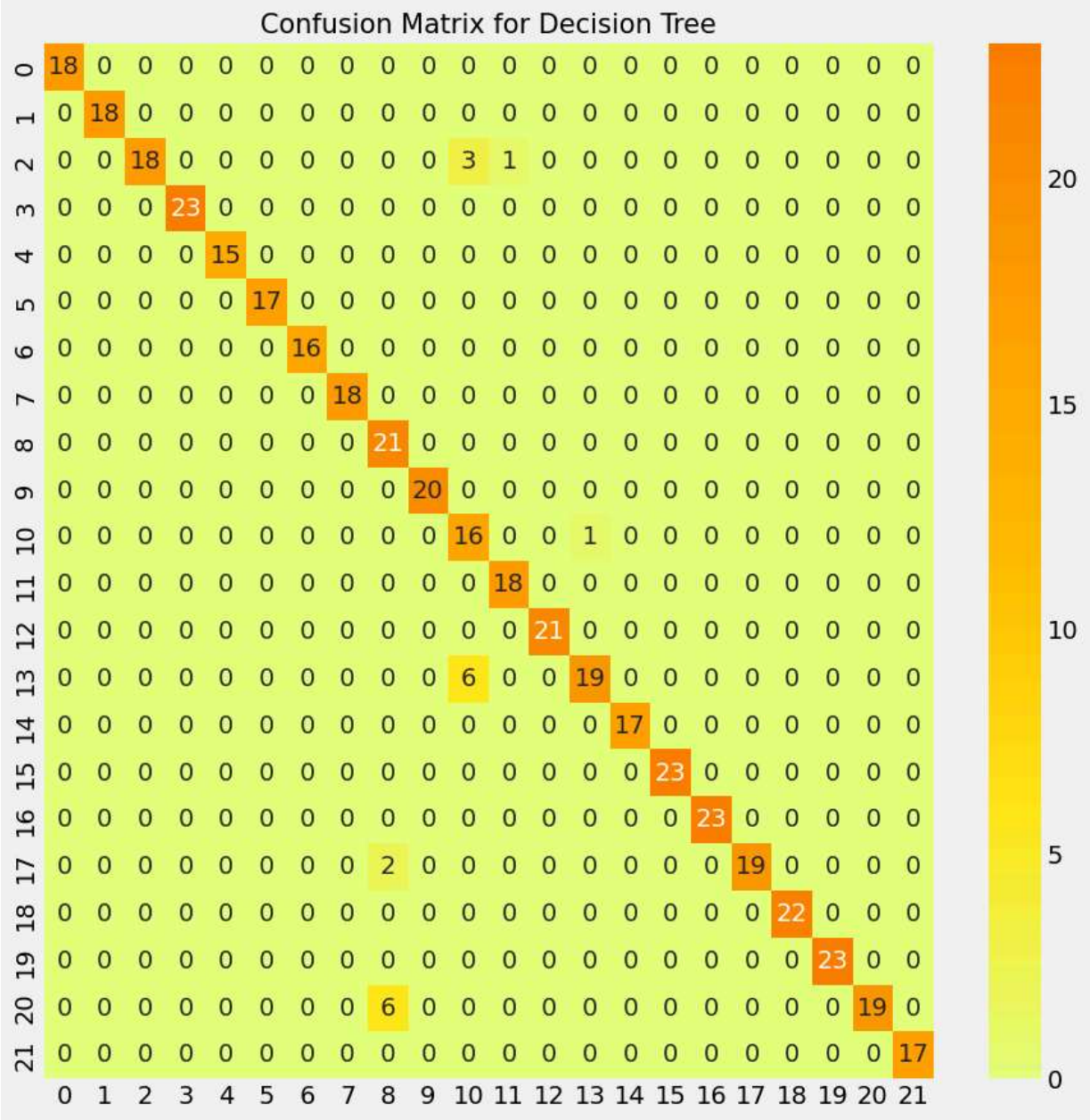
In [41]:
```
from sklearn.tree import DecisionTreeClassifier

dec_tree = DecisionTreeClassifier(criterion="entropy",splitter="best",max_features
dec_tree.fit(x_train,y_train)
y_pred_dec = dec_tree.predict(x_test)
```

In [42]:
```
core = dec_tree.score(x_test,y_test)
print("Accuracy for Decision Tree is "+str(score))
```

```
Accuracy for Decision Tree is 0.9772727272727273
```

In [43]:
```
# lets evaluate the Model Performance
from sklearn.metrics import classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')

# lets print the Confusion matrix first
plt.rcParams['figure.figsize'] = (10, 10)
cm = confusion_matrix(y_test, y_pred_dec)
sns.heatmap(cm, annot = True, cmap = 'Wistia')
plt.title('Confusion Matrix for Decision Tree', fontsize = 15)
plt.show()

# lets print the Classification Report also
```

```
cr = classification_report(y_test, y_pred_dec)
print(cr)
```



	precision	recall	f1-score	support
apple	1.00	1.00	1.00	18
banana	1.00	1.00	1.00	18
blackgram	1.00	0.82	0.90	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	0.72	1.00	0.84	21
kidneybeans	1.00	1.00	1.00	20
lentil	0.64	0.94	0.76	17
maize	0.95	1.00	0.97	18
mango	1.00	1.00	1.00	21
mothbeans	0.95	0.76	0.84	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	0.90	0.95	21
pigeonpeas	1.00	1.00	1.00	22
pomegranate	1.00	1.00	1.00	23
rice	1.00	0.76	0.86	25
watermelon	1.00	1.00	1.00	17
accuracy			0.96	440
macro avg	0.97	0.96	0.96	440
weighted avg	0.97	0.96	0.96	440

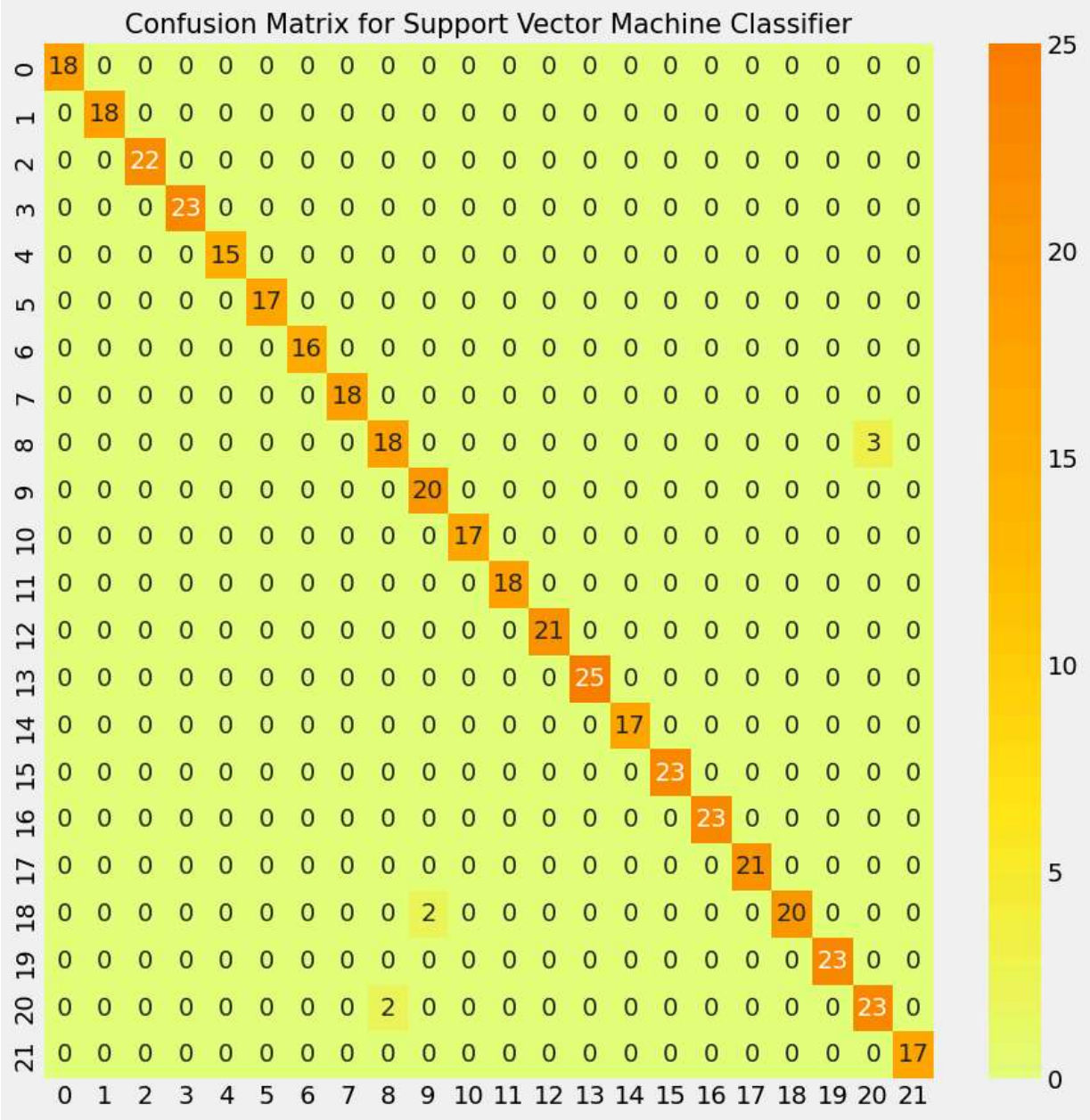
In [44]:
```
from sklearn.svm import SVC

svc = SVC(kernel='poly')
svc.fit(x_train,y_train)
y_pred_svc=svc.predict(x_test)
```

In [45]:
```
score = svc.score(x_test,y_test)
print("Accuracy for Support Vector Classifier is "+str(score))
```

```
Accuracy for Support Vector Classifier is 0.9840909090909091
```

In [46]:
```
# lets evaluate the Model Performance
from sklearn.metrics import classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')

# lets print the Confusion matrix first
plt.rcParams['figure.figsize'] = (10, 10)
cm = confusion_matrix(y_test, y_pred_svc)
sns.heatmap(cm, annot = True, cmap = 'Wistia')
plt.title('Confusion Matrix for Support Vector Machine Classifier', fontsize = 15)
plt.show()

# lets print the Classification Report also
```

```
cr = classification_report(y_test, y_pred_svc)
print(cr)
```



	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	0.90	0.86	0.88	21
kidneybeans	0.91	1.00	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	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	0.88	0.92	0.90	25
watermelon	1.00	1.00	1.00	17
accuracy			0.98	440
macro avg	0.99	0.99	0.99	440
weighted avg	0.98	0.98	0.98	440

In [47]:
```
n_real = input('Enter Nitrogen :')
p_real = input('Enter Phosphorus:')
k_real = input('Enter Pottasium :')
temp_real = input('Enter Temperature :')
hum_real = input('Enter Humidity :')
ph_real = input('Enter Ph :')
Rainfall_real = input('Enter Rainfall :')
color=input('Enter the soil color :')
pre=input('Enter the precipitation Sum :')
```

In [48]:
```
prediction = dec_tree.predict((np.array([[n_real,p_real,k_real,temp_real,hum_real,p
print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

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