In [9]:
```
# For Manipulations
import pandas as pd
import numpy as np

# For Data Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# For interactivity
from ipywidgets import interact

import warnings
warnings.filterwarnings("ignore")
import os
```

In [11]:
```
os.chdir(r'C:\UoG\Curriculum')
```

In [12]:
```
data = pd.read_csv('agri_data.csv')
```

In [13]:
```
data.head()
```

Out[13]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	Tomato
1	85	58	41	21.770462	80.319644	7.038096	226.655537	Tomato
2	60	55	44	23.004459	82.320763	7.840207	263.964248	Tomato
3	74	35	40	26.491096	80.158363	6.980401	242.864034	Tomato
4	78	42	42	20.130175	81.604873	7.628473	262.717340	Tomato

In [14]:
```
# Lets check the shape of the dataset
print("Shape of the Dataset :",data.shape)
```

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

In [15]:
```
# Lets check the missing value present in the dataset
data.isnull().sum()
```

Out[15]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [16]:
```
# Lets check the crops present in the dataset
data['label'].value_counts()
```

Out[16]:
```
label
Lettuce            200
Tomato             100
Swiss Chard        100
Broccoli           100
Carrots            100
Endive             100
Cabbage            100
Green tomatoes     100
Watermelon         100
Grapes             100
Fennel             100
Cucumber           100
Raspberry          100
Celery             100
Zucchini           100
Mungbean           100
Green beans        100
Red peppers        100
Cherry tomatoes    100
Eggplant           100
Cauliflower        100
Name: count, dtype: int64
```

In [17]:
```
# 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 Phosphorus 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 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 :50.551818
Average Ratio of Phosphorus in the Soil :53.362727
Average Ratio of Potassium in the Soil :48.149091
Average Temperature in Celsius :25.616244
Average Relative Humidity in % :71.481779
Average pH value of the Soil :6.469480
Average Rainfall in mm :103.463655
```

In [18]:
```
# 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 Phosphorus")
    print("Minimum Phosphorus Required :",x['P'].min())
    print("Average Phosphorus Required :",x['P'].mean())
    print("Maximum Phosphorus 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()))
```

```
interactive(children=(Dropdown(description='crops', options=('Lettuce', 'Tomato', 'Swiss Chard', 'Broccoli', '…
```

In [19]:
```
# Lets Check 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("Lettuce : {0:.2f}".format(data[(data['label']=='Lettuce')][conditions].mean()))
    print("Tomato : {0:.2f}".format(data[(data['label']=='Tomato')][conditions].mean()))
    print("Broccoli : {0:.2f}".format(data[(data['label']=='Broccoli')][conditions].mean()))
    print("Swiss Chard : {0:.2f}".format(data[(data['label']=='Swiss Chard')][conditions].mean()))
    print("Carrots : {0:.2f}".format(data[(data['label']=='Carrots')][conditions].mean()))
    print("Endive : {0:.2f}".format(data[(data['label']=='Endive')][conditions].mean()))
    print("Cabbage : {0:.2f}".format(data[(data['label']=='Cabbage')][conditions].mean()))
    print("Green tomatoes : {0:.2f}".format(data[(data['label']=='Green tomatoes')][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("Fennel : {0:.2f}".format(data[(data['label']=='Fennel')][conditions].mean()))
    print("Mung Beans : {0:.2f}".format(data[(data['label']=='mungbean')][conditions].mean()))
    print("Cucumber : {0:.2f}".format(data[(data['label']=='Cucumber')][conditions].mean()))
    print("Raspberry : {0:.2f}".format(data[(data['label']=='Raspberry')][conditions].mean()))
    print("Celery : {0:.2f}".format(data[(data['label']=='Celery')][conditions].mean()))
    print("Zucchini : {0:.2f}".format(data[(data['label']=='Zucchini')][conditions].mean()))
    print("Maize : {0:.2f}".format(data[(data['label']=='maize')][conditions].mean()))
    print("Green beans : {0:.2f}".format(data[(data['label']=='Green beans')][conditions].mean()))
    print("Red peppers : {0:.2f}".format(data[(data['label']=='Red peppers')][conditions].mean()))
    print("Cherry tomatoes : {0:.2f}".format(data[(data['label']=='Cherry tomatoes')][conditions].mean()))
    print("Eggplant : {0:.2f}".format(data[(data['label']=='Eggplant')][conditions].mean()))
    print("Cauliflower : {0:.2f}".format(data[(data['label']=='Cauliflower')][conditions].mean()))
```

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

In [20]:
```
# Lets Make this function more Intutive

@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())
```

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

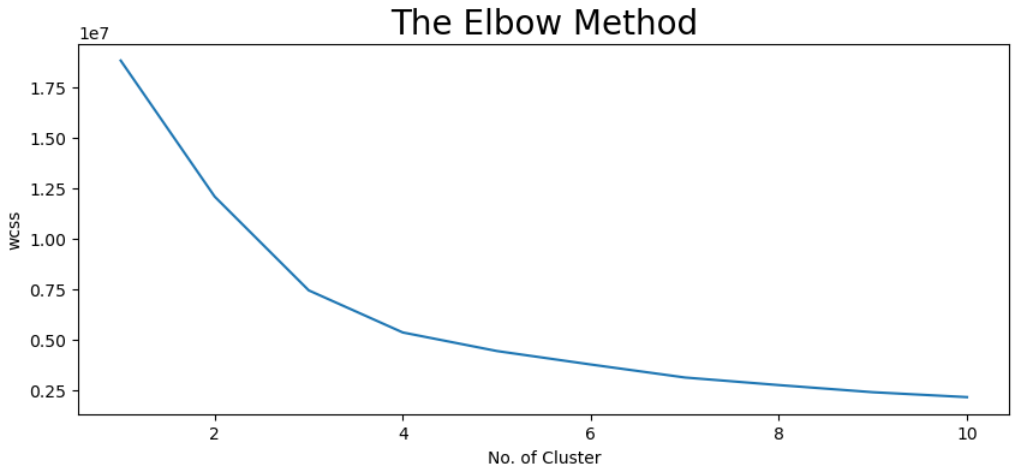
In [21]:
```
# Lets understand which crops can only be grown in Summer season, Winter season and rainy season

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
['Red peppers' 'Green beans' 'Zucchini' 'Swiss Chard' 'Grapes' 'Cabbage'
 'Endive']
----------------------------------------
Winter Crops
['Cucumber' 'Red peppers' 'Celery' 'Raspberry' 'Grapes' 'Cabbage']
----------------------------------------
Rainy Crops
['Tomato' 'Endive' 'Carrots']
```

In [22]:
```
# Clustering Analysis
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 [23]:
```
# Lets Determine the optimum number of Cluster 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 Cluster')
plt.ylabel("wcss")
plt.show()
```



In [24]:
```
# 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 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']==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 applying the K Means Clustering Analysis

Crops in First Cluster : ['Grapes' 'Green tomatoes']
--------------------------
Crops in Second Cluster : ['Cucumber' 'Eggplant' 'Cherry tomatoes' 'Red peppers' 'Green beans'
 'Mungbean' 'Zucchini' 'Celery' 'Raspberry' 'Swiss Chard' 'Cabbage'
 'Endive' 'Carrots']
--------------------------
Crops in Third Cluster : ['Cucumber' 'Fennel' 'Watermelon' 'Lettuce' 'Endive' 'Broccoli']
--------------------------
Crops in Fourth Cluster : ['Tomato' 'Red peppers' 'Endive' 'Carrots' 'Cauliflower' 'Lettuce']
```

In [25]:
```
# Lets split the Dataset for Predictive Modelling

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

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

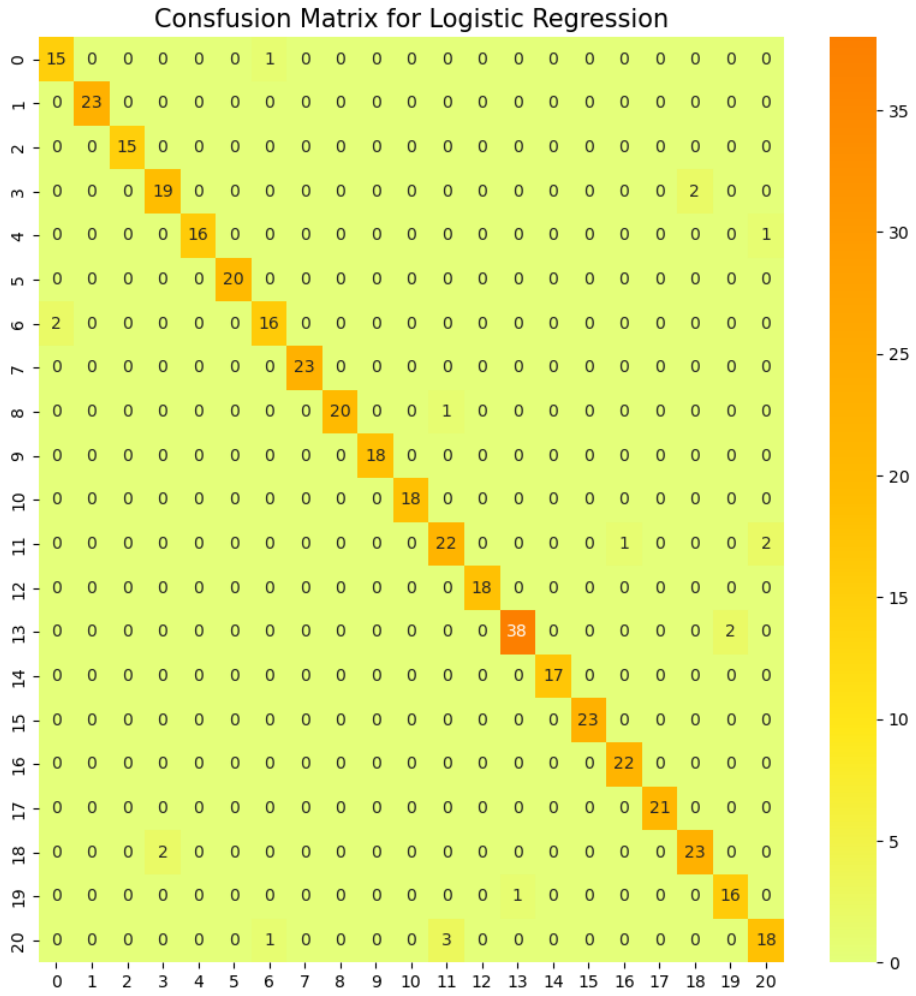
```
Shape of x: (2200, 7)
Shape of y: (2200,)
```

In [26]:
```
# train_test_split method

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 x_train : ",x_train.shape)
print("Shape of x_test : ",x_test.shape)
print("Shape of y_train : ",y_train.shape)
print("Shape of y_test : ",y_test.shape)
```

```
Shape of x_train :  (1760, 7)
Shape of x_test :  (440, 7)
Shape of y_train :  (1760,)
Shape of y_test :  (440,)
```

In [27]:
```
# Lets Create the 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('Consfusion Matrix for Logistic Regression',fontsize=15)
plt.show()
```



In [28]:
```
# Classification Report
from sklearn.metrics import classification_report

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

```
                precision    recall  f1-score   support

      Broccoli       0.88      0.94      0.91        16
       Cabbage       1.00      1.00      1.00        23
       Carrots       1.00      1.00      1.00        15
    Cauliflower      0.90      0.90      0.90        21
        Celery       1.00      0.94      0.97        17
Cherry tomatoes      1.00      1.00      1.00        20
      Cucumber       0.89      0.89      0.89        18
      Eggplant       1.00      1.00      1.00        23
        Endive       1.00      0.95      0.98        21
        Fennel       1.00      1.00      1.00        18
        Grapes       1.00      1.00      1.00        18
   Green beans       0.85      0.88      0.86        25
 Green tomatoes      1.00      1.00      1.00        18
       Lettuce       0.97      0.95      0.96        40
      Mungbean       1.00      1.00      1.00        17
     Raspberry       1.00      1.00      1.00        23
   Red peppers       0.96      1.00      0.98        22
   Swiss Chard       1.00      1.00      1.00        21
        Tomato       0.92      0.92      0.92        25
    Watermelon       0.89      0.94      0.91        17
      Zucchini       0.86      0.82      0.84        22

      accuracy                           0.96       440
     macro avg       0.96      0.96      0.96       440
  weighted avg       0.96      0.96      0.96       440
```

In [29]:
```
# Lets Check

data.head()
```

Out[29]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	Tomato
1	85	58	41	21.770462	80.319644	7.038096	226.655537	Tomato
2	60	55	44	23.004459	82.320763	7.840207	263.964248	Tomato
3	74	35	40	26.491096	80.158363	6.980401	242.864034	Tomato
4	78	42	42	20.130175	81.604873	7.628473	262.717340	Tomato

In [30]:
```
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 : ['Tomato']
```

In [ ]:
```

```