In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from ipywidgets import interact
```

In [2]:
```
data=pd.read_excel(r'agricultre.xlsx')
data
```

Out[2]:

	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 [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

In [4]:
```
data.describe()
```

Out[4]:

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

In [5]:
```
data.isnull().sum()
```

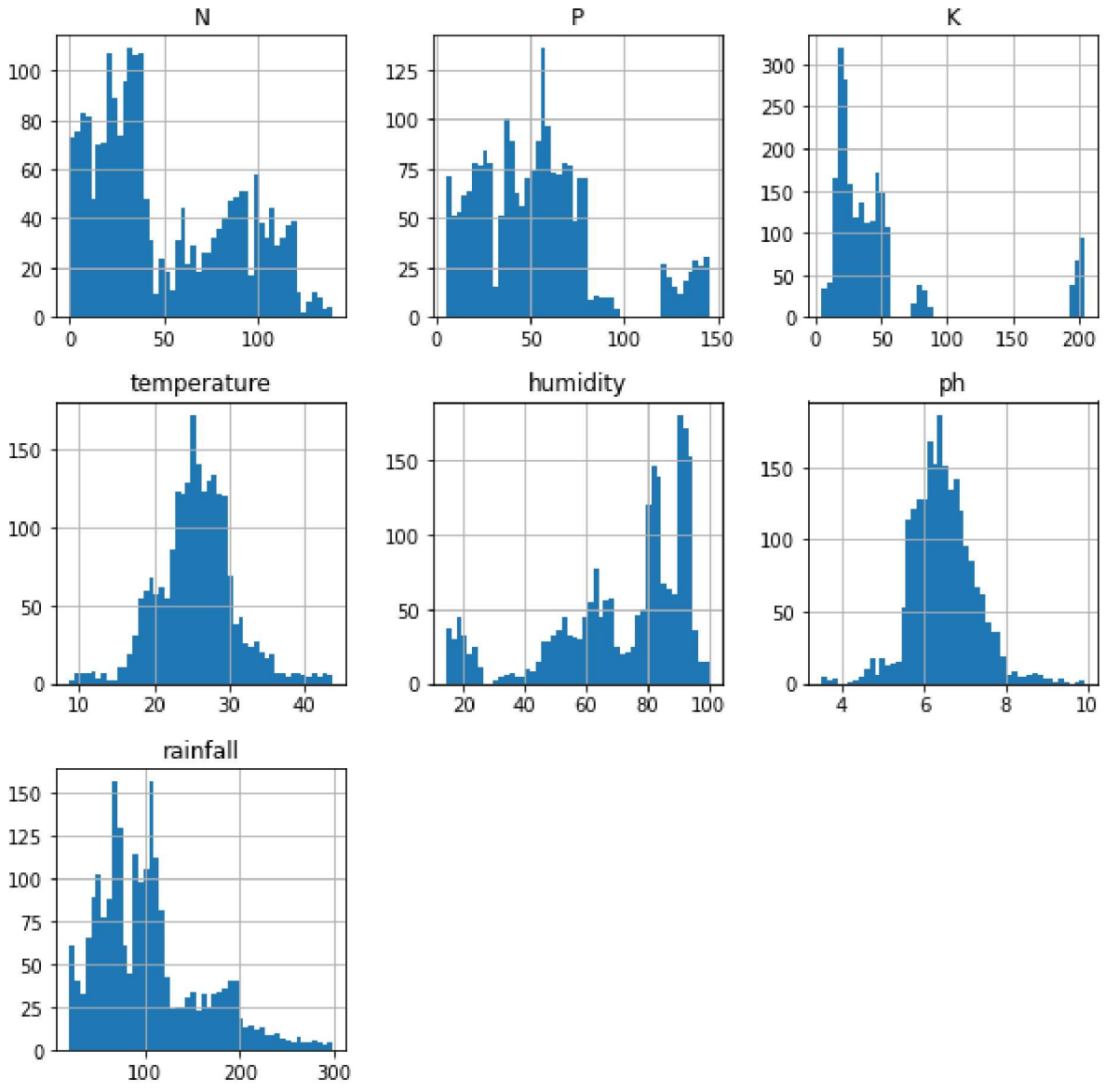
Out[5]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [6]:
```
data['label'].value_counts()
```

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

In [7]:
```
data.hist(bins=50, figsize=(10,10))
```

Out[7]:
```
array([[<AxesSubplot:title={'center':'N'}>,
        <AxesSubplot:title={'center':'P'}>,
        <AxesSubplot:title={'center':'K'}>],
       [<AxesSubplot:title={'center':'temperature'}>,
        <AxesSubplot:title={'center':'humidity'}>,
        <AxesSubplot:title={'center':'ph'}>],
       [<AxesSubplot:title={'center':'rainfall'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



AVERAGE RATIO OF THE FETAURES

In [8]:
```
print(data['N'].mean())
print(data['P'].mean())
print(data['K'].mean())
print(data['temperature'].mean())
print(data['humidity'].mean())
print(data['ph'].mean())
print(data['rainfall'].mean())
```

```
50.551818181818184
53.36272727272727
48.14909090909091
25.616243851779533
71.48177921778648
6.469480065256367
103.46365541576829
```

In [9]:
```
@interact
def summary(crops = list(data["label"].value_counts().index)):
    x = data[data["label"] == crops]
    for i in data.columns:
        print("---------------------------------------")
        print("Stastical data of : {}".format(data[i].name))
        print("Maximum {} required  ({})".format(data[i].name, x[i].max()))
        print("Minimum {} required : ({})".format(data[i].name, x[i].min()))
        print("Average {} required : ({})".format(data[i].name, x[i].mean()))
```

crops jute

```
---------------------------------------
Stastical data of : N
Maximum N required  (100)
Minimum N required : (60)
Average N required : (78.4)
---------------------------------------
Stastical data of : P
Maximum P required  (60)
Minimum P required : (35)
Average P required : (46.86)
---------------------------------------
Stastical data of : K
Maximum K required  (45)
Minimum K required : (35)
Average K required : (39.99)
---------------------------------------
Stastical data of : temperature
Maximum temperature required  (26.98582182)
Minimum temperature required : (23.09433785)
```

In [10]:
```
@interact
def compare(conditions=['N','P','K','temperature','ph','humidity','rainfall']):
    print("Average value for",conditions,"is {0:.2f}".format(data[conditions].mea
    print("**********************************************************************
    print("RICE is {0:.2f}".format(data[(data['label']=='rice')][conditions].mean
    print("BLACK GRAMS is {0:.2f}".format(data[(data['label']=='blackgram')][cond
    print("BANANAis {0:.2f}".format(data[(data['label']=='banana')][conditions].m
    print("COCUNUT is {0:.2f}".format(data[(data['label']=='jute')][conditions].m
    print("APPLE is {0:.2f}".format(data[(data['label']=='apple')][conditions].me
    print("PAPAYA is {0:.2f}".format(data[(data['label']=='papaya')][conditions].
    print("MUSKMELON is {0:.2f}".format(data[(data['label']=='muskmelon')][condit
    print("GRAPES is {0:.2f}".format(data[(data['label']=='grapes')][conditions].
    print("WATERMELON is {0:.2f}".format(data[(data['label']=='watermelon')][cond
    print("KIDNEY BEANS is {0:.2f}".format(data[(data['label']=='kidneybeans')][c
    print("MUNG BEANS is {0:.2f}".format(data[(data['label']=='mungbean')][condit
    print("ORANGES is {0:.2f}".format(data[(data['label']=='oranges')][conditions
    print("CHICKPEAS is {0:.2f}".format(data[(data['label']=='chickpea')][conditi
    print("LENTILS is {0:.2f}".format(data[(data['label']=='lentil')][conditions]
    print("COTTON is {0:.2f}".format(data[(data['label']=='cotton')][conditions].
    print("MAIZE is {0:.2f}".format(data[(data['label']=='maize')][conditions].me
    print("MOTH BEANS is {0:.2f}".format(data[(data['label']=='mothbeans')][condi
    print("PIGEAON PEAS is {0:.2f}".format(data[(data['label']=='pigeonpeas')][co
    print("MANGO is {0:.2f}".format(data[(data['label']=='mango')][conditions].me
    print("POMEGRANATE is {0:.2f}".format(data[(data['label']=='pomegranate')][co
    print("COFFEE is {0:.2f}".format(data[(data['label']=='coffee')][conditions].
```

conditions N

```
Average value for N is 50.55
**********************************************************************
RICE is 79.89
BLACK GRAMS is 40.02
BANANAis 100.23
COCUNUT is 78.40
APPLE is 20.80
PAPAYA is 49.88
MUSKMELON is 100.32
GRAPES is 23.18
WATERMELON is 99.42
KIDNEY BEANS is 20.75
MUNG BEANS is 20.99
ORANGES is nan
CHICKPEAS is 40.09
LENTILS is 18.77
COTTON is 117.77
MAIZE is 77.76
MOTH BEANS is 21.44
PIGEAON PEAS is 20.73
MANGO is 20.07
POMEGRANATE is 18.87
COFFEE is 101.20
```

In [11]:
```
@interact
def compare(conditions=['N','P','K','temperature','ph','humidity','rainfall']):
    print("CROPS WHICH REQUIRE GRAETER THAN AVERAGE", conditions,':')
    print(data[data[conditions]>data[conditions].mean()]['label'].unique())
    print('**********************************************************')
    print("CROPS WHICH REQUIRE less THAN AVERAGE", conditions,':')
    print(data[data[conditions]<=data[conditions].mean()]['label'].unique())
```

conditions N

```
CROPS WHICH REQUIRE GRAETER THAN AVERAGE N :
['rice' 'maize' 'chickpea' 'blackgram' 'banana' 'watermelon' 'muskmelon'
 'papaya' 'cotton' 'jute' 'coffee']
**********************************************************
CROPS WHICH REQUIRE less THAN AVERAGE N :
['chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram'
 'lentil' 'pomegranate' 'mango' 'grapes' 'apple' 'orange' 'papaya'
 'coconut']
```

In [12]:
```
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF NITROGEN CONTENT IN SOIL",data[dat
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF PHOSPHORUS CONTENT IN SOIL",data[d
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF POTASSIUM CONTENT IN SOIL",data[da
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF RAINFALLCONTENT IN SOIL",data[data
print("CROPS WHICH REQUIRES VERY HIGH TEMPERATURE CONTENT IN SOIL",data[data['tem
print("CROPS WHICH REQUIRES VERY LOW TEMPERATURE CONTENT IN SOIL",data[data['temp
print("CROPS WHICH REQUIRES VERY LOW HUMIDITY CONTENT IN SOIL",data[data['humidit
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF LOW PH CONTENT IN SOIL",data[data[
print("CROPS WHICH REQUIRES VERY HIGH RATIO OF HIGH PH CONTENT IN SOIL",data[data
```

```
CROPS WHICH REQUIRES VERY HIGH RATIO OF NITROGEN CONTENT IN SOIL ['cotton']
CROPS WHICH REQUIRES VERY HIGH RATIO OF PHOSPHORUS CONTENT IN SOIL ['grapes' 'a
pple']
CROPS WHICH REQUIRES VERY HIGH RATIO OF POTASSIUM CONTENT IN SOIL ['grapes' 'ap
ple']
CROPS WHICH REQUIRES VERY HIGH RATIO OF RAINFALLCONTENT IN SOIL ['rice' 'papay
a' 'coconut']
CROPS WHICH REQUIRES VERY HIGH TEMPERATURE CONTENT IN SOIL ['grapes']
CROPS WHICH REQUIRES VERY LOW TEMPERATURE CONTENT IN SOIL ['grapes' 'papaya']
CROPS WHICH REQUIRES VERY LOW HUMIDITY CONTENT IN SOIL ['rice' 'maize' 'kidneyb
eans' 'pigeonpeas' 'mothbeans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes' 'watermelon'
 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton' 'jute' 'coffee']
CROPS WHICH REQUIRES VERY HIGH RATIO OF LOW PH CONTENT IN SOIL ['mothbeans']
CROPS WHICH REQUIRES VERY HIGH RATIO OF HIGH PH CONTENT IN SOIL ['mothbeans']
```

In [17]:
```
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())
```

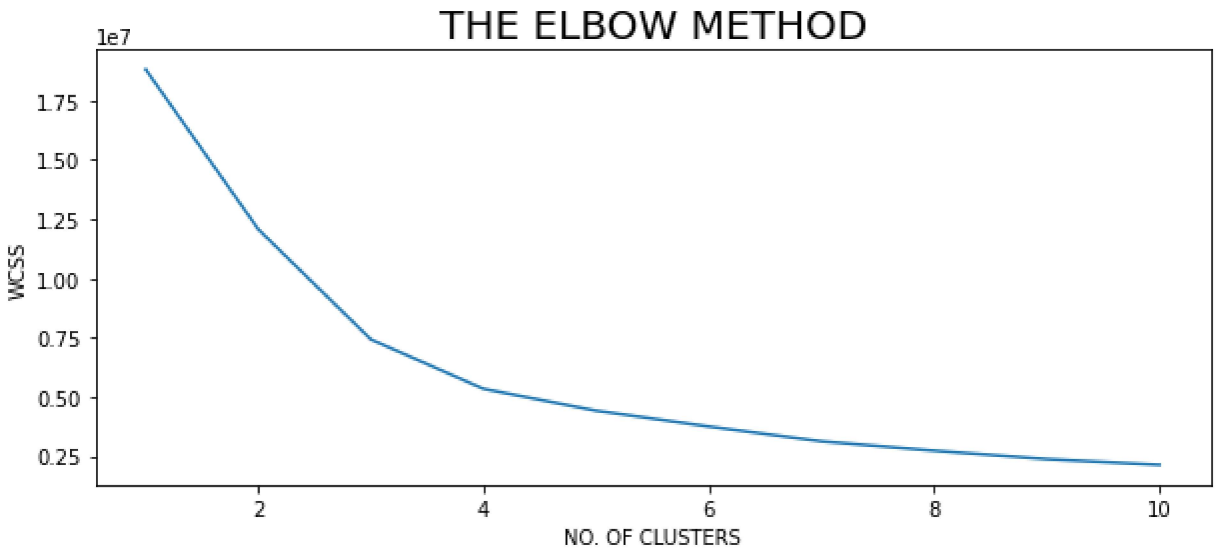
```
SUMMER CROPS
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
**************************************************************
WINTER CROPS
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
**************************************************************
RAINY CROPS
['rice' 'papaya' 'coconut']
```

In [19]:
```
from sklearn.cluster import KMeans
x=data.drop(['label'], axis=1)
x=x.values
print(x.shape)
```

```
(2200, 7)
```

In [23]:
```
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_)


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]:
```
km=KMeans(n_clusters=4,init='k-means++', max_iter=300,n_init=10,random_state=0)
y_means=km.fit_predict(x)

a=data['label']
y_means=pd.DataFrame(y_means)
z=pd.concat([y_means,a],axis=1)
z=z.rename(columns={0:'cluster'})

print("CROP IN FIRST CLUSTER :--", z[z['cluster']==0]['label'].unique())
print("CROP IN FIRST CLUSTER :--", z[z['cluster']==1]['label'].unique())

print("CROP IN FIRST CLUSTER :--", z[z['cluster']==2]['label'].unique())
print("CROP IN FIRST CLUSTER :--", z[z['cluster']==3]['label'].unique())
```

```
CROP IN FIRST CLUSTER :-- ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothb
eans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
CROP IN FIRST CLUSTER :-- ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya'
'cotton' 'coffee']
CROP IN FIRST CLUSTER :-- ['grapes' 'apple']
CROP IN FIRST CLUSTER :-- ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffe
e']
```

In [34]:
```
y=data['label']
x=data.drop(['label'],axis=1)
print(" X SHAPE",x.shape)
print(" Y SHAPE",y.shape)
```

```
 X SHAPE (2200, 7)
 Y SHAPE (2200,)
```

In [33]:
```
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(" x train shape", x_train.shape)
print(" x test shape", x_test.shape)
print(" y train shape", y_train.shape)
print(" y test shape", y_test.shape)
```

```
 x train shape (1760, 7)
 x test shape (440, 7)
 y train shape (1760,)
 y test shape (440,)
```

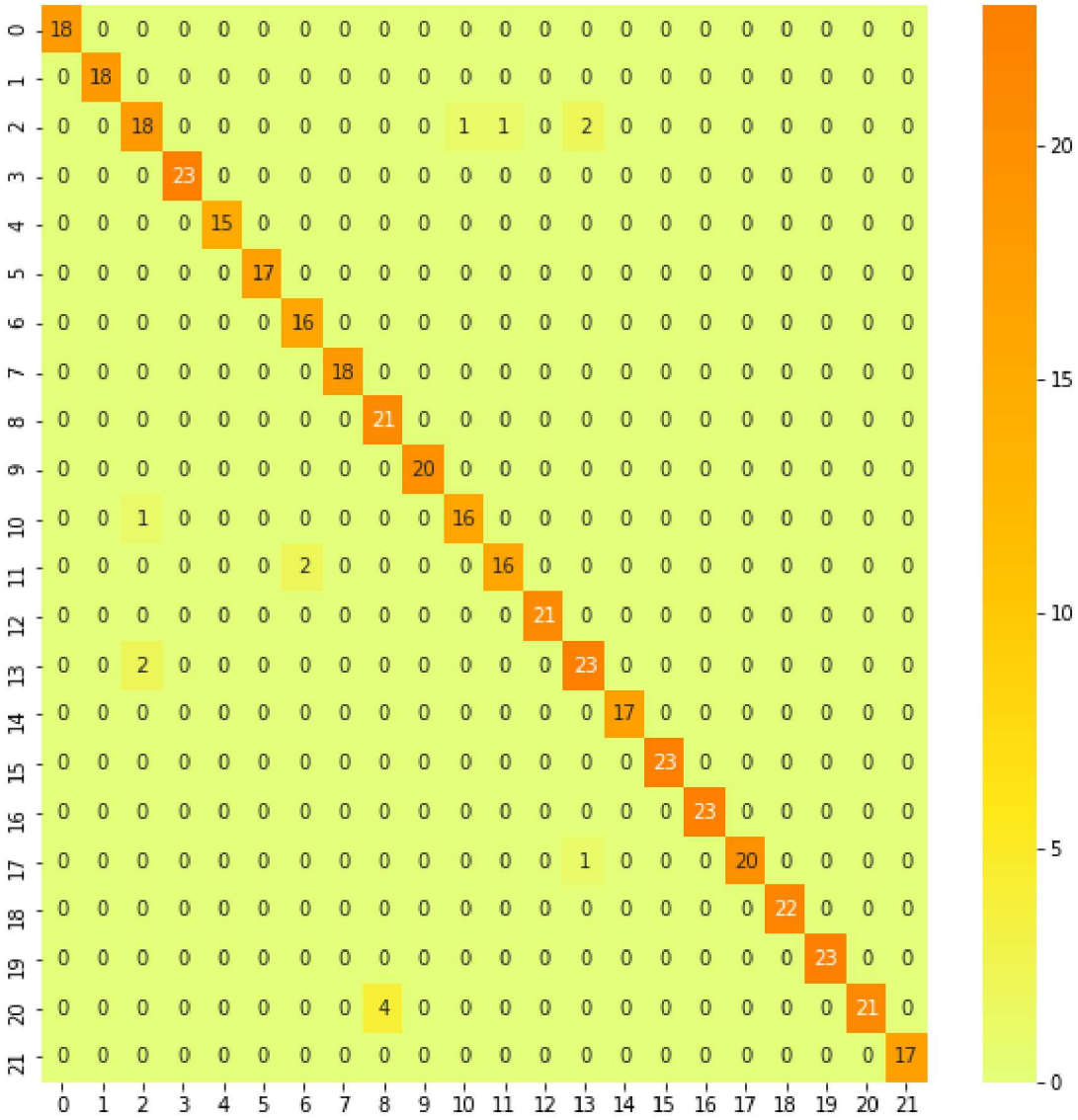
In [35]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_predict=model.predict(x_test)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: 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 (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-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

In [41]:
```
from sklearn.metrics import confusion_matrix

plt.rcParams['figure.figsize']=(10,10)
cm=confusion_matrix(y_test,y_predict)
sns.heatmap(cm,annot=True, cmap='Wistia')
plt.show()
```



In [39]:
```
prediction=model.predict((np.array([[90,40,40,20,80,7,200]])))
print(prediction)
```

```
['rice']
```

In [40]:
```
prediction=model.predict((np.array([[107,34,32,26,66,6,177]])))
print(prediction)
```

```
['coffee']
```

In [ ]:
```

```