In [34]:

```
import IPython
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [35]:

```
df=pd.read_csv(r"C:\Users\Asus\OneDrive\Desktop\project dataset\crp.csv")
```

In [36]:

```
df.head(10)
```

Out[36]:

	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
5	69	37	42	23.058049	83.370118	7.073454	251.055000	rice
6	69	55	38	22.708838	82.639414	5.700806	271.324860	rice
7	94	53	40	20.277744	82.894086	5.718627	241.974195	rice
8	89	54	38	24.515881	83.535216	6.685346	230.446236	rice
9	68	58	38	23.223974	83.033227	6.336254	221.209196	rice

In [37]:

```
df.tail(10)
```

Out[37]:

	N	P	K	temperature	humidity	ph	rainfall	label
2190	103	40	30	27.309018	NaN	6.348316	141.483164	coffee
2191	118	31	34	27.548230	62.881792	6.123796	181.417081	coffee
2192	106	21	35	25.627355	57.041511	7.428524	188.550654	coffee
2193	116	38	34	23.292503	50.045570	6.020947	183.468585	coffee
2194	97	35	26	24.914610	53.741447	6.334610	166.254931	coffee
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

In [38]:
```
df.shape
```

Out[38]:
```
(2200, 8)
```

In [39]:
```
df.columns
```

Out[39]:
```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'],
dtype='object')
```

In [40]:
```
df.drop_duplicates()
```

Out[40]:

	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 [41]:
```
df.duplicated().sum()
```

Out[41]:
```
0
```

In [42]:
```
df.isnull().sum()
```

Out[42]:
```
N              0
P              0
K              0
temperature    0
humidity       1
ph             0
rainfall       0
label          0
dtype: int64
```

In [43]:
```
df.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     2199 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 [44]:
```
df.describe()
```

Out[44]:

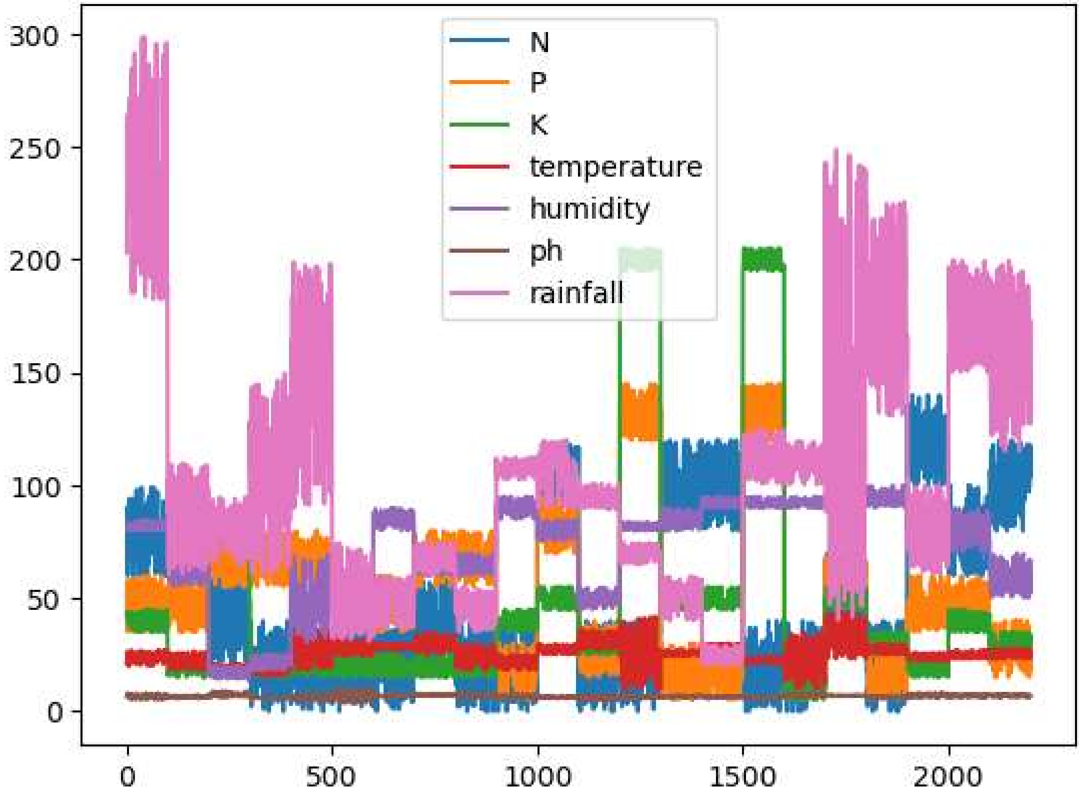
	N	P	K	temperature	humidity	ph	rainf
count	2200.000000	2200.000000	2200.000000	2200.000000	2199.000000	2200.000000	2200.0000
mean	50.551818	53.362727	48.149091	25.616244	71.489185	6.469480	103.4636
std	36.917334	32.985883	50.647931	5.063749	22.266165	0.773938	54.9583
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.2112
25%	21.000000	28.000000	20.000000	22.769375	60.273103	5.971693	64.5516
50%	37.000000	51.000000	32.000000	25.598693	80.474764	6.425045	94.8676
75%	84.250000	68.000000	49.000000	28.561654	89.960531	6.923643	124.2675
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.5601

In [45]:
```
df.nunique()
```

Out[45]:
```
N                137
P                117
K                 73
temperature     2200
humidity        2199
ph              2200
rainfall        2200
label             22
dtype: int64
```

```
In [46]: df.plot()
```

```
Out[46]: <AxesSubplot:>
```



```
In [47]: df['label'].unique()
```

```
Out[47]: 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 [48]:
```
df['label'].value_counts()
```

Out[48]:
```
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 [ ]:
```

```

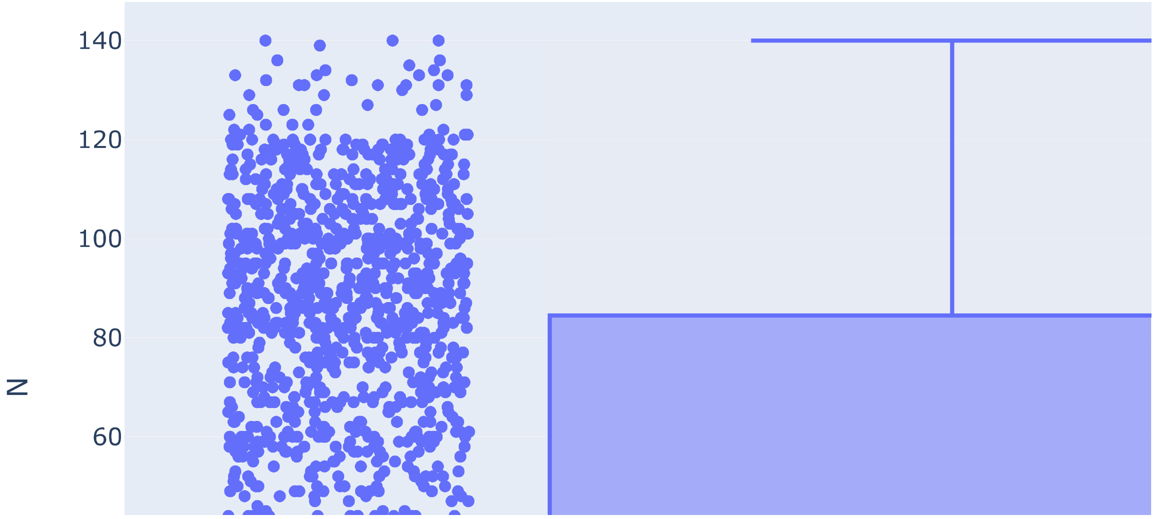
In [49]:
```
df.columns
```

Out[49]:
```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'],
dtype='object')
```

In [50]:
```
import plotly.express as px
```

In [51]:
```
fig=px.box(df,y="N",points="all")
```

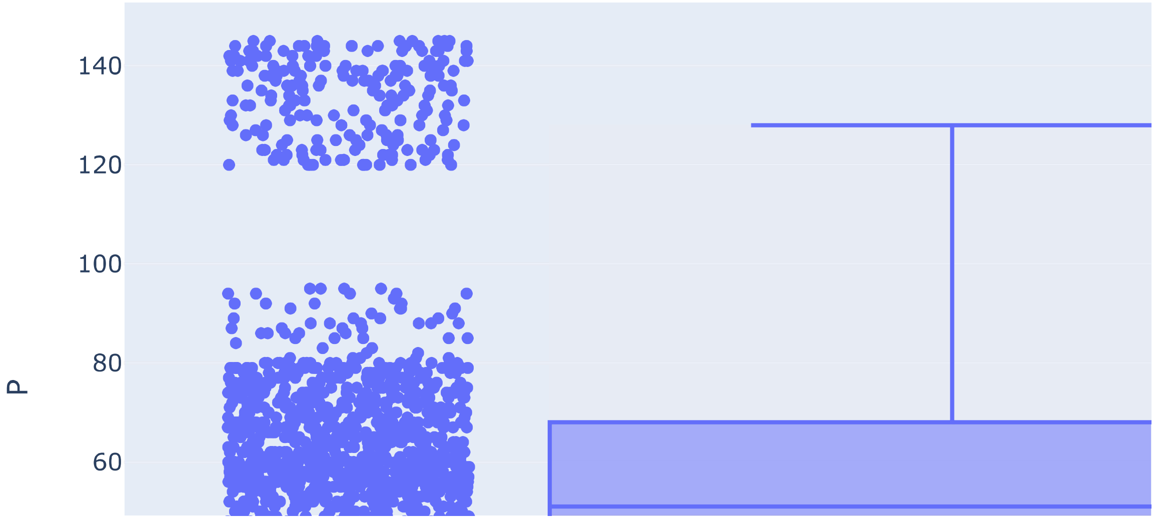
In [52]:
```
fig.show()
```



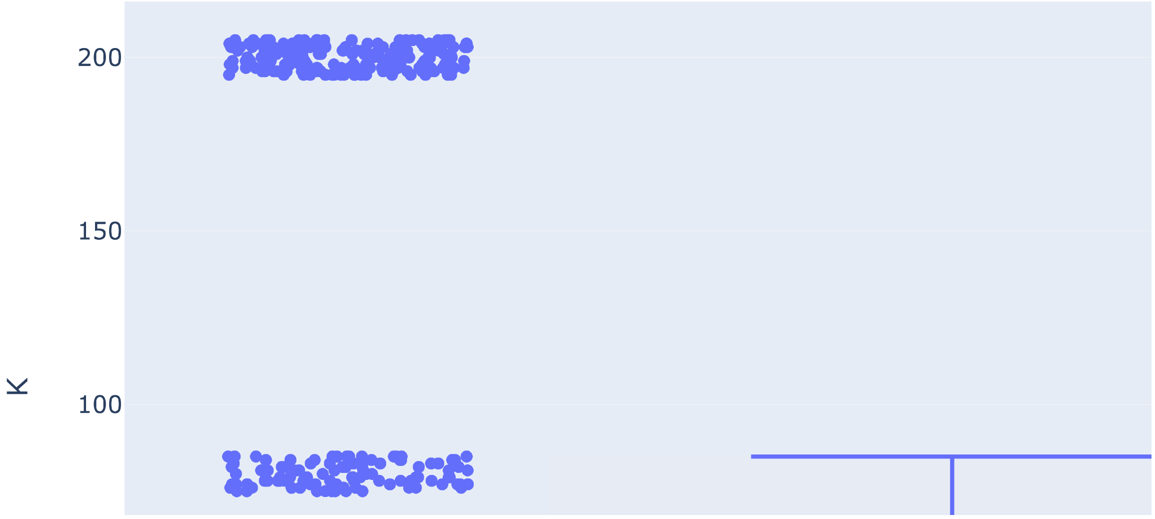
In [53]:
```
# detecting outliers
```

In [54]:
```
fig=px.box(df,y="P",points="all")
```

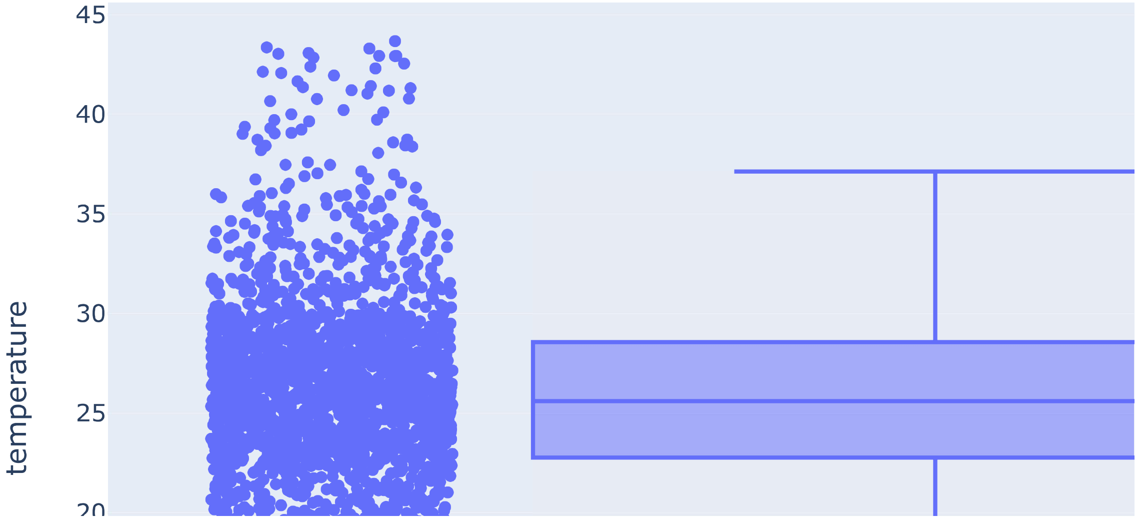
In [55]:
```
fig.show()
```



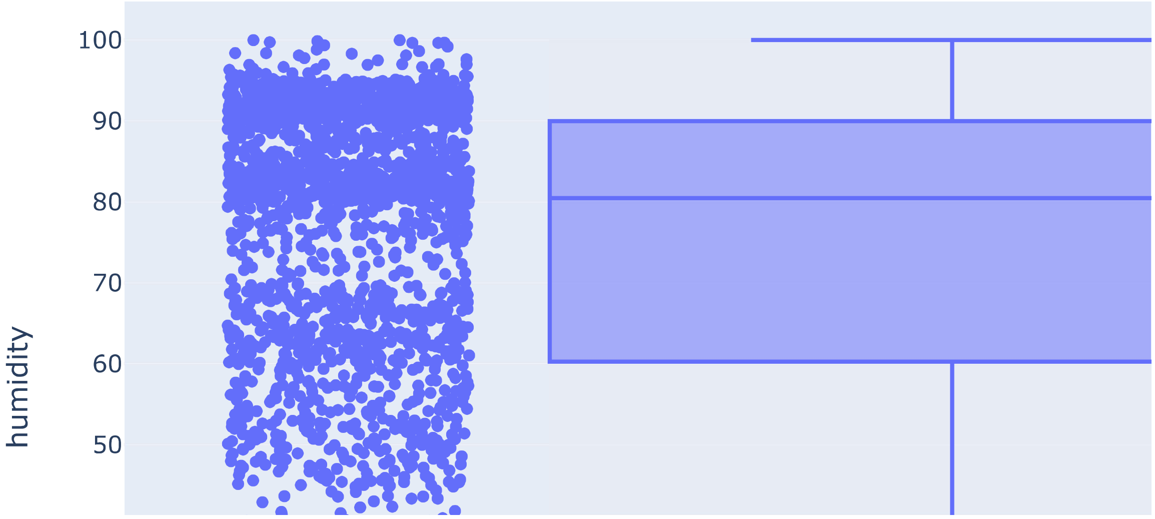
```
In [56]: fig=px.box(df,y="K",points="all")
         fig.show()
```



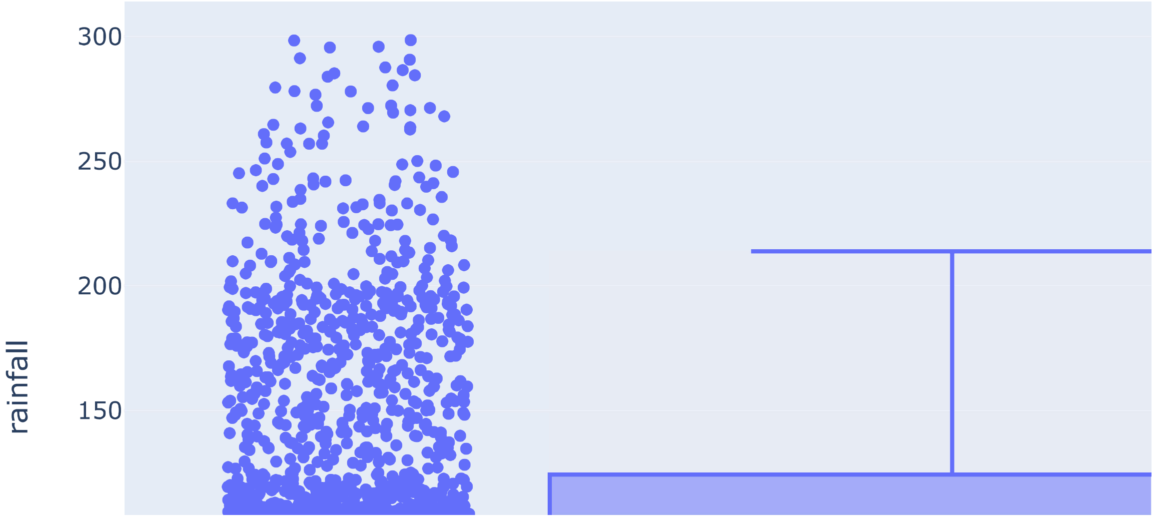
```
In [57]: fig=px.box(df,y="temperature",points="all")
         fig.show()
```



```
In [58]: fig=px.box(df,y="humidity",points="all")
         fig.show()
```



In [59]:
```
fig=px.box(df,y="rainfall",points="all")
fig.show()
```



In [60]:
```
df_boston=df
df_boston.columns=df_boston.columns
```

In [61]:
```
df_boston.head()
```

Out[61]:

	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 [62]: Q1 = np.percentile(df_boston['rainfall'],25,
                            interpolation='midpoint')
```

```
In [63]: Q3=np.percentile(df_boston['rainfall'],75,
                            interpolation='midpoint')
```

```
In [64]: IQR= Q3-Q1
```

```
In [65]: print("OLD SHAPE : ",df_boston.shape)

         OLD SHAPE :  (2200, 8)
```

```
In [66]: # upper bond
```

```
In [67]: upper=np.where(df_boston['rainfall']>=(Q3+1.5*IQR))
```

```
In [68]: # lower bound
```

```
In [69]: lower=np.where(df_boston['rainfall']<=(Q1-1.5*IQR))
```

```
In [70]: #removing outliers
```

```
In [71]: df_boston.drop(upper[0],inplace= True)
         df_boston.drop(lower[0],inplace= True)
```

```
In [72]: print("New Shape : ",df_boston.shape)

         New Shape :  (2101, 8)
```

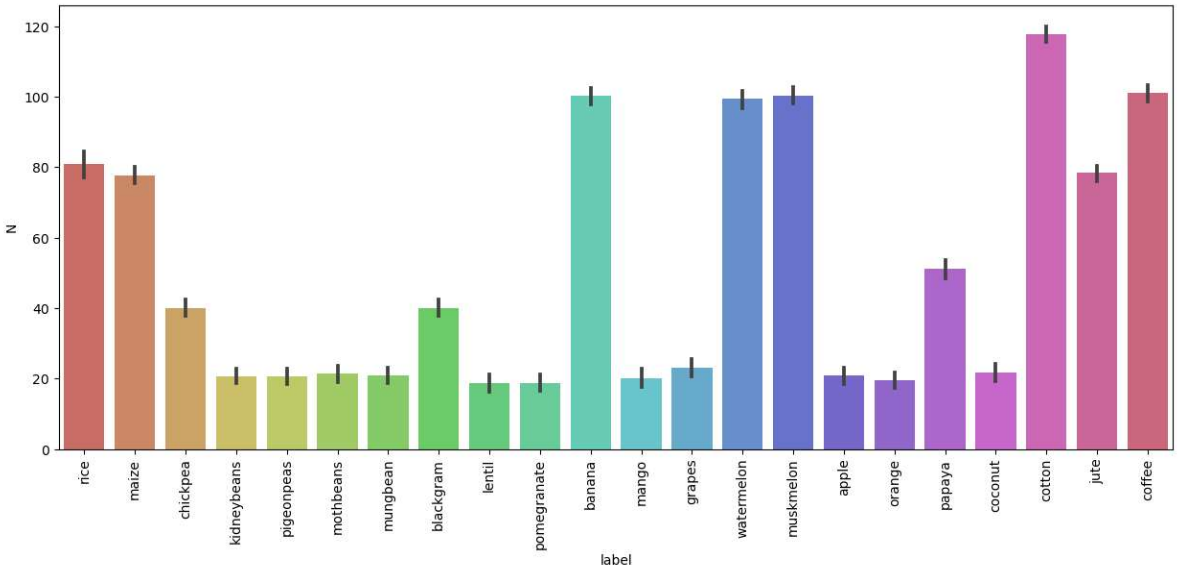
```
In [73]: data=df_boston
```

```
In [74]: import matplotlib.pyplot as plt
```

```
In [75]: import seaborn as sns
```

In [76]:

```
plt.figure(figsize=(15,6))
sns.barplot(y='N',x='label',data=data,palette='hls')
plt.xticks(rotation=90)
plt.show()
```



In [77]:

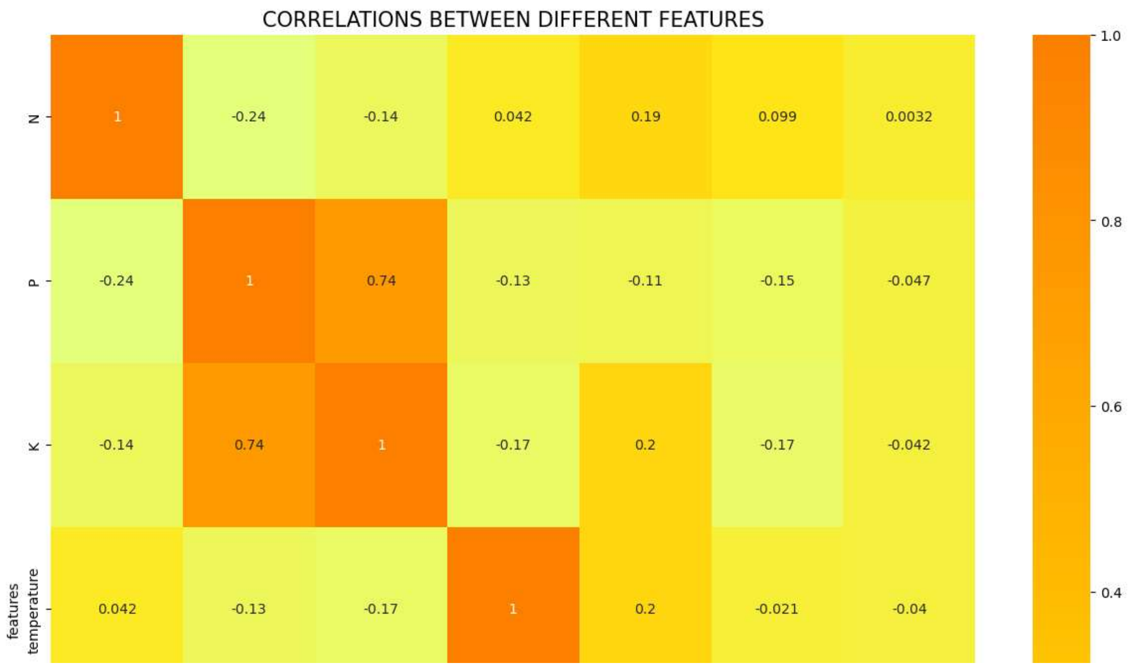
```
df.corr()
```

Out[77]:

	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.237127	-0.139970	0.041633	0.190194	0.099238	0.003231
P	-0.237127	1.000000	0.737806	-0.133415	-0.111882	-0.146018	-0.046656
K	-0.139970	0.737806	1.000000	-0.165188	0.198030	-0.174559	-0.042466
temperature	0.041633	-0.133415	-0.165188	1.000000	0.203929	-0.021339	-0.039570
humidity	0.190194	-0.111882	0.198030	0.203929	1.000000	-0.006008	0.021174
ph	0.099238	-0.146018	-0.174559	-0.021339	-0.006008	1.000000	-0.127166
rainfall	0.003231	-0.046656	-0.042466	-0.039570	0.021174	-0.127166	1.000000

In [78]:
```
fig,ax=plt.subplots(1,1,figsize=(15,15))
sns.heatmap(df.corr(),annot=True,cmap='Wistia')
ax.set(xlabel='features')
ax.set(ylabel='features')

plt.title('CORRELATIONS BETWEEN DIFFERENT FEATURES',fontsize=15,c='black')
plt.show()
```



In [79]:
```
X=df.drop('label',axis=1)
y=df['label']
```

In [80]:
```
from sklearn.model_selection import train_test_split
```

In [81]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,shuffle=True
```

In [82]:
```
pip install lightgbm
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: lightgbm in c:\users\asus\appdata\roaming\pyth
on\python39\site-packages (3.3.5)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (from lightgbm) (1.21.5)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\programdata\anacond
a3\lib\site-packages (from lightgbm) (1.0.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-pac
kages (from lightgbm) (1.9.1)
Requirement already satisfied: wheel in c:\programdata\anaconda3\lib\site-pac
kages (from lightgbm) (0.37.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anacond
a3\lib\site-packages (from scikit-learn!=0.22.0->lightgbm) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\s
ite-packages (from scikit-learn!=0.22.0->lightgbm) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [83]:
```
import lightgbm as lgb
```

In [84]:
```
model= lgb.LGBMClassifier()
model.fit(X_train,y_train)
```

Out[84]:
```
LGBMClassifier()
```

In [85]:
```
y_pred=model.predict(X_test)
```

In [86]:
```
from sklearn.metrics import accuracy_score
```

In [87]:
```
accuracy=accuracy_score(y_pred,y_test)
```

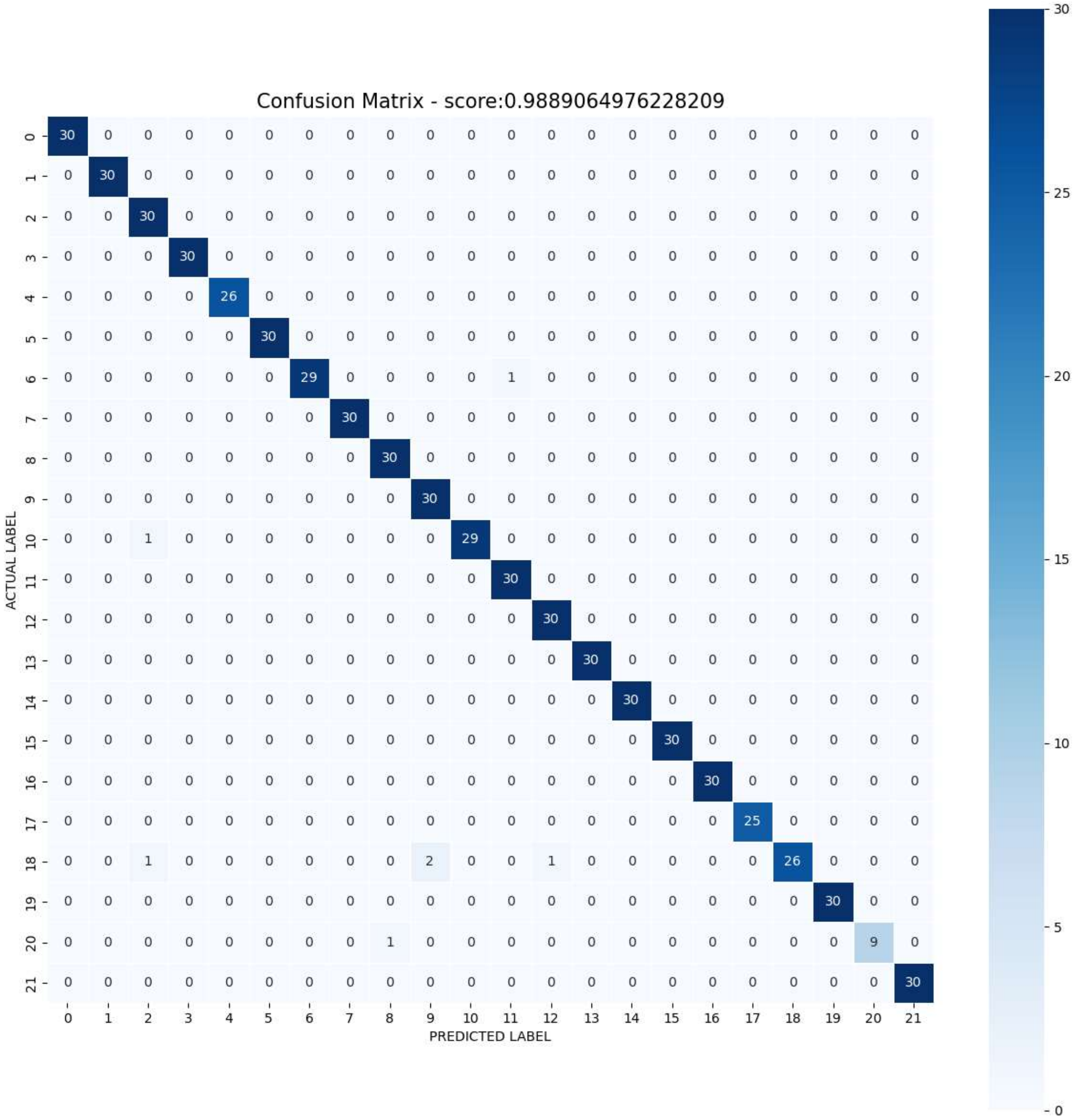
In [88]:
```
print('LightGBM model accuracy score : {0:0.4f}'.format(accuracy_score(y_test,
```

```
LightGBM model accuracy score : 0.9889
```

In [89]:
```
from sklearn.metrics import confusion_matrix
```

In [90]:
```
cm=confusion_matrix(y_test,y_pred)
```

```
In [91]: plt.figure(figsize=(15,15))
         sns.heatmap(cm,annot=True, fmt=".0f",linewidth=.5,square=True,cmap='Blues');
         plt.ylabel('ACTUAL LABEL');
         plt.xlabel('PREDICTED LABEL');
         all_sample_title='Confusion Matrix - score:'+str(accuracy_score(y_test,y_pred)
         plt.title(all_sample_title,size=15);
         plt.show()
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	29	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0
18	0	0	1	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	26	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0
20	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	9	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30

In [92]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        30
      banana       1.00      1.00      1.00        30
   blackgram       0.94      1.00      0.97        30
    chickpea       1.00      1.00      1.00        30
     coconut       1.00      1.00      1.00        26
      coffee       1.00      1.00      1.00        30
      cotton       1.00      0.97      0.98        30
      grapes       1.00      1.00      1.00        30
        jute       0.97      1.00      0.98        30
 kidneybeans       0.94      1.00      0.97        30
      lentil       1.00      0.97      0.98        30
       maize       0.97      1.00      0.98        30
       mango       0.97      1.00      0.98        30
   mothbeans       1.00      1.00      1.00        30
    mungbean       1.00      1.00      1.00        30
   muskmelon       1.00      1.00      1.00        30
      orange       1.00      1.00      1.00        30
      papaya       1.00      1.00      1.00        25
  pigeonpeas       1.00      0.87      0.93        30
 pomegranate       1.00      1.00      1.00        30
        rice       1.00      0.90      0.95        10
  watermelon       1.00      1.00      1.00        30

    accuracy                           0.99       631
   macro avg       0.99      0.99      0.99       631
weighted avg       0.99      0.99      0.99       631
```

In [93]:
```
from sklearn.tree import DecisionTreeClassifier
Classifier=DecisionTreeClassifier(criterion='entropy',random_state=0)
```

In [94]:
```
X = np.nan_to_num(X_train)
Y= np.nan_to_num(y_train)
```

In [95]:
```
Classifier.fit(X,Y)
```

Out[95]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [96]:
```
y_pred=Classifier.predict(X_test)
```

In [97]:
```
from sklearn.metrics import accuracy_score
```

In [98]:
```
accuracy=accuracy_score(y_test,y_pred)
```

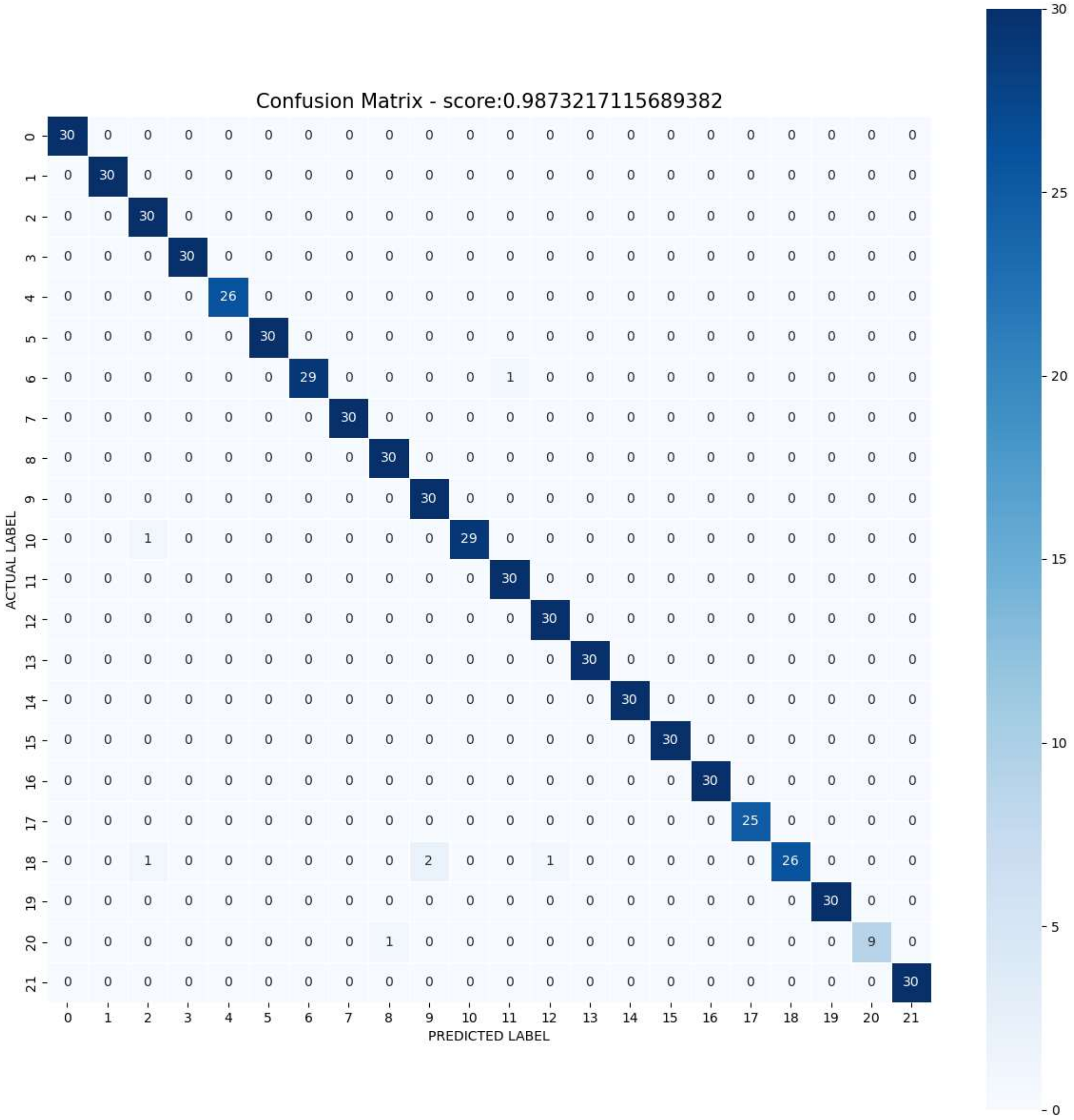
In [99]:

```
print('DECISION TREE MODEL ACCURACY SCORE : {0:0.4f}'.format(accuracy_score(y_
```

```
DECISION TREE MODEL ACCURACY SCORE : 0.9873
```

In [100]:

```
plt.figure(figsize=(15,15))
sns.heatmap(cm,annot=True, fmt=".0f",linewidth=.5,square=True,cmap='Blues');
plt.ylabel('ACTUAL LABEL');
plt.xlabel('PREDICTED LABEL');
all_sample_title='Confusion Matrix - score:'+str(accuracy_score(y_test,y_pred)
plt.title(all_sample_title,size=15);
plt.show()
```



In [101]:

```
X_test[0:1]
```

Out[101]:

	N	P	K	temperature	humidity	ph	rainfall
191	91	55	15	18.093002	72.610242	6.376651	78.961595

```
In [102]: result=Classifier.predict(X_test[0:1])
```

```
In [103]: result
```

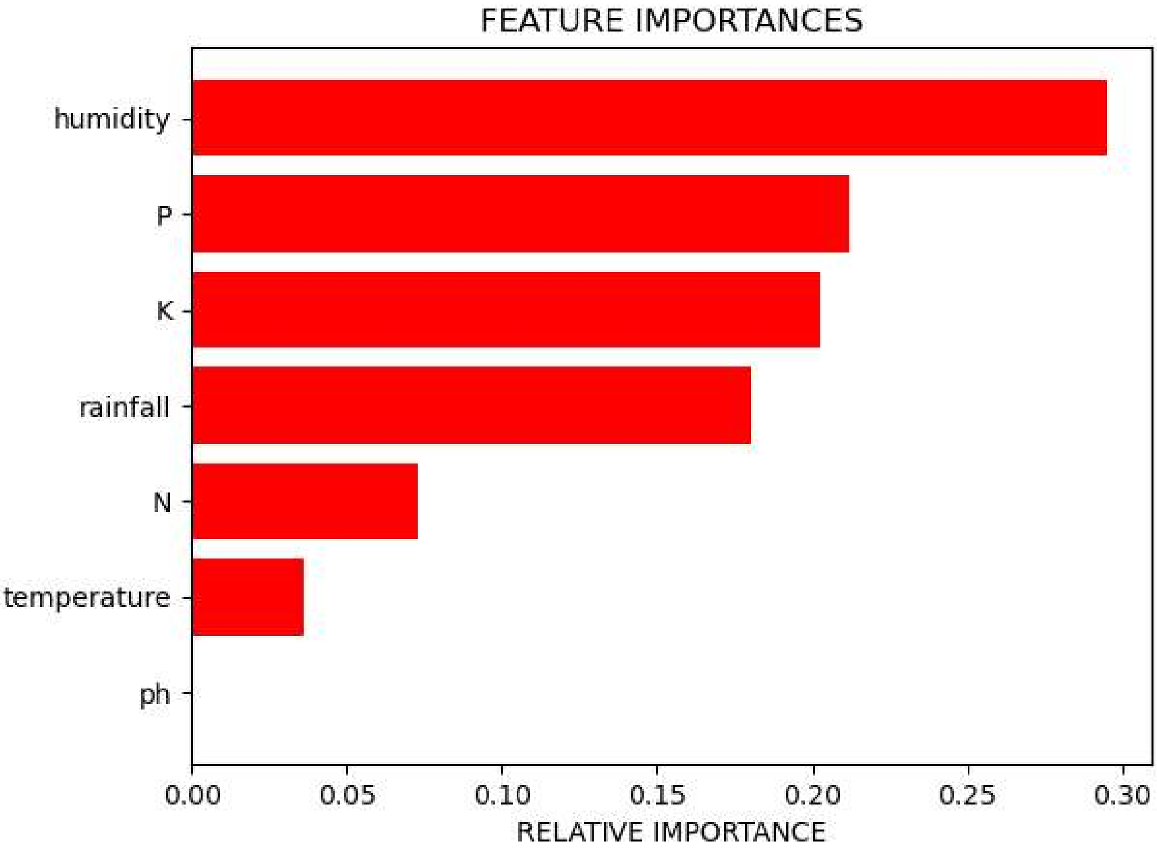
```
Out[103]: array(['maize'], dtype=object)
```

```
In [104]: y_test[0:1]
```

```
Out[104]: 191    maize
          Name: label, dtype: object
```

```
In [105]: features = df.columns
          importances = Classifier.feature_importances_
          indices = np.argsort(importances)

          plt.title('FEATURE IMPORTANCES')
          plt.barh(range(len(indices)),importances[indices], color='r' , align="center")
          plt.yticks(range(len(indices)),[features[i] for i in indices])
          plt.xlabel('RELATIVE IMPORTANCE')
          plt.show()
```


FEATURE IMPORTANCES
humidity
P
K
rainfall
N
temperature
ph
RELATIVE IMPORTANCE