CROP PREDICTION

As we know our Country is an agririan country and since year 2002 our governmnet has declared Agriculture as an prime moving force that is the sector to mostly focus-on . More than 50% of the population in our country is based on agricukture but agriculture sectors contribution in our GDP is less than all the other sectors. Governments are taking initatives to lift the conditions of farmers through various schemes , loans etc. But agriculture is depend mostly on natural factors and most farmers are not aware of these and they are just growing the crops which their ancestors were growing from past years but due to global warming and climate change it is not the condition to take on old crops. So, I have developed a model to predict the CROP based on the natural factors.

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import numpy as np
```

In [3]:
```
df=pd.read_csv("Crop_recommendation.csv")
```

In [4]:
```
df
```

Out[4]:

	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 [5]:
```
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     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 [6]:
```
df.describe()
```

Out[6]:

	N	P	K	temperature	humidity	ph	
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298

In [7]:
```
df.isna().sum()
```

Out[7]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
print(df["label"].nunique())
df["label"].unique()
```

```
22
```

Out[9]:
```
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)
```

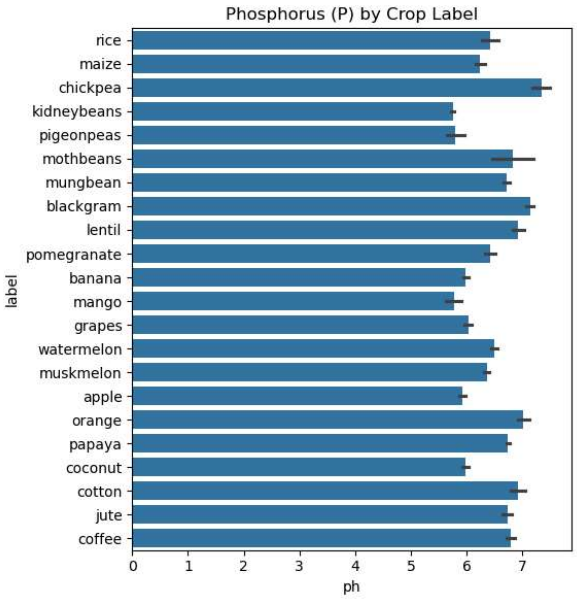
In [10]:
```
crop_mapping = {
    'rice': 'Cereals and Pulses',
    'maize': 'Cereals and Pulses',
    'lentil': 'Cereals and Pulses',
    'blackgram': 'Cereals and Pulses',
    'mungbean': 'Cereals and Pulses',
    'mothbeans': 'Cereals and Pulses',
    'pigeonpeas': 'Cereals and Pulses',
    'kidneybeans': 'Cereals and Pulses',
    'chickpea': 'Cereals and Pulses',
    'papaya': 'Fruits',
    'orange': 'Fruits',
    'apple': 'Fruits',
    'muskmelon': 'Fruits',
    'watermelon': 'Fruits',
    'grapes': 'Fruits',
    'mango': 'Fruits',
    'banana': 'Fruits',
    'pomegranate': 'Fruits',
    'jute': 'Fibres and Beverages',
    'cotton': 'Fibres and Beverages',
    'coffee': 'Fibres and Beverages',
    'coconut': 'Fibres and Beverages'
}
```

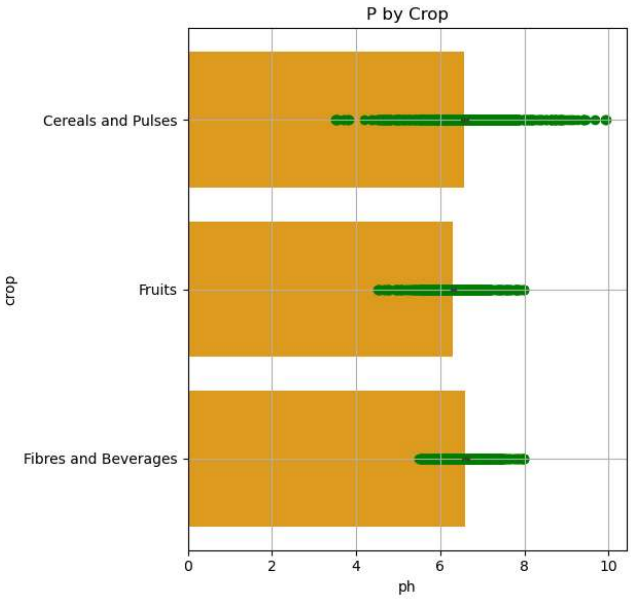
In [11]:
```
df['crop'] = df['label'].replace(crop_mapping)
```

In [12]:
```
df["crop"].value_counts()
```

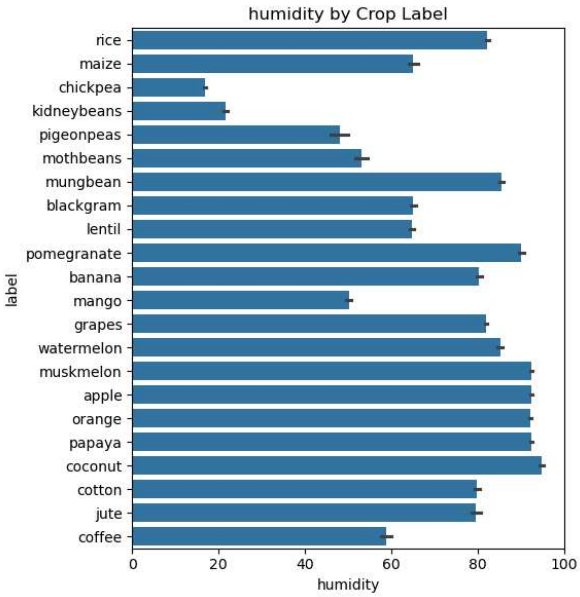
Out[12]:
```
crop
Cereals and Pulses      900
Fruits                  900
Fibres and Beverages    400
Name: count, dtype: int64
```

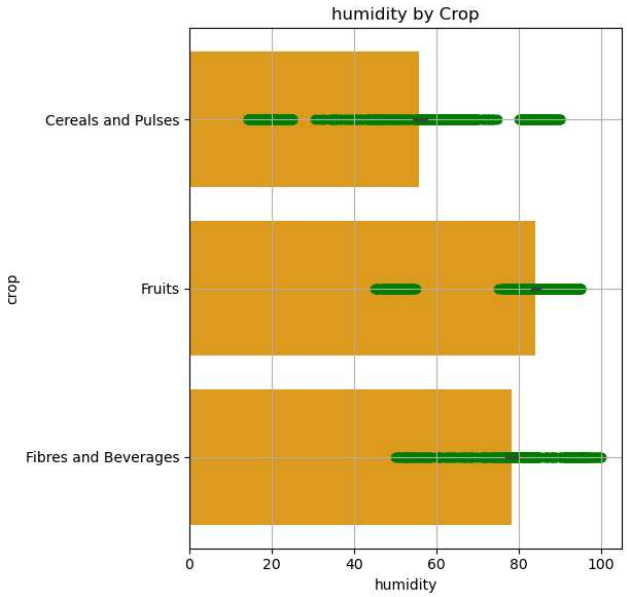
In [13]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="ph", y="label", orient="h", ax=axes[0])
axes[0].set_title("Phosphorus (P) by Crop Label")

sns.barplot(df,x="ph",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["ph"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("P by Crop")

plt.tight_layout()

plt.show()
```





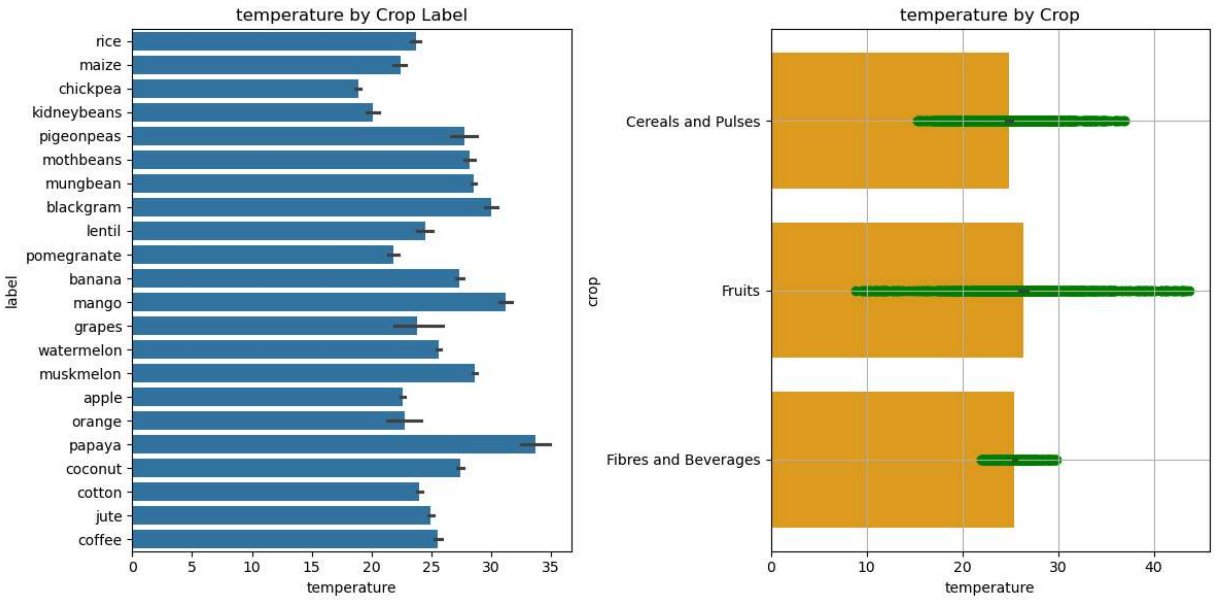
In [14]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="humidity", y="label", orient="h", ax=axes[0])
axes[0].set_title("humidity by Crop Label")

sns.barplot(df,x="humidity",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["humidity"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("humidity by Crop")

plt.tight_layout()
plt.show()
```



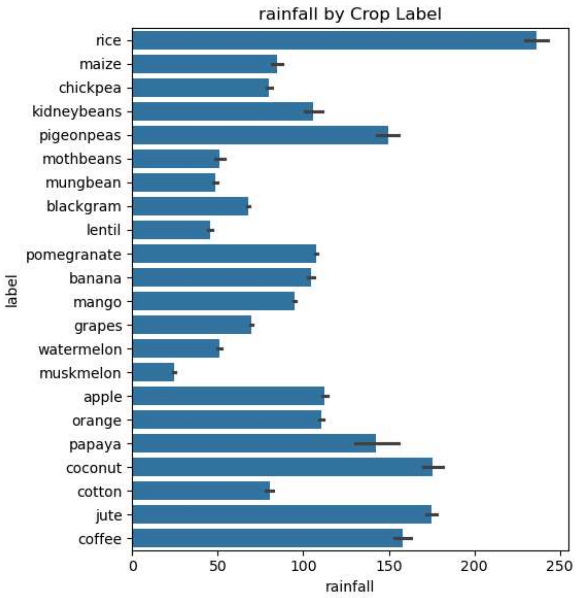


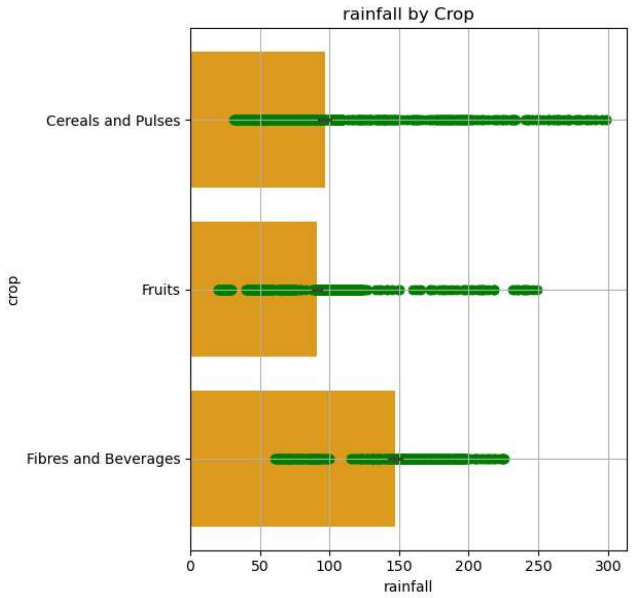
In [15]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="temperature", y="label", orient="h", ax=axes[0])
axes[0].set_title("temperature by Crop Label")
```

```
sns.barplot(df,x="temperature",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["temperature"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("temperature by Crop")

plt.tight_layout()

plt.show()
```

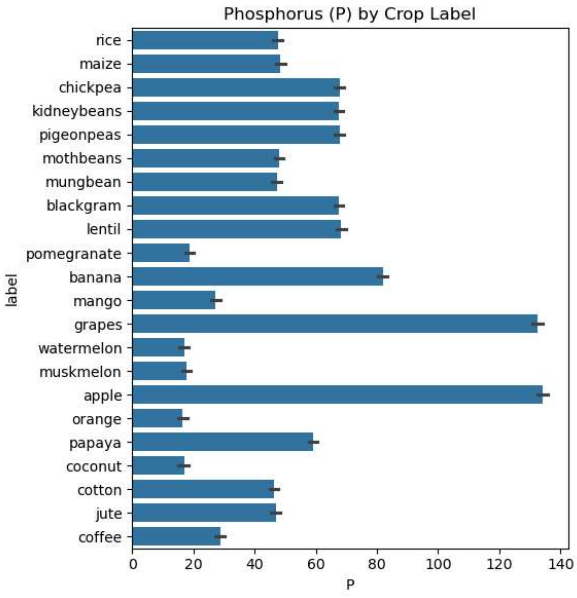


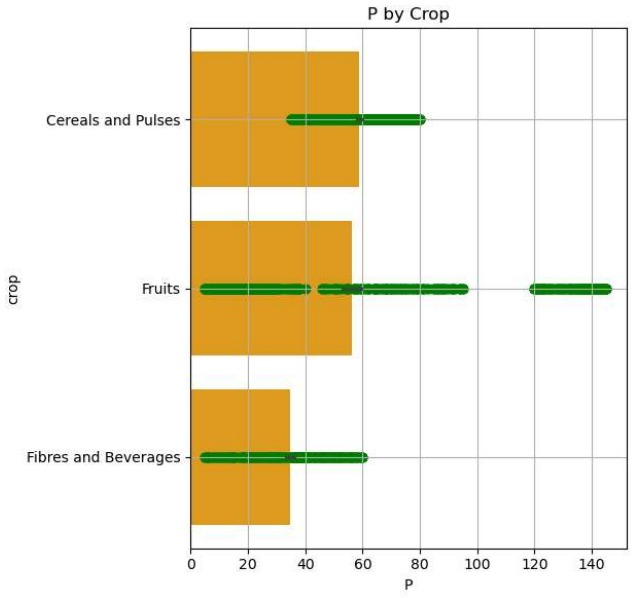
In [16]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="rainfall", y="label", orient="h", ax=axes[0])
axes[0].set_title("rainfall by Crop Label")

sns.barplot(df,x="rainfall",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["rainfall"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("rainfall by Crop")

plt.tight_layout()

plt.show()
```





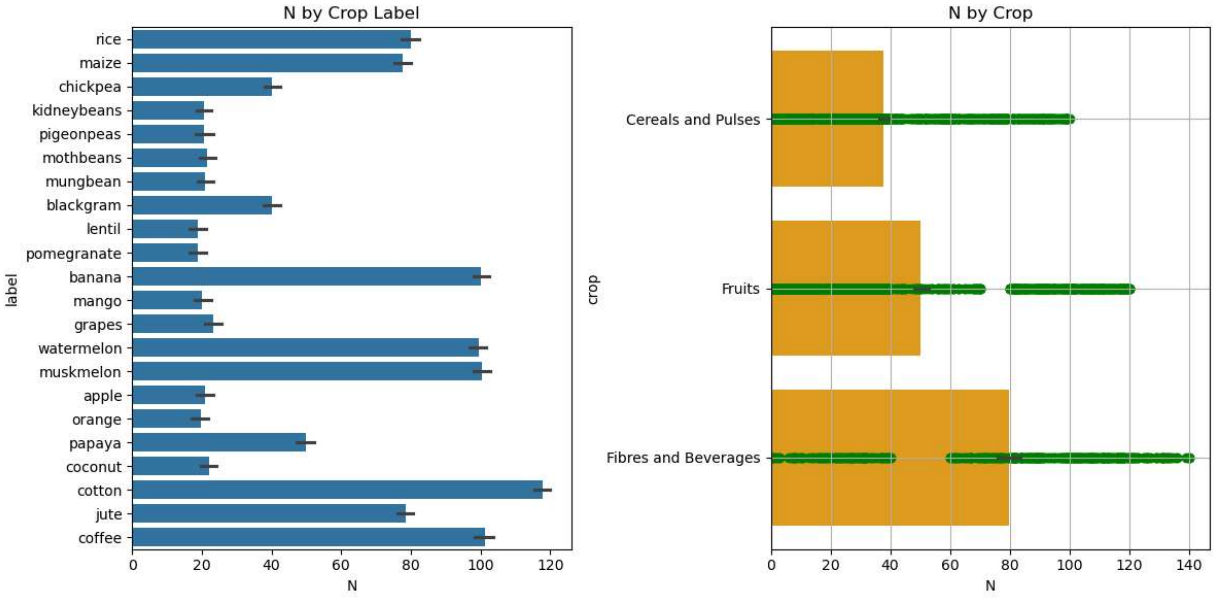
In [17]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="P", y="label", orient="h", ax=axes[0])
axes[0].set_title("Phosphorus (P) by Crop Label")

sns.barplot(df,x="P",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["P"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("P by Crop")

plt.tight_layout()

plt.show()
```



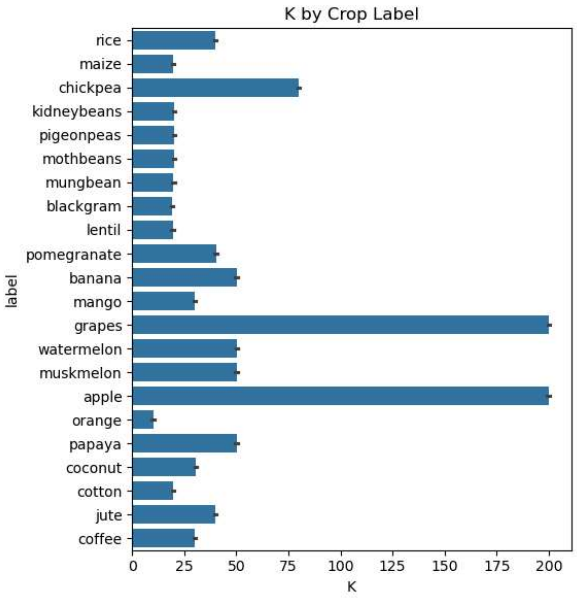


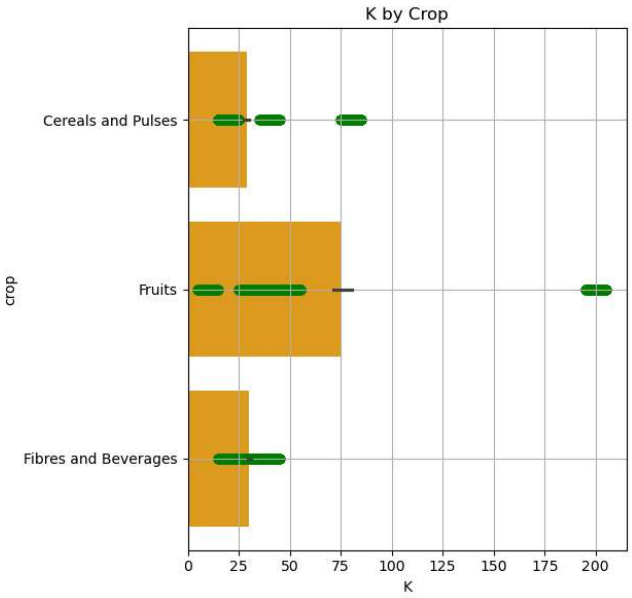
In [18]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="N", y="label", orient="h", ax=axes[0])
axes[0].set_title("N by Crop Label")
```

```
sns.barplot(df,x="N",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["N"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("N by Crop")

plt.tight_layout()

plt.show()
```



In [19]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

sns.barplot(data=df, x="K", y="label", orient="h", ax=axes[0])
axes[0].set_title("K by Crop Label")

sns.barplot(df,x="K",y="crop",orient="h",color="orange", ax=axes[1])
plt.scatter(df["K"],df["crop"],color="green")
plt.grid(True)
axes[1].set_title("K by Crop")

plt.tight_layout()

plt.show()
```





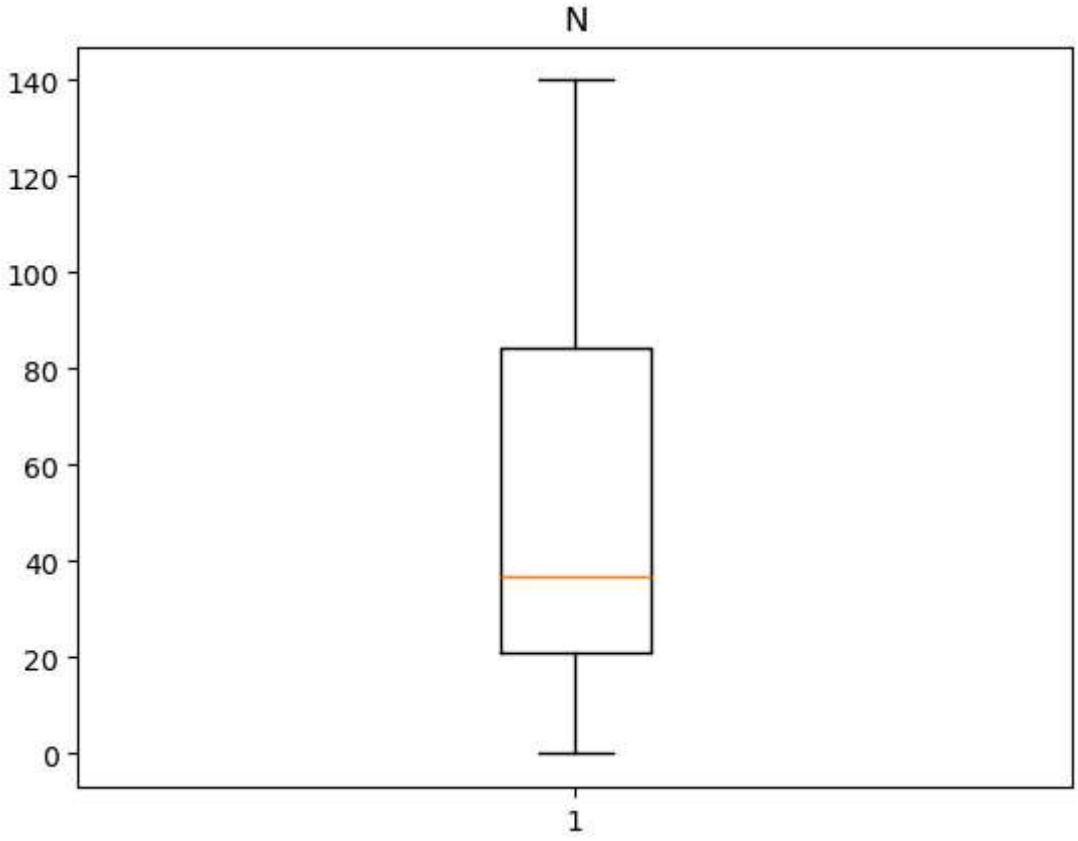
In [20]:
```
cols=df.select_dtypes(exclude="object")
cols
```

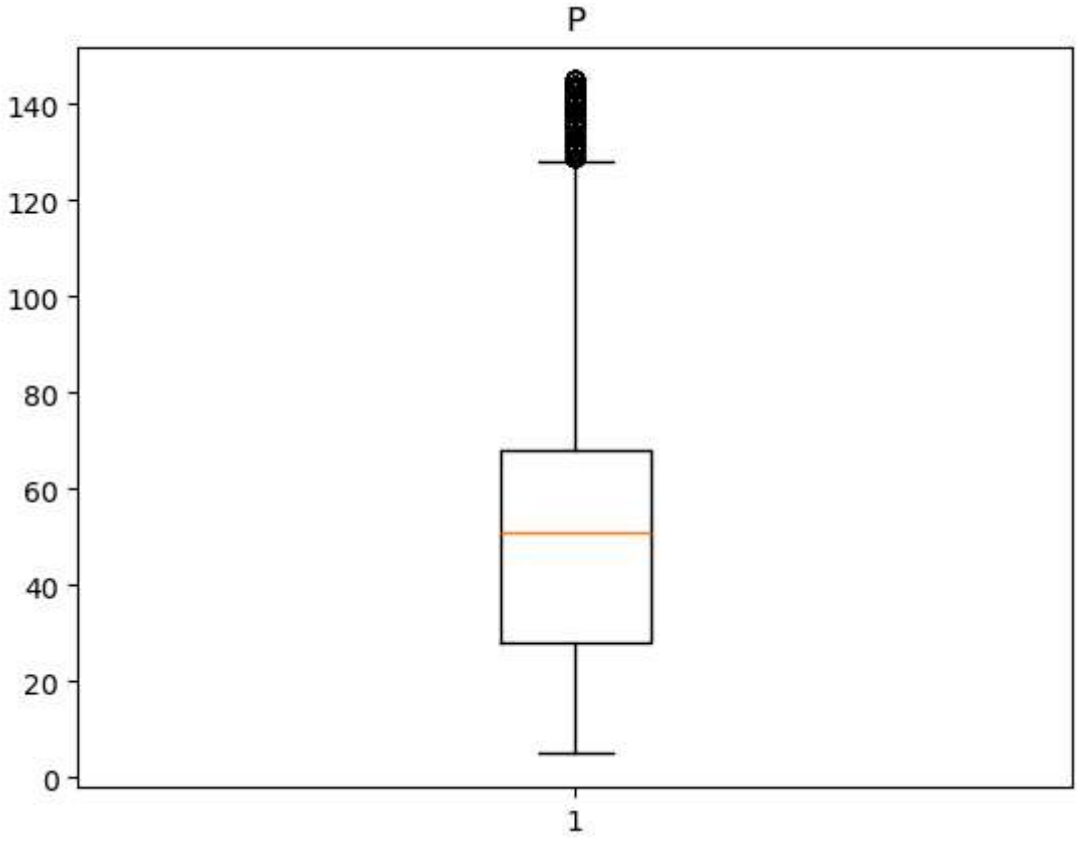
Out[20]:

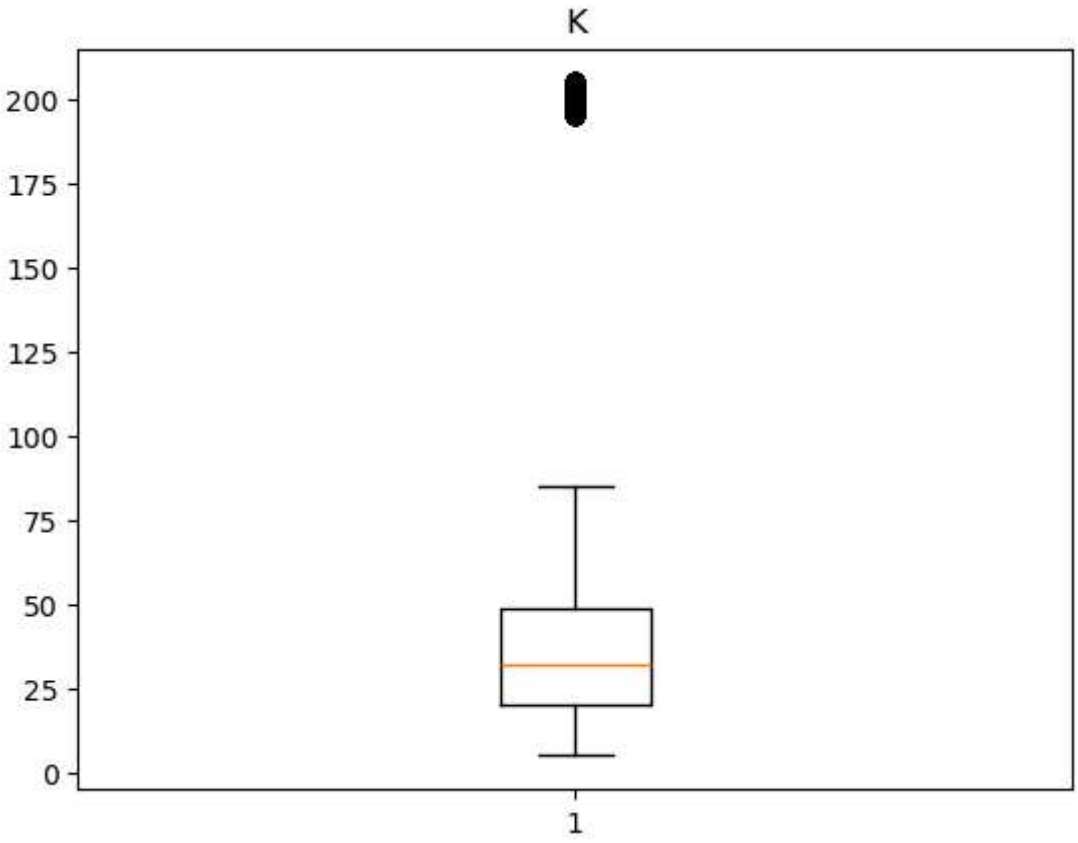
	N	P	K	temperature	humidity	ph	rainfall
0	90	42	43	20.879744	82.002744	6.502985	202.935536
1	85	58	41	21.770462	80.319644	7.038096	226.655537
2	60	55	44	23.004459	82.320763	7.840207	263.964248
3	74	35	40	26.491096	80.158363	6.980401	242.864034
4	78	42	42	20.130175	81.604873	7.628473	262.717340
...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507
2196	99	15	27	27.417112	56.636362	6.086922	127.924610
2197	118	33	30	24.131797	67.225123	6.362608	173.322839
2198	117	32	34	26.272418	52.127394	6.758793	127.175293
2199	104	18	30	23.603016	60.396475	6.779833	140.937041

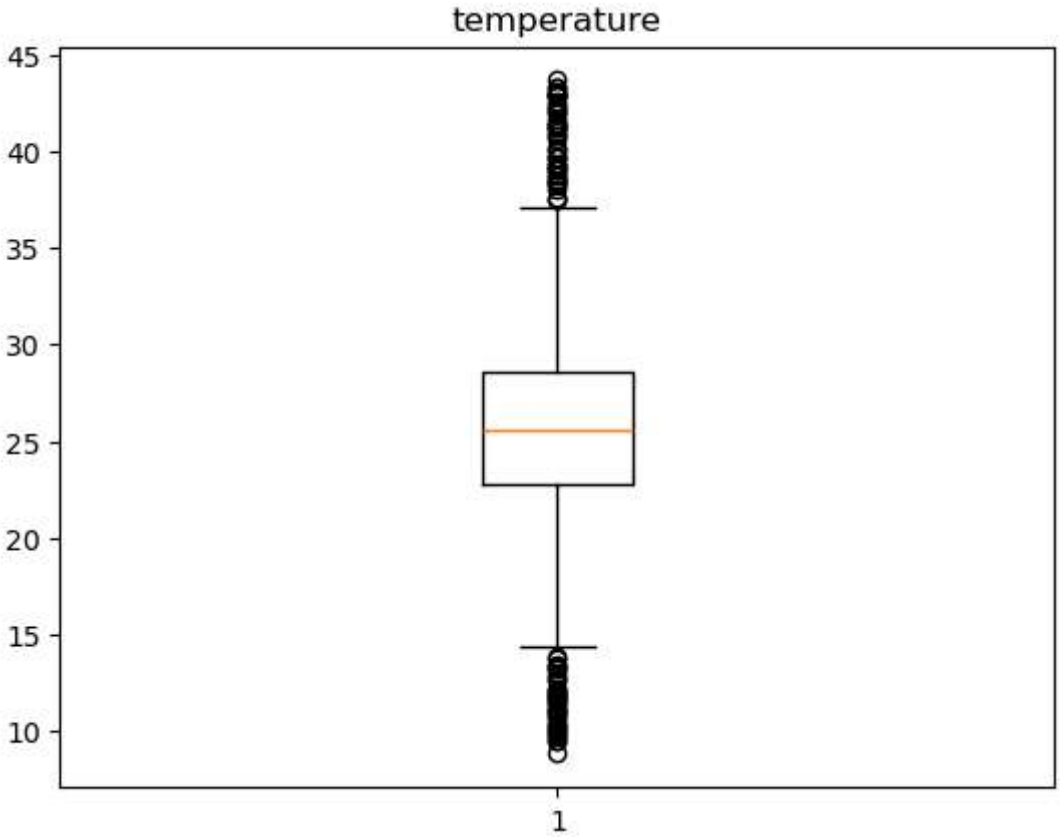
2200 rows × 7 columns

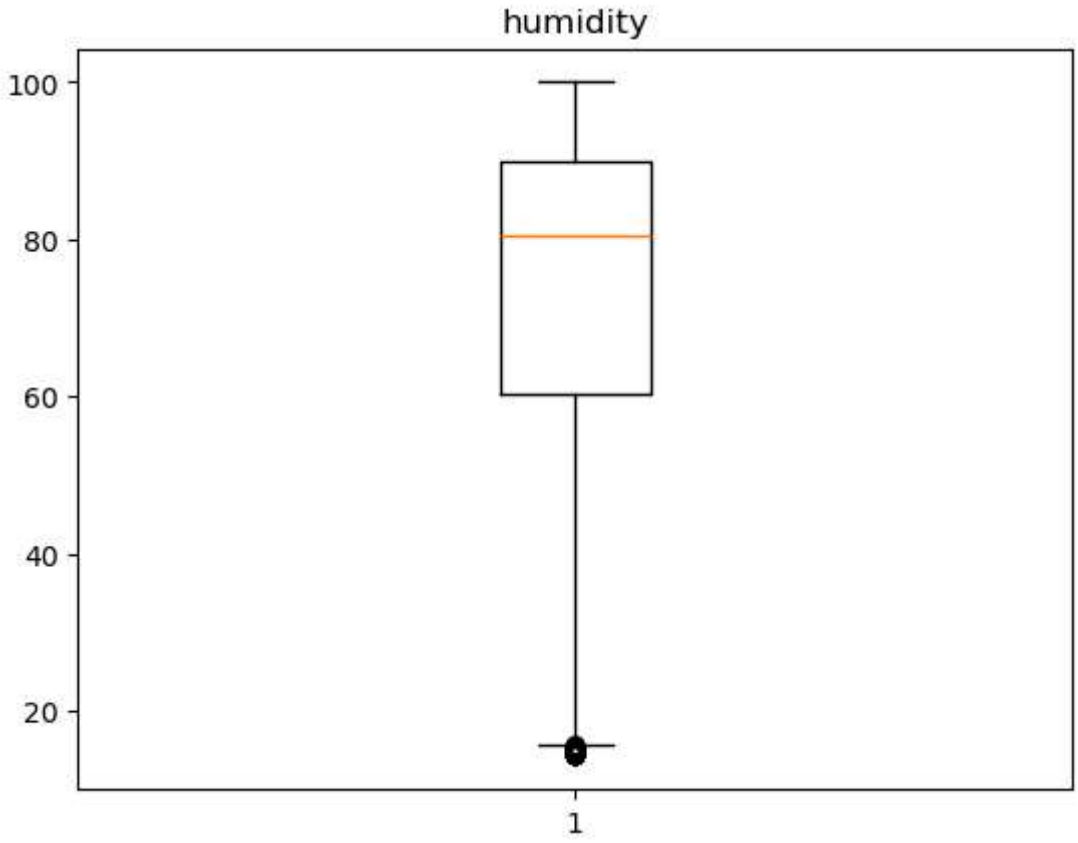
In [21]:
```
for i in cols:
    plt.boxplot(df[i])
    plt.title(i)
    plt.show()
```

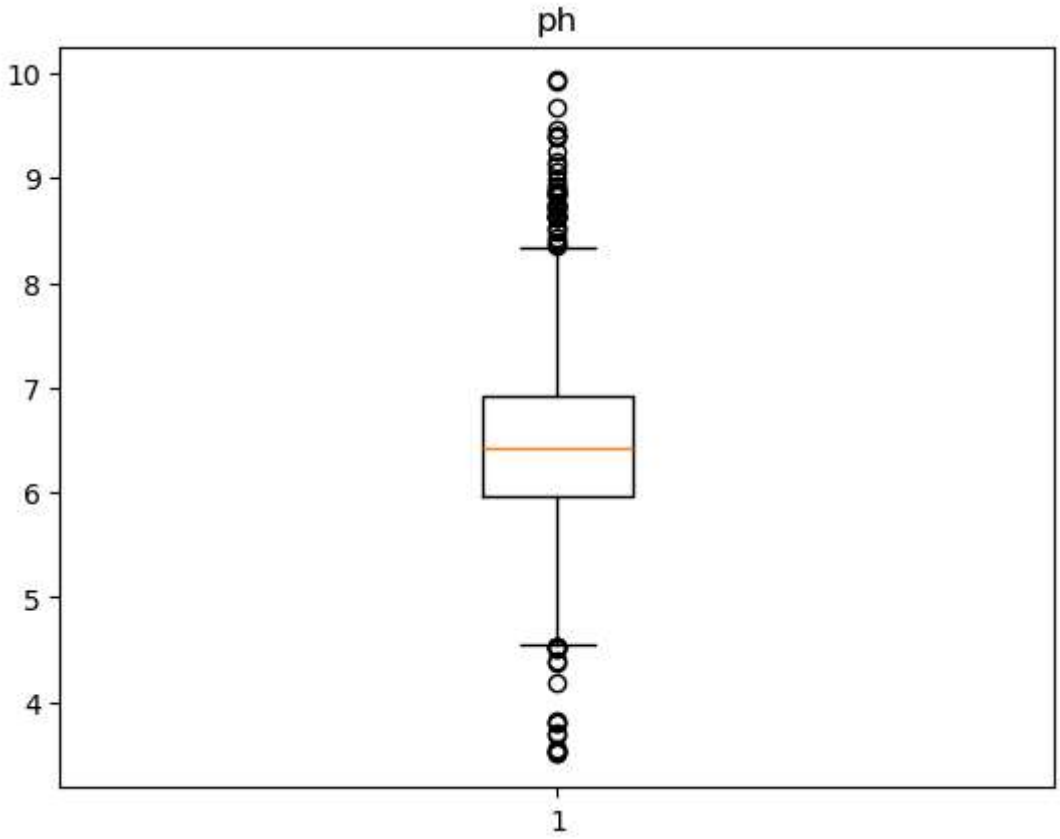

N
140
120
100
80
60
40
20
0
1

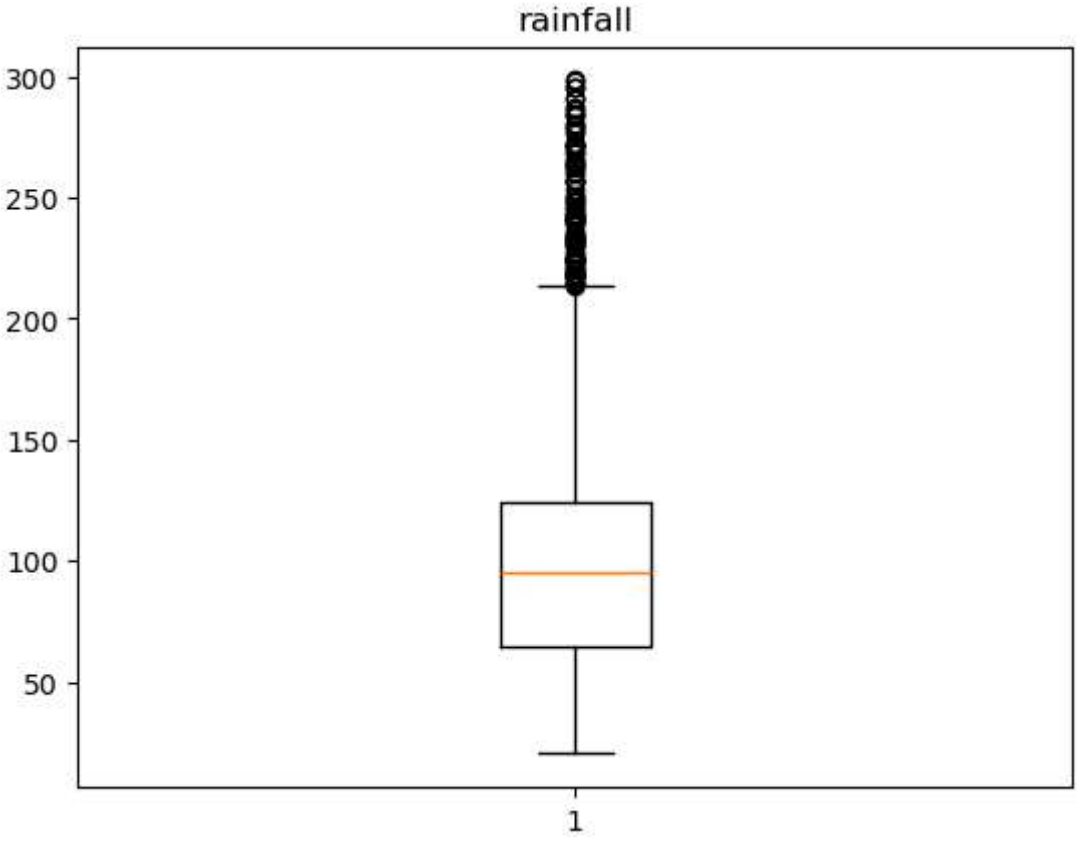

P
140
120
100
80
60
40
20
0
1


K
200
175
150
125
100
75
50
25
0
1


temperature
45
40
35
30
25
20
15
10
1


humidity
100
80
60
40
20
1


ph
10
9
8
7
6
5
4
1



In [22]:
```
df.columns
```

Out[22]:
```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label',
       'crop'],
      dtype='object')
```

In [23]:
```
Q1=df["P"].quantile(0.25)
Q2=df["P"].quantile(0.50)
Q3=df["P"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
128.0
-32.0
```

In [24]:
```
df.loc[df["P"]>UB]
```

Out[24]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1200	24	130	195	29.996772	81.541566	6.112306	67.125345	grapes	Fruits
1201	13	144	204	30.728040	82.426141	6.092242	68.381355	grapes	Fruits
1204	24	131	196	22.032962	83.743728	5.732454	65.344408	grapes	Fruits
1206	35	140	197	16.775573	82.752419	6.106191	66.762855	grapes	Fruits
1209	17	134	204	39.040720	80.183933	6.499605	73.884670	grapes	Fruits
...	...	...	...	...	...	...	...	...	...
1594	35	145	195	22.039115	94.580758	6.231950	110.980401	apple	Fruits
1596	25	132	198	22.319441	90.851744	5.732758	100.117344	apple	Fruits
1597	31	137	196	22.144641	93.825674	6.400321	120.631078	apple	Fruits
1598	36	144	196	23.651676	94.505288	6.496934	115.361127	apple	Fruits
1599	10	140	197	22.169395	90.271856	6.229499	124.468311	apple	Fruits

138 rows × 9 columns

In [25]:
```
df.loc[df["P"]<LB]
```

Out[25]:

N	P	K	temperature	humidity	ph	rainfall	label	crop

In [26]:
```
df.loc[df["P"]>UB,"P"]=UB
```

In [27]:
```
df.loc[df["P"]>UB]
```

Out[27]:

N	P	K	temperature	humidity	ph	rainfall	label	crop

In [28]:
```
Q1=df["K"].quantile(0.25)
Q2=df["K"].quantile(0.50)
Q3=df["K"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
92.5
-23.5
```

In [29]:
```
df.loc[df["K"]>UB]
```

Out[29]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1200	24	128	195	29.996772	81.541566	6.112306	67.125345	grapes	Fruits
1201	13	128	204	30.728040	82.426141	6.092242	68.381355	grapes	Fruits
1202	22	123	205	32.445778	83.885049	5.896343	68.739325	grapes	Fruits
1203	36	125	196	37.465668	80.659687	6.155261	66.838723	grapes	Fruits
1204	24	128	196	22.032962	83.743728	5.732454	65.344408	grapes	Fruits
...	...	...	...	...	...	...	...	...	...
1595	40	120	197	23.805938	92.488795	5.889481	119.633555	apple	Fruits
1596	25	128	198	22.319441	90.851744	5.732758	100.117344	apple	Fruits
1597	31	128	196	22.144641	93.825674	6.400321	120.631078	apple	Fruits
1598	36	128	196	23.651676	94.505288	6.496934	115.361127	apple	Fruits
1599	10	128	197	22.169395	90.271856	6.229499	124.468311	apple	Fruits

200 rows × 9 columns

In [30]:
```
df.loc[df["K"]>UB,"K"]=UB
```

In [31]:
```
df.loc[df["K"]>UB]
```

Out[31]:

N	P	K	temperature	humidity	ph	rainfall	label	crop

In [32]:
```
Q1=df["temperature"].quantile(0.25)
Q2=df["temperature"].quantile(0.50)
Q3=df["temperature"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
37.2500728825
14.080955682499999
```

In [33]:
```
df.loc[df["temperature"]>UB]
```

Out[33]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1203	36	125	92.5	37.465668	80.659687	6.155261	66.838723	grapes	Fruits
1205	2	123	92.5	39.648519	82.210799	6.253035	70.399061	grapes	Fruits
1209	17	128	92.5	39.040720	80.183933	6.499605	73.884670	grapes	Fruits
1210	25	128	92.5	39.707722	82.685935	5.554832	74.915062	grapes	Fruits
1228	30	120	92.5	38.060995	82.247296	6.234904	65.701482	grapes	Fruits
1229	23	128	92.5	39.065555	82.038130	6.000574	69.307729	grapes	Fruits
1247	39	128	92.5	41.186649	81.017834	5.539981	68.688959	grapes	Fruits
1259	17	128	92.5	41.207336	81.610510	6.389783	65.902275	grapes	Fruits
1262	32	128	92.5	40.660123	81.249960	6.372960	74.030301	grapes	Fruits
1269	32	121	92.5	39.371026	81.253539	6.129813	74.081017	grapes	Fruits
1275	8	128	92.5	41.656030	82.221182	5.609256	74.196648	grapes	Fruits
1279	38	128	92.5	41.361063	82.797830	6.444373	69.921075	grapes	Fruits
1287	29	122	92.5	41.948657	81.155952	5.638328	73.068630	grapes	Fruits
1701	58	46	45.0	42.394134	90.790281	6.576261	88.466075	papaya	Fruits
1702	45	47	55.0	38.419163	91.142204	6.751453	119.265388	papaya	Fruits
1704	31	68	45.0	42.923253	90.076005	6.938313	196.240824	papaya	Fruits
1709	69	64	47.0	40.211993	94.507669	6.993473	186.676232	papaya	Fruits
1710	58	51	47.0	42.134740	91.704454	6.757471	197.402901	papaya	Fruits
1716	49	55	53.0	38.441872	93.637390	6.544030	77.715669	papaya	Fruits
1719	57	57	51.0	39.017933	91.488156	6.992234	105.884153	papaya	Fruits
1721	58	67	45.0	38.723828	91.725149	6.702425	62.623771	papaya	Fruits
1722	61	64	52.0	43.302049	92.834054	6.641099	110.562229	papaya	Fruits
1724	31	48	45.0	40.788818	92.909514	6.563135	132.792359	papaya	Fruits
1728	61	51	51.0	39.300500	94.161934	6.574678	120.951247	papaya	Fruits
1729	70	54	46.0	39.731491	91.122206	6.919342	122.762865	papaya	Fruits
1730	44	56	49.0	39.233425	91.255893	6.519780	64.447850	papaya	Fruits
1732	50	59	47.0	40.769987	92.092786	6.747976	209.867841	papaya	Fruits
1738	44	57	53.0	42.304958	90.514318	6.931721	74.876786	papaya	Fruits
1742	70	68	55.0	42.846093	94.635482	6.691202	78.809964	papaya	Fruits
1743	59	62	52.0	43.675493	93.108872	6.608668	103.823566	papaya	Fruits

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1744	60	58	51.0	42.072138	92.922031	6.840802	165.741297	papaya	Fruits
1747	34	65	48.0	41.419684	90.038631	6.665025	199.309643	papaya	Fruits
1748	36	54	46.0	42.547440	94.944821	6.662876	214.410385	papaya	Fruits
1750	37	52	47.0	43.080227	93.903057	6.542777	211.852906	papaya	Fruits
1752	34	48	48.0	41.042244	91.372581	6.805277	181.527598	papaya	Fruits
1758	40	49	47.0	42.933686	91.175675	6.501521	246.361327	papaya	Fruits
1761	59	62	49.0	43.360515	93.351916	6.941497	114.778071	papaya	Fruits
1766	63	58	50.0	43.037143	94.642890	6.720744	41.585659	papaya	Fruits
1774	70	50	53.0	37.462091	90.449678	6.933810	172.345845	papaya	Fruits
1775	44	47	45.0	38.732189	94.736135	6.579441	218.142147	papaya	Fruits
1777	52	51	53.0	38.382315	93.103786	6.985804	210.273535	papaya	Fruits
1778	35	68	45.0	42.936054	90.094481	6.612430	234.846611	papaya	Fruits
1780	32	55	52.0	37.588997	91.997404	6.967760	159.657739	papaya	Fruits
1786	69	66	49.0	40.004391	90.170158	6.527110	92.118774	papaya	Fruits
1791	56	65	45.0	38.201682	93.973800	6.751299	218.090881	papaya	Fruits
1795	42	59	55.0	40.102077	94.351102	6.979102	149.119999	papaya	Fruits
1796	43	64	47.0	38.589545	91.580765	6.825665	102.270823	papaya	Fruits
1797	35	67	49.0	41.313301	91.150880	6.617067	239.742755	papaya	Fruits

In [34]:
```
df.loc[df["temperature"]<LB]
```

Out[34]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1207	11	122	92.5	12.141907	83.568125	5.647202	69.631220	grapes	Fruits
1208	6	123	92.5	12.756796	81.624974	6.130310	66.778446	grapes	Fruits
1211	27	128	92.5	9.467960	82.293355	5.800243	66.027652	grapes	Fruits
1214	32	128	92.5	8.825675	82.897537	5.536646	67.235765	grapes	Fruits
1216	31	128	92.5	11.021054	80.555572	5.870601	68.239632	grapes	Fruits
1225	24	128	92.5	12.087022	83.593987	5.932029	68.668134	grapes	Fruits
1227	5	126	92.5	12.800004	81.208764	6.417501	67.104394	grapes	Fruits
1234	20	128	92.5	10.898759	80.016394	6.207601	68.694204	grapes	Fruits
1239	20	122	92.5	11.797647	80.863254	6.487370	65.069625	grapes	Fruits
1240	40	126	92.5	11.363009	80.031000	6.116983	71.182894	grapes	Fruits
1250	32	120	92.5	10.380048	83.445181	6.138959	67.391738	grapes	Fruits
1254	21	128	92.5	10.723025	80.021306	6.425420	65.298211	grapes	Fruits
1263	37	128	92.5	11.827682	80.282719	5.510925	74.102251	grapes	Fruits
1264	15	128	92.5	13.285043	83.541938	5.699453	65.800060	grapes	Fruits
1288	37	128	92.5	11.189943	80.808431	6.415556	66.342349	grapes	Fruits
1290	38	128	92.5	13.058097	80.282980	5.757010	70.756336	grapes	Fruits
1291	14	121	92.5	9.724458	83.747656	6.158689	74.464111	grapes	Fruits
1293	32	128	92.5	9.535586	80.731127	5.908724	69.441152	grapes	Fruits
1294	11	124	92.5	13.429886	80.066340	6.361141	71.400430	grapes	Fruits
1295	23	128	92.5	9.851243	80.226317	5.965379	68.428024	grapes	Fruits
1299	35	128	92.5	9.949929	82.551390	5.841138	66.008176	grapes	Fruits
1602	27	13	6.0	13.360506	91.356082	7.335158	111.226688	orange	Fruits
1627	11	14	5.0	11.503229	94.893318	6.946355	115.568378	orange	Fruits
1632	8	16	6.0	12.228162	90.264574	7.106650	108.416171	orange	Fruits
1633	15	14	8.0	10.010813	90.223992	6.220943	119.394106	orange	Fruits
1639	1	17	6.0	10.786898	91.384119	6.819827	117.529345	orange	Fruits
1640	1	30	10.0	11.899257	91.346638	7.291406	103.577147	orange	Fruits
1646	32	18	13.0	13.837728	91.747805	6.044167	107.987322	orange	Fruits
1649	15	9	11.0	11.547857	94.148610	7.907956	108.828917	orange	Fruits
1663	26	11	11.0	13.703192	90.955894	7.609348	106.294488	orange	Fruits

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1665	39	21	9.0	13.208444	94.027694	6.354023	106.269616	orange	Fruits
1671	18	12	8.0	12.590940	91.816688	6.206053	119.391672	orange	Fruits
1672	20	20	10.0	11.866319	93.683946	6.976998	106.060149	orange	Fruits
1673	5	8	5.0	11.033679	92.227068	6.562595	112.771592	orange	Fruits
1677	37	18	12.0	10.270888	90.191477	7.401122	106.695520	orange	Fruits
1680	32	25	9.0	10.356096	93.756520	7.796034	101.145695	orange	Fruits
1684	7	17	10.0	10.164313	91.223210	6.465913	106.362551	orange	Fruits
1699	31	26	9.0	11.698946	93.256389	7.566166	103.200599	orange	Fruits

In [35]:
```
df.loc[df["temperature"]>UB,"temperature"]=UB
df.loc[df["temperature"]>UB]
```

Out[35]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [36]:
```
df.loc[df["temperature"]<LB,"temperature"]=LB
df.loc[df["temperature"]<LB]
```

Out[36]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [37]:
```
Q1=df["humidity"].quantile(0.25)
Q2=df["humidity"].quantile(0.50)
Q3=df["humidity"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
134.47899768374998
15.731725873750001
```

In [38]:
```
df.loc[df["humidity"]>UB]
```

Out[38]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [39]:
```
df.loc[df["humidity"]<LB]
```

Out[39]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
202	39	58	85.0	17.887765	15.405897	5.996932	68.549329	chickpea	Cereals and Pulses
203	22	72	85.0	18.868056	15.658092	6.391174	88.510490	chickpea	Cereals and Pulses
205	32	73	81.0	20.450786	15.403121	5.988993	92.683737	chickpea	Cereals and Pulses
211	43	66	79.0	19.462340	15.225390	7.976608	74.585651	chickpea	Cereals and Pulses
212	58	63	81.0	19.813445	14.697653	6.515500	78.965147	chickpea	Cereals and Pulses
214	27	62	77.0	18.197370	14.710705	6.576416	70.181852	chickpea	Cereals and Pulses
219	31	70	77.0	20.888187	14.323138	6.492546	90.462283	chickpea	Cereals and Pulses
221	25	68	77.0	20.093406	15.112796	7.701446	85.749049	chickpea	Cereals and Pulses
222	31	78	76.0	17.572121	14.999275	8.519976	89.310507	chickpea	Cereals and Pulses
225	22	67	78.0	17.166064	14.424575	6.204091	72.326675	chickpea	Cereals and Pulses
230	28	76	82.0	20.566019	14.258040	6.654425	83.759371	chickpea	Cereals and Pulses
239	54	61	77.0	18.811981	15.216182	6.206582	77.542942	chickpea	Cereals and Pulses
245	35	64	78.0	17.928459	14.273280	7.496645	85.373788	chickpea	Cereals and Pulses
247	27	76	83.0	19.128294	14.922415	6.289614	89.618578	chickpea	Cereals and Pulses
252	24	55	78.0	17.302879	15.154059	6.649196	75.577904	chickpea	Cereals and Pulses
253	29	77	75.0	17.503611	15.480832	7.778592	72.944667	chickpea	Cereals and Pulses
254	20	60	78.0	18.172350	14.700860	6.358740	90.776071	chickpea	Cereals and Pulses
261	51	72	75.0	18.888525	14.994511	7.104225	80.111338	chickpea	Cereals and Pulses
262	57	73	85.0	18.493112	14.721150	7.358100	91.945954	chickpea	Cereals and Pulses

	N	P	K	temperature	humidity	ph	rainfall	label	crop
268	52	56	85.0	20.118745	14.442283	6.817124	88.681686	chickpea	Cereals and Pulses
270	42	74	83.0	19.258256	14.280419	7.545258	65.780420	chickpea	Cereals and Pulses
276	32	71	85.0	20.627675	14.440089	6.403982	92.066303	chickpea	Cereals and Pulses
281	47	79	78.0	17.483954	14.760145	6.609697	65.113656	chickpea	Cereals and Pulses
285	37	78	79.0	19.952648	14.826331	7.786366	88.681031	chickpea	Cereals and Pulses
289	22	60	85.0	18.839291	14.740719	7.811998	94.781896	chickpea	Cereals and Pulses
292	39	76	76.0	19.968375	15.573244	8.135901	69.157591	chickpea	Cereals and Pulses
294	30	65	82.0	20.714244	15.278241	7.103798	76.778887	chickpea	Cereals and Pulses
296	48	65	78.0	17.437327	14.338474	7.861128	73.092670	chickpea	Cereals and Pulses
298	40	58	75.0	18.591908	14.779596	7.168096	89.609825	chickpea	Cereals and Pulses
299	49	69	82.0	18.315615	15.361435	7.263119	81.787105	chickpea	Cereals and Pulses

In [40]:
```
df.loc[df["humidity"]<LB,"humidity"]=LB
df.loc[df["humidity"]<LB]
```

Out[40]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [41]:
```
Q1=df["ph"].quantile(0.25)
Q2=df["ph"].quantile(0.50)
Q3=df["ph"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
8.351567354250005
4.543768066249998
```

In [42]:
```
df.loc[df["ph"]>UB]
```

Out[42]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
209	28	74	81.0	18.012723	18.309681	8.753795	81.985688	chickpea	Cereals and Pulses
210	58	66	79.0	20.993736	19.334704	8.718193	93.552801	chickpea	Cereals and Pulses
213	23	62	85.0	18.974248	19.516122	8.490127	80.710875	chickpea	Cereals and Pulses
222	31	78	76.0	17.572121	15.731726	8.519976	89.310507	chickpea	Cereals and Pulses
232	32	60	83.0	19.691417	19.442254	8.829273	91.760716	chickpea	Cereals and Pulses
233	22	78	76.0	17.848517	19.091729	8.621663	76.324707	chickpea	Cereals and Pulses
240	38	60	76.0	18.650541	17.808524	8.868741	77.927987	chickpea	Cereals and Pulses
241	59	55	79.0	20.367204	16.895743	8.766129	82.254558	chickpea	Cereals and Pulses
242	36	76	75.0	18.381204	16.638052	8.736338	70.520567	chickpea	Cereals and Pulses
246	52	60	79.0	19.453399	18.234907	8.380185	75.631757	chickpea	Cereals and Pulses
287	37	55	82.0	19.455918	18.022359	8.423874	78.449106	chickpea	Cereals and Pulses
288	53	65	76.0	20.191378	16.419983	8.719961	77.337954	chickpea	Cereals and Pulses
295	57	56	78.0	17.341502	18.756263	8.861480	67.954543	chickpea	Cereals and Pulses
502	36	58	25.0	28.660242	59.318912	8.399136	36.926297	mothbeans	Cereals and Pulses
503	4	43	18.0	29.029553	61.093875	8.840656	72.980166	mothbeans	Cereals and Pulses
507	5	35	20.0	28.929526	53.570147	9.679241	66.356341	mothbeans	Cereals and Pulses
514	23	45	21.0	31.465113	51.799394	8.985348	74.443307	mothbeans	Cereals and Pulses
518	7	45	22.0	25.506346	44.830255	9.926212	74.326351	mothbeans	Cereals and Pulses
523	28	48	15.0	25.161254	55.254358	9.254089	40.897328	mothbeans	Cereals and Pulses

	N	P	K	temperature	humidity	ph	rainfall	label	crop
536	6	36	22.0	24.216103	59.792363	8.869533	42.247835	mothbeans	Cereals and Pulses
540	25	51	24.0	25.504242	61.668524	9.392695	65.079815	mothbeans	Cereals and Pulses
545	29	44	20.0	30.041323	63.562230	8.620108	31.831924	mothbeans	Cereals and Pulses
546	25	51	18.0	27.777995	54.821308	9.459493	50.284387	mothbeans	Cereals and Pulses
550	3	58	21.0	25.361405	46.826528	9.160692	55.605232	mothbeans	Cereals and Pulses
553	39	36	22.0	29.343174	60.503209	9.072011	34.033355	mothbeans	Cereals and Pulses
560	22	55	24.0	28.568006	57.306360	8.660780	64.530276	mothbeans	Cereals and Pulses
563	28	57	17.0	30.477577	61.582453	9.416003	61.866339	mothbeans	Cereals and Pulses
568	25	35	20.0	28.902454	43.353657	8.923096	71.900186	mothbeans	Cereals and Pulses
572	3	56	17.0	28.199121	53.505676	8.709292	52.135805	mothbeans	Cereals and Pulses
577	7	40	17.0	31.212394	40.926049	8.532079	53.787700	mothbeans	Cereals and Pulses
583	29	41	21.0	31.493981	62.849169	8.869797	64.568076	mothbeans	Cereals and Pulses
584	20	50	22.0	30.996947	46.426937	9.406888	38.315979	mothbeans	Cereals and Pulses
586	15	54	15.0	29.976043	57.031844	8.354958	44.860529	mothbeans	Cereals and Pulses
587	35	55	22.0	30.888831	52.626968	8.634930	55.519324	mothbeans	Cereals and Pulses
590	35	38	19.0	25.326888	63.181803	9.112772	32.711293	mothbeans	Cereals and Pulses
594	35	52	15.0	28.698413	61.147544	9.935091	65.675918	mothbeans	Cereals and Pulses
595	4	59	22.0	29.337434	49.003231	8.914075	42.440543	mothbeans	Cereals and Pulses
596	22	51	16.0	27.965837	61.349001	8.639586	70.104721	mothbeans	Cereals and Pulses

	N	P	K	temperature	humidity	ph	rainfall	label	crop
597	33	47	17.0	24.868040	48.275320	8.621514	63.918765	mothbeans	Cereals and Pulses

In [43]:
```
df.loc[df["ph"]>UB,"ph"]=UB
```

In [44]:
```
df.loc[df["ph"]>UB]
```

Out[44]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [45]:
```
df.loc[df["ph"]<LB]
```

Out[45]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
500	3	49	18.0	27.910952	64.709306	3.692864	32.678919	mothbeans	Cereals and Pulses
501	22	59	23.0	27.322206	51.278688	4.371746	36.503791	mothbeans	Cereals and Pulses
517	0	55	25.0	28.174894	43.667230	4.524172	45.781728	mothbeans	Cereals and Pulses
521	22	49	22.0	28.234947	61.562052	3.711059	72.666664	mothbeans	Cereals and Pulses
526	8	60	18.0	31.216300	46.018682	3.808429	53.120528	mothbeans	Cereals and Pulses
528	22	43	24.0	25.425170	53.220827	4.523636	46.193746	mothbeans	Cereals and Pulses
529	36	43	24.0	27.094006	43.653054	3.510404	41.537495	mothbeans	Cereals and Pulses
535	11	45	19.0	28.700121	44.359648	3.828031	44.116221	mothbeans	Cereals and Pulses
537	17	57	20.0	28.506779	45.200945	3.793575	66.176146	mothbeans	Cereals and Pulses
538	4	47	20.0	25.979490	64.955854	4.193189	72.192458	mothbeans	Cereals and Pulses
541	36	44	21.0	25.125289	51.331894	4.516154	38.486790	mothbeans	Cereals and Pulses
557	4	46	15.0	31.012749	62.403925	3.504752	63.771924	mothbeans	Cereals and Pulses
561	35	51	17.0	28.799292	49.842134	3.558823	40.855347	mothbeans	Cereals and Pulses
575	27	59	20.0	28.009374	52.609500	4.397699	36.012030	mothbeans	Cereals and Pulses

	N	P	K	temperature	humidity	ph	rainfall	label	crop
582	19	51	25.0	26.804744	48.239914	3.525366	43.878020	mothbeans	Cereals and Pulses
599	16	51	21.0	31.019636	49.976752	3.532009	32.812965	mothbeans	Cereals and Pulses
1128	15	27	28.0	33.803987	46.128661	4.507524	90.825492	mango	Fruits
1195	19	38	26.0	31.484517	48.779263	4.525722	93.172220	mango	Fruits

In [46]:
```
df.loc[df["ph"]<LB,"ph"]=LB
df.loc[df["ph"]<LB]
```

Out[46]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop

In [47]:
```
Q1=df["rainfall"].quantile(0.25)
Q2=df["rainfall"].quantile(0.50)
Q3=df["rainfall"].quantile(0.75)
IQR=Q3-Q1
UB=Q3+(1.5*IQR)
LB=Q1-(1.5*IQR)
print(UB)
print(LB)
```

```
213.84124050000003
-25.022046700000033
```

In [48]:
```
df.loc[df["rainfall"]>UB]
```

Out[48]:

	N	P	K	temperature	humidity	ph	rainfall	label	crop
1	85	58	41.0	21.770462	80.319644	7.038096	226.655537	rice	Cereals and Pulses
2	60	55	44.0	23.004459	82.320763	7.840207	263.964248	rice	Cereals and Pulses
3	74	35	40.0	26.491096	80.158363	6.980401	242.864034	rice	Cereals and Pulses
4	78	42	42.0	20.130175	81.604873	7.628473	262.717340	rice	Cereals and Pulses
5	69	37	42.0	23.058049	83.370118	7.073454	251.055000	rice	Cereals and Pulses
...	...	...	...	...	...	...	...	...	...
1867	3	23	30.0	29.701432	95.657544	6.078807	215.196804	coconut	Fibres and Beverages
1881	19	30	30.0	29.565492	91.408963	5.826381	224.831573	coconut	Fibres and Beverages
1886	8	15	33.0	28.973187	98.098610	5.501580	213.901102	coconut	Fibres and Beverages
1892	3	9	35.0	26.916419	99.846716	6.318553	225.632366	coconut	Fibres and Beverages
1894	27	8	30.0	26.446001	98.299378	6.008386	221.225817	coconut	Fibres and Beverages

100 rows × 9 columns

In [49]:
```
df.loc[df["rainfall"]>UB,"rainfall"]=UB
df.loc[df["rainfall"]>UB]
```

Out[49]:

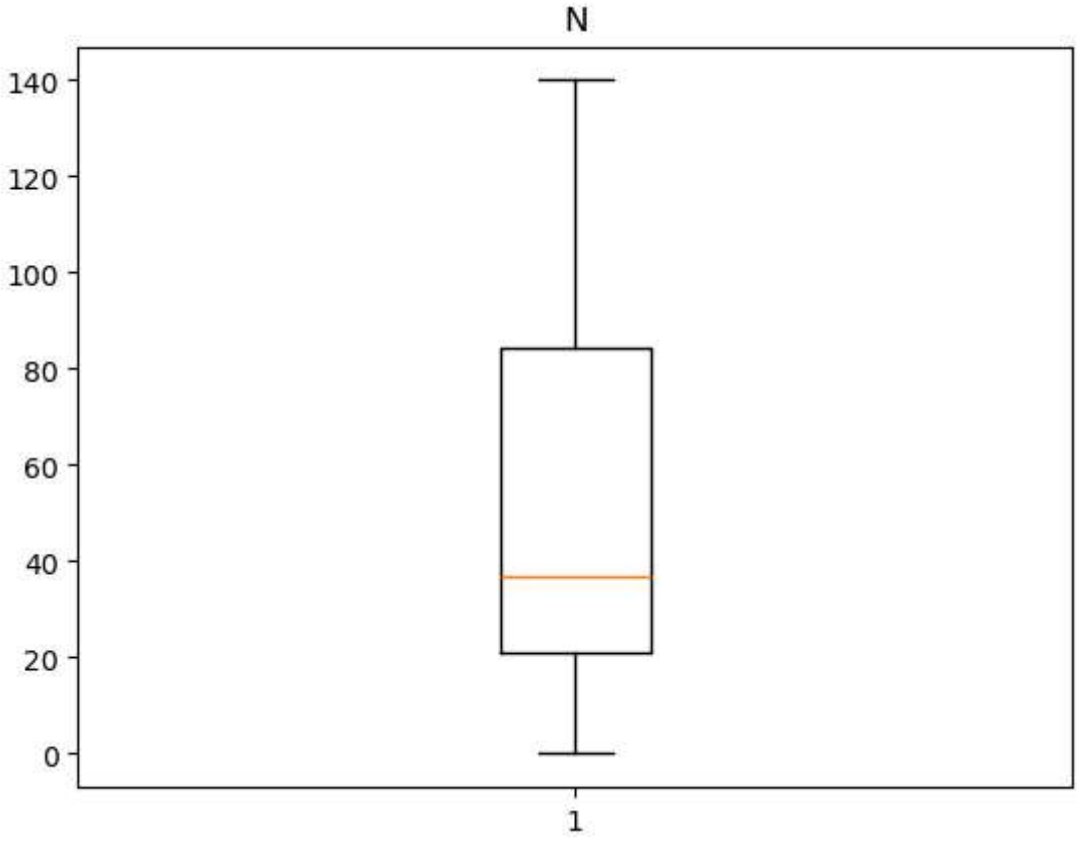
	N	P	K	temperature	humidity	ph	rainfall	label	crop

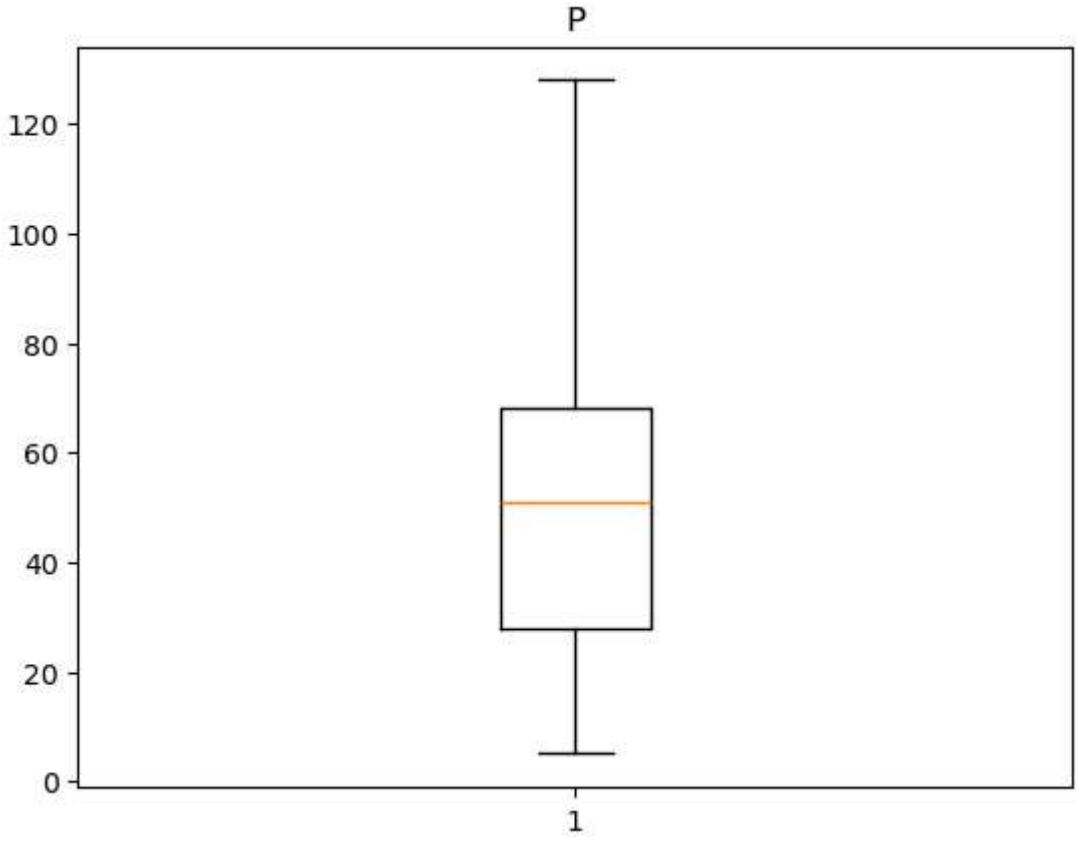
In [50]:
```
df.loc[df["rainfall"]<LB]
```

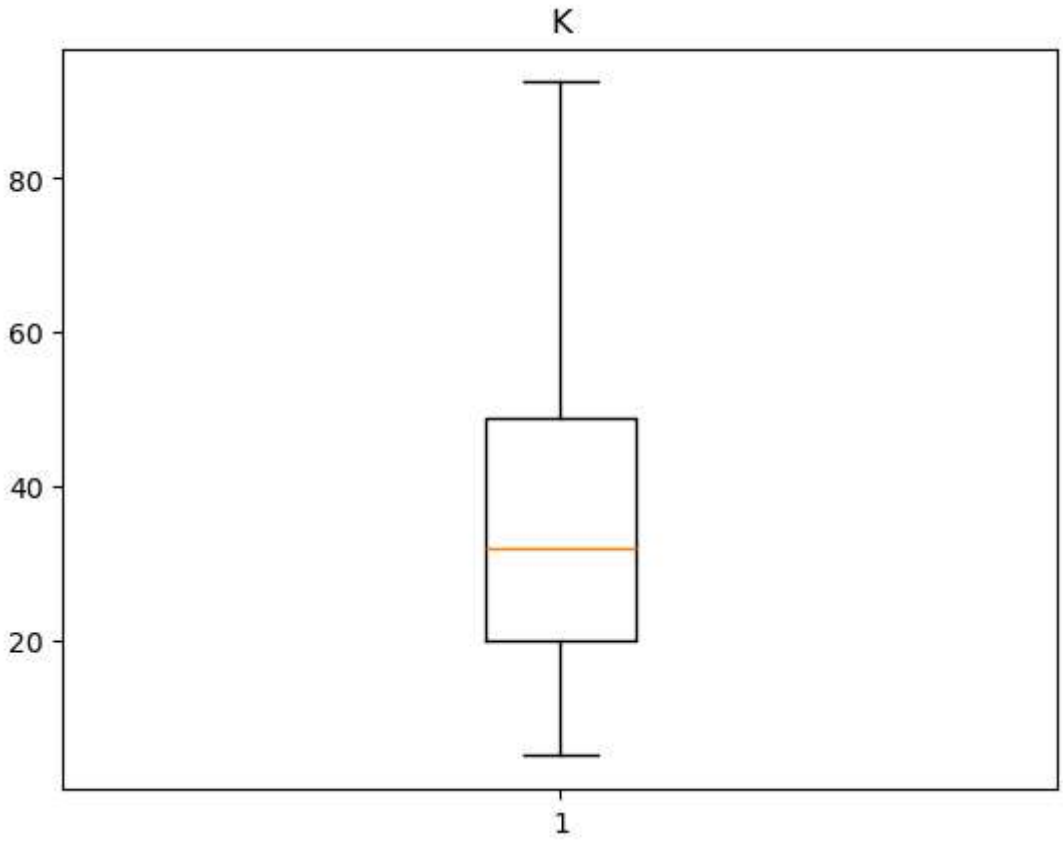
Out[50]:

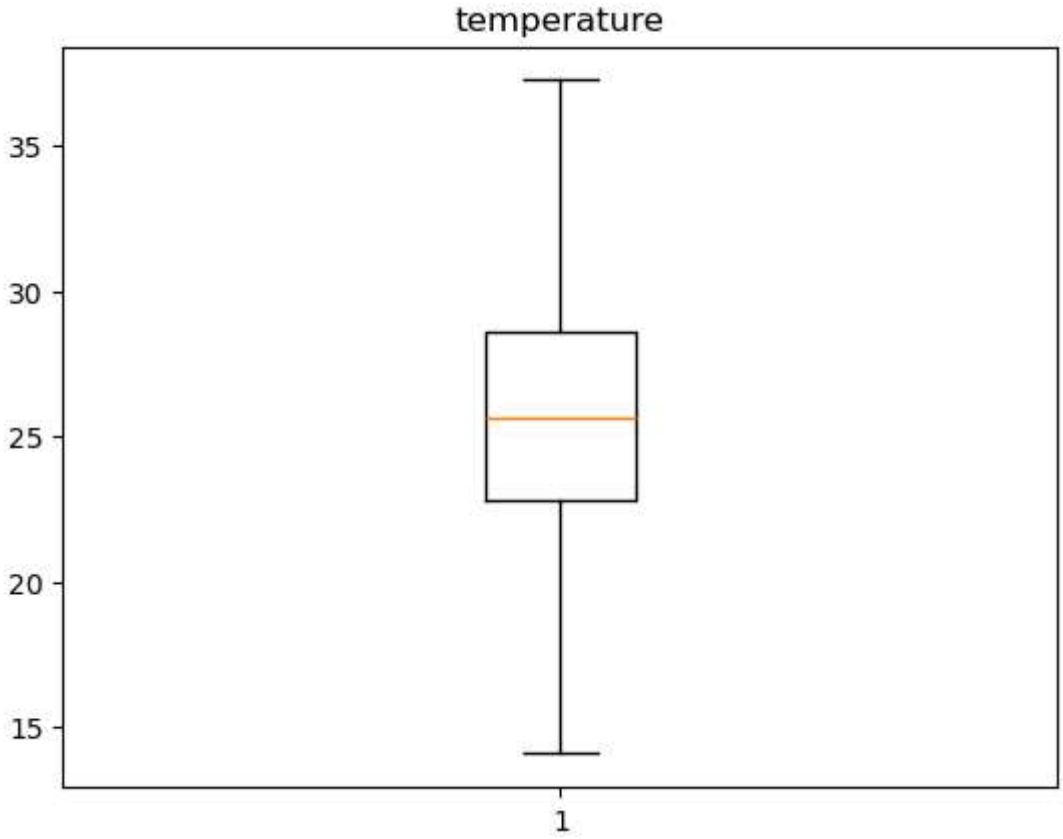
	N	P	K	temperature	humidity	ph	rainfall	label	crop

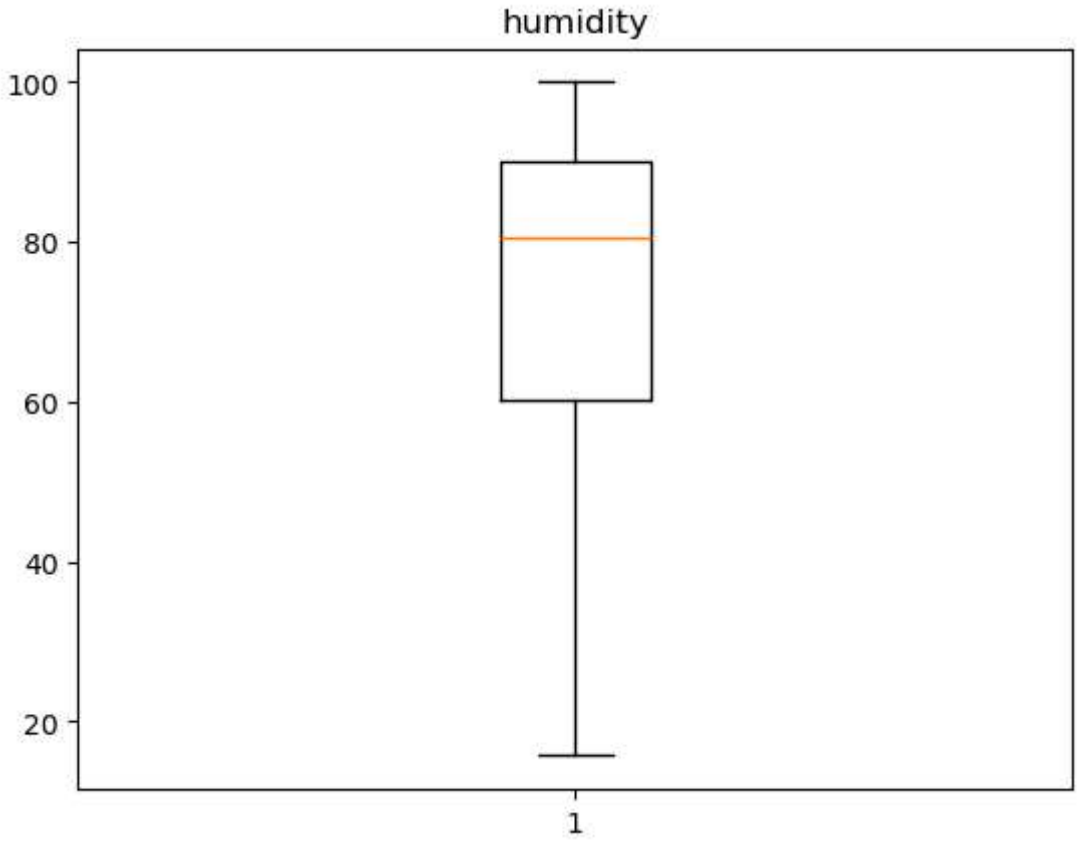
In [51]:
```
for i in cols:
    plt.boxplot(df[i])
    plt.title(i)
    plt.show()
```

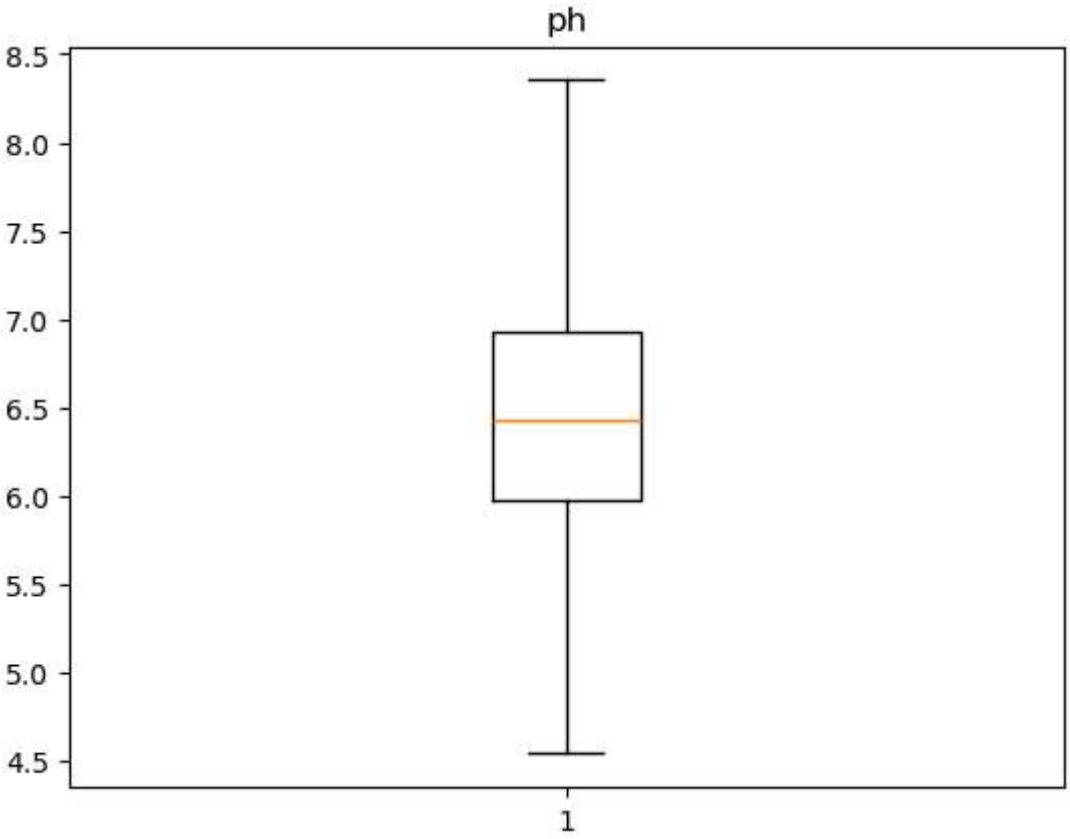

N
140
120
100
80
60
40
20
0
1

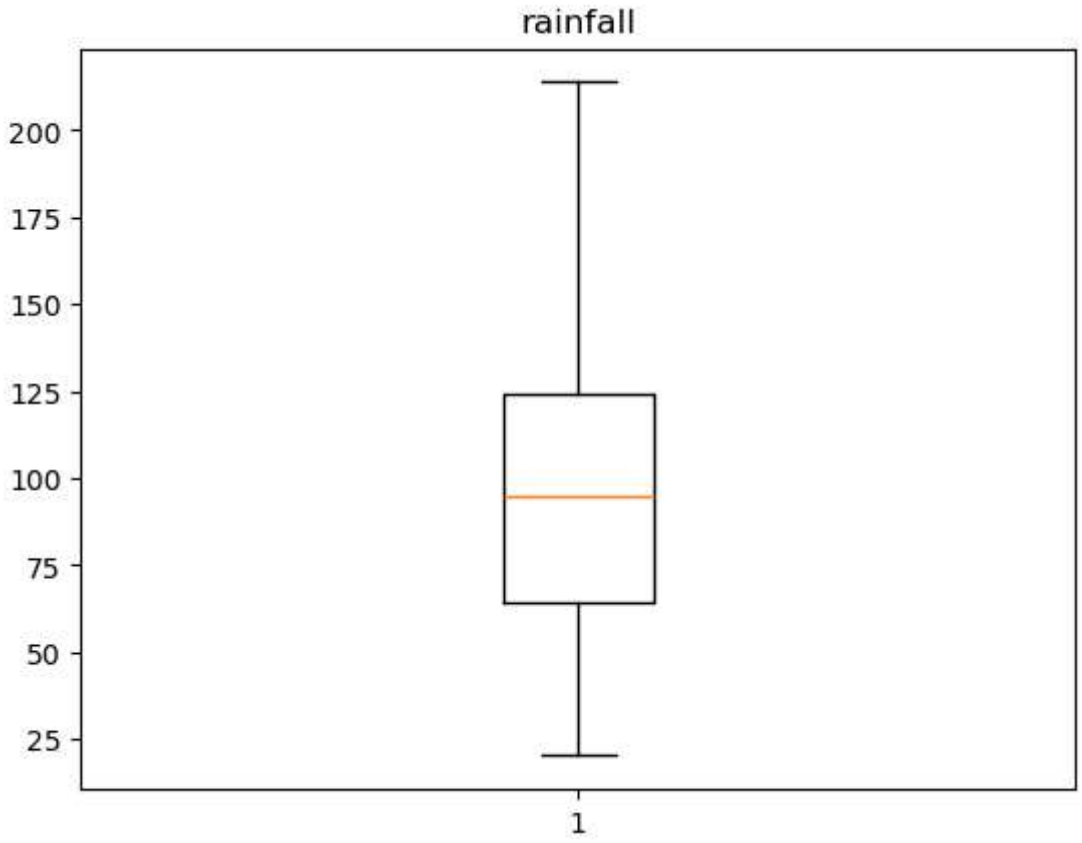

P
120
100
80
60
40
20
0
1


K
80
60
40
20
1


temperature
35
30
25
20
15
1


humidity
100
80
60
40
20
1


ph
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
1



In [52]:
```
df.drop("label",inplace=True,axis=1)
```

In [53]:
```
from scipy.stats import skew
```

In [54]:
```
df1=df.drop("crop",axis=1)
```

In [55]:
```
for i in df1:
    print(i,"----",skew(df1[i]))
```

```
N ---- 0.5093737660143558
P ---- 0.8360389538152234
K ---- 1.1158786974444166
temperature ---- 0.07073009768055255
humidity ---- -1.086833464634107
ph ---- 0.14637087895525935
rainfall ---- 0.6630549439462043
```

In [56]:
```
from sklearn.preprocessing import LabelEncoder
```

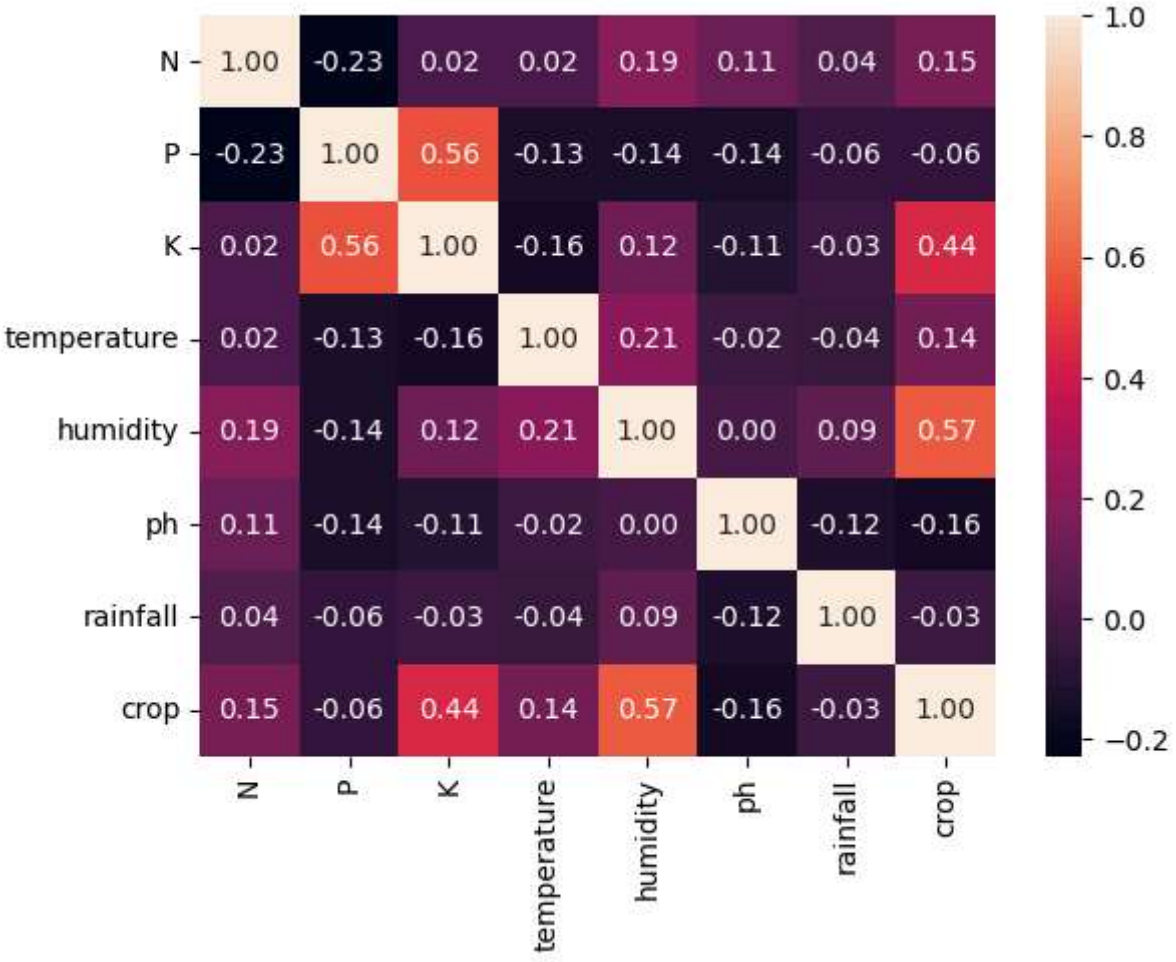
In [57]:
```
LE=LabelEncoder()
```

In [58]:
```
df["crop"]=LE.fit_transform(df["crop"])
```

In [59]:
```
df["crop"].unique()
```

Out[59]:
```
array([0, 2, 1])
```

In [60]:
```
sns.heatmap(df.corr(),fmt='.2f',annot=True)
plt.show()
```



In [61]:
```
from ydata_profiling import ProfileReport
```

In [62]:
```
report=ProfileReport(df,explorative=True)
report
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Pandas Profiling Report

Overview

Brought to you by YData

Overview | Alerts 5 | Reproduction

Dataset statistics

Number of variables	8
Number of observations	2200
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	129.0 KiB
Average record size in memory	60.1 B

Variable types

Numeric	7
Categorical	1

Variables

Out[62]:

In [63]:
```
x=df.drop("crop",axis=1)
y=df["crop"]
```

In [64]:
```
x
```

Out[64]:

	N	P	K	temperature	humidity	ph	rainfall
0	90	42	43.0	20.879744	82.002744	6.502985	202.935536
1	85	58	41.0	21.770462	80.319644	7.038096	213.841241
2	60	55	44.0	23.004459	82.320763	7.840207	213.841241
3	74	35	40.0	26.491096	80.158363	6.980401	213.841241
4	78	42	42.0	20.130175	81.604873	7.628473	213.841241
...	...	...	...	...	...	...	...
2195	107	34	32.0	26.774637	66.413269	6.780064	177.774507
2196	99	15	27.0	27.417112	56.636362	6.086922	127.924610
2197	118	33	30.0	24.131797	67.225123	6.362608	173.322839
2198	117	32	34.0	26.272418	52.127394	6.758793	127.175293
2199	104	18	30.0	23.603016	60.396475	6.779833	140.937041

2200 rows × 7 columns

In [65]:
```
y
```

Out[65]:
```
0       0
1       0
2       0
3       0
4       0
       ..
2195    1
2196    1
2197    1
2198    1
2199    1
Name: crop, Length: 2200, dtype: int32
```

In [66]:
```
from sklearn.preprocessing import StandardScaler
```

In [67]:
```
SS=StandardScaler()
```

In [68]:
```
for features in x:
    x[features]=SS.fit_transform(x[[features]])
```

In [69]:
```
x
```

Out[69]:

	N	P	K	temperature	humidity	ph	rainfall
0	1.068797	-0.341922	0.197128	-0.993936	0.472768	0.053115	1.973430
1	0.933329	0.167308	0.111858	-0.805930	0.397054	0.786021	2.186623
2	0.255986	0.071827	0.239763	-0.545469	0.487075	1.884619	2.186623
3	0.635298	-0.564710	0.069223	0.190462	0.389798	0.707000	2.186623
4	0.743673	-0.341922	0.154493	-1.152149	0.454870	1.594621	2.186623
...	...	...	...	...	...	...	...
2195	1.529390	-0.596536	-0.271855	0.250309	-0.228529	0.432611	1.481563
2196	1.312641	-1.201247	-0.485030	0.385918	-0.668346	-0.516738	0.507059
2197	1.827421	-0.628363	-0.357125	-0.307519	-0.192007	-0.139150	1.394538
2198	1.800327	-0.660190	-0.186586	0.144305	-0.871183	0.403477	0.492411
2199	1.448109	-1.105766	-0.357125	-0.419130	-0.499196	0.432295	0.761436

2200 rows × 7 columns

In [70]:
```
from sklearn.model_selection import train_test_split
```

In [71]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=7)
```

In [144…
```
from sklearn.linear_model import LogisticRegression
```

In [146…
```
model=LogisticRegression()
```

In [152…
```
model.fit(x_train,y_train)
```

Out[152…

▾ LogisticRegression

```
LogisticRegression()
```

In [154…
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,cl
```

In [158…
```
y_pred_train=model.predict(x_train)

clf_report=classification_report(y_pred_train,y_train)
print(clf_report)
print(50*"*")

cm=confusion_matrix(y_pred_train,y_train)
print(cm)
print(50*"*")

accuracy=accuracy_score(y_pred_train,y_train)
print(accuracy*100)
```

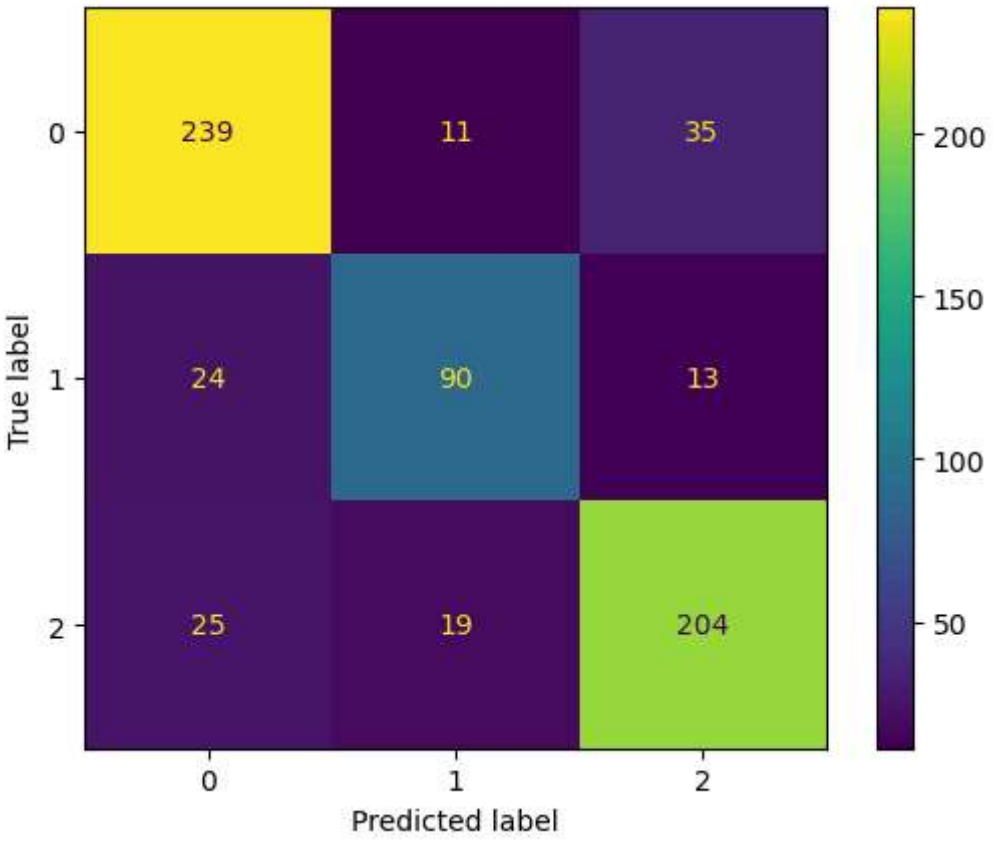
```
              precision    recall  f1-score   support

           0       0.81      0.84      0.83       594
           1       0.75      0.69      0.72       302
           2       0.84      0.85      0.85       644

    accuracy                           0.81      1540
   macro avg       0.80      0.79      0.80      1540
weighted avg       0.81      0.81      0.81      1540

**************************************************
[[498  27  69]
 [ 61 209  32]
 [ 53  44 547]]
**************************************************
81.42857142857143
```

In [160…
```
y_pred_test=model.predict(x_test)

clf_report=classification_report(y_pred_test,y_test)
print(clf_report)
print(50*"*")

cm=confusion_matrix(y_pred_test,y_test)
print(cm)
print(50*"*")

accuracy=accuracy_score(y_pred_test,y_test)
print(accuracy*100)
```

```
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       285
           1       0.75      0.71      0.73       127
           2       0.81      0.82      0.82       248

    accuracy                           0.81       660
   macro avg       0.80      0.79      0.79       660
weighted avg       0.81      0.81      0.81       660

**************************************************
[[239  11  35]
 [ 24  90  13]
 [ 25  19 204]]
**************************************************
80.75757575757576
```

In [162…
```
from sklearn.metrics import ConfusionMatrixDisplay
```

In [164…
```
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot()
plt.show()
```



In [166…
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

In [168…
```
DTC=DecisionTreeClassifier()
```

In [170…
```
DTC.fit(x_train,y_train)
```

Out[170…
DecisionTreeClassifier

DecisionTreeClassifier()

In [172…
```
y_pred_train=DTC.predict(x_train)

clf_report=classification_report(y_pred_train,y_train)
print(clf_report)
print(50*"*")

cm=confusion_matrix(y_pred_train,y_train)
print(cm)
print(50*"*")

accuracy=accuracy_score(y_pred_train,y_train)
print(accuracy*100)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       612
           1       1.00      1.00      1.00       280
           2       1.00      1.00      1.00       648

    accuracy                           1.00      1540
   macro avg       1.00      1.00      1.00      1540
weighted avg       1.00      1.00      1.00      1540

**************************************************
[[612   0   0]
 [  0 280   0]
 [  0   0 648]]
**************************************************
100.0
```

In [88]:
```
y_pred_test=DTC.predict(x_test)

clf_report=classification_report(y_pred_test,y_test)
print(clf_report)
print(50*"*")

cm=confusion_matrix(y_pred_test,y_test)
print(cm)
print(50*"*")

accuracy=accuracy_score(y_pred_test,y_test)
print(accuracy*100)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       285
           1       0.96      0.95      0.95       121
           2       1.00      0.99      1.00       254

    accuracy                           0.98       660
   macro avg       0.98      0.98      0.98       660
weighted avg       0.98      0.98      0.98       660

**************************************************
[[282   3   0]
 [  6 115   0]
 [  0   2 252]]
**************************************************
98.33333333333333
```

In [174…
```
array=DTC.feature_importances_
array
```

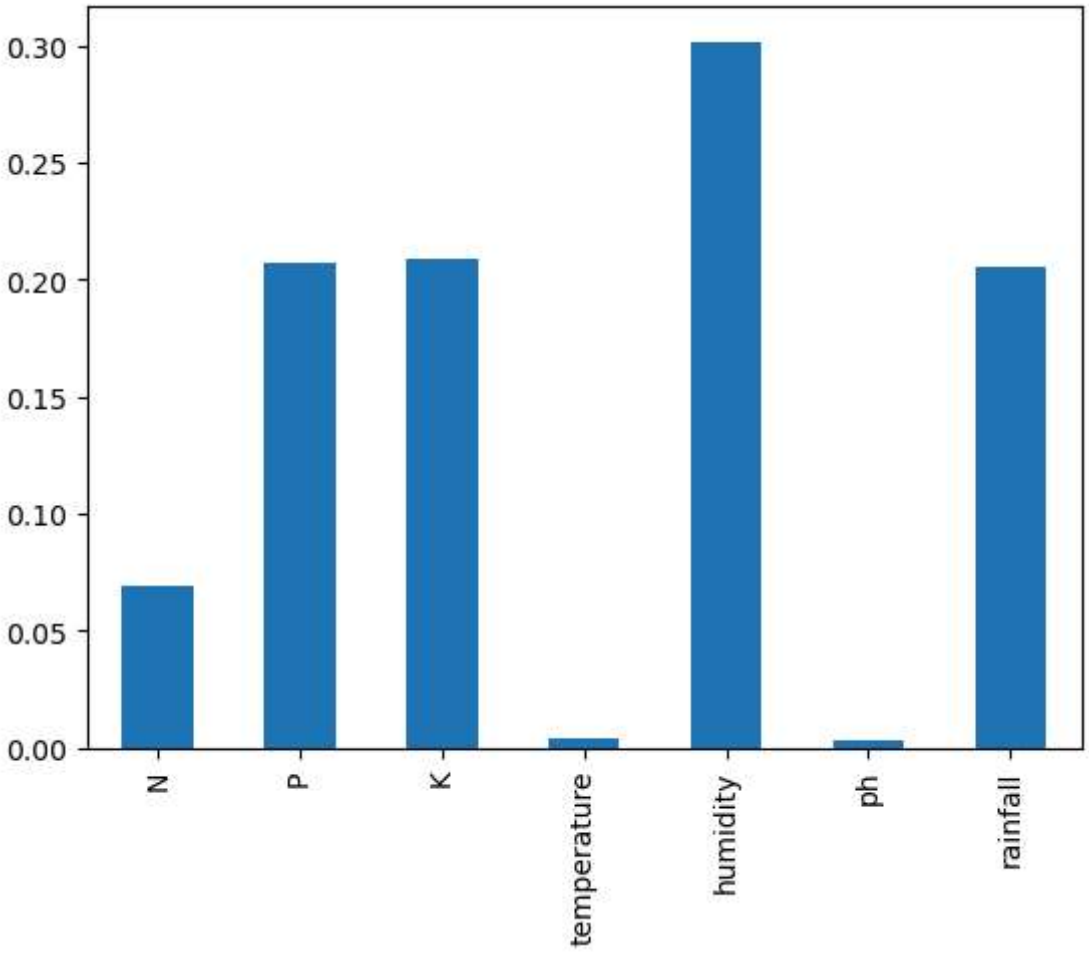
Out[174…
```
array([0.06937446, 0.20747696, 0.20875755, 0.00378673, 0.30154191,
       0.00359633, 0.20546607])
```

In [176…
```
imp_features=pd.Series(array,index=x.columns)
imp_features
```

Out[176…
```
N              0.069374
P              0.207477
K              0.208758
temperature    0.003787
humidity       0.301542
ph             0.003596
rainfall       0.205466
dtype: float64
```

In [178…
```
imp_features.plot(kind="bar")
```

Out[178…
```
<Axes: >
```



In [180…
```
from sklearn.model_selection import GridSearchCV
```

In [182…
```
hyperparameter={"criterion":["gini","entropy"],
                "max_depth":np.arange(3,9),
                "min_samples_split":np.arange(2,20),
               "min_samples_leaf":np.arange(2,15)}
```

In [184…
```
gscv=GridSearchCV(DTC,hyperparameter,cv=5)
```

In [186…
```
gscv.fit(x_train,y_train)
```

Out[186…

GridSearchCV

estimator: DecisionTreeClassifier

DecisionTreeClassifier

In [96]:
```
gscv.best_estimator_
```

Out[96]:

DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_leaf=
2,
                       min_samples_split=5)
```

In [190…
```
new_DTC=gscv.best_estimator_
```

In [192…
```
new_DTC.fit(x_train,y_train)
```

Out[192…

DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_leaf=
3,
                       min_samples_split=3)
```

In [194…
```
y_pred_train=new_DTC.predict(x_train)

cnf=confusion_matrix(y_pred_train,y_train)
print("confusion_matrix",cnf)
print("*"*60)

clf_report=classification_report(y_pred_train,y_train)
print("classification_report",clf_report)
print("*"*60)

accuracy=accuracy_score(y_pred_train,y_train)
print("accuracy_score",accuracy)
```

```
confusion_matrix [[606   0   0]
 [  5 280   0]
 [  1   0 648]]
************************************************************
classification_report               precision    recall  f1-score   support

           0       0.99      1.00      1.00       606
           1       1.00      0.98      0.99       285
           2       1.00      1.00      1.00       649

    accuracy                           1.00      1540
   macro avg       1.00      0.99      1.00      1540
weighted avg       1.00      1.00      1.00      1540

************************************************************
accuracy_score 0.9961038961038962
```

In [196…

```
y_pred_test=new_DTC.predict(x_test)

cnf=confusion_matrix(y_pred_test,y_test)
print("confusion_matrix",cnf)
print("*"*60)

clf_report=classification_report(y_pred_test,y_test)
print("classification_report",clf_report)
print("*"*60)

accuracy=accuracy_score(y_pred_test,y_test)
print("accuracy_score",accuracy)
```

```
confusion_matrix [[282   0   1]
 [  6 120   0]
 [  0   0 251]]
************************************************************
classification_report               precision    recall  f1-score   support

           0       0.98      1.00      0.99       283
           1       1.00      0.95      0.98       126
           2       1.00      1.00      1.00       251

    accuracy                           0.99       660
   macro avg       0.99      0.98      0.99       660
weighted avg       0.99      0.99      0.99       660

************************************************************
accuracy_score 0.9893939393939394
```

In [ ]:

In [ ]: