In [5]:
```
1 pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist
-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
st-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
10/dist-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
n3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
ackages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [7]:
```
1 pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-pack
ages (1.25.2)
```

In [6]:

```
pip install seaborn

```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-pa
ckages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/pytho
n3.10/dist-packages (from seaborn) (1.25.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dis
t-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/py
thon3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/di
st-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
n3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
st-packages (from pandas>=1.2->seaborn) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/
dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-p
ackages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.1
6.0)
```

In [8]:

```
# importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:

```python
# loading the dataset
crop_data=pd.read_csv("Crop_recommendation.csv")
crop_data
```

Out[11]:

	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 [12]:

```python
#rows and columns
crop_data.shape
```

Out[12]:

```
(2200, 8)
```

In [13]:

```python
#checking basic information against columns
crop_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
```

There is no null data rows so we don't need to replace it using mean values or drop columns.

In [ ]:

```python
# dataset columns
crop_data.columns
```

Out[5]:

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'labe
l'], dtype='object')
```

In [ ]:
```python
#Changing the name of label to Crop for readability
crop_data.rename(columns = {'label':'Crop'}, inplace = True)
crop_data
```

Out[6]:

	N	P	K	temperature	humidity	ph	rainfall	Crop
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 [ ]:
```python
# statistical inference of the dataset
crop_data.describe()
```

Out[7]:

	N	P	K	temperature	humidity	ph	ra
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.00
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.46
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.95
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.2
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.55
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.86
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.26
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.56

In [ ]:
```python
#Checking missing values of the dataset in each column
crop_data.isnull().sum()
```

Out[8]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
Crop           0
dtype: int64
```

In [14]:

```
#Dropping missing values
crop_data = crop_data.dropna()
crop_data
```

Out[14]:

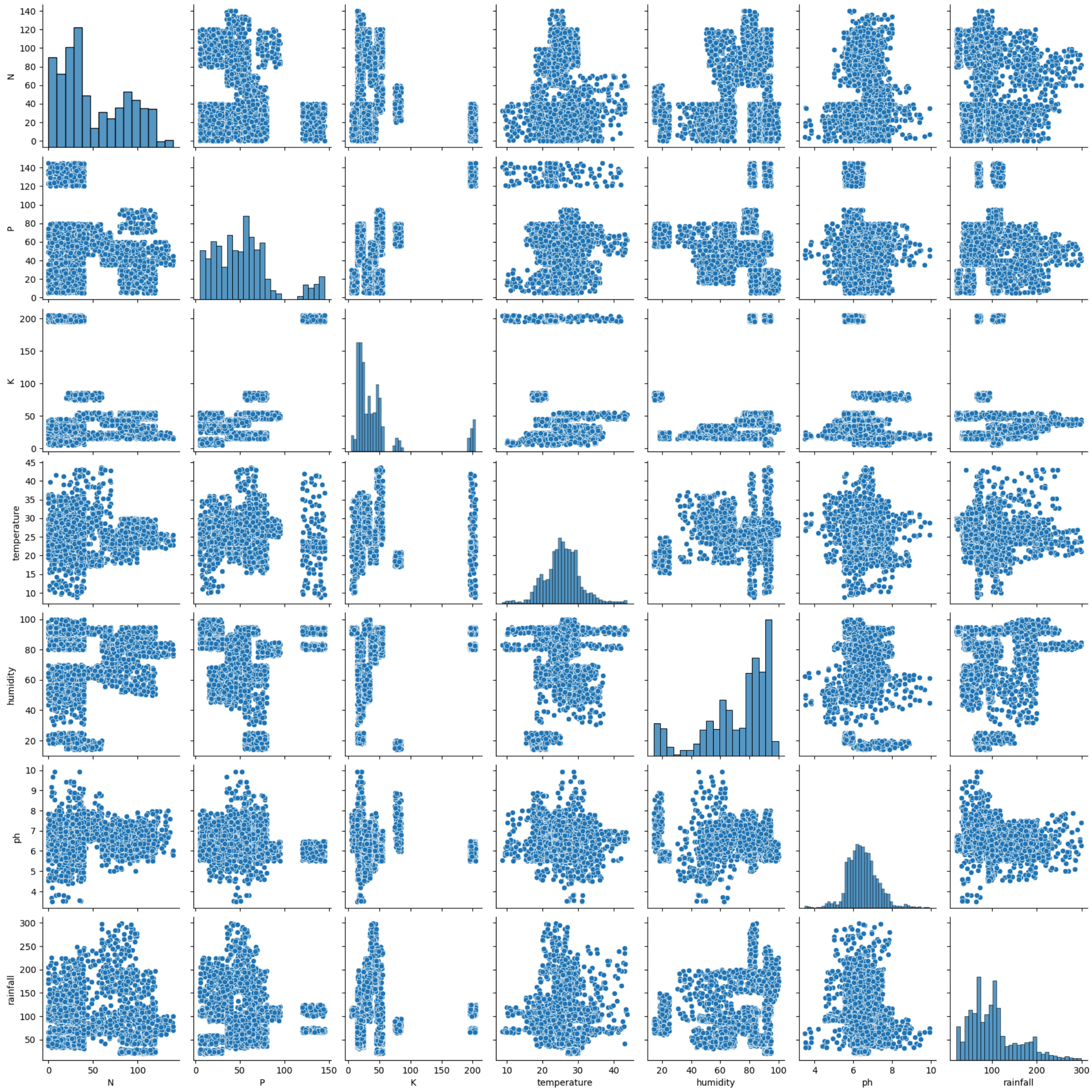
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	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 [15]:

```
# Visualizing the features
ax = sns.pairplot(crop_data)
ax
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x7a437ef757b0>



In [ ]:

```
crop_data.Crop.unique()
```

Out[11]:

```
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 [ ]:

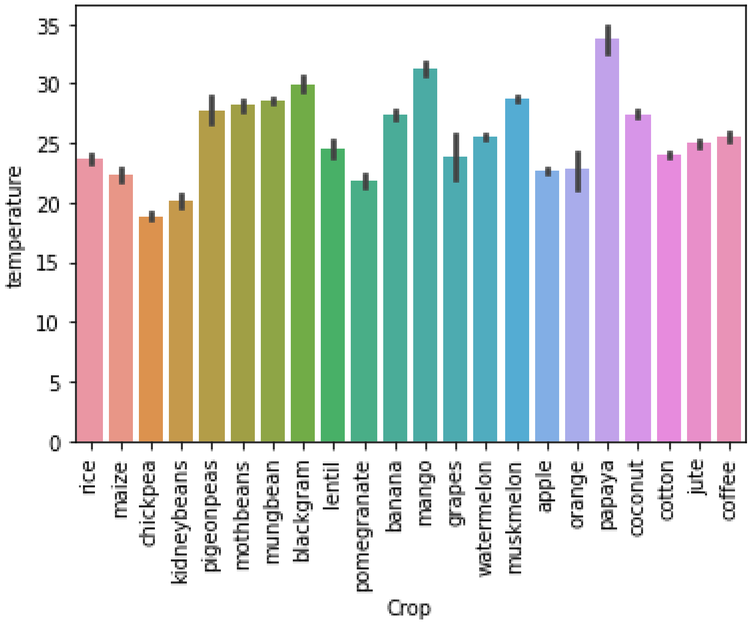
```
# get top 5 most frequent growing crops
n = 5
crop_data['Crop'].value_counts()[:5].index.tolist()
```

Out[12]: ['rice', 'maize', 'jute', 'cotton', 'coconut']

In [ ]:
```
sns.barplot(crop_data["Crop"], crop_data["temperature"])
plt.xticks(rotation = 90)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[13]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text major ticklabel objects>)
```



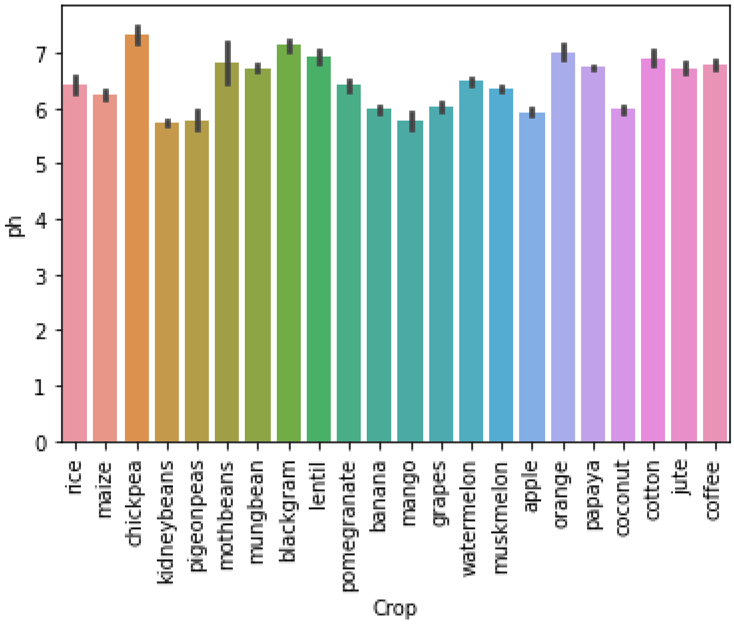
In [ ]:

```
sns.barplot(crop_data["Crop"], crop_data["ph"])
plt.xticks(rotation = 90)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpreta
tion.
  FutureWarning
```

Out[14]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text major ticklabel objects>)
```



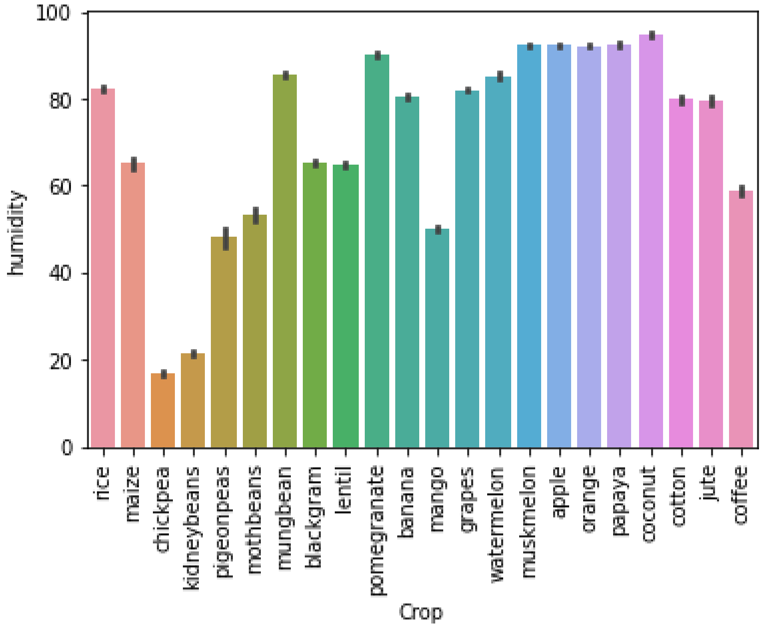
In [ ]:

```
sns.barplot(crop_data["Crop"], crop_data["humidity"])
plt.xticks(rotation = 90)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpreta
tion.
  FutureWarning
```

Out[15]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text major ticklabel objects>)
```



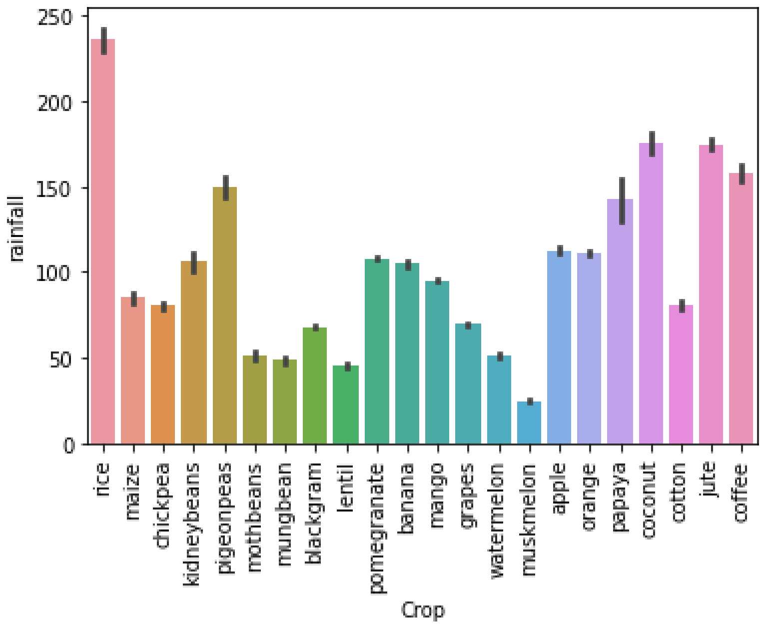
In [ ]:

```
sns.barplot(crop_data["Crop"], crop_data["rainfall"])
plt.xticks(rotation = 90)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpreta
tion.
  FutureWarning
```

Out[16]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text major ticklabel objects>)
```



In [ ]:

```
crop_data.corr()
```

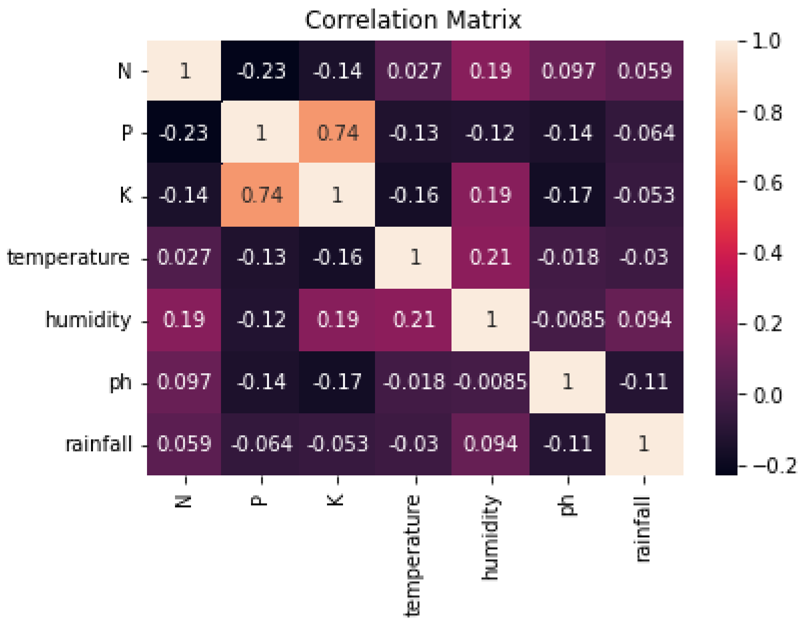
Out[17]:

	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.231460	-0.140512	0.026504	0.190688	0.096683	0.059020
P	-0.231460	1.000000	0.736232	-0.127541	-0.118734	-0.138019	-0.063839
K	-0.140512	0.736232	1.000000	-0.160387	0.190859	-0.169503	-0.053461
temperature	0.026504	-0.127541	-0.160387	1.000000	0.205320	-0.017795	-0.030084
humidity	0.190688	-0.118734	0.190859	0.205320	1.000000	-0.008483	0.094423
ph	0.096683	-0.138019	-0.169503	-0.017795	-0.008483	1.000000	-0.109069
rainfall	0.059020	-0.063839	-0.053461	-0.030084	0.094423	-0.109069	1.000000

In [ ]:

```python
sns.heatmap(crop_data.corr(), annot =True)
plt.title('Correlation Matrix')
```

Out[18]: Text(0.5, 1.0, 'Correlation Matrix')



In [ ]:

```python
# shuffling the dataset to remove order
from sklearn.utils import shuffle

df  = shuffle(crop_data,random_state=5)
df.head()
```

Out[19]:

	N	P	K	temperature	humidity	ph	rainfall	Crop
1270	6	140	205	17.665584	82.929034	6.313086	69.867126	grapes
1481	98	22	47	29.072653	91.915332	6.341401	28.835684	muskmelon
1832	38	14	30	26.924495	91.201060	5.570745	194.902214	coconut
293	35	63	76	17.815645	17.607566	7.714153	90.820976	chickpea
1307	85	22	53	25.965342	89.770767	6.849472	59.463386	watermelon

In [ ]:

```python
# Selection of Feature and Target variables.
x = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['Crop']
```

In [ ]:

```
# Encoding target variable
y = pd.get_dummies(target)
y
```

Out[21]:

	apple	banana	blackgram	chickpea	coconut	coffee	cotton	grapes	jute	kidneybean
1270	0	0	0	0	0	0	0	1	0	
1481	0	0	0	0	0	0	0	0	0	
1832	0	0	0	0	1	0	0	0	0	
293	0	0	0	1	0	0	0	0	0	
1307	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	.
740	0	0	1	0	0	0	0	0	0	
1032	0	1	0	0	0	0	0	0	0	
2121	0	0	0	0	0	1	0	0	0	
1424	0	0	0	0	0	0	0	0	0	
1725	0	0	0	0	0	0	0	0	0	

2200 rows × 22 columns

In [ ]:

```
# Splitting data set - 25% test dataset and 75%
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, ran

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

```
x_train : (1650, 7)
x_test : (550, 7)
y_train : (1650, 22)
y_test : (550, 22)
```

In [ ]:

```
# Importing necessary libraries for multi-output classification

from sklearn.datasets import make_classification
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

Naive Bayes Classification

In [ ]:

```
gnb = GaussianNB()
model = MultiOutputClassifier(gnb, n_jobs=-1)
model.fit(x_train, y_train)
```

Out[24]:

```
MultiOutputClassifier(estimator=GaussianNB(), n_jobs=-1)
```

In [ ]:

```
gnb_pred = model.predict(x_test)
gnb_pred
```

Out[25]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [ ]:

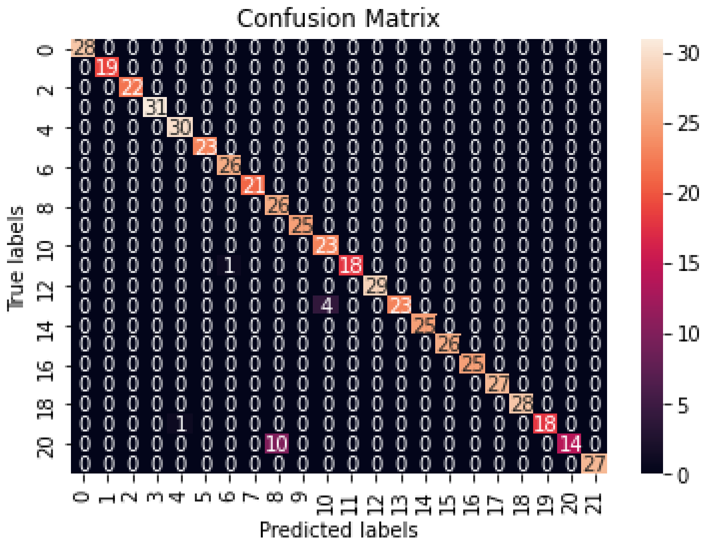
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a1 = accuracy_score(y_test.values.argmax(axis=1), gnb_pred.argmax(axis=1
a1
```

Out[26]:

```
0.9709090909090909
```

In [ ]:

```
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), gnb_pred.argmax(axis=1
#cm = confusion_matrix(y_test, gnb_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```



In [ ]:

```python
from sklearn import metrics
# Print the confusion matrix
print(metrics.confusion_matrix(y_test.values.argmax(axis=1), gnb_pred.ar

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), gnb_pr
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0 18  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  4  0  0 23  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 28  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 18  0  0]
 [ 0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27]]
              precision    recall  f1-score   support

           0      1.000     1.000     1.000        28
           1      1.000     1.000     1.000        19
           2      1.000     1.000     1.000        22
           3      1.000     1.000     1.000        31
           4      0.968     1.000     0.984        30
           5      1.000     1.000     1.000        23
           6      0.963     1.000     0.981        26
           7      1.000     1.000     1.000        21
           8      0.722     1.000     0.839        26
           9      1.000     1.000     1.000        25
          10      0.852     1.000     0.920        23
          11      1.000     0.947     0.973        19
          12      1.000     1.000     1.000        29
          13      1.000     0.852     0.920        27
          14      1.000     1.000     1.000        25
          15      1.000     1.000     1.000        26
          16      1.000     1.000     1.000        25
          17      1.000     1.000     1.000        27
          18      1.000     1.000     1.000        28
          19      1.000     0.947     0.973        19
          20      1.000     0.583     0.737        24
          21      1.000     1.000     1.000        27

    accuracy                          0.971       550
   macro avg      0.977     0.970     0.969       550
weighted avg      0.977     0.971     0.970       550
```

In [ ]:

```
 
```

Decision Tree Classification

In [ ]:

```
# Training
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=6)
multi_target_decision = MultiOutputClassifier(clf, n_jobs=-1)
multi_target_decision.fit(x_train, y_train)
```

Out[66]:

```
MultiOutputClassifier(estimator=DecisionTreeClassifier(random_state=6),
                      n_jobs=-1)
```

In [ ]:

```
# Predicting test results
decision_pred = multi_target_decision.predict(x_test)
decision_pred
```

Out[67]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [ ]:

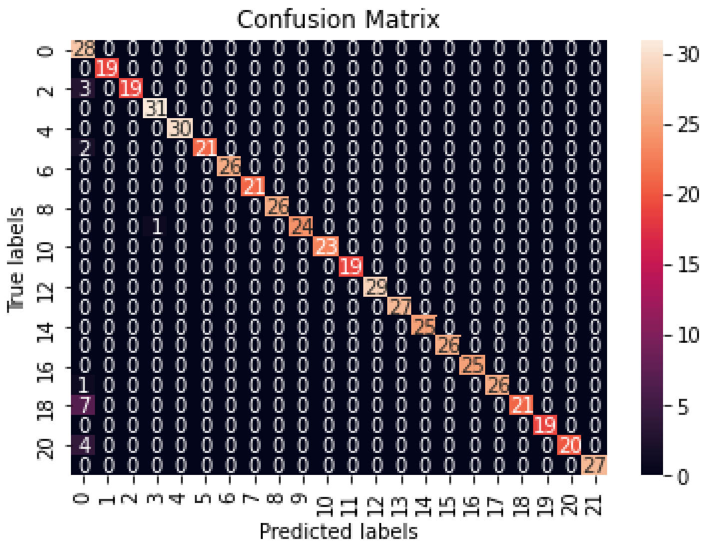
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a2 = accuracy_score(y_test.values.argmax(axis=1), decision_pred.argmax(a
a2
```

Out[68]:

```
0.9672727272727273
```

In [ ]:

```
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), decision_pred.argmax(a
#cm = confusion_matrix(y_test, gnb_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```



In [ ]:

```python
from sklearn import metrics
# Print the confusion matrix
print(metrics.confusion_matrix(y_test.values.argmax(axis=1), decision_pr
 
# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), decisi
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 3  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0 24  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0]
 [ 7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 21  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0]
 [ 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27]]
              precision    recall  f1-score   support

           0      0.622     1.000     0.767        28
           1      1.000     1.000     1.000        19
           2      1.000     0.864     0.927        22
           3      0.969     1.000     0.984        31
           4      1.000     1.000     1.000        30
           5      1.000     0.913     0.955        23
           6      1.000     1.000     1.000        26
           7      1.000     1.000     1.000        21
           8      1.000     1.000     1.000        26
           9      1.000     0.960     0.980        25
          10      1.000     1.000     1.000        23
          11      1.000     1.000     1.000        19
          12      1.000     1.000     1.000        29
          13      1.000     1.000     1.000        27
          14      1.000     1.000     1.000        25
          15      1.000     1.000     1.000        26
          16      1.000     1.000     1.000        25
          17      1.000     0.963     0.981        27
          18      1.000     0.750     0.857        28
          19      1.000     1.000     1.000        19
          20      1.000     0.833     0.909        24
          21      1.000     1.000     1.000        27

    accuracy                          0.967       550
   macro avg      0.981     0.967     0.971       550
weighted avg      0.979     0.967     0.969       550
```

In [ ]:

```

```

Random Forest Classification

In [ ]:

```python
# Training
forest = RandomForestClassifier(random_state=1)
multi_target_forest = MultiOutputClassifier(forest, n_jobs=-1)
multi_target_forest.fit(x_train, y_train)
```

Out[56]:

```
MultiOutputClassifier(estimator=RandomForestClassifier(random_state=1),
                      n_jobs=-1)
```

In [ ]:

```python
# Predicting test results
forest_pred = multi_target_forest.predict(x_test)
forest_pred
```

In [ ]:

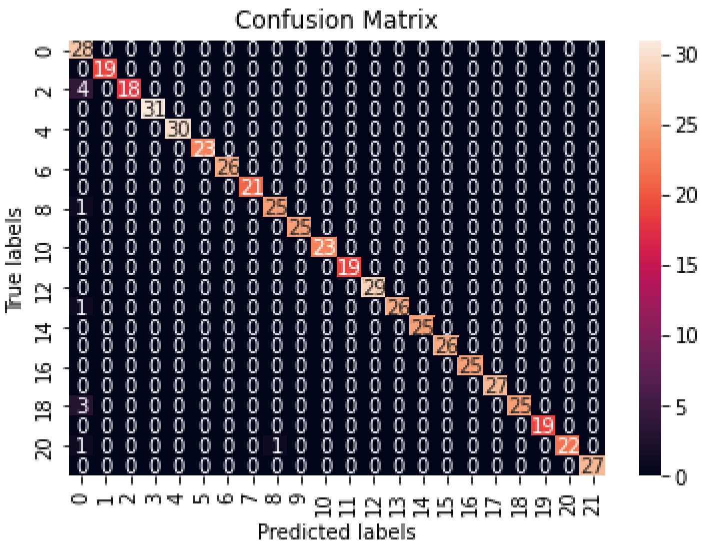
```python
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a3 = accuracy_score(y_test.values.argmax(axis=1), forest_pred.argmax(ax
a3
```

Out[58]:

```
0.98
```

In [ ]:

```python
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), forest_pred.argmax(ax
#cm = confusion_matrix(y_test, gnb_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```



In [ ]:

```python
from sklearn import metrics
# Print the confusion matrix
print(metrics.confusion_matrix(y_test.values.argmax(axis=1), forest_pre

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), forest
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 4  0 18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0]
 [ 1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 22  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27]]
              precision    recall  f1-score   support

           0      0.737     1.000     0.848        28
           1      1.000     1.000     1.000        19
           2      1.000     0.818     0.900        22
           3      1.000     1.000     1.000        31
           4      1.000     1.000     1.000        30
           5      1.000     1.000     1.000        23
           6      1.000     1.000     1.000        26
           7      1.000     1.000     1.000        21
           8      0.962     0.962     0.962        26
           9      1.000     1.000     1.000        25
          10      1.000     1.000     1.000        23
          11      1.000     1.000     1.000        19
          12      1.000     1.000     1.000        29
          13      1.000     0.963     0.981        27
          14      1.000     1.000     1.000        25
          15      1.000     1.000     1.000        26
          16      1.000     1.000     1.000        25
          17      1.000     1.000     1.000        27
          18      1.000     0.893     0.943        28
          19      1.000     1.000     1.000        19
          20      1.000     0.917     0.957        24
          21      1.000     1.000     1.000        27

    accuracy                          0.980       550
   macro avg      0.986     0.980     0.981       550
weighted avg      0.985     0.980     0.981       550
```

In [ ]:
```

```

KNN Classifier

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_clf=KNeighborsClassifier()
model = MultiOutputClassifier(knn_clf, n_jobs=-1)
model.fit(x_train, y_train)
```

Out[61]:
```
MultiOutputClassifier(estimator=KNeighborsClassifier(), n_jobs=-1)
```

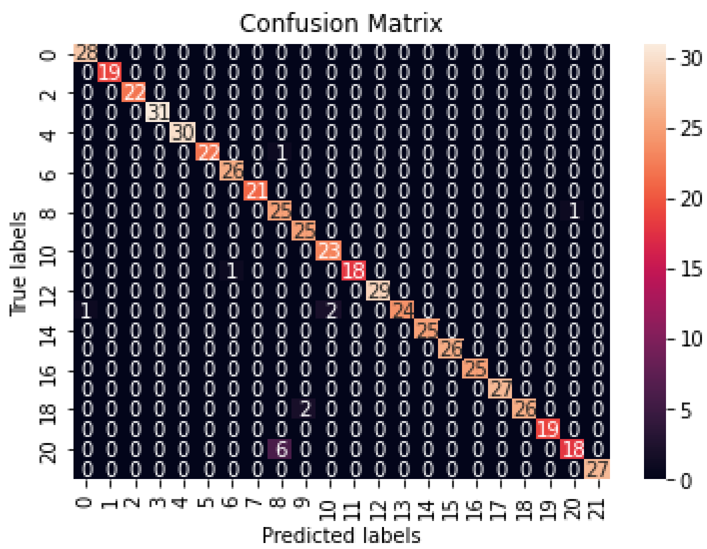
In [ ]:
```
knn_pred = model.predict(x_test)
knn_pred
```

Out[62]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [ ]:
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a4 = accuracy_score(y_test.values.argmax(axis=1), knn_pred.argmax(axis=1
a4
```

Out[63]:
```
0.9745454545454545
```

In [ ]:
```
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), knn_pred.argmax(axis=1
#cm = confusion_matrix(y_test, gnb_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```



In [ ]:

```python
from sklearn import metrics
# Print the confusion matrix
print(metrics.confusion_matrix(y_test.values.argmax(axis=1), knn_pred.ar
      
# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), knn_pr
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 22  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0 18  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  2  0  0 24  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0 26  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0]
 [ 0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0 18  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27]]
              precision    recall  f1-score   support

           0      0.966     1.000     0.982        28
           1      1.000     1.000     1.000        19
           2      1.000     1.000     1.000        22
           3      1.000     1.000     1.000        31
           4      1.000     1.000     1.000        30
           5      1.000     0.957     0.978        23
           6      0.963     1.000     0.981        26
           7      1.000     1.000     1.000        21
           8      0.781     0.962     0.862        26
           9      0.926     1.000     0.962        25
          10      0.920     1.000     0.958        23
          11      1.000     0.947     0.973        19
          12      1.000     1.000     1.000        29
          13      1.000     0.889     0.941        27
          14      1.000     1.000     1.000        25
          15      1.000     1.000     1.000        26
          16      1.000     1.000     1.000        25
          17      1.000     1.000     1.000        27
          18      1.000     0.929     0.963        28
          19      1.000     1.000     1.000        19
          20      0.947     0.750     0.837        24
          21      1.000     1.000     1.000        27

    accuracy                          0.975       550
   macro avg      0.977     0.974     0.974       550
weighted avg      0.977     0.975     0.974       550
```

In [ ]:
```
#Gradient Boosting: In gradient boosting, the goal is to minimize a loss
#through gradient descent. The process involves the following key steps.
```

Gradient Boosting

In [1]:
```
from sklearn.ensemble import GradientBoostingClassifier
gb_clf = GradientBoostingClassifier()
model = MultiOutputClassifier(gb_clf, n_jobs=-1)
model.fit(x_train, y_train)
```

```
E:\AnacondaDATASCIENCE\lib\site-packages\scipy\__init__.py:155: UserWarnin
g: A NumPy version >=1.18.5 and <1.25.0 is required for this version of Sci
Py (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_6896\3412931871.py in <module>
      1 from sklearn.ensemble import GradientBoostingClassifier
      2 gb_clf = GradientBoostingClassifier()
----> 3 model = MultiOutputClassifier(gb_clf, n_jobs=-1)
      4 model.fit(x_train, y_train)

NameError: name 'MultiOutputClassifier' is not defined
```

In [ ]:
```
gf_pred = model.predict(x_test)
gf_pred
```

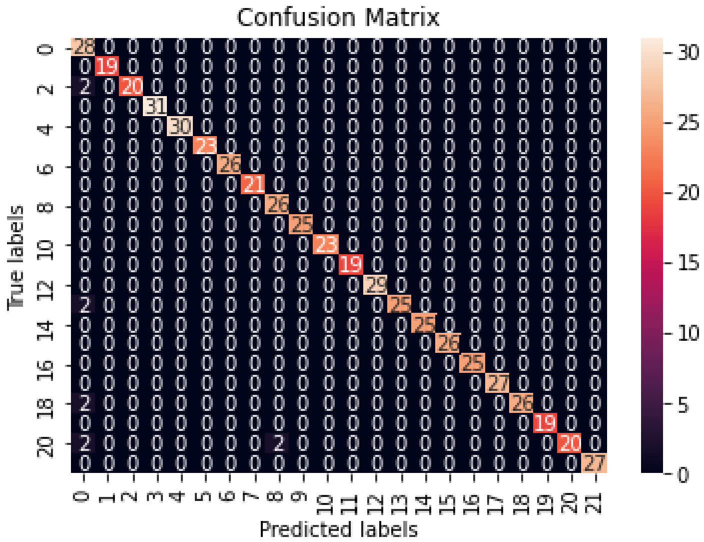
Out[80]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [ ]:
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a5 = accuracy_score(y_test.values.argmax(axis=1), gf_pred.argmax(axis=1)
a5
```

Out[81]:
```
0.9818181818181818
```

In [ ]:

```python
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), gf_pred.argmax(axis=1
#cm = confusion_matrix(y_test, gnb_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```



```python
In [ ]: from sklearn import metrics
        # Print the confusion matrix
        print(metrics.confusion_matrix(y_test.values.argmax(axis=1), gf_pred.arg

        # Print the precision and recall, among other metrics
        print(metrics.classification_report(y_test.values.argmax(axis=1), gf_pre
```

```
[[28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0]
 [ 2  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0]
 [ 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0]
 [ 2  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0 20  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27]]
              precision    recall  f1-score   support

           0      0.778     1.000     0.875        28
           1      1.000     1.000     1.000        19
           2      1.000     0.909     0.952        22
           3      1.000     1.000     1.000        31
           4      1.000     1.000     1.000        30
           5      1.000     1.000     1.000        23
           6      1.000     1.000     1.000        26
           7      1.000     1.000     1.000        21
           8      0.929     1.000     0.963        26
           9      1.000     1.000     1.000        25
          10      1.000     1.000     1.000        23
          11      1.000     1.000     1.000        19
          12      1.000     1.000     1.000        29
          13      1.000     0.926     0.962        27
          14      1.000     1.000     1.000        25
          15      1.000     1.000     1.000        26
          16      1.000     1.000     1.000        25
          17      1.000     1.000     1.000        27
          18      1.000     0.929     0.963        28
          19      1.000     1.000     1.000        19
          20      1.000     0.833     0.909        24
          21      1.000     1.000     1.000        27

    accuracy                          0.982       550
   macro avg      0.987     0.982     0.983       550
weighted avg      0.985     0.982     0.982       550
```

Complete