In [1]:

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

In [2]:

```python
# loading the dataset

crop_data=pd.read_csv("C:\\Users\\Admin\\Downloads\\cpdata.csv")
crop_data
```

Out[2]:

	temperature	humidity	ph	rainfall	label
0	20.879744	82.002744	6.502985	202.935536	rice
1	21.770462	80.319644	7.038096	226.655537	rice
2	23.004459	82.320763	7.840207	263.964248	rice
3	26.491096	80.158363	6.980401	242.864034	rice
4	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...
3095	25.287846	89.636679	6.765095	58.286977	watermelon
3096	26.638386	84.695469	6.189214	48.324286	watermelon
3097	25.331045	84.305338	6.904242	41.532187	watermelon
3098	26.897502	83.892415	6.463271	43.971937	watermelon
3099	26.986037	89.413849	6.260839	58.548767	watermelon

3100 rows × 5 columns

In [3]:

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

Out[3]:

```
(3100, 5)
```

In [4]:

```python
#checking basic information against columns
crop_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3100 entries, 0 to 3099
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   temperature  3100 non-null   float64
 1   humidity     3100 non-null   float64
 2   ph           3100 non-null   float64
 3   rainfall     3100 non-null   float64
 4   label        3100 non-null   object 
dtypes: float64(4), object(1)
memory usage: 121.2+ KB
```

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

In [5]:

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

Out[5]:

```
Index(['temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

In [6]:

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

Out[6]:

	temperature	humidity	ph	rainfall	Crop
0	20.879744	82.002744	6.502985	202.935536	rice
1	21.770462	80.319644	7.038096	226.655537	rice
2	23.004459	82.320763	7.840207	263.964248	rice
3	26.491096	80.158363	6.980401	242.864034	rice
4	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...
3095	25.287846	89.636679	6.765095	58.286977	watermelon
3096	26.638386	84.695469	6.189214	48.324286	watermelon
3097	25.331045	84.305338	6.904242	41.532187	watermelon
3098	26.897502	83.892415	6.463271	43.971937	watermelon
3099	26.986037	89.413849	6.260839	58.548767	watermelon

3100 rows × 5 columns

In [7]:

```
# statistical inference of the dataset
crop_data.describe()
```

Out[7]:

	temperature	humidity	ph	rainfall
count	3100.000000	3100.000000	3100.000000	3100.000000
mean	27.108466	66.005312	6.368913	110.213031
std	7.566308	24.007713	0.809477	64.048562
min	8.825675	10.034048	3.504752	20.211267
25%	22.810495	55.244920	5.895343	64.909095
50%	26.102848	68.980529	6.342518	97.057093
75%	29.365644	84.446524	6.841616	141.210784
max	54.986760	99.981876	9.935091	397.315380

In [8]:

```
#Checking missing values of the dataset in each column
crop_data.isnull().sum()
```

Out[8]:

```
temperature    0
humidity       0
ph             0
rainfall       0
Crop           0
dtype: int64
```

In [9]:

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

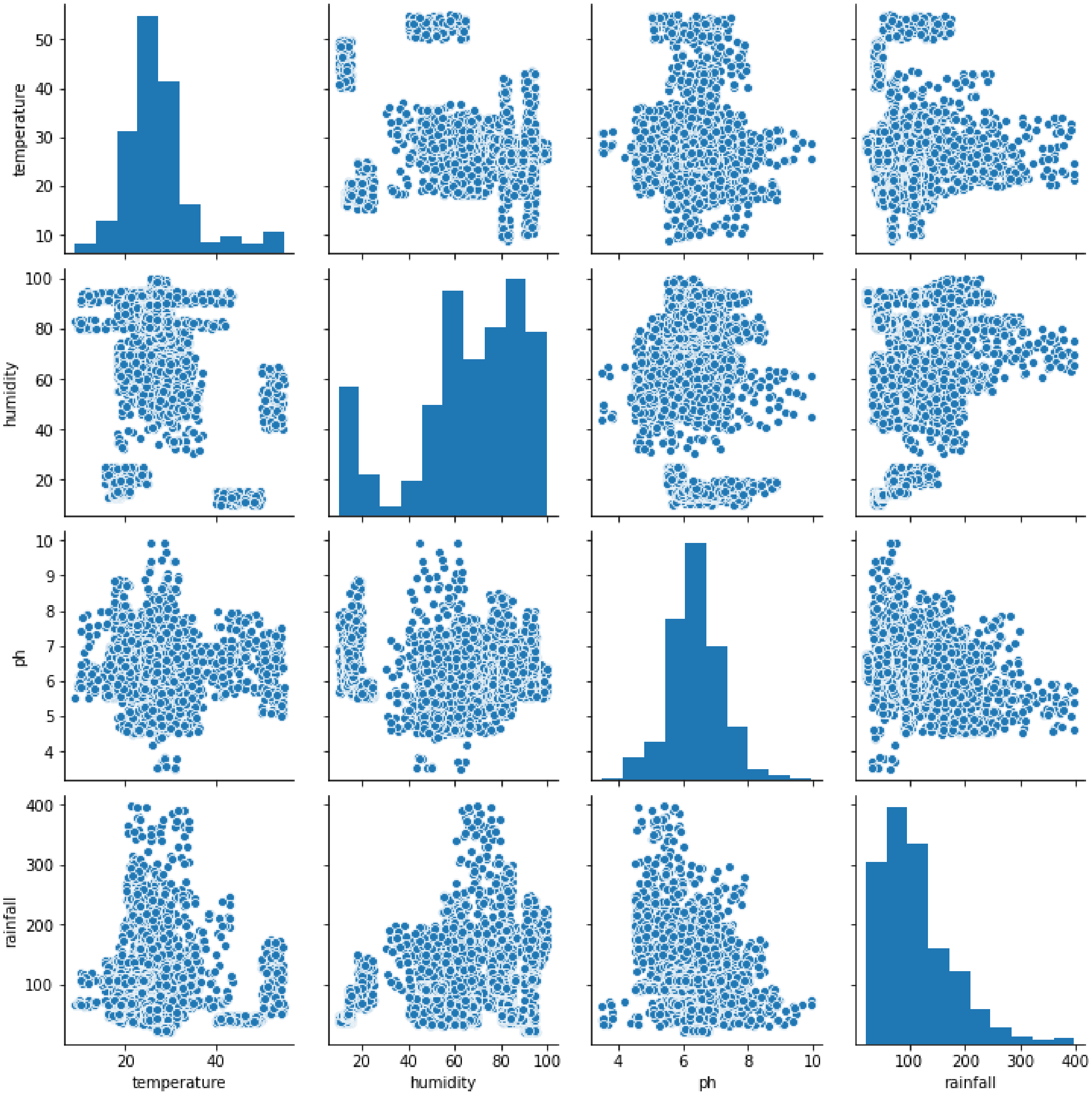
Out[9]:

```
      temperature   humidity        ph    rainfall        Crop
   0    20.879744  82.002744  6.502985  202.935536        rice
   1    21.770462  80.319644  7.038096  226.655537        rice
   2    23.004459  82.320763  7.840207  263.964248        rice
   3    26.491096  80.158363  6.980401  242.864034        rice
   4    20.130175  81.604873  7.628473  262.717340        rice
 ...          ...        ...       ...         ...         ...
3095    25.287846  89.636679  6.765095   58.286977  watermelon
3096    26.638386  84.695469  6.189214   48.324286  watermelon
3097    25.331045  84.305338  6.904242   41.532187  watermelon
3098    26.897502  83.892415  6.463271   43.971937  watermelon
3099    26.986037  89.413849  6.260839   58.548767  watermelon

3100 rows × 5 columns
```

```
In [10]: # Visualizing the features
         ax = sns.pairplot(crop_data)
         ax
```

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x27107932d00>
```



```
In [11]: crop_data.Crop.unique()
```

```
Out[11]: array(['rice', 'wheat', 'Mung Bean', 'Tea', 'millet', 'maize', 'Lentil',
                'Jute', 'Coffee', 'Cotton', 'Ground Nut', 'Peas', 'Rubber',
                'Sugarcane', 'Tobacco', 'Kidney Beans', 'Moth Beans', 'Coconut',
                'Black gram', 'Adzuki Beans', 'Pigeon Peas', 'Chickpea', 'banana',
                'grapes', 'apple', 'mango', 'muskmelon', 'orange', 'papaya',
                'pomegranate', 'watermelon'], dtype=object)
```

```
In [12]: # get top 5 most frequent growing crops
         n = 5
         crop_data['Crop'].value_counts()[:5].index.tolist()
```

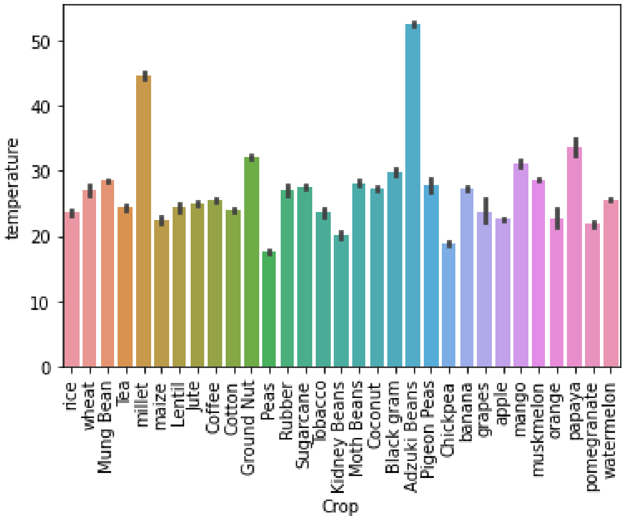
```
Out[12]: ['muskmelon', 'Kidney Beans', 'Sugarcane', 'Jute', 'Cotton']
```

In [13]:

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

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, 22, 23, 24, 25, 26, 27, 28, 29, 30]),
 <a list of 31 Text major ticklabel objects>)
```

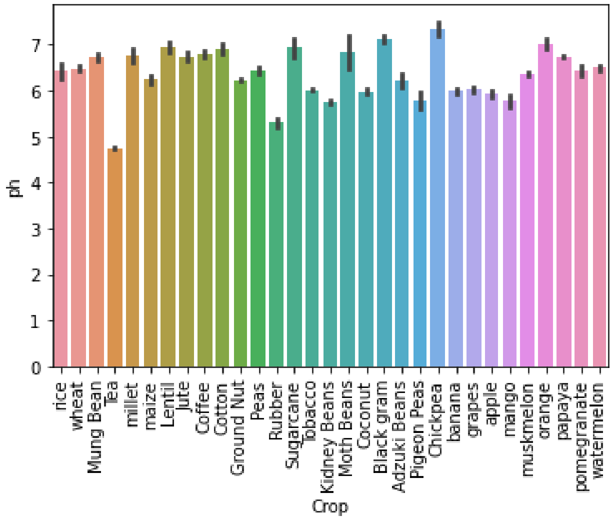


In [14]:

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

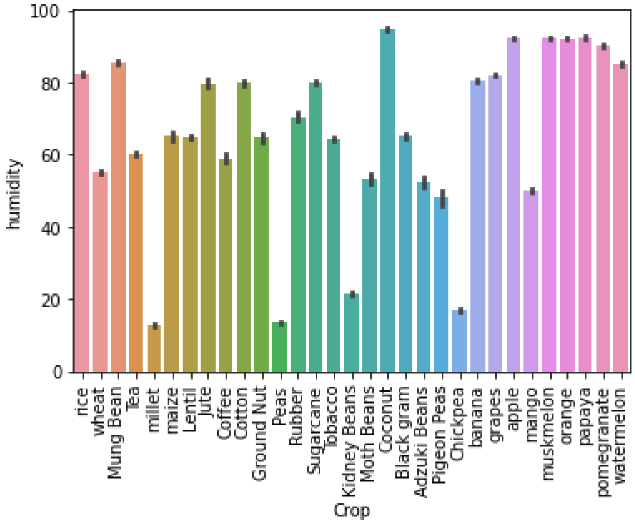
Out[14]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]),
 <a list of 31 Text major ticklabel objects>)
```



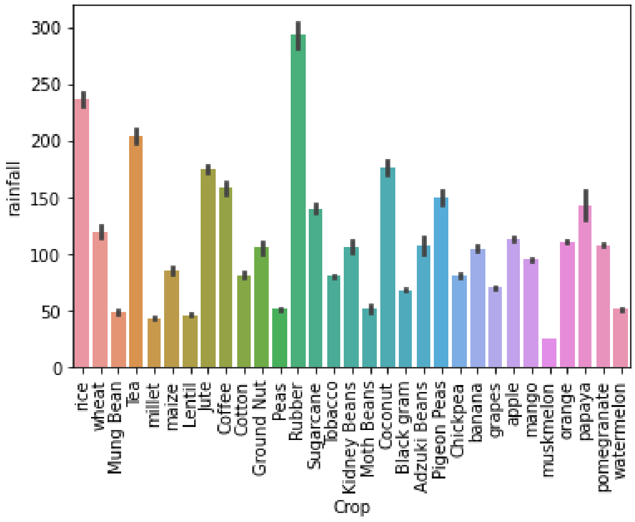
```
In [15]: sns.barplot(crop_data["Crop"], crop_data["humidity"])
         plt.xticks(rotation = 90)
```

```
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, 22, 23, 24, 25, 26, 27, 28, 29, 30]),
          <a list of 31 Text major ticklabel objects>)
```



```
In [16]: sns.barplot(crop_data["Crop"], crop_data["rainfall"])
         plt.xticks(rotation = 90)
```

```
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, 22, 23, 24, 25, 26, 27, 28, 29, 30]),
          <a list of 31 Text major ticklabel objects>)
```



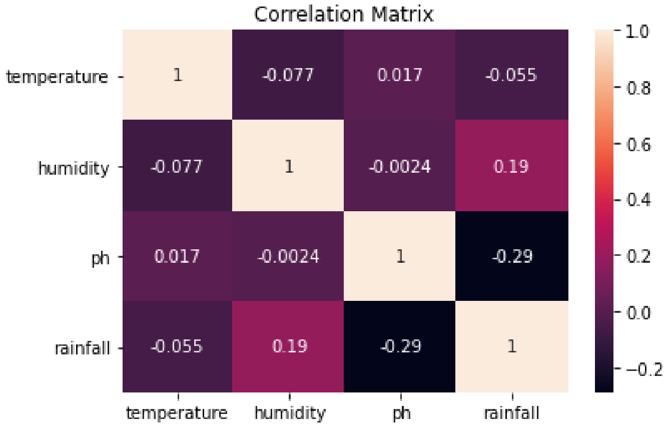
```
In [17]: crop_data.corr()
```

Out[17]:

	temperature	humidity	ph	rainfall
temperature	1.000000	-0.076999	0.017024	-0.055143
humidity	-0.076999	1.000000	-0.002359	0.192074
ph	0.017024	-0.002359	1.000000	-0.288598
rainfall	-0.055143	0.192074	-0.288598	1.000000

In [18]:
```python
sns.heatmap(crop_data.corr(), annot =True)
plt.title('Correlation Matrix')
```

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



In [19]:
```python
# shuffling the dataset to remove order
from sklearn.utils import shuffle

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

Out[19]:

	temperature	humidity	ph	rainfall	Crop
1141	16.912919	13.881680	5.959978	54.026676	Peas
2262	27.486130	76.112398	6.212369	109.276885	banana
1964	53.751483	61.805135	5.410117	130.090866	Adzuki Beans
1456	19.978657	63.458462	5.944788	84.685380	Tobacco
142	28.721646	59.375796	6.743792	121.484053	wheat

In [20]:
```python
# Selection of Feature and Target variables.
x = df[['temperature', 'humidity', 'ph', 'rainfall']]
target = df['Crop']
```

In [21]:
```python
# Encoding target variable
y = pd.get_dummies(target)
y
```

Out[21]:

	Adzuki Beans	Black gram	Chickpea	Coconut	Coffee	Cotton	Ground Nut	Jute	Kidney Beans	Lentil	...	maize	mango	millet	muskmelon	orange
1141	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2262	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1964	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1456	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
142	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1424	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3046	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1725	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0
2254	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2915	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

3100 rows × 31 columns

In [22]:

```
# 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, random_state= 0)

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 : (2325, 4)
x_test : (775, 4)
y_train : (2325, 31)
y_test : (775, 31)
```

In [23]:

```
# 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 [24]:

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

Out[24]:

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

In [25]:

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

Out[25]:

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

In [26]:

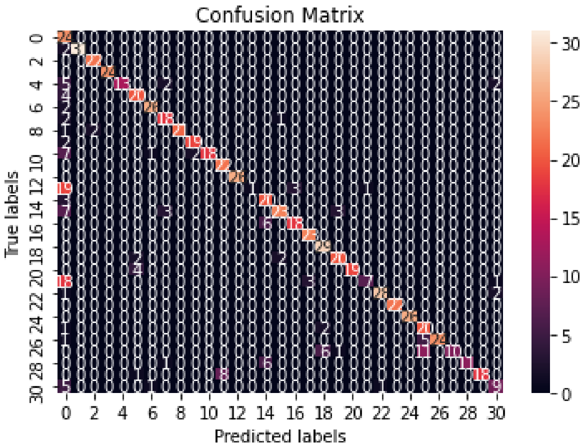
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a1 = accuracy_score(y_test.values.argmax(axis=1), gnb_pred.argmax(axis=1))
a1
```

Out[26]:

```
0.7896774193548387
```

In [27]:

```python
# 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 [28]:

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

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), gnb_pred.argmax(axis=1), digits=3))
```

```
[[24  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  0  0]
 [ 2 31  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  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  0  0  0  0  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  0  0  0  0  0]
 [ 5  0  0  0 13  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  2]
 [ 4  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  0  0  0]
 [ 2  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  0  0]
 [ 2  0  0  0  0  0  0 18  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  0  2  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]
 [ 2  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]
 [ 7  0  0  0  0  0  1  0  0  2 18  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  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  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]
 [19  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  3  0  0  0  0  1  0  0
   0  0  0  0  0  0  0]
 [ 3  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]
 [ 7  0  0  0  0  0  0  3  0  0  0  0  0  0  0 23  0  0  0  3  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  6  0 18  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  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  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  2  0  0  0  0  0  0  0  0  0  2  0  0  0 20  0  0  0  0
   0  0  0  0  0  0  0]
 [ 0  0  0  0  0  4  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]
 [18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  7  0  0
   0  0  0  0  0  0  1]
 [ 1  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  2]
 [ 0  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  0  0  0  0  0  0  0  0  0
  26  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0
   0 20  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  5 24  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  1  0  0  0  0
   0 11  0 10  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0
   0  0  0  0 11  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  8  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0 18  0]
 [ 5  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0  0  0  9]]
              precision    recall  f1-score   support

           0      0.231     1.000     0.375        24
           1      1.000     0.939     0.969        33
           2      0.917     1.000     0.957        22
           3      1.000     1.000     1.000        24
           4      1.000     0.591     0.743        22
           5      0.741     0.833     0.784        24
           6      0.929     0.929     0.929        28
           7      0.750     0.857     0.800        21
           8      1.000     0.875     0.933        24
           9      0.905     0.905     0.905        21
```

```
          10      1.000     0.643     0.783        28
          11      0.733     1.000     0.846        22
          12      1.000     1.000     1.000        26
          13      1.000     0.042     0.080        24
          14      0.625     0.870     0.727        23
          15      0.885     0.639     0.742        36
          16      0.857     0.750     0.800        24
          17      0.885     1.000     0.939        23
          18      0.784     1.000     0.879        29
          19      0.833     0.833     0.833        24
          20      1.000     0.826     0.905        23
          21      0.875     0.241     0.378        29
          22      0.966     0.903     0.933        31
          23      1.000     1.000     1.000        22
          24      1.000     1.000     1.000        26
          25      0.556     0.870     0.678        23
          26      1.000     0.800     0.889        30
          27      1.000     0.357     0.526        28
          28      1.000     0.611     0.759        18
          29      1.000     0.667     0.800        27
          30      0.643     0.562     0.600        16

    accuracy                          0.790       775
   macro avg      0.875     0.792     0.790       775
weighted avg      0.882     0.790     0.792       775
```

Decision Tree Classification

In [29]:

```python
# 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[29]:

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

In [30]:

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

Out[30]:

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

In [31]:

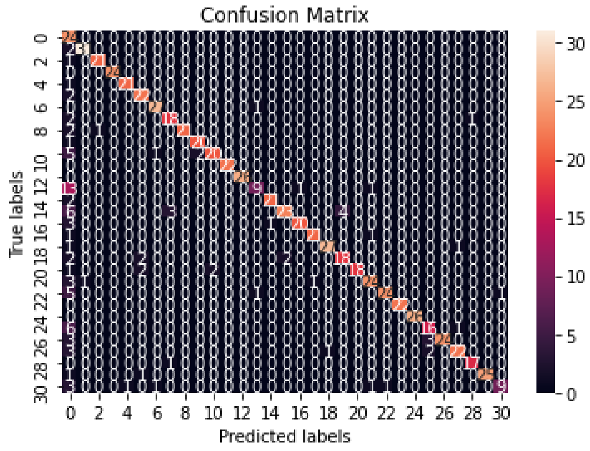
```python
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a2 = accuracy_score(y_test.values.argmax(axis=1), decision_pred.argmax(axis=1))
a2
```

Out[31]:

```
0.8554838709677419
```

In [32]:

```
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), decision_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 [33]:

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

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), decision_pred.argmax(axis=1), digits=3))
```

```
[[24  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  0  0]
 [ 2 31  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  0]
 [ 1  0 21  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]
 [ 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  0  0  0  0  0]
 [ 1  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  0  0  0  0]
 [ 2  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  0  0  0]
 [ 0  0  0  0  0  0 27  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 2  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  1  0  0]
 [ 2  0  1  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]
 [ 1  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]
 [ 5  0  0  0  0  0  1  0  0  2 20  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  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  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]
 [13  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  1  0  0  0  0  1  0  0
   0  0  0  0  0  0  0]
 [ 2  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]
 [ 6  0  0  0  0  0  0  3  0  0  0  0  0  0  0 23  0  0  0  4  0  0  0  0
   0  0  0  0  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 20  0  0  0  0  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 21  0  0  0  1  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 27  0  0  0  0  0
   0  0  0  1  0  0  0]
 [ 2  0  0  0  0  2  0  0  0  0  0  0  0  0  0  2  0  0  0 18  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  0  0  0  0  2  0  0  0  0  2  0  0  0  0  0  0  0  0  0 18  0  0  0
   0  0  0  0  0  0  0]
 [ 3  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 24  0  0
   0  0  0  0  0  0  0]
 [ 5  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0 24  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  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  0  0  0  0  0  0  0  0  0
  26  0  0  0  0  0  0]
 [ 6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0 16  0  1  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  3 24  0  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
   0  2  0 22  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  0  0  0 17  0  0]
 [ 1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0 25  0]
 [ 3  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0
   0  0  0  0  0  0  9]]
              precision    recall  f1-score   support

           0      0.258     1.000     0.410        24
           1      0.969     0.939     0.954        33
           2      0.955     0.955     0.955        22
           3      1.000     1.000     1.000        24
           4      0.955     0.955     0.955        22
           5      0.815     0.917     0.863        24
           6      0.931     0.964     0.947        28
           7      0.818     0.857     0.837        21
           8      1.000     0.875     0.933        24
           9      0.909     0.952     0.930        21
```

```
          10      0.909     0.714     0.800        28
          11      1.000     1.000     1.000        22
          12      1.000     1.000     1.000        26
          13      0.818     0.375     0.514        24
          14      0.955     0.913     0.933        23
          15      0.920     0.639     0.754        36
          16      0.952     0.833     0.889        24
          17      0.955     0.913     0.933        23
          18      0.964     0.931     0.947        29
          19      0.818     0.750     0.783        24
          20      1.000     0.783     0.878        23
          21      0.889     0.828     0.857        29
          22      0.960     0.774     0.857        31
          23      1.000     1.000     1.000        22
          24      1.000     1.000     1.000        26
          25      0.762     0.696     0.727        23
          26      1.000     0.800     0.889        30
          27      0.917     0.786     0.846        28
          28      0.944     0.944     0.944        18
          29      1.000     0.926     0.962        27
          30      0.900     0.562     0.692        16

    accuracy                          0.855       775
   macro avg      0.912     0.857     0.871       775
weighted avg      0.915     0.855     0.871       775
```

Random Forest Classification

In [34]:

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

Out[34]:

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

In [35]:

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

Out[35]:

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

In [36]:

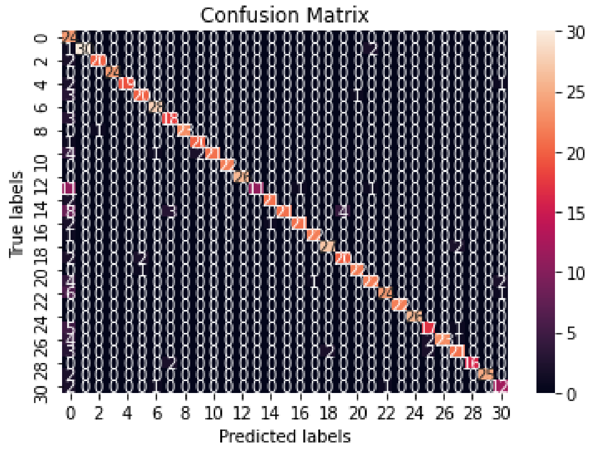
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a3 = accuracy_score(y_test.values.argmax(axis=1), forest_pred.argmax(axis=1))
a3
```

Out[36]:

```
0.8619354838709677
```

In [37]:

```python
# creating a confusion matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test.values.argmax(axis=1), forest_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 [38]:

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

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), forest_pred.argmax(axis=1), digits=3))
```

```
[[24  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  0  0]
 [ 1 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  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  0  0  0  0  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  0  0  0  0  0]
 [ 2  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  0  0  0  1]
 [ 3  0  0  0  0 20  0  0  0  0  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 28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 3  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  0]
 [ 0  0  1  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]
 [ 1  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]
 [ 4  0  0  0  0  0  1  0  0  2 21  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  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  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]
 [11  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  1  0  0  0  0  1  0  0
   0  0  0  0  0  0  0]
 [ 2  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]
 [ 8  0  0  0  0  0  0  3  0  0  0  0  0  0  0 21  0  0  0  4  0  0  0  0
   0  0  0  0  0  0  0]
 [ 2  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 21  0  0  0  0  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 22  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  0  0  0 27  0  0  0  0  0
   0  0  0  2  0  0  0]
 [ 2  0  0  0  0  2  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  1  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]
 [ 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 22  0  0
   0  0  0  0  0  0  2]
 [ 6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 24  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  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  0  0  0  0  0  0  0  0  0
  26  0  0  0  0  0  0]
 [ 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0 17  0  1  0  0  0]
 [ 4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  2 23  1  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0
   0  2  0 21  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0 16  0  0]
 [ 2  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 25  0]
 [ 2  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0  0  0 12]]
              precision    recall  f1-score   support

           0      0.261     1.000     0.414        24
           1      1.000     0.909     0.952        33
           2      0.952     0.909     0.930        22
           3      1.000     1.000     1.000        24
           4      1.000     0.864     0.927        22
           5      0.870     0.833     0.851        24
           6      0.933     1.000     0.966        28
           7      0.783     0.857     0.818        21
           8      1.000     0.958     0.979        24
           9      0.909     0.952     0.930        21
```

```
          10      1.000     0.750     0.857        28
          11      1.000     1.000     1.000        22
          12      1.000     1.000     1.000        26
          13      1.000     0.458     0.629        24
          14      0.955     0.913     0.933        23
          15      1.000     0.583     0.737        36
          16      0.955     0.875     0.913        24
          17      0.957     0.957     0.957        23
          18      0.931     0.931     0.931        29
          19      0.833     0.833     0.833        24
          20      0.957     0.957     0.957        23
          21      0.880     0.759     0.815        29
          22      0.960     0.774     0.857        31
          23      1.000     1.000     1.000        22
          24      1.000     1.000     1.000        26
          25      0.810     0.739     0.773        23
          26      1.000     0.767     0.868        30
          27      0.840     0.750     0.792        28
          28      1.000     0.889     0.941        18
          29      1.000     0.926     0.962        27
          30      0.750     0.750     0.750        16

    accuracy                          0.862       775
   macro avg      0.920     0.868     0.880       775
weighted avg      0.925     0.862     0.879       775
```

KNN Classifier

In [39]:

```
from sklearn.neighbors import KNeighborsClassifier

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

Out[39]:

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

In [40]:

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

Out[40]:

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

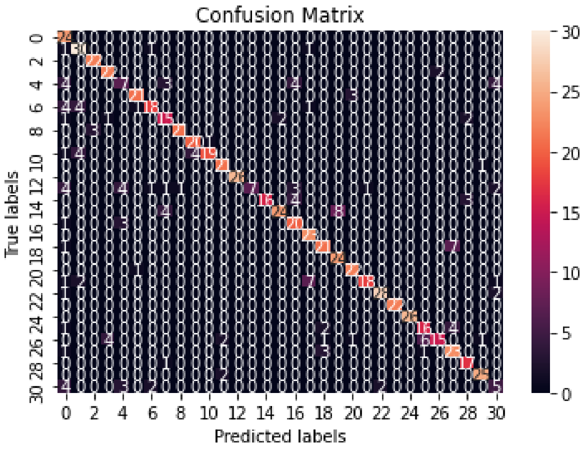
In [41]:

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

Out[41]:

```
0.7974193548387096
```

```
In [42]: # 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 [43]:

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

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), knn_pred.argmax(axis=1), digits=3))
```

```
[[24  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  0  0]
 [ 1 30  0  0  0  0  1  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 22  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]
 [ 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  2  0  0  0  0]
 [ 4  0  0  0  7  0  0  3  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0
   0  0  0  0  0  0  4]
 [ 0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0
   0  0  0  0  0  0  0]
 [ 4  4  0  0  0  1 18  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  0  0  1  0  0  0 15  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0
   0  0  0  0  2  0  0]
 [ 0  0  3  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]
 [ 0  0  0  0  0  0  0  0  0 20  1  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  4  0  0  0  0  0  0  0  4 19  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  0  0  0 21  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  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]
 [ 4  0  0  0  4  0  1  1  1  0  0  0  0  7  0  0  3  0  0  0  0  1  0  0
   0  0  0  0  0  0  2]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  4  0  0  0  0  0  0  0
   0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0  0  4  0  0  0  0  0  0  0 24  0  0  0  8  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  0  0  0  3  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  0  0  0  0  0  0  0  0  0 23  0  0  0  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 21  0  0  0  0  0
   0  0  0  7  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 24  0  0  0  0
   0  0  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 22  0  0  0
   0  0  0  0  0  0  0]
 [ 1  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0 18  0  0
   0  0  0  0  0  0  1]
 [ 1  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  2]
 [ 0  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  0  0  0  0  0  0  0  0  0
  26  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0
   0 16  0  4  0  0  0]
 [ 0  0  0  4  0  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0  1  0  0  0
   0  6 15  0  0  1  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0
   0  1  0 23  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  0  0  0 17  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0 25  0]
 [ 4  0  0  0  3  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0
   0  0  0  0  0  0  5]]
              precision    recall  f1-score   support

           0      0.490     1.000     0.658        24
           1      0.750     0.909     0.822        33
           2      0.880     1.000     0.936        22
           3      0.815     0.917     0.863        24
           4      0.412     0.318     0.359        22
           5      0.913     0.875     0.894        24
           6      0.818     0.643     0.720        28
           7      0.625     0.714     0.667        21
           8      0.955     0.875     0.913        24
           9      0.833     0.952     0.889        21
```

```
          10      0.950     0.679     0.792        28
          11      0.840     0.955     0.894        22
          12      1.000     1.000     1.000        26
          13      1.000     0.292     0.452        24
          14      1.000     0.696     0.821        23
          15      0.923     0.667     0.774        36
          16      0.645     0.833     0.727        24
          17      0.719     1.000     0.836        23
          18      0.778     0.724     0.750        29
          19      0.750     1.000     0.857        24
          20      0.846     0.957     0.898        23
          21      0.947     0.621     0.750        29
          22      0.933     0.903     0.918        31
          23      1.000     1.000     1.000        22
          24      1.000     1.000     1.000        26
          25      0.696     0.696     0.696        23
          26      0.882     0.500     0.638        30
          27      0.676     0.821     0.742        28
          28      0.773     0.944     0.850        18
          29      0.926     0.926     0.926        27
          30      0.357     0.312     0.333        16

    accuracy                          0.797       775
   macro avg      0.811     0.798     0.786       775
weighted avg      0.821     0.797     0.791       775
```

Gradient Boosting

In [44]:

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

Out[44]:

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

In [45]:

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

Out[45]:

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

In [46]:

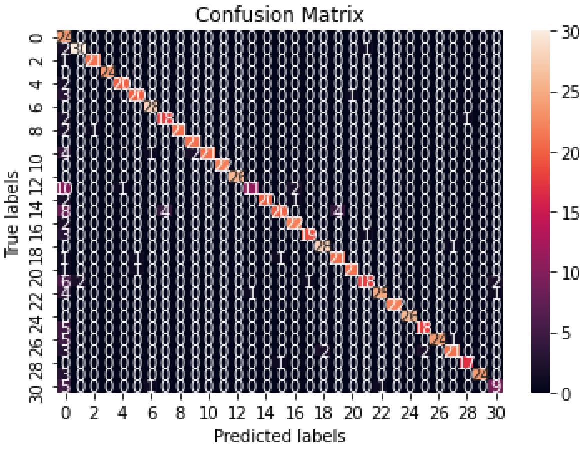
```
# Calculating Accuracy
from sklearn.metrics import accuracy_score
a5 = accuracy_score(y_test.values.argmax(axis=1), gf_pred.argmax(axis=1))
a5
```

Out[46]:

```
0.8541935483870968
```

In [47]:

```
# 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');
```



In [48]:

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

# Print the precision and recall, among other metrics
print(metrics.classification_report(y_test.values.argmax(axis=1), gf_pred.argmax(axis=1), digits=3))
```

```
[[24  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  0  0]
 [ 2 30  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  0  0]
 [ 1  0 21  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]
 [ 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  0  0  0  0  0]
 [ 2  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  0  0  0  0]
 [ 3  0  0  0  0 20  0  0  0  0  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 28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 2  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  1  0  0]
 [ 2  0  1  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]
 [ 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]
 [ 4  0  0  0  0  0  1  0  0  2 21  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  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  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]
 [10  0  0  0  1  0  0  0  0  0  0  0  0 11  0  0  2  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 2  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  1  0  0  0  0  0  0  0
   0  0  0  0  0  0  0]
 [ 8  0  0  0  0  0  0  4  0  0  0  0  0  0  0 20  0  0  0  4  0  0  0  0
   0  0  0  0  0  0  0]
 [ 2  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]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  1  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 28  0  0  0  0  0
   0  0  0  1  0  0  0]
 [ 1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0 21  0  0  0  0
   0  0  0  0  0  0  0]
 [ 1  0  0  0  0  1  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]
 [ 6  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 18  0  0
   0  0  0  0  0  0  2]
 [ 4  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0 25  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  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  0  0  0  0  0  0  0  0  0
  26  0  0  0  0  0  0]
 [ 5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0 18  0  0  0  0  0]
 [ 5  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 24  1  0  0  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0
   0  2  0 21  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  0
   0  0  0  0 17  0  0]
 [ 3  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 24  0]
 [ 5  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0  0  0  9]]
              precision    recall  f1-score   support

           0      0.245     1.000     0.393        24
           1      0.938     0.909     0.923        33
           2      0.955     0.955     0.955        22
           3      1.000     1.000     1.000        24
           4      0.952     0.909     0.930        22
           5      0.909     0.833     0.870        24
           6      0.933     1.000     0.966        28
           7      0.818     0.857     0.837        21
           8      1.000     0.875     0.933        24
           9      0.913     1.000     0.955        21
```

```
          10      1.000     0.750     0.857        28
          11      1.000     1.000     1.000        22
          12      1.000     1.000     1.000        26
          13      0.917     0.458     0.611        24
          14      1.000     0.870     0.930        23
          15      0.909     0.556     0.690        36
          16      0.880     0.917     0.898        24
          17      0.950     0.826     0.884        23
          18      0.933     0.966     0.949        29
          19      0.840     0.875     0.857        24
          20      0.955     0.913     0.933        23
          21      0.900     0.621     0.735        29
          22      0.962     0.806     0.877        31
          23      1.000     1.000     1.000        22
          24      1.000     1.000     1.000        26
          25      0.900     0.783     0.837        23
          26      1.000     0.800     0.889        30
          27      0.913     0.750     0.824        28
          28      0.944     0.944     0.944        18
          29      1.000     0.889     0.941        27
          30      0.750     0.562     0.643        16

    accuracy                          0.854       775
   macro avg      0.917     0.859     0.873       775
weighted avg      0.920     0.854     0.872       775
```