SKILL ACTIVITY NO:03

Name:Mahesh Pandurang Shelar Date:01/08/2021

PRN:2018033700025397

School:Symboisis Skill and Professional University,Pune

Program: Machine Learning & Artificial Intelligence Analyst

Batch: ML11

Module Name: Python Programming

Module Code: ML11

####Title: Perform Classification on the Glass Dataset. Use appropriate libraries

####Skills/Competencies to be acquired:
1.To gain an understanding of data and find clues from the data.
2.Assess assumptions on which statistical inference will be based.
3.To check the quality of data for further processing and cleaning if necessary.
4.To check for outliers that may impact model.
5.Data Visualization.
6.scaling the data.
7.classify data into training and testing.
8.machine learning classification algorithms.

####Duration of activity: 6 to 7 Hour

1.What is the purpose of this activity?

1)Preview data.
2)Check total number of entries and column types.
3)Check any null values.
4)Check duplicate entries.
5)Plot distribution of numeric data.
6)Plot count distribution of categorical data.
7)Prepare a Classification on the Glass Dataset, and Find the best model on the dataset..

2.Steps performed in this activity.

1)import all necessary libraries.
2)read dataset using pandas.
3)check the null valus,duplicates,type of datacolumns etc.
4)check statistics of dataset.
5)check for outliers and then removed outliers.

6)check for correlation.
7)data visualization.
8)seperate target column.
9)splitting the data into training and testing.
10)scaling.
11)used different machine learning algorithms and evaluation for checking accuracy.

3.What resources / materials / equipment / tools did you use for this activity?

1)Jupyter Notebook

4.What skills did you acquire?

1)data cleaning
2)data visualization
3)exploratory data analysis
4)data preprocessing
5)python
6)Machine Learning

5.Time taken to complete the activity?

6 to 7 hours

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv("C:/Users/Expert/Downloads/glass.csv - glass.csv.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [4]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [5]:

```
1 df.isna().sum()
```

Out[5]:

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

In [6]:

```
1 df.describe()
```

Out[6]:

	RI	Na	Mg	Al	Si	K	Ca	2
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	
25%	1.516523	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	

In [7]:

```
1 df.duplicated()
```

Out[7]:

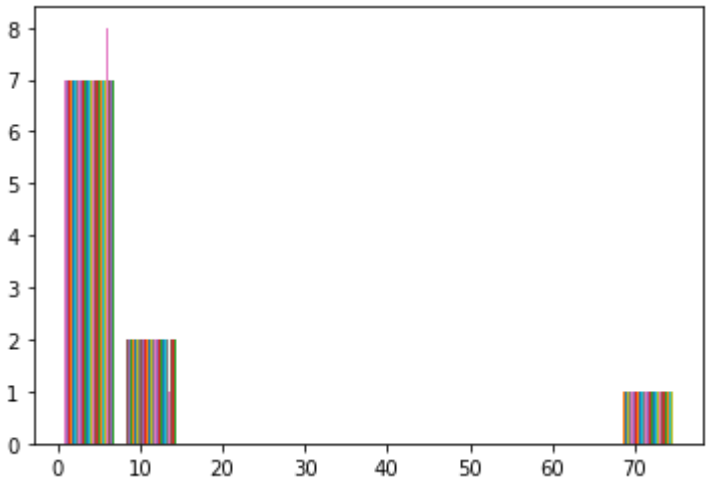
```
0      False
1      False
2      False
3      False
4      False
       ...
209    False
210    False
211    False
212    False
213    False
Length: 214, dtype: bool
```

In [8]:

```
1 plt.hist(df)
```

Out[8]:

```
(array([[7., 2., 0., ..., 0., 0., 1.],
        [7., 2., 0., ..., 0., 0., 1.],
        [7., 2., 0., ..., 0., 0., 1.],
        ...,
        [7., 2., 0., ..., 0., 0., 1.],
        [7., 2., 0., ..., 0., 0., 1.],
        [7., 2., 0., ..., 0., 0., 1.]]),
 array([ 0.   ,  7.541, 15.082, 22.623, 30.164, 37.705, 45.246, 52.787,
        60.328, 67.869, 75.41 ]),
 <a list of 214 Lists of Patches objects>)
```

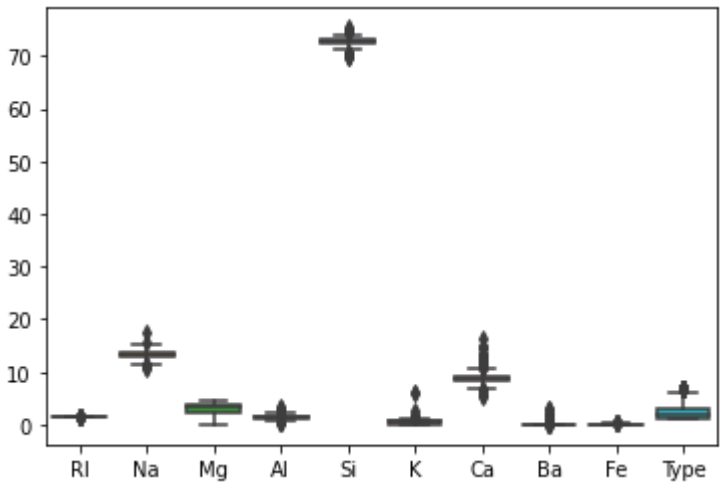


In [9]:

```
1 sns.boxplot(data=df)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29c3ff8a040>
```



In [10]:

```
1 np.unique(df["Type"])
```

Out[10]:

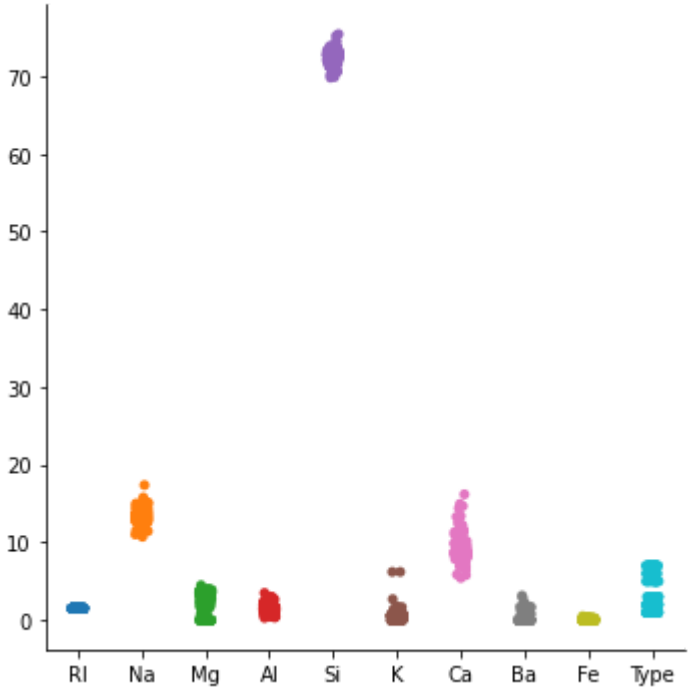
```
array([1, 2, 3, 5, 6, 7], dtype=int64)
```

In [11]:

```
1 sns.catplot(data=df)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x29c3f7f4af0>
```

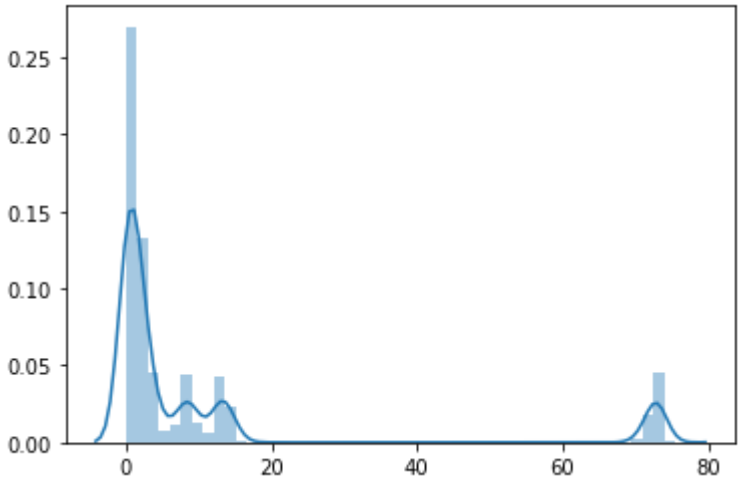


In [12]:

```
1 sns.distplot(df)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29c3ecaedc0>
```

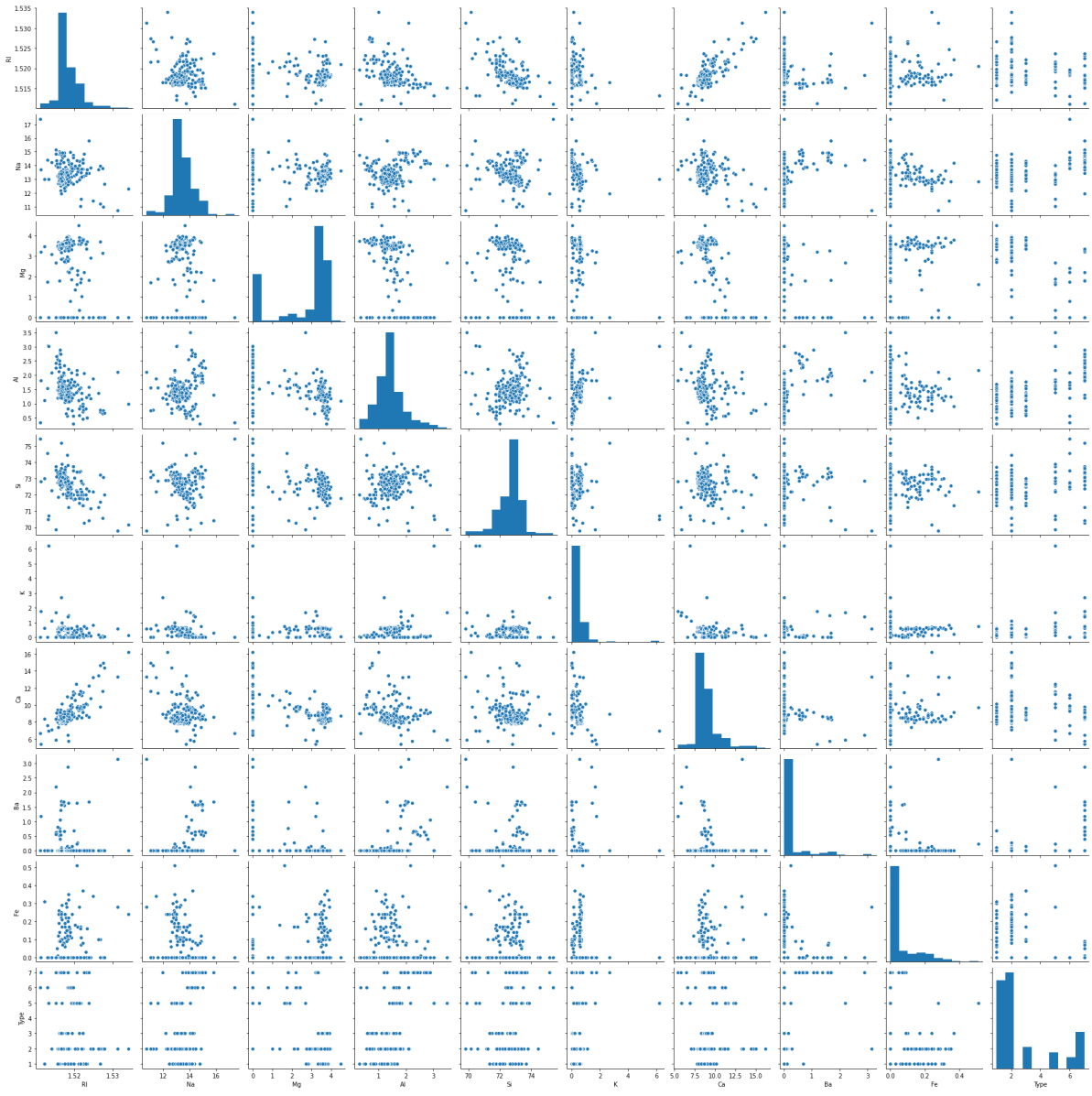


In [13]:

```
1 sns.pairplot(data=df)
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x29c402c91f0>
```



In [14]:

```
1 df.corr()
```

Out[14]:

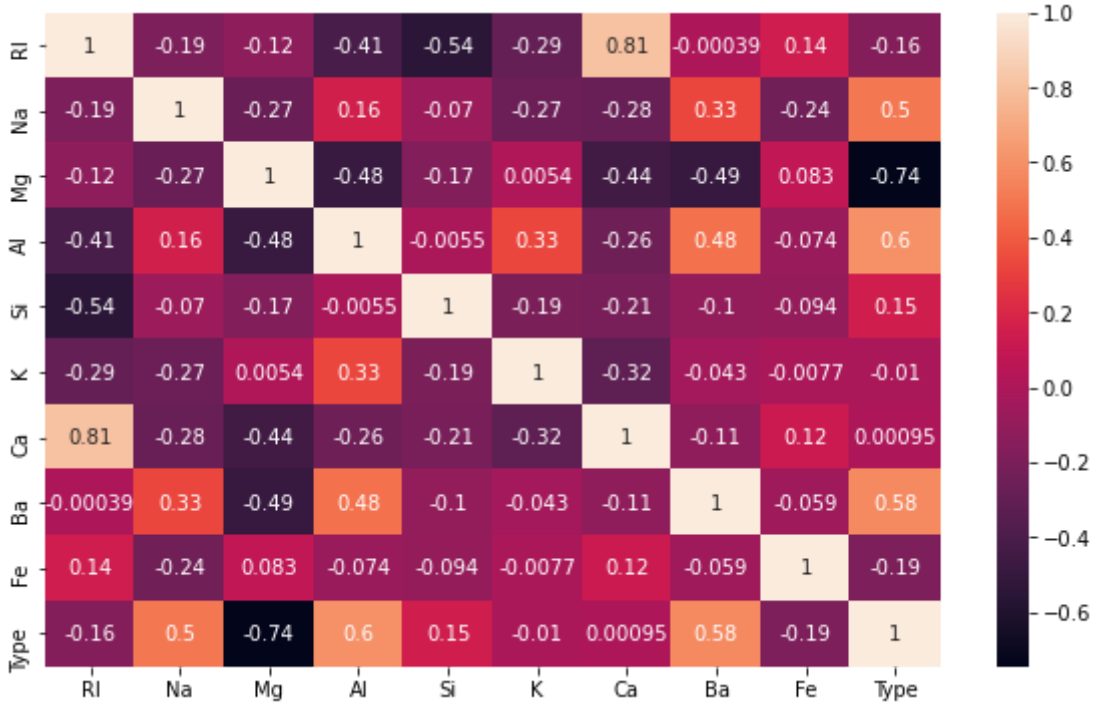
	RI	Na	Mg	Al	Si	K	Ca	Ba	
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.

In [15]:

```
plt.figure(figsize=[10,6])
sns.heatmap(df.corr(),annot=True)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29c43a190d0>
```



In [16]:

```
x=df.drop(columns=["Type"])
y=df["Type"]
```

In [17]:

```
#splitting the data into training and testing
```

In [18]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [19]:

```
#Implementing Various Machine Learning Algorithms To Find Best Fit Model
```

LogisticRegression

In [20]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(xtrain,ytrain)
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to conver
ge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[20]:

```
LogisticRegression()
```

In [21]:

```
ypred=model.predict(xtest)
```

In [22]:

```
ypred.shape
```

Out[22]:

```
(43,)
```

Evaluation

In [23]:

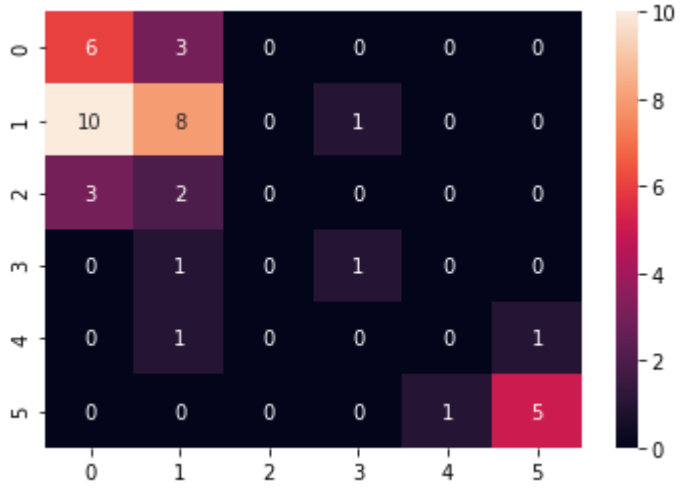
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.46511627906976744
              precision    recall  f1-score   support

           1       0.32      0.67      0.43         9
           2       0.53      0.42      0.47        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.83      0.83      0.83         6

    accuracy                           0.47        43
   macro avg       0.36      0.40      0.37        43
weighted avg       0.44      0.47      0.44        43


C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



Scaling

In [24]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc_xtrain=sc.fit_transform(xtrain)
sc_xtest=sc.fit_transform(xtest)
```

In [25]:

```
#implementing model on scaled data
```

In [26]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(sc_xtrain,ytrain)
```

Out[26]:

```
LogisticRegression()
```

In [27]:

```
ypred=model.predict(sc_xtest)
```

In [28]:

```
#Evaluation
```

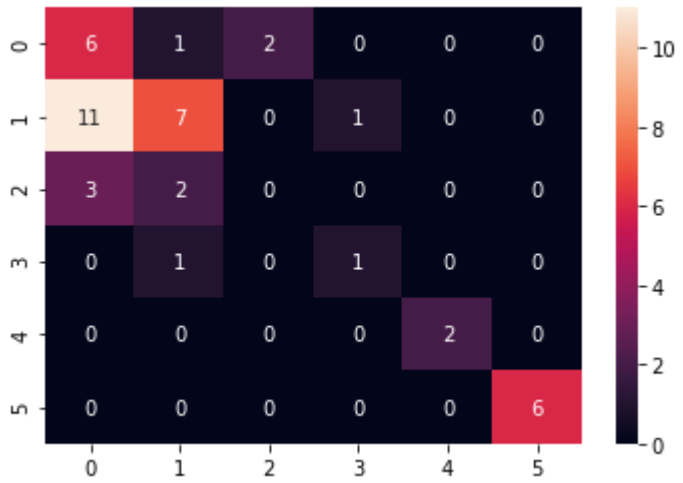
In [29]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.5116279069767442
              precision    recall  f1-score   support

           1       0.30      0.67      0.41         9
           2       0.64      0.37      0.47        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         6

    accuracy                           0.51        43
   macro avg       0.57      0.59      0.56        43
weighted avg       0.55      0.51      0.50        43
```



##after scaling accuracy is increased by 0.47 to 0.51

Hyperparameter Tuning

In [30]:

```
help(LogisticRegression())
```

```
Help on LogisticRegression in module sklearn.linear_model._logistic obje
ct:

class LogisticRegression(sklearn.base.BaseEstimator, sklearn.linear_mode
l._base.LinearClassifierMixin, sklearn.linear_model._base.SparseCoefMixi
n)
 |  LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, f
it_intercept=True, intercept_scaling=1, class_weight=None, random_state=
None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_
start=False, n_jobs=None, l1_ratio=None)
 |
 |  Logistic Regression (aka logit, MaxEnt) classifier.
 |
 |  In the multiclass case, the training algorithm uses the one-vs-rest
(OvR)
 |  scheme if the 'multi_class' option is set to 'ovr', and uses the
 |  cross-entropy loss if the 'multi_class' option is set to 'multinomia
l'.
 |  (Currently the 'multinomial' option is supported only by the 'lbfg
```

In [31]:

```
model=LogisticRegression()
#Parameters
solver=['newton-cg','lbfgs','liblinear','sag','saga']
multi_class=['auto','ovr','multinomial']
penalty=['l1','l2','elasticnet']
grid=dict(solver=solver,multi_class=multi_class,penalty=penalty)
#cv
from sklearn.model_selection import RepeatedStratifiedKFold
cv=RepeatedStratifiedKFold(n_splits=5,n_repeats=3,random_state=0)
#GridSearchCV
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(estimator=model,param_grid=grid,cv=cv,scoring="accuracy")
res=grid_cv.fit(sc_xtrain,ytrain)
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklear
n\model_selection\_validation.py:548: FitFailedWarning: Estimator fit fa
iled. The score on this train-test partition for these parameters will b
e set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\linear_model\_logistic.py", line 1304, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\linear_model\_logistic.py", line 442, in _check_solver
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver newton-cg supports only 'l2' or 'none' penalties, got
l1 penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
```

In [32]:

```
1 res.best_score_
```

Out[32]:

```
0.6457142857142857
```

In [33]:

```
1 res.best_params_
```

Out[33]:

```
{'multi_class': 'auto', 'penalty': 'l1', 'solver': 'saga'}
```

In [34]:

```
1 from sklearn.linear_model import LogisticRegression
2 model=LogisticRegression(multi_class='auto',penalty='l1',solver='saga')
3 model.fit(sc_xtrain,ytrain)
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\linear_model\_sag.py:329: ConvergenceWarning: The max_iter was reached wh
ich means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

Out[34]:

```
LogisticRegression(penalty='l1', solver='saga')
```

In [35]:

```
1 ypred=model.predict(sc_xtest)
```

In [36]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(ytest,ypred)
print(cm)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
[[7 2 0 0 0 0]
 [9 9 0 1 0 0]
 [3 2 0 0 0 0]
 [0 1 0 1 0 0]
 [0 0 0 0 2 0]
 [0 0 0 0 1 5]]
accuracy is: 0.5581395348837209
              precision    recall  f1-score   support

           1       0.37      0.78      0.50         9
           2       0.64      0.47      0.55        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.67      1.00      0.80         2
           7       1.00      0.83      0.91         6

    accuracy                           0.56        43
   macro avg       0.53      0.60      0.54        43
weighted avg       0.55      0.56      0.53        43
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Note:after hyperparameter tuning accuracy of model increased by 0.51 to 0.56

KNeighborsClassifier

In [37]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=50)
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

Evaluation

In [38]:

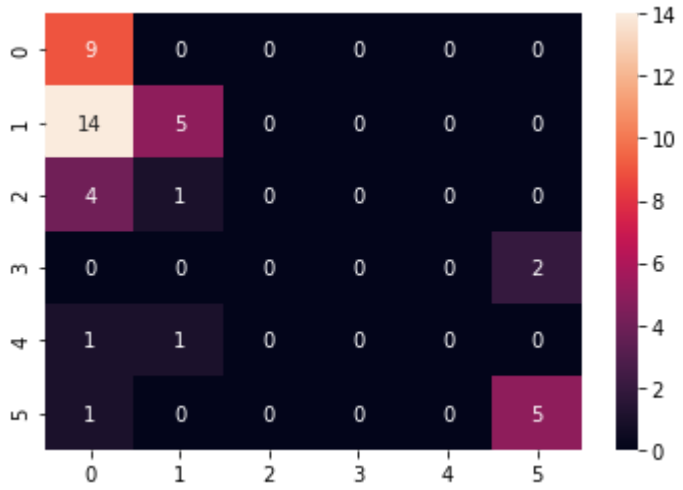
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(ytest,ypred)
print(cm)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
[[ 9  0  0  0  0  0]
 [14  5  0  0  0  0]
 [ 4  1  0  0  0  0]
 [ 0  0  0  0  0  2]
 [ 1  1  0  0  0  0]
 [ 1  0  0  0  0  5]]
accuracy is: 0.4418604651162791
              precision    recall  f1-score   support

           1       0.31      1.00      0.47         9
           2       0.71      0.26      0.38        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.71      0.83      0.77         6

    accuracy                           0.44        43
   macro avg       0.29      0.35      0.27        43
weighted avg       0.48      0.44      0.38        43
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



hyperparameter tuning

In [39]:

```
help(KNeighborsClassifier(n_neighbors=50))
```

```
Help on KNeighborsClassifier in module sklearn.neighbors._classification
object:

class KNeighborsClassifier(sklearn.neighbors._base.NeighborsBase, sklear
n.neighbors._base.KNeighborsMixin, sklearn.neighbors._base.SupervisedInt
egerMixin, sklearn.base.ClassifierMixin)
 |  KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm
='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jo
bs=None, **kwargs)
 |
 |  Classifier implementing the k-nearest neighbors vote.
 |
 |  Read more in the :ref:`User Guide <classification>`.
 |
 |  Parameters
 |  ----------
 |  n_neighbors : int, default=5
 |      Number of neighbors to use by default for :meth:`kneighbors` que
ries.
```

In [40]:

```
model=KNeighborsClassifier()
#parameters
n_neighbors=[50,100,130]
weights=['uniform', 'distance']
algorithm=['auto', 'ball_tree', 'kd_tree', 'brute']
grid=dict(n_neighbors=n_neighbors,weights=weights,algorithm=algorithm)
#cv
from sklearn.model_selection import RepeatedStratifiedKFold
cv=RepeatedStratifiedKFold(n_splits=5,n_repeats=3,random_state=0)
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(estimator=model,param_grid=grid,cv=cv,scoring="accuracy")
res=grid_cv.fit(xtrain,ytrain)
print(res.best_score_)
```

```
0.6140616246498599
```

In [41]:

```
res.best_params_
```

Out[41]:

```
{'algorithm': 'auto', 'n_neighbors': 50, 'weights': 'distance'}
```

In [42]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=50,weights='distance',algorithm='auto')
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

In [43]:

```
#Evaluation
```

In [44]:

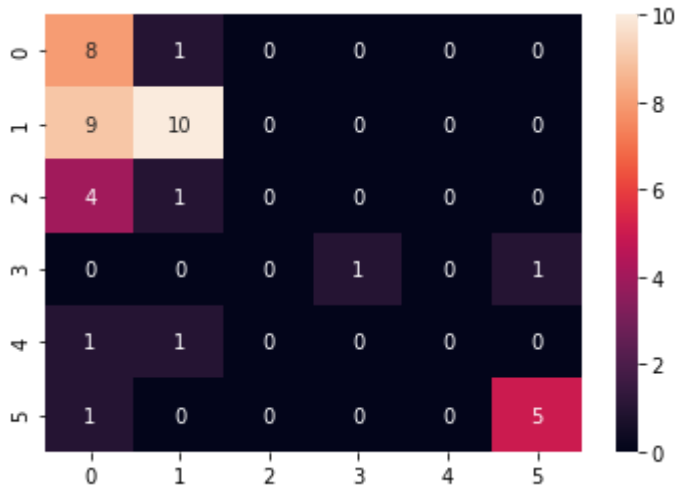
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(ytest,ypred)
print(cm)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
[[ 8  1  0  0  0  0]
 [ 9 10  0  0  0  0]
 [ 4  1  0  0  0  0]
 [ 0  0  0  1  0  1]
 [ 1  1  0  0  0  0]
 [ 1  0  0  0  0  5]]
accuracy is: 0.5581395348837209
              precision    recall  f1-score   support

           1       0.35      0.89      0.50         9
           2       0.77      0.53      0.62        19
           3       0.00      0.00      0.00         5
           5       1.00      0.50      0.67         2
           6       0.00      0.00      0.00         2
           7       0.83      0.83      0.83         6

    accuracy                           0.56        43
   macro avg       0.49      0.46      0.44        43
weighted avg       0.58      0.56      0.53        43


C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



##Note:after hyperparameter tuning accuracy of KNeighborsClassifier increased by 0.44 to 0.56

DecisionTreeClassifier

In [45]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [46]:

```
model=DecisionTreeClassifier()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

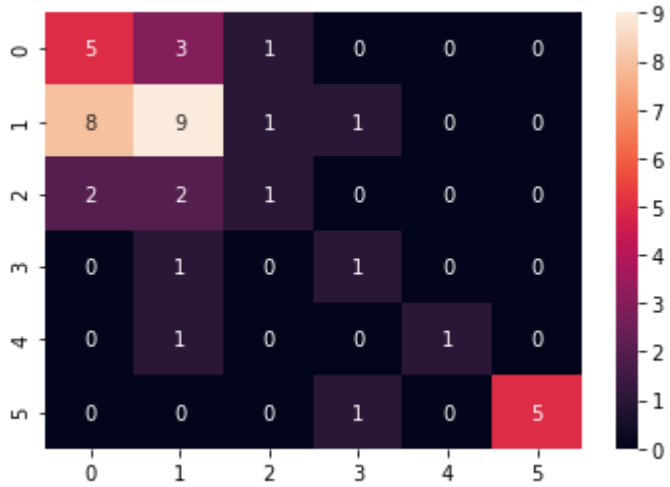
In [47]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.5116279069767442
              precision    recall  f1-score   support

           1       0.33      0.56      0.42         9
           2       0.56      0.47      0.51        19
           3       0.33      0.20      0.25         5
           5       0.33      0.50      0.40         2
           6       1.00      0.50      0.67         2
           7       1.00      0.83      0.91         6

    accuracy                           0.51        43
   macro avg       0.59      0.51      0.53        43
weighted avg       0.56      0.51      0.52        43
```



In [48]:

```
#decisiontreeclassifier on scaled data
```

In [49]:

```
model=DecisionTreeClassifier()
model.fit(sc_xtrain,ytrain)
ypred=model.predict(sc_xtest)
```

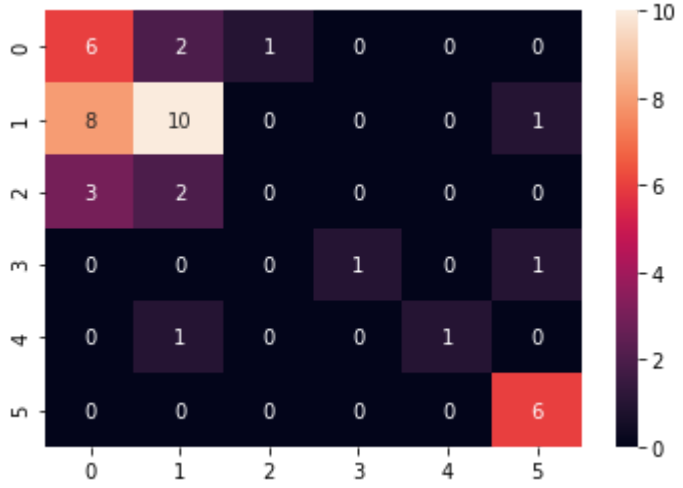
In [50]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.5581395348837209
              precision    recall  f1-score   support

           1       0.35      0.67      0.46         9
           2       0.67      0.53      0.59        19
           3       0.00      0.00      0.00         5
           5       1.00      0.50      0.67         2
           6       1.00      0.50      0.67         2
           7       0.75      1.00      0.86         6

    accuracy                           0.56        43
   macro avg       0.63      0.53      0.54        43
weighted avg       0.57      0.56      0.54        43
```



Note:accuracy of decisiontreeclassifier before tuning the parameter is 0.56

hyperparameter tuning

In [51]:

```
help(DecisionTreeClassifier())
```

```
Help on DecisionTreeClassifier in module sklearn.tree._classes object:

class DecisionTreeClassifier(sklearn.base.ClassifierMixin, BaseDecisionT
ree)
 |  DecisionTreeClassifier(*, criterion='gini', splitter='best', max_dep
th=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_le
af=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_i
mpurity_decrease=0.0, min_impurity_split=None, class_weight=None, presor
t='deprecated', ccp_alpha=0.0)
 |
 |  A decision tree classifier.
 |
 |  Read more in the :ref:`User Guide <tree>`.
 |
 |  Parameters
 |  ----------
 |  criterion : {"gini", "entropy"}, default="gini"
 |      The function to measure the quality of a split. Supported criter
ia are
```

In [52]:

```
model=DecisionTreeClassifier()
#parameters
splitter=["best", "random"]
criterion=["gini", "entropy"]
max_features=["auto", "sqrt", "log2"]
grid=dict(splitter=splitter,criterion=criterion,max_features=max_features)
#cv
from sklearn.model_selection import RepeatedStratifiedKFold
cv=RepeatedStratifiedKFold(n_splits=5,n_repeats=3,random_state=0)
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(estimator=model,param_grid=grid,cv=cv,scoring="accuracy")
res=grid_cv.fit(xtrain,ytrain)
```

In [53]:

```
res.best_params_
```

Out[53]:

```
{'criterion': 'gini', 'max_features': 'log2', 'splitter': 'best'}
```

In [54]:

```
res.best_score_
```

Out[54]:

```
0.7120448179271707
```

In [55]:

```
model=DecisionTreeClassifier(criterion='gini',max_features='auto',splitter='best')
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

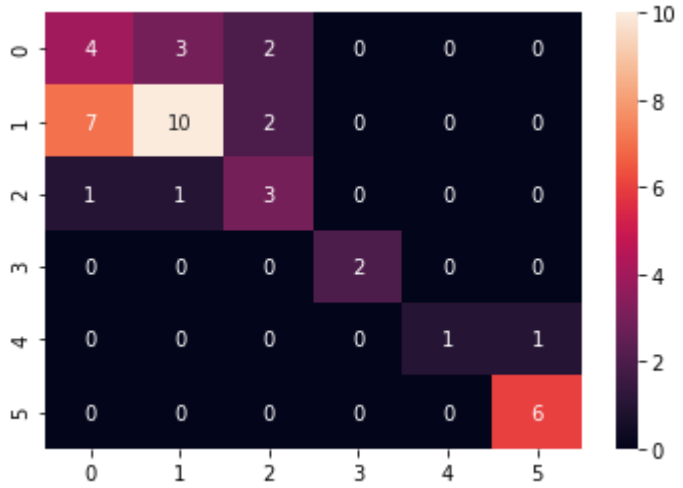
In [56]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.6046511627906976
              precision    recall  f1-score   support

           1       0.33      0.44      0.38         9
           2       0.71      0.53      0.61        19
           3       0.43      0.60      0.50         5
           5       1.00      1.00      1.00         2
           6       1.00      0.50      0.67         2
           7       0.86      1.00      0.92         6

    accuracy                           0.60        43
   macro avg       0.72      0.68      0.68        43
weighted avg       0.65      0.60      0.61        43
```



Noyte:accuracy of decesiontreeclassifier after tuning the hyperparameter is 0.60

RandomForestClassifier

In [57]:

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()
model.fit(xtrain,ytrain)
```

Out[57]:

```
RandomForestClassifier()
```

In [58]:

```
help(RandomForestClassifier())
```

```
Help on RandomForestClassifier in module sklearn.ensemble._forest objec
t:

class RandomForestClassifier(ForestClassifier)
 |  RandomForestClassifier(n_estimators=100, *, criterion='gini', max_de
pth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_l
eaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease
=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=N
one, random_state=None, verbose=0, warm_start=False, class_weight=None,
ccp_alpha=0.0, max_samples=None)
 |
 |  A random forest classifier.
 |
 |  A random forest is a meta estimator that fits a number of decision t
ree
 |  classifiers on various sub-samples of the dataset and uses averaging
to
 |  improve the predictive accuracy and control over-fitting.
 |  The sub-sample size is controlled with the `max samples` parameter i
```

In [59]:

```
ypred=model.predict(xtest)
```

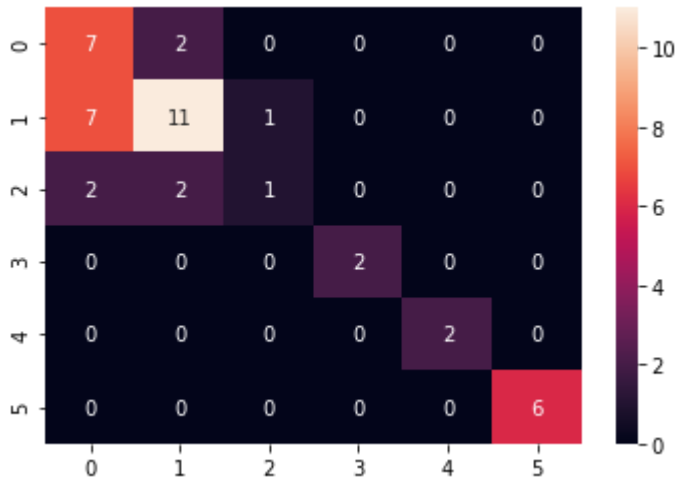
In [60]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.6744186046511628
              precision    recall  f1-score   support

           1       0.44      0.78      0.56         9
           2       0.73      0.58      0.65        19
           3       0.50      0.20      0.29         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         6

    accuracy                           0.67        43
   macro avg       0.78      0.76      0.75        43
weighted avg       0.71      0.67      0.67        43
```



Note:accuracy of RandomForestClassifier before tuning the hyperparameter is 0.67

Hyperparameter Tuning

In [61]:

```
model=RandomForestClassifier()
#parameters
criterion=["gini", "entropy"]
n_estimators=[5,50,100]
max_features=["auto", "sqrt", "log2"]
class_weight=["balanced", "balanced_subsample"]
grid=dict(n_estimators=n_estimators,criterion=criterion,max_features=max_features,c
#cv
from sklearn.model_selection import RepeatedStratifiedKFold
cv=RepeatedStratifiedKFold(n_splits=5,n_repeats=3,random_state=0)
#GridSearchCV
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(estimator=model,param_grid=grid,cv=cv,scoring="accuracy")
res=grid_cv.fit(xtrain,ytrain)
print(res.best_params_)
print(res.best_score_)
```

```
{'class_weight': 'balanced_subsample', 'criterion': 'gini', 'max_feature
s': 'auto', 'n_estimators': 50}
0.8015686274509805
```

In [62]:

```
model=RandomForestClassifier(class_weight='balanced_subsample', criterion= 'gini',
model.fit(xtrain,ytrain)
```

Out[62]:

```
RandomForestClassifier(class_weight='balanced_subsample', max_features='sq
rt')
```

In [63]:

```
ypred=model.predict(xtest)
```

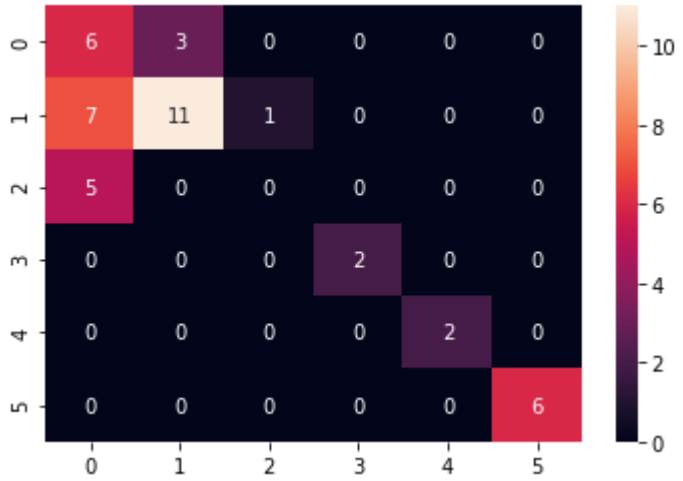
In [64]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.627906976744186
              precision    recall  f1-score   support

           1       0.33      0.67      0.44         9
           2       0.79      0.58      0.67        19
           3       0.00      0.00      0.00         5
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         6

    accuracy                           0.63        43
   macro avg       0.69      0.71      0.69        43
weighted avg       0.65      0.63      0.62        43
```



Note:accuracy of RandomForestClassifier after tuning the hyperparameter is 0.63

SVM

In [65]:

```
from sklearn.svm import SVC
```

In [66]:

```
model=SVC()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

Evaluation

In [67]:

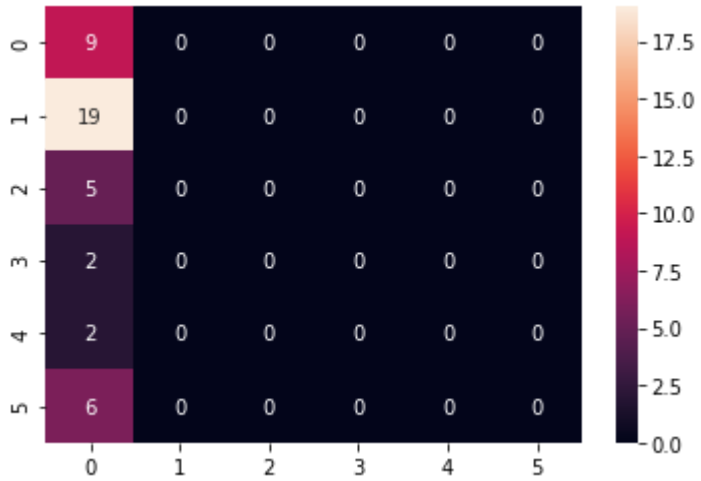
```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.20930232558139536
              precision    recall  f1-score   support

           1       0.21      1.00      0.35         9
           2       0.00      0.00      0.00        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         6

    accuracy                           0.21        43
   macro avg       0.03      0.17      0.06        43
weighted avg       0.04      0.21      0.07        43
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



Note:accuracy of SVC model is 0.21 which not better for classification

GaussianNB

In [68]:

```
from sklearn.naive_bayes import GaussianNB
```

In [69]:

```
model=GaussianNB()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

Evaluation

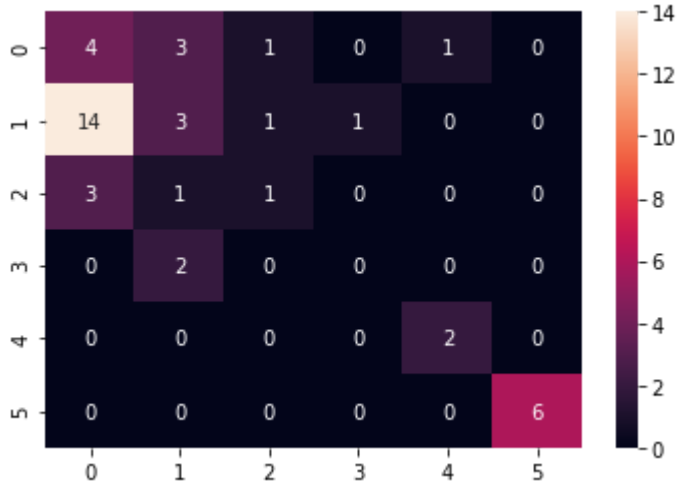
In [70]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.37209302325581395
              precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6

    accuracy                           0.37        43
   macro avg       0.42      0.47      0.42        43
weighted avg       0.40      0.37      0.36        43
```



Note:accuracy of model is 0.37

MultinomialNB

In [71]:

```
from sklearn.naive_bayes import MultinomialNB
model=MultinomialNB()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
```

Evaluation

In [72]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.37209302325581395
              precision    recall  f1-score   support

           1       0.28      0.89      0.42         9
           2       0.40      0.11      0.17        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.67      1.00      0.80         6

    accuracy                           0.37        43
   macro avg       0.22      0.33      0.23        43
weighted avg       0.33      0.37      0.27        43


C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Note:accuracy of model is 0.37

AdaBoostClassifier

In [73]:

```
from sklearn.ensemble import AdaBoostClassifier
model=AdaBoostClassifier(n_estimators=100)
model.fit(xtrain,ytrain)
```

Out[73]:

```
AdaBoostClassifier(n_estimators=100)
```

In [74]:

```
ypred=model.predict(xtest)
```

Evaluation

In [75]:

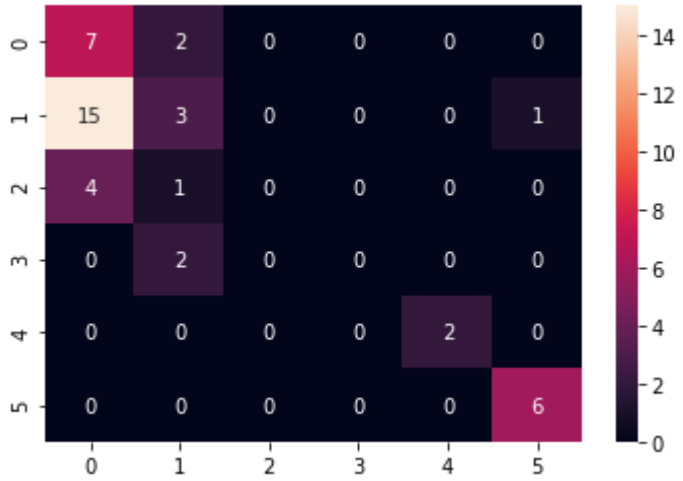
```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.4186046511627907
              precision    recall  f1-score   support

           1       0.27      0.78      0.40         9
           2       0.38      0.16      0.22        19
           3       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         2
           6       1.00      1.00      1.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.42        43
   macro avg       0.42      0.49      0.42        43
weighted avg       0.39      0.42      0.36        43


C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



Note:accuracy of AdaBoostClassifier is 0.42

GradientBoostingClassifier

In [76]:

```
from sklearn.ensemble import GradientBoostingClassifier
model=GradientBoostingClassifier()
model.fit(xtrain,ytrain)
```

Out[76]:

```
GradientBoostingClassifier()
```

In [77]:

```
help(GradientBoostingClassifier())
```

```
Help on GradientBoostingClassifier in module sklearn.ensemble._gb objec
t:

class GradientBoostingClassifier(sklearn.base.ClassifierMixin, BaseGradi
entBoosting)
 |  GradientBoostingClassifier(*, loss='deviance', learning_rate=0.1, n_
estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_spl
it=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min
_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state
=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=Fal
se, presort='deprecated', validation_fraction=0.1, n_iter_no_change=Non
e, tol=0.0001, ccp_alpha=0.0)
 |
 |  Gradient Boosting for classification.
 |
 |  GB builds an additive model in a
 |  forward stage-wise fashion; it allows for the optimization of
 |  arbitrary differentiable loss functions. In each stage ``n_classes_`
`
```

In [78]:

```
ypred=model.predict(xtest)
```

Evaluation

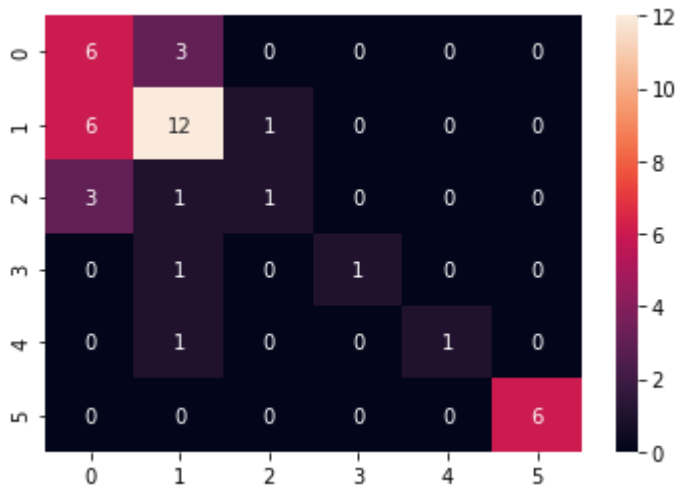
In [79]:

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.627906976744186
              precision    recall  f1-score   support

           1       0.40      0.67      0.50         9
           2       0.67      0.63      0.65        19
           3       0.50      0.20      0.29         5
           5       1.00      0.50      0.67         2
           6       1.00      0.50      0.67         2
           7       1.00      1.00      1.00         6

    accuracy                           0.63        43
   macro avg       0.76      0.58      0.63        43
weighted avg       0.67      0.63      0.63        43
```



Note:accuracy of GradientBoostingClassifier before tuning the hyperparameter is 0.63

Hyperparameter Tuning

In [80]:

```
model=GradientBoostingClassifier()
#parameters
loss=['deviance', 'exponential']
learning_rate=[0.1,1]
criterion=['friedman_mse', 'mse', 'mae']
max_features=['auto', 'sqrt', 'log2']
#grid
grid=dict(loss=loss,learning_rate=learning_rate,criterion=criterion,max_features=ma
#cv
from sklearn.model_selection import RepeatedStratifiedKFold
cv=RepeatedStratifiedKFold(n_splits=5,n_repeats=3,random_state=1)
#GridSearchCV
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(estimator=model,param_grid=grid,cv=cv,scoring="accuracy")
res=grid_cv.fit(xtrain,ytrain)
print("best parameters are:",res.best_params_)
print("best score:",res.best_score_)
```

```
C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklear
n\model_selection\_validation.py:548: FitFailedWarning: Estimator fit fa
iled. The score on this train-test partition for these parameters will b
e set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\ensemble\_gb.py", line 441, in fit
    self._check_params()
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\ensemble\_gb.py", line 248, in _check_params
    self.loss_ = loss_class(self.n_classes_)
  File "C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-package
s\sklearn\ensemble\_gb_losses.py", line 792, in __init__
    raise ValueError("{0:s} requires 2 classes; got {1:d} class(es)"
ValueError: ExponentialLoss requires 2 classes; got 6 class(es)
```

In [86]:

```
model=GradientBoostingClassifier(criterion='friedman_mse',learning_rate= 0.1, loss=
model.fit(xtrain,ytrain)
```

Out[86]:

```
GradientBoostingClassifier(max_features='log2', random_state=0)
```

In [87]:

```
ypred=model.predict(xtest)
```

In [88]:

```
#evaluation
```

In [89]:

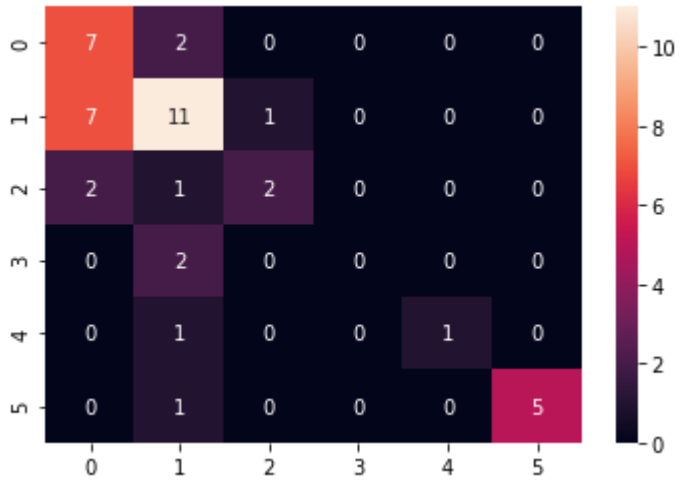
```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print("accuracy is:",accuracy_score(ytest,ypred))
print(classification_report(ytest,ypred))
```

```
accuracy is: 0.6046511627906976
              precision    recall  f1-score   support

           1       0.44      0.78      0.56         9
           2       0.61      0.58      0.59        19
           3       0.67      0.40      0.50         5
           5       0.00      0.00      0.00         2
           6       1.00      0.50      0.67         2
           7       1.00      0.83      0.91         6

    accuracy                           0.60        43
   macro avg       0.62      0.52      0.54        43
weighted avg       0.63      0.60      0.60        43


C:\Users\Expert\Documents\Desktop\soft\ANACONDA\lib\site-packages\sklearn
\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-
score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



Note:accuracy of GradientBoostingClassifier is decreased to 0.60 after hyperparameter tuning

In [94]:

```
model=["LogisticRegression","LogisticRegression after scaling","LogisticRegression
```

In [95]:

```
accuracy=[ 0.46511627906976744, 0.5116279069767442, 0.5581395348837209,0.4418604651
```

In [97]:

```
pd.DataFrame({"model":model,"Accuracy":accuracy})
```

Out[97]:

	model	Accuracy
0	LogisticRegression	0.465116
1	LogisticRegression after scaling	0.511628
2	LogisticRegression after tuning	0.558140
3	KNeighborsClassifier	0.441860
4	KNeighborsClassifier after tuning	0.558140
5	DecisionTreeClassifier	0.511628
6	DecisionTreeClassifier after tuning	0.604651
7	RandomForestClassifier	0.674419
8	RandomForestClassifier after tuning	0.627907
9	svm	0.209302
10	GaussianNB	0.372093
11	MultinomialNB	0.372093
12	AdaBoostClassifier	0.418605
13	GradientBoostingClassifier	0.627907
14	GradientBoostingClassifier after tuning	0.604651

Note:RandomForestClassifier is the best fit model with the accuracy 0.674419

In [ ]:

```

```