In [ ]:
```
```

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [4]:
```
df=pd.read_excel("Company_Data.csv.xlsx")
```

In [6]:
```
df.head()
```

Out[6]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban
0	9.50	138.0	73.0	11.0	276.0	120.0	Bad	42.0	17.0	Yes
1	11.22	111.0	48.0	16.0	260.0	83.0	Good	65.0	10.0	Yes
2	10.06	113.0	35.0	10.0	269.0	80.0	Medium	59.0	12.0	Yes
3	7.40	117.0	100.0	4.0	466.0	97.0	Medium	55.0	14.0	Yes
4	4.15	141.0	64.0	3.0	340.0	128.0	Bad	38.0	13.0	Yes

In [23]:
```
df.drop(['ShelveLoc','Urban','US'], axis=1 , inplace=True)
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    float64
 2   Income       400 non-null    float64
 3   Advertising  400 non-null    float64
 4   Population   400 non-null    float64
 5   Price        400 non-null    float64
 6   Age          400 non-null    float64
 7   Education    400 non-null    float64
dtypes: float64(8)
memory usage: 25.1 KB
```

In [25]:
```
df.isna().sum()
```

Out[25]:
```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
Age            0
Education      0
dtype: int64
```

In [36]:
```
x=df.drop(columns=["Education"])
y=df["Education"]
```

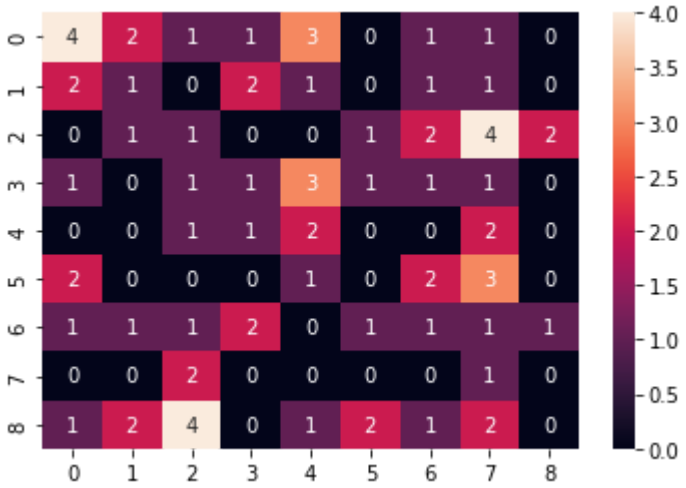
In [37]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [38]:
```
#Bagging Meta Estimator Classifier
from sklearn.ensemble import BaggingClassifier
#making the bagging classifier with 100 decision trees
model=BaggingClassifier(n_estimators=100)
#fitting data to bagging model
model.fit(xtrain,ytrain)
#testing on test dataset
ypred=model.predict(xtest)
#Model Evaluation
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print("accuracy is :",accuracy_score(ytest,ypred))
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print(classification_report(ytest,ypred))
```

```
accuracy is : 0.1375
              precision    recall  f1-score   support

        10.0       0.36      0.31      0.33        13
        11.0       0.14      0.12      0.13         8
        12.0       0.09      0.09      0.09        11
        13.0       0.14      0.11      0.12         9
        14.0       0.18      0.33      0.24         6
        15.0       0.00      0.00      0.00         8
        16.0       0.11      0.11      0.11         9
        17.0       0.06      0.33      0.11         3
        18.0       0.00      0.00      0.00        13

    accuracy                           0.14        80
   macro avg       0.12      0.16      0.13        80
weighted avg       0.13      0.14      0.13        80
```



Implementing The Bagging Classifier with Logistic Model

In [39]:
```
#Bagging Meta Estimator Classifier
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LogisticRegression
#making the bagging classifier with 100 Logistic Regression models
model=BaggingClassifier(base_estimator=LogisticRegression(),n_estimators=100)
#fitting data to bagging model
model.fit(xtrain,ytrain)
#testing on test dataset
ypred=model.predict(xtest)
#Model Evaluation
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print("accuracy is :",accuracy_score(ytest,ypred))
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print(classification_report(ytest,ypred))
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-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-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
```

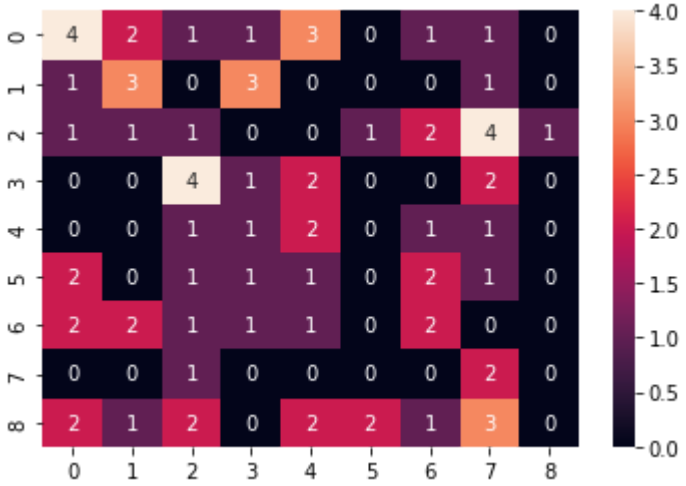
Random Forest Classifier

In [40]:
```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100)
#fitting data to Random Forest model
model.fit(xtrain,ytrain)
#testing on test dataset
ypred=model.predict(xtest)
#Model Evaluation
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print("accuracy is :",accuracy_score(ytest,ypred))
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True)
print(classification_report(ytest,ypred))
```

```
accuracy is : 0.1875
              precision    recall  f1-score   support

        10.0       0.33      0.31      0.32        13
        11.0       0.33      0.38      0.35         8
        12.0       0.08      0.09      0.09        11
        13.0       0.12      0.11      0.12         9
        14.0       0.18      0.33      0.24         6
        15.0       0.00      0.00      0.00         8
        16.0       0.22      0.22      0.22         9
        17.0       0.13      0.67      0.22         3
        18.0       0.00      0.00      0.00        13

    accuracy                           0.19        80
   macro avg       0.16      0.23      0.17        80
weighted avg       0.16      0.19      0.16        80
```



Tuning HyperParameters of Random Forest

In [42]:
```
#model
model=RandomForestClassifier()
n_estimators =[10,50,100,1000]
criterion =["gini", "entropy"]
max_features =["auto", "sqrt", "log2"]
#grid
grid=dict(n_estimators=n_estimators,criterion=criterion,max_features=max_features
#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')
#results
res=grid_cv.fit(xtrain,ytrain)
print("best parameters are :",res.best_params_)
print("best accuracy is :",res.best_score_)
```

```
best parameters are : {'criterion': 'entropy', 'max_features': 'auto', 'n_estim
ators': 10}
best accuracy is : 0.128125
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: