In [1]:

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

In [2]:

```
company_data=pd.read_csv('Company_Data.csv')
company_data
```

Out[2]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	U
0	9.50	138	73	11	276	120	Bad	42	17	
1	11.22	111	48	16	260	83	Good	65	10	
2	10.06	113	35	10	269	80	Medium	59	12	
3	7.40	117	100	4	466	97	Medium	55	14	
4	4.15	141	64	3	340	128	Bad	38	13	
...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14	
396	6.14	139	23	3	37	120	Medium	55	11	
397	7.41	162	26	12	368	159	Medium	40	18	
398	5.94	100	79	7	284	95	Bad	50	12	
399	9.71	134	37	0	27	120	Good	49	16	

400 rows × 11 columns

In [74]:

```
# Initial analysis
```

In [3]:

```
company_data.shape
```

Out[3]:

```
(400, 11)
```

In [4]:

```
company_data.dtypes
```

Out[4]:

```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc       object
Age              int64
Education        int64
Urban           object
US              object
dtype: object
```

In [5]:

```
company_data.isna().sum()
```

Out[5]:

```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

In [6]:

```
company_data.head()
```

Out[6]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urba
0	9.50	138	73	11	276	120	Bad	42	17	Y
1	11.22	111	48	16	260	83	Good	65	10	Y
2	10.06	113	35	10	269	80	Medium	59	12	Y
3	7.40	117	100	4	466	97	Medium	55	14	Y
4	4.15	141	64	3	340	128	Bad	38	13	Y

In [7]:

```
company_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    int64
 2   Income       400 non-null    int64
 3   Advertising  400 non-null    int64
 4   Population   400 non-null    int64
 5   Price        400 non-null    int64
 6   ShelveLoc    400 non-null    object
 7   Age          400 non-null    int64
 8   Education    400 non-null    int64
 9   Urban        400 non-null    object
 10  US           400 non-null    object
dtypes: float64(1), int64(7), object(3)
memory usage: 34.5+ KB
```

In [8]:

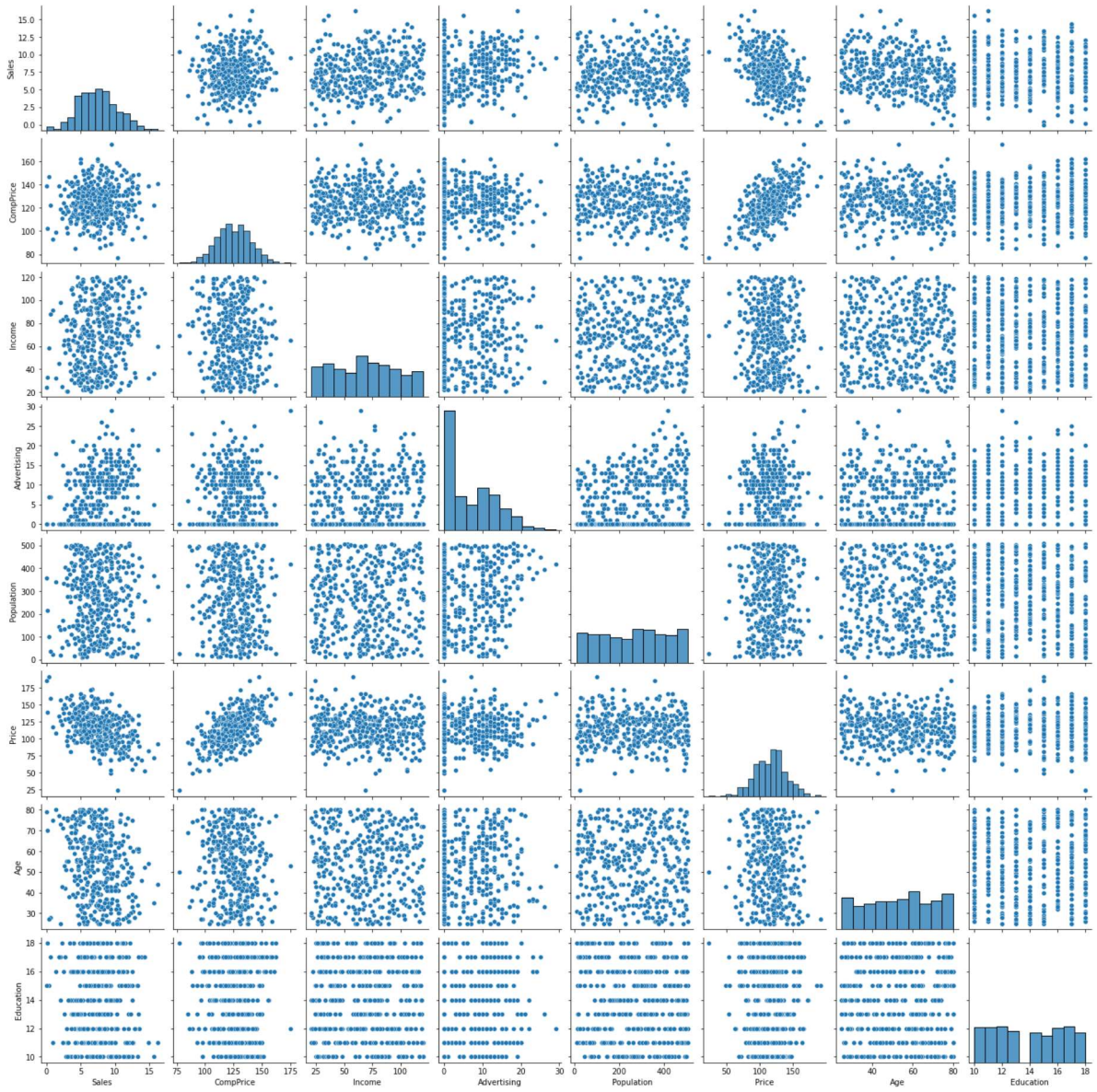
```
company_data.describe()
```

Out[8]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	E
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	40
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	1
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	1
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	1
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	1
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	1
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	1

In [9]:

```
import seaborn as sns
sns.pairplot(company_data)
plt.show()
```



In [12]:

```
round(company_data.corr(),3)
```

Out[12]:

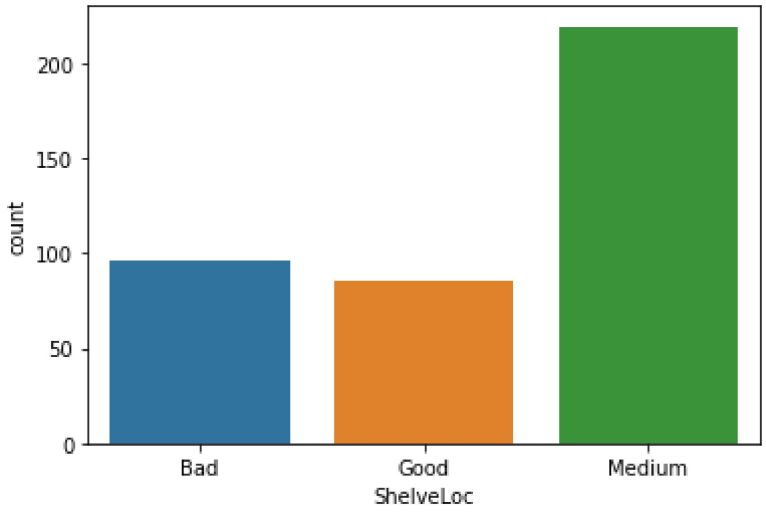
	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
Sales	1.000	0.064	0.152	0.270	0.050	-0.445	-0.232	-0.052
CompPrice	0.064	1.000	-0.081	-0.024	-0.095	0.585	-0.100	0.025
Income	0.152	-0.081	1.000	0.059	-0.008	-0.057	-0.005	-0.057
Advertising	0.270	-0.024	0.059	1.000	0.266	0.045	-0.005	-0.034
Population	0.050	-0.095	-0.008	0.266	1.000	-0.012	-0.043	-0.106
Price	-0.445	0.585	-0.057	0.045	-0.012	1.000	-0.102	0.012
Age	-0.232	-0.100	-0.005	-0.005	-0.043	-0.102	1.000	0.006
Education	-0.052	0.025	-0.057	-0.034	-0.106	0.012	0.006	1.000

In [11]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [14]:

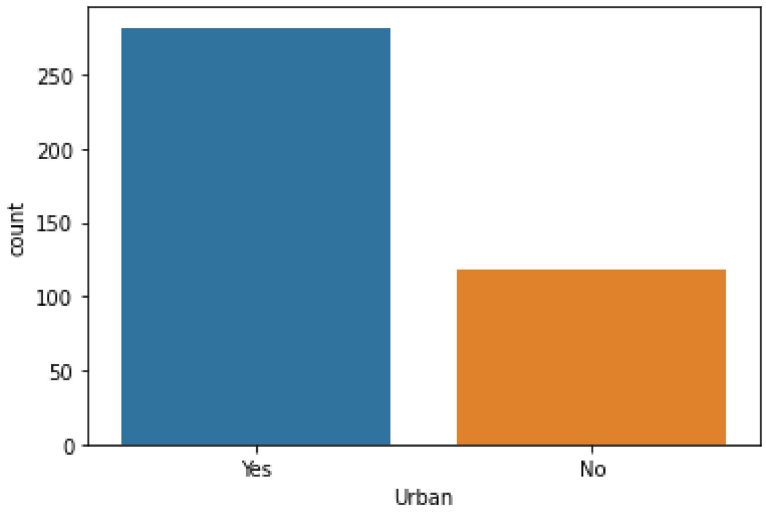
```
sns.countplot(company_data['ShelveLoc'])
plt.show()
```

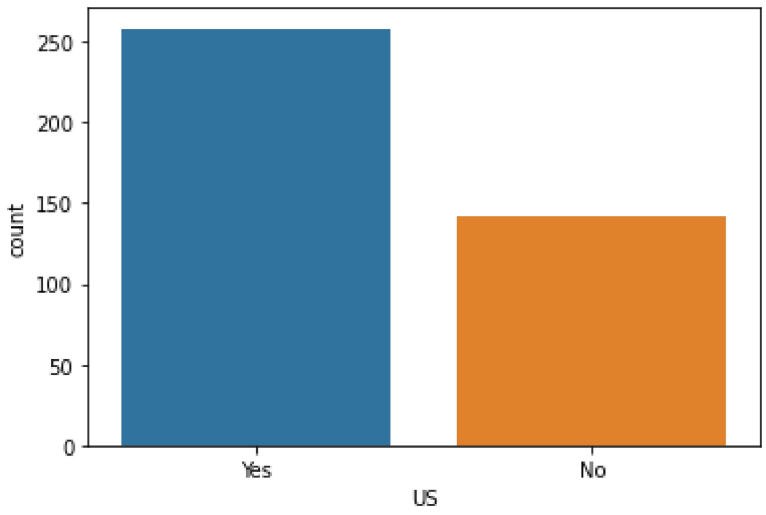


In [93]:

```
sns.countplot(company_data['Urban'])
plt.show()

sns.countplot(company_data['US'])
plt.show()
```





In [32]:

```
#encoding categorial data
```

In [35]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
company_data['Sales']=le.fit_transform(company_data['Sales'])
company_data['CompPrice']=le.fit_transform(company_data['CompPrice'])
company_data['Income']=le.fit_transform(company_data['Income'])
company_data['Advertising']=le.fit_transform(company_data['Advertising'])
company_data['Population']=le.fit_transform(company_data['Population'])
company_data['Price']=le.fit_transform(company_data['Price'])
company_data['ShelveLoc']=le.fit_transform(company_data['ShelveLoc'])
company_data['Age']=le.fit_transform(company_data['Age'])
company_data['Education']=le.fit_transform(company_data['Education'])
company_data['Urban']=le.fit_transform(company_data['Urban'])
company_data['US']=le.fit_transform(company_data['US'])
company_data
```

Out[35]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	U
0	2	0	0	2	2	2	0	1	0	
1	2	2	2	2	2	2	1	0	1	
2	2	2	1	2	2	2	2	2	1	
3	2	2	0	1	0	2	2	2	2	
4	1	0	2	1	0	2	0	1	2	
...	...	...	...	...	...	...	...	...	...	
395	0	0	0	2	2	2	1	1	2	
396	2	0	1	1	1	2	2	2	1	
397	2	0	1	2	0	0	2	1	0	
398	1	2	0	1	2	2	0	2	1	
399	2	0	1	1	1	2	1	2	0	

400 rows × 11 columns

Model Building

In [36]:

```
X =  company_data.drop(labels = 'Sales' , axis = 1)
y = company_data[['Sales']]
```

In [37]:

```
X
```

Out[37]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	U
0	0	0	2	2	2	0	1	0	1	
1	2	2	2	2	2	1	0	1	1	
2	2	1	2	2	2	2	2	1	1	
3	2	0	1	0	2	2	2	2	1	
4	0	2	1	0	2	0	1	2	1	
...	...	...	...	...	...	...	...	...	...	
395	0	0	2	2	2	1	1	2	1	
396	0	1	1	1	2	2	2	1	0	
397	0	1	2	0	0	2	1	0	1	
398	2	0	1	2	2	0	2	1	1	
399	0	1	1	1	2	1	2	0	1	

400 rows × 10 columns

In [38]:

```
y
```

Out[38]:

	Sales
0	2
1	2
2	2
3	2
4	1
...	...
395	0
396	2
397	2
398	1
399	2

400 rows × 1 columns

In [58]:

```
from sklearn.model_selection import train_test_split
X_train, X_test , y_train, y_test = train_test_split(X,y, test_size = 0.20, random_state=12
```

In [59]:

```
X_train.shape , y_train.shape
```

Out[59]:

```
((320, 10), (320, 1))
```

In [60]:

```
X_test.shape , y_test.shape
```

Out[60]:

```
((80, 10), (80, 1))
```

Building Model based on C5.0 Algorithm

In [66]:

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(criterion = 'entropy' , max_depth=3)
dt_model.fit(X_train,y_train)
```

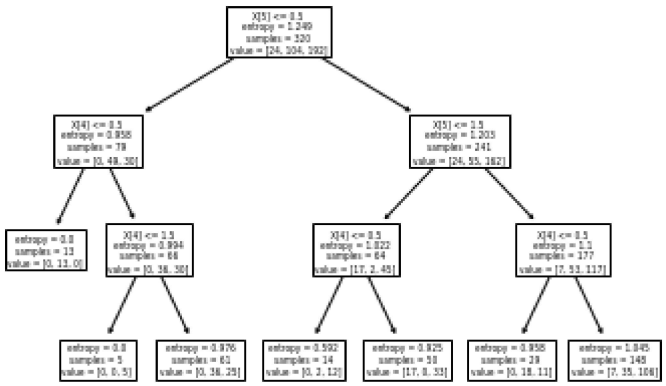
Out[66]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [67]:

```
# Plotting Decision Tree

from sklearn import tree
tree.plot_tree(dt_model);
```



In [68]:

```
y_train_pred = dt_model.predict(X_train)
y_train_pred
pd.Series(y_train_pred).value_counts()
```

Out[68]:

```
2    217
1    103
dtype: int64
```

In [69]:

```
y_test_pred = dt_model.predict(X_test)
y_test_pred
pd.Series(y_test_pred).value_counts()
```

Out[69]:

```
2    55
1    25
dtype: int64
```

In [70]:

```
dt_model.score(X_test,y_test)
```

Out[70]:

```
0.775
```

Building Model based on CART Algorithm

In [71]:

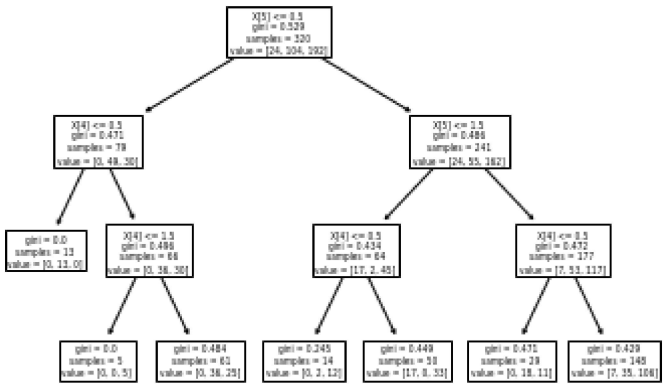
```
dt_model_cart = DecisionTreeClassifier(criterion = 'gini', max_depth=3)
dt_model_cart.fit(X_train,y_train)
```

Out[71]:

```
DecisionTreeClassifier(max_depth=3)
```

In [73]:

```
# Plotting Decision Tree
tree.plot_tree(dt_model_cart);
```



In [74]:

```
y_train_pred = dt_model_cart.predict(X_train)
pd.Series(y_train_pred).value_counts()
```

Out[74]:

```
2    217
1    103
dtype: int64
```

In [75]:

```
y_test_pred = dt_model_cart.predict(X_test)
pd.Series(y_test_pred).value_counts()
```

Out[75]:

```
2    55
1    25
dtype: int64
```

In [76]:

```
dt_model_cart.score(X_test,y_test)
```

Out[76]:

```
0.775
```

In [ ]:

```

```