In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
```

In [2]:

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

In [3]:

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

In [4]:

```
data.head()
```

Out[4]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
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

In [5]:

```
data.sample(10)
```

Out[5]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
343	5.99	117	42	10	371	121	Bad	26	14
209	3.02	98	21	11	326	90	Bad	76	11
346	8.97	132	107	0	144	125	Medium	33	13
394	5.35	130	58	19	366	139	Bad	33	16
396	6.14	139	23	3	37	120	Medium	55	11
51	4.42	121	90	0	150	108	Bad	75	16
231	8.09	132	69	0	123	122	Medium	27	11
164	8.22	148	64	0	58	141	Medium	27	13
364	10.50	122	21	16	488	131	Good	30	14
129	4.47	143	120	7	279	147	Bad	40	10

In [6]:

```
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 [7]:

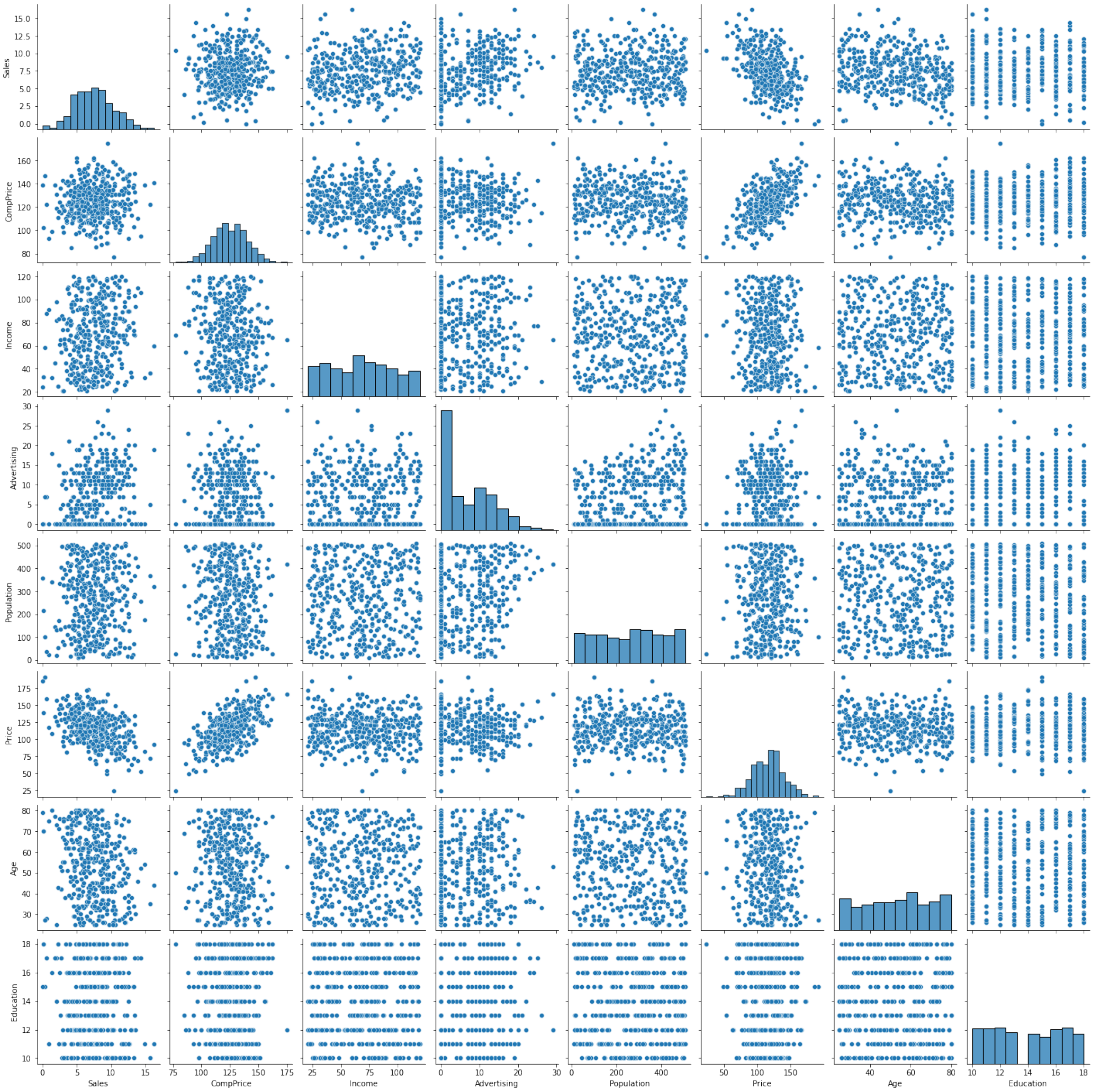
```
data.describe()
```

Out[7]:

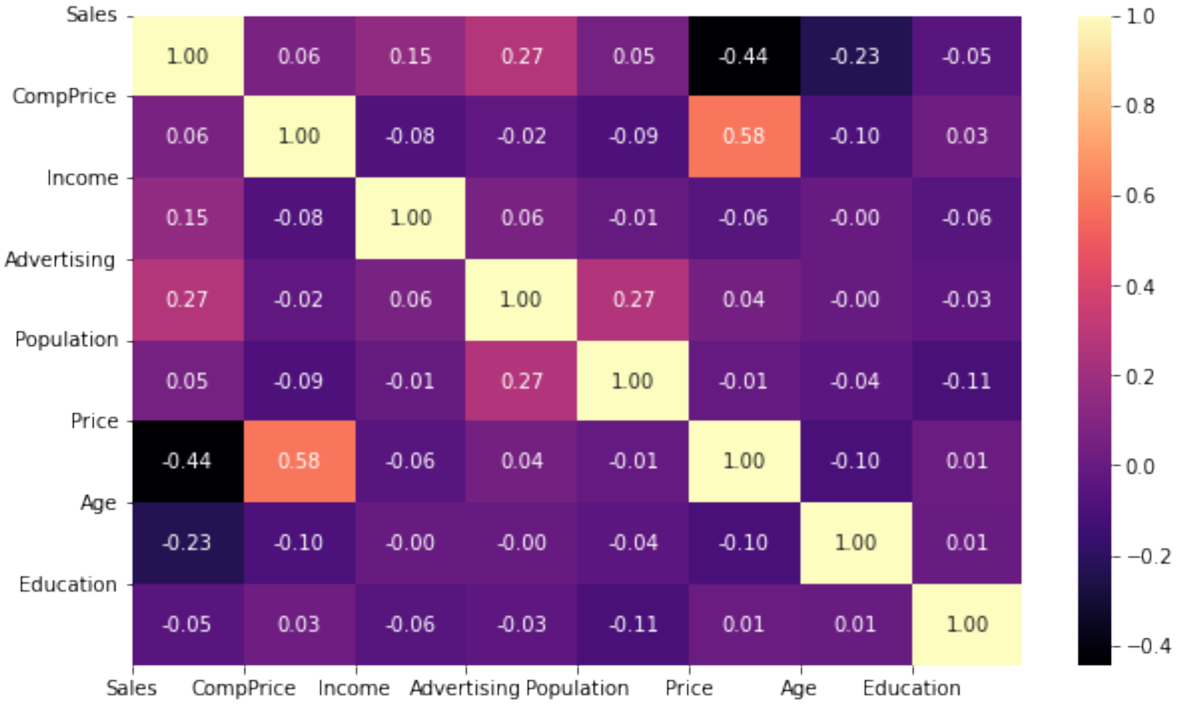
	Sales	CompPrice	Income	Advertising	Population	Price	Age
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297
min	0.000000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

```
In [8]: import seaborn as sns
        sns.pairplot(data)
```

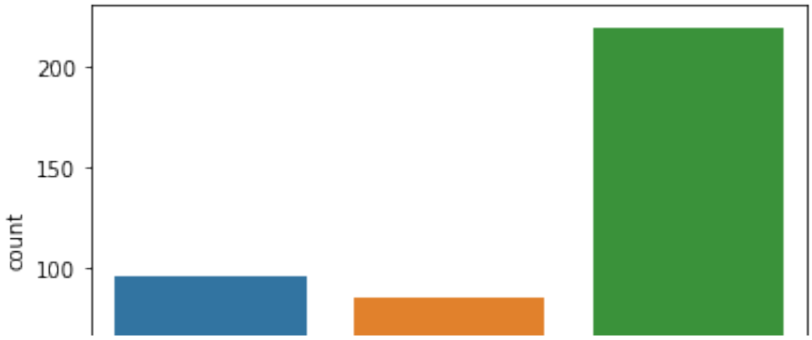
```
Out[8]: <seaborn.axisgrid.PairGrid at 0x7f9a7d290130>
```

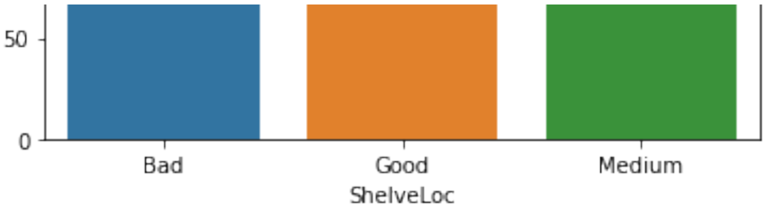


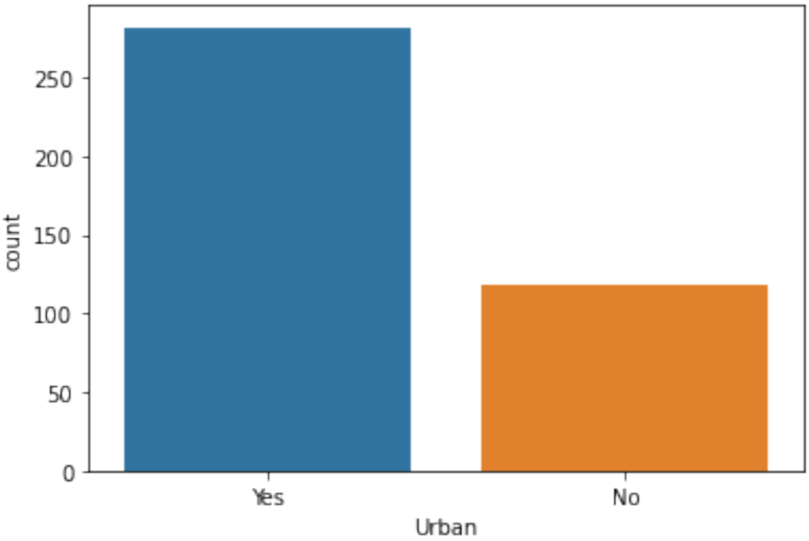
In [9]:
```
# Correlation analysis for data
corr = data.corr()

fig, ax = plt.subplots(figsize=(10, 6))

sns.heatmap(corr, cmap='magma', annot=True, fmt=".2f")

plt.xticks(range(len(corr.columns)), corr.columns);

plt.yticks(range(len(corr.columns)), corr.columns)

plt.show()
```

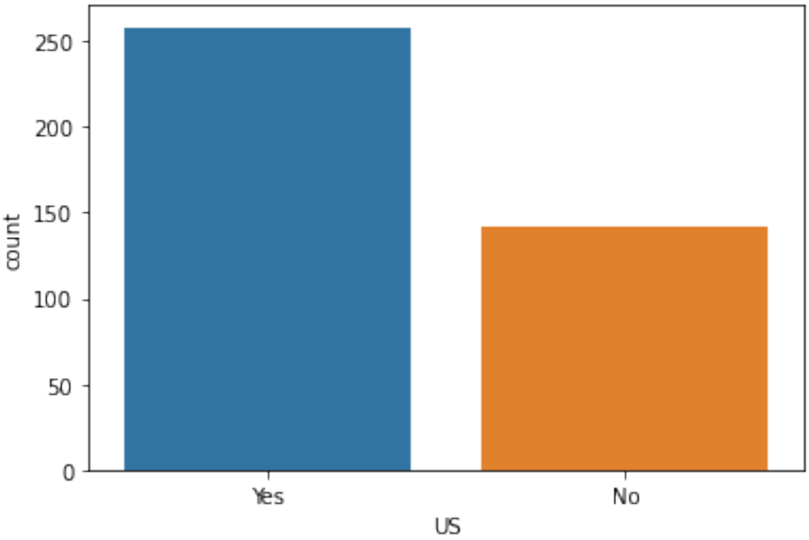


In [10]:
```
# checking count of categories for categorical columns colums
sns.countplot(data['ShelveLoc'])
plt.show()

sns.countplot(data['Urban'])
plt.show()

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




50
0
Bad
Good
Medium
ShelveLoc


count
250
200
150
100
50
0
Yes
No
Urban

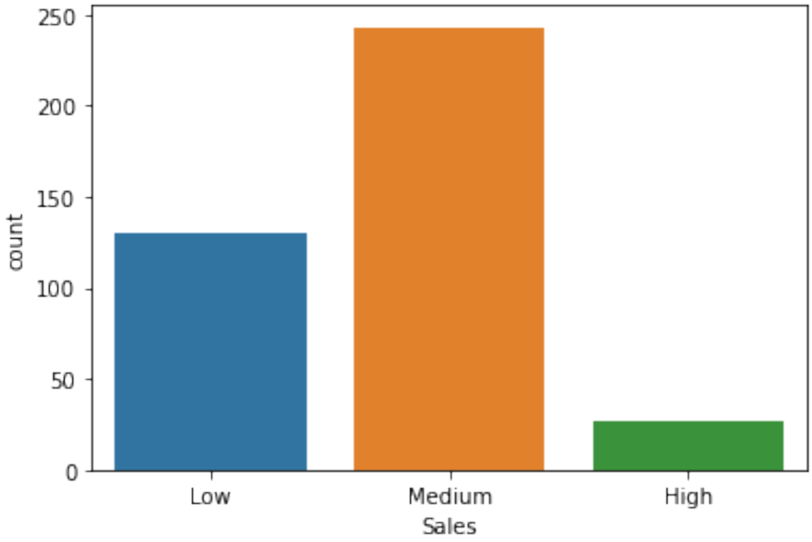

count
250
200
150
100
50
0
Yes
No
US

In [11]:
```
data['Sales'] = pd.cut(x=data['Sales'],bins=[0, 6, 12, 17], labels=
data['Sales']
```

Out[11]:
```
0      Medium
1      Medium
2      Medium
3      Medium
4         Low
        ...
395      High
396    Medium
397    Medium
398       Low
399    Medium
Name: Sales, Length: 400, dtype: category
Categories (3, object): ['Low' < 'Medium' < 'High']
```

In [12]:
```
sns.countplot(data['Sales'])
```

Out[12]:
```
<AxesSubplot:xlabel='Sales', ylabel='count'>
```



In [13]:
```
data['Sales'].value_counts()
```

Out[13]:
```
Medium    243
Low       130
High       27
Name: Sales, dtype: int64
```

In [14]:

```
# Converting other attributes into categories
data['CompPrice'] = pd.cut(x=data['CompPrice'],bins=[77, 100, 133,

data['Income'] = pd.cut(x=data['Income'],bins=[21, 46, 71, 121], la

data['Advertising'] = pd.cut(x=data['Advertising'],bins=[0, 10, 20,

data['Population'] = pd.cut(x=data['Population'],bins=[10, 170, 340

data['Price'] = pd.cut(x=data['Price'],bins=[24, 80, 136, 192], lab

data['Age'] = pd.cut(x=data['Age'],bins=[25, 45, 60, 81], labels=['

data['Education'] = pd.cut(x=data['Education'],bins=[10, 12.5, 15,
```

In [15]:

```
data.head()
```

Out[15]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Edu
0	Medium	High	High	Medium	Medium	Medium	Bad	Low	
1	Medium	Medium	Medium	Medium	Medium	Medium	Good	High	
2	Medium	Medium	Low	Medium	Medium	Medium	Medium	Medium	
3	Medium	Medium	High	Low	High	Medium	Medium	Medium	M
4	Low	High	Medium	Low	High	Medium	Bad	Low	M

In [16]:

```
#encoding categorical data
label_encoder = preprocessing.LabelEncoder()

data['Sales'] = label_encoder.fit_transform(data['Sales'])
data['CompPrice'] = label_encoder.fit_transform(data['CompPrice'])
data['Income'] = label_encoder.fit_transform(data['Income'])
data['Advertising'] = label_encoder.fit_transform(data['Advertising
data['Population'] = label_encoder.fit_transform(data['Population']
data['Price'] = label_encoder.fit_transform(data['Price'])
data['ShelveLoc'] = label_encoder.fit_transform(data['ShelveLoc'])
data['Age'] = label_encoder.fit_transform(data['Age'])
data['Education'] = label_encoder.fit_transform(data['Education'])
data['Urban'] = label_encoder.fit_transform(data['Urban'])
data['US'] = label_encoder.fit_transform(data['US'])

data
```

Out[16]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
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

In [17]:

```
# Dividing data into independent variables and dependent variable
X = data.drop('Sales', axis = 1)
y = data['Sales']
```

In [19]:
```
X
```

Out[19]:

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

400 rows × 10 columns

In [20]:
```
y
```

Out[20]:
```
0      2
1      2
2      2
3      2
4      1
      ..
395    0
396    2
397    2
398    1
399    2
Name: Sales, Length: 400, dtype: int64
```

In [21]:
```
# Splitting data into training and testing data
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size
```

In [22]:
```
x_train
```

Out[22]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urbai
258	2	1	1	2	2	0	0	2	(
177	0	0	1	1	2	2	1	0	'
119	2	0	1	1	2	2	0	1	'
194	2	0	2	0	2	2	2	1	'
229	1	0	1	0	1	2	1	0	(
...	...	...	...	...	...	...	...	...	..
71	0	2	2	1	0	2	2	0	(
106	2	1	1	2	0	2	0	0	(
270	2	1	1	2	2	1	1	1	'
348	2	0	0	0	2	1	2	1	'
102	2	1	1	1	2	2	0	0	(

268 rows × 10 columns

In [23]:
```
y_train
```

Out[23]:
```
258    1
177    2
119    2
194    2
229    2
      ..
71     2
106    1
270    2
348    0
102    1
Name: Sales, Length: 268, dtype: int64
```

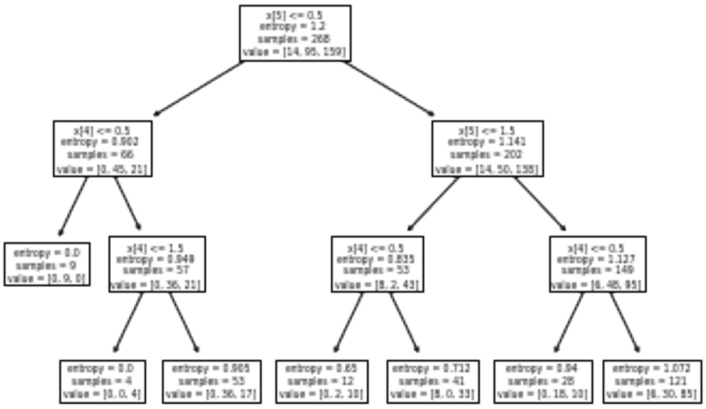
In [24]:
```
y_test
```

Out[24]:
```
209    1
280    1
33     2
210    1
93     2
      ..
332    1
167    2
245    2
311    2
145    2
Name: Sales, Length: 132, dtype: int64
```

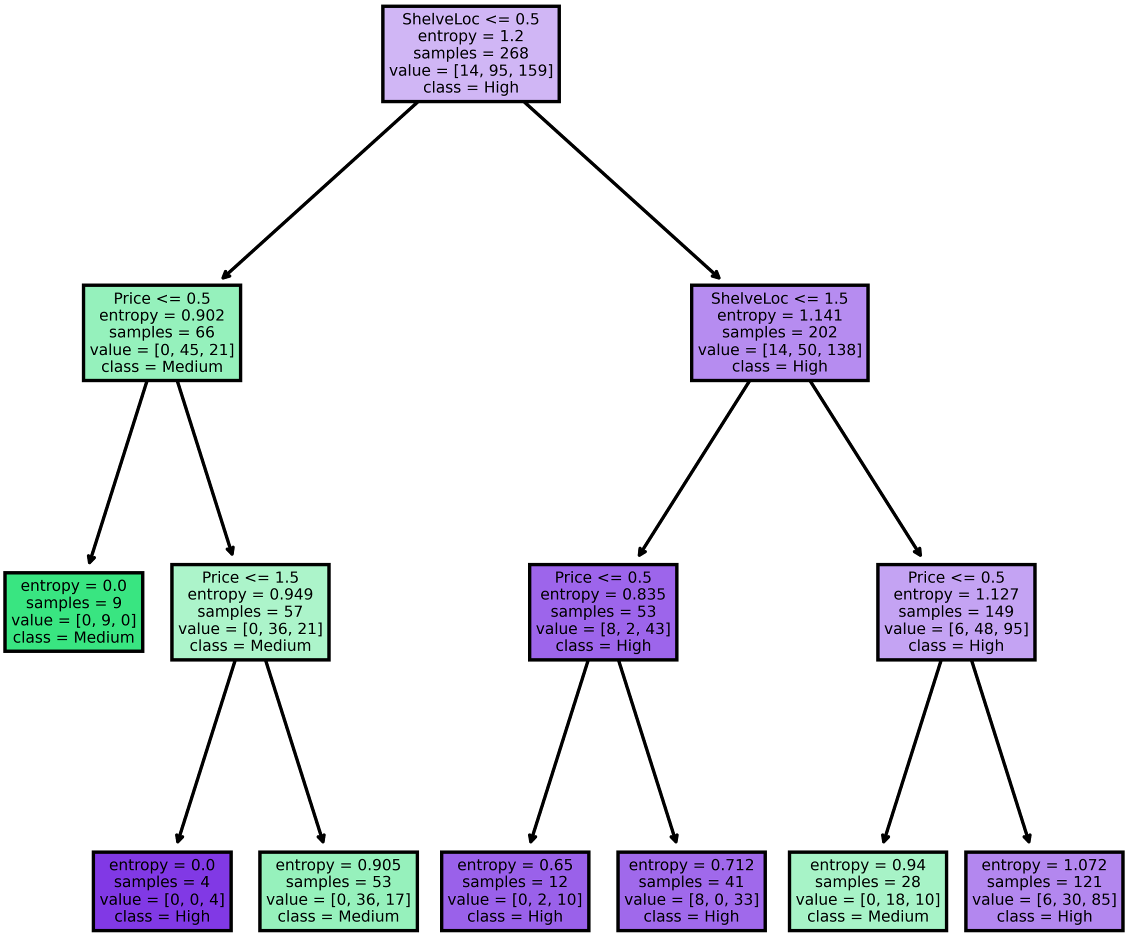
In [25]:
```
model_c5 = DecisionTreeClassifier(criterion = 'entropy', max_depth=
model_c5.fit(x_train, y_train)
```

Out[25]:
```
▾                  DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [26]:
```
# Plotting Decision tree
tree.plot_tree(model_c5);
```



In [27]:
```
fn=['CompPrice', 'Income', 'Advertising', 'Population', 'Price',
      'ShelveLoc', 'Age', 'Education', 'Urban', 'US']
cn=['Low', 'Medium', 'High']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (6,6), dpi=6
tree.plot_tree(model_c5,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [28]:
```
preds = model_c5.predict(x_test)
pd.Series(preds).value_counts()
```

Out[28]:
```
2    94
1    38
dtype: int64
```

In [29]:
```
preds
```

Out[29]:
```
array([1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 1,
       1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1,
2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1,
2, 2,
       2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1, 2, 2, 2,
2, 2,
       1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2,
2, 2,
       2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2,
1, 2])
```

In [30]:
```
pd.crosstab(y_test, preds)
```

Out[30]:

col_0	1	2
Sales		
0	0	13
1	22	13
2	16	68

In [31]:
```
# Checking accuracy of model
model_c5.score(x_test, y_test)
```

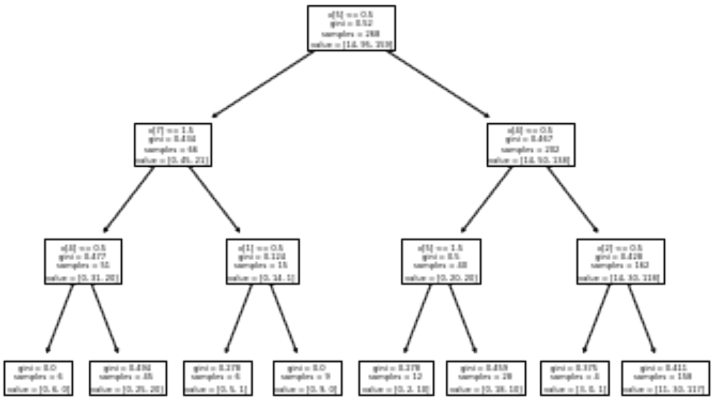
Out[31]:
```
0.6818181818181818
```

In [32]:
```
model_CART = DecisionTreeClassifier(criterion = 'gini', max_depth=
model_CART.fit(x_train, y_train)
```

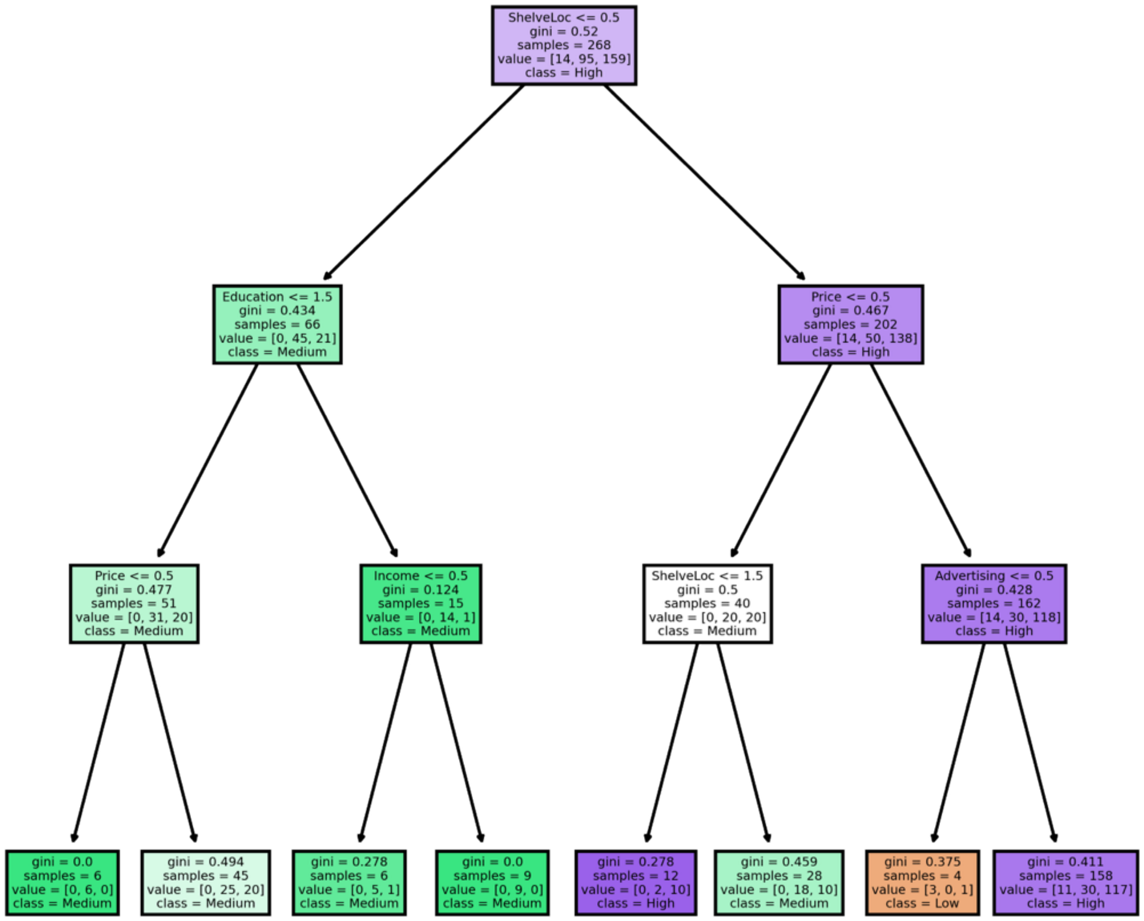
Out[32]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [33]:

```
# Plotting Decision tree
tree.plot_tree(model_CART);
```



In [34]:
```
fn=['CompPrice', 'Income', 'Advertising', 'Population', 'Price',
      'ShelveLoc', 'Age', 'Education', 'Urban', 'US']
cn=['Low', 'Medium', 'High']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (6,6), dpi=6
tree.plot_tree(model_CART,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [35]:
```
# Predicting Data
preds = model_CART.predict(x_test)
pd.Series(preds).value_counts()
```

Out[35]:
```
2    89
1    40
0     3
dtype: int64
```

In [36]:
```
preds
```

Out[36]:
```
array([1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 1,
       1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1,
2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1,
2, 2,
       2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 0, 1, 1, 2, 2, 1, 2, 2, 2,
2, 2,
       1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2,
2, 2,
       2, 2, 2, 1, 1, 1, 1, 2, 0, 2, 2, 2, 2, 2, 1, 2, 2, 0, 2, 2,
1, 2])
```

In [37]:
```
pd.crosstab(y_test, preds)
```

Out[37]:

col_0	0	1	2
Sales			
0	0	0	13
1	1	22	12
2	2	18	64

In [38]:
```
model_CART.score(x_test, y_test)
```

Out[38]:
```
0.6515151515151515
```

In [ ]: