In [60]:

```
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 [62]:

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

Out[62]:

	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 [63]:

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

Out[63]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	U
137	6.52	128	42	0	436	118	Medium	80	11	
120	6.87	128	105	11	249	131	Medium	63	13	
317	6.41	142	30	0	472	136	Good	80	15	
179	7.78	144	25	3	70	116	Medium	77	18	
100	4.11	113	69	11	94	106	Medium	76	12	
172	9.03	104	102	13	123	110	Good	35	16	
66	8.85	127	92	0	508	91	Medium	56	18	
285	7.60	146	26	11	261	131	Medium	39	10	
158	12.53	142	90	1	189	112	Good	39	10	
343	5.99	117	42	10	371	121	Bad	26	14	

In [64]:

```
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 [65]:

```
data.describe()
```

Out[65]:

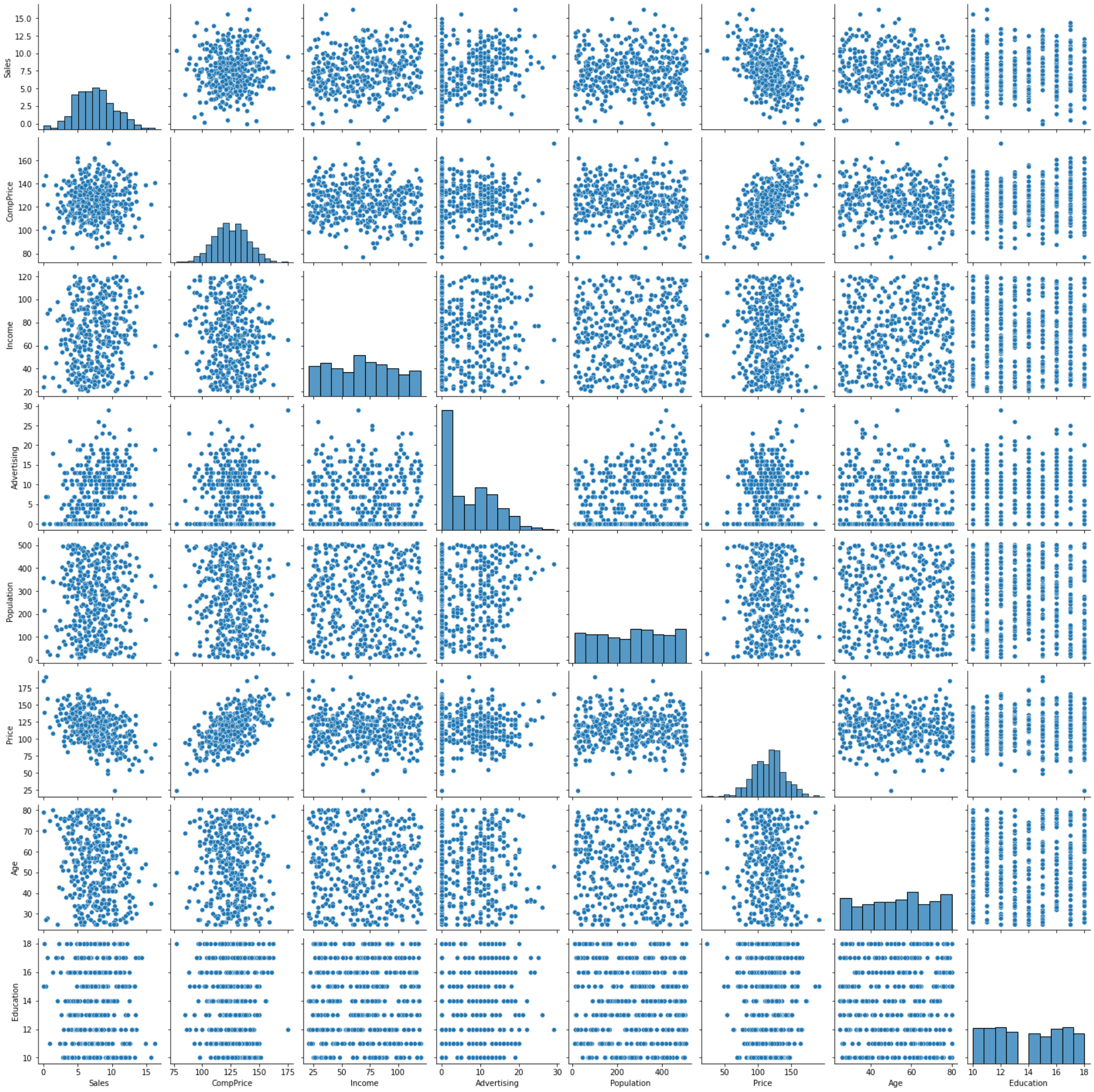
	Sales	CompPrice	Income	Advertising	Population	Price	Age	E
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	40
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	1
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	
min	0.000000	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 [66]:

```
# pairplot
import seaborn as sns
sns.pairplot(data)
```

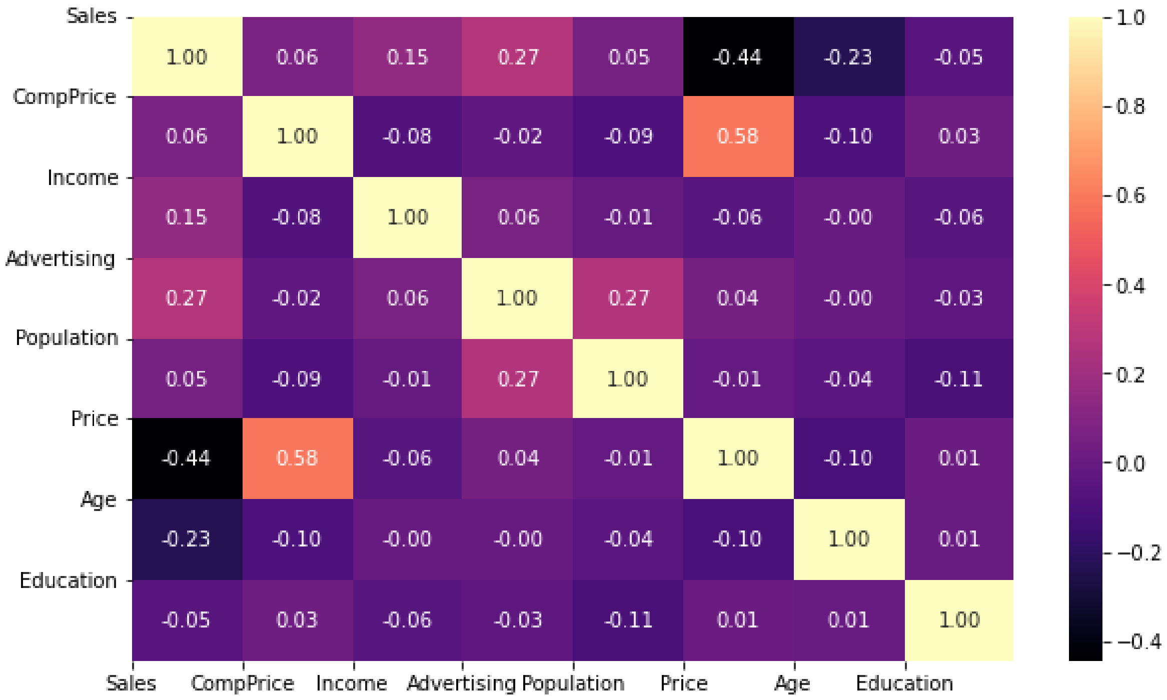
Out[66]:

```
<seaborn.axisgrid.PairGrid at 0x1d21fea6580>
```



In [67]:

```
# Correlation analysis for data
corr = data.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 6))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='magma', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```

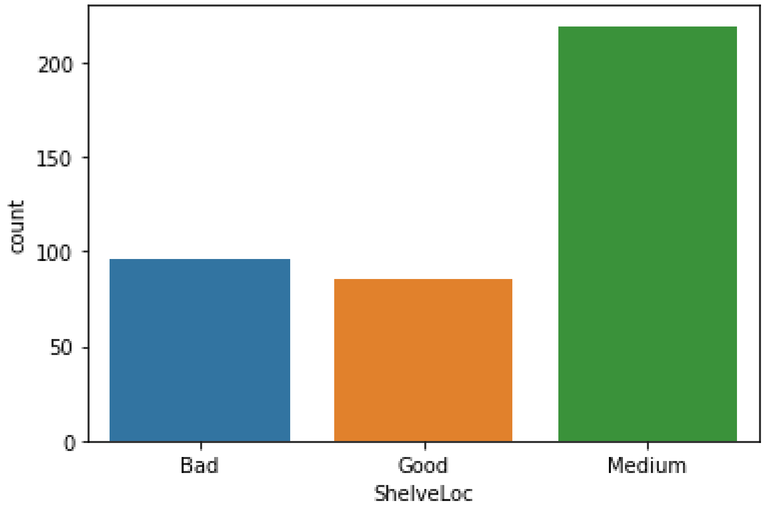


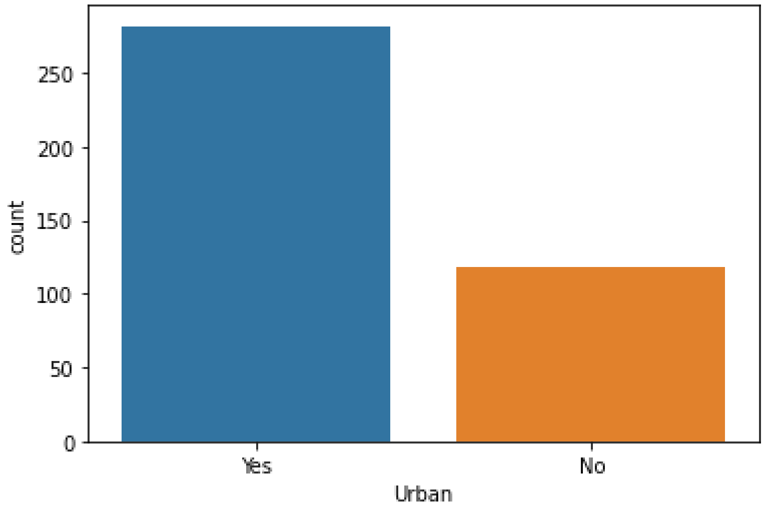
In [68]:

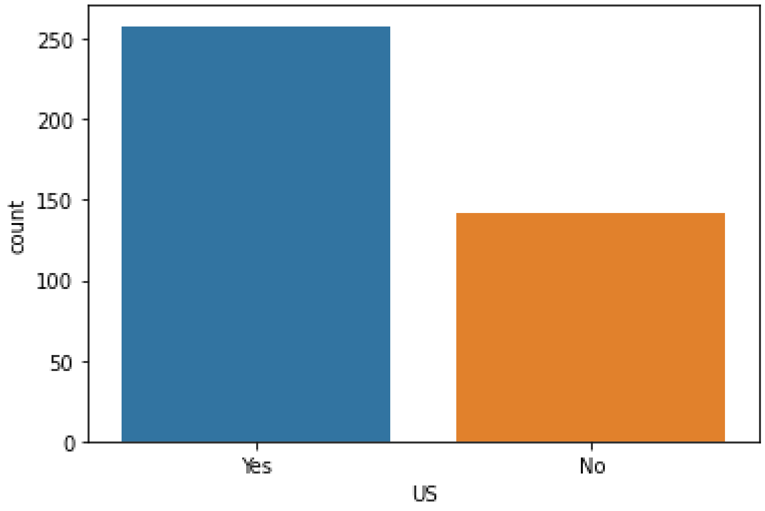
```
# checking count of categories for categorical columns colums
sns.countplot(data['ShelveLoc'])
plt.show()

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

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







In [69]:

```
# Converting Target variable 'Sales' into categories Low, Medium and High.
data['Sales'] = pd.cut(x=data['Sales'],bins=[0, 6, 12, 17], labels=['Low','Medium', 'High']
data['Sales']
```

Out[69]:

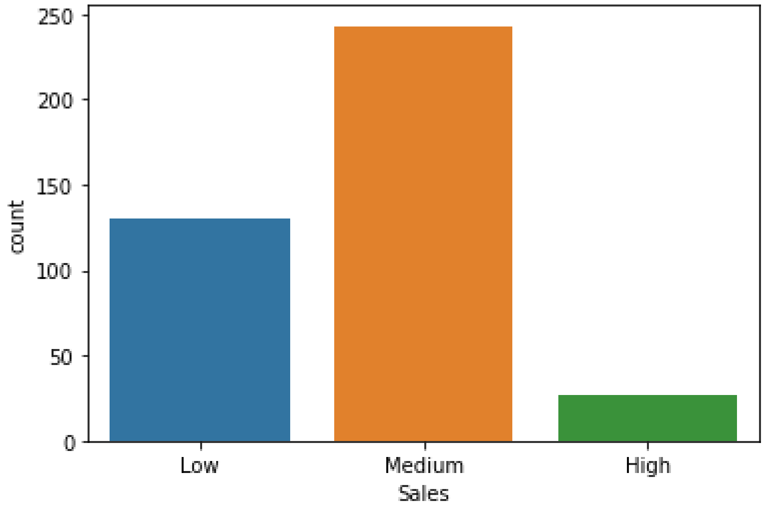
```
0      Medium
1      Medium
2      Medium
3      Medium
4         Low
        ...  
395      High
396    Medium
397    Medium
398       Low
399    Medium
Name: Sales, Length: 400, dtype: category
Categories (3, object): ['Low' < 'Medium' < 'High']
```

In [70]:

```
sns.countplot(data['Sales'])
```

Out[70]:

```
<AxesSubplot:xlabel='Sales', ylabel='count'>
```



In [71]:

```
data['Sales'].value_counts()
```

Out[71]:

```
Medium    243
Low       130
High       27
Name: Sales, dtype: int64
```

In [72]:

```
# Converting other attributes into categories
data['CompPrice'] = pd.cut(x=data['CompPrice'],bins=[77, 100, 133, 176], labels=['Low','Med

data['Income'] = pd.cut(x=data['Income'],bins=[21, 46, 71, 121], labels=['Low','Medium', 'H

data['Advertising'] = pd.cut(x=data['Advertising'],bins=[0, 10, 20, 30], labels=['Low','Med

data['Population'] = pd.cut(x=data['Population'],bins=[10, 170, 340, 510], labels=['Low','M

data['Price'] = pd.cut(x=data['Price'],bins=[24, 80, 136, 192], labels=['Low','Medium', 'Hi

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

data['Education'] = pd.cut(x=data['Education'],bins=[10, 12.5, 15, 19], labels=['Low','Medi
```

In [73]:

```
data.head()
```

Out[73]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Educatio
0	Medium	High	High	Medium	Medium	Medium	Bad	Low	Hig
1	Medium	Medium	Medium	Medium	Medium	Medium	Good	High	Lo
2	Medium	Medium	Low	Medium	Medium	Medium	Medium	Medium	Lo
3	Medium	Medium	High	Low	High	Medium	Medium	Medium	Mediu
4	Low	High	Medium	Low	High	Medium	Bad	Low	Mediu

In [74]:

```
#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[74]:

	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

In [75]:

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

In [76]:

```
X
```

Out[76]:

	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 [77]:

```
y
```

Out[77]:

```
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: int32
```

In [78]:

```
# Splitting data into training and testing data
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= 0.33, random_state= 42
```

In [79]:

```
x_train
```

Out[79]:

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

268 rows × 10 columns

In [80]:

```
y_train
```

Out[80]:

```
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: int32
```

In [81]:

```
y_test
```

Out[81]:

```
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: int32
```

In [82]:

```
# Building model based on C5.0 Algorithm
model_c5 = DecisionTreeClassifier(criterion = 'entropy', max_depth= 3)
model_c5.fit(x_train, y_train)
```

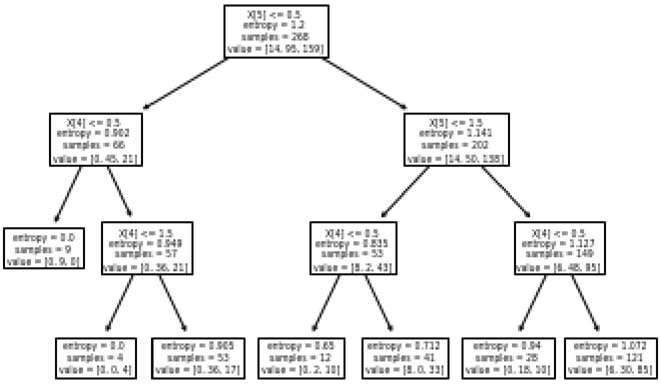
Out[82]:

```
▾                  DecisionTreeClassifier

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

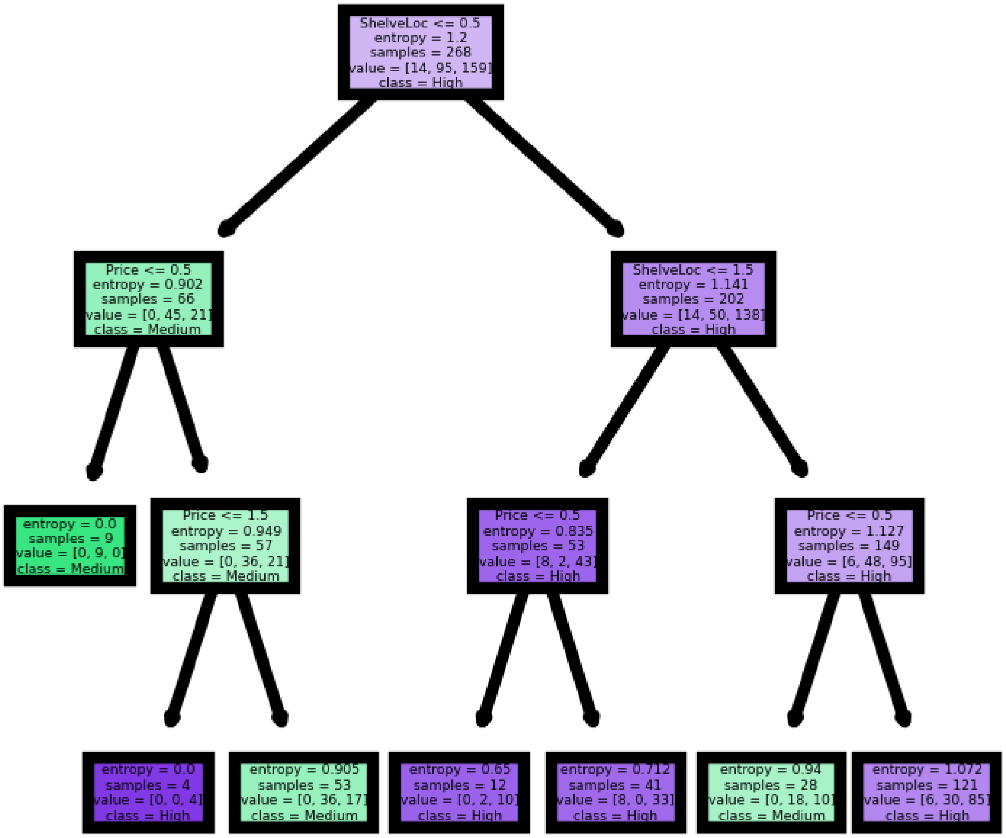
In [83]:

```
# Plotting Decision tree
tree.plot_tree(model_c5);
```



In [100]:

```
fn=['CompPrice', 'Income', 'Advertising', 'Population', 'Price',
       'ShelveLoc', 'Age', 'Education', 'Urban', 'US']
cn=['Low', 'Medium', 'High']
fig, axes = plt.subplots(nrows=1,ncols=1,figsize=(1.5,1.5),dpi=600)
tree.plot_tree(model_c5,feature_names=fn,class_names=cn,filled=True);
```



In [85]:

```
# Predicting Data
preds = model_c5.predict(x_test)
pd.Series(preds).value_counts()
```

Out[85]:

```
2    94
1    38
dtype: int64
```

In [86]:

```
preds
```

Out[86]:

```
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 [87]:

```
# Creating cross tables for checking model
pd.crosstab(y_test, preds)
```

Out[87]:

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

In [88]:

```
# Checking accuracy of model
model_c5.score(x_test, y_test)
```

Out[88]:

```
0.6818181818181818
```

In [89]:

```
# Building model based on CART Algorithm
model_CART = DecisionTreeClassifier(criterion = 'gini', max_depth= 3)
model_CART.fit(x_train, y_train)
```

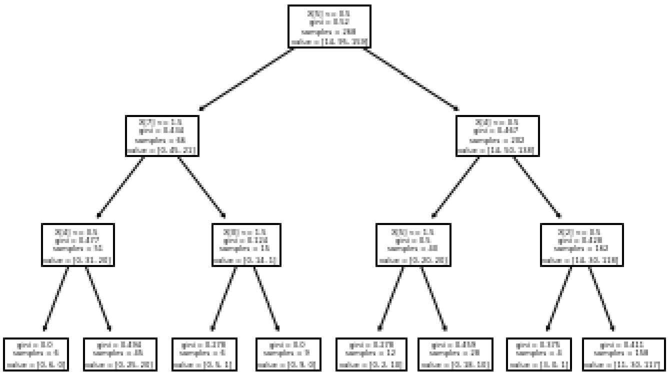
Out[89]:

```
▾      DecisionTreeClassifier

DecisionTreeClassifier(max_depth=3)
```

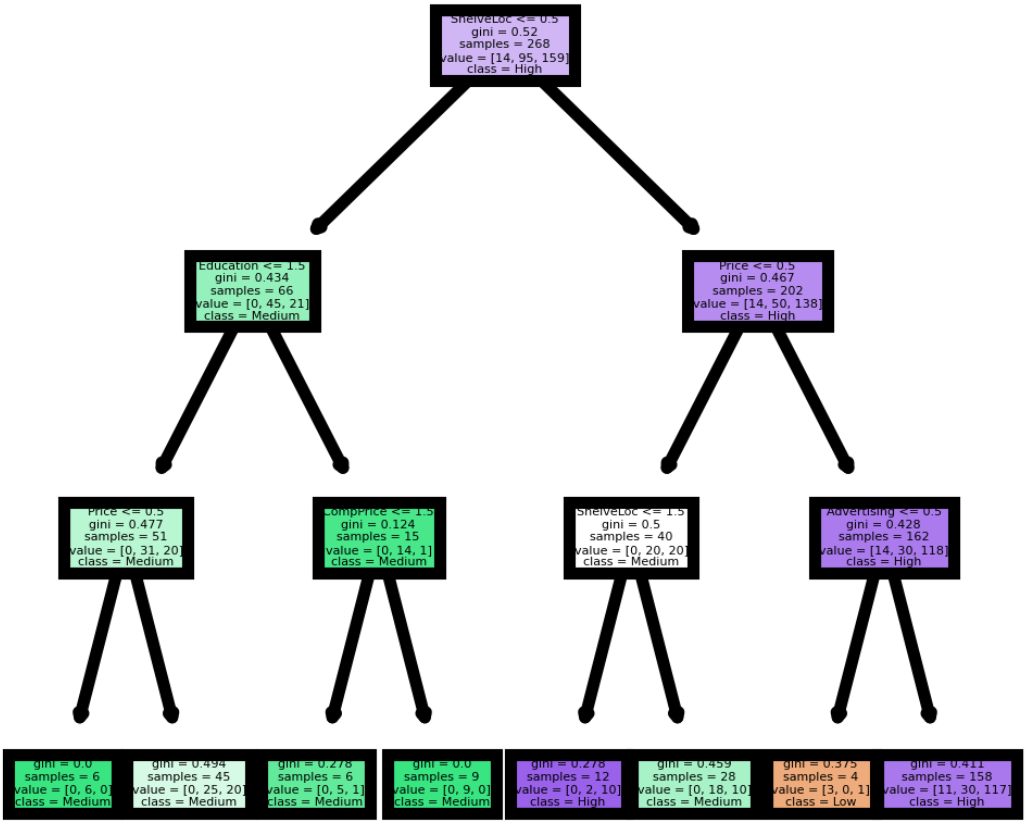
In [90]:

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



In [104]:

```
fn=['CompPrice', 'Income', 'Advertising', 'Population', 'Price',
       'ShelveLoc', 'Age', 'Education', 'Urban', 'US']
cn=['Low', 'Medium', 'High']
fig, axes = plt.subplots(nrows=1,ncols=1,figsize=(1.5,1.5),dpi=800)
tree.plot_tree(model_CART,feature_names=fn,class_names=cn,filled=True);
```



In [92]:

```
# Predicting Data
preds = model_CART.predict(x_test)
pd.Series(preds).value_counts()
```

Out[92]:

```
2    89
1    40
0     3
dtype: int64
```

In [93]:

```
preds
```

Out[93]:

```
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 [94]:

```
# Creating cross tables for checking model
pd.crosstab(y_test, preds)
```

Out[94]:

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

In [95]:

```
# Checking accuracy of model
model_CART.score(x_test, y_test)
```

Out[95]:

```
0.6515151515151515
```

In [ ]: