In [22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline#for encoding
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import plot_tree
```

```
UsageError: unrecognized arguments: encoding
```

In [23]:
```
df = pd.read_csv('Company_Data.csv')
df
```

Out[23]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urbar
0	9.50	138	73	11	276	120	Bad	42	17	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes
...	...	...	...	...	...	...	...	...	...	..
395	12.57	138	108	17	203	128	Good	33	14	Yes
396	6.14	139	23	3	37	120	Medium	55	11	No
397	7.41	162	26	12	368	159	Medium	40	18	Yes
398	5.94	100	79	7	284	95	Bad	50	12	Yes
399	9.71	134	37	0	27	120	Good	49	16	Yes

400 rows × 11 columns

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

Out[24]:

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

In [25]:
```
df.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 [26]:
```
df.shape
```

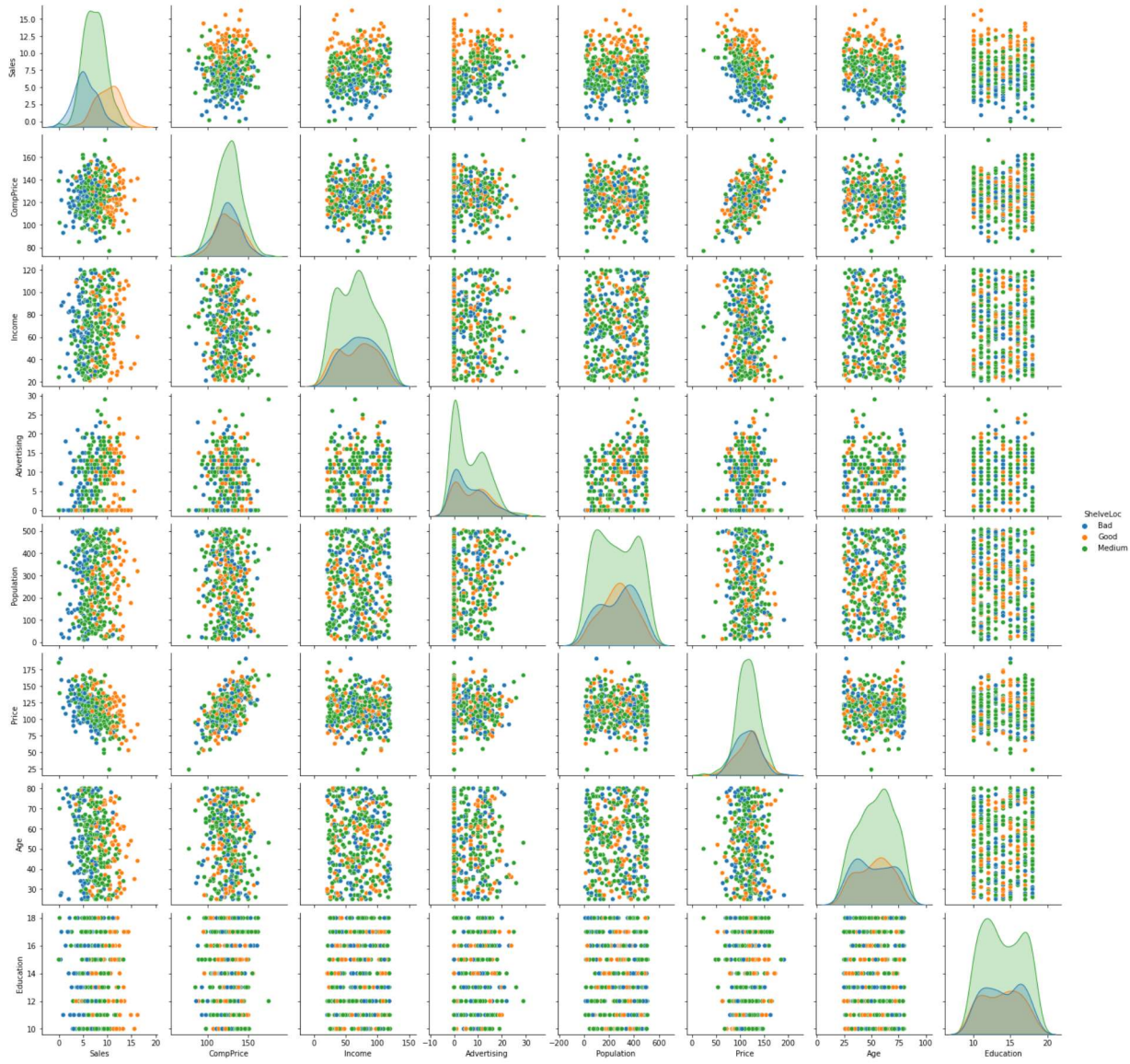
Out[26]:
```
(400, 11)
```

In [27]:
```
df.isnull().any()
```

Out[27]:
```
Sales          False
CompPrice      False
Income         False
Advertising    False
Population     False
Price          False
ShelveLoc      False
Age            False
Education      False
Urban          False
US             False
dtype: bool
```

In [28]:
```
# let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue = 'ShelveLoc')
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0x252999a1ca0>
```



In [29]:
```
#Creating dummy vairables dropping first dummy variable
df=pd.get_dummies(df,columns=['Urban','US'], drop_first=True)
```

In [30]:
```
print(df.head())
```

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

   Education  Urban_Yes  US_Yes
0         17          1       1
1         10          1       1
2         12          1       1
3         14          1       1
4         13          1       0
```

In [31]:
```
df.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_Yes    400 non-null    uint8
 10  US_Yes       400 non-null    uint8
dtypes: float64(1), int64(7), object(1), uint8(2)
memory usage: 29.0+ KB
```

In [32]:
```
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
```

In [33]:
```
df['ShelveLoc']=df['ShelveLoc'].map({'Good':1,'Medium':2,'Bad':3})
```

In [34]:
```
print(df.head())
```

```
   Sales  CompPrice  Income  Advertising  Population  Price  ShelveLoc  Age  \
0   9.50        138      73           11         276    120          3   42
1  11.22        111      48           16         260     83          1   65
2  10.06        113      35           10         269     80          2   59
3   7.40        117     100            4         466     97          2   55
4   4.15        141      64            3         340    128          3   38

   Education  Urban_Yes  US_Yes
0         17          1       1
1         10          1       1
2         12          1       1
3         14          1       1
4         13          1       0
```

In [35]:
```
x=df.iloc[:,0:6]
y=df['ShelveLoc']
```

In [36]:
```
x
```

Out[36]:

	Sales	CompPrice	Income	Advertising	Population	Price
0	9.50	138	73	11	276	120
1	11.22	111	48	16	260	83
2	10.06	113	35	10	269	80
3	7.40	117	100	4	466	97
4	4.15	141	64	3	340	128
...	...	...	...	...	...	...
395	12.57	138	108	17	203	128
396	6.14	139	23	3	37	120
397	7.41	162	26	12	368	159
398	5.94	100	79	7	284	95
399	9.71	134	37	0	27	120

400 rows × 6 columns

In [37]:
```
y
```

Out[37]:
```
0      3
1      1
2      2
3      2
4      3
      ..
395    1
396    2
397    2
398    3
399    1
Name: ShelveLoc, Length: 400, dtype: int64
```

In [38]:
```
df['ShelveLoc'].unique()
```

Out[38]:
```
array([3, 1, 2], dtype=int64)
```

In [39]:
```
df.ShelveLoc.value_counts()
```

Out[39]:
```
2    219
3     96
1     85
Name: ShelveLoc, dtype: int64
```

In [40]:
```
colnames = list(df.columns)
colnames
```

Out[40]:
```
['Sales',
 'CompPrice',
 'Income',
 'Advertising',
 'Population',
 'Price',
 'ShelveLoc',
 'Age',
 'Education',
 'Urban_Yes',
 'US_Yes']
```

In [41]:
```
# Splitting data into training and testing data set
x_train, x_test,y_train,y_test = train_test_split(x,y, test_size=0.2,random_state
```

Building Decision Tree Classifier using Entropy Criteria

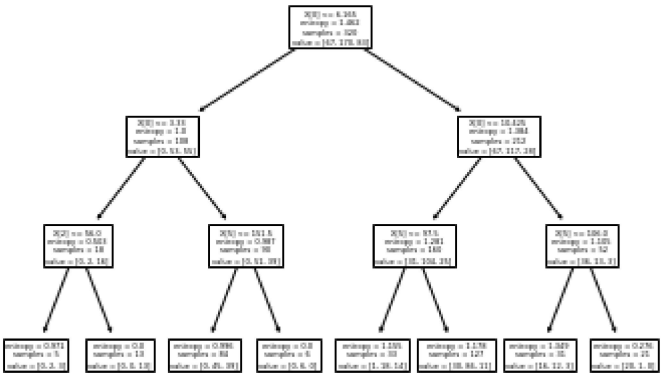
In [43]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [44]:
```
model = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
model.fit(x_train,y_train)
```

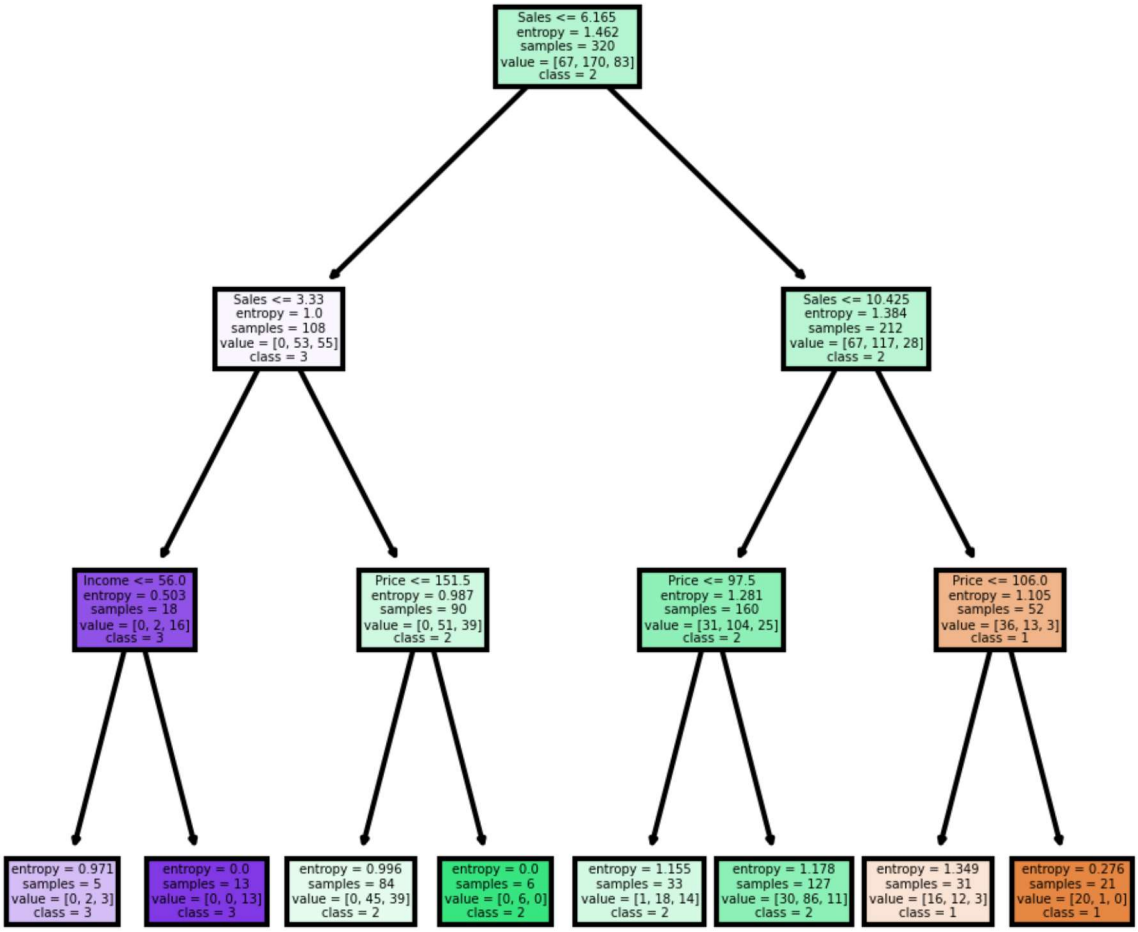
Out[44]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [45]:
```
from sklearn import tree
```

In [47]:
```
#PLot the decision tree
tree.plot_tree(model);
```



In [48]:
```
fn=['Sales','CompPrice','Income','Advertising','Population','Price']
cn=['1', '2', '3']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
tree.plot_tree(model,
               feature_names = fn,
               class_names=cn,
               filled = True);
```



In [49]:
```
#Predicting on test data
preds = model.predict(x_test) # predicting on test data set
pd.Series(preds).value_counts() # getting the count of each category
```

Out[49]:
```
2    63
1    13
3     4
dtype: int64
```

In [50]:
```
preds
```

Out[50]:
```
array([2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 3, 2,
       2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 1, 1, 2, 2, 3, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2], dtype=int64)
```

In [51]:
```
pd.crosstab(y_test,preds) # getting the 2 way table to understand the correct and
```

Out[51]:

col_0	1	2	3
ShelveLoc			
1	8	10	0
2	5	41	3
3	0	12	1

In [52]:
```
# Accuracy
np.mean(preds==y_test)
```

Out[52]:
```
0.625
```

Building Decision Tree Classifier (CART) using Gini Criteria

In [53]:
```
from sklearn.tree import DecisionTreeClassifier
model_gini = DecisionTreeClassifier(criterion='gini', max_depth=3)
```

In [54]:
```
model_gini.fit(x_train, y_train)
```

Out[54]:
```
DecisionTreeClassifier(max_depth=3)
```

In [55]:
```
#Prediction and computing the accuracy
pred=model.predict(x_test)
np.mean(preds==y_test)
```

Out[55]:
```
0.625
```

Decision Tree Regression Example

In [56]:
```
# Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
```

In [57]:
```
array = df.values
X = array[:,0:3]
y = array[:,3]
```

In [58]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [59]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[59]:
```
DecisionTreeRegressor()
```

In [60]:
```
#Find the accuracy
model.score(X_test,y_test)
```

Out[60]:
```
-1.3189125584522756
```

In [ ]: