# Assignment-14-Decision_Tree - Company_data

In [79]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import LabelEncoder # for encoding
from sklearn.model_selection import train_test_split # for train test splitting
from sklearn.tree import DecisionTreeClassifier # for decision tree object
from sklearn.metrics import classification_report, confusion_matrix # for checking testing results
from sklearn.tree import plot_tree # for visualizing tree
```

In [80]:
```
# Importing data
df = pd.read_csv('Company_Data.csv')
df.head()
```

Out[80]:

| | Sales | CompPrice | Income | Advertising | Population | Price | ShelveLoc | Age | Education | Urban | US |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 9.50 | 138 | 73 | 11 | 276 | 120 | Bad | 42 | 17 | Yes | Yes |
| **1** | 11.22 | 111 | 48 | 16 | 260 | 83 | Good | 65 | 10 | Yes | Yes |
| **2** | 10.06 | 113 | 35 | 10 | 269 | 80 | Medium | 59 | 12 | Yes | Yes |
| **3** | 7.40 | 117 | 100 | 4 | 466 | 97 | Medium | 55 | 14 | Yes | Yes |
| **4** | 4.15 | 141 | 64 | 3 | 340 | 128 | Bad | 38 | 13 | Yes | No |

In [81]:
```
# Getting information of dataset
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 [82]:
```
df.shape
```

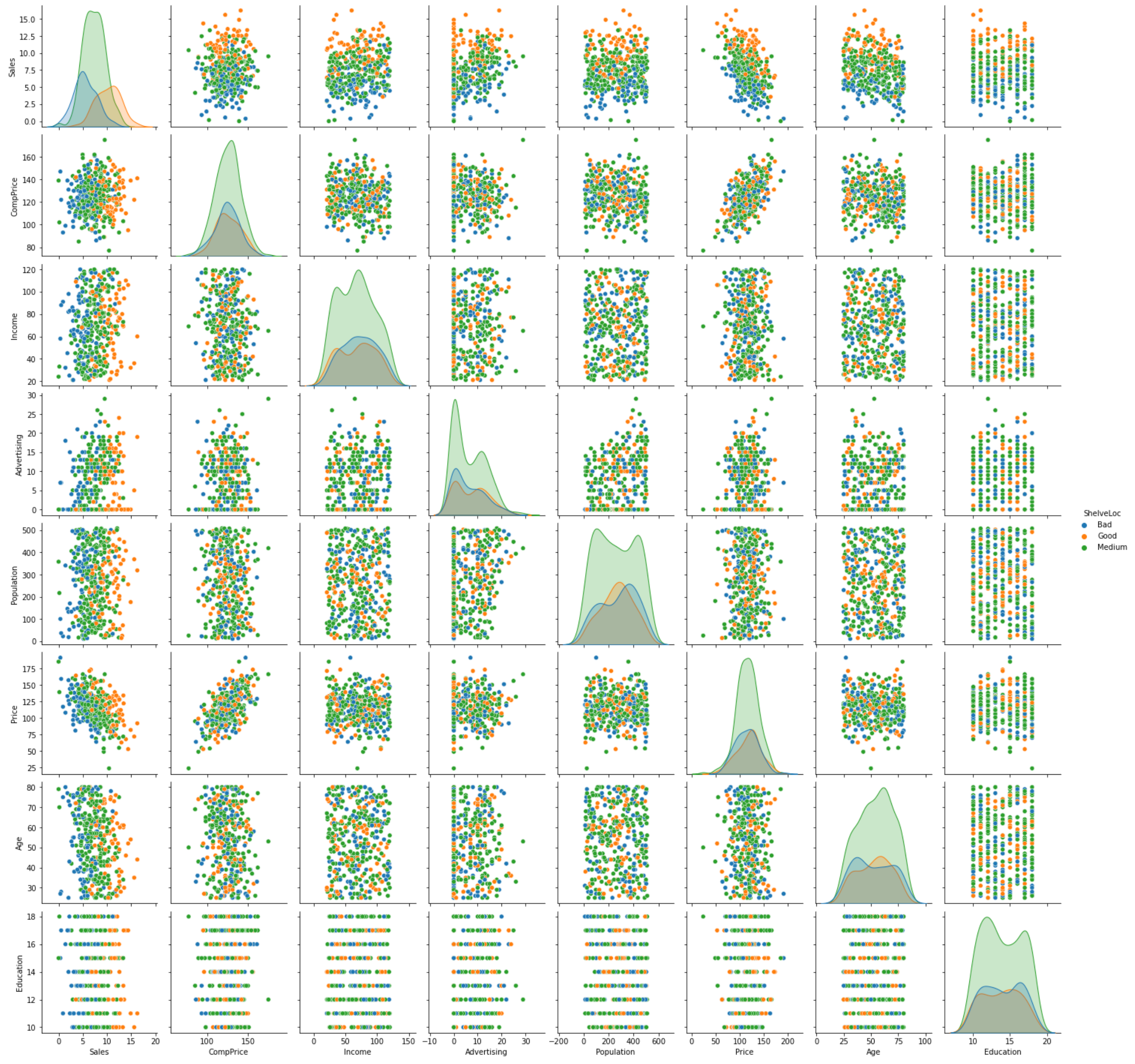
Out[82]:
```
(400, 11)
```

In [83]:
```
df.isnull().any()
```

Out[83]:
```
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 [84]:
```python
# lets plot pair plot to visualize the attributes all at once
sns.pairplot(data=df, hue = 'ShelveLoc')
```

Out[84]:
```
<seaborn.axisgrid.PairGrid at 0x229fd845d00>
```



In [85]:
```python
# Creating dummy variables dropping first dummy variable
df=pd.get_dummies(df,columns=['Urban','US'], drop_first=True)
```

In [86]:
```
df
```

Out[86]:

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

400 rows × 11 columns

In [87]:
```
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 [88]:
```
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
```

In [89]:
```
df['ShelveLoc']=df['ShelveLoc'].map({'Good':1,'Medium':2,'Bad':3})
```

In [90]:
```
df.head()
```

Out[90]:

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

In [91]:
```
x=df.iloc[:,0:6]
y=df['ShelveLoc']
```

In [92]:
```
x
```

Out[92]:

|  | 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 [93]:
```
y
```

Out[93]:
```
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 [94]:
```
df['ShelveLoc'].unique()
```

Out[94]:
```
array([3, 1, 2], dtype=int64)
```

In [95]:
```
df.ShelveLoc.value_counts()
```

Out[95]:
```
2    219
3     96
1     85
Name: ShelveLoc, dtype: int64
```

In [96]:
```
colnames = list(df.columns)
colnames
```

Out[96]:
```
['Sales',
 'CompPrice',
 'Income',
 'Advertising',
 'Population',
 'Price',
 'ShelveLoc',
 'Age',
 'Education',
 'Urban_Yes',
 'US_Yes']
```

In [97]:
```
# Splitting data into training and testinh dataset
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,random_state=40)
```

## Building Decision Tree Classifier using Entropy Criteria.

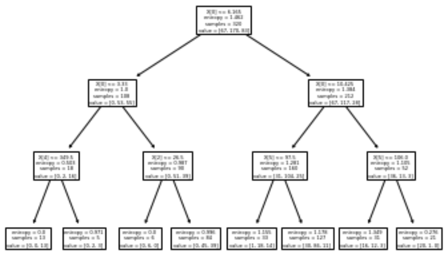
In [98]:
```
model = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
model.fit(x_train,y_train)
```

Out[98]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

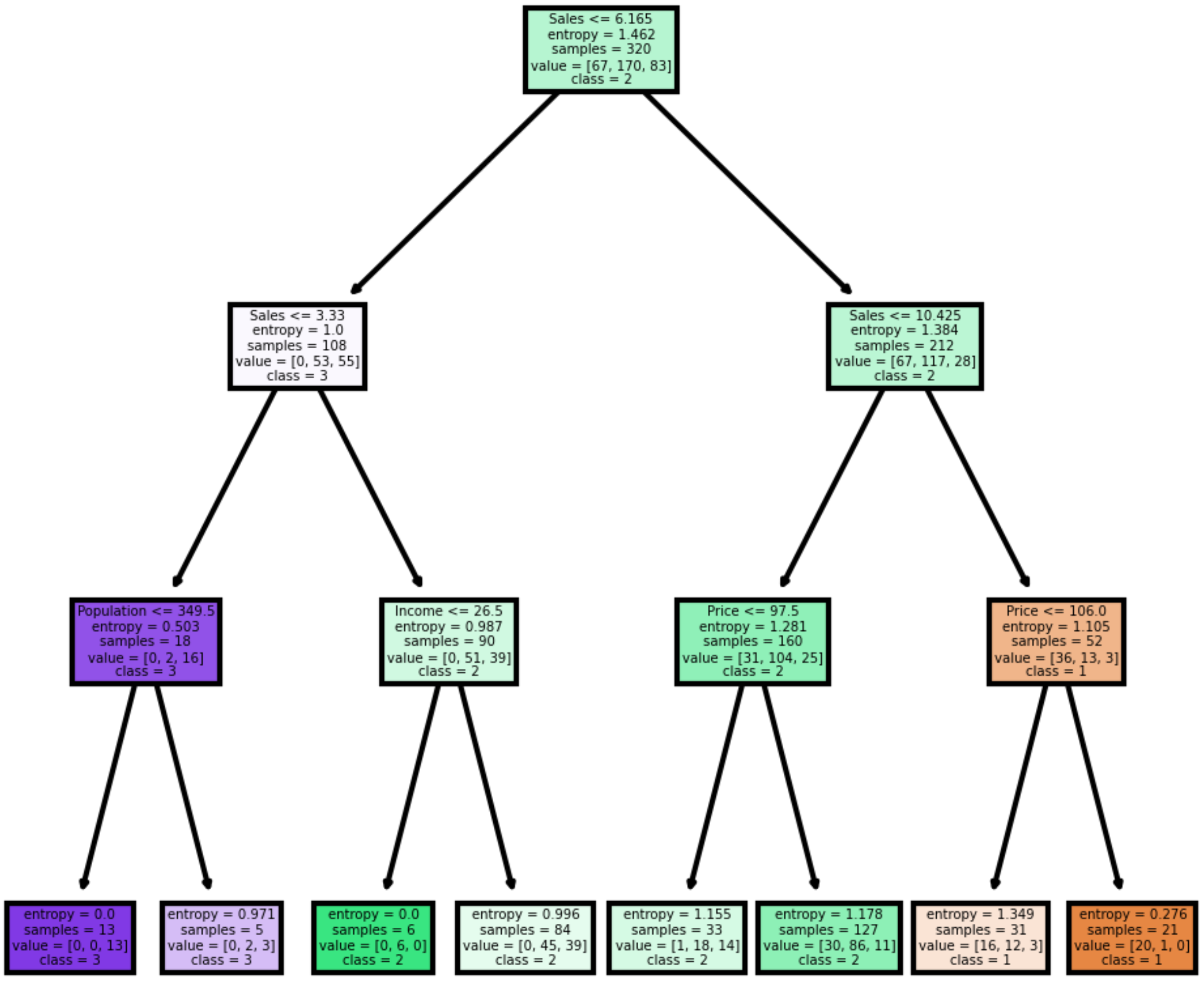
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [99]:
```
from sklearn import tree
```

In [100]:
```
# Plot the decision tree
tree.plot_tree(model);
```



In [101]:
```
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 [102]:
```
# Predicting on test data
preds = model.predict(x_test) # prediciting on test dataset
pd.Series(preds).value_counts() # getting the count of each catergory
```

Out[102]:
```
2    63
1    13
3     4
dtype: int64
```

In [103]:
```
preds
```

Out[103]:
```
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 [104]:
```
pd.crosstab(y_test,preds)  # getting the 2 way table to understand the correct and wrong predicitions
```

Out[104]:

| col_0 | 1 | 2 | 3 |
|---|---|---|---|
| **ShelveLoc** | | | |
| **1** | 8 | 10 | 0 |
| **2** | 5 | 41 | 3 |
| **3** | 0 | 12 | 1 |

In [105]:
```
# Accuracy
np.mean(preds==y_test)
```

Out[105]:
```
0.625
```

## Building Decision Tree Classifier (CART) using Gini Criteria

In [106]:
```
from sklearn.tree import DecisionTreeClassifier
model_gini = DecisionTreeClassifier(criterion='gini', max_depth=3)
model_gini.fit(x_train, y_train)
```

Out[106]:
```
DecisionTreeClassifier(max_depth=3)
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [107]:
```
# Predicition and computing the accuracy
pred = model.predict(x_test)
np.mean(preds==y_test)
```

Out[107]:
```
0.625
```

## Decision Tree Regression Example

In [108]:
```
# Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
```

In [109]:
```
array = df.values
X = array[:,0:3]
y = array[:,3]
```

In [110]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
```

In [111]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

Out[111]:
```
DecisionTreeRegressor()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [112]:
```
# Find the accuracy
model.score(X_test,y_test)
```

Out[112]:
```
-1.2646168892692322
```

In [ ]:
```

```

In [ ]:
```

```