In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import GradientBoostingClassifier

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

In [4]:
```
data = pd.read_csv('Company_Data.csv')
data
```

Out[4]:

	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 [5]:
```
data.loc[data['Sales'] >=7,'sales'] = 'High'
data.loc[data['Sales'] <7 ,'sales'] = 'Low'
```

In [6]:
```
data.drop('Sales',axis = 1,inplace = True)
```

In [7]:
```
data.shape
```

Out[7]:
```
(400, 11)
```

In [8]:
```
data.isna().sum()
```

Out[8]:
```
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
sales          0
dtype: int64
```

In [9]:
```
data.dtypes
```

Out[9]:
```
CompPrice       int64
Income          int64
Advertising     int64
Population      int64
Price           int64
ShelveLoc      object
Age             int64
Education       int64
Urban          object
US             object
sales          object
dtype: object
```

In [10]:
```
le = LabelEncoder()
data['ShelveLoc'] = le.fit_transform(data['ShelveLoc'])
data['ShelveLoc'].unique()
```

Out[10]:
```
array([0, 1, 2])
```

In [11]:
```
data['Urban'] = le.fit_transform(data['Urban'])
data['Urban'].unique()
```

Out[11]:
```
array([1, 0])
```

In [12]:
```
data['US'] = le.fit_transform(data['US'])
data['US'].unique()
```

Out[12]:
```
array([1, 0])
```

In [13]:
```
data.head()
```

Out[13]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	sa
0	138	73	11	276	120	0	42	17	1	1	H
1	111	48	16	260	83	1	65	10	1	1	H
2	113	35	10	269	80	2	59	12	1	1	H
3	117	100	4	466	97	2	55	14	1	1	H
4	141	64	3	340	128	0	38	13	1	0	L

In [14]:
```
x = data.drop('sales',axis=1)
y = data[['sales']]
```

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=
```

In [16]:
```
x_train.shape,y_train.shape
```

Out[16]:
```
((320, 10), (320, 1))
```

In [17]:
```
x_test.shape,y_test.shape
```

Out[17]:
```
((80, 10), (80, 1))
```

In [18]:
```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```

In [19]:
```
y_pred_train = dt_model.predict(x_train)
```

In [20]:
```
print('Accuracy score :',accuracy_score(y_train,y_pred_train))
```

```
Accuracy score : 1.0
```

In [21]:
```
print('Confusion Metric :\n',confusion_matrix(y_train,y_pred_train))
```

```
Confusion Metric :
 [[174   0]
 [  0 146]]
```

In [22]:
```
y_pred_test = dt_model.predict(x_test)
```

In [23]:
```
print('Accuracy score :', accuracy_score(y_test,y_pred_test))
```

```
Accuracy score : 0.7125
```

In [24]:
```
print('Confusion matrix :\n ',confusion_matrix(y_test,y_pred_test))
```

```
Confusion matrix :
  [[31 11]
 [12 26]]
```

In [25]:
```
gbc = GradientBoostingClassifier()
gbc.fit(x_train,y_train)
```

Out[25]:
```
GradientBoostingClassifier()
```

In [27]:
```
y_pred = gbc.predict(x_test)
```

In [28]:
```
print('Accuracy score :', accuracy_score(y_test,y_pred))
```

```
Accuracy score : 0.8625
```

In [30]:
```
print('Confusion matrix :\n ',confusion_matrix(y_test,y_pred))
```

```
Confusion matrix :
  [[35  7]
 [ 4 34]]
```

In [31]:
```
y_pred_tr = gbc.predict(x_train)
```

In [32]:
```
print('Accuracy score :', accuracy_score(y_train,y_pred_tr))
```

```
Accuracy score : 0.996875
```

In [33]:
```
print('Confusion matrix :\n ',confusion_matrix(y_train,y_pred_tr))
```

```
Confusion matrix :
  [[174   0]
 [  1 145]]
```

In [ ]:
```

```