Import neccessery libraries

In [90]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import metrics
from sklearn import externals
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

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

problem

A cloth manufacturing company is interested to know about the segment or attributes causes high sale

Import data

In [2]:

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

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	
395	12.57	138	108	17	203	128	Good	33	14	Y
396	6.14	139	23	3	37	120	Medium	55	11	N
397	7.41	162	26	12	368	159	Medium	40	18	Y
398	5.94	100	79	7	284	95	Bad	50	12	Y
399	9.71	134	37	0	27	120	Good	49	16	Y

400 rows × 11 columns

Data understanding

In [3]:
```
company_data.shape
```

Out[3]:
```
(400, 11)
```

In [4]:
```
company_data.isna().sum()
```

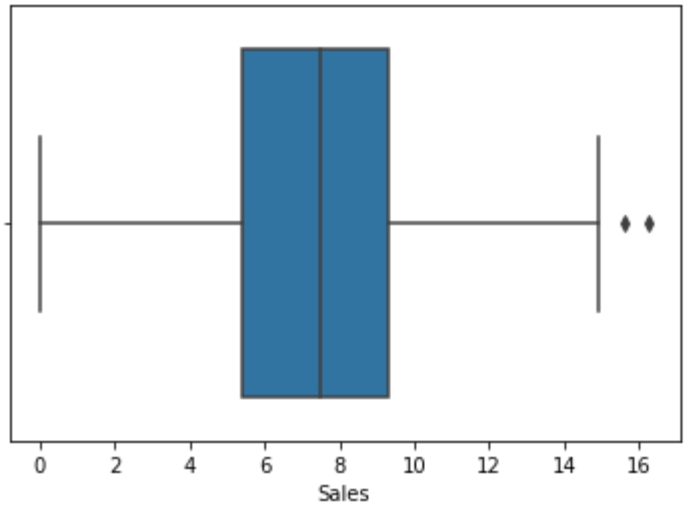
Out[4]:
```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

In [5]:
```
company_data.dtypes
```

Out[5]:
```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc       object
Age              int64
Education        int64
Urban           object
US              object
dtype: object
```

outlier check

In [9]:
```
ax = sns.boxplot(company_data['Sales'])
```



Data has 2 outlier instances

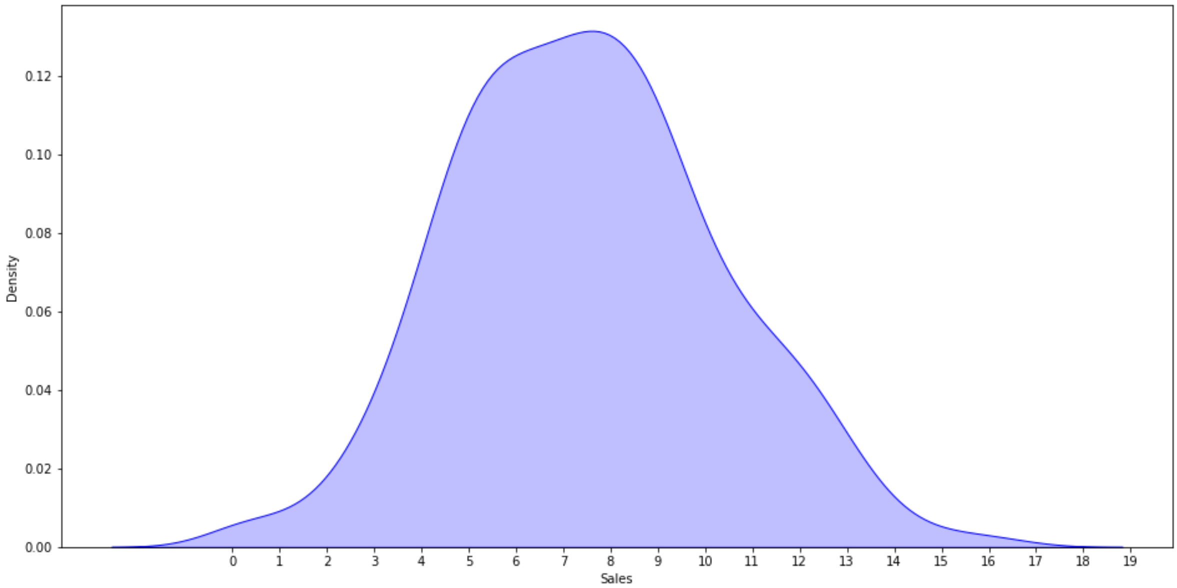
In [12]:

```
plt.rcParams["figure.figsize"] = 9,5
```

In [14]:

```
plt.figure(figsize=(16,8))
print("Skew: {}".format(company_data['Sales'].skew()))
print("Kurtosis: {}".format(company_data['Sales'].kurtosis()))
ax = sns.kdeplot(company_data['Sales'],shade=True,color='b')
plt.xticks([i for i in range(0,20,1)])
plt.show()
```

```
Skew: 0.18556036318721578
Kurtosis: -0.08087736743346197
```



The data is Skwed on the right

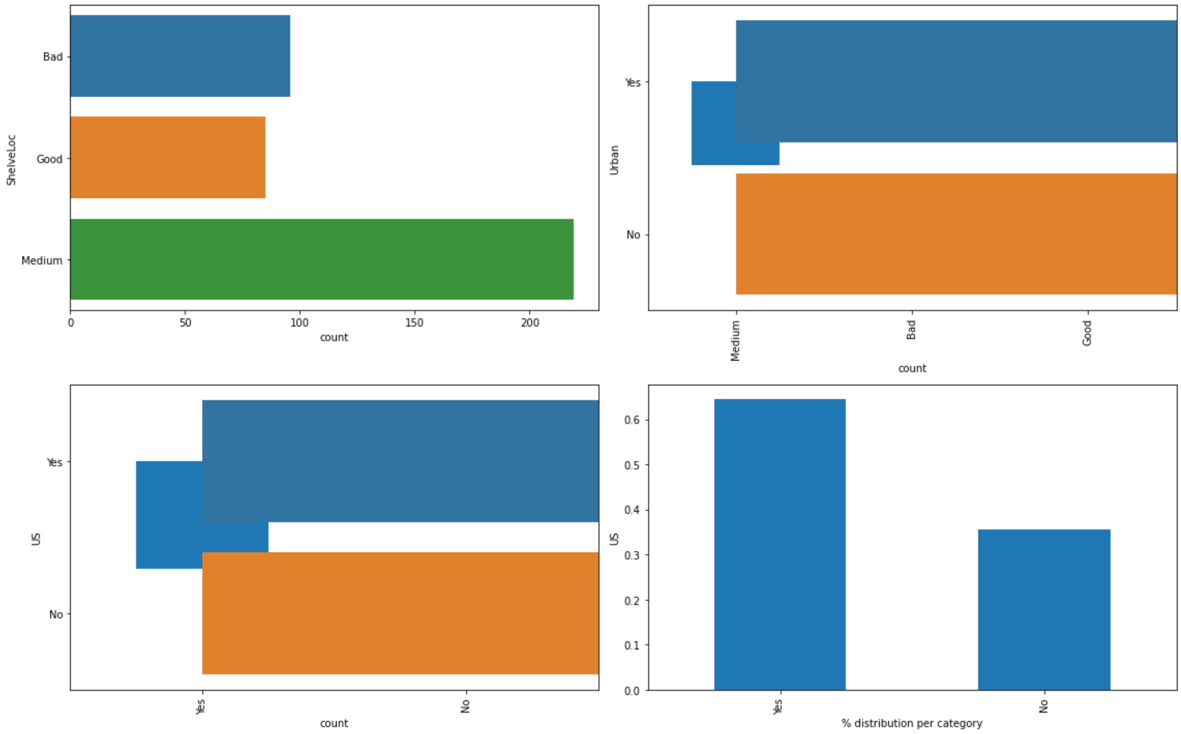
Data has negative Kurtosis

In [16]:

```
obj_colum = company_data.select_dtypes(include='object').columns.tolist()
```

In [19]:

```
plt.figure(figsize=(16,10))
for i,col in enumerate(obj_colum,1):
    plt.subplot(2,2,i)
    sns.countplot(data=company_data,y=col)
    plt.subplot(2,2,i+1)
    company_data[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

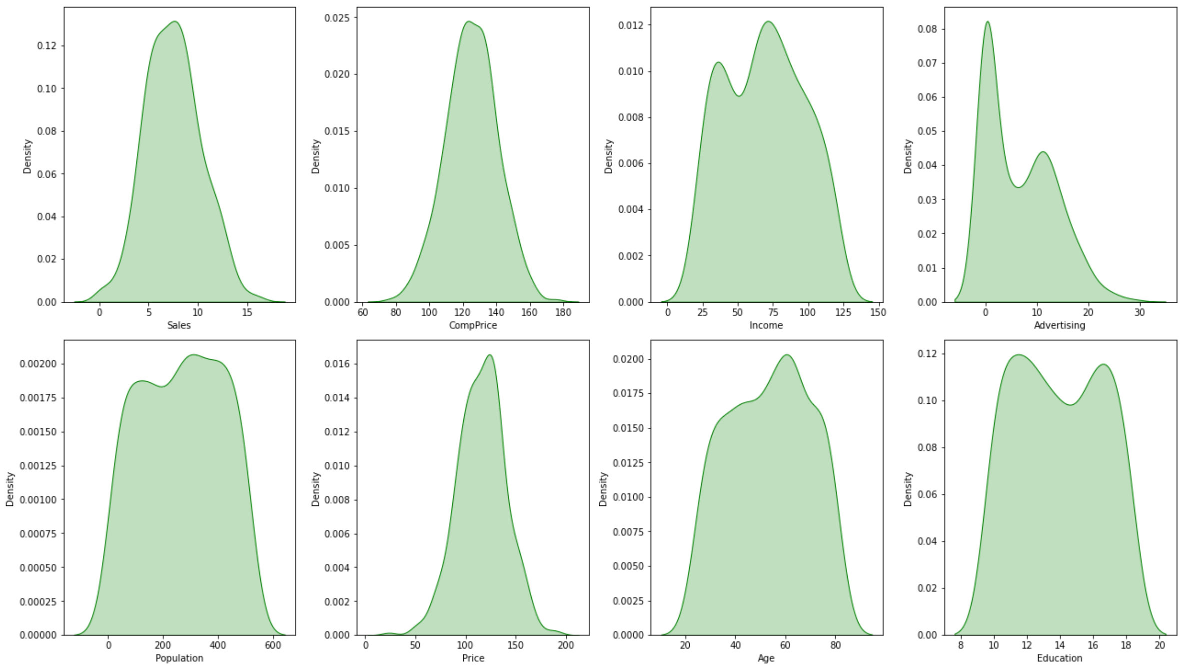


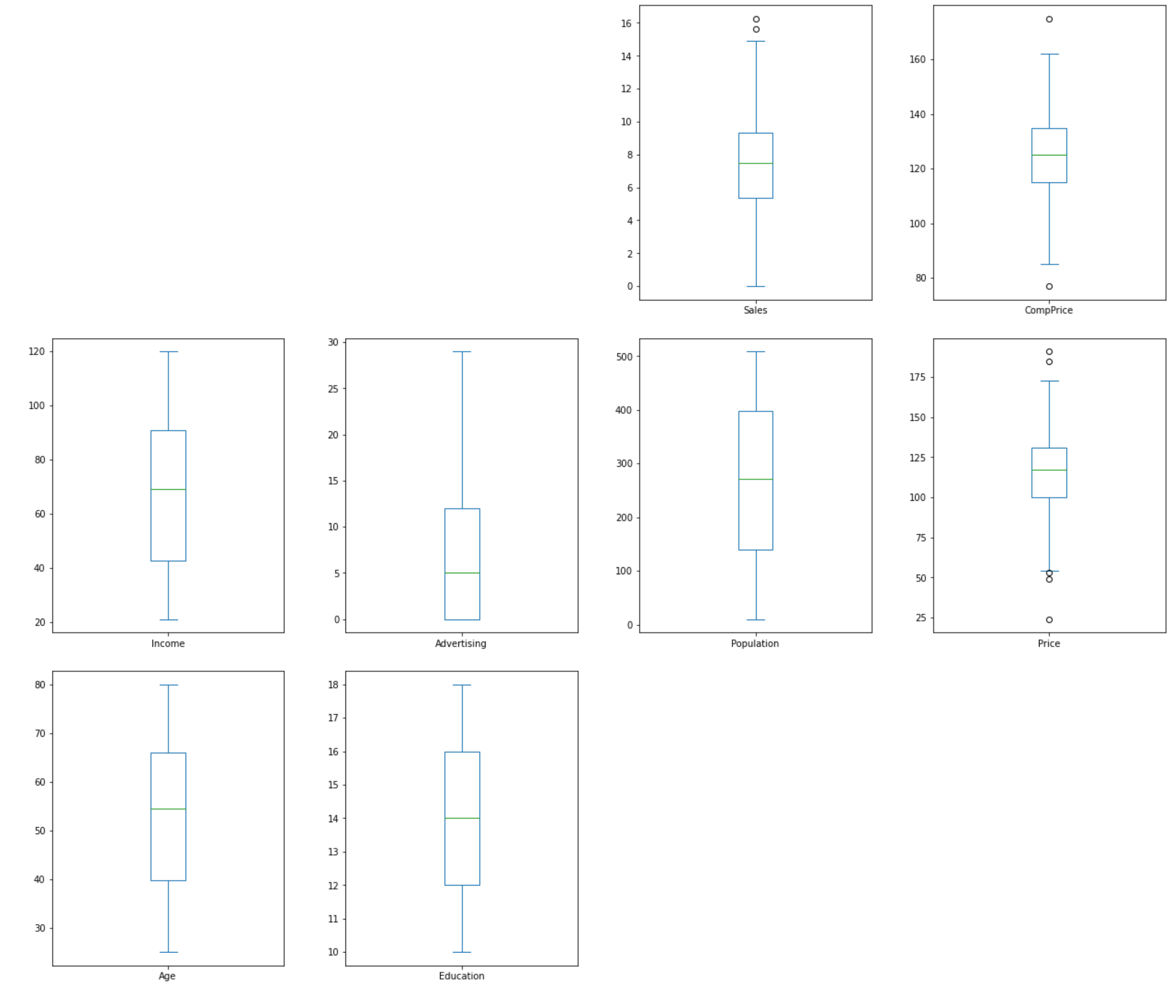
In [21]:

```
number_columns = company_data.select_dtypes(exclude='object').columns.tolis
```

In [24]:

```
plt.figure(figsize=(18,40))
for i,col in enumerate(number_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(company_data[col],color='g',shade=True)
    plt.subplot(8,4,i+10)
    company_data[col].plot.box()
plt.tight_layout()
plt.show()
number_data = company_data[number_columns]
pd.DataFrame(data=[number_data.skew(),number_data.kurtosis()],index=['skewn
```





Out[24]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Educat
skewness	0.185560	-0.042755	0.049444	0.639586	-0.051227	-0.125286	-0.077182	0.044(
kurtosis	-0.080877	0.041666	-1.085289	-0.545118	-1.202318	0.451885	-1.134392	-1.298:

In [25]:

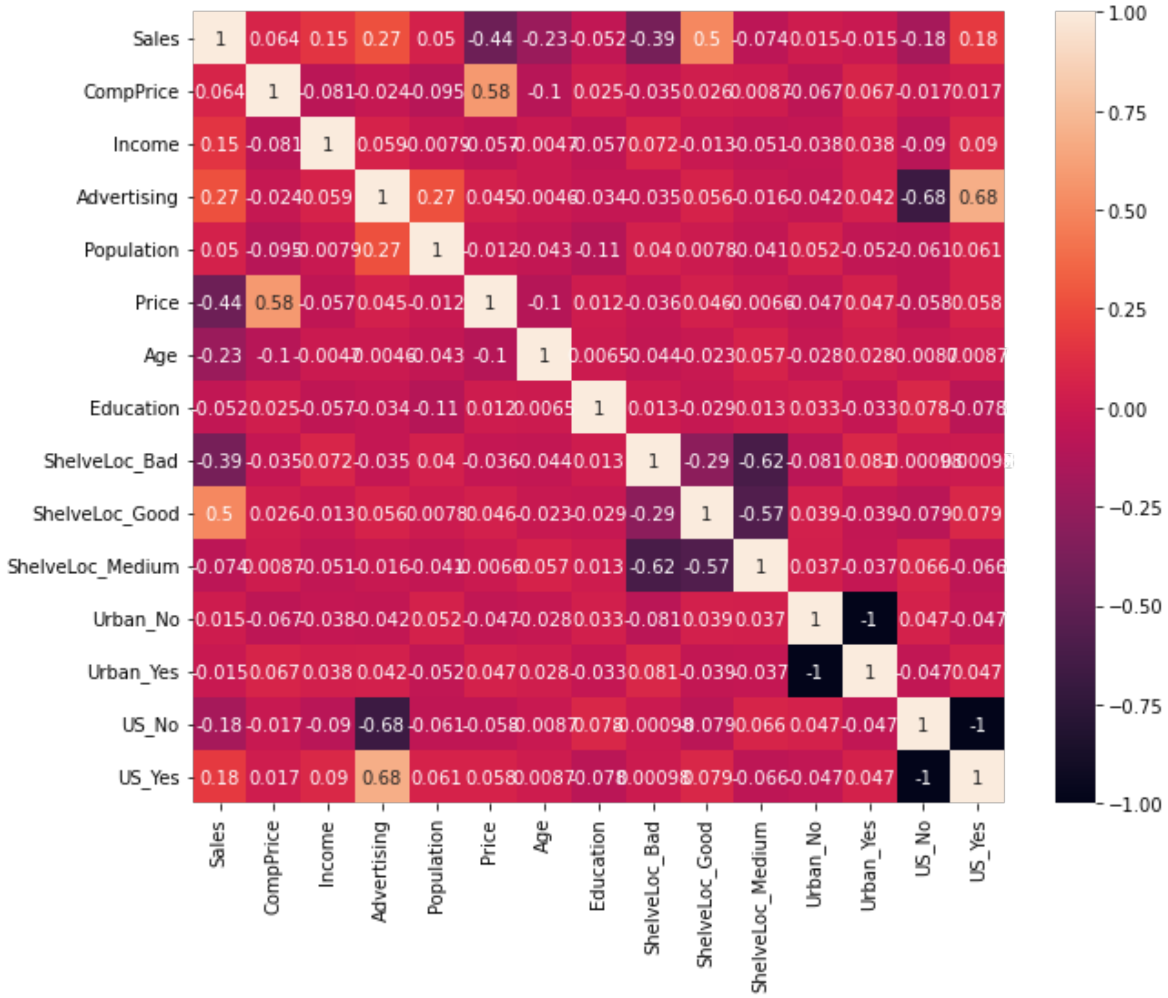
```
corr = company_data.corr()
```

In [33]:
```
com_data2 = pd.get_dummies(company_data, columns = ['ShelveLoc','Urban','US
```

In [34]:
```
corr = com_data2.corr()
```

In [35]:
```
plt.figure(figsize=(10,8))
sns.heatmap(corr,annot=True)
```

Out[35]:
```
<AxesSubplot:>
```



Decision Tree

Since the target variable is continious, we create a class of the value based on the mean

<= 7.49 = "Small" and > 7.49 = "large"

In [37]:
```
com_data2["sales"]="small"
com_data2.loc[com_data2["Sales"]>7.49,"sales"]="large"
com_data2.drop(["Sales"],axis=1,inplace=True)
```

In [51]:
```
X = com_data2.iloc[:,0:14]
y = com_data2.iloc[:,14]
```

In [52]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.2, strat
```
In [53]:
```
y_train.value_counts()
```
Out[53]:
```
small    161
large    159
Name: sales, dtype: int64
```
In [54]:
```
model = DecisionTreeClassifier()
model.fit(x_train,y_train)
```
Out[54]:
```
DecisionTreeClassifier()
```
In [56]:
```
pred_train = model.predict(x_train)
accuracy_score(y_train,pred_train)
```
Out[56]:
```
1.0
```
In [57]:
```
confusion_matrix(y_train,pred_train)
```
Out[57]:
```
array([[159,   0],
       [  0, 161]], dtype=int64)
```
In [58]:
```
pred_test = model.predict(x_test)
accuracy_score(y_test,pred_test)
```
Out[58]:
```
0.7125
```
In [59]:
```
confusion_matrix(y_test,pred_test)
```
Out[59]:
```
array([[27, 13],
       [10, 30]], dtype=int64)
```
In [60]:
```
df_t=pd.DataFrame({'Actual':y_test, 'Predicted':pred_test})
```
In [61]:
```
df_t
```
Out[61]:

	Actual	Predicted
13	large	large
340	large	small
29	large	small
282	large	small
167	small	large
...	...	...
220	large	large
55	small	small

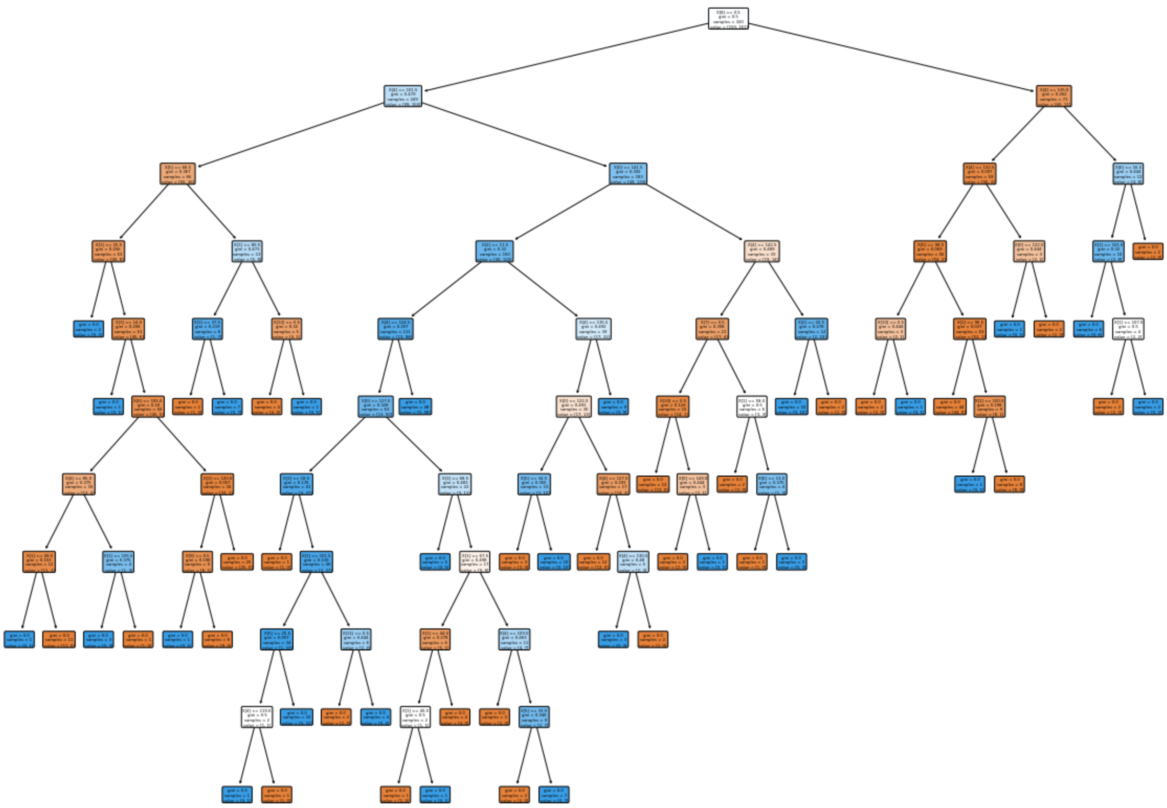
	Actual	Predicted
323	large	large
361	large	large
0	large	small

4 - Conclusion

Since the accuracy of the Training set is 100% we test the accurancy on the test data which is 70% As seen in the confusion matrix of Test data 56 instances are presdected correctly and 24 instances are not

In [85]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(20,15))
plot_tree(decision_tree=model,filled=True,rounded=True)
plt.show()
```



In [86]:

```
model.feature_importances_
```

Out[86]:

```
array([0.12426237, 0.13668003, 0.05749532, 0.03491962, 0.3438158 ,
       0.08232412, 0.02270922, 0.01005992, 0.13828723, 0.01111155,
       0.01166712, 0.01666732, 0.01000039, 0.        ])
```

In [88]:

```
fig = pd.DataFrame({'feature': list(x_train.columns),
                    'importance': model.feature_importances_}).\
                     sort_values('importance', ascending = False)
```

In [89]:
```
fig
```

Out[89]:

	feature	importance
4	Price	0.343816
8	ShelveLoc_Good	0.138287
1	Income	0.136680
0	CompPrice	0.124262
5	Age	0.082324
2	Advertising	0.057495
3	Population	0.034920
6	Education	0.022709
11	Urban_Yes	0.016667
10	Urban_No	0.011667
9	ShelveLoc_Medium	0.011112
7	ShelveLoc_Bad	0.010060
12	US_No	0.010000
13	US_Yes	0.000000

As seen in the above table Price is most important feature

In [ ]:
```

```