In [1]:
```python
#Import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import datasets,tree
from sklearn.tree import export_graphviz
from sklearn import externals
from io import StringIO
import pydotplus
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

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

In [2]:
```python
# import the dataset
company=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Random_Forests\\Comp
```

In [3]:
```python
company
```

Out[3]:

	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 [4]:
```
company.head()
```

Out[4]:

	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 [5]:
```
company1=company.copy()
```

In [6]:
```
company1.describe()
```

Out[6]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Educ
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.0(
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.9(
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.62
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.0(
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.0(
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.0(
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.0(
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.0(

In [7]:
```
company1.isnull().sum()
```

Out[7]:
```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

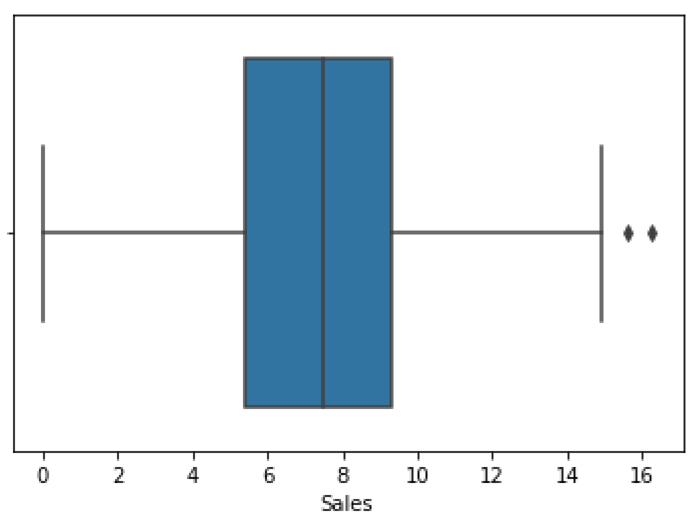
In [8]:
```
company1.dtypes
```

Out[8]:
```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc       object
Age              int64
Education        int64
Urban           object
US              object
dtype: object
```

CHECK FOR OUTLIERS

In [9]:
```
outL=sns.boxplot(company1['Sales'])
```

```
C:\Users\nishi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

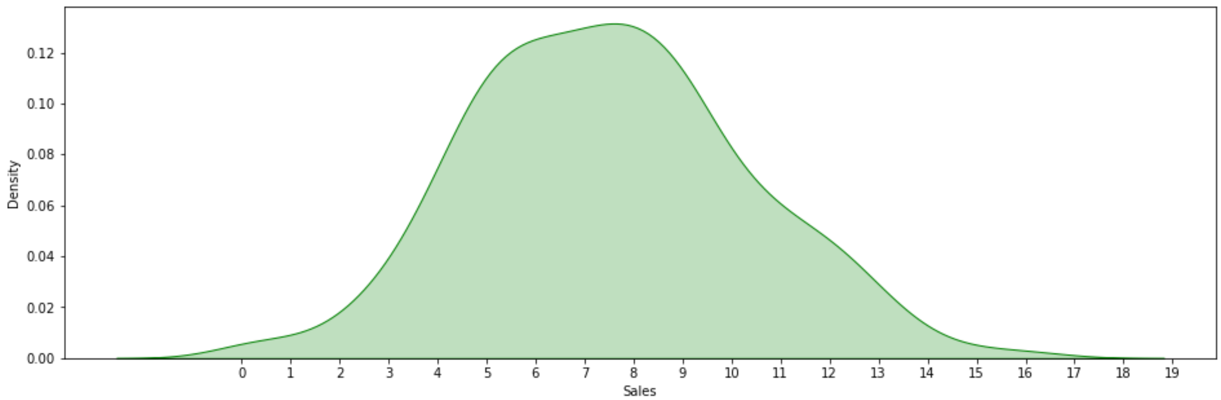


There are two ouliers.

In [10]:
```
plt.rcParams["figure.figsize"] = 9,5
```

In [11]:
```
plt.figure(figsize=(16,5))
print("Skew: {}".format(company1['Sales'].skew()))
print("Kurtosis: {}".format(company1['Sales'].kurtosis()))
ax = sns.kdeplot(company1['Sales'],shade=True,color='g')
plt.xticks([i for i in range(0,20,1)])
plt.show()
```

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

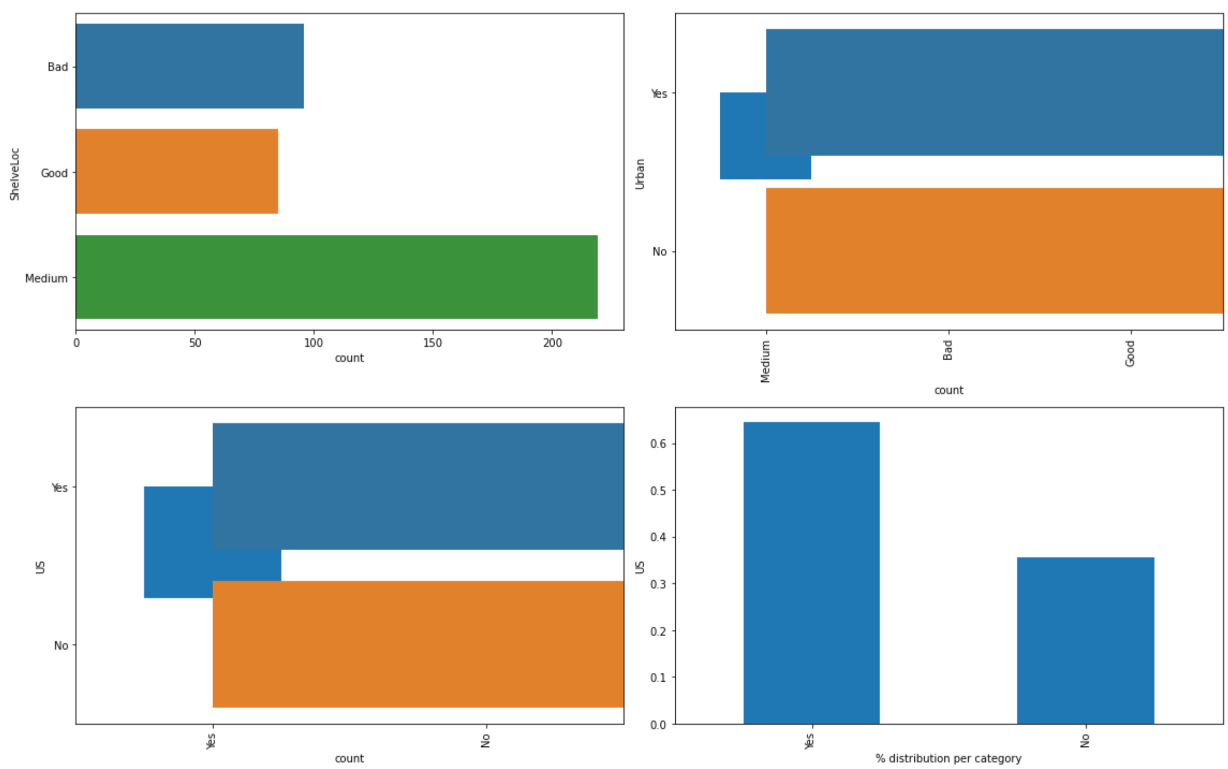


The data is skewed to the right.

In [12]:
```
obj_colum = company1.select_dtypes(include='object').columns.tolist()
```

In [13]:
```
plt.figure(figsize=(16,10))
for i,col in enumerate(obj_colum,1):
    plt.subplot(2,2,i)
    sns.countplot(data=company1,y=col)
    plt.subplot(2,2,i+1)
    company1[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

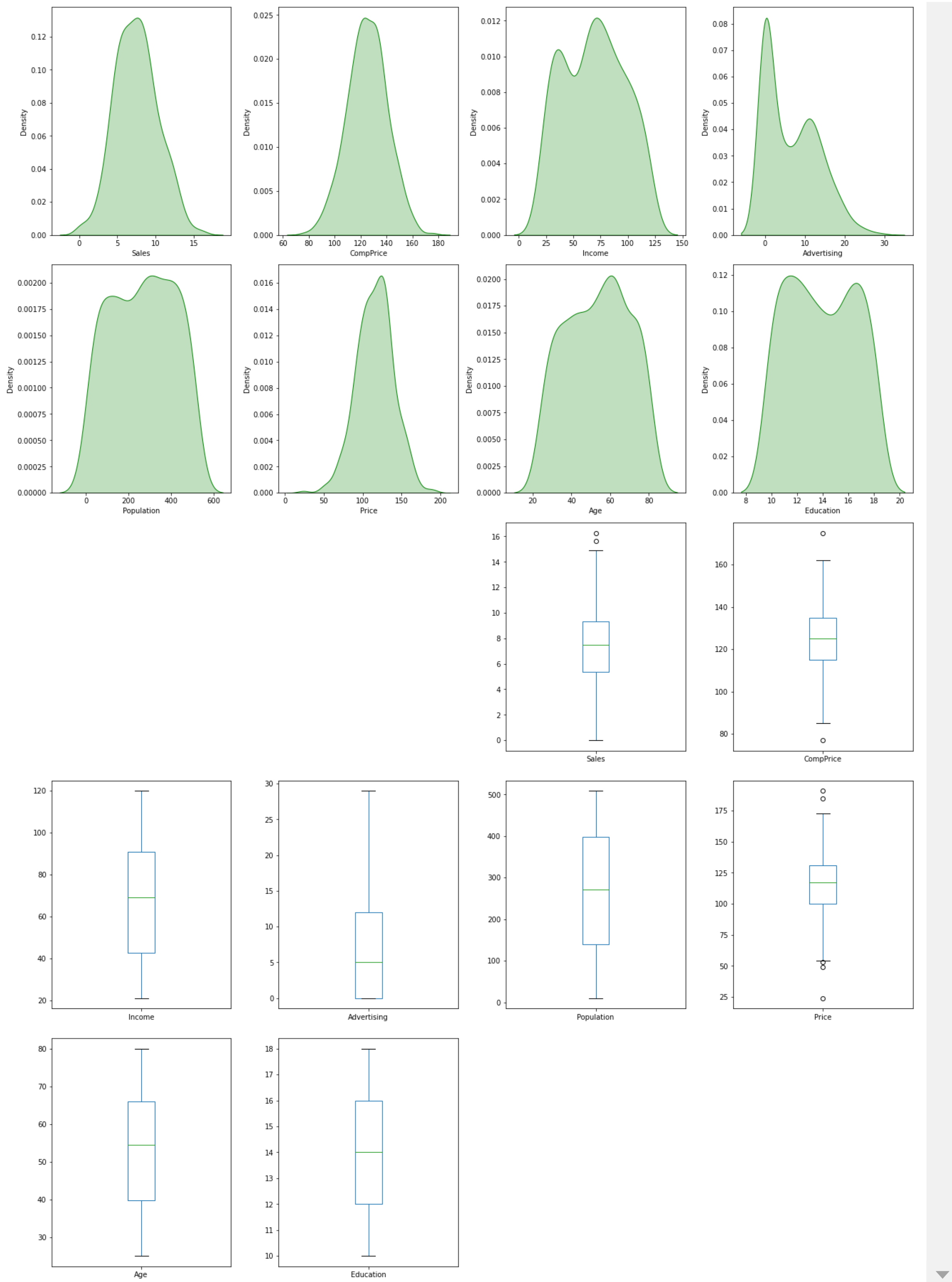
```
<ipython-input-13-971a39a9eccf>:3: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier instan
ce.  In a future version, a new instance will always be created and returned.
Meanwhile, this warning can be suppressed, and the future behavior ensured, by
passing a unique label to each axes instance.
  plt.subplot(2,2,i)
<ipython-input-13-971a39a9eccf>:3: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the earlier instan
ce.  In a future version, a new instance will always be created and returned.
Meanwhile, this warning can be suppressed, and the future behavior ensured, by
passing a unique label to each axes instance.
  plt.subplot(2,2,i)
```



In [14]:
```
num_columns = company1.select_dtypes(exclude='object').columns.tolist()
```

In [15]:

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



Out[15]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Educatic
skewness	0.185560	-0.042755	0.049444	0.639586	-0.051227	-0.125286	-0.077182	0.0440
kurtosis	-0.080877	0.041666	-1.085289	-0.545118	-1.202318	0.451885	-1.134392	-1.2983

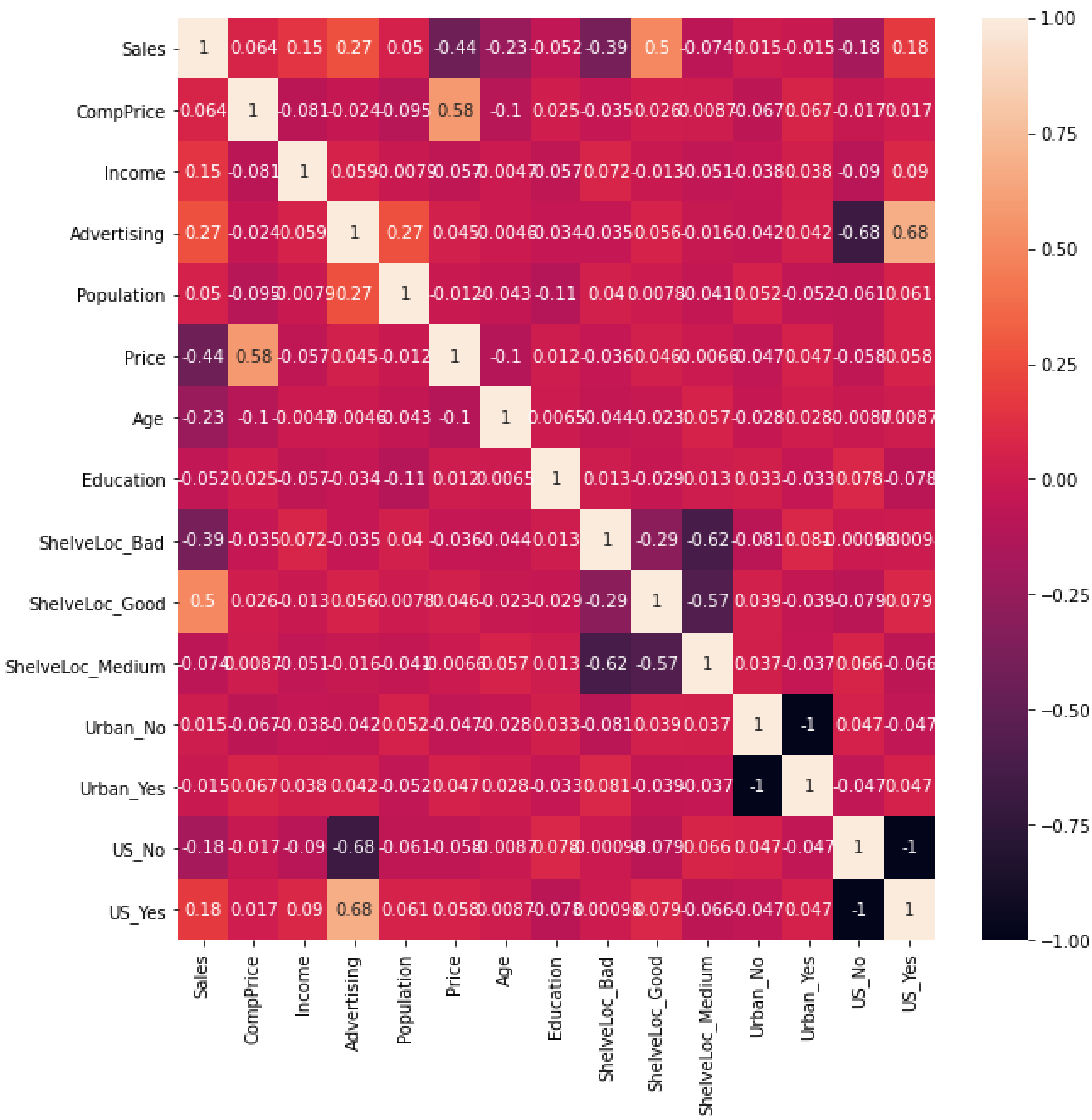
```
In [16]: corr = company1.corr()
```

```
In [17]: company1 = pd.get_dummies(company1, columns = ['ShelveLoc','Urban','US'])
```

```
In [18]: corr = company1.corr()
```

```
In [19]: plt.figure(figsize=(10,10))
         sns.heatmap(corr,annot=True)
```

```
Out[19]: <AxesSubplot:>
```



RANDOM FOREST MODEL

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 [20]:
```
company1["sales"]="small"
company1.loc[company1["Sales"]>7.49,"sales"]="large"
company1.drop(["Sales"],axis=1,inplace=True)
```

In [21]:
```
X = company1.iloc[:,0:14]
y = company1.iloc[:,14]
```

In [22]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

In [23]:
```
y_train.value_counts()
```

Out[23]:
```
small    163
large    157
Name: sales, dtype: int64
```

In [24]:
```
model =RF(n_jobs=4,n_estimators = 150, oob_score =True,criterion ='entropy')
model.fit(x_train,y_train)
model.oob_score_
```

Out[24]:
```
0.809375
```

In [25]:
```
pred_train = model.predict(x_train)
```

In [26]:
```
accuracy_score(y_train,pred_train)
```

Out[26]:
```
1.0
```

In [27]:
```
confusion_matrix(y_train,pred_train)
```

Out[27]:
```
array([[157,   0],
       [  0, 163]], dtype=int64)
```

In [28]:
```
pred_test = model.predict(x_test)
```

In [29]:
```
accuracy_score(y_test,pred_test)
```

Out[29]:
```
0.75
```

In [30]:
```
confusion_matrix(y_test,pred_test)
```

Out[30]:
```
array([[31, 11],
       [ 9, 29]], dtype=int64)
```

In [31]:
```python
df_t=pd.DataFrame({'Actual':y_test, 'Predicted':pred_test})
```

In [32]:
```python
df_t
```

Out[32]:

	Actual	Predicted
86	large	small
345	small	small
394	small	small
251	small	small
39	small	small
...	...	...
118	large	small
257	large	small
15	large	small
6	small	small
52	large	small

80 rows × 2 columns

In [33]:
```python
cols = list(company1.columns)
```

In [34]:
```python
predictors = cols[0:14]
target = cols[14]
```

In [35]:
```python
tree1 = model.estimators_[20]
```

In [36]:
```python
dot_data = StringIO()
```

In [37]:
```python
export_graphviz(tree1, out_file = dot_data, feature_names =predictors, class_name
```

In [38]:
```python
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

In [39]:
```python
graph.write_png('company_full.png')
```

Out[39]:
```
True
```

CONCLUSION

Since the accuracy of the Training set is 100% we test the accurancy on the test data which is 75%
As seen in the confusion matrix of Test data 60 instances are presdicted correctly and 20 instances are not.

In [40]:
```
rf_small = RF(n_estimators=10, max_depth = 3)
```

In [41]:
```
rf_small.fit(x_train,y_train)
```

Out[41]:
```
RandomForestClassifier(max_depth=3, n_estimators=10)
```

In [42]:
```
tree_small = rf_small.estimators_[5]
```

In [43]:
```
export_graphviz(tree_small, out_file = dot_data, feature_names = predictors, rour
```

In [44]:
```
graph_small = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

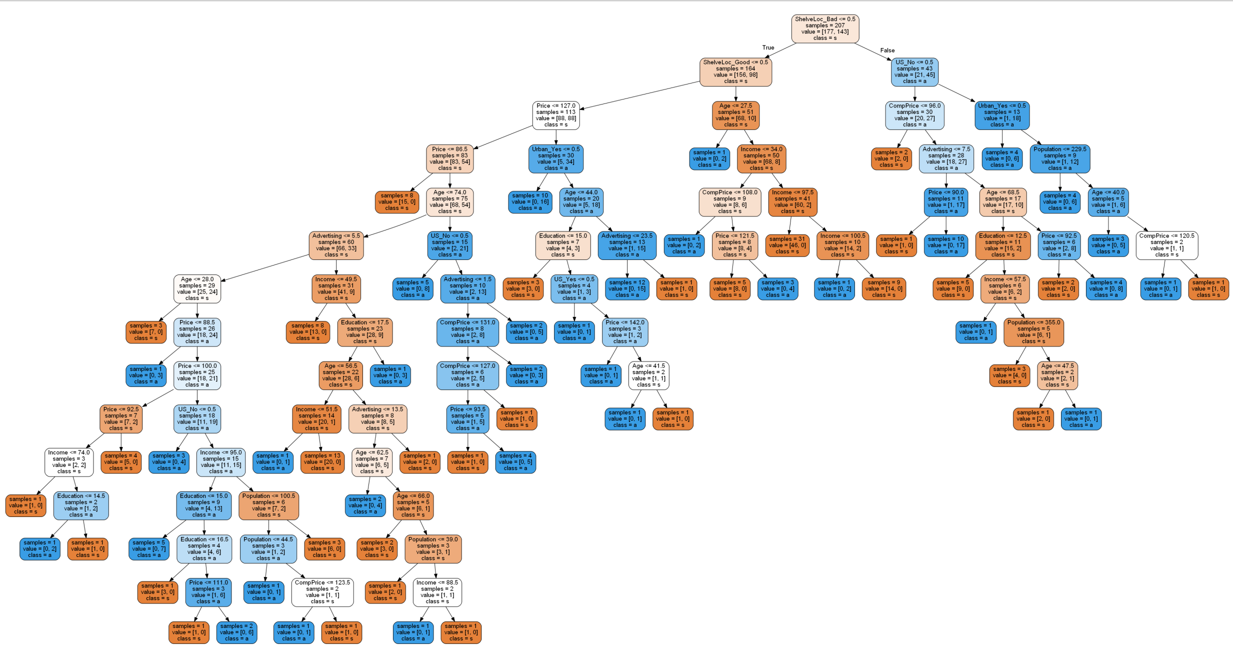
In [45]:
```
graph.write_png('company_small.png')
```

Out[45]:
```
True
```

In [46]:
```
from PIL import Image
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
image=Image.open('company_small.png')
```

In [47]:
```
image
```

Out[47]:



In [48]:
```python
model.feature_importances_
```

Out[48]:
```
array([0.10567617, 0.0936802 , 0.10257796, 0.08986176, 0.253447  ,
       0.11758797, 0.05703248, 0.03610121, 0.07488381, 0.02320449,
       0.01213742, 0.01201735, 0.01092122, 0.01087098])
```

In [49]:
```python
fi = pd.DataFrame({'feature': list(x_train.columns),
                   'importance': model.feature_importances_}).\
                    sort_values('importance', ascending = False)
```

In [50]:
```python
fi
```

Out[50]:

	feature	importance
4	Price	0.253447
5	Age	0.117588
0	CompPrice	0.105676
2	Advertising	0.102578
1	Income	0.093680
3	Population	0.089862
8	ShelveLoc_Good	0.074884
6	Education	0.057032
7	ShelveLoc_Bad	0.036101
9	ShelveLoc_Medium	0.023204
10	Urban_No	0.012137
11	Urban_Yes	0.012017
12	US_No	0.010921
13	US_Yes	0.010871

PRICE is the most important feature.