Import Necessary Details

In [2]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from io import StringIO
import pydotplus
from sklearn.tree import export_graphviz
import matplotlib.image as mpimg
```

Data Collection

In [3]:
```
df = pd.read_csv('Company_Data (1).csv')
```

In [4]:
```
company = df.copy()
```

In [5]:
```
company.head()
```

Out[5]:

	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 [6]:
```
company.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Sales	400.0	7.496325	2.824115	0.0	5.39	7.49	9.32	16.27
CompPrice	400.0	124.975000	15.334512	77.0	115.00	125.00	135.00	175.00
Income	400.0	68.657500	27.986037	21.0	42.75	69.00	91.00	120.00
Advertising	400.0	6.635000	6.650364	0.0	0.00	5.00	12.00	29.00
Population	400.0	264.840000	147.376436	10.0	139.00	272.00	398.50	509.00
Price	400.0	115.795000	23.676664	24.0	100.00	117.00	131.00	191.00
Age	400.0	53.322500	16.200297	25.0	39.75	54.50	66.00	80.00
Education	400.0	13.900000	2.620528	10.0	12.00	14.00	16.00	18.00

In [7]:
```
company.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
```

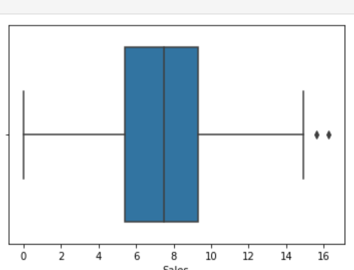
In [8]:
```
company.dtypes
```

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
```

In [ ]:

In [10]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [11]:
```
ax = sns.boxplot(company['Sales'])
```



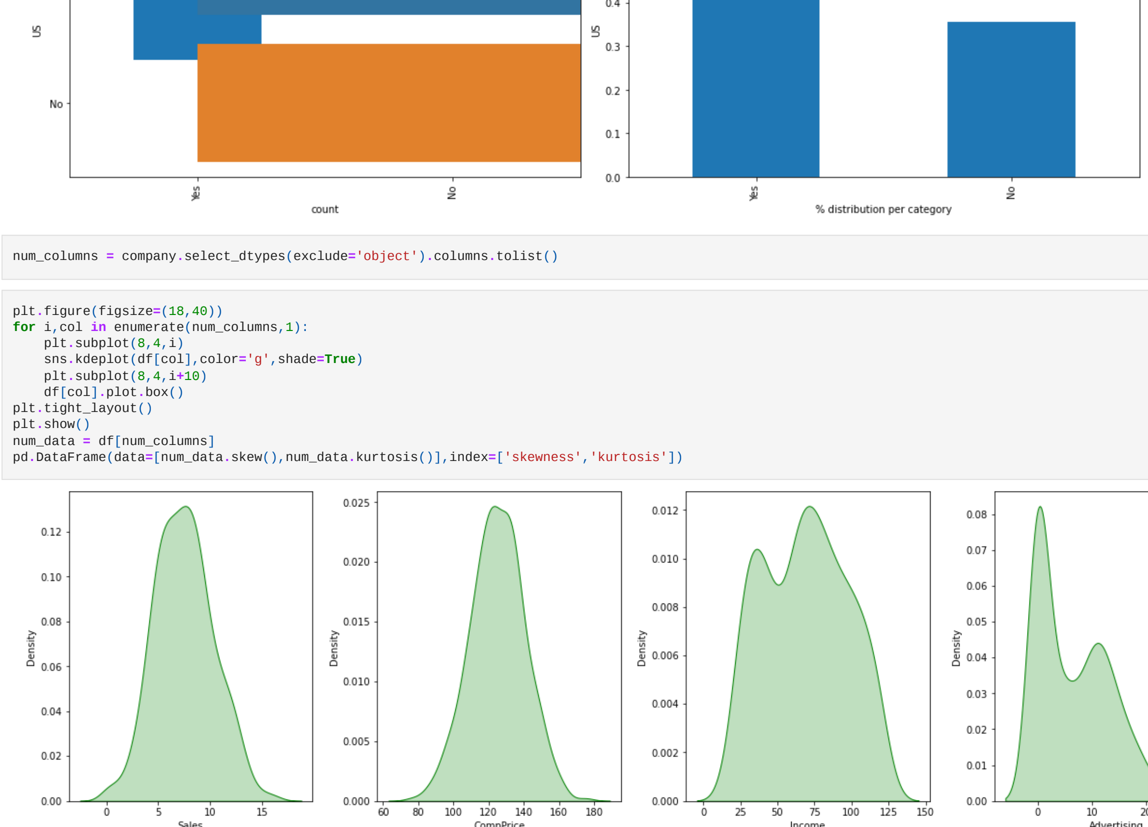
In [12]:
```
plt.rcParams["figure.figsize"] = 9,5
```

In [13]:
```
plt.figure(figsize=(16,5))
print("Skew: {}".format(company['Sales'].skew()))
print("Kurtosis: {}".format(company['Sales'].kurtosis()))
ax = sns.kdeplot(company['Sales'],shade=True,color='g')
plt.xticks([i for i in range(0,20,1)])
plt.show()
```

```
Skew: 0.185560363187 21578
Kurtosis: -0.08087736743346197
```

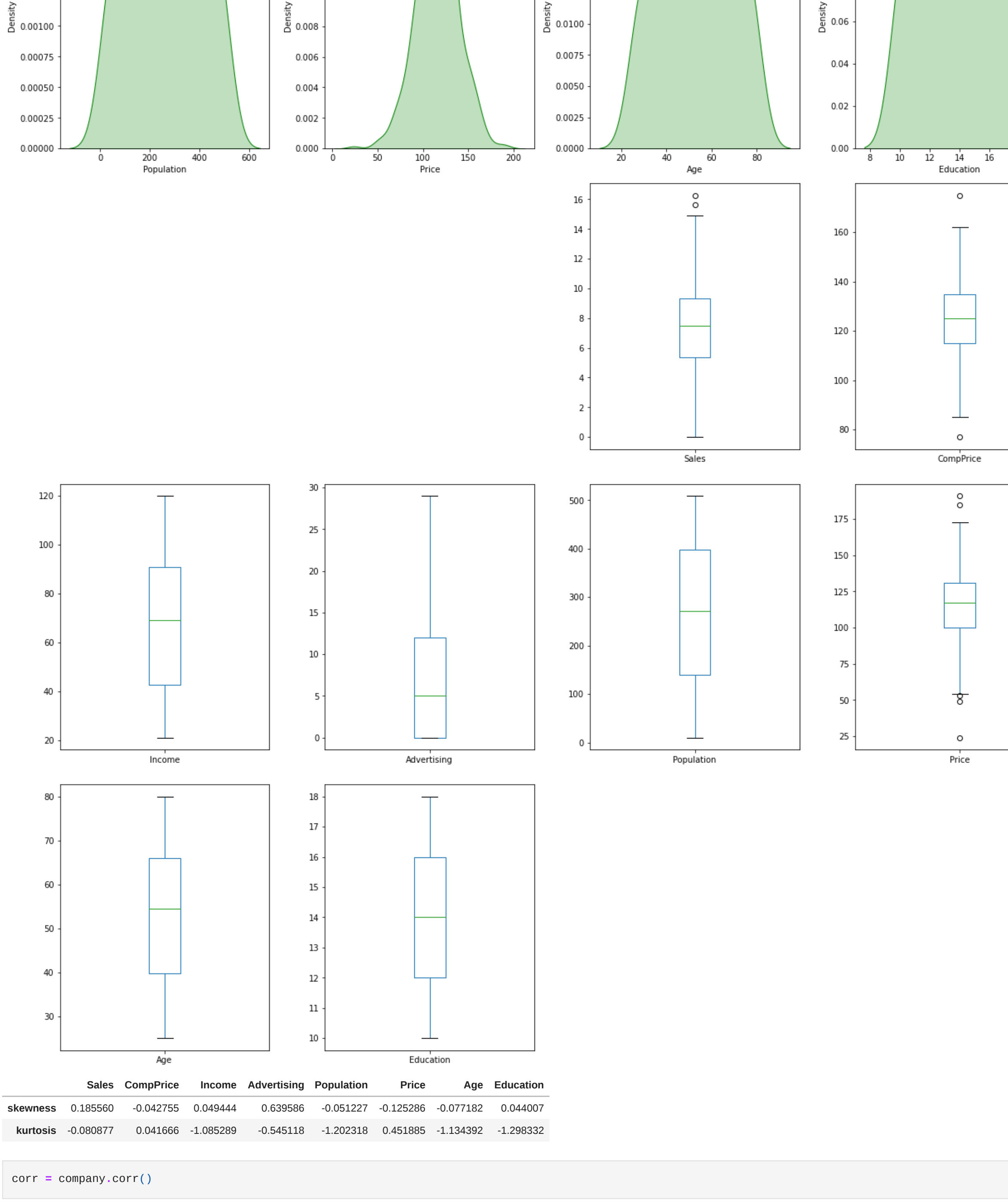
In [14]:
```
obj_colum = company.select_dtypes(include='object').columns.tolist()
```

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



In [17]:
```
num_columns = company.select_dtypes(exclude='object').columns.tolist()
```

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



Out[18]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
skewness	0.185560	-0.042755	0.049444	0.639586	-0.051227	-0.125286	-0.077182	0.044007
kurtosis	-0.080877	0.041666	-1.085289	-0.545118	-1.202318	0.451885	-1.134392	-1.298332

In [19]:
```
corr = company.corr()
```

In [20]:
```
company = pd.get_dummies(company, columns = ['ShelveLoc','Urban','US'])
```

In [21]:
```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

Out[21]: <AxesSubplot:>

In [ ]:

In [23]:
```
company["sales"]="small"
company.loc[company["Sales"]>7.49,"sales"]="large"
company.drop(["Sales"],axis=1,inplace=True)
```

In [24]:
```
X = company.iloc[:,0:14]
y = company.iloc[:,14]
```

In [25]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

In [26]:
```
y_train.value_counts()
```

Out[26]:
```
large    162
small    158
Name: sales, dtype: int64
```

In [27]:
```
model =RF(n_jobs=4,n_estimators = 150, oob_score =True,criterion ='entropy')
model.fit(x_train,y_train)
model.oob_score_
```

Out[27]: 0.80625

In [28]:
```
pred_train = model.predict(x_train)
```

In [29]:
```
accuracy_score(y_train,pred_train)
```

Out[29]: 1.0

In [30]:
```
confusion_matrix(y_train,pred_train)
```

Out[30]:
```
array([[162,   0],
       [  0, 158]], dtype=int64)
```

In [31]:
```
pred_test = model.predict(x_test)
```

In [32]:
```
accuracy_score(y_test,pred_test)
```

Out[32]: 0.75

In [33]:
```
confusion_matrix(y_test,pred_test)
```

Out[33]:
```
array([[26, 11],
       [ 9, 34]], dtype=int64)
```

In [34]:
```
df_t=pd.DataFrame({'Actual':y_test, 'Predicted':pred_test})
```

In [35]:
```
df_t
```

Out[35]:

	Actual	Predicted
325	large	large
230	small	small
135	small	large
365	small	small
71	small	small
...	...	...
119	small	small
326	small	large
131	small	small
158	large	large
170	large	large

80 rows × 2 columns

In [37]:
```
cols = list(company.columns)
```

In [38]:
```
predictors = cols[0:14]
target = cols[14]
```

In [39]:
```
tree1 = model.estimators_[20]
```

In [40]:
```
dot_data = StringIO()
```

In [42]:
```
export_graphviz(tree1, out_file = dot_data, feature_names =predictors, class_names = target, filled =True,rounded=True,impurity =False,proportion=False)
```

In [43]:
```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

In [44]:
```
graph.write_png('company_full.png')
```

Out[44]: True

In [ ]:

In [45]:
```
rf_small = RF(n_estimators=10, max_depth = 3)
```

In [46]:
```
rf_small.fit(x_train,y_train)
```

Out[46]: RandomForestClassifier(max_depth=3, n_estimators=10)

In [47]:
```
tree_small = rf_small.estimators_[5]
```

In [48]:
```
export_graphviz(tree_small, out_file = dot_data, feature_names = predictors, rounded = True, precision = 1)
```

In [49]:
```
graph_small = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

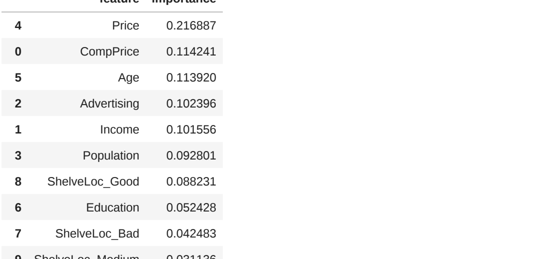
In [50]:
```
graph.write_png('company_small.png')
```

Out[50]: True

In [51]:
```
img = mpimg.imread('company_full.png')
```

In [52]:
```
plt.imshow(img)
```

Out[52]: <matplotlib.image.AxesImage at 0x1408075b3d0>



In [53]:
```
model.feature_importances_
```

Out[53]:
```
array([0.11424132, 0.10155598, 0.1023964 , 0.09280051, 0.21688652,
       0.11391961, 0.0524279 , 0.04248318, 0.08823116, 0.03113596,
       0.0111756 , 0.01213655, 0.01063965, 0.00996967])
```

In [54]:
```
fi = pd.DataFrame({'feature': list(x_train.columns),
                   'importance': model.feature_importances_}).\
                    sort_values('importance', ascending = False)
```

In [55]:
```
fi
```

Out[55]:

	feature	importance
4	Price	0.216887
0	CompPrice	0.114241
5	Age	0.113920
2	Advertising	0.102396
1	Income	0.101556
3	Population	0.092801
8	ShelveLoc_Good	0.088231
6	Education	0.052428
7	ShelveLoc_Bad	0.042483
9	ShelveLoc_Medium	0.031136
11	Urban_Yes	0.012137
10	Urban_No	0.011176
12	US_No	0.010640
13	US_Yes	0.009970

In [ ]:

In [ ]: