In [2]:

```
#Importing all liabraries
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import GridSearchCV
```

In [3]:

```
company_data=pd.read_csv("Company_Data (1).csv")
company_data.head(10)
```

Out[3]:

	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
5	10.81	124	113	13	501	72	Bad	78	16	N
6	6.63	115	105	0	45	108	Medium	71	15	Y
7	11.85	136	81	15	425	120	Good	67	10	Y
8	6.54	132	110	0	108	124	Medium	76	10	N
9	4.69	132	113	0	131	124	Medium	76	17	N

Initial Investigation

In [4]:

```
company_data.shape
```

Out[4]:

```
(400, 11)
```

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
```

In [6]:

```
company_data.isna().sum()
```

Out[6]:

```
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 [7]:

```
company_data.describe(include='all')
```

Out[7]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400	40
unique	NaN	NaN	NaN	NaN	NaN	NaN	3	
top	NaN	NaN	NaN	NaN	NaN	NaN	Medium	
freq	NaN	NaN	NaN	NaN	NaN	NaN	219	
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	NaN	5
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	NaN	1
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	NaN	2
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	NaN	3
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	NaN	5
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	NaN	6
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	NaN	8

Data Preprocessing

In [8]:

```
#Converted numerical into categorical as per the problem statement
company_data.loc[company_data['Sales']<= 9.50, '<= 9.50']='low'
company_data.loc[company_data['Sales']>9.50, '> 9.50']='high'
```

In [9]:

```
#Converting all data into numerical data
company_data[['Urban','US']]=pd.get_dummies(company_data[['Urban','US']],drop_first=True)
```

In [10]:

```
company_data['ShelveLoc']=pd.get_dummies(company_data['ShelveLoc'],drop_first=True)
```

In [11]:

```
company_data.head(10)
```

Out[11]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urba
0	9.50	138	73	11	276	120	0	42	17	
1	11.22	111	48	16	260	83	1	65	10	
2	10.06	113	35	10	269	80	0	59	12	
3	7.40	117	100	4	466	97	0	55	14	
4	4.15	141	64	3	340	128	0	38	13	
5	10.81	124	113	13	501	72	0	78	16	
6	6.63	115	105	0	45	108	0	71	15	
7	11.85	136	81	15	425	120	1	67	10	
8	6.54	132	110	0	108	124	0	76	10	
9	4.69	132	113	0	131	124	0	76	17	

In [12]:

```
del company_data['Sales']
```

In [14]:

```
del company_data['<= 9.50']
```

In [15]:

```
company_data.head(20)
```

Out[15]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	138	73	11	276	120	0	42	17	1	1
1	111	48	16	260	83	1	65	10	1	1
2	113	35	10	269	80	0	59	12	1	1
3	117	100	4	466	97	0	55	14	1	1
4	141	64	3	340	128	0	38	13	1	0
5	124	113	13	501	72	0	78	16	0	1
6	115	105	0	45	108	0	71	15	1	0
7	136	81	15	425	120	1	67	10	1	1
8	132	110	0	108	124	0	76	10	0	0
9	132	113	0	131	124	0	76	17	0	1
10	121	78	9	150	100	0	26	10	0	1
11	117	94	4	503	94	1	50	13	1	1
12	122	35	2	393	136	0	62	18	1	0
13	115	28	11	29	86	1	53	18	1	1
14	107	117	11	148	118	1	52	18	1	1
15	149	95	5	400	144	0	76	18	0	0
16	118	32	0	284	110	1	63	13	1	0
17	147	74	13	251	131	1	52	10	1	1
18	110	110	0	408	68	1	46	17	0	1
19	129	76	16	58	121	0	69	12	1	1

In [16]:

```
company_data.fillna(value=0,axis=0,inplace=True)
```

In [17]:

```
company_data.head(20)
```

Out[17]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	138	73	11	276	120	0	42	17	1	´
1	111	48	16	260	83	1	65	10	1	´
2	113	35	10	269	80	0	59	12	1	´
3	117	100	4	466	97	0	55	14	1	´
4	141	64	3	340	128	0	38	13	1	(
5	124	113	13	501	72	0	78	16	0	´
6	115	105	0	45	108	0	71	15	1	(
7	136	81	15	425	120	1	67	10	1	´
8	132	110	0	108	124	0	76	10	0	(
9	132	113	0	131	124	0	76	17	0	´
10	121	78	9	150	100	0	26	10	0	´
11	117	94	4	503	94	1	50	13	1	´
12	122	35	2	393	136	0	62	18	1	(
13	115	28	11	29	86	1	53	18	1	´
14	107	117	11	148	118	1	52	18	1	´
15	149	95	5	400	144	0	76	18	0	(
16	118	32	0	284	110	1	63	13	1	(
17	147	74	13	251	131	1	52	10	1	´
18	110	110	0	408	68	1	46	17	0	´
19	129	76	16	58	121	0	69	12	1	´

In [20]:

```
company_data['Sales']=pd.get_dummies(company_data['> 9.50'],drop_first=True)
```

In [21]:

```
company_data.head(10)
```

Out[21]:

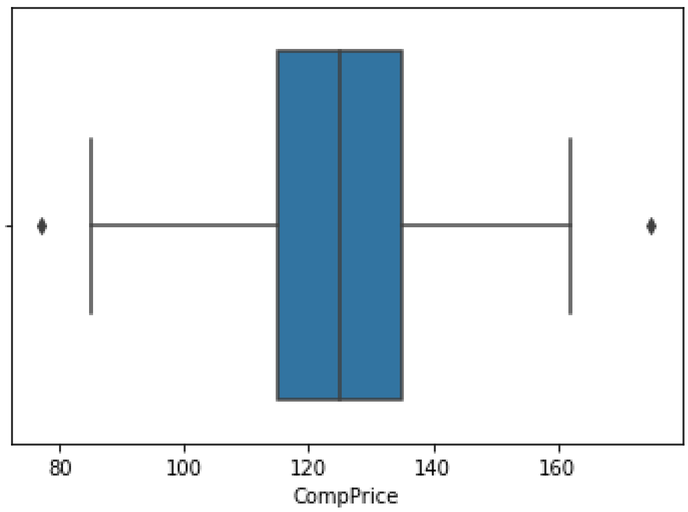
	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	138	73	11	276	120	0	42	17	1	1
1	111	48	16	260	83	1	65	10	1	1
2	113	35	10	269	80	0	59	12	1	1
3	117	100	4	466	97	0	55	14	1	1
4	141	64	3	340	128	0	38	13	1	0
5	124	113	13	501	72	0	78	16	0	1
6	115	105	0	45	108	0	71	15	1	0
7	136	81	15	425	120	1	67	10	1	1
8	132	110	0	108	124	0	76	10	0	0
9	132	113	0	131	124	0	76	17	0	1

In [27]:

```
sns.boxplot(x='CompPrice',data=company_data)
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1c6914208>
```



In [68]:

```
company_data['Sales'].unique()
```

Out[68]:

```
array([0, 1], dtype=uint64)
```

Model Building

In [28]:

```
X=company_data.iloc[:, :-2]
y=company_data[["Sales"]]
```

In [32]:

```
X
```

Out[32]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	U
0	138	73	11	276	120	0	42	17	1	
1	111	48	16	260	83	1	65	10	1	
2	113	35	10	269	80	0	59	12	1	
3	117	100	4	466	97	0	55	14	1	
4	141	64	3	340	128	0	38	13	1	
...	...	...	...	...	...	...	...	...	...	
395	138	108	17	203	128	1	33	14	1	
396	139	23	3	37	120	0	55	11	0	
397	162	26	12	368	159	0	40	18	1	
398	100	79	7	284	95	0	50	12	1	
399	134	37	0	27	120	1	49	16	1	

400 rows × 10 columns

In [33]:

```
y
```

Out[33]:

	Sales
0	0
1	1
2	1
3	0
4	0
...	...
395	1
396	0
397	0
398	0
399	1

400 rows × 1 columns

In [34]:

```
company_data.std()
```

Out[34]:

```
CompPrice       15.334512
Income          27.986037
Advertising      6.650364
Population     147.376436
Price           23.676664
ShelveLoc        0.409589
Age             16.200297
Education        2.620528
Urban            0.456614
US               0.479113
Sales            0.413062
dtype: float64
```

In [38]:

```
from sklearn.preprocessing import StandardScaler
scaled_data=StandardScaler()
x_scale=scaled_data.fit_transform(X)
```

Model Training|Model Testing|Model Evaluation

In [39]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x_scale,y,test_size=0.20,random_state=21)
```

In [40]:

```
X_train
```

Out[40]:

```
array([[ 2.02574788, -0.95372149,  0.95828906, ...,  0.03820804,
         0.64686916,  0.74188112],
       [-0.71660219,  1.58643516, -0.69782715, ...,  0.4202884 ,
        -1.54590766,  0.74188112],
       [-0.78189624, -1.27571318, -0.99893918, ..., -0.72595268,
         0.64686916, -1.34792485],
       ...,
       [ 0.26280855,  1.72954258,  0.20550897, ..., -1.4901134 ,
         0.64686916,  0.74188112],
       [ 0.78516094,  1.72954258, -0.24615909, ..., -1.4901134 ,
         0.64686916,  0.74188112],
       [-0.91248434,  1.30022032, -0.99893918, ..., -1.4901134 ,
        -1.54590766, -1.34792485]])
```

In [41]:

```
y_train
```

Out[41]:

	Sales
191	0
278	0
272	1
128	0
285	0
...	...
368	1
48	0
260	0
312	0
207	0

320 rows × 1 columns

In [59]:

```
rf_model=RandomForestClassifier(n_estimators=120,n_jobs=-1,random_state=21)
rf_model.fit(X_train,y_train)
```

Out[59]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=120,
                       n_jobs=-1, oob_score=False, random_state=21, verbose=
0,
                       warm_start=False)
```

In [60]:

```
y_pred_train=rf_model.predict(X_train)
y_pred_test=rf_model.predict(X_test)
```

In [61]:

```
print(classification_report(y_test,y_pred_test))
print(confusion_matrix(y_test,y_pred_test))
print(accuracy_score(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        74
           1       0.29      0.33      0.31         6

    accuracy                           0.89        80
   macro avg       0.62      0.63      0.62        80
weighted avg       0.90      0.89      0.89        80

[[69  5]
 [ 4  2]]
0.8875
```

Model Testing Accuracy: 88.75%

In [64]:

```
#we are getting the accuracy 88.75,we can say model is good
```

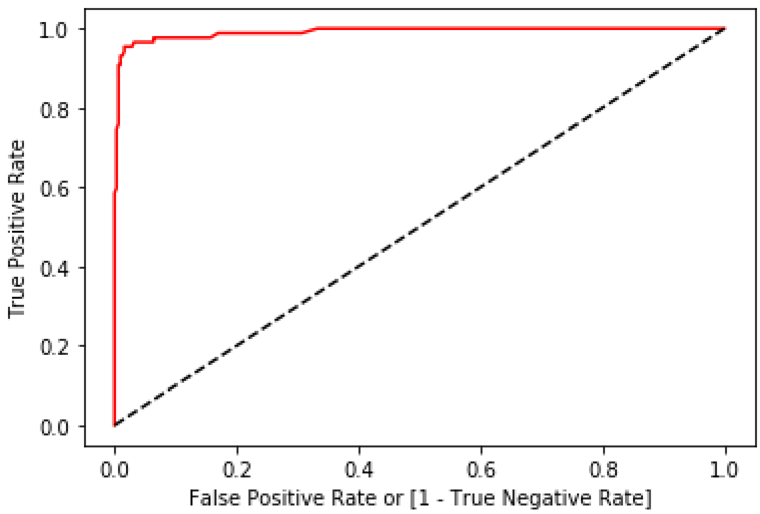
In [71]:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

fpr, tpr, thresholds = roc_curve(y, rf_model.predict_proba (x_scale)[:,1])

auc = roc_auc_score(y_test, y_pred_test)
print(auc)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red', label='rf_model ( area  = %0.2f)'%auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
0.6328828828828829
```

Out[71]:

```
Text(0, 0.5, 'True Positive Rate')
```



In [ ]:

```
#we can say that our ROC curve lies at 89%
```