In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
```

executed in 15.7s, finished 16:03:26 2022-01-05

In [2]:

```
sales_data=pd.read_csv("Company_Data.csv")
sales_data.head()
```

executed in 147ms, finished 16:03:29 2022-01-05

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

Initial investigation

In [3]:

```
sales_data.shape
```

Out[3]:

```
(400, 11)
```

In [4]:

```
sales_data.dtypes
```

Out[4]:

```
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 [5]:

```
sales_data.isnull().sum()
```

Out[5]:

```
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 [6]:

```
sales_data.isnull().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]:

```
sales_data.describe()
```

Out[7]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	E
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	4(
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	1
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	1
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	1
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	1
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	1
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	1

Number of features and records in the given data set is 11 and 400 respesctively

There is no null values in the data set

The categorical data can be converted into numeric data type by using encoder so that the model can learn the things more easily

Data preparation

In [8]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [9]:

```
sales_data['ShelveLoc']=le.fit_transform(sales_data['ShelveLoc'])
sales_data['Urban']=le.fit_transform(sales_data['Urban'])
sales_data['US']=le.fit_transform(sales_data['US'])
sales_data.dtypes
```

Out[9]:

```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc        int32
Age              int64
Education        int64
Urban            int32
US               int32
dtype: object
```

Converting sales to category of high,medium and low sales

In [10]:

```
sales_data.insert(11,'sales_category','')
sales_data
```

Out[10]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
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	2	59	12
3	7.40	117	100	4	466	97	2	55	14
4	4.15	141	64	3	340	128	0	38	13
...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	1	33	14
396	6.14	139	23	3	37	120	2	55	11
397	7.41	162	26	12	368	159	2	40	18
398	5.94	100	79	7	284	95	0	50	12
399	9.71	134	37	0	27	120	1	49	16

400 rows × 12 columns

In [11]:

```
for i in range(0,len(sales_data['Sales']),1):
    if sales_data['Sales'][i]>=11.0:
        sales_data["sales_category"][i]='high'
    elif sales_data['Sales'][i]<=6.0:
        sales_data['sales_category'][i]='Low'
    else:
        sales_data['sales_category'][i]='Medium'
```

In [12]:

```
sales_data['sales_category'].nunique()
```

Out[12]:

```
3
```

In [13]:

```
sales_data.head()
```

Out[13]:

	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	2	59	12	
3	7.40	117	100	4	466	97	2	55	14	
4	4.15	141	64	3	340	128	0	38	13	

Model building

In [14]:

```
x=sales_data.iloc[:,1:11]
y=sales_data.iloc[:,11:12]
```

In [15]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Model training before selecting important feature

In [16]:

```
from sklearn.ensemble import RandomForestClassifier
rf_model=RandomForestClassifier()
```

In [18]:

```
rf_model.fit(x_train,y_train)
y_pred=rf_model.predict(x_test)
```

In [19]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [20]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         Low       0.69      0.44      0.54        25
      Medium       0.61      0.86      0.71        43
        high       0.67      0.17      0.27        12

    accuracy                           0.62        80
   macro avg       0.65      0.49      0.50        80
weighted avg       0.64      0.62      0.59        80
```

In [21]:

```
print(accuracy_score(y_test,y_pred))
```

```
0.625
```

In [24]:

```
confusion_matrix_data=confusion_matrix(y_test,y_pred)
confusion_matrix_data
```

Out[24]:

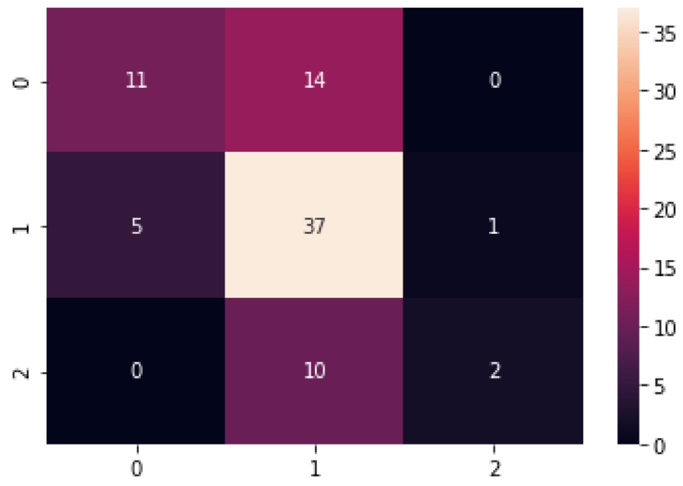
```
array([[11, 14,  0],
       [ 5, 37,  1],
       [ 0, 10,  2]], dtype=int64)
```

In [26]:

```
sns.heatmap(confusion_matrix_data,annot=True)
```

Out[26]:

```
<AxesSubplot:>
```



Hyperparameter tweaking by using GridsearchCV

In [27]:

```
from sklearn.model_selection import GridSearchCV
grid_model=GridSearchCV(estimator = rf_model,param_grid={'criterion':['entropy','gini'],
                                                        'max_depth':[2,4,8,10],
                                                         'min_samples_split':[2,4,6,8],
                                                         'min_samples_leaf':[1,2,3,4],
                                                         'n_estimators':[20,30,20,100,150]})
grid_model.fit(x_train,y_train)
print(grid_model.best_params_)
print(grid_model.best_score_)
```

```
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_sample
s_split': 4, 'n_estimators': 30}
0.734375
```

In [30]:

```
rf_model_cv=RandomForestClassifier(max_depth=10,min_samples_leaf=2,criterion='entropy',n_es
rf_model_cv.fit(x_train,y_train)
y_pred=rf_model_cv.predict(x_test)
print(accuracy_score(y_test,y_pred))
```

```
0.6375
```

Feature importance plot

In [31]:

```
feature_rf=x_train.columns
len(feature_rf)
```

Out[31]:

```
10
```

In [32]:

```
imp_feature_rf=rf_model.feature_importances_
len(imp_feature_rf)
```

Out[32]:

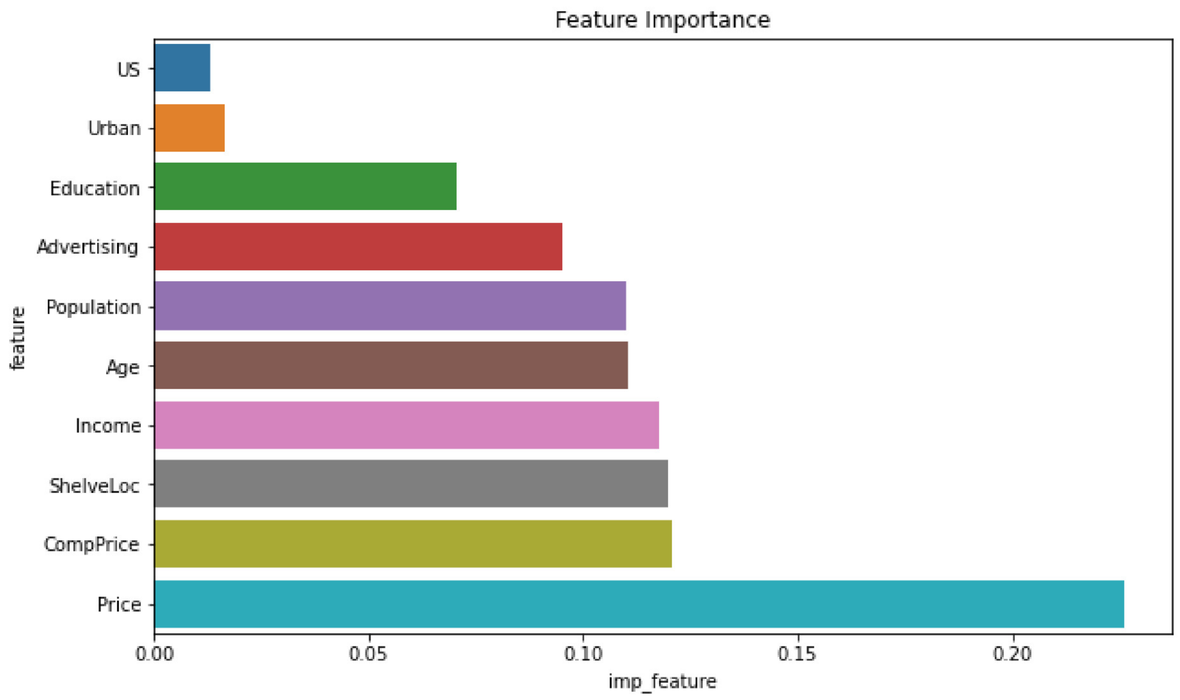
```
10
```

In [33]:

```
data_rf=pd.DataFrame({'feature':feature_rf,'imp_feature':imp_feature_rf})
data_rf=data_rf.sort_values('imp_feature')
```

In [35]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,6))
plt.title('Feature Importance')
sns.barplot(y='feature', x='imp_feature', data=data_rf)
plt.show()
```



The plot clearly shows that some of the features donot contribute much for model building

Hence the accuracy can be imporved by droping those insignificant features

Feature selection by feature importance for Decision tree

In [36]:

```
from sklearn.feature_selection import SelectFromModel
```

In [37]:

```
selector=SelectFromModel(estimator=RandomForestClassifier())
selector.fit(x_train,y_train)
```

Out[37]:

```
SelectFromModel(estimator=RandomForestClassifier())
```

In [38]:

```
selector.get_support()
```

Out[38]:

```
array([ True,  True, False,  True,  True,  True,  True, False, False,
       False])
```

In [39]:

```
len(x_train.columns)
```

Out[39]:

```
10
```

In [40]:

```
feature=x_train.columns[selector.get_support()]
```

In [41]:

```
len(x_train.columns[selector.get_support()])
```

Out[41]:

```
6
```

Out of 10 features only 5 features is selscted for building models

In [42]:

```
x_train_rf=selector.transform(x_train)
x_test_rf=selector.transform(x_test)
```

In [45]:

```
rf_model_imp=RandomForestClassifier().fit(x_train_rf,y_train)
y_pred_imp=rf_model_imp.predict(x_test_rf)
```

In [46]:

```
print(accuracy_score(y_test,y_pred_imp))
```

```
0.6625
```

In [47]:

```
print(confusion_matrix(y_test,y_pred_imp))
```

```
[[14 11  0]
 [ 7 35  1]
 [ 0  8  4]]
```

In [48]:

```
print(classification_report(y_test,y_pred_imp))
```

```
              precision    recall  f1-score   support

         Low       0.67      0.56      0.61        25
      Medium       0.65      0.81      0.72        43
        high       0.80      0.33      0.47        12

    accuracy                           0.66        80
   macro avg       0.70      0.57      0.60        80
weighted avg       0.68      0.66      0.65        80
```

In [49]:

```
imp_feature=rf_model_imp.feature_importances_
imp_feature
```

Out[49]:

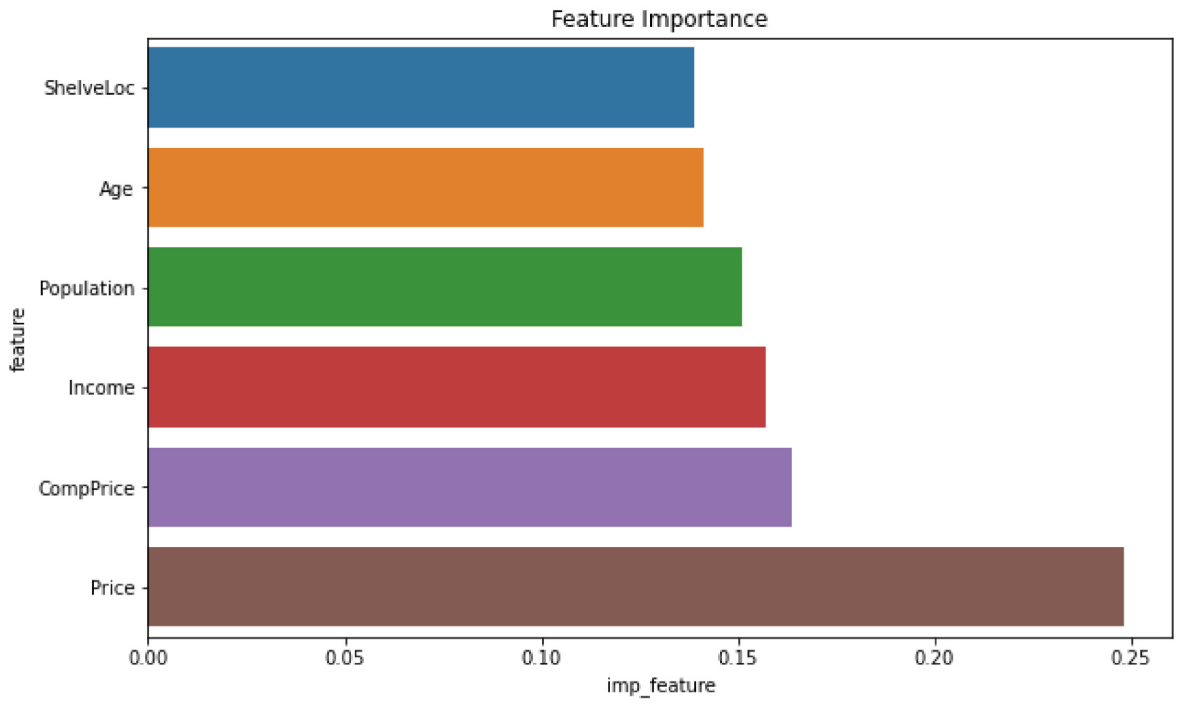
```
array([0.16352072, 0.15718252, 0.15106064, 0.24807131, 0.13891877,
       0.14124603])
```

In [50]:

```
data_imp=pd.DataFrame({'feature':feature,'imp_feature':imp_feature})
data_imp=data_imp.sort_values('imp_feature')
```

In [51]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,6))
plt.title('Feature Importance')
sns.barplot(y='feature', x='imp_feature', data=data_imp)
plt.show()
```



Recursive feature elimination

In [52]:

```
from sklearn.feature_selection import RFE
```

In [53]:

```
selector_rfe=RFE(RandomForestClassifier())
selector_rfe.fit(x_train,y_train)
```

Out[53]:

```
RFE(estimator=RandomForestClassifier())
```

In [54]:

```
selector_rfe.get_support()
```

Out[54]:

```
array([ True,  True, False,  True,  True, False,  True, False, False,
       False])
```

In [55]:

```
feature_rfe=x_train.columns[selector_rfe.get_support()]
feature_rfe
```

Out[55]:

```
Index(['CompPrice', 'Income', 'Population', 'Price', 'Age'], dtype='object')
```

In [56]:

```
len(x_train.columns[selector_rfe.get_support()])
```

Out[56]:

```
5
```

Here 5 out of 10 feature is selected as an important feature

In [57]:

```
x_train_rfe=selector_rfe.transform(x_train)
x_test_rfe=selector_rfe.transform(x_test)
```

In [66]:

```
rf_model_rfe=RandomForestClassifier().fit(x_train_rfe,y_train)
```

In [67]:

```
y_pred_rfe=rf_model_rfe.predict(x_test_rfe)
```

In [68]:

```
print(accuracy_score(y_test,y_pred_rfe))
```

```
0.575
```

In [69]:

```
print(confusion_matrix(y_test,y_pred_rfe))
```

```
[[13 12  0]
 [12 31  0]
 [ 1  9  2]]
```

In [70]:

```
print(classification_report(y_test,y_pred_rfe))
```

```
              precision    recall  f1-score   support

         Low       0.50      0.52      0.51        25
      Medium       0.60      0.72      0.65        43
        high       1.00      0.17      0.29        12

    accuracy                           0.57        80
   macro avg       0.70      0.47      0.48        80
weighted avg       0.63      0.57      0.55        80
```

In [72]:

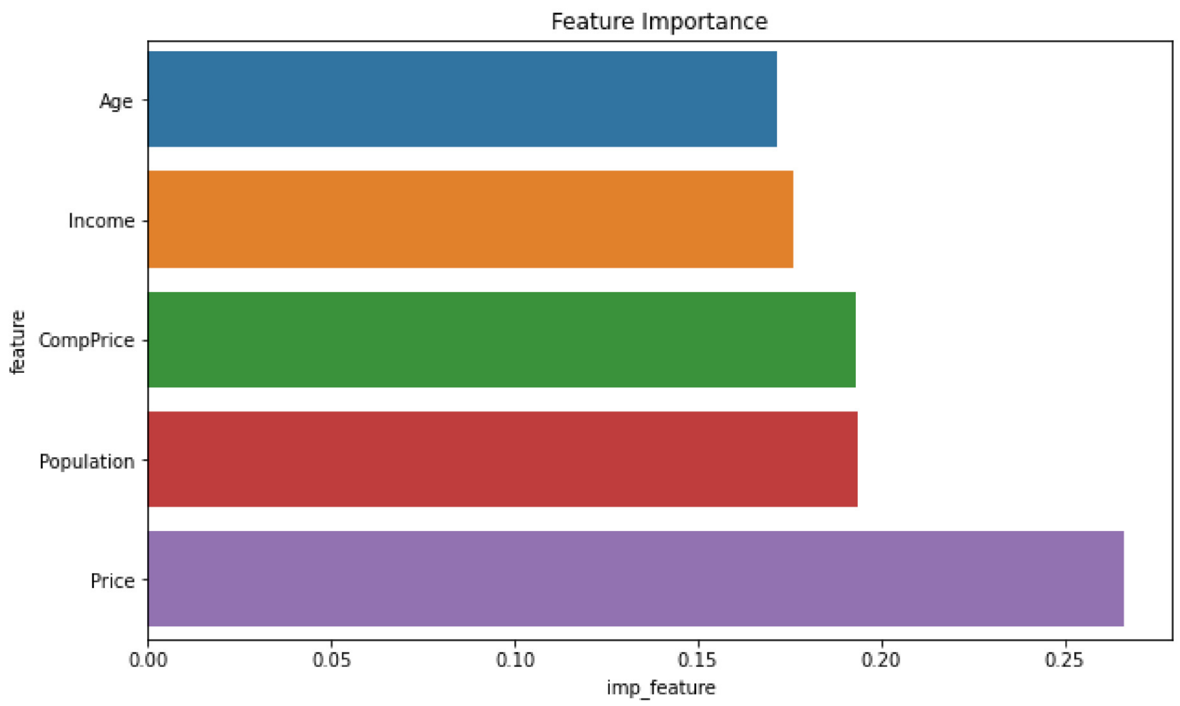
```
imp_feature_rfe=rf_model_rfe.feature_importances_
```

In [73]:

```
data_rfe=pd.DataFrame({'feature':feature_rfe,'imp_feature':imp_feature_rfe})
data_rfe=data_rfe.sort_values('imp_feature')
```

In [74]:

```
plt.figure(figsize=(10,6))
plt.title('Feature Importance')
sns.barplot(y='feature', x='imp_feature', data=data_rfe)
plt.show()
```



Inference

Price is the feature which affect/contibute more for the sales

Competerior price fallows price which affect the most