1.Import datatypes

In [1]:
```
#Task 2:A cloth manufacturing company is interested to know about the segment or
```

In [2]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix
import matplotlib.pyplot as plt #import Libraries
```

2.Import libraries

In [3]:
```
data=pd.read_csv('Company_Data.csv') #import dataset
data.head()  #Viewing top 5 rows of dataframe
```

Out[3]:

	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

3 Data Undestanding

In [4]:
```
data.info() #infomation of all datapoint
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    int64
 2   Income       400 non-null    int64
 3   Advertising  400 non-null    int64
 4   Population   400 non-null    int64
 5   Price        400 non-null    int64
 6   ShelveLoc    400 non-null    object
 7   Age          400 non-null    int64
 8   Education    400 non-null    int64
 9   Urban        400 non-null    object
 10  US           400 non-null    object
dtypes: float64(1), int64(7), object(3)
memory usage: 34.5+ KB
```

In [5]:
```
data.dtypes #datatypes of each columns
```

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]:
```
data.Sales.max(),data.Sales.min()  #Find maximun and minimum value in sales colum
```

Out[6]:
```
(16.27, 0.0)
```

In [7]:
```
data['Sales']=pd.cut(data['Sales'],bins=[1,5,12,17],labels=["low","medium","high"
```

In [8]:
```
le=LabelEncoder()
for i in data.columns:
    if data[i].dtypes=="object" or "category":
        data[i]=le.fit_transform(data[i])  #apply label encoder in non numeric co
```

In [9]:
```
data.head()
```

Out[9]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban
0	2	49	51	11	141	54	0	17	7	1
1	2	22	27	16	129	18	1	40	0	1
2	2	24	14	10	138	15	2	34	2	1
3	2	28	77	4	249	31	2	30	4	1
4	1	52	42	3	178	62	0	13	3	1

In [10]:
```
data.dtypes #datatype of each columns
```

Out[10]:
```
Sales          int32
CompPrice      int64
Income         int64
Advertising    int64
Population     int64
Price          int64
ShelveLoc      int32
Age            int64
Education      int64
Urban          int32
US             int32
dtype: object
```

In [11]:
```
list(data.columns) #list of all columns name
```

Out[11]:
```
['Sales',
 'CompPrice',
 'Income',
 'Advertising',
 'Population',
 'Price',
 'ShelveLoc',
 'Age',
 'Education',
 'Urban',
 'US']
```

In [12]:
```
data.rename(columns={"Sales":'sales','CompPrice':'compPrice','Income':'income',"A
                     "Population":'population','Price':"price","ShelveLoc":"shelve
                     "Education":"education","Urban":"urban","US":"us"},inplace=Tr
```

In [13]:
```
data.head()
```

Out[13]:

	sales	compPrice	income	advertising	population	price	shelveLoc	age	education	urban	us
0	2	49	51	11	141	54	0	17	7	1	1
1	2	22	27	16	129	18	1	40	0	1	1
2	2	24	14	10	138	15	2	34	2	1	1
3	2	28	77	4	249	31	2	30	4	1	1
4	1	52	42	3	178	62	0	13	3	1	0

4. Data Preparing

In [14]:
```
data['sales'].value_counts()  #Show values of each labels
```

Out[14]:
```
2    296
1     72
0     27
3      5
Name: sales, dtype: int64
```

In [15]:
```
x=data.drop(['sales'],axis=1)
y=data['sales']   #Devide data in input and output columns
```

In [16]:
```
#Dividing data for training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,stratify=y,random_state=12,shu
```

5.Build The Model(Gini criteria)

In [17]:
```
#Building Decision Tree Classifier using gini Criteria
```

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='gini',max_depth=7)
model.fit(x_train,y_train)  #Build Model
```

Out[18]:
```
DecisionTreeClassifier(max_depth=7)
```

In [19]:
```
#predict on train data
y_train_pred=model.predict(x_train)
```

In [20]:
```
#Find out accuracy
accuracy_score(y_train,y_train_pred)
```

Out[20]:
```
0.9633333333333334
```

In [21]:
```
#Find out confision_matrix
confusion_matrix(y_train,y_train_pred)
```

Out[21]:
```
array([[ 19,   0,   1,   0],
       [  0,  52,   2,   0],
       [  1,   5, 216,   0],
       [  0,   2,   0,   2]], dtype=int64)
```

In [22]:
```
#predict on test data
y_test_pred=model.predict(x_test)
```

In [23]:
```
#Find out accuracy
accuracy_score(y_test_pred,y_test)
```

Out[23]:
```
0.81
```

In [24]:
```
#confusion_mtrix
confusion_matrix(y_test,y_test_pred)
```

Out[24]:
```
array([[ 2,  1,  4,  0],
       [ 0, 14,  4,  0],
       [ 4,  4, 65,  1],
       [ 0,  0,  1,  0]], dtype=int64)
```

In [25]:
```
acc = [['Train_accuracy', '96%'], ['Test_accuracy', '81%']]
acc=pd.DataFrame(acc)
acc
```

Out[25]:

	0	1
0	Train_accuracy	96%
1	Test_accuracy	81%

6 Model Building (entropy criteria)

In [26]:
```
#Build Decision Tree Classifier using Entropy Criteria
```

In [27]:
```
model1=DecisionTreeClassifier(criterion='entropy',max_depth=7)
```

In [28]:
```
model1.fit(x_train,y_train) #fit  the model
```

Out[28]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [29]:
```
#predict on train data
y_pred_train=model.predict(x_train)
```

In [30]:
```
#Find out accuracy
accuracy_score(y_train,y_pred_train)
```

Out[30]:
```
0.9633333333333334
```

In [31]:
```
#predict on test data
y_pred_test=model.predict(x_test)
```

In [32]:
```
#Find out accuracy
accuracy_score(y_test,y_pred_test)
```

Out[32]:
```
0.81
```

In [33]:
```
acc = [['Train_accuracy', '96%'], ['Test_accuracy', '81%']]
acc=pd.DataFrame(acc)
acc
```

Out[33]:

	0	1
0	Train_accuracy	96%
1	Test_accuracy	81%

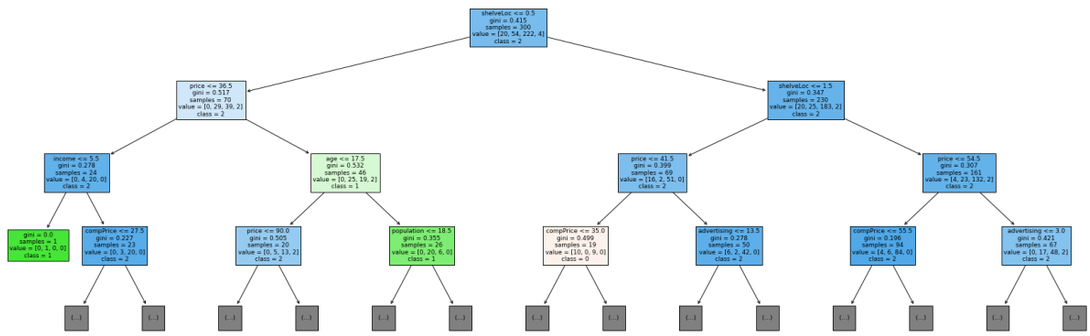
7.Plot the tree

In [34]:
```
#plot the tree
from sklearn import tree
```

In [35]:
```
fn=['compPrice','income','advertising','population','price','shelveLoc','age','ed
cn=["0","1","2","3"]  #Labels
```

In [36]:
```
plt.figure(figsize=(30,10))   #Size of Tree

tree.plot_tree(model,max_depth=3,filled=True,feature_names=fn,class_names=cn)  #
plt.show()
```



8.Model Building (Random Forest(Entropy Criteria))

In [37]:
```
##Building Random Forest Classifier using Entropy Criteria.
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=15,criterion='entropy',max_depth=7)
rf.fit(x_train,y_train)   #fit the model
```

Out[38]:
```
RandomForestClassifier(criterion='entropy', max_depth=7, n_estimators=15)
```

In [39]:
```
#Predict on train Data
y_pred1=rf.predict(x_train)
```

In [40]:
```
# Find OAccuracy
accuracy_score(y_pred1,y_train)
```

Out[40]:
```
0.9533333333333334
```

In [41]:
```
#Predict on test Data
y_pred2=rf.predict(x_test)
```

In [42]:
```
#Accuracy
accuracy_score(y_pred2,y_test)
```

Out[42]:
```
0.78
```

In [43]:
```
acc = [['Train_accuracy', '95%'], ['Test_accuracy', '78%']]
acc=pd.DataFrame(acc)
acc
```

Out[43]:

	0	1
0	Train_accuracy	95%
1	Test_accuracy	78%

9.Model Building (Random Forest(Gini Criteria))

In [44]:
```
##Building Random Forest Classifier using Gini Criteria.
```

In [45]:
```
from sklearn.ensemble import RandomForestClassifier
rf1=RandomForestClassifier(n_estimators=15,criterion='gini',max_depth=10)
rf1.fit(x_train,y_train)   #fit the model
```

Out[45]:
```
RandomForestClassifier(max_depth=10, n_estimators=15)
```

In [46]:
```
#Predict on train Data
y_pred1=rf1.predict(x_train)
```

In [47]:
```
# Find OAccuracy
accuracy_score(y_pred1,y_train)
```

Out[47]:
```
0.99
```

In [48]:
```
#Predict on test Data
y_pred2=rf1.predict(x_test)
```

In [49]:
```
#Accuracy
accuracy_score(y_pred2,y_test)
```

Out[49]:
```
0.76
```

In [50]:
```
acc = [['Train_accuracy', '99%'], ['Test_accuracy', '76%']]
acc=pd.DataFrame(acc)
acc
```

Out[50]:

	0	1
0	Train_accuracy	99%
1	Test_accuracy	76%