Problem Statement

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

1. Importing Necessary Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import  DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn import preprocessing
```

2. Importing data

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

0	9.50	138	73	11	276	120	Bad	42	17
1	11.22	111	48	16	260	83	Good	65	10
2	10.06	113	35	10	269	80	Medium	59	12
3	7.40	117	100	4	466	97	Medium	55	14
4	4.15	141	64	3	340	128	Bad	38	13
...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14
396	6.14	139	23	3	37	120	Medium	55	11
397	7.41	162	26	12	368	159	Medium	40	18
398	5.94	100	79	7	284	95	Bad	50	12
399	9.71	134	37	0	27	120	Good	49	16

400 rows × 11 columns

3. Data Understanding

3.1 Initial Analysis

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
```

Note :we will create the nuemarical variable for the categorical data

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
```

4. Data Preparation

*Note : Since machine will not understand the objective type data, We will create the nuemarical variable for the categorical data

In [7]:
```
from sklearn import preprocessing
```

In [8]:
```
LabelEncoder = preprocessing.LabelEncoder()
company_data['ShelveLoc'] = LabelEncoder.fit_transform(company_data['ShelveLoc'])
company_data['Urban'] = LabelEncoder.fit_transform(company_data['Urban'])
company_data['US'] =  LabelEncoder.fit_transform(company_data['US'])
```

In [9]:
```
company_data
```

Out[9]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Ur
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	

In [10]:
```
company_data.dtypes
```

Out[10]:
```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc        int32
Age              int64
Education        int64
Urban            int32
US               int32
dtype: object
```

In [17]:
```
company_data['Sales'] = company_data['Sales'].astype('int')
```

In [18]:
```
company_data.dtypes
```

Out[18]:
```
Sales          int32
CompPrice      int64
Income         int64
Advertising    int64
Population     int64
Price          int64
ShelveLoc      int32
Age            int64
Education      int64
Urban          int32
US             int32
dtype: object
```

5. Model Building

- 2 steps in model building 1 - Separate input & output features

2. Go for train test split for model validation

5.1 Separate input and output features

In [19]:
```
X = company_data.drop(labels='Sales', axis=1)
y = company_data[['Sales']]
```

In [20]:
```
X
```

Out[20]:

	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	2	59	12	1	1
3	117	100	4	466	97	2	55	14	1	1
4	141	64	3	340	128	0	38	13	1	0
...	...	...	...	...	...	...	...	...	...	...
395	138	108	17	203	128	1	33	14	1	1
396	139	23	3	37	120	2	55	11	0	1
397	162	26	12	368	159	2	40	18	1	1
398	100	79	7	284	95	0	50	12	1	1
399	134	37	0	27	120	1	49	16	1	1

400 rows × 10 columns

In [21]:
```
y
```

Out[21]:

	Sales
0	9
1	11
2	10
3	7
4	4
...	...
395	12
396	6
397	7
398	5
399	9

400 rows × 1 columns

5.2 Train test split

In [22]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.20, random_st
```

In [23]:
```
X_train
```

93	145	30	0	67	104	2	55	17	1
23	121	31	0	292	109	2	79	10	1
299	135	40	17	497	96	2	54	17	0
13	115	28	11	29	86	1	53	18	1
90	115	22	0	491	103	2	64	11	0
...	...	...	...	...	...	...	...	...	...
255	123	81	8	198	81	0	80	15	1
72	115	45	0	432	116	2	25	15	1
396	139	23	3	37	120	2	55	11	0
235	126	32	8	95	132	2	50	17	1
37	121	41	5	412	110	2	54	10	1

320 rows × 10 columns

In [24]:
```
y_train
```

Out[24]:

	Sales
93	8
23	5
299	9
13	10
90	5
...	...
255	7
72	5
396	6
235	5
37	4

320 rows × 1 columns

In [25]:
```
# For training data
X_train.shape,y_train.shape
```

Out[25]:
```
((320, 10), (320, 1))
```

In [26]:
```
# For test data
X_test.shape,y_test.shape
```

Out[26]:
```
((80, 10), (80, 1))
```

6. Model training

In [27]:
```
import warnings
warnings.filterwarnings('ignore')
```

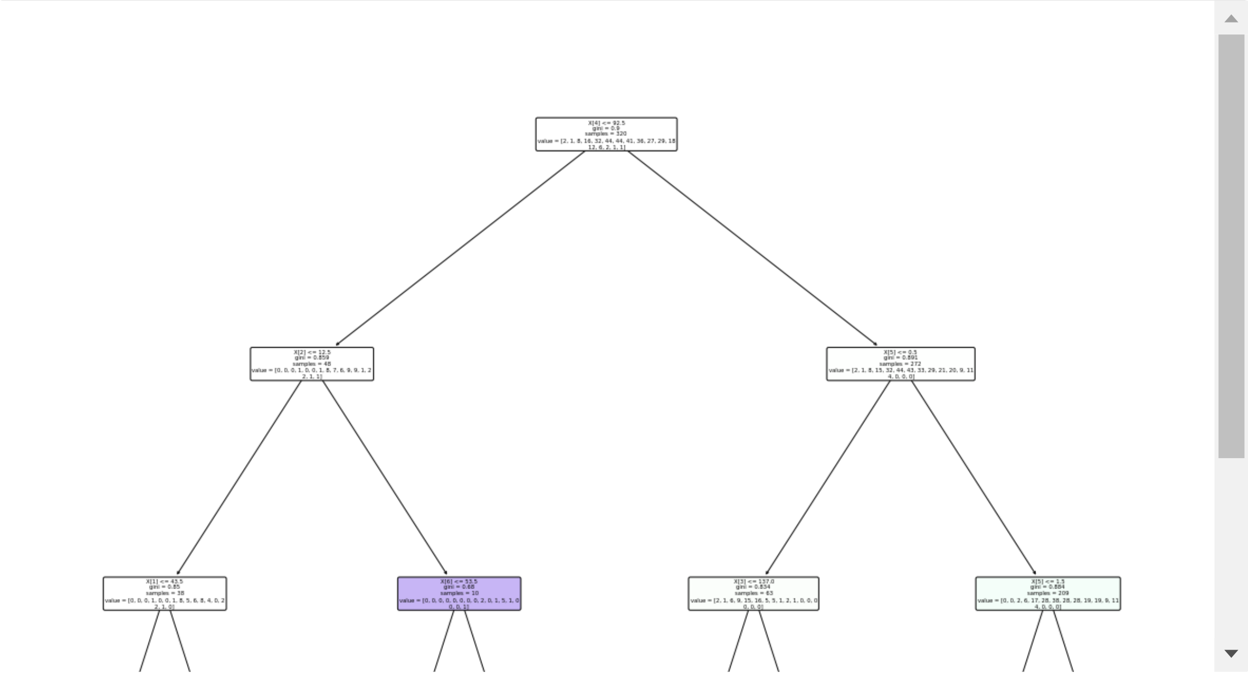
In [67]:
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(criterion='gini', max_depth=3)
dt_model.fit(X_train,y_train)
```

Out[67]:
```
DecisionTreeClassifier(max_depth=3)
```

Plot tree

In [68]:
```
#Prepare a plot figure with set size
from sklearn.tree import plot_tree
from matplotlib import pyplot as plt
```

In [69]:
```
plt.figure(figsize=(20,16))
plot_tree(dt_model,rounded=True,filled=True)
plt.show()
```



7. Model Testing

In [70]:
```
# For training data
y_train_pred = dt_model.predict(X_train)
```

In [71]:
```
#For Testing data
y_test_pred = dt_model.predict(X_test)
```

8 . Model Evaluation

In [72]:
```
from sklearn.metrics import accuracy_score,precision_score,confusion_matrix,recal
```

For training data

In [73]:
```
accuracy_score(y_train,y_train_pred)
```

Out[73]:
```
0.246875
```

In [74]:
```
print(confusion_matrix(y_train,y_train_pred))
```

```
[[ 0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  3  3  2  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  5  0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0 14  1 16  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0 10  6 28  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  4  1 32  1  0  0  6  0  0  0  0  0  0]
 [ 0  0  0  0  4  1 21  7  0  0  8  0  0  0  0  0  0]
 [ 0  0  0  0  1  0 22  5  2  0  6  0  0  0  0  0  0]
 [ 0  0  0  0  2  0 11  6  0  0  8  0  0  0  0  0  0]
 [ 0  0  0  0  1  0 10  4  1  0 13  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  4  0  0  7  5  0  0  0  0  0]
 [ 0  0  0  0  0  0  3  0  1  0  8  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  4  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0]]
```

In [75]:
```
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         8
           3       0.00      0.00      0.00        16
           4       0.29      0.44      0.35        32
           5       0.43      0.14      0.21        44
           6       0.21      0.73      0.33        44
           7       0.23      0.17      0.20        41
           8       0.40      0.06      0.10        36
           9       0.00      0.00      0.00        27
          10       0.20      0.45      0.28        29
          11       1.00      0.28      0.43        18
          12       0.00      0.00      0.00        12
          13       0.00      0.00      0.00         6
          14       0.00      0.00      0.00         2
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         1

    accuracy                           0.25       320
   macro avg       0.16      0.13      0.11       320
weighted avg       0.27      0.25      0.19       320
```

For Testing Data

In [76]:
```
accuracy_score(y_test,y_test_pred)
```

Out[76]: 0.0875

In [77]:
```
print(confusion_matrix(y_test,y_test_pred))
```

```
[[0 0 0 0 0 1 2 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 1 0 0 0 1 0 0 0]
 [0 0 0 0 2 0 7 0 0 0 0 0 0 0]
 [0 0 0 0 2 1 6 0 0 0 0 0 0 0]
 [0 0 0 0 2 0 4 1 0 0 2 1 0 0]
 [0 0 0 0 2 0 6 0 0 0 3 0 0 0]
 [0 0 0 0 1 0 8 0 0 0 3 2 0 0]
 [0 0 0 0 0 0 4 1 0 0 3 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 1 0 0 3 0 0 0]
 [0 0 0 0 0 0 1 1 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0]]
```

In [78]:
```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         2
           3       0.00      0.00      0.00         3
           4       0.20      0.22      0.21         9
           5       0.25      0.11      0.15         9
           6       0.10      0.40      0.16        10
           7       0.00      0.00      0.00        11
           8       0.00      0.00      0.00        14
           9       0.00      0.00      0.00         8
          10       0.00      0.00      0.00         1
          11       0.00      0.00      0.00         4
          12       0.00      0.00      0.00         4
          13       0.00      0.00      0.00         1

    accuracy                           0.09        80
   macro avg       0.04      0.05      0.04        80
```

9. Model Deployment

In [83]:
```
from pickle import dump
```

In [84]:
```
dump(dt_model,open('logf_model.pkl','wb'))
```

In [85]:
```
from pickle import load
```

In [86]:
```
dt_model_pickle = load(open('logf_model.pkl','rb'))
```

In [87]:
```
pickle_pred = dt_model_pickle.predict(X_test)
```

Conclusion:-We can see maximum depth of tree 3 is good as accuracy prospective & classsification is good technique for predict the sale & regression is not usual to good at this dataset

In [ ]:

```
```