Problem Statement:

A cloth manufacturing company is interested to know about the segment or attributes causes high sale. Approach - A decision tree can be built with target variable Sale (we will first convert it in categorical variable) & all other variable will be independent in the analysis.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

+ Code + Text

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Company_Data.csv")
```

```
df.head()
```

	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

```
df.describe()
```

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.900000
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.620528
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.000000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.000000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.000000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.000000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.000000

```
df.info()
```

```
<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
```

```
df.isnull().sum()
```

```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

```
df.isnull().count()
```

```
Sales          400
CompPrice      400
Income         400
Advertising    400
Population     400
Price          400
ShelveLoc      400
Age            400
Education      400
Urban          400
US             400
dtype: int64
```

```
df['ShelveLoc'].unique()
```

```
array(['Bad', 'Good', 'Medium'], dtype=object)
```

```
df['ShelveLoc'].value_counts()
```

```
Medium    219
Bad        96
Good       85
Name: ShelveLoc, dtype: int64
```

```
df['US'].unique()
```

```
array(['Yes', 'No'], dtype=object)
```

Convert categorical variables into numerical value

```
from sklearn.preprocessing import LabelEncoder
```

```
ls = LabelEncoder()

df['Urban'] = ls.fit_transform(df['Urban'])
df['US'] = ls.fit_transform(df['US'])
df['ShelveLoc'] = ls.fit_transform(df['ShelveLoc'])
```

```
df.drop(columns='US')
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban
0	9.50	138	73	11	276	120	0	42	17	1
1	11.22	111	48	16	260	83	1	65	10	1
2	10.06	113	35	10	269	80	2	59	12	1
3	7.40	117	100	4	466	97	2	55	14	1
4	4.15	141	64	3	340	128	0	38	13	1

Split The Data

397	7.41	162	26	12	368	159	2	40	18	1

```
X = df[['Sales', 'CompPrice', 'Income','Population', 'Price', 'Urban']]
y = df['ShelveLoc']
```

399	9.71	134	37	0	27	120	1	49	16	1

```
X
```

	Sales	CompPrice	Income	Population	Price	Urban
0	9.50	138	73	276	120	1
1	11.22	111	48	260	83	1
2	10.06	113	35	269	80	1
3	7.40	117	100	466	97	1
4	4.15	141	64	340	128	1
...	...	...	...	...	...	...
395	12.57	138	108	203	128	1
396	6.14	139	23	37	120	0
397	7.41	162	26	368	159	1
398	5.94	100	79	284	95	1
399	9.71	134	37	27	120	1

400 rows × 6 columns

```
from sklearn.model_selection import train_test_split

X_test, X_train, y_test, y_train = train_test_split(X, y, test_size = 0.25, random_state=100)

X_train
```

	Sales	CompPrice	Income	Population	Price	Urban
248	5.36	111	52	12	101	1
318	10.08	116	72	456	130	0
173	6.38	135	91	207	128	1
285	7.60	146	26	261	131	1
184	9.95	132	33	35	97	0
...	...	...	...	...	...	...
380	9.64	106	64	17	89	1
311	6.15	146	68	328	132	1
110	9.00	128	62	125	116	1
90	5.33	115	22	491	103	0
6	6.63	115	105	45	108	1

100 rows × 6 columns

Train the model

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
from sklearn.metrics import accuracy_score
predict_test = clf.predict(X_test)

accuracy_test = accuracy_score(y_test, predict_test)
accuracy_test
```

```
0.5533333333333333
```

Double-click (or enter) to edit

Checking the accuracy of Training Model

```
from sklearn.metrics import accuracy_score
predict_train = clf.predict(X_train)

accuracy_train = accuracy_score(y_train, predict_train)
accuracy_train
```

```
1.0
```

training dataset accuracy > test dataset

hence our model is overfitted.

Decision tree

```
from sklearn import tree

plt.figure(figsize=(15,10))
tree.plot_tree(clf, filled = True)

plt.show()
```