Decision Tree

# Assignment

About the data:

Let’s consider a Company dataset with around 10 variables and 400 records.

The attributes are as follows:

 Sales -- Unit sales (in thousands) at each location

 Competitor Price -- Price charged by competitor at each location

 Income -- Community income level (in thousands of dollars)

 Advertising -- Local advertising budget for company at each location (in thousands of dollars)

 Population -- Population size in region (in thousands)

 Price -- Price company charges for car seats at each site

 Shelf Location at stores -- A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site

 Age -- Average age of the local population

 Education -- Education level at each location

 Urban -- A factor with levels No and Yes to indicate whether the store is in an urban or rural location

 US -- A factor with levels No and Yes to indicate whether the store is in the US or not

The company dataset looks like this:



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.

import pandas as pd#import pandas module as pd

import matplotlib.pyplot as plt #import plotting function

**#read a dataset**

comp=pd.read\_csv("C:\\Users\\jeeva\\Downloads\\R assignment\\decision tree\\Company\_Data.csv")

**#change bad=0 medium=1 good=2 in shelveloc column**

def tochange(i):

if i=='Bad':

return 0

if i=='Medium':

return 1

if i=='Good':

return 2

comp['ShelveLoc']=comp['ShelveLoc'].apply(tochange)

**#convert salesRanges values to some unique values**

def toconv(x):

if x=='[0-1.0]':

return 0

if x=='[1-2.0]':

return 1

if x=='[2-3.0]':

return 2

if x=='[3-4.0]':

return 3

if x=='[4-5.0]':

return 4

if x=='[5-6.0]':

return 5

if x=='[6-7.0]':

return 6

if x=='[7-8.0]':

return 7

if x=='[8-9.0]':

return 8

if x=='[9-10.0]':

return 9

if x=='[10-11.0]':

return 10

if x=='[11-12.0]':

return 11

if x=='[12-13.0]':

return 12

if x=='[13-14.0]':

return 13

if x=='[14-15.0]':

return 14

if x=='[15-16.0]':

return 15

if x=='[16-17.0]':

return 16

**#Change categorical to numeical**

comp.Urban[comp.Urban=='Yes']=1

comp.Urban[comp.Urban=='No']=0

comp.US[comp.US=='Yes']=1

comp.US[comp.US=='No']=0

comp.Sales.max()

**#they give sales ranges**

Sales\_ranges=["[{0}-{1}]".format(Sales,Sales+1.0)for Sales in range(0,17,1)]

Sales\_ranges#we identify sales ranges

count\_Sales\_ranges=len(Sales\_ranges)#count sales ranges

**#count the sales ranges values ex:[9-10.0]=53 counts [16-17.0]=2counts**

comp['Salesranges\_c']=pd.cut(x=comp['Sales'],bins=count\_Sales\_ranges,labels=Sales\_ranges)

Sales\_len=comp['Salesranges\_c'].value\_counts()

comp\_range=pd.DataFrame(Sales\_len).reset\_index()

**#create a column with ranges and count values of sales\_ranges**

comp\_range.columns=['Salesranges\_c','count']

comp\_range

**#plotting sales ranges values**

plt.bar(comp\_range['Salesranges\_c'],comp\_range['count'])

plt.show()

**#create column to change particular ranges to numerical for identifying**

comp['Sales\_out']=comp['Salesranges\_c'].apply(toconv)#call toconv function

**#create column change numerical data to categorical data**

comp['sale']=comp['Sales\_out'].replace([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16],['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q'])

**#########Converted to Categorical datas#####**

import numpy as np#for mathematical operation

from sklearn.model\_selection import train\_test\_split #split a train and test data

**#inputs are first and output as last in different dataframe**

comp1=comp[['CompPrice','Income','Advertising','Population','Price','ShelveLoc','Age','Education','Urban','US','sale']]

colnames=list(comp1.columns)

pred=colnames[0:10] #predict a input columns

targ=colnames[10] #output as target

**#split the data**

train,test = train\_test\_split(comp1,test\_size = 0.2)

**#import decision tree**

from sklearn.tree import DecisionTreeClassifier

**#model prepared by decision tree classifier**

model = DecisionTreeClassifier(criterion = 'entropy')

model.fit(train[pred],train[targ]) #fitt values to input to output

preds = model.predict(test[pred]) #predict a model with test input

pd.Series(preds).value\_counts() #count predictions

pd.crosstab(test[targ],preds)#cross tab for mismatch function

np.mean(preds==test.sale) # 0.075 for gini: 0.1625 for entropy

#I 13 #G 13 #J 11

#E 8 #L 8 #F 6

#K 4 #N 3 #A 3

#C 3 #H 2 #M 2

#D 2 #B 2