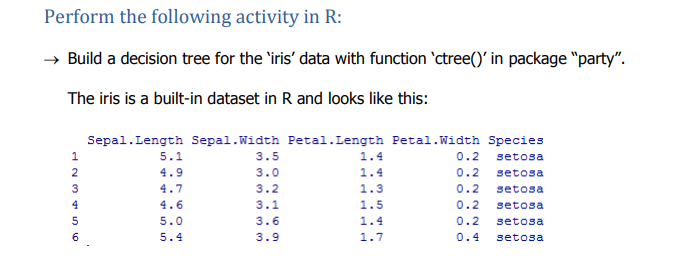
1)ctree function with iris data:



Code:

library(party)

str(iris)

head(iris)

set.seed(1234)

ind <- sample(2,nrow(iris), replace=TRUE, prob=c(0.7,0.3))

trainData <- iris[ind==1,]

testData <- iris[ind==2,]

myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width

iris\_ctree <- ctree(myFormula, data=trainData)

train\_predict <- predict(iris\_ctree,trainData,type="response")

table(train\_predict,trainData$Species)

mean(train\_predict != trainData$Species) \* 100

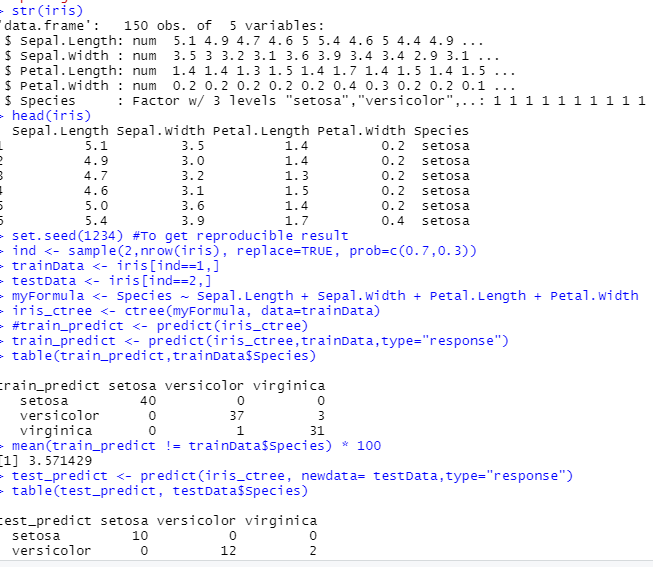
test\_predict <- predict(iris\_ctree, newdata= testData,type="response")

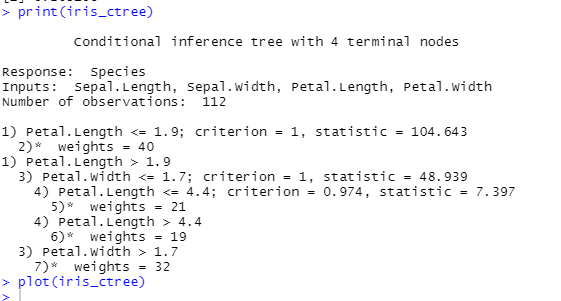
table(test\_predict, testData$Species)

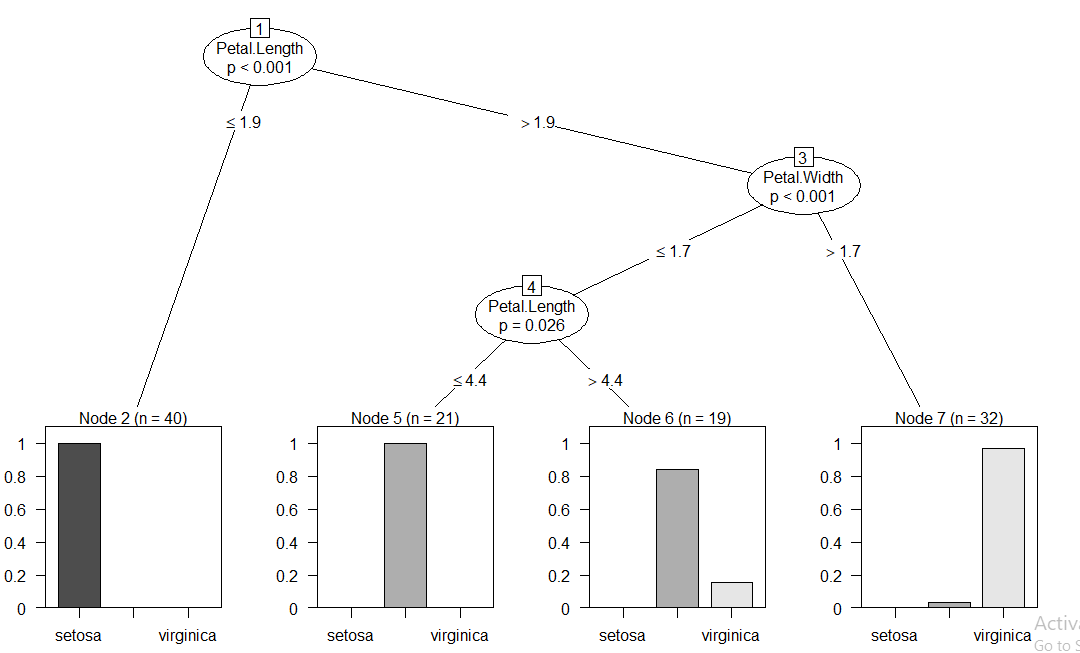
mean(test\_predict != testData$Species) \* 100

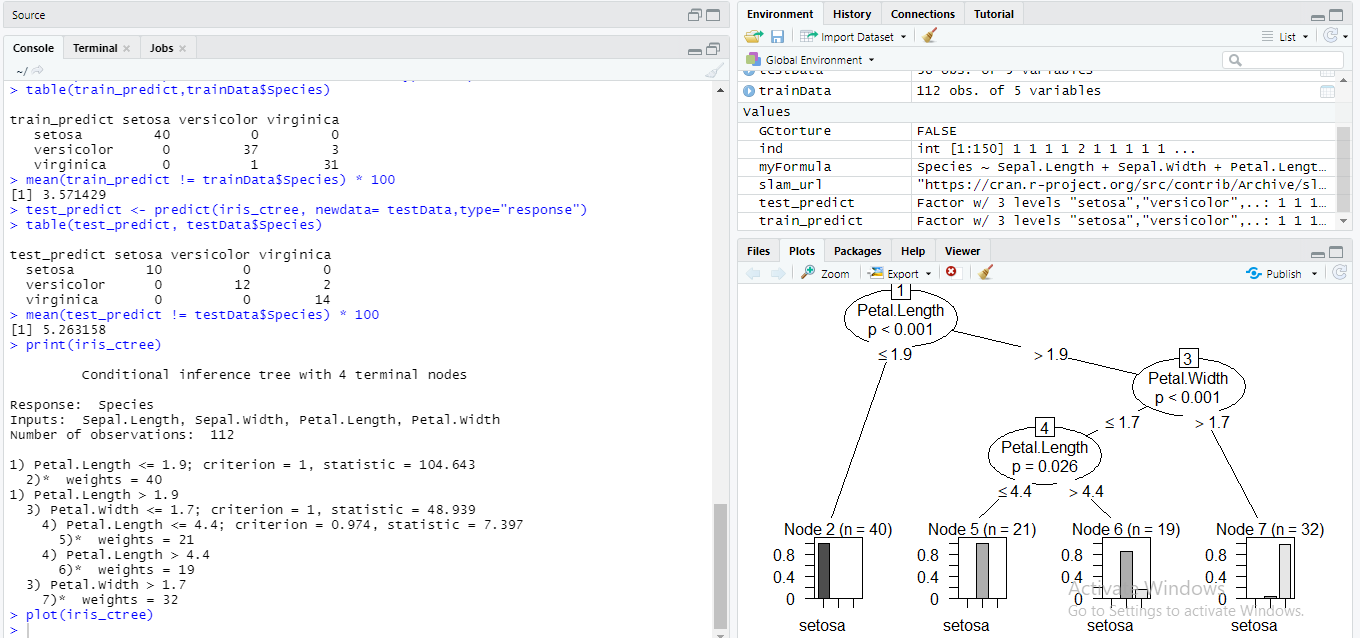
print(iris\_ctree)

plot(iris\_ctree)









2) 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.

In this model we get that Train accuracy =1 & Test acuuracy =0.825

Company\_data.csv:

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 9 19:49:08 2020

@author: sunil

"""

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import preprocessing

data = pd.read\_csv("Company\_Data.csv")

string\_columns=['ShelveLoc','Urban']

number = preprocessing.LabelEncoder()

for i in string\_columns:

data[i] = number.fit\_transform(data[i])

data['US'].unique()

data.US.value\_counts()

colnames = list(data.columns)

predictors = colnames[:6]

target = colnames[6]

import numpy as np

from sklearn.model\_selection import train\_test\_split

train,test = train\_test\_split(data,test\_size = 0.2,random\_state=0)

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(criterion = 'entropy')

model.fit(train[predictors],train[target])

preds = model.predict(test[predictors])

type(preds)

pd.Series(preds).value\_counts()

pd.crosstab(test[target],preds)

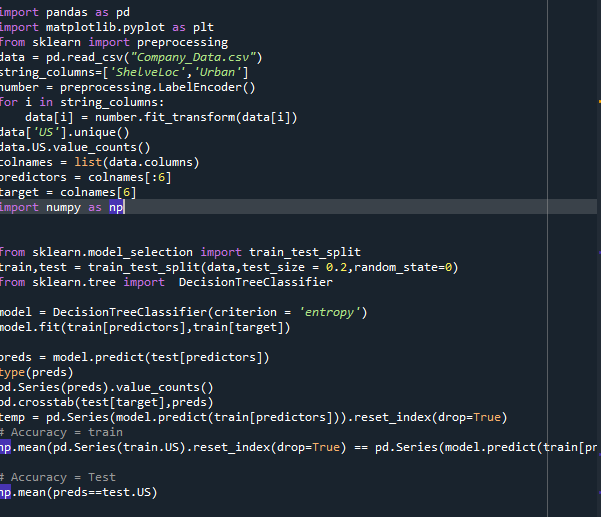
temp = pd.Series(model.predict(train[predictors])).reset\_index(drop=True)

# Accuracy = train

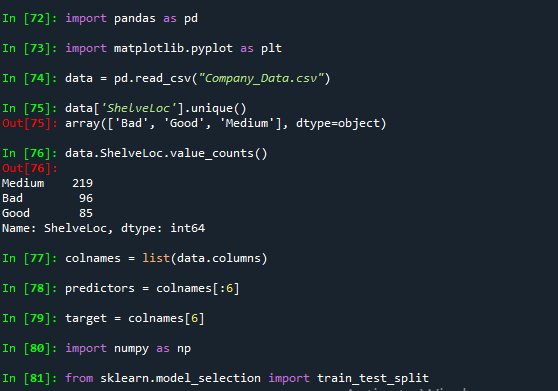
np.mean(pd.Series(train.US).reset\_index(drop=True) == pd.Series(model.predict(train[predictors])))

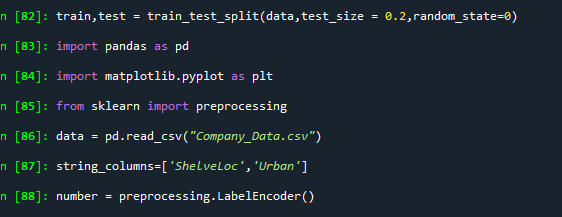
# Accuracy = Test

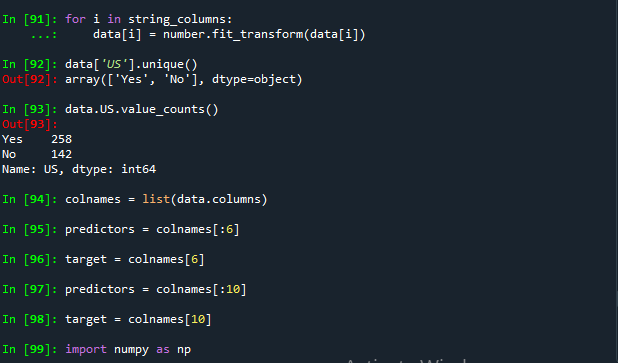
np.mean(preds==test.US

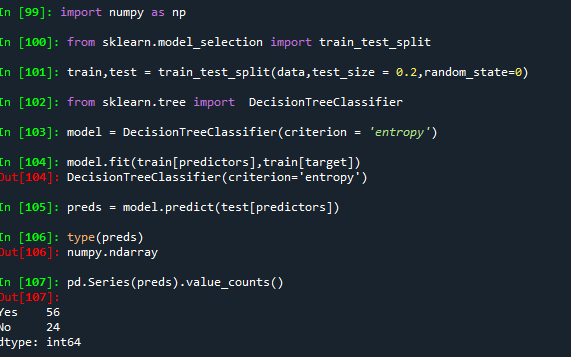


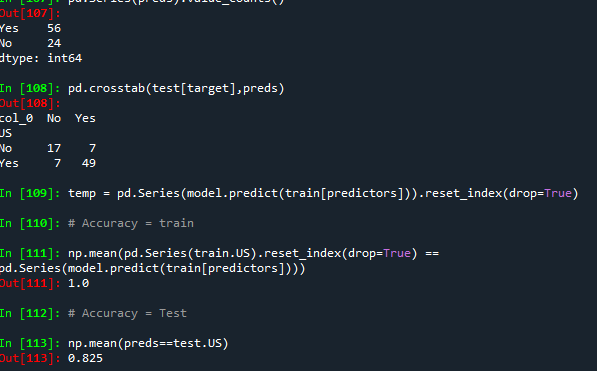
Output:











3) Use decision trees to prepare a model on fraud data

treating those who have taxable\_income <= 30000 as "Risky" and others are "Good"

Data Description :

Undergrad : person is under graduated or not

Marital.Status : marital status of a person

Taxable.Income : Taxable income is the amount of how much tax an individual owes to the government

Work Experience : Work experience of an individual person

Urban : Whether that person belongs to urban area or not

In this model we get that Train accuracy =1 & Test acuuracy =0.825

Code:

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 9 20:51:40 2020

@author: sunil

"""

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import preprocessing

data = pd.read\_csv("Fraud\_check.csv")

string\_columns=['Undergrad','Marital.Status']

number = preprocessing.LabelEncoder()

for i in string\_columns:

data[i] = number.fit\_transform(data[i])

data['Urban'].unique()

data.Urban.value\_counts()

colnames = list(data.columns)

predictors = colnames[:5]

target = colnames[5]

import numpy as np

from sklearn.model\_selection import train\_test\_split

train,test = train\_test\_split(data,test\_size = 0.2,random\_state=0)

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(criterion = 'entropy')

model.fit(train[predictors],train[target])

preds = model.predict(test[predictors])

type(preds)

pd.Series(preds).value\_counts()

pd.crosstab(test[target],preds)

temp = pd.Series(model.predict(train[predictors])).reset\_index(drop=True)

# Accuracy = train

np.mean(pd.Series(train.Urban).reset\_index(drop=True) == pd.Series(model.predict(train[predictors])))

# Accuracy = Test

np.mean(preds==test.Urban)

