Use decision trees to prepare a model on fraud data

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

library(C50)

library(caret)

library(gmodels)

library(dplyr)

##import the data in r##

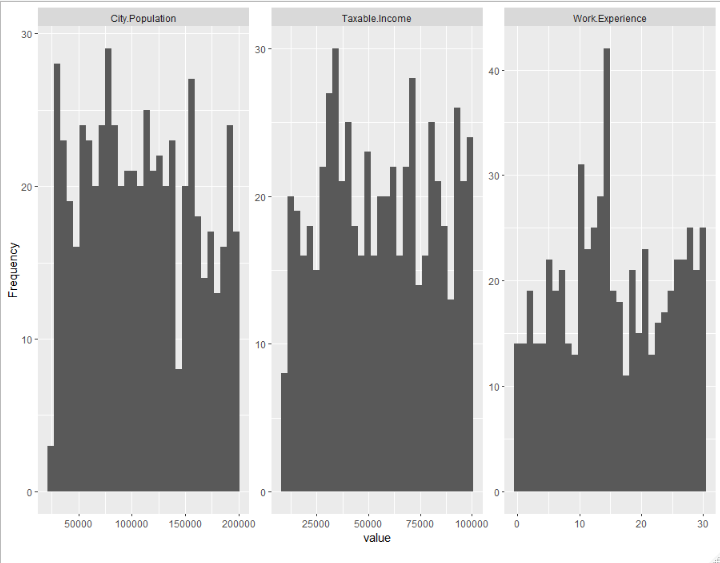
Data <- read.csv(file.choose())

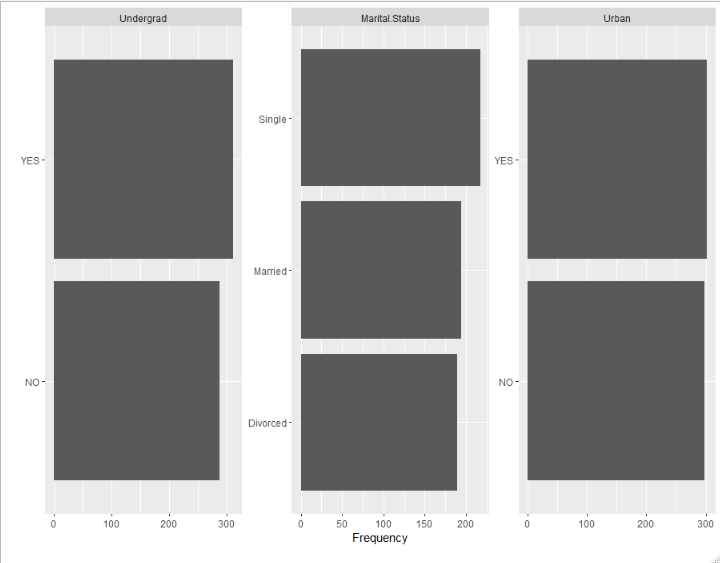
View(Data)

str(Data)

DataExplorer::plot\_histogram(Data)

##there is no distributions in the variable





##The factor variable are evenly distributed

## making the taxable income to categorical variable##

Fraud<- if\_else(Data$Taxable.Income<=30000,'Risky','Good')

Data <- data.frame(Data,Fraud)

##splitting the data into two##

set.seed(345)

train <- sample(1:600,450)

train\_data <- Data[train,-3]

test\_data <- Data[-train,-3]

##Model building ##

model <- C5.0(train\_data[-6],train\_data$Fraud)

model

plot(model)

##prediction of the model on the test data##

pre <- predict(model,test\_data)

pre

##evaluation of the model##

mean(pre==test\_data$Fraud)

confusionMatrix(pre,test\_data$Fraud)##82.67accuracy

## in this model we have the values of good more then the risky so we have gone for over sampling technique

library(ROSE)

over <- ovun.sample(Fraud~.,data = Data,method = "over",N=952)$data

over$Fraud

(table(over$Fraud))

##splitting the data into two##

set.seed(345)

train1 <- sample(1:952,500)

train\_data1 <- Data[train1,-3]

test\_data1 <- Data[-train1,-3]

##Model building ##

model1 <- C5.0(train\_data1[-6],train\_data1$Fraud)

model1

plot(model1)

##prediction of the model on the test data##

pre1 <- predict(model1,test\_data1)

pre1

##evaluation of the model##

mean(pre1==test\_data1$Fraud)

confusionMatrix(pre1,test\_data1$Fraud)##.79 accuracy

company Data

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

##importing the data set

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

data.info()## this function is used to known about the variable in the data set

data.head()

##To get the view of the data set

##in this data one float variable 7 int variable and 3 categorical variable are present

## we should covert the independent variable to numerical and dependent variable to categorical

data["US"].replace(['Yes','No'],[1,0],inplace=True)

data["Urban"].replace(['Yes','No'],[1,0],inplace=True)

data['ShelveLoc'].replace(['Bad','Medium','Good'],[0,1,2],inplace=True)

def sales\_to\_cat(x):

if x > 5:

return "High sales"

else :

return "low sales"

data["sales\_categorical"]=data["Sales"].apply(sales\_to\_cat)

data["sales\_categorical"].value\_count()

##We will do some analysis on data set

data.describe()

data.mean()

data.var()

data.std()

data.skew()

data.kurt()

data.hist(figsize=(20,20))

data.corr()

##the main variable effecting the sales is the compprice

data.corrwith

## split the data set feauters and target variable

ind\_var=['CompPrice','Income','Advertising','Population','Price','Age','Education','US','Urban','ShelveLoc']

x=data[ind\_var]

y=data['sales\_categorical']

from sklearn.model\_selection import train\_test\_split # Import train\_test\_split function

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=1) # 70% training and 30% test

from sklearn.model\_selection import train\_test\_split # Import train\_test\_split function

##model buiding##

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer

clf = clf.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

# Model Accuracy

metrics.accuracy\_score(y\_test, y\_pred)

##0.708333

# Create Decision Tree classifer object##model 2

clf = DecisionTreeClassifier(criterion="entropy", max\_depth=3)

# Train Decision Tree Classifer

clf = clf.fit(X\_train,y\_train)

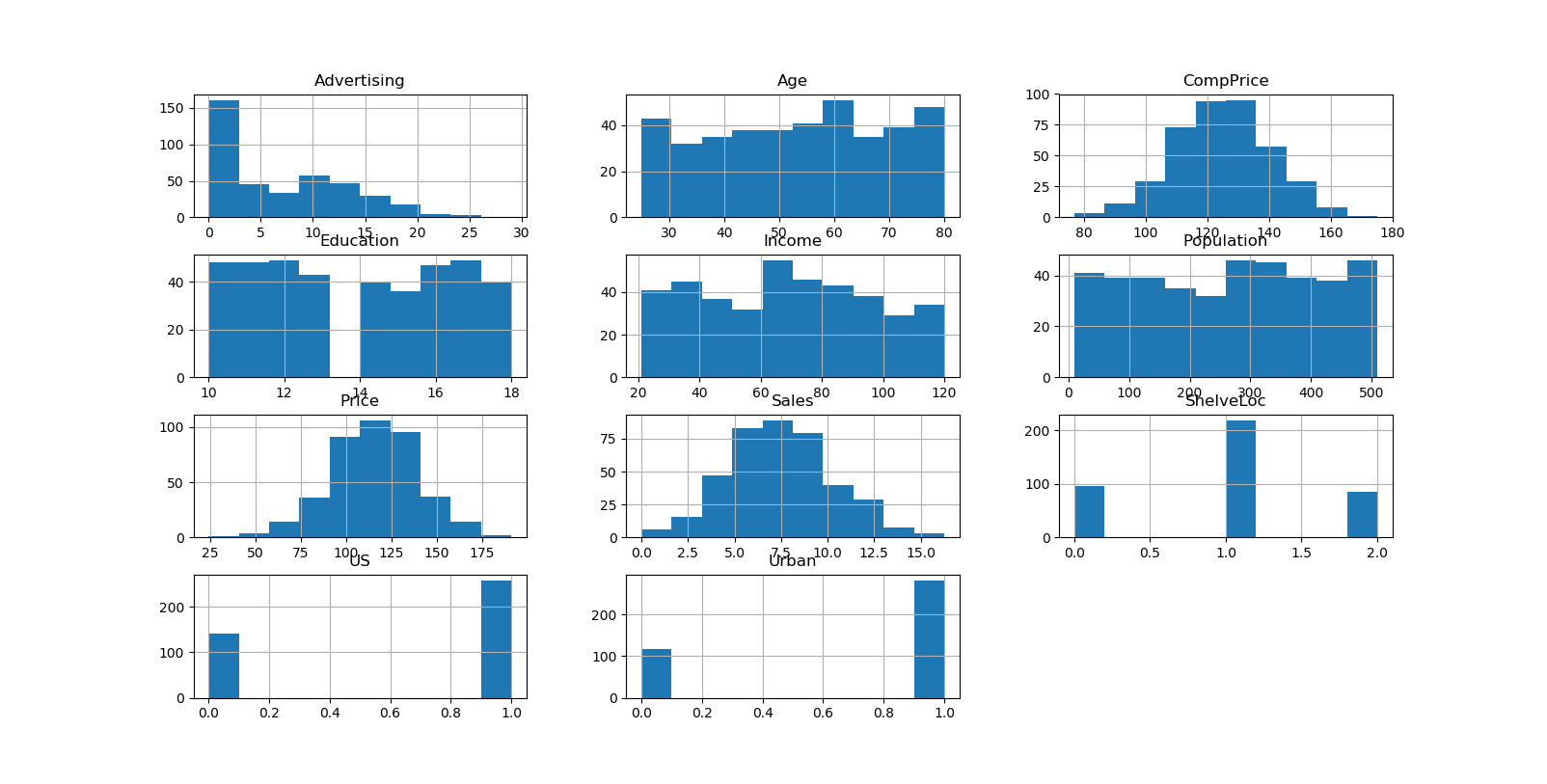
#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

# Model Accuracy, how often is the classifier correct?

metrics.accuracy\_score(y\_test, y\_pred)

Accuracy: 0.825



Iris data set

install.packages("C50") # we neeed to install C50 package to use ak

install.packages("tree")

library(C50)

data()

data("iris")

# Splitting data into training and testing. As the species are in order

# splitting the data based on species

iris\_setosa<-iris[iris$Species=="setosa",] # 50

iris\_versicolor <- iris[iris$Species=="versicolor",] # 50

iris\_virginica <- iris[iris$Species=="virginica",] # 50

iris\_train <- rbind(iris\_setosa[1:25,],iris\_versicolor[1:25,],iris\_virginica[1:25,])

iris\_test <- rbind(iris\_setosa[26:50,],iris\_versicolor[26:50,],iris\_virginica[26:50,])

#### Using tree function

library(tree)

# Building a model on training data

iris\_tree <- tree(Species~.,data=iris\_train)

plot(iris\_tree)

text(iris\_tree,pretty = 0)

# Predicting the test data using the model

pred\_tree <- as.data.frame(predict(iris\_tree,newdata=iris\_test))

pred\_tree["final"] <- NULL

pred\_test\_df <- predict(iris\_tree,newdata=iris\_test)

# for (i in 1:nrow(pred\_tree)){

# pred\_tree[i,"final"]<-ifelse(pred\_tree[i,"setosa"]>0.5,"setosa",ifelse(pred\_tree[i,"versicolor"]>0.5,"versicolor","virginica"))

# }

pred\_tree$final <- colnames(pred\_test\_df)[apply(pred\_test\_df,1,which.max)]

mean(pred\_tree$final==iris\_test$Species) # Accuracy = 94.66%

CrossTable(iris\_test$Species,pred\_tree$final)