using decision trees to prepare a model on fraud data

library(readr)  
library(caret)

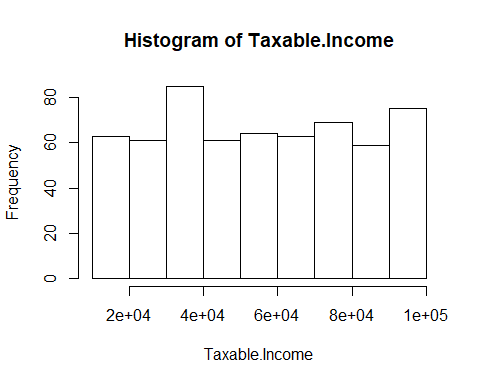
## Loading required package: lattice

## Loading required package: ggplot2

data<-read.csv("file:///E:/assignments data/descition tree/Fraud\_check.csv")  
attach(data)  
str(data)

## 'data.frame': 600 obs. of 6 variables:  
## $ Undergrad : Factor w/ 2 levels "NO","YES": 1 2 1 2 1 1 1 2 1 2 ...  
## $ Marital.Status : Factor w/ 3 levels "Divorced","Married",..: 3 1 2 3 2 1 1 3 3 1 ...  
## $ Taxable.Income : int 68833 33700 36925 50190 81002 33329 83357 62774 83519 98152 ...  
## $ City.Population: int 50047 134075 160205 193264 27533 116382 80890 131253 102481 155482 ...  
## $ Work.Experience: int 10 18 30 15 28 0 8 3 12 4 ...  
## $ Urban : Factor w/ 2 levels "NO","YES": 2 2 2 2 1 1 2 2 2 2 ...

hist(Taxable.Income)



sum(is.na(data))

## [1] 0

data$taxble\_income<-ifelse (Taxable.Income<30000, "Risky", "Good")  
data$taxble\_income<-as.factor(data$taxble\_income)  
fraud\_data<-data[-3]  
head(fraud\_data)

## Undergrad Marital.Status City.Population Work.Experience Urban taxble\_income  
## 1 NO Single 50047 10 YES Good  
## 2 YES Divorced 134075 18 YES Good  
## 3 NO Married 160205 30 YES Good  
## 4 YES Single 193264 15 YES Good  
## 5 NO Married 27533 28 NO Good  
## 6 NO Divorced 116382 0 NO Good

intraininglocal<-createDataPartition(fraud\_data$taxble\_income, p=0.80, list=F)  
train<-fraud\_data[intraininglocal,]  
test<-fraud\_data[-intraininglocal,]  
### decision tree  
library(C50)  
model<-C5.0(train$taxble\_income~., data=train)  
summary(model)

##   
## Call:  
## C5.0.formula(formula = train$taxble\_income ~ ., data = train)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Sat May 30 10:38:56 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 481 cases (6 attributes) from undefined.data  
##   
## Decision tree:  
## Good (481/100)  
##   
##   
## Evaluation on training data (481 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 1 100(20.8%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 381 (a): class Good  
## 100 (b): class Risky  
##   
##   
## Time: 0.0 secs

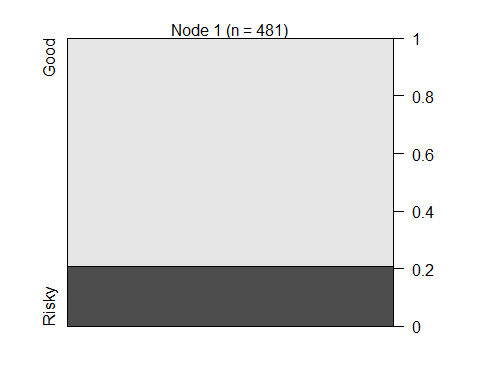
pred<-predict.C5.0(model, test)  
a<-table(test$taxble\_income, pred)  
a

## pred  
## Good Risky  
## Good 95 0  
## Risky 24 0

confusionMatrix(test$taxble\_income, pred)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Good Risky  
## Good 95 0  
## Risky 24 0  
##   
## Accuracy : 0.7983   
## 95% CI : (0.7149, 0.8663)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 2.668e-06   
##   
## Sensitivity : 0.7983   
## Specificity : NA   
## Pos Pred Value : NA   
## Neg Pred Value : NA   
## Prevalence : 1.0000   
## Detection Rate : 0.7983   
## Detection Prevalence : 0.7983   
## Balanced Accuracy : NA   
##   
## 'Positive' Class : Good   
##

plot(model)



##### boosting#####  
model1<-C5.0(train$taxble\_income~., data=train, trails=40)  
summary(model1)

##   
## Call:  
## C5.0.formula(formula = train$taxble\_income ~ ., data = train, trails = 40)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Sat May 30 10:38:56 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 481 cases (6 attributes) from undefined.data  
##   
## Decision tree:  
## Good (481/100)  
##   
##   
## Evaluation on training data (481 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 1 100(20.8%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 381 (a): class Good  
## 100 (b): class Risky  
##   
##   
## Time: 0.0 secs

pred1<-predict.C5.0(model1, test[-6])  
confusionMatrix(test$taxble\_income, pred1)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Good Risky  
## Good 95 0  
## Risky 24 0  
##   
## Accuracy : 0.7983   
## 95% CI : (0.7149, 0.8663)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 2.668e-06   
##   
## Sensitivity : 0.7983   
## Specificity : NA   
## Pos Pred Value : NA   
## Neg Pred Value : NA   
## Prevalence : 1.0000   
## Detection Rate : 0.7983   
## Detection Prevalence : 0.7983   
## Balanced Accuracy : NA   
##   
## 'Positive' Class : Good   
##

acc<-c()  
for(i in 1:100)  
{  
 print(i)  
 inTraininglocal<-createDataPartition(fraud\_data$taxble\_income, p=0.90, list=F)  
 train1<-fraud\_data[inTraininglocal,]  
 test1<-fraud\_data[-inTraininglocal,]  
   
 fit<-C5.0(train1$taxble\_income~.,data=train1)  
 pred2<-predict.C5.0(fit, test1[-6])  
 a<-table(test1$taxble\_income,pred2)  
   
 acc<-c(acc,sum(diag(a))/sum(a))  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10  
## [1] 11  
## [1] 12  
## [1] 13  
## [1] 14  
## [1] 15  
## [1] 16  
## [1] 17  
## [1] 18  
## [1] 19  
## [1] 20  
## [1] 21  
## [1] 22  
## [1] 23  
## [1] 24  
## [1] 25  
## [1] 26  
## [1] 27  
## [1] 28  
## [1] 29  
## [1] 30  
## [1] 31  
## [1] 32  
## [1] 33  
## [1] 34  
## [1] 35  
## [1] 36  
## [1] 37  
## [1] 38  
## [1] 39  
## [1] 40  
## [1] 41  
## [1] 42  
## [1] 43  
## [1] 44  
## [1] 45  
## [1] 46  
## [1] 47  
## [1] 48  
## [1] 49  
## [1] 50  
## [1] 51  
## [1] 52  
## [1] 53  
## [1] 54  
## [1] 55  
## [1] 56  
## [1] 57  
## [1] 58  
## [1] 59  
## [1] 60  
## [1] 61  
## [1] 62  
## [1] 63  
## [1] 64  
## [1] 65  
## [1] 66  
## [1] 67  
## [1] 68  
## [1] 69  
## [1] 70  
## [1] 71  
## [1] 72  
## [1] 73  
## [1] 74  
## [1] 75  
## [1] 76  
## [1] 77  
## [1] 78  
## [1] 79  
## [1] 80  
## [1] 81  
## [1] 82  
## [1] 83  
## [1] 84  
## [1] 85  
## [1] 86  
## [1] 87  
## [1] 88  
## [1] 89  
## [1] 90  
## [1] 91  
## [1] 92  
## [1] 93  
## [1] 94  
## [1] 95  
## [1] 96  
## [1] 97  
## [1] 98  
## [1] 99  
## [1] 100

summary(acc)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.7966 0.7966 0.7966 0.7966 0.7966 0.7966

acc

## [1] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [8] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [15] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [22] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [29] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [36] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [43] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [50] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [57] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [64] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [71] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [78] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [85] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [92] 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102 0.7966102  
## [99] 0.7966102 0.7966102