Use Random Forest to prepare a model on fraud data

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

> library(randomForest)

> library(caret)

> frauddata <- read.csv(file.choose())

> View(frauddata)

> summary(frauddata)

Undergrad Marital.Status Taxable.Income City.Population Work.Experience Urban

NO :288 Divorced:189 Min. :10003 Min. : 25779 Min. : 0.00 NO :298

YES:312 Married :194 1st Qu.:32872 1st Qu.: 66967 1st Qu.: 8.00 YES:302

Single :217 Median :55075 Median :106494 Median :15.00

Mean :55208 Mean :108747 Mean :15.56

3rd Qu.:78612 3rd Qu.:150114 3rd Qu.:24.00

Max. :99619 Max. :199778 Max. :30.00

> str(frauddata)

'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 ...

> attach(frauddata)

The following object is masked from company (pos = 3):

Urban

The following object is masked from company (pos = 4):

Urban

#Converting Taxable.Income into categorical type

> tax\_cat <- ifelse(Taxable.Income<=30000,"risky","good")

> frauddata <- data.frame(tax\_cat,frauddata[,-3])

> table(frauddata$tax\_cat)

good risky

476 124

#Splitting of data to train and test

> set.seed(100)

> cut <- createDataPartition(tax\_cat,p=0.7,list = F)

> train\_f <- frauddata[cut,]

> test\_f <- frauddata[-cut,]

#Model building

#Model based on train data

> forest <- randomForest(tax\_cat~.,data = train\_f,importance=TRUE,mtry=2)

> forest

Call:

randomForest(formula = tax\_cat ~ ., data = train\_f, importance = TRUE, mtry = 2)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 21.85%

Confusion matrix:

good risky class.error

good 329 5 0.01497006

risky 87 0 1.00000000

#Prediction and accuracy based on train data

> predict\_train <- predict(forest,train\_f)

> mean(predict\_train==train\_f$tax\_cat)

[1] 0.9311164

> confusionMatrix(predict\_train,train\_f$tax\_cat)

Confusion Matrix and Statistics

Reference

Prediction good risky

good 334 29

risky 0 58

Accuracy : 0.9311

95% CI : (0.9026, 0.9534)

No Information Rate : 0.7933

P-Value [Acc > NIR] : 4.011e-15

Kappa : 0.7604

Mcnemar's Test P-Value : 1.999e-07

Sensitivity : 1.0000

Specificity : 0.6667

Pos Pred Value : 0.9201

Neg Pred Value : 1.0000

Prevalence : 0.7933

Detection Rate : 0.7933

Detection Prevalence : 0.8622

Balanced Accuracy : 0.8333

'Positive' Class : good

#Prediction and accuracy based on test data

> predict\_test <- predict(forest,test\_f)

> mean(predict\_test==test\_f$tax\_cat)

[1] 0.7877095

> confusionMatrix(predict\_test,test\_f$tax\_cat)

Confusion Matrix and Statistics

Reference

Prediction good risky

good 140 36

risky 2 1

Accuracy : 0.7877

95% CI : (0.7205, 0.8452)

No Information Rate : 0.7933

P-Value [Acc > NIR] : 0.6154

Kappa : 0.0196

Mcnemar's Test P-Value : 8.636e-08

Sensitivity : 0.98592

Specificity : 0.02703

Pos Pred Value : 0.79545

Neg Pred Value : 0.33333

Prevalence : 0.79330

Detection Rate : 0.78212

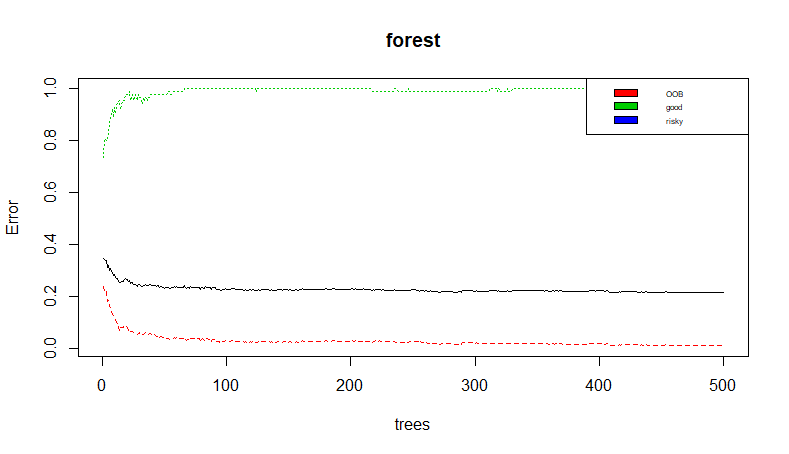
Detection Prevalence : 0.98324

Balanced Accuracy : 0.50647

'Positive' Class : good

> plot(forest)

> legend("topright",col = 2:5,colnames(forest$err.rate),fill = 2:5,cex = 0.5)



#Variable Importance

> importance(forest)

good risky MeanDecreaseAccuracy MeanDecreaseGini

Undergrad 1.1627972 -4.3810085 -0.8720646 5.042825

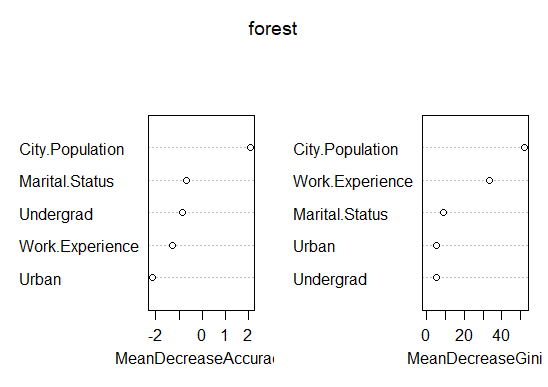
Marital.Status 0.8730290 -3.3739011 -0.6847354 8.818015

City.Population 2.7506298 -0.5286923 2.0807624 52.310102

Work.Experience -0.3418739 -2.1363791 -1.2746731 33.678917

Urban -2.1144501 -0.6071139 -2.1351853 5.177122

> varImpPlot(forest)



#From Observation we can say most significant variable is city population.

#Bagging

> acc <- c()

> i=2

>for(i in 2:10){

set.seed(100)

d <- createDataPartition(tax\_cat,p=0.8,list = F)

train\_d <- frauddata[d,]

test\_d <- frauddata[-d,]

model\_d <- randomForest(tax\_cat~.,data = train\_d,mtry=i)

pred\_b <- predict(model\_d,test\_d)

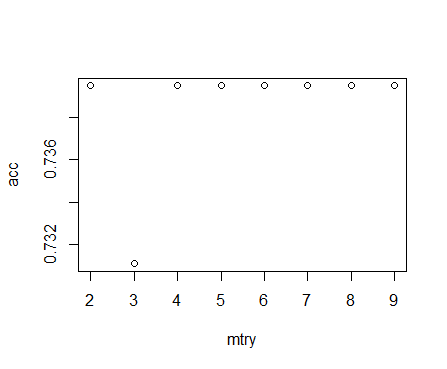
acc[i-2]=mean(pred\_b==test\_d$tax\_cat)

}

> acc

[1] 0.7394958 0.7310924 0.7394958 0.7394958 0.7394958 0.7394958 0.7394958 0.7394958

> plot(2:9,acc,xlab = "mtry",ylab = "acc")



#From above Observations, highest accuracy is for mtry=2

#So Building Our Final Model with , highest accuracy of mtry=2

> finalmodel <- randomForest(tax\_cat~.,data = train\_f,mtry=2)

> finalmodel

Call:

randomForest(formula = tax\_cat ~ ., data = train\_f, mtry = 2)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 21.38%

Confusion matrix:

good risky class.error

good 329 5 0.01497006

risky 85 2 0.97701149

> mean(predict(finalmodel,test\_f)==test\_f$tax\_cat)

[1] 0.7877095

#Our Final Model will have 78.77% Accuracy**.**