Machine Learning - Block 01 Lab 2

Agustín Valencia

12/5/2019

Assignment 2. Analysis of credit scoring

The data file creditscoring.xls contains data retrieved from a database in a private enterprise. Each row contains information about one customer. The variable good/bad indicates how the customers have managed their loans. The other features are potential predictors. Your task is to derive a prediction model that can be used to predict whether or not a new customer is likely to pay back the loan.

1. Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.

```
## Data set size        : 1000 20
```

```
## Training set size    : 500 20
```

```
## Validation set size  : 250 20
```

```
## Testing set size     : 250 20
```

2. Fit a decision tree to the training data by using the following measures of impurity and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.

a. Deviance

```
##
## Classification tree:
## tree(formula = f, data = train, split = "deviance")
## Variables actually used in tree construction:
## [1] "duration" "history"  "age"      "property" "job"      "savings"  "amount"
## Number of terminal nodes:  12
## Residual mean deviance:  0.9879 = 476.2 / 482
## Misclassification error rate: 0.247 = 122 / 494
```

```
## Classification Performance : Tree. split = deviance
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   22   43
##    good  44  141
## Rates details:
##  TPR = 76.63043 % - TNR = 33.33333 % - FPR = 23.36957 % - FNR = 66.66667 %
##  Misclassification Rate =  34.8 %
```

b. Gini index

```
##
## Classification tree:
## tree(formula = f, data = train, split = "gini")
## Variables actually used in tree construction:
##  [1] "foreign"  "coapp"    "depends"  "existcr"  "telephon" "savings"
##  [7] "history"  "property" "employed" "resident" "marital"  "purpose"
## [13] "duration" "housing"  "installp" "amount"   "job"      "age"
## Number of terminal nodes:  70
## Residual mean deviance:  1.059 = 449.1 / 424
## Misclassification error rate: 0.247 = 122 / 494
```

```
## Classification Performance : Tree. split = gini
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   19   46
##    good  42  143
## Rates details:
##  TPR = 75.66138 % - TNR = 31.14754 % - FPR = 24.33862 % - FNR = 68.85246 %
##  Misclassification Rate =  35.2 %
```

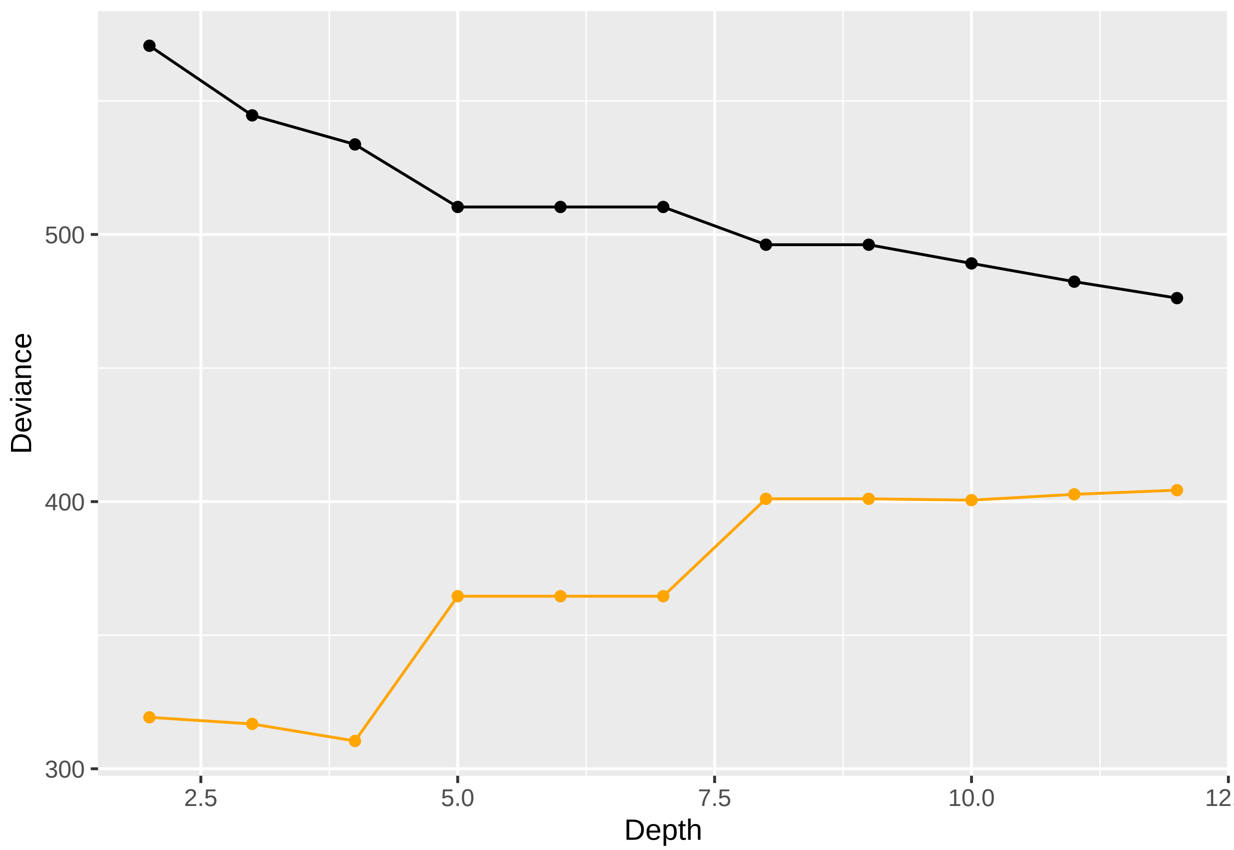
In summary, the tree trained using deviance as split method performs slighty better than the one using Gini index because it gets better True Positive and Misclassification rates, also it is considerably smaller having 12 terminal nodes against the 70 from the Gini one.

3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.

Crossvalidating the deviance-trained tree:

```
## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length
```

```
## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length

## Warning in trees[i] <- prunedTree: number of items to replace is not a multiple
## of replacement length
```




```
## The optimal tree is at depth 4 with deviance 310.4075
```

In the graph are shown the deviances obtained depending on the depths. Black curve corresponds to train scores and orange to validation. The optimal tree while using validation data is at depth = 4 having a deviance of 310.41

4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

```
## Classification Performance : Naive Bayes - Train
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad  103   45
```

```
##    good 111  241
## Rates details:
##  TPR = 84.26573 % - TNR = 48.13084 % - FPR = 15.73427 % - FNR = 51.86916 %
##  Misclassification Rate =  62.4 %
```

```
## Classification Performance : Naive Bayes - test
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   42   23
##    good  71  114
## Rates details:
##  TPR = 83.21168 % - TNR = 37.16814 % - FPR = 16.78832 % - FNR = 62.83186 %
##  Misclassification Rate =  37.6 %
```

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle: $\hat{Y} = 1$ if $p(Y = good|X) > \pi$, otherwise $\hat{Y} = 0$ where $\pi = 0.05, 0.1, 0.15, ..., 0.9, 0.95$. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

```r
thresholds <- seq(.05, .95, .05)
optTree <- trees[optimal]
optTree
```

```
## [[1]]
##        var   n       dev yval splits.cutleft splits.cutright  yprob.bad
## 1 duration 494 598.01210 good          <15.5           >15.5 0.29352227
## 2  history 210 192.42056 good           <2.5            >2.5 0.17142857
## 4   <leaf> 126 138.31634 good                                0.23809524
## 5   <leaf>  84  43.22953 good                                0.07142857
## 3  savings 284 378.22844 good           <2.5            >2.5 0.38380282
## 6   <leaf> 187 259.10334 good                                0.48663102
## 7   <leaf>  97  93.06779 good                                0.18556701
##   yprob.good
## 1 0.70647773
## 2 0.82857143
## 4 0.76190476
## 5 0.92857143
## 3 0.61619718
## 6 0.51336898
## 7 0.81443299
```

6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix.

and report the confusion matrix for the training and test data. Compare the results with the resukts from step 4 and discuss how the rates have changed and why.

Appendix A : Code

```r
#### ----------------------------------------
####                Setup
#### ----------------------------------------
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(tree)
library(ggplot2)
library(e1071)
set.seed(12345)




#### ----------------------------------------
####                Question 1
#### ----------------------------------------


## 1.1 Split data

data <- read_xls("data/creditscoring.xls")
n <- dim(data)[1]
data$good_bad <- as.factor(data$good_bad)

# training set
id <- sample(1:n, floor(n*0.5))
train <- data[id,]

# validation set
id1 <- setdiff(1:n, id)
id2 <- sample(id1, floor(n*0.25))
valid <- data[id2,]

# test set
id3 <- setdiff(id1,id2)
test <- data[id3,]

cat("Data set size \t\t:", dim(data))
cat("Training set size \t:", dim(train))
cat("Validation set size \t:", dim(valid))
cat("Testing set size \t:", dim(test))

## 1.2 Trees

f <- good_bad ~ .

# util function
get_performance <- function(targets, predictions, text) {
    cat("Classification Performance :", text, "\n")
    t <- table(targets, predictions)
    print("Confusion Matrix")
    print(t)
```

```r
    tn <- t[1,1]
    tp <- t[2,2]
    fp <- t[1,2]
    fn <- t[2,1]
    total <- dim(test)[1]
    tpr <- tp/(tp+fp) * 100
    tnr <- tn/(tn+fn) * 100
    fpr <- fp/(tp+fp) * 100
    fnr <- fn/(tn+fn) * 100

    cat("Rates details:\n")
    cat(" TPR =", tpr, "% -")
    cat(" TNR =", tnr, "% -")
    cat(" FPR =", fpr, "% -")
    cat(" FNR =", fnr, "%")
    cat("\n Misclassification Rate = ", (fp+fn)/total * 100, "%\n")
}



### 1.2.a Deviance Tree
devTree <- tree(formula = f, data = train, split = "deviance")
#plot(devTree)
#text(devTree)
summary(devTree)

true <- test$good_bad
predictions <- predict(devTree, newdata = test, type = "class")
get_performance(true, predictions, "Tree. split = deviance")



### 1.2.b Gini Index
giniTree <- tree(formula = f, data = train, split = "gini")
#plot(giniTree)
#text(giniTree)
summary(giniTree)

predictions <- predict(giniTree, newdata = test, type = "class")
get_performance(true, predictions, "Tree. split = gini")

### 1.3 Optimal depth by train/validation
maxDepth <- 12
depth <- 2:maxDepth
trainScore <- rep(0,maxDepth)
testScore <- rep(0,maxDepth)
trees <- vector(length = maxDepth-1)

for (i in depth) {
    prunedTree <- prune.tree(devTree, best=i)
    trees[i] <- prunedTree
    predictions <- predict(prunedTree, newdata=valid, type="tree")
    trainScore[i] <- deviance(prunedTree)
```

```r
    testScore[i] <- deviance(predictions)
}
trainScore <- trainScore[depth]
testScore <- testScore[depth]
df <- data.frame(
    train = trainScore,
    test = testScore,
    depth = depth
)
p <- ggplot(data=df) +
    geom_line(aes(x = depth, y=train), color="black") +
    geom_line(aes(x = depth, y=test), color="orange") +
    geom_point(aes(x = depth, y=train), color="black") +
    geom_point(aes(x = depth, y=test), color="orange") +
    ylab("Deviance") + xlab("Depth")
p

optDev <- min(testScore)
optimal <- depth[which.min(testScore)]
cat("The optimal tree is at depth", optimal, "with deviance", optDev)


### 1.3 Optimal depth by train/validation
nb <- naiveBayes(f, data=train)

trainPred <- predict(nb, newdata=train)
get_performance(train$good_bad, trainPred, "Naive Bayes - Train")
testPred <- predict(nb, newdata=test)
get_performance(test$good_bad, testPred, "Naive Bayes - test")

thresholds <- seq(.05, .95, .05)
optTree <- trees[optimal]
optTree
```