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] "savings"  "duration" "history"  "age"      "purpose"  "amount"   "resident"
## [8] "other"
## Number of terminal nodes:  15
## Residual mean deviance:  0.9569 = 458.3 / 479
## Misclassification error rate: 0.2105 = 104 / 494
```

```
## Classification Performance : Tree. split = deviance
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   24   54
##    good  17  155
## Rates details:
##  TPR = 74.16268 % - TNR = 58.53659 % - FPR = 25.83732 % - FNR = 41.46341 %
##  Misclassification Rate =  28.4 %
```

b. Gini index

```
##
## Classification tree:
## tree(formula = f, data = train, split = "gini")
## Variables actually used in tree construction:
##  [1] "foreign"  "coapp"    "depends"  "telephon" "existcr"  "savings"
##  [7] "history"  "property" "marital"  "duration" "employed" "age"
```

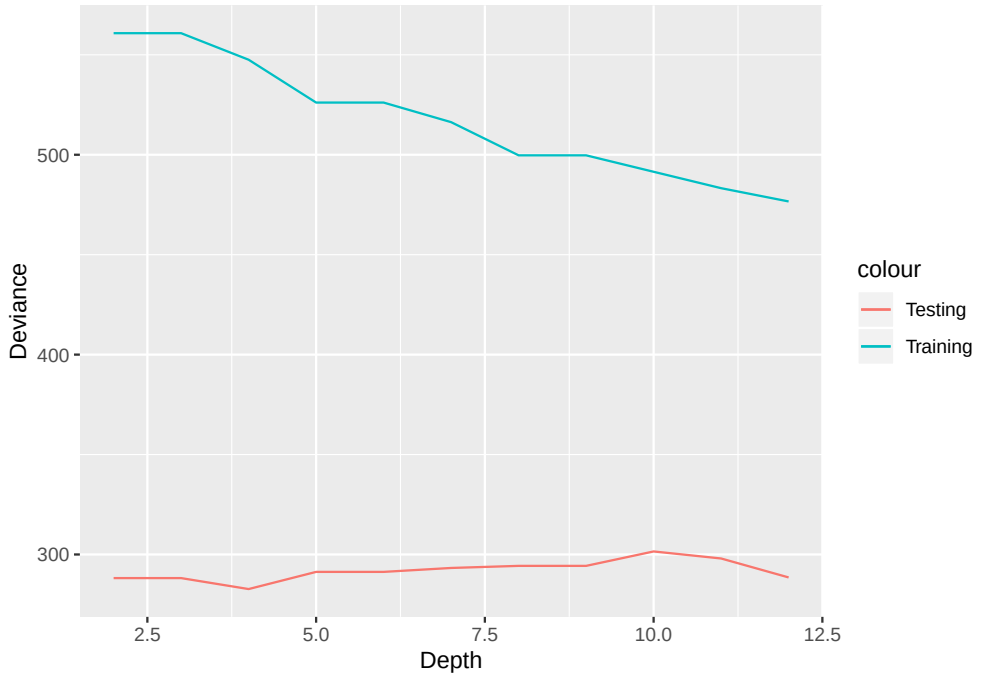
```
## [13] "housing"  "amount"   "purpose"  "resident" "job"      "installp"
## Number of terminal nodes:  72
## Residual mean deviance:  1.015 = 428.5 / 422
## Misclassification error rate: 0.2368 = 117 / 494
```

```
## Classification Performance : Tree. split = gini
## [1] "Confusion Matrix"
##         predictions
## targets bad good
##    bad   22   56
##    good  32  140
## Rates details:
##  TPR = 71.42857 % - TNR = 40.74074 % - FPR = 28.57143 % - FNR = 59.25926 %
##  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:




```
## The optimal tree is at depth 4 with deviance 282.6919
```

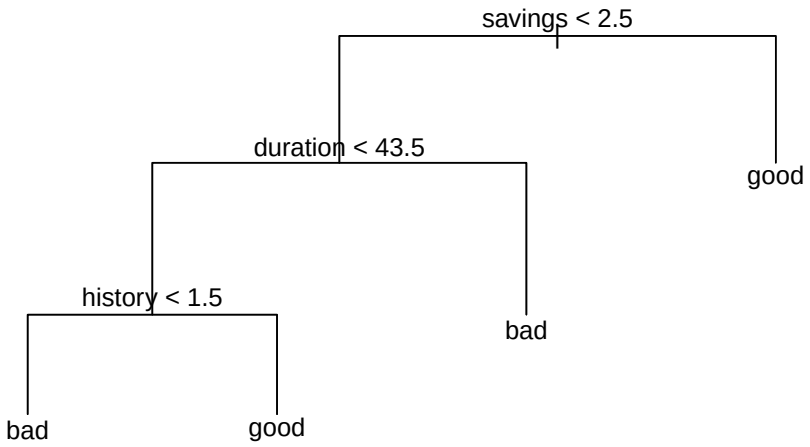
In the graph are shown the deviances obtained depending on the depths. Black curve corresponds to train scores and orange to validation. The optimal tree while using validation data is at depth = 4 having a deviance of 310.41

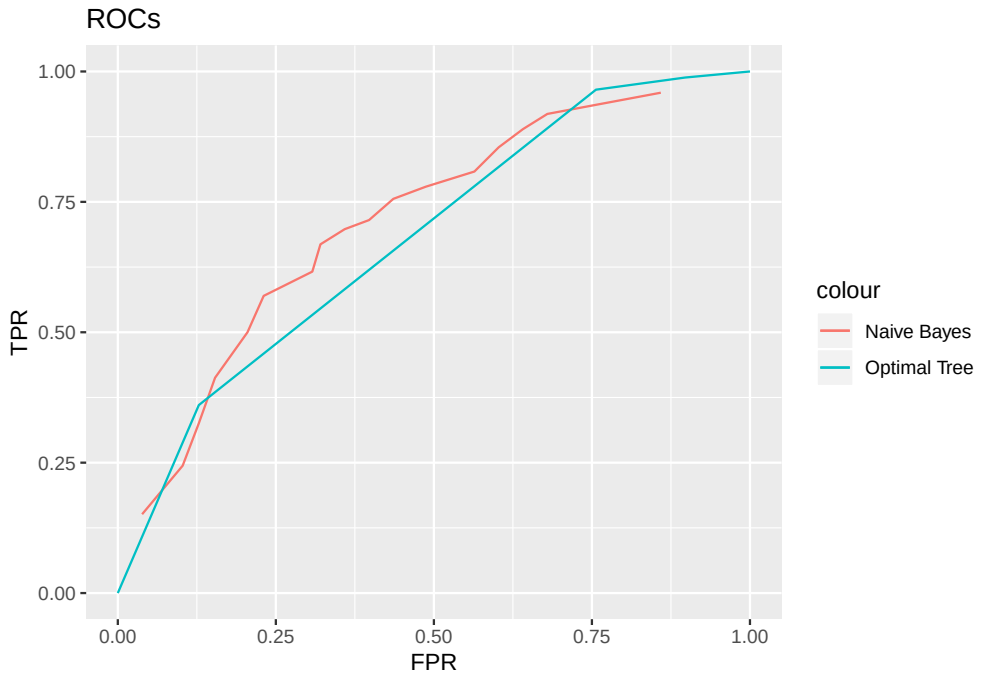
4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

```
## Classification Performance : Naive Bayes - Train
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   95   52
##    good  98  255
## Rates details:
##  TPR = 83.06189 % - TNR = 49.2228 % - FPR = 16.93811 % - FNR = 50.7772 %
##  Misclassification Rate =  60 %

## Classification Performance : Naive Bayes - test
## [1] "Confusion Matrix"
##        predictions
## targets bad good
##    bad   47   31
##    good  49  123
## Rates details:
##  TPR = 79.87013 % - TNR = 48.95833 % - FPR = 20.12987 % - FNR = 51.04167 %
##  Misclassification Rate =  32 %
```

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?





Orange curve is the ROC for the optimal tree and black curve corresponds to naive Bayes classifier.

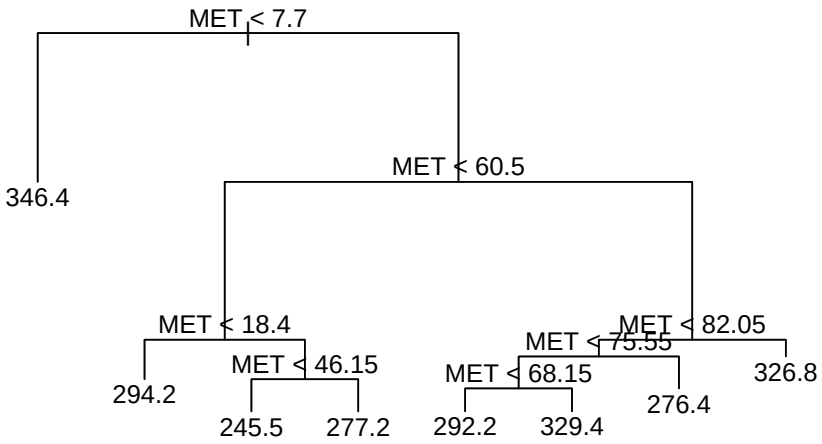
6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix. and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates have changed and why.

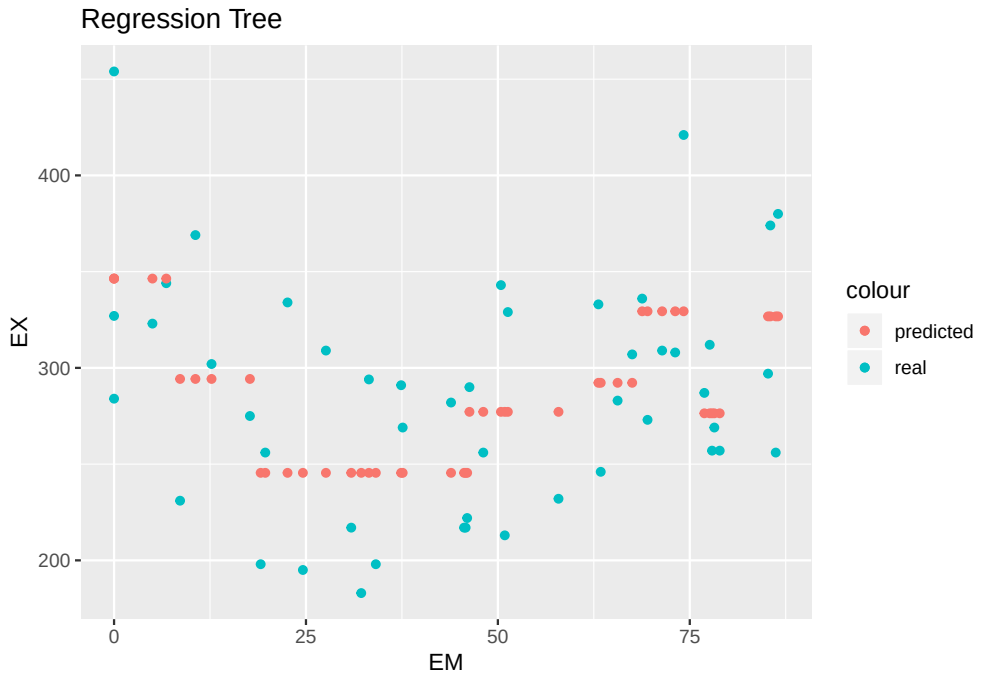
Assignment 3 : Uncertainty estimation

The data file State.csv contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables

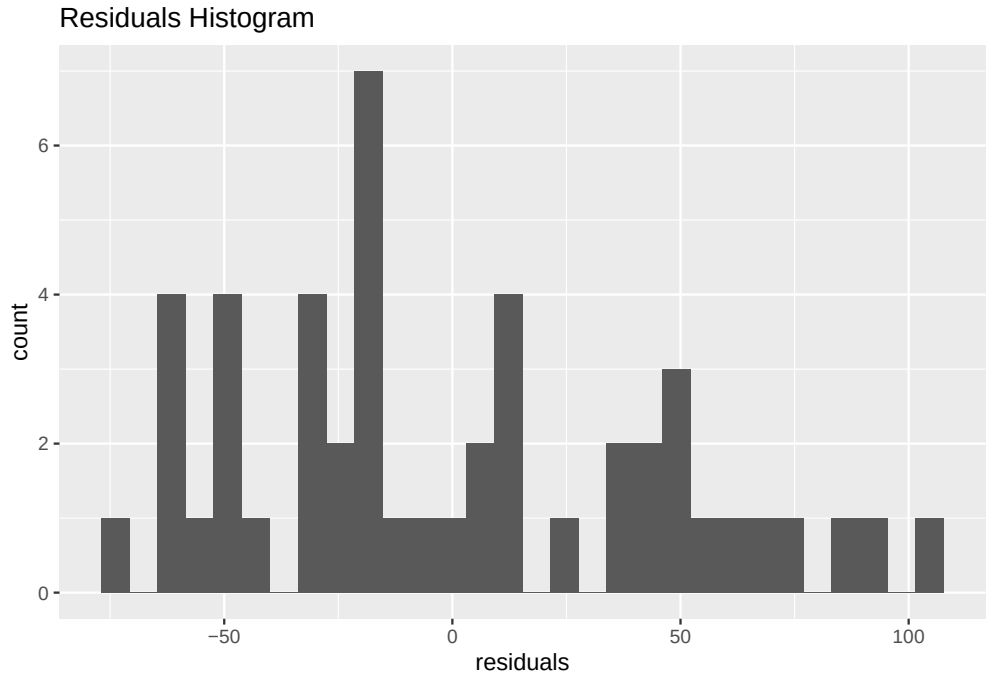
- MET: Percentage of population living in standard metropolitan areas
- EX: Per capita state and local public expenditures ($)

1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2-5.
2. Use package tree and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting minsize in tree.control). Report the selected tree. Plot the original and the fitted data and histogram of residuals. Comment on the distribution of the residuals and the quality of the fit.






```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



3. Compute and plot the 95% confidence bands for the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a non-parametric bootstrap. Comment whether the band is smooth or bumpy and try to explain why. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable.

4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled

data) by using a parametric bootstrap, assume $Y \ N(\mu_i, \sigma^2)$ where μ_i are labels in the tree leaves and σ^2 is the residual variance. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. Does it look like only 5% of data are outside the prediction band? Should it be?

5. Consider the histogram of residuals from step 2 and suggest what kind of bootstrap is actually more appropriate here.

Appendix A : Code

```
#### ----------------------------------------
#####                Setup
#### ----------------------------------------
knitr::opts_chunk$set(echo = TRUE)
RNGversion("3.5.1")
library(readxl)
library(tree)
library(ggplot2)
library(e1071)
set.seed(12345)



#### ----------------------------------------
#####                Question 2
#### ----------------------------------------


## 2.1 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))

## 2.2 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)
```

```
    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")

### 2.3 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)
```

```
    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="Training")) +
    geom_line(aes(x = depth, y=test, color="Testing")) +
    ylab("Deviance") + xlab("Depth")
p

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


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

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

optTree <- prune.tree(devTree, best = optimal)
plot(optTree)
text(optTree)
nbTestPred <- predict(nb, newdata=test, type="raw")
otTestPred <- predict(optTree, newdata=test, type="vector")
nbTestPred <- nbTestPred[,"good"]
otTestPred <- otTestPred[,"good"]
totalNegative <- sum(test$good_bad == "bad")
totalPositive <- sum(test$good_bad == "good")

thresholds <- seq(.05, .95, .05)
nbFP = vector(length = length(thresholds))
nbTP = vector(length = length(thresholds))
otFP = vector(length = length(thresholds))
otTP = vector(length = length(thresholds))

for(i in 1:length(thresholds)) {
    # Naive Bayes
    nbPred <- as.numeric(nbTestPred > thresholds[i])
    nbPredFactor <- factor(x = nbPred, levels = c(0,1), labels=c("good", "bad"))
    nbTable <- table(test$good_bad, nbPredFactor)
    nbFP[i] <- nbTable[1,2]
    nbTP[i] <- nbTable[2,2]

    # Optimal Tree
```

```
    otPred <- as.numeric(otTestPred > thresholds[i])
    otPredFactor <- factor(x = otPred, levels = c(0,1), labels=c("good", "bad"))
    otTable <- table(test$good_bad, otPredFactor)
    otFP[i] <- otTable[1,2]
    otTP[i] <- otTable[2,2]
}

nbFPR <- nbFP/totalNegative
nbTPR <- nbTP/totalPositive
otFPR <- otFP/totalNegative
otTPR <- otTP/totalPositive

p <- ggplot()
p <- p + geom_line(aes(x=nbFPR, y=nbTPR, color="Naive Bayes"))
p <- p + geom_line(aes(x=otFPR, y=otTPR, color="Optimal Tree"))
p <- p + ylab("TPR") + xlab("FPR") + ggtitle("ROCs")
p


data <- read.csv2("data/State.csv")
data <- data[order(data$EX),]
formula <- EX ~ MET
regTree <- tree(formula = formula,
                data = data,
                control = tree.control(nobs = length(data$EX), minsize = 8)
            )

depth <- 2:12
testScore <- vector(length = (length(depth)-1))
for (i in depth) {
    prunedTree <- prune.tree(regTree, best=i)
    predictions <- predict(prunedTree, newdata=data, type="tree")
    testScore[i] <- deviance(predictions)
}
optScore <- min(testScore)
optDepth <- depth[which.min(testScore)-1]
optTree <- tree(formula = formula,
                data = data,
                control = tree.control(nobs = length(data$EX),minsize = 8)
            )
plot(optTree)
text(optTree)
predictions <- predict(optTree, newdata = data)
p <- ggplot()
p <- p + geom_point(aes(x=data$MET, y=data$EX, color="real"))
p <- p + geom_point(aes(x=data$MET, y=predictions, color="predicted"))
p <- p + ylab("EX") + xlab("EM") + ggtitle("Regression Tree")
p

residuals <- data$EX - predictions
h <- ggplot(as.data.frame(residuals), aes(x=residuals)) + geom_histogram()
h <- h + ggtitle("Residuals Histogram")
h
```