Introduction to Machine Learning

Lab 3

Rasmus Holm

2016-11-23

Contents

Assignment 1

1

Figure 1: Carapace length versus rear width.

Figure 1 shows that the crabs could be separated pretty well based on sex by a linear model. With the proper covariance matrices the groups could be seen as Gaussian distributed and the linear discriminant analysis (LDA) would therefore be a reasonable classifier in this case.

2

Below are are resulting discriminant functions

$$\delta_{\text{female}}(x) = -22.4287694 + 8.2486981\text{RW} + -2.1613185\text{CL},$$

$$\delta_{\text{male}}(x) = -12.5634175 + 2.5658514\text{RW} + -0.2138144\text{CL},$$

and the equation for the decision boundary

$$CL = 2.9180154RW + -5.0656387.$$

3

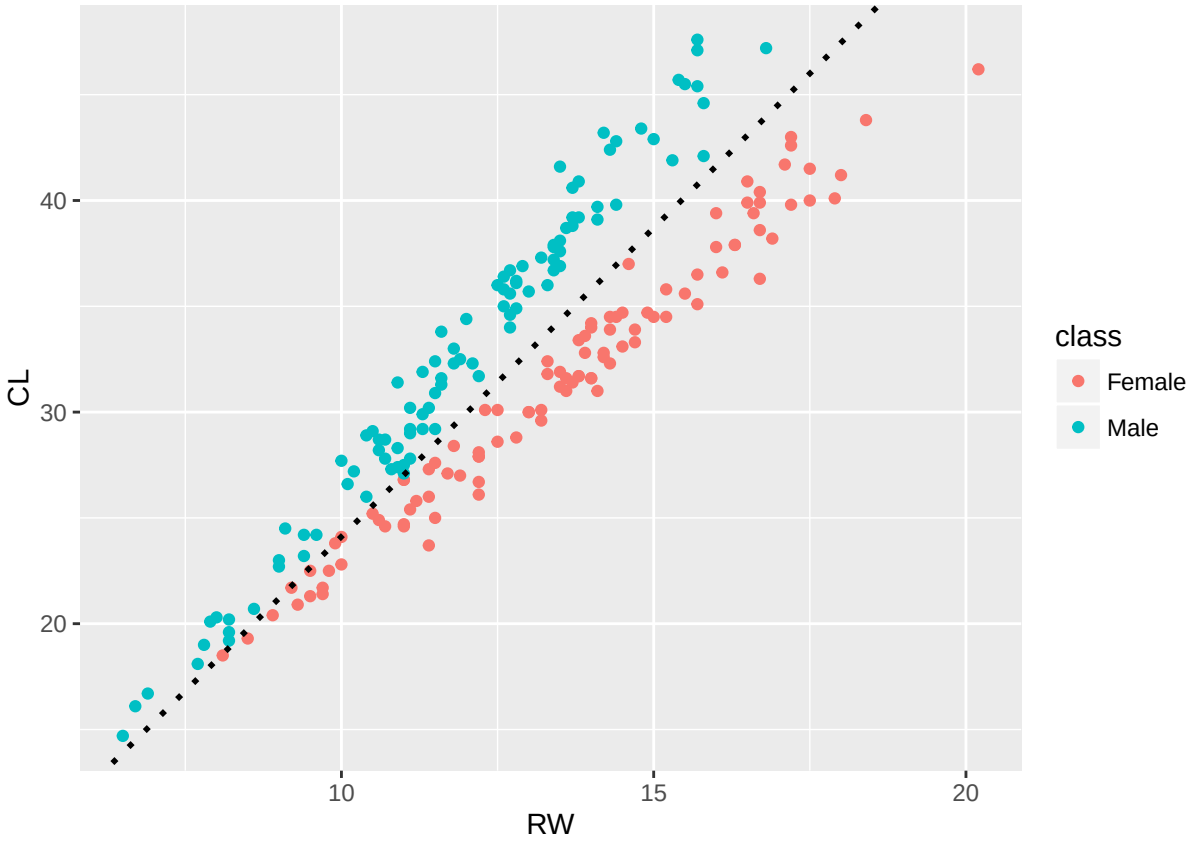

Figure 2: Classification by LDA including the decision boundary.

The LDA classifier does work pretty well in this case as shown in figure 2 with a few misclassifications, 7 out of 200 observations to be precise. This is to be expected since the two classes are not linearly separable

4

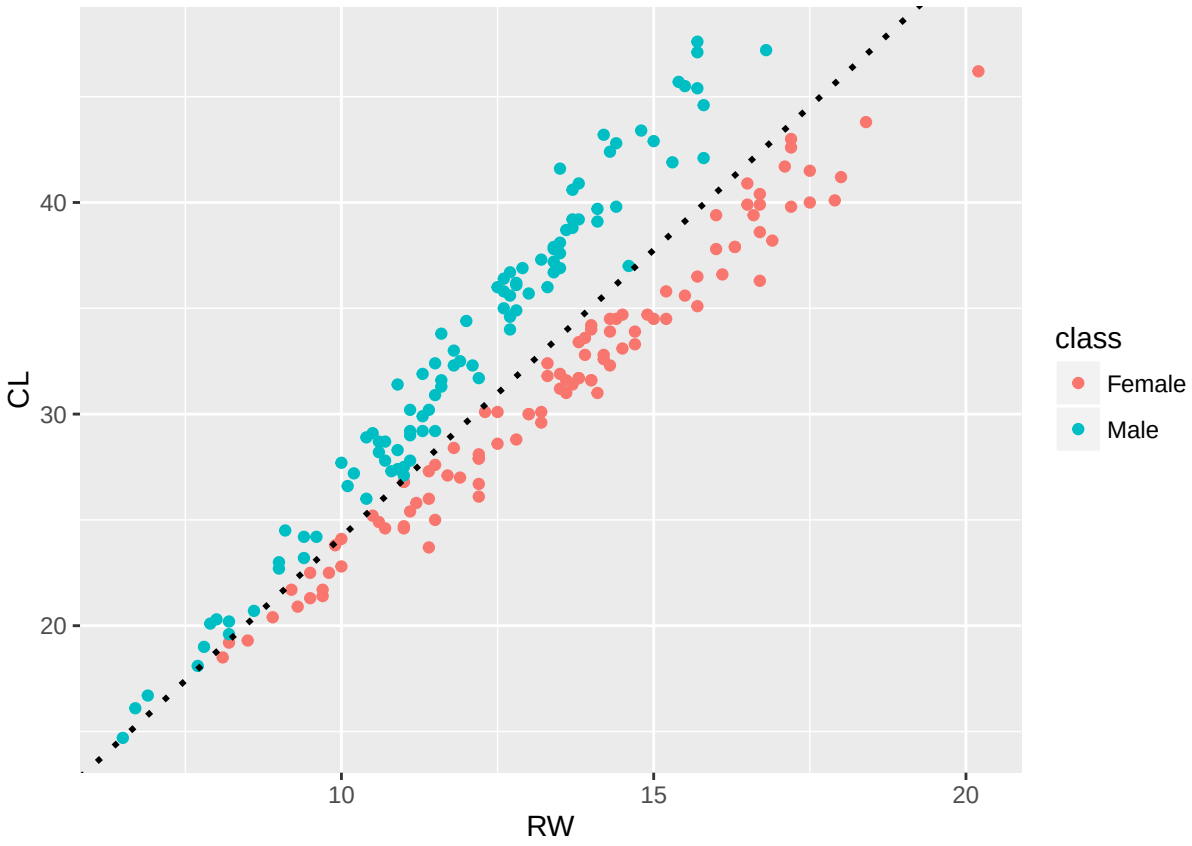

Figure 3: Classification by logistic regression including the decision boundary.

Here I have used logistic regression instead of LDA to classify based on sex. Figure 3 shows a similar result that came from LDA in figure 2. The number of miscalssifications are 7 as well but it is rather difficult to know exactly which one is preferred in this case. Logistic regression does not assume the data to be normally distributed which LDA does and since that assumption is not always true I would probably prefer logistic regression in this case.

Assignment 2

In order to do the following exercises I have divided the creditscoring data set into three sets; train (50%), validation (25%), and test (25%). The variable good/bad indicates how the customers have managed their loans and I will be using decision trees and naive Bayes classifiers to predict the aforementioned variable using all other variables in the data set.

2

Here I am interested in whether the deviance or Gini index as impurity measurement give rise to the lowest misclassification rates in decision trees. I have trained the decision trees on the training data set and then tested on the train and test sets.

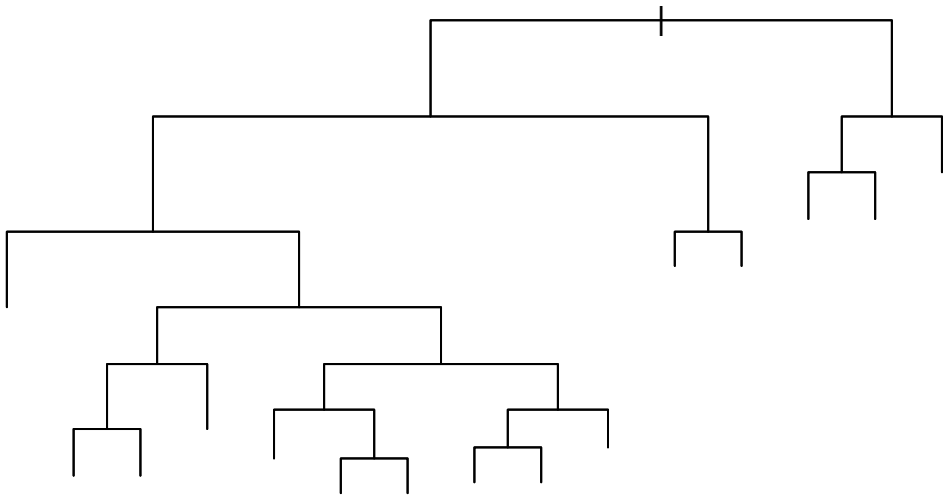

Figure 4: The resulting decision tree using deviance.

Below are the results using the deviance to measure the impurity. The first result is on the train set and the second on the test set.

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   61   20
#>   good  86  333
#>
#> $misclassification_rate
```

```
#> [1] 0.212
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   34   21
#>   good  41  154
#>
#> $misclassification_rate
#> [1] 0.248
```

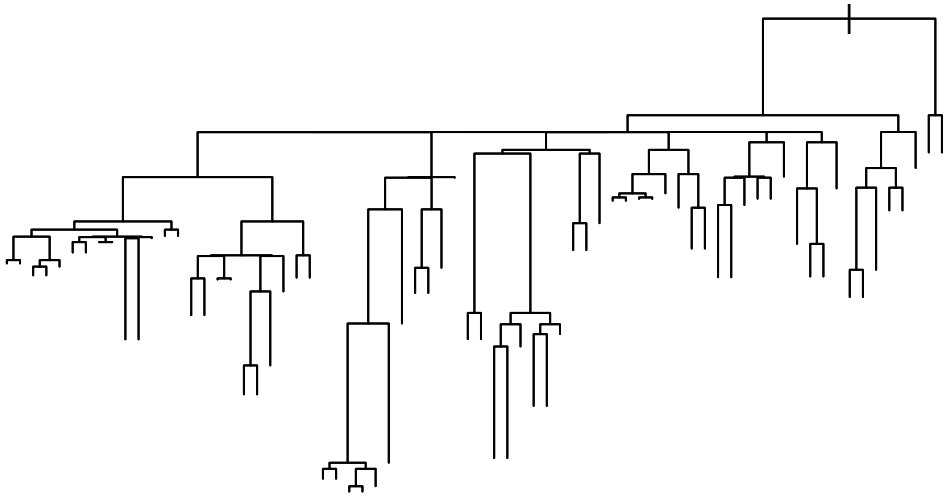

Figure 5: The resulting decision tree using Gini index.

We can see that the decision tree constructed by using Gini index (fig. 5) is a lot bigger than that of deviance (fig. 4).

Below are the results using the Gini index to measure the impurity. The first result is on the train set and the second on the test set.

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   59   32
#>   good  88  321
#>
#> $misclassification_rate
#> [1] 0.24
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   18   26
#>   good  57  149
#>
#> $misclassification_rate
#> [1] 0.332
```

We can see from the results that the deviance impurity measurement performs better on both the train and test data sets. I will therefore only be using the deviance measure in the following exercises.

3

Here I was interested in finding the optimal number of leaves in the tree and I used the validation set to measure the performance by computing the deviance. Figure 6 shows how the deviance varies with increasing number of leave nodes in the decision tree. We can see as the number of leaves increases the deviance decreases for the train set which indicate that the model starts overfitting the data. The deviance on the validation set tells a different story and the optimal number of leaves, i.e. yield the lowest deviance, is 4.

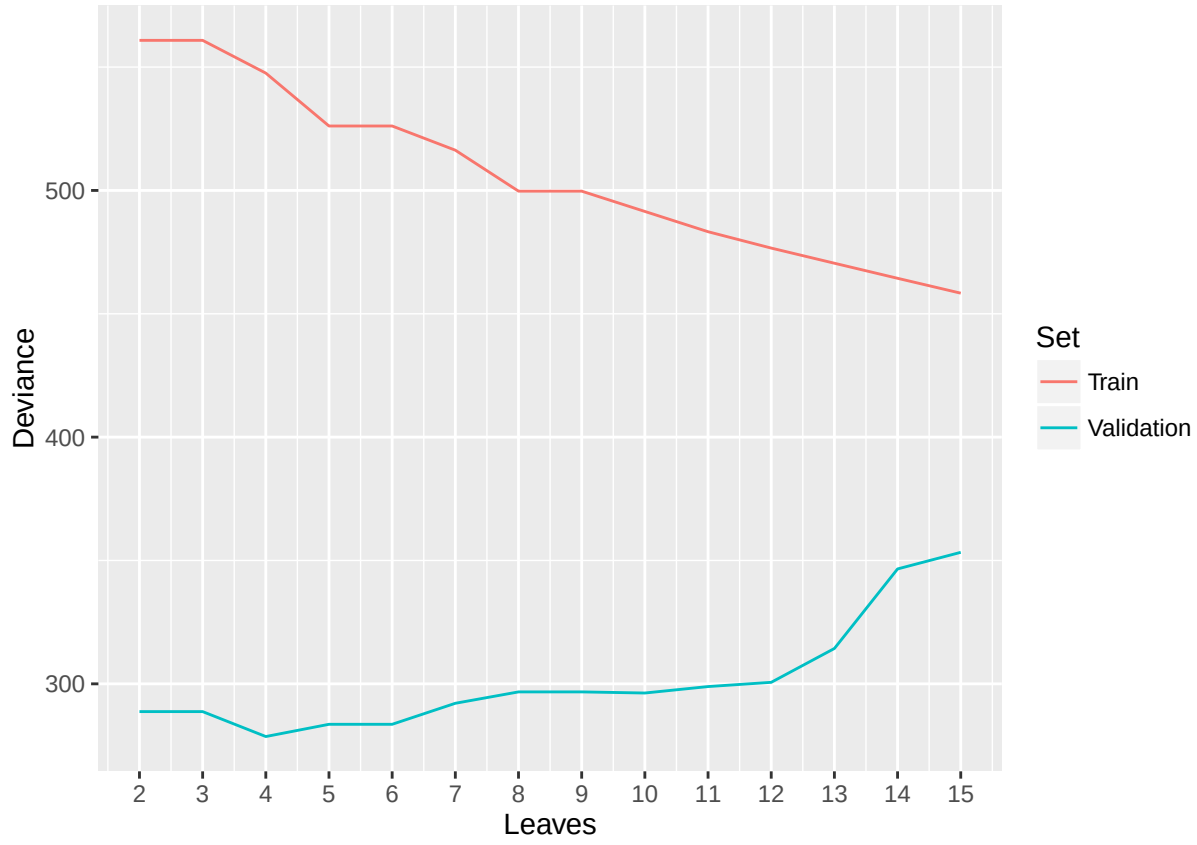

Figure 6: Deviance score when adjusting the number of leaves (terminal nodes) in the decision tree.

Given the optimal number of leaves being 4 from the validation set I pruned the tree to match that. Below are the predictions and misclassification rate on the test data set. It has increased the misclassification rate

on the test set than the tree found previously but the complexity has been reduced significantly, from 15 leaves to 4.

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   22   12
#>   good  53  163
#>
#> $misclassification_rate
#> [1] 0.26
```

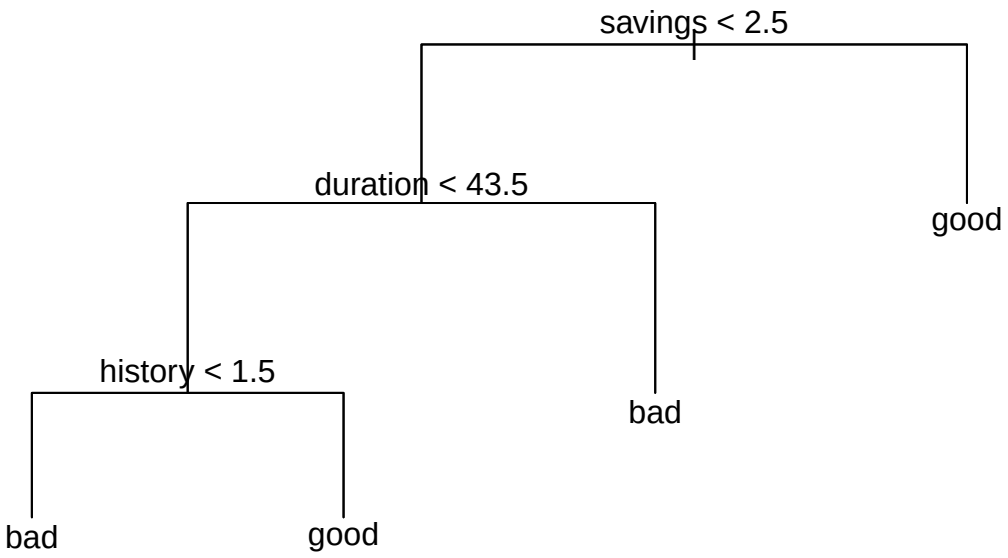

Figure 7: A visual representation of the optimal decision tree.

Figure 7 shows the optimal tree with a depth of 3. We can see that having large savings are indicative of being good which is intuitively sound and if a customer have been a member of the company for a longer period of time it also indicates good loan management. I do not understand what the history variable means and can therefore draw any conclusions as to say if it make sense or not.

4

Here I will be using the naive Bayes classifier instead and below are the results. The first result is on the train set and the second on the test set.

```
#> $confusion_matrix
#>       true
```

```
#> pred   bad good
#>   bad   95   98
#>   good  52  255
#>
#> $misclassification_rate
#> [1] 0.3
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   50   61
#>   good  25  114
#>
#> $misclassification_rate
#> [1] 0.344
```

The misclassification rate on the test data is far worse than the gotten from the optimal decision tree found in exercise 3.

5

By changing the loss matrix, i.e. how much each error costs, the classification result will look differently. In the above example the cost was equally costly but here I have set that predicting good when the truth is bad, i.e. true negative, is 10 times as costly. Below are the results and the first result is on the train set and the second on the test set.

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad  137  263
#>   good  10   90
#>
#> $misclassification_rate
#> [1] 0.546
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   66  130
#>   good   9   45
#>
#> $misclassification_rate
#> [1] 0.556
```

What we can observe is that there are a lot of false negatives, i.e. predicting bad when the true outcome is good, because it is less costly to make that kind of error. The classifier has basically become scared of predicting good and this has in turn increased the misclassification rate significantly.

Appendix

Code for Assignment 1

```
library(ggplot2)
library(glmnet)

data <- read.csv("../data/australian-crabs.csv", sep=",")
ggplot(data) +
    geom_point(aes(x=RW, y=CL, color=sex))
LDA <- function(X, y) {
    n <- nrow(X)
    p <- ncol(X)

    labels <- unique(y)
    priors <- table(y) / n

    means <- aggregate(X, list(y), mean)
    means <- as.matrix(means[, -1], ncol=p)

    lengths <- by(X, list(y), nrow)

    cov_mats <- by(X, list(y), cov)
    cov_mats <- lapply(1:length(lengths), function(i) {
        cov_mats[[i]] * lengths[[i]]
    })

    sigma <- as.matrix(Reduce("+", cov_mats) / n, nrow=p)
    sigma_inv <- solve(sigma)

    w0 <- sapply(1:length(labels), function(i) {
        -(1 / 2) * t(means[i,]) %*% sigma_inv %*% means[i,] + log(priors[i])
    })

    w1 <- sapply(1:length(labels), function(i) {
        sigma_inv %*% means[i, ]
    })

    names(w0) <- levels(labels)
    colnames(w1) <- levels(labels)

    list(w0=w0, w1=w1, sigma=sigma)
}

X <- cbind(data$RW, data$CL)
y <- data$sex

result <- LDA(X, y)

w1 <- result$w1[, 2] - result$w1[, 1]
w0 <- result$w0[2] - result$w0[1]

intercept <- -w0 / w1[2]
```

```
slope <- -w1[1] / w1[2]
predicted <- as.numeric((w0 + w1 %*% t(X)) > 0)
predicted <- factor(predicted, levels=c(0, 1), labels=c("Female", "Male"))
plot_data <- data.frame(RW=data$RW, CL=data$CL, class=predicted)
line_data <- data.frame(intercept=intercept, slope=slope, row.names=NULL)

ggplot() +
    geom_point(data=plot_data, aes(x=RW, y=CL, color=class)) +
    geom_abline(data=line_data, intercept=intercept, slope=slope,
                color="black", linetype="dotted", size=1)
logistic_data <- data.frame(sex=as.numeric(data$sex) - 1, RW=data$RW, CL=data$CL)

glmfit <- glm(sex ~ RW + CL, data=logistic_data, family=binomial(link=logit))
coefficients <- coef(glmfit)

predicted <- as.numeric(glmfit$fitted.values > 0.5)
predicted <- factor(predicted, levels=c(0, 1), labels=c("Female", "Male"))

intercept <- -coefficients[1] / coefficients[3]
slope <- -coefficients[2] / coefficients[3]

plot_data <- data.frame(RW=data$RW, CL=data$CL, class=predicted)
line_data <- data.frame(intercept=intercept, slope=slope)

ggplot() +
    geom_point(data=plot_data, aes(x=RW, y=CL, color=class)) +
    geom_abline(data=line_data, intercept=intercept, slope=slope,
                color="black", linetype="dotted", size=1)
```

Code for Assignment 2

```
library(gdata)
library(tree)
library(partykit)
library(ggplot2)
library(reshape2)
library(e1071)

data_division <- function(n, train, test, validation) {
    indices <- 1:n

    train <- sample(indices, floor(n * train))
    test <- sample(indices[-train], floor(n * test))
    validation <- indices[-c(train, test)]

    list(train=train, test=test, validation=validation)
}

data <- read.xls("../data/creditscoring.xls")
set.seed(12345)
indices <- data_division(n=nrow(data), train=0.5, test=0.25, validation=0.25)
```

```
train <- data[indices$train,]
test <- data[indices$test,]
validation <- data[indices$validation,]

prediction.tree <- function(model, X, y) {
    predicted <- predict(model, X)
    predicted <- factor(ifelse(predicted[, 1] > predicted[, 2], 0, 1),
                        levels=c(0, 1), labels=c("bad", "good"))

    confusion_matrix <- table(pred=predicted, true=y)
    list(confusion_matrix=confusion_matrix,
         misclassification_rate= 1 - sum(diag(confusion_matrix)) / sum (confusion_matrix))
}

dtreefit <- tree(good_bad ~ ., data=train, split="deviance")
gtreefit <- tree(good_bad ~ ., data=train, split="gini")
plot(dtreefit)
prediction.tree(dtreefit, train[, -ncol(train)], train$good_bad)
prediction.tree(dtreefit, test[, -ncol(test)], test$good_bad)
plot(gtreefit)
prediction.tree(gtreefit, train[, -ncol(train)], train$good_bad)
prediction.tree(gtreefit, test[, -ncol(test)], test$good_bad)

leaves <- 2:summary(dtreefit)[4]$size
train_score <- rep(0, max(leaves))
validation_score <- rep(0, max(leaves))

for(i in leaves) {
    prunedTree <- prune.tree(dtreefit, best=i)
    pred <- predict(prunedTree, newdata=validation, type="tree")
    train_score[i] <- deviance(prunedTree)
    validation_score[i] <- deviance(pred)
}
plot_data <- data.frame(Leaves=leaves,
                        Train=train_score[leaves],
                        Validation=validation_score[leaves])
plot_data <- melt(plot_data, id="Leaves", value.name="Deviance", variable.name="Set")

ggplot() +
    geom_line(data=plot_data, aes(x=Leaves, y=Deviance, color=Set)) +
    scale_x_continuous(breaks=leaves)
optimal_leaves <- which.min(validation_score[leaves]) + 1
optimal_tree <- prune.tree(dtreefit, best=optimal_leaves)
prediction.tree(optimal_tree, test[, -ncol(test)], test$good_bad)
plot(optimal_tree)
text(optimal_tree, pretty=0)

prediction.bayes <- function(model, X, y, loss) {
    predicted <- predict(model, X, type="raw")
    predicted <- factor(ifelse(predicted[, 1] / predicted[, 2] > loss, 0, 1),
                        levels=c(0, 1), labels=c("bad", "good"))

    confusion_matrix <- table(pred=predicted, true=y)
```

```
    list(confusion_matrix=confusion_matrix,
         misclassification_rate= 1 - sum(diag(confusion_matrix)) / sum (confusion_matrix))
}

bayesfit <- naiveBayes(good_bad ~ ., data=train)
prediction.bayes(bayesfit, train[, -ncol(train)], train$good_bad, 1)
prediction.bayes(bayesfit, test[, -ncol(test)], test$good_bad, 1)

bayesfit <- naiveBayes(good_bad ~ ., data=train)
prediction.bayes(bayesfit, train[, -ncol(train)], train$good_bad,  1 / 10)
prediction.bayes(bayesfit, test[, -ncol(test)], test$good_bad, 1 / 10)
```