Introduction to Machine Learning

Lab 3

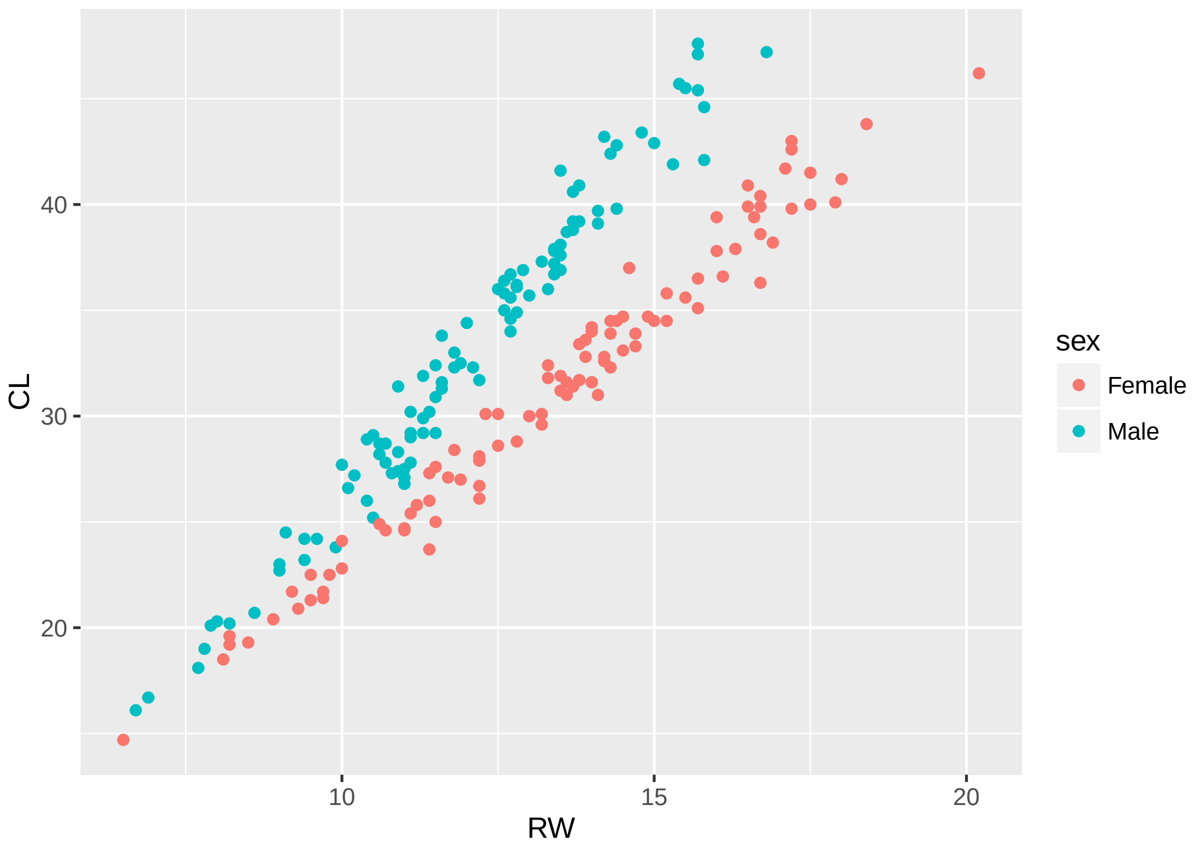
Rasmus Holm

2016-11-18

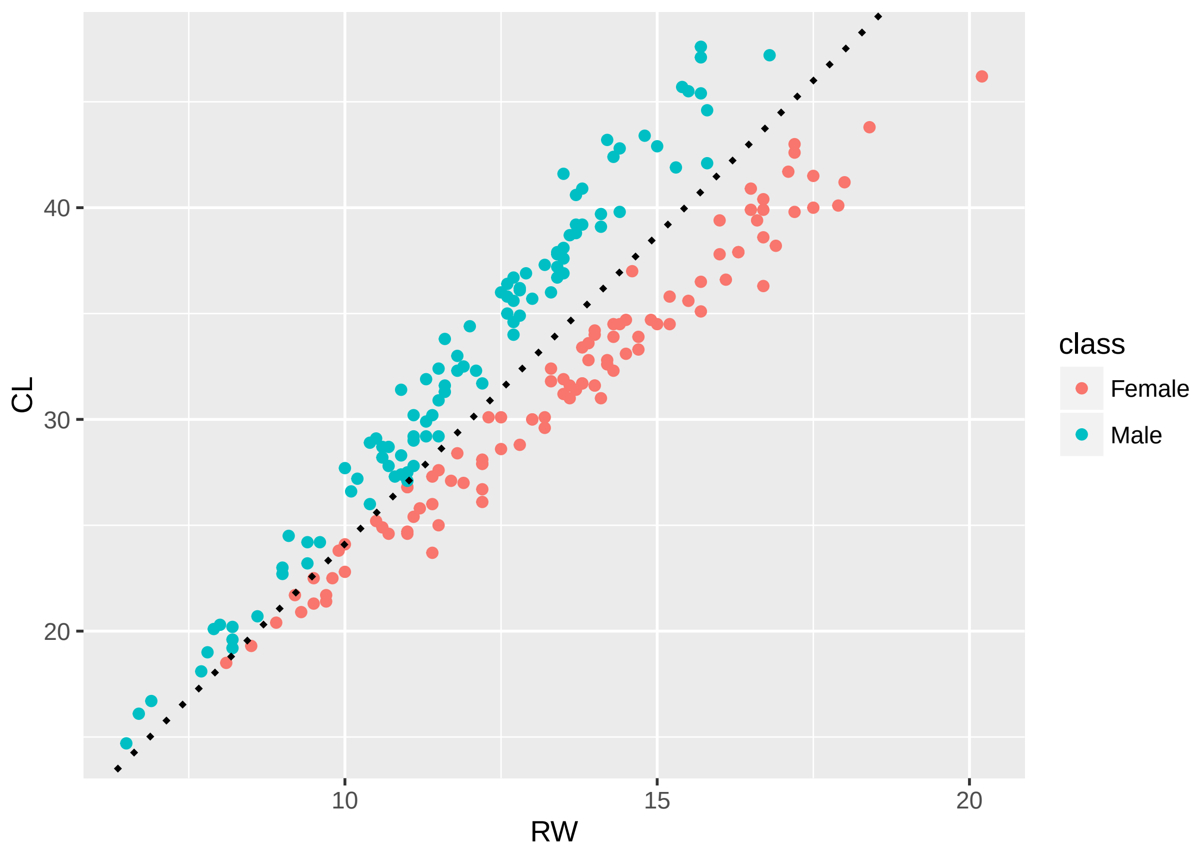
Contents

Assignment 1

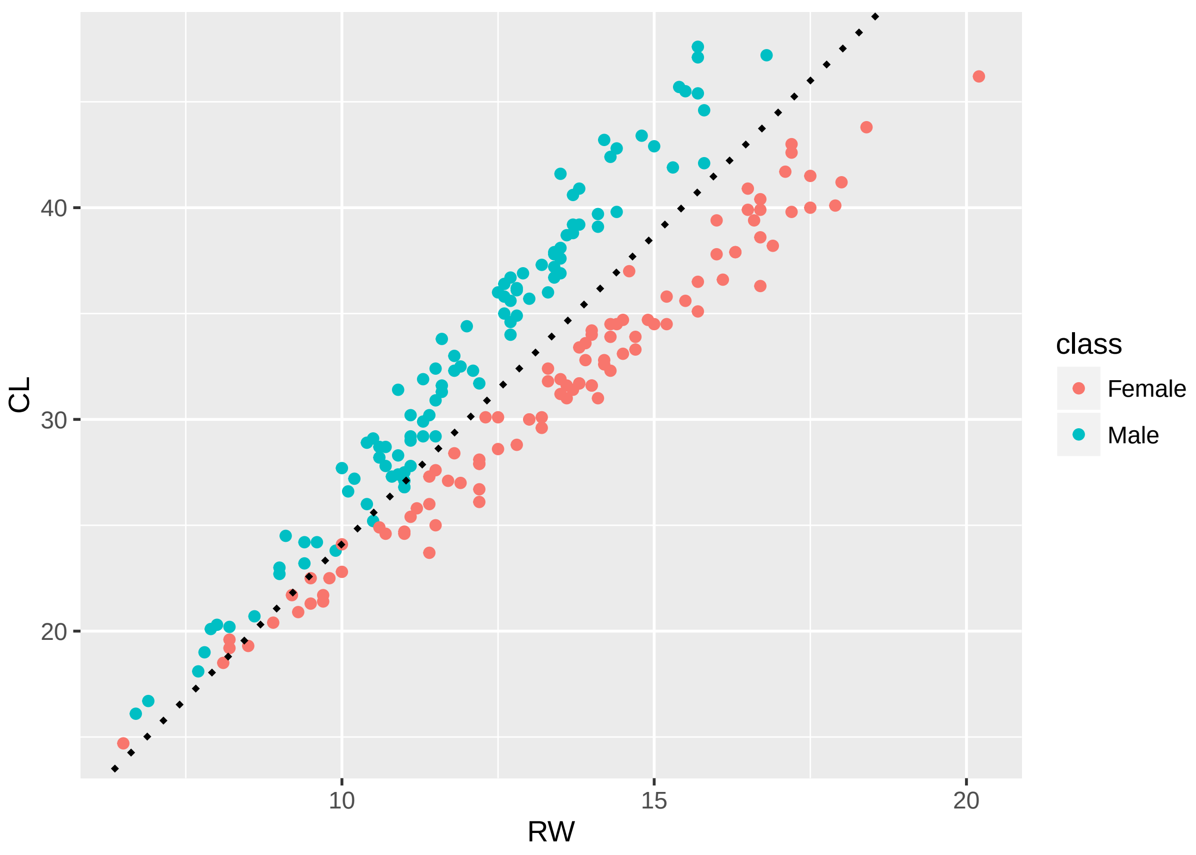
1



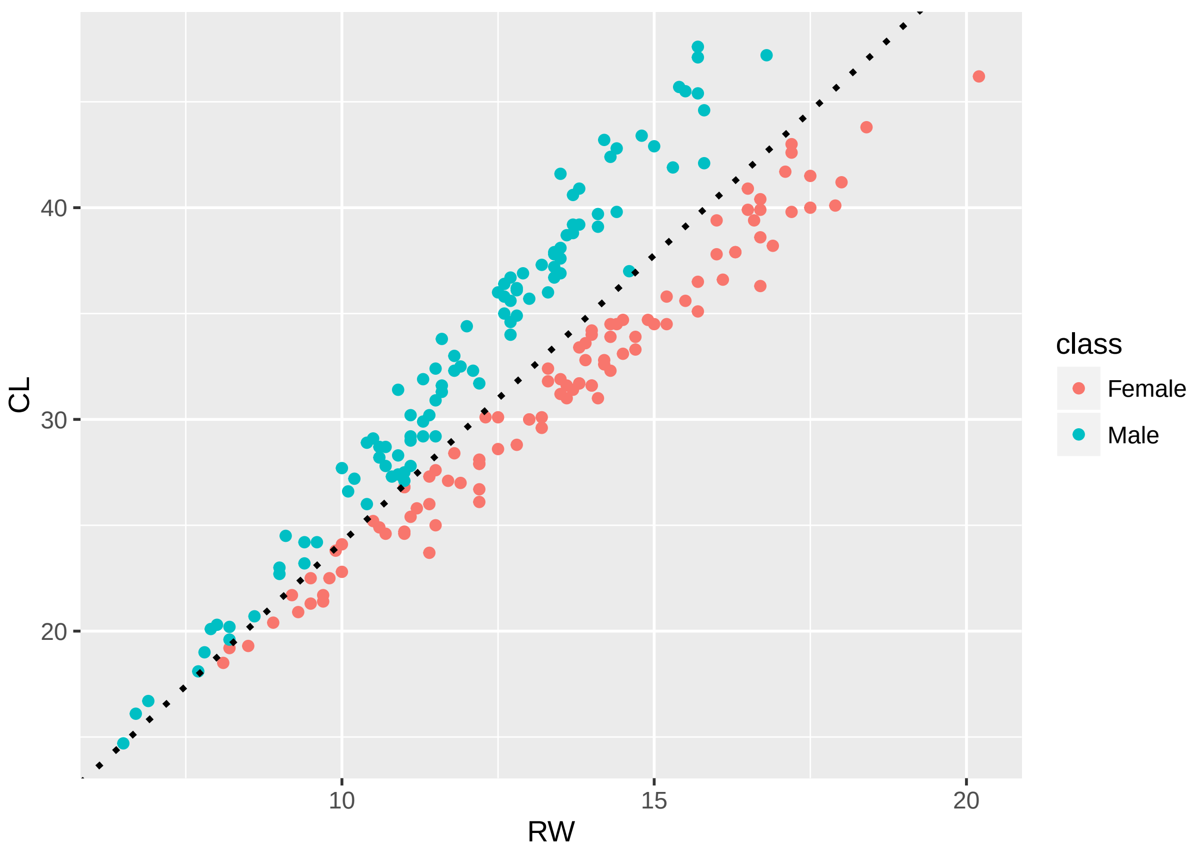
2



3



4



Assignment 2

1

2

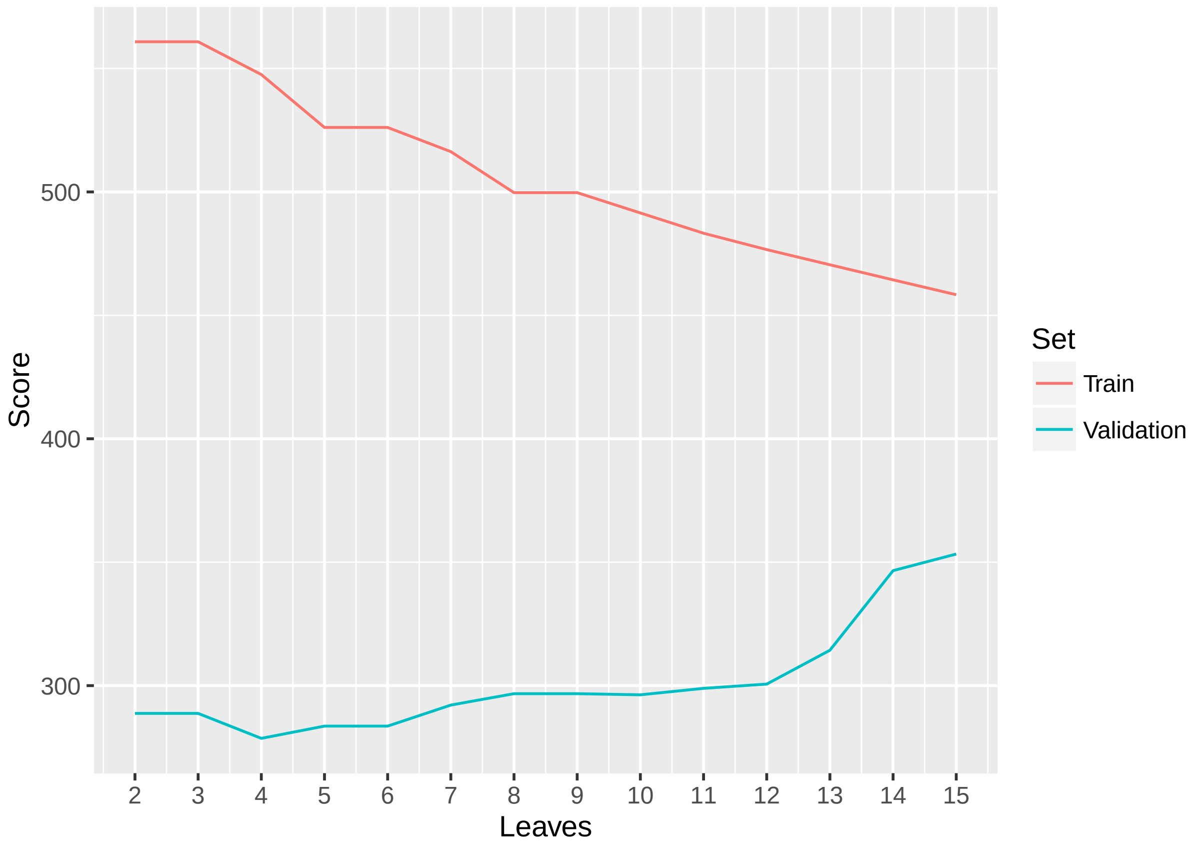
```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   61   20
#>   good  86  333
#> 
#> $classification_rate
#> [1] 0.788
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   34   21
#>   good  41  154
#> 
#> $classification_rate
#> [1] 0.752
```

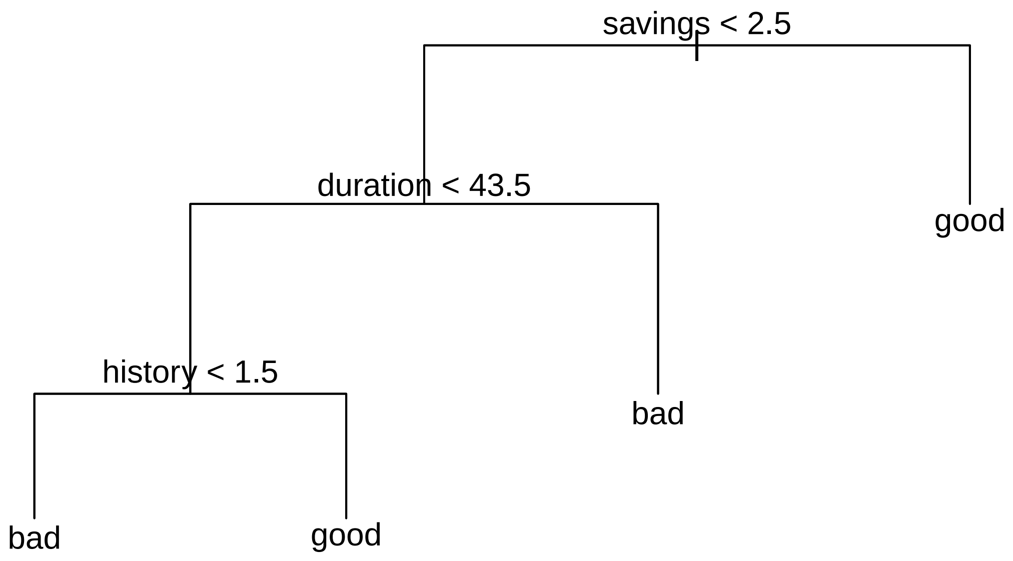
```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   59   32
#>   good  88  321
#> 
#> $classification_rate
#> [1] 0.76
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   18   26
#>   good  57  149
#> 
#> $classification_rate
#> [1] 0.668
```

3



```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   22   12
#>   good  53  163
#> 
#> $classification_rate
#> [1] 0.74
```



4

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   95   98
#>   good  52  255
#> 
#> $classification_rate
#> [1] 0.7
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   50   61
#>   good  25  114
#> 
#> $classification_rate
#> [1] 0.656
```

5

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   95   98
#>   good  52  255
#>
#> $classification_rate
#> [1] 0.7
```

```
#> $confusion_matrix
#>       true
#> pred   bad good
#>   bad   50   61
#>   good  25  114
#>
#> $classification_rate
#> [1] 0.656
```

Appendix

Code for Assignment 1

```r
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) / length(y)

    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) / sum(lengths), 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]
```

```r
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)

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)
plot_data <- data.frame(RW=data$RW, CL=data$CL, class=data$sex)
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)
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

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

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

    train <- sample(indices, floor(n * 0.5))
    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(nrow(data), 0.5, 0.25, 0.25)

train <- data[indices$train,]
test <- data[indices$test,]
validation <- data[indices$validation,]
prediction <- function(model, X, y) {
    predicted <- predict(model, X)

    if (is.matrix(predicted)) {
        predicted <- factor(ifelse(predicted[, 1] > predicted[, 2], 0, 1), levels=c(0, 1), labels=c("ba
    }

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

dtreefit <- tree(good_bad ~ ., data=train, split="deviance")
gtreefit <- tree(good_bad ~ ., data=train, split="gini")

prediction(dtreefit, train[, -ncol(train)], train$good_bad)
prediction(dtreefit, test[, -ncol(test)], test$good_bad)
prediction(gtreefit, train[, -ncol(train)], train$good_bad)
prediction(gtreefit, test[, -ncol(test)], test$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="Score", variable.name="Set")

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

bayesfit <- naiveBayes(good_bad ~ ., data=train)
```

```r
prediction(bayesfit, train[, -ncol(train)], train$good_bad)
prediction(bayesfit, test[, -ncol(test)], test$good_bad)

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