```
## ----setup,
include=FALSE------------------------------------------------------
knitr::opts_chunk$set(echo = TRUE)
library(glmnet)
library(tree)
library(caret)
library(dplyr)


## ----
echo=FALSE-------------------------------------------------------------
tecator = read.csv("tecator.csv", header = T)
n = dim(tecator)[1]
set.seed(12345)
df = data.frame(tecator[c(2:102)])

id=sample(1:n, floor(n*0.5))
train = df[id,]
test = df[-id,]

fit = lm(Fat~ ., data = train)
train_preds = predict(fit, train)
test_preds = predict(fit, test)
sum = summary(fit)

MSE_train=mean((train_preds - train$Fat)^2)
MSE_test=mean((test_preds - test$Fat)^2)

print("Test error")
MSE_test
print("Train error")
MSE_train


## ---- dev='png',warning=FALSE,
echo=FALSE-----------------------------------------

y = train$Fat
x = train[1:100]
model_lasso= glmnet(as.matrix(x), as.matrix(y), alpha=1,family="gaussian")

plot(model_lasso, xvar = "lambda")
ynew=predict(model_lasso, newx=as.matrix(x), type="response")


## ----
echo=FALSE-------------------------------------------------------------

y = train$Fat
x = train[1:100]
model_lasso= glmnet(as.matrix(x), as.matrix(y), alpha=0,family="gaussian")

plot(model_lasso, xvar = "lambda")
ynew=predict(model_lasso, newx=as.matrix(x), type="response")
```

```
## ---- warning=FALSE,
echo=FALSE---------------------------------------------------


model_lasso= cv.glmnet(as.matrix(x), as.matrix(y), alpha=1,family="gaussian")


lambda_min = model_lasso$lambda.min
plot(model_lasso, xvar = "lambda")

better_model = glmnet(as.matrix(x), as.matrix(y), lambda = lambda_min, alpha =
1, family = "gaussian")

ynew=predict(better_model, newx=as.matrix(x), s = lambda_min ,
type="response")
plot(y, ynew, xlab = "Original", ylab = "Predicted",col = "red", main =
"Scatter plot")
abline(0,1)




## ---- echo=FALSE,
warning=FALSE---------------------------------------------------

d = read.csv("bank-full.csv", sep = ";", stringsAsFactors = TRUE)
data = d
data$duration = c() #remove duration column
output = d['y']
n = dim(data)[1]

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]




## ---- echo=FALSE,
warning=FALSE---------------------------------------------------

fit=tree(as.factor(y)~., data=train)
```

```
plot(fit)
text(fit, pretty=0)


fit2=tree(as.factor(y)~., data=train, minsize=7000)
plot(fit2)
text(fit2, pretty=0)


fit3=tree(as.factor(y)~., data=train, mindev=0.0005)
plot(fit3)
text(fit3, pretty=0)




## ---- echo=FALSE,
warning=FALSE----------------------------------------------------
Yfit_t=predict(fit, newdata=train, type="class")
t1<-table(train$y,Yfit_t)
mis_t1 <- 1-sum(diag(t1))/sum(t1)

Yfit_t2=predict(fit2, newdata=train, type="class")
t2<-table(train$y,Yfit_t2)
mis_t2 <- 1-sum(diag(t2))/sum(t2)

Yfit_t3=predict(fit3, newdata=train, type="class")
t3<-table(train$y,Yfit_t3)
mis_t3 <- 1-sum(diag(t3))/sum(t3)

Yfit_v=predict(fit, newdata=valid, type="class")
v1<-table(valid$y,Yfit_v)
mis_v1<-1-sum(diag(v1))/sum(v1)

Yfit_v2=predict(fit2, newdata=valid, type="class")
v2<-table(valid$y,Yfit_v2)
mis_v2<-1-sum(diag(v2))/sum(v2)

Yfit_v3=predict(fit3, newdata=valid, type="class")
v3<-table(valid$y,Yfit_v3)
mis_v3<-1-sum(diag(v3))/sum(v3)


print("1) Training and validation")
print(mis_t1)
print(mis_v1)
print("2) Training and validation")
print(mis_t2)
print(mis_v2)
print("3) Training and validation")
print(mis_t3)
print(mis_v3)
```

```
## ---- echo=FALSE,
warning=FALSE--------------------------------------------------
trainScore=rep(0,50)
testScore=rep(0,50)
for(i in 2:50) {
  prunedTree=prune.tree(fit3,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:50, trainScore[2:50], type="b", col="red", ylim=c(min(testScore[-1]),
max(trainScore[-1])))
points(2:50, testScore[2:50], type="b", col="blue")
optimal_leaves = which.min(testScore[2:50])

finalTree=prune.tree(fit3, best=optimal_leaves)
finalfit=predict(finalTree, newdata=valid, type="class")
tab = table(valid$y,finalfit)
plot(finalTree)

summary(fit3)
summary(finalTree)
#text(fit3, pretty=0)



## ---- echo=FALSE,
warning=FALSE--------------------------------------------------

opt_tree=prune.tree(fit3, best=optimal_leaves)

ffitTest<-predict(opt_tree, newdata=test, type="class")

c_m = table(test$y, ffitTest)
c_m

acc = sum(c_m[1], c_m[4])/sum(c_m[1:4])
prec = c_m[4] / sum(c_m[4], c_m[2])
recall = c_m[4] / sum(c_m[4], c_m[3])
f1_score = (2*(recall*prec)) / (recall + prec)




## ---- echo=FALSE,
warning=FALSE--------------------------------------------------


predtree5 <- predict(opt_tree, newdata=test, type="vector")

probY <- predtree5[,2]
```

```
probN <- predtree5[,1]

pred5 <- ifelse((probY/probN)>1/5, "yes", "no")

c_m <- table(test$y, pred5)
c_m
acc = sum(c_m[1], c_m[4])/sum(c_m[1:4])
prec = c_m[4] / sum(c_m[4], c_m[2])
recall = c_m[4] / sum(c_m[4], c_m[3])
f1_score = (2*(recall*prec)) / (recall + prec)




## ---- echo=FALSE,
warning=FALSE--------------------------------------------------
optimalTree <- tree(as.factor(y)~., data=train, mindev=0.0005)
optimalTree <- prune.tree(optimalTree, best=21)

pi <- seq(0.05, 0.95, 0.05)

logic_model <- glm(as.factor(y)~.,data = train  ,family="binomial")
pred6_probY = predict(logic_model, newdata = test, type = "response")
pred6_probN = 1 -pred6_probY

tree_pred = predict(optimalTree, newdata = test, type = "vector")

fpr_1 <- c(1:length(pi))
tpr_1 <- c(1:length(pi))
fpr_2 <- c(1:length(pi))
tpr_2 <- c(1:length(pi))
pred6 = tree_pred[, 1]

for (i in 1:length(pi)){

  #Tree
  tpr_1[i] = 0
  fpr_1[i] = 0

  pred6 <- ifelse(tree_pred[,2]>pi[i],  "yes", "no")
  pred6_matrix <- table(test$y, pred6)

  pred6_matrix

  if(ncol(pred6_matrix) > 1){
    tpr_1[i] <- pred6_matrix[2,2] / (pred6_matrix[2,1]+pred6_matrix[2,2])
    fpr_1[i] <- pred6_matrix[1,2] / (pred6_matrix[1,1]+pred6_matrix[1,2])
  }

  #Logistic regression#
  tpr_2[i] = 0
  fpr_2[i] = 0
  pred6_logic <- ifelse(pred6_probY > pi[i], "yes", "no")
  pred6_logic_matrix <- table(test$y,pred6_logic)
  tpr_2[i] <- pred6_logic_matrix[2,2] / (pred6_logic_matrix[2,1]
+pred6_logic_matrix[2,2])
  fpr_2[i] <- (pred6_logic_matrix[1,2] / (pred6_logic_matrix[1,1]
+pred6_logic_matrix[1,2]))


}
plot(fpr_1,tpr_1, type='l',xlim = c(0,1), ylim = c(0,1),
     xlab='FPR', ylab='TPR', col='red')
lines(fpr_2,tpr_2, type='l',xlim = c(0,1),
     xlab='FPR', ylab='TPR', col='blue')
legend(x = "bottomright" , col = c("red", "blue", "black"), legend = c("Tree",
"Logistic", "Reference"), lwd = 2,title = "Lines", lty = c(1,1,5))
abline(0,1, lty = 5)



## ----echo=FALSE,
warning=FALSE---------------------------------------------------
rm(list = ls(all = TRUE))
graphics.off()
shell("cls")

data = read.csv(file = "communities.csv",
                header = TRUE)

index <- names(data) %in% "ViolentCrimesPerPop"

data.scaled <- scale(x = data[, !index],
                     center = TRUE,
                     scale = TRUE)

e = eigen(cov(data[, -1]))

e.scaled = eigen(cov(data.scaled))

cum_var = cumsum(e.scaled$values/sum(e.scaled$values))
sum(cum_var<0.95)
e.scaled$values[1:2]/sum(e.scaled$values)




## ---- warning=FALSE,
echo=FALSE------------------------------------------------------
data = read.csv(file = "communities.csv",
                header = TRUE)

index <- names(data) %in% "ViolentCrimesPerPop"

data.scaled <- scale(x = data[, !index],
                     center = TRUE,
                     scale = TRUE)
```

```
pr=princomp(data.scaled)

#eigenvalues
lambda=pr$sdev^2


#proportion of variation
var = sprintf("%2.3f",lambda/sum(lambda)*100)


ev1 = pr$loadings[,1]

ev1[order(abs(ev1),decreasing = TRUE)[1:5]]


library(ggfortify)

autoplot(pr, data = data, colour = "ViolentCrimesPerPop")



## ----
echo=FALSE------------------------------------------------------------------
df = read.csv("communities.csv") #reload the data.

#scale and split 50/50
df = scale(df, TRUE, TRUE)
set.seed(12345)

n <- dim(df)[1]
id <- sample(1:n,floor(n*0.5))
df_train <- data.frame(df[id,])
df_test <- data.frame(df[-id,])


lr = lm(ViolentCrimesPerPop ~ .,df_train)

train.pred = predict(lr, df_train)
test.pred = predict(lr, df_test)

train_MSE = mean((train.pred - df_train$ViolentCrimesPerPop) ^ 2)
test_MSE = mean((test.pred - df_test$ViolentCrimesPerPop) ^ 2)

print("Train error")
train_MSE
print("Test error")
test_MSE



## ----
echo=FALSE------------------------------------------------------------------
train_error <<- numeric(0)
test_error <<-numeric(0)
```

```
set.seed(12345)

cost <- function(theta, train, acc_train, test, acc_test){
  pred_train = train %*% theta
  pred_test = test %*% theta


  mse_train = mean((acc_train-pred_train)^2)
  train_error <<- append(train_error,mse_train)

  mse_test = mean((acc_test - pred_test)^2)
  test_error <<- append(test_error,mse_test)

  return(mse_train)
}



trainy = as.matrix(df_train[,1:(dim(df_train)[2]-1)])
acc_train = as.matrix(df_train['ViolentCrimesPerPop'])

testy = as.matrix(df_test[,1:(dim(df_test)[2]-1)])
acc_test = as.matrix(df_test['ViolentCrimesPerPop'])



theta =numeric(dim(trainy)[2])
theta = as.matrix(theta)


opt = optim(par=theta,fn=cost, train = trainy, acc_train = acc_train,
test=testy, acc_test=acc_test,method = "BFGS")

opt_theta = opt$par

train_opt_error = opt$value

test_opt_error = mean((acc_test - (testy %*% opt_theta))^2)

print("calculated optimal train")
train_opt_error
print("Lm train error")
train_MSE

print("calculated optimal test")
test_opt_error
print("Lm test error")
test_MSE

excluded = c(TRUE,rep(FALSE,500))

rest_train = train_error[excluded]
rest_test = test_error[excluded]
```

```
test_min_ind = which(test_error==min(test_error))

print("Early stopping index and MSE")
test_min_ind
min(test_error)

plot(rest_train, xlim=c(0,length(rest_train)), ylim=c(0,1.5), col = "blue")
points(rest_test, col="red")
lines(c(0,1000), rep(train_MSE, 2), col="blue")
lines(c(0,1000), rep(test_MSE, 2), col="red")
```