# 5) A)Fit a logistic regression model that uses income and balance to

# predict default.

attach(Default)

set.seed(1)

reg.glm = glm(default ~ income + balance, data = Default, family = "binomial")

summary(reg.glm)

# Call:

# glm(formula = default ~ income + balance, family = "binomial",

# data = Default)

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.4725 -0.1444 -0.0574 -0.0211 3.7245

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*

# income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*

# balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# (Dispersion parameter for binomial family taken to be 1)

# Null deviance: 2920.6 on 9999 degrees of freedom

# Residual deviance: 1579.0 on 9997 degrees of freedom

# AIC: 1585

# Number of Fisher Scoring iterations: 8

## B)Using the validation set approach, estimate the test error of this

# model. In order to do this, you must perform the following steps:

## I)Split the sample set into a training set and a validation set.

train = sample(dim(Default)[1], dim(Default)[1] / 2)

## II)Fit a multiple logistic regression model using only the training

# observations.

reg.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

summary(reg.glm)

# Call:

# glm(formula = default ~ income + balance, family = "binomial",

# data = Default, subset = train)

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.3583 -0.1268 -0.0475 -0.0165 3.8116

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) -1.208e+01 6.658e-01 -18.148 <2e-16 \*\*\*

# income 1.858e-05 7.573e-06 2.454 0.0141 \*

# balance 6.053e-03 3.467e-04 17.457 <2e-16 \*\*\*

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# (Dispersion parameter for binomial family taken to be 1)

# Null deviance: 1457.0 on 4999 degrees of freedom

# Residual deviance: 734.4 on 4997 degrees of freedom

# AIC: 740.4

# Number of Fisher Scoring iterations: 8

## III)Obtain a prediction of default status for each individual in

# the validation set by computing the posterior probability of

# default for that individual, and classifying the individual to

# the default category if the posterior probability is greater

# than 0.5.

# probs = predict(reg.glm, newdata = Default[-train, ], type = "response")

# pred.glm = rep("No", length(probs))

# pred.glm[probs > 0.5] <- "Yes"

## IV)Compute the validation set error, which is the fraction of

# the observations in the validation set that are misclassified.

mean(pred.glm != Default[-train, ]$default)

# 0.0286

## C)Repeat the process in (b) three times, using three different splits

# of the observations into a training set and a validation set. Comment

# on the results obtained.

train = sample(dim(Default)[1], dim(Default)[1] / 2)

reg.glm = glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

probs = predict(reg.glm, newdata = Default[-train, ], type = "response")

pred.glm = rep("No", length(probs))

pred.glm[probs > 0.5] <- "Yes"

mean(pred.glm != Default[-train, ]$default)

# 0.0236

train = sample(dim(Default)[1], dim(Default)[1] / 2)

reg.glm = glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

probs = predict(reg.glm, newdata = Default[-train, ], type = "response")

pred.glm = rep("No", length(probs))

pred.glm[probs > 0.5] <- "Yes"

mean(pred.glm != Default[-train, ]$default)

# 0.028

train = sample(dim(Default)[1], dim(Default)[1] / 2)

reg.glm = glm(default ~ income + balance, data = Default, family = "binomial", subset = train)

probs = predict(reg.glm, newdata = Default[-train, ], type = "response")

pred.glm = rep("No", length(probs))

pred.glm[probs > 0.5] <- "Yes"

mean(pred.glm != Default[-train, ]$default)

# 0.0268

# A estimativa de validação da taxa de erro de teste pode

# ser variável dependendo de quais observações são incluídas

# no conjunto de treinamento e no conjunto de validação.

## D)Now consider a logistic regression model that predicts the probability

# of default using income, balance, and a dummy variable

# for student. Estimate the test error for this model using the validation

# set approach. Comment on whether or not including a

# dummy variable for student leads to a reduction in the test error

# rate.

train = sample(dim(Default)[1], dim(Default)[1] / 2)

reg.glm = glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)

pred.glm = rep("No", length(probs))

probs = predict(reg.glm, newdata = Default[-train, ], type = "response")

pred.glm[probs > 0.5] <- "Yes"

mean(pred.glm != Default[-train, ]$default)

# 0.0264

# Adicionar a variável dummy de student não reduz a

# estimativa do conjunto de validação da taxa de erro de teste.

# 6) A)Using the summary() and glm() functions, determine the estimated

# standard errors for the coefficients associated with income

# and balance in a multiple logistic regression model that uses

# both predictors.

set.seed(1)

attach(Default)

# The following objects are masked from Default (pos = 3):

# balance, default, income, student

reg.glm = glm(default ~ income + balance, data = Default, family = "binomial")

summary(reg.glm)

# Call:

# glm(formula = default ~ income + balance, family = "binomial",

# data = Default)

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.4725 -0.1444 -0.0574 -0.0211 3.7245

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*

# income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*

# balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# (Dispersion parameter for binomial family taken to be 1)

# Null deviance: 2920.6 on 9999 degrees of freedom

# Residual deviance: 1579.0 on 9997 degrees of freedom

# AIC: 1585

# Number of Fisher Scoring iterations: 8

# As estimativas de glm() dos erros padrão para os coeficientes

# beta 0, beta 1 e beta 2 são 0.4348, 4,985 e 2,274.

## B)Write a function, boot.fn(), that takes as input the Default data

# set as well as an index of the observations, and that outputs

# the coefficient estimates for income and balance in the multiple

# logistic regression model.

boot.fn = function(data, index) {

reg = glm(default ~ income + balance, data = data, family = "binomial", subset = index)

return (coef(reg))

}

## C)Use the boot() function together with your boot.fn() function to

# estimate the standard errors of the logistic regression coefficients

# for income and balance.

library(boot)

boot(Default, boot.fn, 1000)

# ORDINARY NONPARAMETRIC BOOTSTRAP

# Call:

# boot(data = Default, statistic = boot.fn, R = 1000)

# Bootstrap Statistics :

# original bias std. error

# t1\* -1.154047e+01 -8.008379e-03 4.239273e-01

# t2\* 2.080898e-05 5.870933e-08 4.582525e-06

# t3\* 5.647103e-03 2.299970e-06 2.267955e-04

# As estimativas de bootstrap dos erros padrão para os

# coeficientes beta 0, beta 1 e beta 2 são 0,4239, 4,583 e 2,268.

## D)Comment on the estimated standard errors obtained using the

# glm() function and using your bootstrap function.

# Os erros padrão estimados pelos dois métodos são muito próximos.

# 8) A)Generate a simulated data set as follows:

set.seed(1)

y = rnorm(100)

x = rnorm(100)

y = x - 2 \* x^2 + rnorm(100)

# In this data set, what is n and what is p? Write out the model

# used to generate the data in equation form.

# n = 100 e p = 2.

# O modelo usado foi Y= X-2X²+E.

## B)Create a scatterplot of X against Y . Comment on what you find.

plot(x, y)

# Os dados possuem um relacionamento curvo.

## C)Set a random seed, and then compute the LOOCV errors that

# result from fitting the following four models using least squares:

# I)Y = β0 + β1X + E

library(boot)

set.seed(1)

Data = data.frame(x, y)

reg.glm = glm(y ~ x)

cv.glm(Data, reg.glm)$delta[1]

# 5.890979

# II)Y = β0 + β1X + β2X2 + E

reg1.glm <- glm(y ~ poly(x, 2))

cv.glm(Data, reg1.glm)$delta[1]

# 1.086596

# III)Y = β0 + β1X + β2X2 + β3X3 + E

reg2.glm <- glm(y ~ poly(x, 3))

cv.glm(Data, reg2.glm)$delta[1]

# 1.102585

# IV)Y= β0 + β1X + β2X2 + β3X3 + β4X4 + .

reg3.glm <- glm(y ~ poly(x, 4))

cv.glm(Data, reg3.glm)$delta[1]

# 1.114772

## D)Repeat (c) using another random seed, and report your results.

# Are your results the same as what you got in (c)? Why?

set.seed(10)

reg.glm = glm(y ~ x)

cv.glm(Data, reg.glm)$delta[1]

# 5.890979

reg1.glm = glm(y ~ poly(x, 2))

cv.glm(Data, reg1.glm)$delta[1]

# 1.086596

reg2.glm = glm(y ~ poly(x, 3))

cv.glm(Data, reg2.glm)$delta[1]

# 1.102585

reg3.glm = glm(y ~ poly(x, 4))

cv.glm(Data, reg3.glm)$delta[1]

# 1.114772

# Os resultados obtidos foram exatamente iguais aos de C.

## E)Which of the models in (c) had the smallest LOOCV error? Is

# this what you expected? Explain your answer.

# reg1.glm. Esperava pois a relação entre x e y é quadrática.

## F)Comment on the statistical significance of the coefficient estimates

# that results from fitting each of the models in (c) using

# least squares. Do these results agree with the conclusions drawn

# based on the cross-validation results?

summary(reg3.glm)

# Call:

# glm(formula = y ~ poly(x, 4))

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.8914 -0.5244 0.0749 0.5932 2.7796

# Coefficients:

# Estimate Std. Error t value Pr(>|t|)

# (Intercept) -1.8277 0.1041 -17.549 <2e-16 \*\*\*

# poly(x, 4)1 2.3164 1.0415 2.224 0.0285 \*

# poly(x, 4)2 -21.0586 1.0415 -20.220 <2e-16 \*\*\*

# poly(x, 4)3 -0.3048 1.0415 -0.293 0.7704

# poly(x, 4)4 -0.4926 1.0415 -0.473 0.6373

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# (Dispersion parameter for gaussian family taken to be 1.084654)

# Null deviance: 552.21 on 99 degrees of freedom

# Residual deviance: 103.04 on 95 degrees of freedom

# AIC: 298.78

# Number of Fisher Scoring iterations: 2

# Os p valores mostram que os termos linear e quadrático

# são estatisticamente significativos e que os termos

# cúbico e de 4º não são.

# 9) A)Based on this data set, provide an estimate for the population

# mean of medv. Call this estimate ˆμ.

library(MASS)

attach(Boston)

mu.hat = mean(medv)

mu.hat

# 22.53281

## B) Provide an estimate of the standard error of ˆμ. Interpret this

# result.

# Hint: We can compute the standard error of the sample mean by

# dividing the sample standard deviation by the square root of the

# number of observations.

se.hat = sd(medv) / sqrt(dim(Boston)[1])

se.hat

# 0.4088611

## C)Now estimate the standard error of ˆμ using the bootstrap. How

# does this compare to your answer from (b)?

set.seed(1)

boot.fn = function(data, index) {

mu = mean(data[index])

return (mu)

}

boot(medv, boot.fn, 1000)

# ORDINARY NONPARAMETRIC BOOTSTRAP

# Call:

# boot(data = medv, statistic = boot.fn, R = 1000)

# Bootstrap Statistics :

# original bias std. error

# t1\* 22.53281 0.008517589 0.4119374

# O erro padrão estimado com bootstrap foi muito

# próximo do encontrado em B.

## D)Based on your bootstrap estimate from (c), provide a 95% confidence

# interval for the mean of medv. Compare it to the results

# obtained using t.test(Boston$medv).

# Hint: You can approximate a 95% confidence interval using the

# formula [ˆμ − 2SE(ˆμ), ˆμ + 2SE(ˆμ)].

t.test(medv)

# One Sample t-test

# data: medv

# t = 55.111, df = 505, p-value < 2.2e-16

# alternative hypothesis: true mean is not equal to 0

# 95 percent confidence interval:

# 21.72953 23.33608

# sample estimates:

# mean of x

# 22.53281

CI.mu.hat = c(22.53 - 1.96 \* 0.4119, 22.53 + 1.96 \* 0.4119)

CI.mu.hat

# 21.72268 23.33732

# O intervalo de confiança é próximo ao de t.test().

## E)Based on this data set, provide an estimate, ˆμmed, for the median

# value of medv in the population.

med.hat = median(medv)

med.hat

# 21.2

## F)We now would like to estimate the standard error of ˆμmed. Unfortunately,

# there is no simple formula for computing the standard

# error of the median. Instead, estimate the standard error of the

# median using the bootstrap. Comment on your findings.

boot.fn = function(data, index) {

mu = median(data[index])

return (mu)

}

boot(medv, boot.fn, 1000)

# ORDINARY NONPARAMETRIC BOOTSTRAP

# Call:

# boot(data = medv, statistic = boot.fn, R = 1000)

# Bootstrap Statistics :

# original bias std. error

# t1\* 21.2 -0.0098 0.3874004

# Foi obtido um valor médio de 21,2 e um erro padrão de 0,3874.

## G)Based on this data set, provide an estimate for the tenth percentile

# of medv in Boston suburbs. Call this quantity ˆμ0.1. (You can use the quantile() function.)

percent.hat = quantile(medv, c(0.1))

percent.hat

# 10%

# 12.75

## H)Use the bootstrap to estimate the standard error of ˆμ0.1. Comment

# on your findings.

boot.fn = function(data, index) {

mu = quantile(data[index], c(0.1))

return (mu)

}

boot(medv, boot.fn, 1000)

# ORDINARY NONPARAMETRIC BOOTSTRAP

# Call:

# boot(data = medv, statistic = boot.fn, R = 1000)

# Bootstrap Statistics :

# original bias std. error

# t1\* 12.75 0.00515 0.5113487

# O valor percentil estimado foi de 12,75 e teve

# um erro padrão de 0,5113.