Laporan Praktikum 5 AMP

Antonius Aditya Rizky Wijaya

G5402221003

2025-02-20

# Cross-Validation and the Bootstrap

## The Validation Set Approach

library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.3

set.seed(1)  
train <- sample(392, 196)

Penjelasan: Training set dipilih secara acak sebanyak 196 observasi dari total 392 data.

lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)

Penjelasan: Model regresi linear sederhana dibuat dengan horsepower sebagai prediktor untuk memprediksi mpg.

attach(Auto)  
mean((mpg - predict(lm.fit, Auto))[-train]^2)

## [1] 23.26601

Penjelasan: MSE dihitung untuk test set, yang mengukur seberapa baik model memprediksi mpg untuk data yang belum terlihat.

lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto,   
 subset = train)  
mean((mpg - predict(lm.fit2, Auto))[-train]^2)

## [1] 18.71646

lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,   
 subset = train)  
mean((mpg - predict(lm.fit3, Auto))[-train]^2)

## [1] 18.79401

Penjelasan: Model polinomial mungkin memberikan MSE lebih kecil dibanding model linear, artinya model lebih cocok menangkap pola data. Perbandingan MSE antar model bisa membantu memilih model terbaik.

set.seed(2)  
train <- sample(392, 196)  
lm.fit <- lm(mpg ~ horsepower, subset = train)  
mean((mpg - predict(lm.fit, Auto))[-train]^2)

## [1] 25.72651

lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto,   
 subset = train)  
mean((mpg - predict(lm.fit2, Auto))[-train]^2)

## [1] 20.43036

lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto,   
 subset = train)  
mean((mpg - predict(lm.fit3, Auto))[-train]^2)

## [1] 20.38533

Penjelasan:

Hasilnya bisa berbeda karena pemilihan training set yang berbeda. Jika MSE polinomial tetap lebih rendah dari linear, ini memperkuat bahwa regresi polinomial lebih baik dalam menangkap pola hubungan antara horsepower dan mpg. Variasi MSE antar pemilihan training set menunjukkan bahwa model bisa dipengaruhi oleh pemilihan data training secara acak.

## Leave-One-Out Cross-Validation

glm.fit <- glm(mpg ~ horsepower, data = Auto)  
coef(glm.fit)

## (Intercept) horsepower   
## 39.9358610 -0.1578447

Penjelasan:

Model regresi linear dibuat untuk memprediksi mpg berdasarkan horsepower, dan hasilnya sama dengan model dari lm().

lm.fit <- lm(mpg ~ horsepower, data = Auto)  
coef(lm.fit)

## (Intercept) horsepower   
## 39.9358610 -0.1578447

Penjelasan:

Menggunakan glm() atau lm() untuk regresi linear tidak membuat perbedaan signifikan dalam konteks ini.

library(boot)  
glm.fit <- glm(mpg ~ horsepower, data = Auto)  
cv.err <- cv.glm(Auto, glm.fit)  
cv.err$delta

## [1] 24.23151 24.23114

Penjelasan:

LOOCV digunakan untuk mengukur akurasi model regresi. Hasilnya adalah MSE estimasi model pada data baru.

cv.error <- rep(0, 10)  
for (i in 1:10) {  
 glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)  
 cv.error[i] <- cv.glm(Auto, glm.fit)$delta[1]  
}  
cv.error

## [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115  
## [9] 19.06863 19.49093

Penjelasan:

LOOCV digunakan untuk membandingkan model polinomial dari derajat 1 hingga 10. Jika MSE semakin kecil seiring bertambahnya derajat, model polinomial lebih baik dari model linear.

## -Fold Cross-Validation

set.seed(17)  
cv.error.10 <- rep(0, 10)  
for (i in 1:10) {  
 glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)  
 cv.error.10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]  
}  
cv.error.10

## [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19.14666  
## [9] 18.87013 20.95520

Penjelasan:

10-fold CV lebih efisien daripada LOOCV karena tidak perlu membangun model sebanyak jumlah data. Hasil MSE dibandingkan untuk menemukan model polinomial terbaik. Jika tren mirip dengan LOOCV, berarti model polinomial lebih cocok untuk data ini.

## The Bootstrap

### Estimating the Accuracy of a Statistic of Interest

alpha.fn <- function(data, index) {  
 X <- data$X[index]  
 Y <- data$Y[index]  
 (var(Y) - cov(X, Y)) / (var(X) + var(Y) - 2 \* cov(X, Y))  
}

Penjelasan:

Fungsi ini akan digunakan untuk melakukan bootstrap estimasi alpha dengan berbagai sampel dari dataset Portfolio.

alpha.fn(Portfolio, 1:100)

## [1] 0.5758321

Penjelasan:

Nilai alpha dihitung tanpa resampling, sehingga ini menjadi dasar perbandingan dengan metode bootstrap nanti.

set.seed(7)  
alpha.fn(Portfolio, sample(100, 100, replace = T))

## [1] 0.5385326

Penjelasan:

Bootstrap memungkinkan kita memperkirakan variabilitas alpha berdasarkan sampel yang berbeda-beda.

boot(Portfolio, alpha.fn, R = 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 0.5758321 0.0007959475 0.08969074

Penjelasan:

Bootstrap digunakan untuk mengestimasi distribusi alpha, termasuk standar errornya.

### Estimating the Accuracy of a Linear Regression Model

boot.fn <- function(data, index)  
 coef(lm(mpg ~ horsepower, data = data, subset = index))  
boot.fn(Auto, 1:392)

## (Intercept) horsepower   
## 39.9358610 -0.1578447

Penjelasan:

Fungsi ini akan digunakan untuk melakukan bootstrap estimasi koefisien regresi linear.

set.seed(1)  
boot.fn(Auto, sample(392, 392, replace = T))

## (Intercept) horsepower   
## 40.3404517 -0.1634868

boot.fn(Auto, sample(392, 392, replace = T))

## (Intercept) horsepower   
## 40.1186906 -0.1577063

Penjelasan:

Bootstrap digunakan untuk memahami variabilitas koefisien regresi. Hasil koefisien bisa berbeda tiap iterasi, menunjukkan distribusi kemungkinan nilai koefisien.

boot(Auto, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Auto, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 39.9358610 0.0544513229 0.841289790  
## t2\* -0.1578447 -0.0006170901 0.007343073

Penjelasan:

Bootstrap memberikan distribusi yang lebih baik untuk koefisien regresi dibandingkan metode konvensional.

summary(lm(mpg ~ horsepower, data = Auto))$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 39.9358610 0.717498656 55.65984 1.220362e-187  
## horsepower -0.1578447 0.006445501 -24.48914 7.031989e-81

Penjelasan:

Bootstrap memungkinkan perbandingan antara estimasi standar regresi dan distribusi bootstrap untuk mengukur ketidakpastian estimasi.

boot.fn <- function(data, index)  
 coef(  
 lm(mpg ~ horsepower + I(horsepower^2),   
 data = data, subset = index)  
 )  
set.seed(1)  
boot(Auto, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Auto, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 56.900099702 3.511640e-02 2.0300222526  
## t2\* -0.466189630 -7.080834e-04 0.0324241984  
## t3\* 0.001230536 2.840324e-06 0.0001172164

summary(  
 lm(mpg ~ horsepower + I(horsepower^2), data = Auto)  
 )$coef

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 56.900099702 1.8004268063 31.60367 1.740911e-109  
## horsepower -0.466189630 0.0311246171 -14.97816 2.289429e-40  
## I(horsepower^2) 0.001230536 0.0001220759 10.08009 2.196340e-21

Penjelasan:

Bootstrap memungkinkan estimasi lebih akurat untuk regresi polinomial dengan memperhitungkan ketidakpastian lebih baik dibandingkan metode konvensional. Jika hasil bootstrap mirip dengan regresi standar, berarti model stabil. Jika berbeda, berarti ada ketidakpastian tinggi dalam estimasi koefisien.

# Exercise

# *Nomor 5*

In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

1. Fit a logistic regression model that uses income and balance to predict default.

library(ISLR2)  
defult\_model <- glm(default ~ income + balance, data = Default, family = "binomial")

1. Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:
   1. Split the sample set into a training set and a validation set.
   2. Fit a multiple logistic regression model using only the training observations.
   3. 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.
   4. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

set.seed(2)  
# i. Split the sample  
train\_index <- sample(nrow(Default), nrow(Default)/2)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
  
# ii. Fit a multiple logistic regression model  
model\_train <- glm(default ~ income + balance, data = train\_data, family = "binomial")  
  
# iii. Obtain a prediction of default status and classifying for each individual in the validation set  
prob\_predict <- predict(model\_train, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
  
# iv. Compute the validation set error  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 4819 101  
## Yes 18 62

(mean(class\_predict != valid\_data$default))

## [1] 0.0238

1. 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.

# Pengulangan pertama  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.8)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_d <- glm(default ~ income + balance, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_d, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 1929 33  
## Yes 9 29

(mean(class\_predict != valid\_data$default))

## [1] 0.021

# Pengulangan kedua  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.7)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_d <- glm(default ~ income + balance, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_d, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 2894 53  
## Yes 11 42

(mean(class\_predict != valid\_data$default))

## [1] 0.02133333

# Pengulangan ketiga  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.4)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_d <- glm(default ~ income + balance, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_d, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 5782 130  
## Yes 19 69

(mean(class\_predict != valid\_data$default))

## [1] 0.02483333

Dengan menggunakan set.seed(2), proporsi menghasilkan validation error yang lebih kecil (%) dibanding proporsi (%) dan proporsi (%), bahkan proporsi pada soal (b) (%).

Dari sini saya bisa menyimpulkan:

1. Jika proporsi training set lebih besar, akan memungkinkan model lebih akurat karena lebih banyak data untuk di train. Namun, validation set yang terlalu kecil dapat membuat validation error menjadi kurang stabil akibat kurangnya data untuk divalidasi.
2. Jika proporsi training set lebih kecil, akan membuat model kurang stabil (underfitting) karena data untuk di-train-nya terbatas. Namun, validation set yang lebih besar akan membuat validation error menjadi lebih representatif terhadap populasi (distribusi datanya lebih mendekati distribusi populasi).

Bagaimana menentukan proporsi data yang tepat?

1. Training data > validation data (umum digunakan), dapat digunakan jika dataset kecil atau model yang kompleks. Karena butuh lebih banyak data untuk di train agar model lebih akurat, selain itu sedikit persentase validation data untuk dataset yang kecil sudah cukup untuk merepresentasikan populasi.
2. Training data < validation data, dapat digunakan jika dataset besar (misal 1 juta observasi) atau variasi data yang kompleks. Karena butuh lebih banyak data untuk validasi agar hasil evaluasi model lebih akurat, selain itu sedikit persentase training data untuk dataset yang besar sudah cukup besar untuk di train.
3. Bisa menggunakan tujuan pemodelan sebagai alasan membagi data. Jika tujuannya untuk generalisasi model, gunakan training data lebih besar. Jika tujuannya untuk evaluasi model (gunakan validation data lebih besar).
4. 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.

# Pengulangan pertama  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.8)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_stu <- glm(default ~ income + balance + student, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_stu, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 1928 35  
## Yes 10 27

(mean(class\_predict != valid\_data$default))

## [1] 0.0225

# Pengulangan kedua  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.7)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_stu <- glm(default ~ income + balance + student, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_stu, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 2894 57  
## Yes 11 38

(mean(class\_predict != valid\_data$default))

## [1] 0.02266667

# Pengulangan ketiga  
set.seed(2)  
train\_index <- sample(nrow(Default), nrow(Default) \*0.4)   
train\_data <- Default[train\_index, ]  
valid\_data <- Default[-train\_index, ]  
model\_train\_stu <- glm(default ~ income + balance + student, data = train\_data, family = "binomial")  
prob\_predict <- predict(model\_train\_stu, valid\_data, type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Yes", "No")  
(table(class\_predict, valid\_data$default))

##   
## class\_predict No Yes  
## No 5784 139  
## Yes 17 60

(mean(class\_predict != valid\_data$default))

## [1] 0.026

Bisa dilihat untuk semua proporsi data, validation error-nya meningkat dibanding yang tanpa variabel dummy student, hal ini mengindikasikan bahwa student tidak relevan karena model menjadi lebih kompleks namun tidak memberikan manfaat tambahan (overfitting).

# *Nomor 6*

We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

1. 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.

library(ISLR2)  
set.seed(3)  
model\_6 <- glm(default ~ income + balance, data = Default, family = binomial)  
summary(model\_6)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = Default)  
##   
## 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

Didapat dengan standard error-nya dan dengan standard error-nya

1. 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) {  
 model <- glm(default ~ income + balance, data = data[index, ], family = 'binomial')  
 return(coef(model)[2:3])  
}

1. 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)  
set.seed(3)  
(hasil\_bootstrap <- boot(data = Default, statistic = boot.fn, R = 1000))

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Default, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 2.080898e-05 2.058964e-07 4.857844e-06  
## t2\* 5.647103e-03 1.968308e-05 2.310337e-04

1. Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Output dari summary(model\_6) pada bagian (a) memberikan standard error berdasarkan metode formula bawaan dari regresi logistik (glm()). Sementara output dari boot() pada bagian (c) memberikan standard error berdasarkan metode bootstrap, yang menghitung variasi.

Dengan metode bootstrap didapat dengan standard error-nya dan dengan standard error-nya .

Standard error berdasarkan metode bootstrap sedikit lebih besar dari standard error berdasarkan metode formula bawaan, hal ini karena metode bootstrap lebih sensitif terhadap variasi data.

# *Nomor 7*

In Sections 5.3.2 and 5.3.3, we saw that the cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

1. Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

library(ISLR2)  
data(Weekly)  
model\_7a <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = 'binomial')  
summary(model\_7a)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22122 0.06147 3.599 0.000319 \*\*\*  
## Lag1 -0.03872 0.02622 -1.477 0.139672   
## Lag2 0.06025 0.02655 2.270 0.023232 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1496.2 on 1088 degrees of freedom  
## Residual deviance: 1488.2 on 1086 degrees of freedom  
## AIC: 1494.2  
##   
## Number of Fisher Scoring iterations: 4

1. Fit a logistic regression model that predicts Direction using Lag1 and Lag2 *using all but the first observation*.

model\_7b <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = 'binomial')  
summary(model\_7b)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly[-1,   
## ])  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22324 0.06150 3.630 0.000283 \*\*\*  
## Lag1 -0.03843 0.02622 -1.466 0.142683   
## Lag2 0.06085 0.02656 2.291 0.021971 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1494.6 on 1087 degrees of freedom  
## Residual deviance: 1486.5 on 1085 degrees of freedom  
## AIC: 1492.5  
##   
## Number of Fisher Scoring iterations: 4

1. Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if Direction="Up" | Lag1 , Lag2. Was this observation correctly classified?

prob\_predict <- predict(model\_7b, Weekly[1,], type = "response")  
class\_predict <- ifelse(prob\_predict > 0.5, "Up", "Down")  
  
cat("Prediksi:\n")

## Prediksi:

class\_predict

## 1   
## "Up"

cat("\nYang benar:\n")

##   
## Yang benar:

Weekly$Direction[1]

## [1] Down  
## Levels: Down Up

Prediksinya salah, data Direction pada observasi pertama seharusnya “Down”, namun hasil prediksinya “Up”.

1. Write a for loop from to , where is the number of observations in the data set, that performs each of the following steps:
   1. Fit a logistic regression model using all but the th observation to predict Direction using Lag1 and Lag2 .
   2. Compute the posterior probability of the market moving up for the th observation.
   3. Use the posterior probability for the th observation in order to predict whether or not the market moves up.
   4. Determine whether or not an error was made in predicting the direction for the th observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

error <- numeric(nrow(Weekly))  
for (i in 1:nrow(Weekly)) {  
 model <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = "binomial")  
 pred <- predict(model, Weekly[i, ], type = "response")  
 class <- ifelse(pred > 0.5, "Up", "Down")  
 error[i] <- ifelse(class != Weekly$Direction[i], 1, 0)  
}

1. Take the average of the numbers obtained in (d) in order to obtain the LOOCV estimate for the test error. Comment on the results.

loocv\_error <- mean(error)  
cat("LOOCV Test Error:", loocv\_error, "\n")

## LOOCV Test Error: 0.4499541

LOOCV Test Error sekitar % yang mengindikasikan bahwa prediksi mungkin akan sedikit lebih sering benar daripada salah, tapi memang tidak menutup fakta bahwa kemungkinan salahnya pun besar.

# *Nomor 8*

We will now perform cross-validation on a simulated data set.

1. Generate a simulated data set. In this data set, what is and what is ? Write out the model used to generate the data in equation form.

set.seed(1)  
x <- rnorm(100)  
y <- x - 2 \* x^2 + rnorm(100)  
n <- length(x) # n = 100  
p <- 1  
cat("Jumlah observasi (n):", n, "\n")

## Jumlah observasi (n): 100

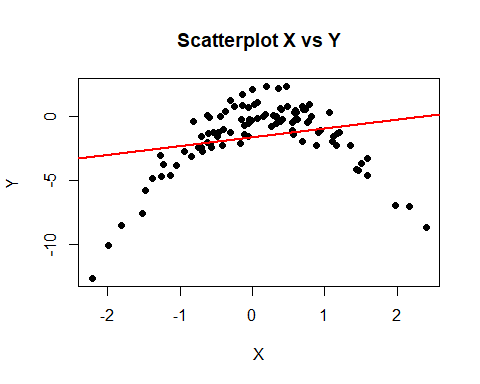
cat("Jumlah prediktor (p):", p, "\n")

## Jumlah prediktor (p): 1

Model matematis: , di mana ~ N(0, 1)

1. Create a scatterplot of against . Comment on what you find.

plot(x, y, main = "Scatterplot X vs Y", xlab = "X", ylab = "Y", pch = 16, col = "black")  
abline(lm(y ~ x), col = "red", lwd = 2)



Scatterplot menunjukkan hubungan non-linear antara x dan y. Secara visual, terlihat bahwa y memiliki pola kuadratik terhadap x.

1. Set a random seed, and then compute the Leave-One-Out Cross-Validation (LOOCV) errors that result from fitting the following four models using least squares:
   1. .

Note you may find it helpful to use the data.frame() function to create a single data set containing both and .

data <- data.frame(x = x, y = y)  
  
loocv\_error <- function(degree) {  
 errors <- rep(0, n)  
 for (i in 1:n) {  
 train\_data <- data[-i, ]  
 test\_data <- data[i, ]  
 model <- lm(y ~ poly(x, degree), data = train\_data)  
 pred <- predict(model, newdata = test\_data)  
 errors[i] <- (pred - test\_data$y)^2  
 }  
 return(mean(errors))  
}  
  
set.seed(1)  
loocv\_1 <- loocv\_error(1)   
loocv\_2 <- loocv\_error(2)  
loocv\_3 <- loocv\_error(3)  
loocv\_4 <- loocv\_error(4)  
  
cat("LOOCV Error Linear:", loocv\_1, "\n")

## LOOCV Error Linear: 7.288162

cat("LOOCV Error Kuadratik:", loocv\_2, "\n")

## LOOCV Error Kuadratik: 0.9374236

cat("LOOCV Error Kubi:", loocv\_3, "\n")

## LOOCV Error Kubi: 0.9566218

cat("LOOCV Error Kuartik:", loocv\_4, "\n")

## LOOCV Error Kuartik: 0.9539049

1. Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

set.seed(2)  
loocv\_1\_d <- loocv\_error(1)  
loocv\_2\_d <- loocv\_error(2)  
loocv\_3\_d <- loocv\_error(3)  
loocv\_4\_d <- loocv\_error(4)  
  
cat("Hasil LOOCV dengan seed berbeda:\n")

## Hasil LOOCV dengan seed berbeda:

cat("LOOCV Error Linear:", loocv\_1, "\n")

## LOOCV Error Linear: 7.288162

cat("LOOCV Error Kuadratik:", loocv\_2, "\n")

## LOOCV Error Kuadratik: 0.9374236

cat("LOOCV Error Kubik:", loocv\_3, "\n")

## LOOCV Error Kubik: 0.9566218

cat("LOOCV Error Kuartik:", loocv\_4, "\n")

## LOOCV Error Kuartik: 0.9539049

Hasil LOOCV error kemungkinan sama, karena LOOCV tidak bergantung pada pengacakan subset seperti metode k-fold.

1. Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

Model linear (derajat 1) memiliki error lebih tinggi karena tidak mampu menggambarkan hubungan non-linear yang jelas dalam data.

Model kubik (derajat 3) dan kuartik (derajat 4) dapat memiliki error yang lebih rendah dibandingkan model linear, tetapi mungkin tidak jauh lebih baik dari model kuadratik. Sehingga model kuadratik (derajat 2) memiliki LOOCV error terkecil.

Ini sesuai ekspektasi, karena model yang sebenarnya digunakan untuk mensimulasikan data adalah kuadratik.

1. 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?

model\_8f\_1 <- lm(y ~ poly(x, 1), data = data)   
model\_8f\_2 <- lm(y ~ poly(x, 2), data = data)   
model\_8f\_3 <- lm(y ~ poly(x, 3), data = data)   
model\_8f\_4 <- lm(y ~ poly(x, 4), data = data)  
  
summary(model\_8f\_1)

##   
## Call:  
## lm(formula = y ~ poly(x, 1), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.5161 -0.6800 0.6812 1.5491 3.8183   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.550 0.260 -5.961 3.95e-08 \*\*\*  
## poly(x, 1) 6.189 2.600 2.380 0.0192 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.6 on 98 degrees of freedom  
## Multiple R-squared: 0.05465, Adjusted R-squared: 0.045   
## F-statistic: 5.665 on 1 and 98 DF, p-value: 0.01924

summary(model\_8f\_2)

##   
## Call:  
## lm(formula = y ~ poly(x, 2), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.9650 -0.6254 -0.1288 0.5803 2.2700   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.5500 0.0958 -16.18 < 2e-16 \*\*\*  
## poly(x, 2)1 6.1888 0.9580 6.46 4.18e-09 \*\*\*  
## poly(x, 2)2 -23.9483 0.9580 -25.00 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.958 on 97 degrees of freedom  
## Multiple R-squared: 0.873, Adjusted R-squared: 0.8704   
## F-statistic: 333.3 on 2 and 97 DF, p-value: < 2.2e-16

summary(model\_8f\_3)

##   
## Call:  
## lm(formula = y ~ poly(x, 3), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.9765 -0.6302 -0.1227 0.5545 2.2843   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.55002 0.09626 -16.102 < 2e-16 \*\*\*  
## poly(x, 3)1 6.18883 0.96263 6.429 4.97e-09 \*\*\*  
## poly(x, 3)2 -23.94830 0.96263 -24.878 < 2e-16 \*\*\*  
## poly(x, 3)3 0.26411 0.96263 0.274 0.784   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9626 on 96 degrees of freedom  
## Multiple R-squared: 0.8731, Adjusted R-squared: 0.8691   
## F-statistic: 220.1 on 3 and 96 DF, p-value: < 2.2e-16

summary(model\_8f\_4)

##   
## Call:  
## lm(formula = y ~ poly(x, 4), data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.0550 -0.6212 -0.1567 0.5952 2.2267   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.55002 0.09591 -16.162 < 2e-16 \*\*\*  
## poly(x, 4)1 6.18883 0.95905 6.453 4.59e-09 \*\*\*  
## poly(x, 4)2 -23.94830 0.95905 -24.971 < 2e-16 \*\*\*  
## poly(x, 4)3 0.26411 0.95905 0.275 0.784   
## poly(x, 4)4 1.25710 0.95905 1.311 0.193   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9591 on 95 degrees of freedom  
## Multiple R-squared: 0.8753, Adjusted R-squared: 0.8701   
## F-statistic: 166.7 on 4 and 95 DF, p-value: < 2.2e-16

Koefisien Signifikan (p-value < 0.05): Model Linear: signifikan namun kurang kuat Model Kuadratik: semua koefisien sangat signifikan Model Kubik: ada koefisien yang tidak signifikan Model Kuartik: ada koefisien yang tidak signifikan

Jika digabungkan dengan hasil LOOCV Error setiap model, dapat disimpulkan bahwa model kuadratik kemungkinan besar optimal, dengan koefisien yang signifikan dan LOOCV error yang rendah. Meski model kubik dan kuartik memiliki LOOCV error yang cukup rendah juga, namun dua model ini memiliki koefisien yang tidak signifikan, sehingga menunjukkan bahwa terjadi overfitting.

# *Nomor 9*

We will now consider the Boston housing data set, from the ISLR2 library.

1. Based on this data set, provide an estimate for the population mean of medv. Call this estimate .

library(ISLR2)  
data("Boston")  
mean\_medv <- mean(Boston$medv)  
cat("Rata-rata populasi (µ̂):", mean\_medv, "\n")

## Rata-rata populasi (µ̂): 22.53281

1. Provide an estimate of the standard error of . 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\_mean <- sd(Boston$medv) / sqrt(length(Boston$medv))  
cat("SE mean:", se\_mean, "\n")

## SE mean: 0.4088611

1. Now estimate the standard error of using the bootstrap. How does this compare to your answer from (b)?

set.seed(4)  
bootstrap\_means <- replicate(1000, mean(sample(Boston$medv, length(Boston$medv), replace = TRUE)))  
se\_mean\_bootstrap <- sd(bootstrap\_means)  
cat("SE mean (Bootstrap):", se\_mean\_bootstrap, "\n")

## SE mean (Bootstrap): 0.4069406

Standard error menggunakan bootstrap adalah 0.4069406. Nilai ini sangat dekat dengan standard error yang didapat dari formula soal (b) yaitu 0.4088611. Standard error dari bootstrap biasanya lebih akurat karena memperhitungkan variasi distribusi data secara empiris.

1. 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*

ci\_bootstrap <- c(mean\_medv - 2 \* se\_mean\_bootstrap, mean\_medv + 2 \* se\_mean\_bootstrap)  
cat("95% CI (Bootstrap):", ci\_bootstrap, "\n")

## 95% CI (Bootstrap): 21.71893 23.34669

hasil\_t\_test <- t.test(Boston$medv)  
cat("95% CI (t-test):", hasil\_t\_test$conf.int, "\n")

## 95% CI (t-test): 21.72953 23.33608

1. Based on this data set, provide an estimate, , for the median value of medv in the population.

median\_medv <- median(Boston$medv)  
cat("Median populasi (µ̂med):", median\_medv, "\n")

## Median populasi (µ̂med): 21.2

1. We now would like to estimate the standard error of . 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.

set.seed(4)  
bootstrap\_medians <- replicate(1000, median(sample(Boston$medv, length(Boston$medv), replace = TRUE)))  
se\_median\_bootstrap <- sd(bootstrap\_medians)  
cat("SE Median (Bootstrap):", se\_median\_bootstrap, "\n")

## SE Median (Bootstrap): 0.3749919

Estimasi standard error median adalah 0.3749919, ini lebih kecil dibanding standard error mean (0.4069406).

1. Based on this data set, provide an estimate for the tenth percentile of medv in Boston census tracts. Call this quantity . (You can use the quantile() function.)

persentil\_10 <- quantile(Boston$medv, 0.1)  
cat("Persentil ke-10 (µ̂0.1):", persentil\_10, "\n")

## Persentil ke-10 (µ̂0.1): 12.75

1. Use the bootstrap to estimate the standard error of . Comment on your findings.

set.seed(4)  
bootstrap\_percentile\_10 <- replicate(1000, quantile(sample(Boston$medv, length(Boston$medv), replace = TRUE), 0.1))  
se\_percentile\_10 <- sd(bootstrap\_percentile\_10)  
cat("SE Percentile ke-10 (Bootstrap):", se\_percentile\_10, "\n")

## SE Percentile ke-10 (Bootstrap): 0.4900443

Kita dapatkan standard error = 0.4900443, ini lebih besar dibanding standard error median (0.3749919).