Compulsory exercise 1: Group 4

TMA4268 Statistical Learning V2021

Øystein Alvestad, Lars Evje and William Scott Grundeland Olsen

19 februar, 2021

Problem 1

We consider the regression problem

$$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon,$$

where $\mathrm{E}(\varepsilon) = 0$ and $\mathrm{Var}(\varepsilon) = \sigma^2$. We define $\mathbf{x}, \boldsymbol{\beta} \in \mathbb{R}^{p+1}$ such that $f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$.

a)

We consider the estimator

$$\tilde{\boldsymbol{\beta}} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{Y},$$

for $\boldsymbol{\beta}$. Here X is the design matrix, $\mathbf{Y}$ is the response vector, I is the identity matrix, and $\lambda \geq 0$ is a constant. We recall that for a constant matrix A of appropriate dimensions, and a random vector $\mathbf{Z}$, we have that

$$\mathrm{E}(A\mathbf{Z}) = A\,\mathrm{E}(\mathbf{Z}) \quad \text{and} \quad \mathrm{Var}(A\mathbf{Z}) = A\,\mathrm{E}(\mathbf{Z})A^\top. \tag{1}$$

First we now find the expected value

$$\mathrm{E}(\tilde{\boldsymbol{\beta}}) = \mathrm{E}((X^\top X + \lambda I)^{-1} X^\top \mathbf{Y}) = (X^\top X + \lambda I)^{-1} X^\top\,\mathrm{E}(\mathbf{Y}),$$

and because $\mathrm{E}(\mathbf{Y}) = X\boldsymbol{\beta}$, we get that

$$\mathrm{E}(\tilde{\boldsymbol{\beta}}) = (X^\top X + \lambda I)^{-1} X^\top X \boldsymbol{\beta}.$$

Similarly, the variance-covariance matrix is

$$\mathrm{Var}(\tilde{\boldsymbol{\beta}}) = \mathrm{Var}((X^\top X + \lambda I)^{-1} X^\top \mathbf{Y}) = (X^\top X + \lambda I)^{-1} X^\top\,\mathrm{Var}(\mathbf{Y})[(X^\top X + \lambda I)^{-1} X^\top]^\top,$$

and because $\mathrm{Var}(\mathbf{Y}) = \sigma^2 I$, we get that

$$\mathrm{Var}(\tilde{\boldsymbol{\beta}}) = \sigma^2 (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1},$$

where we used $(B^{-1})^\top = (B^\top)^{-1}$, for an invertible matrix B.

b)

We now let $\tilde{f}(\mathbf{x}_0) = \mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}$ be the prediction at a new covariate vector $\mathbf{x}_0$, and we wish the expected value and variance of this. Using what we learned in **a)** and Equation (1), we find that

$$\mathrm{E}(\tilde{f}(\mathbf{x}_0)) = \mathrm{E}(\mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top\,\mathrm{E}(\tilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top (X^\top X + \lambda I)^{-1} X^\top X \boldsymbol{\beta}.$$

Similarly, we find that

$$\mathrm{Var}(\tilde{f}(\mathbf{x}_0)) = \mathrm{Var}(\mathbf{x}_0^\top \tilde{\boldsymbol{\beta}}) = \mathbf{x}_0^\top\,\mathrm{Var}(\tilde{\boldsymbol{\beta}})\mathbf{x}_0 = \sigma^2 \mathbf{x}_0^\top (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1} \mathbf{x}_0. \tag{2}$$

c)

We want to find the expected MSE at $\mathbf{x}_0$, and this is most easily done using the relationship between the expected value and the variance-covariance matrix,

$$\mathrm{E}((y_0 - \tilde{f}(\mathbf{x}_0))^2) = \mathrm{E}(\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0))^2 + \mathrm{Var}(\tilde{f}(\mathbf{x}_0)) + \mathrm{Var}(\varepsilon).$$

The last two terms are known to us now, so we look further at the first term, the squared bias,

$$\begin{aligned} \mathrm{E}(\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0))^2 &= [\mathrm{E}(\tilde{f}(\mathbf{x}_0)) - \mathrm{E}(f(\mathbf{x}_0))]^2 = [\mathrm{E}(\tilde{f}(\mathbf{x}_0)) - f(\mathbf{x}_0)]^2 \\ &= [\mathbf{x}_0^\top (X^\top X + \lambda I)^{-1} X^\top X \boldsymbol{\beta} - \mathbf{x}_0^\top \boldsymbol{\beta}]^2. \end{aligned} \tag{3}$$

We then get that

$$\begin{aligned} \mathrm{E}((y_0 - \tilde{f}(\mathbf{x}_0))^2) = \sigma^2 &+ [\mathbf{x}_0^\top (X^\top X + \lambda I)^{-1} X^\top X \boldsymbol{\beta} - \mathbf{x}_0^\top \boldsymbol{\beta}]^2 \\ &+ \sigma^2 \mathbf{x}_0^\top (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1} \mathbf{x}_0. \end{aligned}$$

d)

We start by importing the relevant quantities, as given in the project description.

```
id <- "1X_8OKcoYbng1XvYFDirxjEWr7LtpNr1m"  # Google file ID
values <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))

X <- values$X
x0 <- values$x0
beta <- values$beta
sigma <- values$sigma
```

We may then calculate the squared bias $\mathrm{E}(\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0))^2$, using Equation (3). Plotting this for $\lambda \in [0, 2]$, we get the result as in Figure 1. We see that the squared bias increases with large $\lambda > 0.5$, and has a minimum when $\lambda = 0$ and $\lambda \approx 0.5$. This is expected for $\lambda = 0$, because we then get that $\tilde{\boldsymbol{\beta}}$ is equal to the OLS estimator. The bias measures how good the estimator is in estimating the real value, so this makes sense. From the figure it also appears that for $\lambda \approx 0.5$, the estimator $\tilde{\boldsymbol{\beta}}$ us estimating the real value $\boldsymbol{\beta}$ very good. For increasing λ after this, the estimator seems to do a worse job in estimating $\boldsymbol{\beta}$.

```
library(ggplot2)
bias <- function(lambda, X, x0, beta) {
  p <- ncol(X)
  inv <- solve(t(X) %*% X + lambda * diag(p))
  value <- (t(x0) %*% inv %*% t(X) %*% X %*% beta - t(x0) %*% beta)^2
  return(value)
}

lambdas <- seq(0, 2, length.out = 500)
BIAS <- rep(NA, length(lambdas))

for (i in 1:length(lambdas)) {
  BIAS[i] <- bias(lambdas[i], X, x0, beta)
}
dfBias <- data.frame(lambdas = lambdas, bias = BIAS)

ggplot(dfBias, aes(x = lambdas, y = bias)) +
  geom_line(color = "red") +
  xlab(expression(lambda)) +
  ylab(expression(bias^2))
```

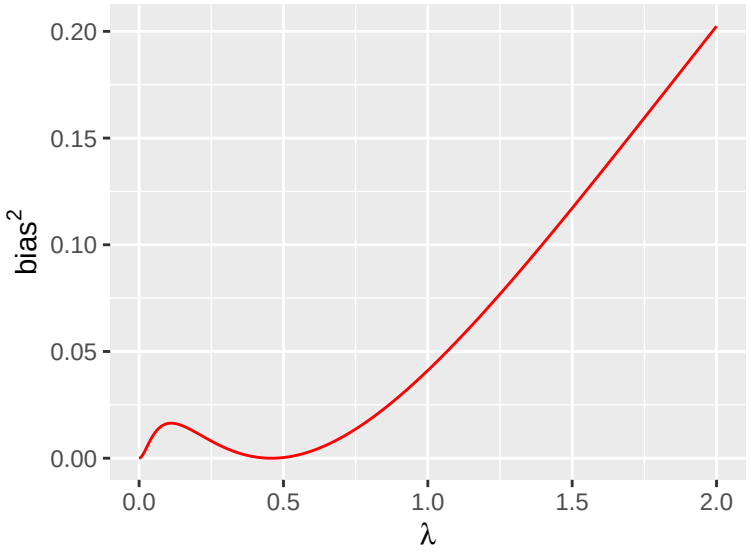


Figure 1: The squared bias $\mathrm{E}(\tilde{f}(\mathbf{x}_0) - f(\mathbf{x}_0))^2$ as a function of λ.

e)

We are now interested in $\mathrm{Var}(\tilde{f}(\mathbf{x}_0))$, which can be calculated using Equation (2). Using $\lambda \in [0, 2]$, the result is plotted in Figure 2. We notice that the variance is decreasing for increasing values of λ. That is, as long as λ increases, the model is less prone to overfitting. Letting $\lambda \to \infty$, we also see that the variance tends to zero.

```
variance <- function(lambda, X, x0, sigma) {
  p <- ncol(X)
  inv <- solve(t(X) %*% X + lambda * diag(p))
  value <- sigma^2 * t(x0) %*% inv %*% t(X) %*% X %*% inv %*% x0
  return(value)
}

lambdas <- seq(0, 2, length.out = 500)
VAR <- rep(NA, length(lambdas))

for (i in 1:length(lambdas)) {
  VAR[i] <- variance(lambdas[i], X, x0, sigma)
}
dfVar <- data.frame(lambdas = lambdas, var = VAR)

ggplot(dfVar, aes(x = lambdas, y = var)) +
  geom_line(color = "green4") +
  xlab(expression(lambda)) +
  ylab("variance") +
  theme(plot.title = element_text(hjust = 0.5))
```

f)

Lastly, we are interested in the expected MSE at $\mathbf{x}_0$, $\mathrm{E}((y_0 - \tilde{f}(\mathbf{x}_0))^2)$, which, when we know the squared bias and the variance from **e)**, is given as $(\text{bias})^2 + \mathrm{Var}(\tilde{f}(\mathbf{x}_0)) + \sigma^2$, because the irreducible error is σ^2. The plot of the expected MSE, the squared bias and the variance is shown in Figure 3, and by using `lambdas[which.min(exp_mse)]` we find that the minimal expected MSE is found when $\lambda \approx 0.993988$. That

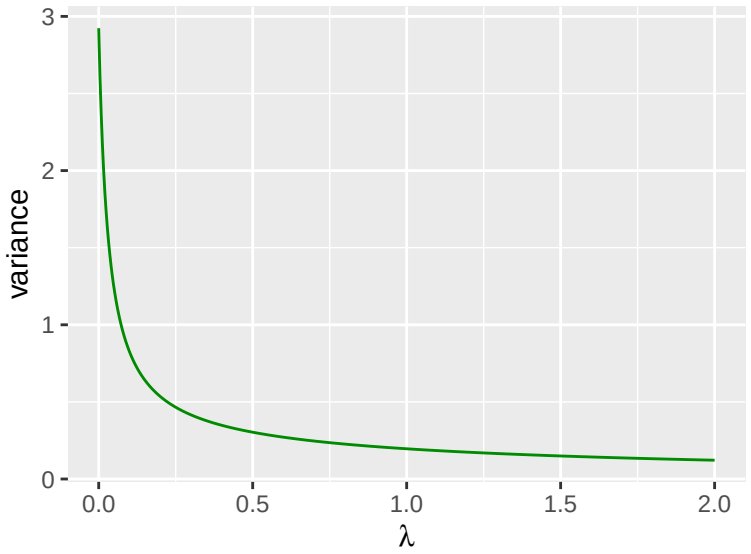


Figure 2: The variance $\text{Var}(\tilde{f}(\mathbf{x}_0))$ as a function of λ.

is, it is possbile to move to $\lambda > 0$, and taking a little bias, and reducing the variance, leading to a reduction in the expected MSE.

```
exp_mse <- BIAS + VAR + sigma^2

cols <- c("exp_mse" = "blue", "bias" = "red", "variance" = "green4")
dfAll <- data.frame(lambda = lambdas, bias = BIAS, var = VAR, exp_mse = exp_mse)

ggplot(dfAll) +
  geom_line(aes(x = lambda, y = exp_mse, color = "exp_mse")) +
  geom_line(aes(x = lambda, y = bias, color = "bias")) +
  geom_line(aes(x = lambda, y = var, color = "variance")) +
  xlab(expression(lambda)) +
  ylab(expression(E(MSE))) +
  theme(legend.title = element_blank()) +
  scale_color_manual(values = cols)
```

Problem 2

a)

b)

c)

d)

Problem 3

```
# read file
id <- "1i1cQPeoLLC_FyAHOnnqCnnrSBpn05_hO"  # google file ID
diab <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
```

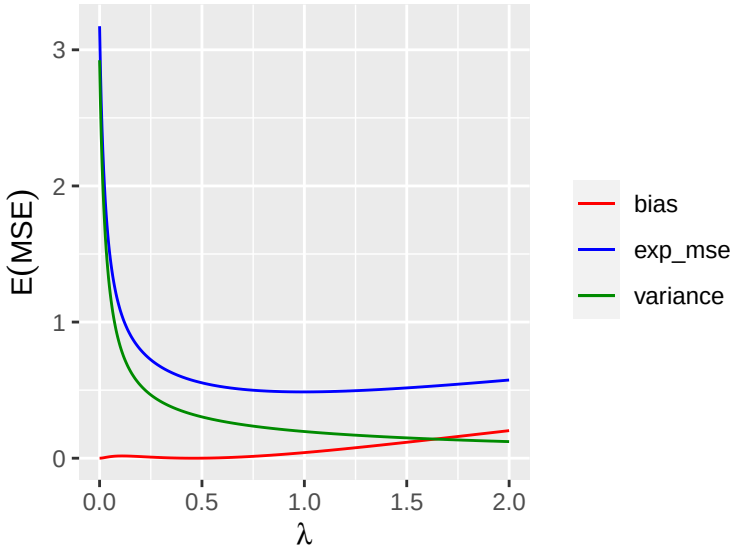


Figure 3: The expected MSE $\mathrm{E}((y_0 - \tilde{f}(\mathbf{x}_0))^2)$ as a function of λ, together with the squared bias and the variance.

```
t = MASS::Pima.tr2
train = diab$ctrain
test = diab$ctest
```

a)

```
logReg = glm(diabetes ~ ., data = train, family = "binomial")
summary(logReg)
```

```
##
## Call:
## glm(formula = diabetes ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8155  -0.6367  -0.3211   0.6147   2.2408
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.583538   1.428276  -7.410 1.26e-13 ***
## npreg         0.105109   0.062721   1.676 0.093775 .
## glu           0.035586   0.005892   6.039 1.55e-09 ***
## bp           -0.014654   0.013982  -1.048 0.294615
## skin          0.020379   0.020575   0.990 0.321962
## bmi           0.094683   0.031265   3.028 0.002458 **
## ped           1.931666   0.529573   3.648 0.000265 ***
## age           0.038291   0.020247   1.891 0.058594 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 381.91  on 299  degrees of freedom
## Residual deviance: 253.84  on 292  degrees of freedom
## AIC: 269.84
##
## Number of Fisher Scoring iterations: 5
```

i)

For simplicity we write $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_7 x_{i7} = \mathbf{x}^T\beta$, where $\beta = (\beta_0 \ \ldots \ \beta_7)^T$ and $\mathbf{x} = (1 \ x_1 \ \ldots \ x_7)^T$). Furthermore let $P(y_i = 1|\mathbf{X} = \mathbf{x}_i) = p_i$. Starting from the expression for p_i we get

$$
\begin{aligned}
p_i &= \frac{e^{\mathbf{x}^T\beta}}{1 + e^{\mathbf{x}^T\beta}} \\
p_i + p_i e^{\mathbf{x}^T\beta} &= e^{\mathbf{x}^T\beta} \\
p_i &= e^{\mathbf{x}^T\beta}(1 - p_i) \\
\frac{p_i}{1 - p_i} &= e^{\mathbf{x}^T\beta} \\
log(\frac{p_i}{1 - p_i}) &= \mathbf{x}^T\beta = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_7 x_{i7} \quad \blacksquare
\end{aligned}
$$

ii)

```
logReg.prob = predict(logReg, newdata = test, type = "response")
logReg.pred = ifelse(logReg.prob > 0.5, "1", "0")
c = table(predicted = logReg.pred, true = test$diabetes)

library(yardstick)
library(ggplot2)

cm = conf_mat(c)

autoplot(cm, type = "heatmap") + ggtitle("Confusion table Logistic Regression")
```

From the table above we find sensitivity $= \frac{48}{77} = 0.623$, and specificity $= \frac{137}{155} = 0.884$.

b)

(i) π_k is the prior probability for class k, i.e $Pr(Y = k)$, so for the diabetes problem, π_1 is the probability for a random person of having diabetes, while π_0 is the probability of not having diabetes. $\boldsymbol{\mu}_k$ is the mean vector for each class, so in this case μ_1 and μ_0 will be the mean values of each predictors for a correspondingly diabetic or non diabetic person. $\boldsymbol{\Sigma}$ is the covariance matrix between the predictors and is in lda assumed to be the same for all classes. $f_k(x)$ is the conditional distribution of the predictors of each class, and is assumed to be normal distribution, with mean $\boldsymbol{\mu_k}$ and covariance matrix $\boldsymbol{\Sigma}$.

(ii)

```
library(MASS)
lda.model = lda(diabetes ~ ., data = train)
qda.model = qda(diabetes ~ ., data = train)

lda.modelProb = predict(lda.model, newdata = test)$posterior
lda.modelPred = predict(lda.model, newdata = test)$class
table(predicted = lda.modelPred, true = test$diabetes)
```

```
##          true
## predicted   0   1
##         0 138  30
##         1  17  47
```

```
qda.modelProb = predict(qda.model, newdata = test)$posterior
qda.modelPred = predict(qda.model, newdata = test)$class
table(predicted = qda.modelPred, true = test$diabetes)
```

```
##          true
## predicted   0   1
##         0 131  32
##         1  24  45
```

```
# lda.model
```

The difference between LDA and QDA models, is that in LDA one assumes that the covariance matrices is the same for all classes ($\Sigma_k = \Sigma$). By cancellations this leads to a linear decision boundary. In QDA one does not assume that the covariance matrices is the same for all classes, and thus we do not obtain the same cancellation. This gives us a quadratic decision boundary.

c)

i) When classifying a new observation in the KNN approach, one first identifies the K nearest points in the training data, measured in the euclidean metric. Then we classify the observation to the majority class amongst those K nearest points, i.e. we assign the observation to the class with the highest proportion of members among the K points in question.

ii) The tuning parameter k can be chosen by looking at the test error for a range of different values of k.Then choose the value that gives the lowest error. Can often be useful to combine this with Cross-Validation.

iii)

```
library(class)
knnMod = knn(train = train, test = test, cl = train$diabetes, k = 25, prob = T)
table(knnMod, test$diabetes)
```

```
##
## knnMod   0   1
```

```
##       0 144  36
##       1  11  41
```

Sensitivity $= \frac{41}{77} = 0.532$, specificity $= \frac{144}{155} = 0.929$.

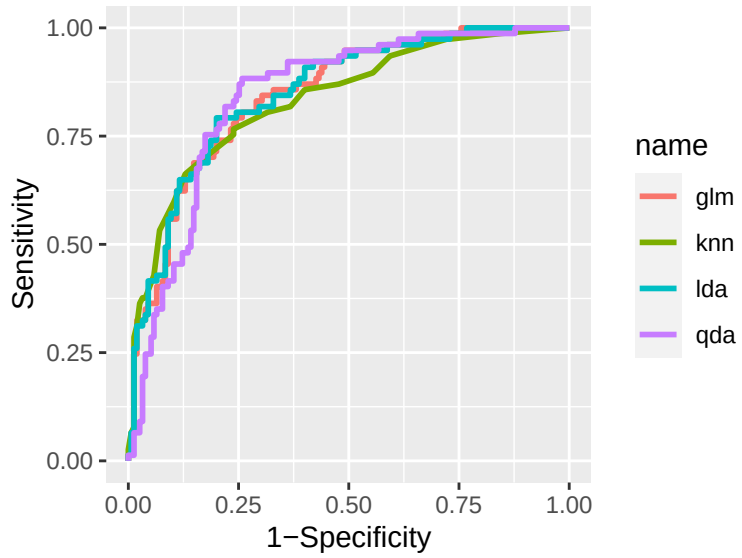
d)

```
knnModProb = attributes(knnMod)$prob
knnModProb = ifelse(knnMod == 0, 1 - knnModProb, knnModProb)


library(pROC)
library(plotROC)
library(ggplot2)

glmroc = roc(response = test$diabetes, predictor = logReg.prob)
ldaroc = roc(response = test$diabetes, predictor = lda.modelProb[, 2])
qdaroc = roc(response = test$diabetes, predictor = qda.modelProb[, 2])
knnroc = roc(response = test$diabetes, predictor = knnModProb)

dat = data.frame(diabetes = test$diabetes, glm = logReg.prob, lda = lda.modelProb[,
    2], qda = qda.modelProb[, 2], knn = knnModProb)
dat_long = melt_roc(dat, "diabetes", c("glm", "lda", "qda", "knn"))
ggplot(dat_long, aes(d = D, m = M, color = name)) + geom_roc(n.cuts = F) + xlab("1-Specificity") +
    ylab("Sensitivity")
```




```
auc(glmroc)
```

```
## Area under the curve: 0.8451
```

```
auc(ldaroc)
```

```
## Area under the curve: 0.849
```

```
auc(qdaroc)
```

```
## Area under the curve: 0.8415
```

```
auc(knnroc)
```

```
## Area under the curve: 0.8334
```

We observe that the LDA model has the highest AUC value, 0.849, followed by the Logistic Regression model, AUC = 0.8451. Thus measured in the AUC metric, we would say that LDA performs the best. Since the difference in AUC between Logistic Regression and LDA is relatively small, we would prefer Logistic Regression for interpretability?

Problem 4

a)

We wish to show that for the linear regression model $Y = X\beta + \varepsilon$, the LOOCV statistic is given by

$$\text{CV} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1-h_i}\right)^2,$$

where $h_i = \mathbf{x}_i^\top (X^\top X)^{-1}\mathbf{x}_i$, and $\mathbf{x}_i^\top$ is the i-th row of X.

Generally we can write $\text{CV} = \sum_{i=1}^{N} e_{(-i)}^2/N$, where $e_{(-i)} = y_i - \hat{y}_{(-i)}$. We use the notation $A_{(-i)}$ to symbolize that element i is removed from A if it is a vector, and row i is removed from A if it is a matrix. For a linear regression model $\mathbf{Y} = X\boldsymbol{\beta} + \varepsilon$, the estimate of $\boldsymbol{\beta}$ without the i-th case is

$$\hat{\boldsymbol{\beta}}_{(-i)} = (X_{(-i)}^\top X_{(-i)})^{-1} X_{(-i)}^\top \mathbf{Y}_{(-i)}.$$

From what we then know, we may write $e_{(-i)} = y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}_{(-i)}$. We want another expression for $\hat{\boldsymbol{\beta}}_{(-i)}$, and using the Sherman-Morrison formula,

$$(X_{(-i)}^\top X_{(-i)})^{-1} = (X^\top X - \mathbf{x}_i\mathbf{x}_i^\top)^{-1} = (X^\top X)^{-1} + \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top (X^\top X)^{-1}}{1 - \mathbf{x}_i(X^\top X)^{-1}\mathbf{x}_i^\top}.$$

By the definition of h_i, we then get that

$$(X_{(-i)}^\top X_{(-i)})^{-1} = (X^\top X)^{-1} + \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top (X^\top X)^{-1}}{1 - h_i}.$$

It is also clear that $X_{(-i)}^\top \mathbf{Y}_{(-i)} = X^\top \mathbf{Y} - \mathbf{x}_i y_i$, and thus

$$\hat{\boldsymbol{\beta}}_{(-i)} = (X_{(-i)}^\top X_{(-i)})^{-1} X_{(-i)}^\top \mathbf{Y}_{(-i)} = \left[(X^\top X)^{-1} + \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top (X^\top X)^{-1}}{1-h_i}\right](X^\top \mathbf{Y} - \mathbf{x}_i y_i).$$

Multiplying out this expression we then get

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{(-i)} &= (X^\top X)^{-1}X^\top \mathbf{Y} + \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top (X^\top X)^{-1}}{1-h_i} X^\top \mathbf{Y} - (X^\top X)^{-1}\mathbf{x}_i y_i - \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top (X^\top X)^{-1}}{1-h_i}\mathbf{x}_i y_i \\
&= \hat{\boldsymbol{\beta}} + \frac{(X^\top X)^{-1}\mathbf{x}_i\mathbf{x}_i^\top \hat{\boldsymbol{\beta}}}{1-h_i} - (X^\top X)^{-1}\mathbf{x}_i y_i - \frac{(X^\top X)^{-1}\mathbf{x}_i h_i}{1-h_i} y_i \\
&= \hat{\boldsymbol{\beta}} + \frac{(X^\top X)^{-1}\mathbf{x}_i}{1-h_i}\left[\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} - y_i(1-h_i) - h_i y_i\right] \\
&= \hat{\boldsymbol{\beta}} + \frac{(X^\top X)^{-1}\mathbf{x}_i}{1-h_i}(\hat{y}_i - y_i) = \hat{\boldsymbol{\beta}} - \frac{(X^\top X)^{-1}\mathbf{x}_i}{1-h_i} e_i,
\end{aligned}$$

where we let $e_i = y_i - \hat{y}_i$. This allows us to find

$$e_{(-i)} = y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}_{(-i)} = y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} + \frac{\mathbf{x}_i^\top (X^\top X)^{-1}\mathbf{x}_i}{1-h_i} e_i = e_i + \frac{h_i}{1-h_i} e_i = \frac{e_i}{1-h_i}.$$

It then follows that

$$\text{CV} = \frac{1}{N}\sum_{i=1}^{N} e_{(-i)}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{e_i}{1-h_i}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i-\hat{y}_i}{1-h_i}\right)^2,$$

which was what to be shown. ***Q.E.D.***

b)

False, True, True, False.

Problem 5

a)

b)