Assignment 4

Yue Zhang

2025-10-07

```
setwd("/Users/yuezhang/Documents/Biostat/PH1976")
getwd()
library(ISLR2)
library(ggplot2)
library(tidyr)
library(dtplyr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggcorrplot)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
library(boot)
library(class)
```

3.

(a). We divide the dataset into K equal-sized (or nearly equal) subsets (or folds). For each fold, one subset is held out as the test set, and we train on the remaining K-1 subsets and evaluated on the test fold. The process is repeated K times with a different fold being used as the test set each time.

(b). Validation-set approach: When using a validation set, we can only train on a small portion of the data as we must reserve the rest for validation. As a result it can overestimate the test error rate. The estimate of validation set error can be highly variable as it depends on which observations are used in the train and test dataset. However, the validation-set approach has less computational consumption since we only need

to perform the model once. K-fold has less variance in result that validation set approach and provides a more reliable estimate of model performance compared to a simple train-val split. However, it increases computational complexity.

LOOCV: We can train on n-1 observations and test on the left out observation. As a result, LOOCV can have high variance and risk of overfitting as we catch all the noises. It is also costly in terms of processing time. This approach is more preferred for small datasets. K-fold approach is less consuming than LOOCV and lower variance.

4.

We could use bootstrap by resampling the training data many times. Then refit the model and make a prediction at the target X each time. Finally, we will compute the standard deviation of these bootstrap predictions across the samples.

6.

(a).

```
set.seed(123)

model_6a = glm(data = Default, default ~ income + balance, family = binomial)
summary(model_6a)
```

```
##
## 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
```

The estimated standard error for income is 4.985e-06, the estimated standard error for balance is 2.274e-04.

(b).

```
boot.fn = function(x, i){
  model = glm(data = x[i, ], default ~ income + balance, family = binomial)
  c(income = coef(model)["income"],
    balance = coef(model)["balance"])
}

boot.fn(Default, sample(seq_len(nrow(Default)), replace = TRUE))
```

```
##   income.income balance.balance
##    1.803436e-05    5.448714e-03
```

(c).

```
boot(Default, boot.fn, R = 999)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 999)
##
##
## Bootstrap Statistics :
##         original       bias     std. error
## t1* 2.080898e-05 1.572571e-07 4.767736e-06
## t2* 5.647103e-03 1.288058e-05 2.267663e-04
```

(d). The standard errors obtained by bootstrapping are similar to what we've obtained from part(a).

8.

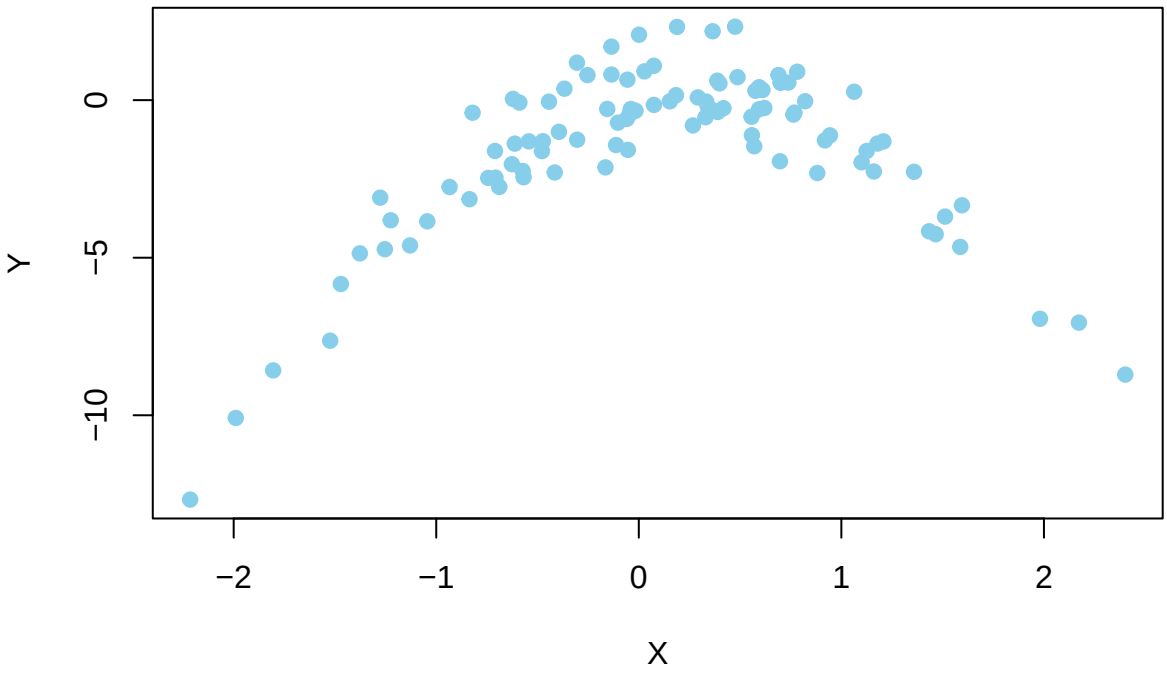
(a).

```
set.seed(1)
x = rnorm(100)
y = x - 2 * x ^ 2 + rnorm(100)
```

$n = 100$, $p = 1$, $y = -2x^2 + x + \epsilon$

(b).

```
plot(
  x,
  y,
  xlab = "X",
  ylab = "Y",
  pch = 19,
  col = "skyblue"
)
```



By looking at the plot we can see that y has a negative quadratic relationship with x.

(c).

```
set.seed(123)
df = data.frame(x, y)
sapply(1:4, function(i) cv.glm(data = df, glm(y ~ poly(x, i)))$delta[1])
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

(d).

```
set.seed(1215)
sapply(1:4, function(i) cv.glm(data = df, glm(y ~ poly(x, i)))$delta[1])
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

The results are the same as for each iteration, one data point is left out as the test set, and the model is trained on the remaining data for LOOCV.

(e).

The model: $y = \beta_0 + \beta_1 * x + \beta_2 * x^2 + \epsilon$ has the smallest LOOCV. Because the data is in quadratic form and we are measuring the test error rate to evaluate performance.

(f).

```
for (i in 1:4)
  printCoefmat(coef(summary(glm(data = df, y ~ poly(
    x, i
  ))))
  )
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -1.55002    0.26001 -5.9613 3.954e-08 ***
## poly(x, i)   6.18883    2.60014  2.3802   0.01924 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)  -1.550023   0.095803 -16.1792 < 2.2e-16 ***
## poly(x, i)1   6.188826   0.958032   6.4599 4.185e-09 ***
## poly(x, i)2 -23.948305   0.958032 -24.9974 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)  -1.550023   0.096263 -16.1019 < 2.2e-16 ***
## poly(x, i)1   6.188826   0.962632   6.4291 4.972e-09 ***
## poly(x, i)2 -23.948305   0.962632 -24.8779 < 2.2e-16 ***
## poly(x, i)3   0.264106   0.962632   0.2744    0.7844
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)  -1.550023   0.095905 -16.1620 < 2.2e-16 ***
## poly(x, i)1   6.188826   0.959051   6.4531 4.591e-09 ***
## poly(x, i)2 -23.948305   0.959051 -24.9708 < 2.2e-16 ***
## poly(x, i)3   0.264106   0.959051   0.2754    0.7836
## poly(x, i)4   1.257095   0.959051   1.3108    0.1931
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the summary we can see that only the coefficients in model 2 are significant as they all have p-values less than 0.05. Even we add more higher terms in later models, those coefficients are not significant. Therefore, the results agree with the conclusions drawn based on the cross-validation results.