2019 학년도 1학기
DATA MINING
HW5

과목명	데이터마이닝
담당교수명	송종우 교수님
제출일	2019.04.17
학번	182STG27
이름	임지연

Ⅰ. Description

일반적으로 Resampling 문제에 대해서 Validation Set approach, Leave-One-Out Cross-Validation (LOOCV), k-Fold Cross-Validation, Bootstrap 방법을 주로 사용한다. 각각의 방법별로 특징이 있으며 데이터에 따라서 사용할 방법을 알맞게 고를 수 있다. 그 후 분석 시 상황에 맞게 가장 성능이 좋은 모델을 선택하게 된다. 본 과제에서는 R 에서 제공하는 함수인 glm, poly, cv.glm, boot 함수를 이용하여 데이터를 적합해 본 후, 그 의미를 알고자 한다. Lab 에 나와있는 Auto 데이터에 대해서 분석할 수 있는 예제 코드를 실행해본 후, Example 2, 5, 7,11 문제를 풀어보며 실제 데이터에 적합해보는 연습을 한다.

Ⅱ. Implementation

Question 1.

Lab : Cross-Validation and the Bootstrap 의 코드를 실행해보고 감상문을 써라

R에서 Resampling 방법인 Validation Set approach, Leave-One-Out Cross-Validation (LOOCV), k-Fold Cross-Validation, Bootstrap 에 대하여 실습을 하면서 방법을 이해하고 장단점을 이해할 수 있었다. 보통 선형함수를 적합할 때 lm, glm 함수를 사용하는데 boot library 에 내장 된 함수인 cv.glm 함수를 사용하기 위하여 glm 함수를 사용하여 적합한 후, LOOCV 방법을 사용하여 test error 를 나타내는 test set MSE 값을 계산한다. 이 방법은 데이터 개수가 n개일 때 i번째 관측치를 하나씩 제외한 데이터로 training set 을 만들어 I 번째 데이터를 예측하는 방법인다. 만약 데이터 수가 매우 클 때 n번을 계산해야 하기 때문에 계산용량이 커지며 비효율적일 수 있는데, 이러한 단점을 보완하기 위해 만들어진 방법이 k-Fold Cross-Validation 방법이다. 이 방법은 k 개의 그룹으로 나눈 후 Cross-Validation 방법을 사용하는 것인데, k개 그룹 중 i번째 그룹을 제외한 나머지 그룹을 training set 으로 사용하고, k 번째 그룹을 test set 으로 사용하는 것이다. 다음으로 Bootstrap 방법은 중복을 허용하여 samling 하는 방법이다.

Question 2.

5.4 Excercises - Example 2, 5, 7, 9 풀어라

[Example 2]

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.
(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

▶ 모든 관측치 중 j 번째 관측치를 선택하지 않을 확률은 $\frac{n-1}{n}$ 이다.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

▶ bootstrap 방법은 복원추출이기 때문에 이전 bootstrap sample 에 영향을 받지 않는다. 따라서 앞서 (a)번과 같은 확률인 $\frac{n-1}{n}$ 가 된다.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is $(1 - 1/n)^n$.

▶ j 번째 관측치를 선택하지 않을 확률은 매 시행마다 동일하므로 j 번째 관측치가 n 번의 bootstrap sample 에 따라 동일한 확률을 갖게 된다. 따라서 확률은 $(\frac{n-1}{n})^n = (1 - \frac{1}{n})^n$

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

▶ (c) 식에 n=5 대입하면, $(1 - \frac{1}{5})^5$ = 0.32768

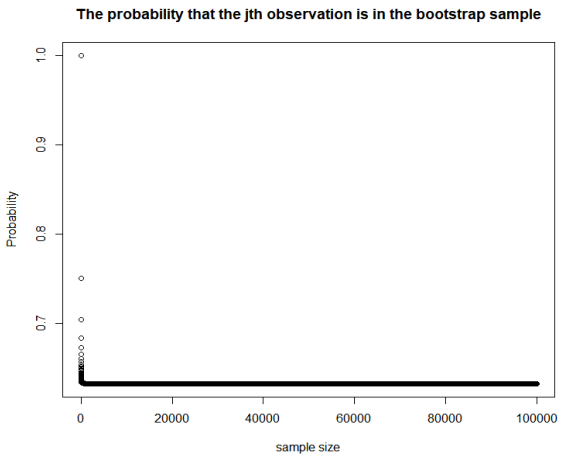
(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

▶ (c) 식에 n=100 대입하면, $(1 - \frac{1}{100})^{100}$ = 0.3660323

(f) When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?

▶ (c) 식에 n=10000 대입하면, $(1 - \frac{1}{10000})^{10000}$ = 0.367861

(g) Create a plot that displays, for each integer value of n from 1 to 100, 000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.



▶ 그래프를 그려본 결과, sample size 가 커질수록 j 번째 관측치가 관찰된 확률은 초반에 급격히 감소하며, 0.6 근처로 수렴하는 것을 볼 수 있다.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 ontains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample. Comment on the results obtained.

```
> store=rep (NA , 10000)
> for (i in 1:10000) { store[i]=sum(sample (1:100 , rep =TRUE)==4) >0 }
> mean(store)
```

▶ 4 번째 관측치의 bootstrap sample 을 갖는 경우, 공식에 대입했을 때의 값인 $1 - (1 - \frac{1}{10000})^{10000}$ = 1- 0.367861 = 0.632 와 비슷한 값인 0.6284 을 갖는 것을 알 수 있다.

[Example 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.

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

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-11.54	0.435	-26.545	0
income	0	0	4.174	0
balance	0.006	0	24.836	0

▶ 회귀모형 적합 결과, income, balance 변수 모두 매우 유의한 것을 알 수 있다.

		Observe	
		No	Yes
Predict	No	9629	225
	Yes	38	108

▶또한 모형 적합 결과 test data set 에서 2.63 % 의 error rate 를 갖는 것을 알 수 있다.

(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.

ii. Fit a multiple logistic regression model using only the training observations.

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.

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

		Observe	
		No	Yes
Predict	No	4816	115
	Yes	19	50

▶ train data set 을 나눈 후 적합 결과, test error rate = 2.68%

(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.

Observe

Predict	**Seed(5)**	No	Yes
	No	4769	159
	Yes	69	3

▶ train data set 을 나눈 후 적합 결과, test error rate = 2.46%

Observe

Predict	**Seed(14)**	No	Yes
	No	4771	160
	Yes	67	2

▶ train data set 을 나눈 후 적합 결과, test error rate = 2.86%

Observe

Predict	**Seed(157)**	No	Yes
	No	4816	102
	Yes	22	60

▶ train data set 을 나눈 후 적합 결과, test error rate = 2.48%

▶ test error 가 거의 2.5 % 로 나타난다.

(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.

Observe

Predict		No	Yes
	No	4766	165
	Yes	67	2

▶ train data set 을 나눈 후 적합 결과, test error rate = 2.88%

▶ student 변수를 포함시킨 결과, error rate 에 큰 변화가 없음을 알 수 있다.

[Example 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 2005. Resampling Methods 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).
(a) Fit a logistic regressionmodel that predicts Direction using Lag1 and Lag2.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.22	0.06	3.6	0
Lag1	-0.04	0.03	-1.48	0.14
Lag2	0.06	0.03	2.27	0.02

(b) Fit a logistic regressionmodel that predicts Direction using Lag1 and Lag2 using all but the first observation.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.22	0.06	3.63	0
Lag1	-0.04	0.03	-1.47	0.14
Lag2	0.06	0.03	2.29	0.02

(c) 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 P(Direction="Up"|Lag1, Lag2) > 0.5. Was this observation correctly classified?

▶ 실제 첫번째 관측값은 Down, 하지만 모델이 예측한 결과는 Up 으로 예측한다.

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

▶ R code 안에 첨부

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

▶ For 문을 이용하여 (d) 에서 코드를 작성하여 수행해 본 결과, test 데이터의 error rate 는 44.4%로 나타났다.

[Example 9]

We will now consider the Boston housing data set, from the MASS library.
(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{\mu}$.

▶ 약 22.53 으로 추정되었다.

(b) Provide an estimate of the standard error of $\hat{\mu}$. 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.

▶ 약 0.4088 로 추정되었다.

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

▶ bootstrap 방법의 standard error 값은 약 0.381 로 (b)와 비슷하게 추정되었다.

(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 $[\hat{\mu} - 2SE(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

▶ 신뢰구간은 [20.4633 , 21.9367] 이며 t test 결과 p-value 가 0 에 매우 가깝기 때문에 귀무가설을 기각하게 된다.

(e) Based on this data set, provide an estimate, $\hat{\mu}_{med}$, for the median value of medv in the population.

▶ median 값은 21.2 로 나타난다.

(f) We now would like to estimate the standard error of $\hat{\mu}_{med}$. 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.

▶ 추정된 standard error 값은 0.368 이다.

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$. (You can use the quantile() function.)

▶ 10% quantile 값은 12.75 로 나타난다.

(h) Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

▶ 추정된 standard error 값은 0.499 이다.

Ⅲ. Discussion

이번 과제를 통해 Resampling 방법에 대해 공부할 수 있었다. 이번 과제를 하기 전까지는 잘 알지 못했던 개념이였는데, 공부하면서 정리할 수 있었다. 대표적인 Resampling 방법인 Validation Set approach, Leave-One-Out Cross-Validation (LOOCV), k-Fold Cross-Validation, Bootstrap 에 대하여 깊은 공부를 할 수 있었는데, 먼저 Validation Set approach 방법은 기존에 가지고 있는 데이터를 적절한 비율로 training set , validation set 으로 나눈 후 training set 을 가지고 모형을 만든 후 validation set 을 가지고 오분류율이 가장 낮은 모형을 선택하는 방법이다. 또한 LOOCV 방법은 한 개의 관측치만 제외한 관측치들을 training data set 으로 사용하여 test error 를 나타내는 test set MSE 값을 계산하는 방법을 관측치 수 n 만큼 반복하면 된다. 하지만 만약 데이터 수가 매우 크면 n번을 계산해야 하기 때문에 계산용량이 커지게 된다. 이러한 단점을 보완하기 위해 만들어진 방법이 k-Fold Cross-Validation 방법이다. 이 방법은 k 개의 그룹으로 나눈 후 Cross-Validation 방법을 사용하는 것인데, k개 그룹 중 i번째 그룹을 제외한 나머지 그룹을 training set 으로 사용하고, k 번째 그룹을 test set 으로 사용하는 것이다. 마지막으로 Bootstrap 방법은 중복을 허용하여 Resamling 하는 방법이다. 적절한 상황에 맞게 Resampling 방법을 사용한다면 성능이 좋은 모형을 찾는 데 많은 도움이 될 것이다.

Ⅳ. Appendix – R code

```
### EX 2
#d
(1-1/5)^5
#e
(1-1/100)^100
#f
(1-1/10000)^10000
#g
x = 1:100000
options(scipen = 100)
plot(x, 1 - (1 - 1/x)^x,
     main ="The probability that the jth observation is in the bootstrap sample",
     xlab ="sample size" , ylab = "Probability")
#h
store=rep (NA , 10000)
for(i in 1:10000) { store[i]=sum(sample (1:100 , rep =TRUE)==4) >0 }
mean(store)
a = (1-1/10000)^10000
1-a

### EX 5
#a
```

```
set.seed(10)
glm.fit = glm(default ~ income + balance, data = Default, family = "binomial")
summary(glm.fit)
glm.prob = predict(glm.fit, Default, type="response")
glm.pred = ifelse(glm.prob > 0.5, "Yes", "No")
mean(Default$default != glm.pred)
#b
set.seed(10)
train = sample( nrow(Default), nrow(Default)*0.5 )
glm.fit = glm(default ~ income + balance, data=Default, family=binomial, subset=train)
glm.prob = predict(glm.fit, Default[-train,], type="response")
glm.pred = ifelse(glm.prob > 0.5, "Yes", "No")
table(glm.pred, Default[-train,]$default)
mean(Default[-train,]$default != glm.pred)    # test mse
#c
set.seed(5)
train = sample( nrow(Default), nrow(Default)*0.5 )
glm.fit1 = glm(default ~ income + balance, data=Default, family=binomial, subset=train)
glm.prob1 = predict(glm.fit1, Default[-train,], type="response")
glm.pred1 = ifelse(glm.prob1 > 0.5, "Yes", "No")
table(glm.pred1, Default[-train,]$default)
mean(Default[-train,]$default != glm.pred1)    # test mse
set.seed(14)
train = sample( nrow(Default), nrow(Default)*0.5 )
glm.fit2 = glm(default ~ income + balance, data=Default, family=binomial, subset=train)
glm.prob2 = predict(glm.fit2, Default[-train,], type="response")
glm.pred2 = ifelse(glm.prob2 > 0.5, "Yes", "No")
table(glm.pred2, Default[-train,]$default)
mean(Default[-train,]$default != glm.pred2)    # test mse
set.seed(157)
train = sample( nrow(Default), nrow(Default)*0.5 )
glm.fit3 = glm(default ~ income + balance, data=Default, family=binomial, subset=train)
glm.prob3 = predict(glm.fit3, Default[-train,], type="response")
glm.pred3 = ifelse(glm.prob3 > 0.5, "Yes", "No")
table(glm.pred3, Default[-train,]$default)
mean(Default[-train,]$default != glm.pred3)    # test mse
#d
set.seed(1)
train = sample(nrow(Default), nrow(Default)*0.5)
glm.fit = glm(default ~ income + balance + student, data=Default, family=binomial, subset=train)
glm.fit = predict(glm.fit, Default[-train,], type="response")
glm.fit = ifelse(glm.fit > 0.5, "Yes", "No")
mean(Default[-train,]$default != glm.fit)     # test mse

### EX 7
#a
glm.fit = glm(Direction ~ Lag1 + Lag2 , data = Weekly, family = "binomial")
summary(glm.fit)
#b
glm.fit = glm(Direction ~ Lag1 + Lag2 , data = Weekly, subset = 2:nrow(Weekly), family = "binomial")
summary(glm.fit)
```

```
#c
ifelse(predict(glm.fit, Weekly, type="response")>0.5, "Up", "Down")[1]
Weekly$Direction[1]
#d
set.seed(1)
LOOCV <- c()
for (i in 1:nrow(Weekly)) {
  glm.fit = glm(Direction ~ Lag1 + Lag2, data=Weekly[-i,], family=binomial)
  glm.pred = ifelse(predict(glm.fit, Weekly[1,], type="response")>0.5, "Up", "Down")
  LOOCV[i] = ifelse(Weekly[i,]$Direction==glm.pred, 0, 1)
}
#e
mean(LOOCV)

### EX 9
#a
mean(Boston$medv)
#b
sd(Boston$medv)/sqrt(nrow(Boston))
#c
set.seed(1)
mean.fn <- function(var, id) {
  return( mean(var[id]) )
}
boot(Boston$medv, mean.fn, R=100)
#d
boot.res$t0 - 2*sd(boot.res$t)   # lower bound
boot.res$t0 + 2*sd(boot.res$t)   # upper bound
t.test(Boston$medv)
#e
median(Boston$medv)
#f
set.seed(1)
median.fn <- function(var, id) {
  return(median(var[id]))
}
(boot.res <- boot(Boston$medv, median.fn, R=100))
#g
quantile(Boston$medv, 0.1)
#h
set.seed(1)
quantile10.fn <- function(var, id) {
  return(quantile(var[id], 0.1))
}
boot(Boston$medv, quantile10.fn, R=100)
```