데이터마이닝 7장 (K-NN 알고리즘)

# 7.2 개인 대출 수락

문제풀이를 수행하기 전, 우선적으로 데이터 셋에 대한 전처리 과정이 필요한 것으로 파악. 3개 이상의 범주를 가지는 범주형 예측변수들을 가변수로 전환한 후, ID, 우편번호, 범주형 예측변수를 제거하는 전처리 과정을 수행했다. 이후, 데이터 셋을 ‘학습 데이터’와 ‘평가 데이터’로 분할한 후, 문제풀이를 진행했다.

|  |
| --- |
| > setwd("C:/Users/ycg00/Desktop/2019-1/데이터마이닝/Dataset/DMBA-R-datasets")  > Bank <- read.csv("UniversalBank.csv")  > View(Bank)  > # 3개 이상 범주형 변수 -> 가변수 변호  >  > Bank$Education <- as.factor(Bank$Education)  >  > edu\_type <- model.matrix( ~ 0 + Education, data = Bank)  > str(edu\_type)  num [1:5000, 1:3] 1 1 1 0 0 0 0 0 0 0 ...  - attr(\*, "dimnames")=List of 2  ..$ : chr [1:5000] "1" "2" "3" "4" ...  ..$ : chr [1:3] "Education1" "Education2" "Education3"  - attr(\*, "assign")= int [1:3] 1 1 1  - attr(\*, "contrasts")=List of 1  ..$ Education: chr "contr.treatment"  > edu\_type <- as.data.frame(edu\_type)  > str(edu\_type)  'data.frame': 5000 obs. of 3 variables:  $ Education1: num 1 1 1 0 0 0 0 0 0 0 ...  $ Education2: num 0 0 0 1 1 1 1 0 1 0 ...  $ Education3: num 0 0 0 0 0 0 0 1 0 1 ...  >  > t(t(names(edu\_type)))  [,1]  [1,] "Education1"  [2,] "Education2"  [3,] "Education3"    > Bank <- cbind(Bank[, -c(1,5,8)],edu\_type)  > View(Bank)  >  > # 더미변수를 한 개 제거하려 했으나  > # 하지 않는다. 198p에 K-NN은 더미변수를 제거하지 않는다.고 써있다.  > dim(Bank)  [1] 5000 14  > #Bank <- Bank[ ,-14]  >  > # 종속변수 위치 조절, 이후 사용을 위해 종속변수를 맨 끝으로 옮기려 한다.  > Bank <- Bank[,c(1:6,8:14,7)]  >  > # 전처리 끝  > ######################################################  > # rownames 는 행에 이름 붙이는 것, 고로 얘는 숫자  > train\_bank\_index <- sample(row.names(Bank),0.6\*dim(Bank)[1])  > valid\_bank\_index <- setdiff(row.names(Bank), train\_bank\_index)  >  > train\_df <- Bank[train\_bank\_index, ]  > valid\_df <- Bank[valid\_bank\_index, ]  >  > ### a ###  >  이제 k=1을 사용하여 데이터의 성공, 실패 클래스를 구분하려고 한다.  > set.seed(1234)  >  > library(caret)  > dim(train\_df)  [1] 3000 14  >  > train\_norm\_bk <- train\_df  > valid\_norm\_bk <- valid\_df  > Bank\_norm\_bk <- Bank  >  > norm.value <- preProcess(train\_df[,1:13], method = c("center","scale"))  >  > train\_norm\_bk[, 1:13] <- predict(norm.value, train\_df[, 1:13])  > valid\_norm\_bk[, 1:13] <- predict(norm.value, valid\_df[, 1:13])  > Bank\_norm\_bk[, 1:13] <- predict(norm.value, Bank[,1:13])  >  > library(FNN)  >  > ## 근접한 k값만 확인할 때  > knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],  + cl = train\_norm\_bk[,14], k = 1)  > A <-table(knn\_pred, valid\_norm\_bk[, 14])  > confusionMatrix(A, positive = "1")  Confusion Matrix and Statistics    knn\_pred 0 1  0 1775 58  1 24 143    Accuracy : 0.959  95% CI : (0.9494, 0.9673)  No Information Rate : 0.8995  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.7548    Mcnemar's Test P-Value : 0.0002682    Sensitivity : 0.7114  Specificity : 0.9867  Pos Pred Value : 0.8563  Neg Pred Value : 0.9684  Prevalence : 0.1005  Detection Rate : 0.0715  Detection Prevalence : 0.0835  Balanced Accuracy : 0.8491    'Positive' Class : 1 |
|  |
| |  | | --- | |  | |

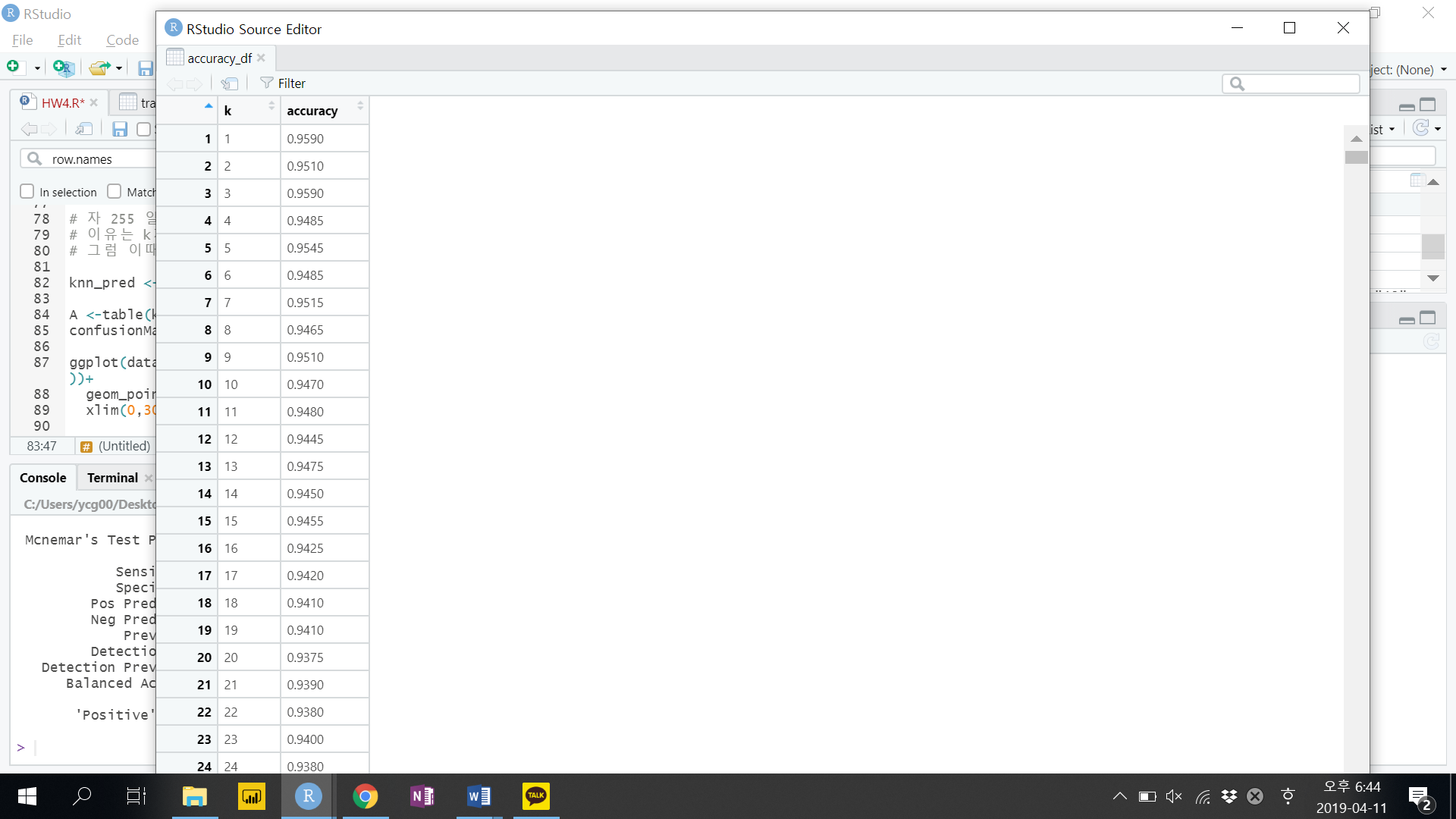
k = 1일 때, 즉 1-NN에서의 ConfusionMatrix 값은 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| 1-NN | | 실제 | |
| 0 | 1 |
| 예측 | 0 | 1775 | 58 |
| 1 | 24 | 143 |

하여, 데이터 셋 내에서 실제 1인 값, 즉 ‘대출 허가’를 예측할 확률은 P(예측 = 1 | 실제 = 1) = 143/201  
 = 0.711, 즉 71.1%의 민감도를 가지고 있다.

반면, 데이터 셋 내에서 실제 0인 값, 즉, ‘대출 불허’를 예측할 확률은 P(예측 = 0 | 실제 = 0) = 1775/1799 = 0.997, 즉 99.7%의 특이도를 가지고 있다.

b.

 반복문을 통해 k값에 따른 정확도 측정을 했다. 그 결과, 가장 높은 값에 해당하는 k들은 다음과 같았다.

이 중, 정확도가 높은 1, 2, 3, 5, 7, 9를 정확도가 높은 k값으로 선정했다. 그리고 개별 k의 정오행렬을 만들어 정확도 측정을 한 결과,

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 1)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1775 58

1 24 143

Accuracy : 0.959

95% CI : (0.9494, 0.9673)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7548

Mcnemar's Test P-Value : 0.0002682

Sensitivity : 0.7114

Specificity : 0.9867

Pos Pred Value : 0.8563

Neg Pred Value : 0.9684

Prevalence : 0.1005

Detection Rate : 0.0715

Detection Prevalence : 0.0835

Balanced Accuracy : 0.8491

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 2)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1795 94

1 4 107

Accuracy : 0.951

95% CI : (0.9406, 0.96)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6617

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.5323

Specificity : 0.9978

Pos Pred Value : 0.9640

Neg Pred Value : 0.9502

Prevalence : 0.1005

Detection Rate : 0.0535

Detection Prevalence : 0.0555

Balanced Accuracy : 0.7651

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 3)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1792 75

1 7 126

Accuracy : 0.959

95% CI : (0.9494, 0.9673)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7331

Mcnemar's Test P-Value : 1.373e-13

Sensitivity : 0.6269

Specificity : 0.9961

Pos Pred Value : 0.9474

Neg Pred Value : 0.9598

Prevalence : 0.1005

Detection Rate : 0.0630

Detection Prevalence : 0.0665

Balanced Accuracy : 0.8115

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 5)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1792 84

1 7 117

Accuracy : 0.9545

95% CI : (0.9444, 0.9632)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6967

Mcnemar's Test P-Value : 1.626e-15

Sensitivity : 0.5821

Specificity : 0.9961

Pos Pred Value : 0.9435

Neg Pred Value : 0.9552

Prevalence : 0.1005

Detection Rate : 0.0585

Detection Prevalence : 0.0620

Balanced Accuracy : 0.7891

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 7)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1794 92

1 5 109

Accuracy : 0.9515

95% CI : (0.9412, 0.9605)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6679

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.5423

Specificity : 0.9972

Pos Pred Value : 0.9561

Neg Pred Value : 0.9512

Prevalence : 0.1005

Detection Rate : 0.0545

Detection Prevalence : 0.0570

Balanced Accuracy : 0.7698

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 9)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1794 93

1 5 108

Accuracy : 0.951

95% CI : (0.9406, 0.96)

No Information Rate : 0.8995

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6636

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.5373

Specificity : 0.9972

Pos Pred Value : 0.9558

Neg Pred Value : 0.9507

Prevalence : 0.1005

Detection Rate : 0.0540

Detection Prevalence : 0.0565

Balanced Accuracy : 0.7673

'Positive' Class : 1

과 같았다. '정확도', '민감도', '특이도', ‘균형잡힌 정확도’만 따로 값을 추출해 정리하면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | Accuracy | Sensitivity | Specificity | **Balanced Accuracy** |
| 1 | 0.959 | 0.7114 | 0.9867 | **0.8491** |
| 2 | 0.951 | 0.5323 | 0.9978 | **0.7651** |
| 3 | 0.959 | 0.6269 | 0.9961 | **0.8115** |
| 5 | 0.9545 | 0.5821 | 0.9961 | **0.7891** |
| 7 | 0.9515 | 0.5423 | 0.9972 | **0.7698** |
| 9 | 0.951 | 0.5373 | 0.9972 | **0.7673** |

위 표를 통해 정확도가 높은 값들 중 해당 k값들이 가장 높은 정확도를 보유함을 파악했다. 그 중 가장 높은 정확도를 보유하고 있는 k는 1과 3으로 파악된다. 그러나 같은 k값을 가지고 있다 하더라도 다른 '민감도', ‘특이도’를 나타냄을 알 수 있다. 하여, Balanced Accuracy를 통해 균형 잡힌 k값을 1과 3중 고르려 한다. **k = 1**일 때, **Balanced Accuracy는 0.849**1이며, **k = 3**일 때, **Balanced Accuracy는 0.8115**를 나타낸다.

고로, 가장 균형 잡히면서 높은 정확도를 가진 k값은 **‘k =1’**이다.

c.  
 앞서, 문제 b. 를 통해 최적의 k값은 1임을 확인했다.  
 k는 1일 때의 정오행렬은

> knn\_pred <- knn(train\_norm\_bk[,1:13], valid\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 1)

> A <-table(knn\_pred, valid\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 1775 58

1 24 143

**이다**.

d.

최적의 ‘k=1’의 정오행렬을 토대로 보면

knn\_pred 0 1

0 1775 58

1 24 143

다음과 같다.

‘**민감도**’, 즉 예측을 통해서 실제 대출을 ‘수락’할 고객을 맞힐 확률은  
 P(예측 = 1 | 실제 = 1) = 143/199 = 0.719로 나온다.  
 반면**, ‘민감도’의 오차율**, 즉 실제로 ‘수락’이지만, 예측에서 ‘불가’를 내릴 확률은  
 P(예측 = 0 | 실제 = 1) = 58/199 = 0.291 로 나온다.  
  
 ‘**특이도**’, 즉 실제 ‘불가’인 고객을 예측을 통해서도 ‘불가’로 맞출 확률은  
 P(예측 = 0 | 실제 = 0) = 17751799 = 0.987 로 나온다.  
 반면, ‘**특이도’의 오차율**, 즉 실제로는 ‘불가’이지만, 예측에서 ‘수락’을 내릴 확률은  
 P(예측 = 1 | 실제 = 0) = 24/1799 = 0.013 로 나온다.

e.

우선, 학습용(50%), 검증용(30%), 평가용(20%)으로 데이터를 나눈다.

> train\_bank\_index <- sample(row.names(Bank),0.5\*dim(Bank)[1])

> valid\_bank\_index <- sample(setdiff(row.names(Bank),train\_bank\_index), 0.3\*dim(Bank)[1])

> test\_bank\_index <- setdiff(row.names(Bank), union(train\_bank\_index,valid\_bank\_index))

>

>

> train\_df <- Bank[train\_bank\_index, ]

> valid\_df <- Bank[valid\_bank\_index, ]

> test\_df <- Bank[test\_bank\_index, ]

>

>

> ### 전처리 완료

>

> set.seed(1234)

>

> library(caret)

> dim(train\_df)

[1] 2500 14

>

> train\_norm\_bk <- train\_df

> valid\_norm\_bk <- valid\_df

> test\_norm\_bk <- test\_df

>

> norm.value <- preProcess(train\_df[,1:13], method = c("center","scale"))

>

> train\_norm\_bk[, 1:13] <- predict(norm.value, train\_df[, 1:13])

> valid\_norm\_bk[, 1:13] <- predict(norm.value, valid\_df[, 1:13])

> test\_norm\_bk[, 1:13] <- predict(norm.value, test\_df[,1:13])

> library(FNN)

그리고 ‘평가’세트에 대한 분류행렬을 ‘학습’세트 및 ‘검증’세트의 정오행렬표외 비교하려 한다.

> ## Training ~ Test

먼저 ‘학습’ 세트와 ‘평가’세트의 비교다.

>

> accuracy\_df <- data.frame(k = seq(1,2500,1), accuracy = rep(0,2500))

> for (i in 1:2500) {

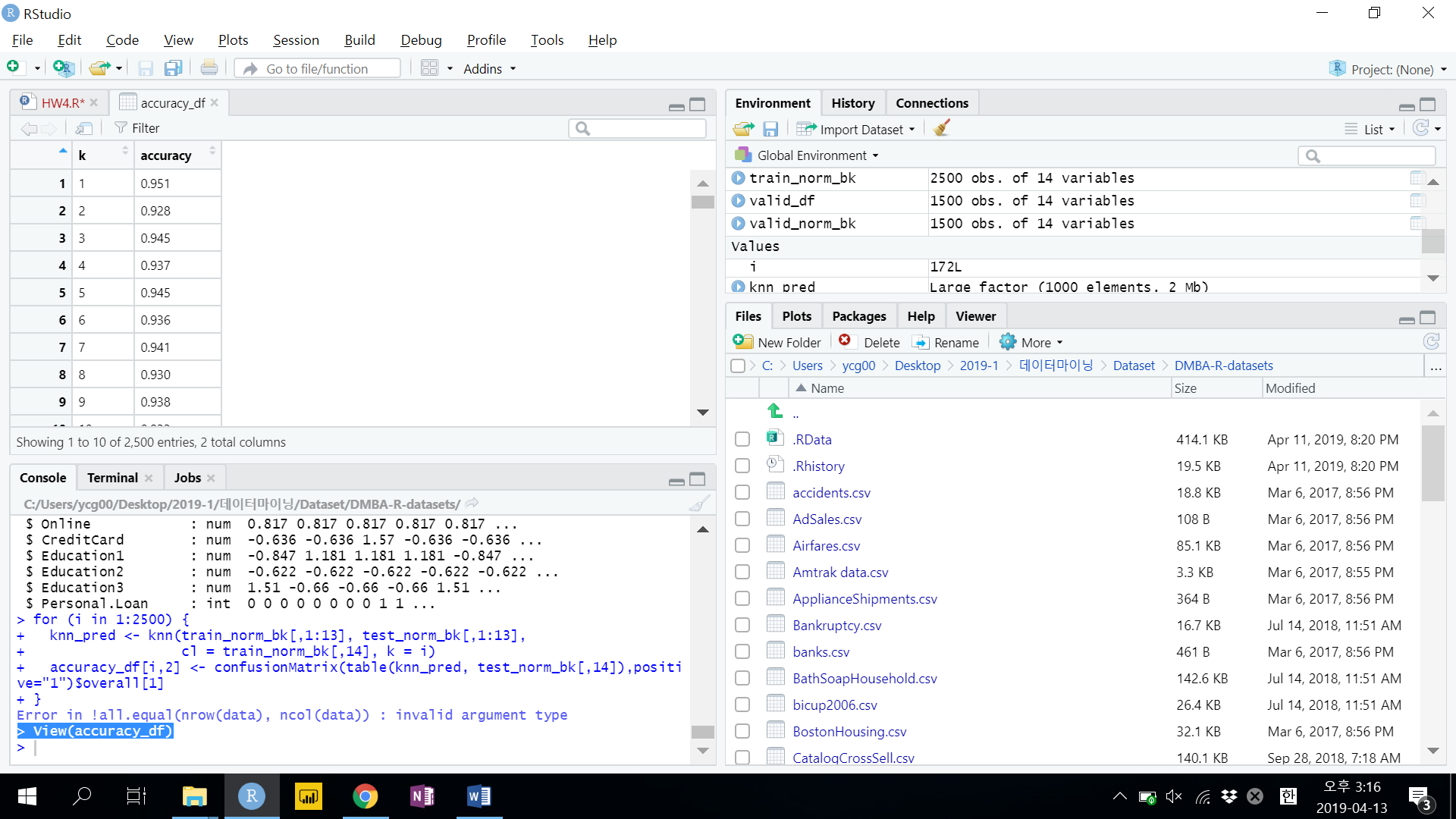
+ knn\_pred <- knn(train\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = i)

+ accuracy\_df[i,2] <- confusionMatrix(table(knn\_pred, test\_norm\_bk[,14]),positive="1")$overall[1]

+ }

> View(accuracy\_df)



View를 통해 정확도가 높은 k값을 찾아본 결과, k = 1, 3, 5, 7의 k값이 정확도가 높은 것으로 나타났다.  
그것을 기반으로 각 k값이 따라 ConfusionMatrix를 진행하면 다음과 같다.

> knn\_pred <- knn(train\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 1)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 880 42

1 7 71

Accuracy : 0.951

95% CI : (0.9357, 0.9635)

No Information Rate : 0.887

P-Value [Acc > NIR] : 9.854e-13

Kappa : 0.7174

Mcnemar's Test P-Value : 1.191e-06

Sensitivity : 0.6283

Specificity : 0.9921

Pos Pred Value : 0.9103

Neg Pred Value : 0.9544

Prevalence : 0.1130

Detection Rate : 0.0710

Detection Prevalence : 0.0780

Balanced Accuracy : 0.8102

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 3)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 885 53

1 2 60

Accuracy : 0.945

95% CI : (0.929, 0.9583)

No Information Rate : 0.887

P-Value [Acc > NIR] : 1.606e-10

Kappa : 0.6584

Mcnemar's Test P-Value : 1.562e-11

Sensitivity : 0.5310

Specificity : 0.9977

Pos Pred Value : 0.9677

Neg Pred Value : 0.9435

Prevalence : 0.1130

Detection Rate : 0.0600

Detection Prevalence : 0.0620

Balanced Accuracy : 0.7644

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 5)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 886 54

1 1 59

Accuracy : 0.945

95% CI : (0.929, 0.9583)

No Information Rate : 0.887

P-Value [Acc > NIR] : 1.606e-10

Kappa : 0.655

Mcnemar's Test P-Value : 2.355e-12

Sensitivity : 0.5221

Specificity : 0.9989

Pos Pred Value : 0.9833

Neg Pred Value : 0.9426

Prevalence : 0.1130

Detection Rate : 0.0590

Detection Prevalence : 0.0600

Balanced Accuracy : 0.7605

'Positive' Class : 1

> knn\_pred <- knn(train\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = train\_norm\_bk[,14], k = 7)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 886 58

1 1 55

Accuracy : 0.941

95% CI : (0.9246, 0.9548)

No Information Rate : 0.887

P-Value [Acc > NIR] : 3.272e-09

Kappa : 0.6226

Mcnemar's Test P-Value : 3.086e-13

Sensitivity : 0.4867

Specificity : 0.9989

Pos Pred Value : 0.9821

Neg Pred Value : 0.9386

Prevalence : 0.1130

Detection Rate : 0.0550

Detection Prevalence : 0.0560

Balanced Accuracy : 0.7428

## Validation ~ Test

‘평가’세트와 ‘검증’세트 비교.

> for (i in 1:1500) {

+ knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = valid\_norm\_bk[,14], k = i)

+ accuracy\_df[i,2] <- confusionMatrix(table(knn\_pred, test\_norm\_bk[,14]))

+ }

Error in `[<-.data.frame`(`\*tmp\*`, i, 2, value = list(positive = "0", :

replacement element 2 is a matrix/data frame of 2 rows, need 1

> for (i in 1:1500) {

+ knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = valid\_norm\_bk[,14], k = i)

+ accuracy\_df[i,2] <- confusionMatrix(table(knn\_pred, test\_norm\_bk[,14]))$overall[1]

+ }

Error in !all.equal(nrow(data), ncol(data)) : invalid argument type

> View(test\_norm\_bk)

> View(accuracy\_df)

> for (i in 1:1500) {

+ knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

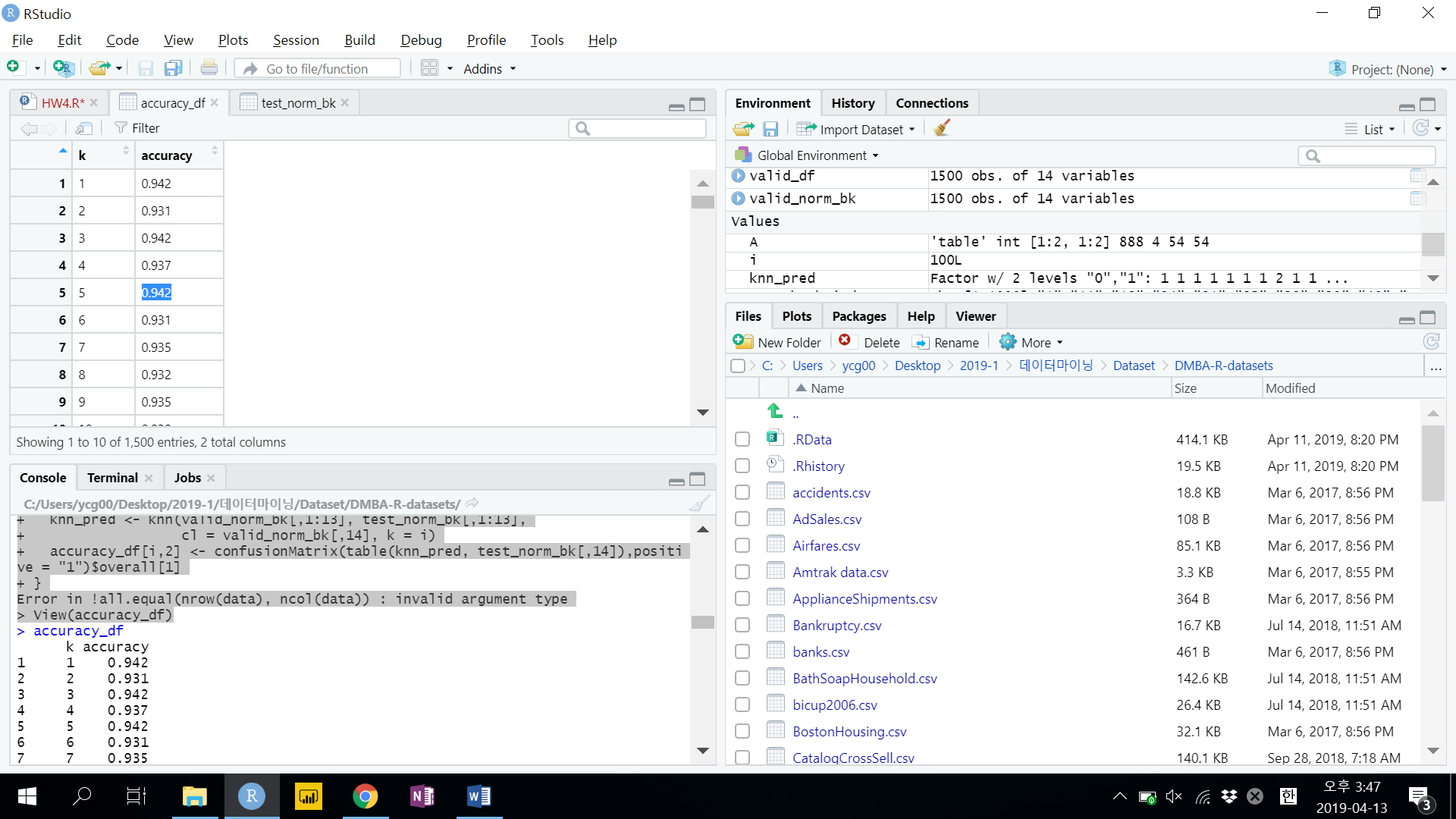
+ cl = valid\_norm\_bk[,14], k = i)

+ accuracy\_df[i,2] <- confusionMatrix(table(knn\_pred, test\_norm\_bk[,14]),positive = "1")$overall[1]

+ }

Error in !all.equal(nrow(data), ncol(data)) : invalid argument type

> View(accuracy\_df)



## k = 1, 3, 5 설정

> knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = valid\_norm\_bk[,14], k = 1)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 878 44

1 14 64

**Accuracy : 0.942**

95% CI : (0.9257, 0.9557)

No Information Rate : 0.892

P-Value [Acc > NIR] : 2.395e-08

Kappa : 0.6571

Mcnemar's Test P-Value : 0.0001402

Sensitivity : 0.5926

Specificity : 0.9843

Pos Pred Value : 0.8205

Neg Pred Value : 0.9523

Prevalence : 0.1080

Detection Rate : 0.0640

Detection Prevalence : 0.0780

**Balanced Accuracy : 0.7884**

'Positive' Class : 1

> knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = valid\_norm\_bk[,14], k = 3)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 889 55

1 3 53

**Accuracy : 0.942**

95% CI : (0.9257, 0.9557)

No Information Rate : 0.892

P-Value [Acc > NIR] : 2.395e-08

Kappa : 0.6182

Mcnemar's Test P-Value : 2.133e-11

Sensitivity : 0.4907

Specificity : 0.9966

Pos Pred Value : 0.9464

Neg Pred Value : 0.9417

Prevalence : 0.1080

Detection Rate : 0.0530

Detection Prevalence : 0.0560

**Balanced Accuracy : 0.7437**

'Positive' Class : 1

> knn\_pred <- knn(valid\_norm\_bk[,1:13], test\_norm\_bk[,1:13],

+ cl = valid\_norm\_bk[,14], k = 5)

> A <-table(knn\_pred, test\_norm\_bk[, 14])

> confusionMatrix(A, positive = "1")

Confusion Matrix and Statistics

knn\_pred 0 1

0 888 54

1 4 54

**Accuracy : 0.942**

95% CI : (0.9257, 0.9557)

No Information Rate : 0.892

P-Value [Acc > NIR] : 2.395e-08

Kappa : 0.6221

Mcnemar's Test P-Value : 1.243e-10

Sensitivity : 0.5000

Specificity : 0.9955

Pos Pred Value : 0.9310

Neg Pred Value : 0.9427

Prevalence : 0.1080

Detection Rate : 0.0540

Detection Prevalence : 0.0580

**Balanced Accuracy : 0.7478**

'Positive' Class : 1