12주차:
모델의 성능 평가

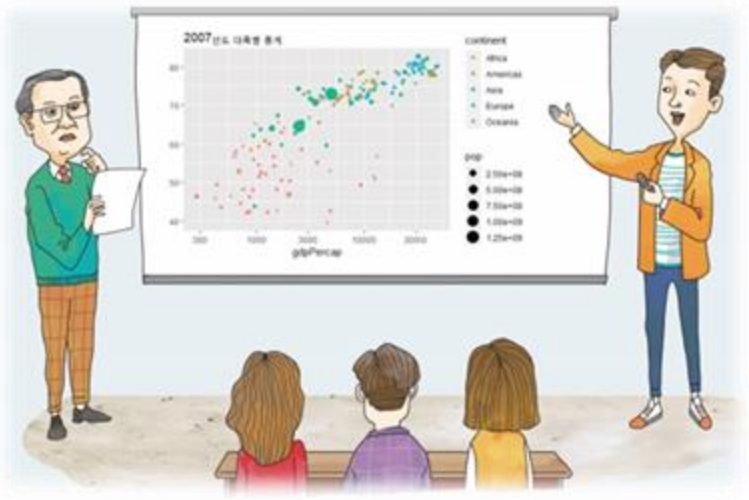
ChulSoo Park

School of Computer Engineering & Information Technology

Korea National University of Transportation

10 CHAPTER

모델의 성능 평가



CONTENTS

10.3 일반화 능력 측정

일반화 능력이란?

- 학습에 사용하지 않은 새로운 샘플에 대해 높은 성능을 가지는 성질

- 예)
  - ✓ 학습에 사용한 훈련 집합에 대한 정확률 : 98%
  - ✓ 새로운 샘플(현장)에 적용 : 97%의 정확률 → 일반화 능력이 뛰어 남
  - ✓ 반면 현장에서 적용하니 65%의 정확률 → 일반화 능력이 낮음

10.3 일반화 능력 측정

- 지금까지는 학습할 때 사용한 데이터를 성능 평가에 다시 사용함
  - 수업시간에 풀어본 문제를 중간,기말고사에 그대로 출제한다면 ?
  - 시험 범위는 배운 곳과 같게 하되 새로운 문제를 출제해야 제대로 평가

- 일반화 능력 측정 방법
  - 현장에 설치하여 일반화를 측정하면 좋지만, 불가능한 경우가 대부분.

    예) 의료 현장
  - 현재 가지고 있는 데이터를 잘 활용하여 측정해야 함

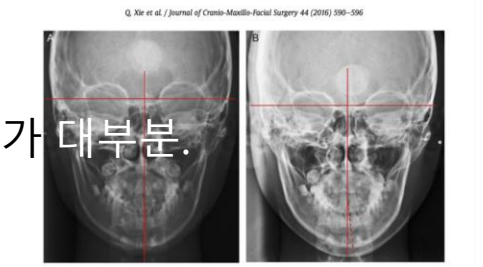
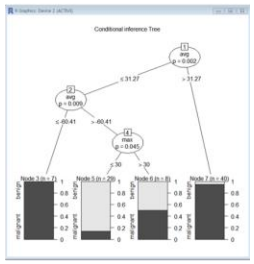


Fig. 6. Before and after PA image: A. first visit: no asymmetry was found, B. revisit: more than 5 mm skeletal asymmetry was noticed.



10.3 일반화 능력 측정

- 수집한 데이터를 훈련 집합과 테스트 집합으로 나누어 사용
  - 훈련 집합으로 모델을 학습하고, 테스트 집합으로 학습된 모델의 성능을 평가함
  - 보통 5:5, 6:4, 또는 7:3으로 나눔

- 캐글과 같은 데이터 과학 대회에서는 (1장의 [표 1-1] 대회 목록 참조)
  - 데이터를 훈련 집합과 테스트 집합으로 나눈 다음 훈련 집합을 공개
  - 테스트 집합은 정답을 빼고 공개 (정답이란 반응 변수의 값)
  - 대회 참가자는 훈련 집합으로 자신의 모델을 학습한 다음, 테스트 집합을 예측하고 예측 결과 파일을 대회 본부에 제출
  - 대회 본부는 숨겨둔 정답을 예측 결과와 비교하여 정확률을 계산함

10.3 일반화 능력 측정

2019 1st ML month with KaKR

캐글 코리아와 함께하는 1st ML 대회 - 타이타닉 생존자를 예측하라!

349 teams · 2 years ago

Overview Data Code Discussion Leaderboard Rules **Late Submission**

Overview

- Description
- Evaluation
- Prize
- Timeline

Introduction

본 대회는 구글 코리아가 후원하고, 캐글 코리아(비영리 페이스북 온라인 커뮤니티)가 진행하는 데이터 사이언스 대회입니다. Academic 목적이며, 대한민국 누구나 참여하실 수 있습니다.

Competition background

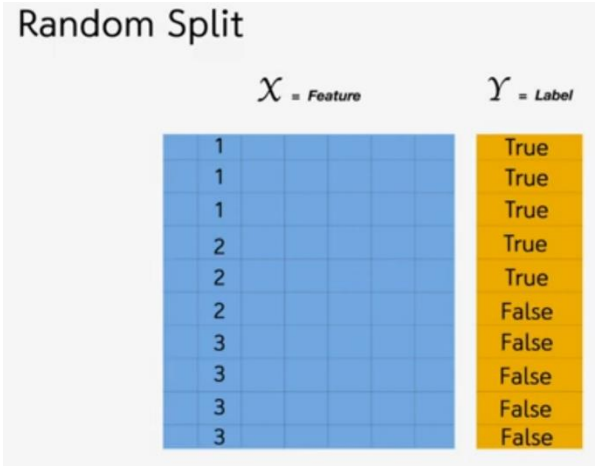
RMS 타이타닉의 침몰은 역사상 가장 악명 높은 참사 중 하나입니다. 1912 년 4 월 15 일, 첫 항해 도중 타이타닉은 빙산과 충돌 한 후 침몰하였고, 이로 인해 2224 명의 승객과 승무원 중 1502 명이 사망했습니다. 이 비극은 국제 사회에 큰 충격을 주었고, 선박 안전 규정을 개선하는 계기가 되었습니다.

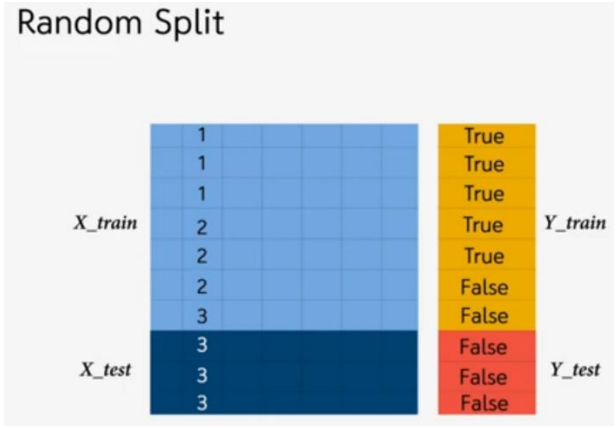
많은 사망자가 생긴 이유 중 하나는 승객과 승무원을위한 구명정이 충분하지 않았기 때문입니다. 침몰에서

10.3 일반화 능력 측정

훈련 집합과 테스트 집합으로 분리

data split (70 : 30)





label(정답) : 합격과 불합격, 이번 예는 타이타닉호 침몰 머신러닝 모델

10.3 일반화 능력 측정

Overview Data Code Discussion Leaderboard Rules **Late Submission**

Data Description

File descriptions

- train.csv - 예측 모델을 만들기 위해 사용하는 학습셋입니다. 각 탑승객의 신상정보와 ground truth(생존유무)가 주어지며, 신상정보 및 파생변수를 토대로 생존유무를 예측하는 모델을 만듭니다.
- test.csv - 학습셋으로 만든 모델을 가지고 예측할 탑승객 정보가 담긴 테스트셋입니다.
- sampleSubmission.csv - 제출시 사용할 수 있는 csv 파일입니다.

```
>_ kaggle competitions download -c 2019-1st-ml-month-with-kakr
```

Data Explorer

90.9 KB

- sample_submission.csv
- test.csv
- train.csv

< **sample_submission.csv** (3.18 KB)

Competition Rules

10.3 일반화 능력 측정

- iris의 예(train data와 test data로 나누기)

train data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5	2	3.5	1	versicolor
5.7	4.4	1.5	0.4	setosa
7.1	3	5.9	2.1	virginica
5.6	2.5	3.9	1.1	versicolor
7.2	3.2	6	1.8	virginica
5.4	3.4	1.7	0.2	setosa
5.7	3.8	1.7	0.3	setosa

test data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.7	2.8	4.5	1.3	
5.7	2.6	3.5	1	
5.7	2.5	5	2	
5.8	4	1.2	0.2	
5.8	2.7	4.1	1	
5.8	2.7	5.1	1.9	

- 대회에 참가했다면,
  - 가진 것은 훈련 집합뿐(테스트 집합이 공개되었더라도 정답이 없어 학습에 활용 불가능)
  - 훈련 집합만 가지고 일반화 능력을 어떻게 측정하나? → 훈련 집합을 전체 데이터로 간주하고 훈련 집합과 테스트 집합으로 나누어 사용함 (또는 4절의 교차 검증 활용)

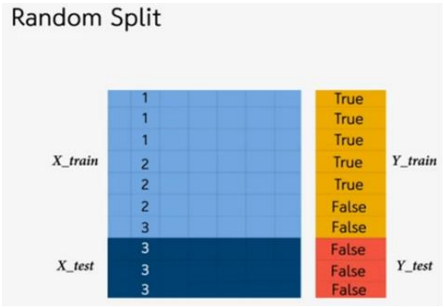
10.3 일반화 능력 측정

데이터를 훈련 집합과 테스트 집합으로 나누기

- iris 데이터를 7:3 비율로 나누는 코드
- sample 함수를 사용하여 랜덤하게 나눔

```
Console C:/RSources/
> n = nrow(iris) # iris 샘플 수
> n
[1] 150
> i = 1:n                          # 1, 2, 3, ..., n을 가진 list i
> train_list = sample(i, n*0.7)    # 70%를 랜덤마게 sample 선택
> test_list = setdiff(i, train_list)   # 나머지(30%)
> iris_train = iris[train_list, ]          # train data select
> iris_test = iris[test_list, ]    # ? test data select
```

iris_train=105 개
iris_test = 45 개



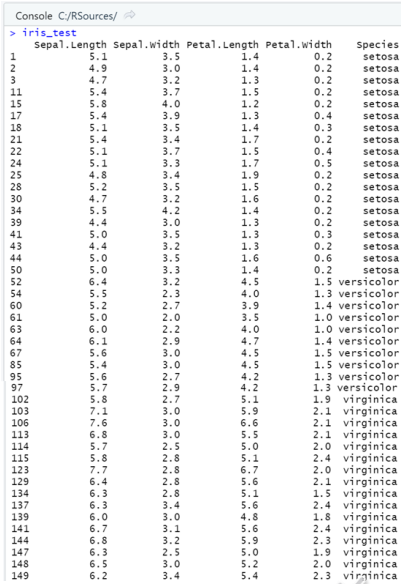

```
Console C:/RSources/
> iris_test
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
11           5.4         3.7          1.5         0.2     setosa
15           5.8         4.0          1.2         0.2     setosa
17           5.4         3.9          1.3         0.4     setosa
18           5.1         3.5          1.4         0.3     setosa
21           5.4         3.4          1.7         0.2     setosa
22           5.1         3.7          1.5         0.4     setosa
24           5.1         3.3          1.7         0.5     setosa
25           4.8         3.4          1.9         0.2     setosa
28           5.2         3.5          1.5         0.2     setosa
30           4.7         3.2          1.6         0.2     setosa
34           5.5         4.2          1.4         0.2     setosa
39           4.4         3.0          1.3         0.2     setosa
41           5.0         3.5          1.3         0.3     setosa
43           4.4         3.2          1.3         0.2     setosa
44           5.0         3.5          1.6         0.6     setosa
50           5.0         3.3          1.4         0.2     setosa
52           6.4         3.2          4.5         1.5 versicolor
54           5.5         2.3          4.0         1.3 versicolor
60           5.2         2.7          3.9         1.4 versicolor
61           5.0         2.0          3.5         1.0 versicolor
63           6.0         2.2          4.0         1.0 versicolor
64           6.1         2.9          4.7         1.4 versicolor
67           5.6         3.0          4.5         1.5 versicolor
85           5.4         3.0          4.5         1.5 versicolor
95           5.6         2.7          4.2         1.3 versicolor
97           5.7         2.9          4.2         1.3 versicolor
102          5.8         2.7          5.1         1.9  virginica
103          7.1         3.0          5.9         2.1  virginica
106          7.6         3.0          6.6         2.1  virginica
113          6.8         3.0          5.5         2.1  virginica
114          5.7         2.5          5.0         2.0  virginica
115          5.8         2.8          5.1         2.4  virginica
123          7.7         2.8          6.7         2.0  virginica
129          6.4         2.8          5.6         2.1  virginica
134          6.3         2.8          5.1         1.5  virginica
137          6.3         3.4          5.6         2.4  virginica
139          6.0         3.0          4.8         1.8  virginica
141          6.7         3.1          5.6         2.4  virginica
144          6.8         3.2          5.9         2.3  virginica
147          6.3         2.5          5.0         1.9  virginica
148          6.5         3.0          5.2         2.0  virginica
149          6.2         3.4          5.4         2.3  virginica
```

10.3 일반화 능력 측정

- 랜덤 포리스트의 일반화 성능을 측정하는 코드
  - randomForest 함수로 학습할 때는 훈련 집합(iris_train)을 사용 (7:3)
  - predict 함수로 예측할 때는 테스트 집합(iris_test)을 사용함 정답
  - 정확률 100% : iris data에 대해서 일반화 능력 우수

예측 결과

```
> f = randomForest(Species~., data = iris_train) # train data로 randomforest 학습
> p = predict(f, newdata=iris_test)
> p
         1          2          3         11         15         17         18         21         22
    setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
        24         25         28         30         34         39         41         43         44
    setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
        50         52         54         60         61         63         64         67         85
    setosa versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
        95         97        102        103        106        113        114        115        123
versicolor versicolor  virginica  virginica  virginica  virginica  virginica  virginica  virginica
       129        134        137        139        141        144        147        148        149
 virginica versicolor  virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```



10.3 일반화 능력 측정

- 랜덤 포리스트의 일반화 성능을 측정하는 코드
  - randomForest 함수로 학습할 때는 훈련 집합(iris_train)을 사용 (6:4)
  - predict 함수로 예측할 때는 테스트 집합(iris_test)을 사용함
  - 정확률 59/60=98.33% : iris data에 대해서 일반화 능력 우수

Console C:/RSources/

111	6.5	3.2	5.1	2.0	virginica
112	6.4	2.7	5.3	1.9	virginica
114	5.7	2.5	5.0	2.0	virginica
116	6.4	3.2	5.3	2.3	virginica
117	6.5	3.0	5.5	1.8	virginica
119	7.7	2.6	6.9	2.3	virginica
120	6.0	2.2	5.0	1.5	virginica
122	5.6	2.8	4.9	2.0	virginica
123	7.7	2.8	6.7	2.0	virginica
125	6.7	3.3	5.7	2.1	virginica
129	6.4	2.8	5.6	2.1	virginica
131	7.4	2.8	6.1	1.9	virginica
133	6.4	2.8	5.6	2.2	virginica
136	7.7	3.0	6.1	2.3	virginica
137	6.3	3.4	5.6	2.4	virginica
142	6.9	3.1	5.1	2.3	virginica
145	6.7	3.3	5.7	2.5	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica

예측 결과 정답

```
> f = randomForest(Species~., data = iris_train) # train data로 randomforest 학습
> p = predict(f, newdata=iris_test)
> p
         3          4          5          7         11         15         17         19         29
    setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
        31         33         35         36         37         39         40         41         44
    setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
        45         50         56         61         64         65         66         69         70
    setosa     setosa versicolor versicolor versicolor versicolor versicolor versicolor versicolor
        74         75         76         80         86         87         89         91         93
versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor versicolor
        96         97         98        100        111        112        114        116        117
versicolor versicolor versicolor versicolor  virginica  virginica  virginica  virginica  virginica
       119        120        122        123        125        129        131        133        136
 virginica versicolor  virginica  virginica  virginica  virginica  virginica  virginica  virginica
       137        142        145        147        148        149
 virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

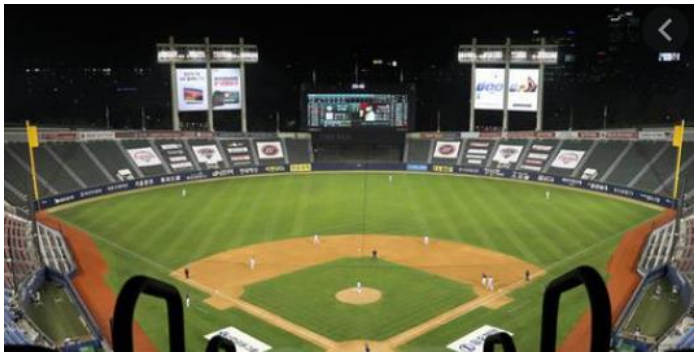
10.3 일반화 능력 측정

- caret 라이브러리(Classification And REgression Training)
  - caret 라이브러리의 다양한 사용
    - ✓ 훈련 집합과 테스트 집합 분할
    - ✓ train 함수 (여러 회귀와 분류 모델을 일관성 있게 평가하는데 유용. 9.6.3절 참조)
    - ✓ 교차 검증 (다음 절)

```
Console C:/RSources/
> library(caret)
> train_list = createDataPartition(y = iris$Species, p = 0.6, list = FALSE)
> iris_train = iris[train_list, ]
> iris_test = iris[-train_list, ]
> f = randomForest(Species~., data = iris_train) # train data로 randomforest 학습
> p = predict(f, newdata = iris_test)
```

```
Console C:/RSources/
> r = train(Species~., data = iris, method = 'rpart')
> f = train(Species~., data = iris, method = 'rf')
> s = train(Species~., data = iris, method = 'svmRadial')
> k = train(Species~., data = iris, method = 'knn')
```

10.4 교차 검증



팀순위 2020. 현재

순위	팀	경기수	승	패	무	승률
1	NC	144	83	55	6	0.601
2	두산	144	79	61	4	0.564
3	KT	144	81	62	1	0.566
4	LG	144	79	61	4	0.564
5	키움	144	80	63	1	0.559
6	KIA	144	73	71	0	0.507
7	롯데	144	71	72	1	0.497
8	삼성	144	64	75	5	0.460
9	SK	144	51	92	1	0.357
10	한화	144	46	95	3	0.326

팀순위 2021. 현재

순위	팀	경기수	승	패	무	승률
1	삼성	33	20	13	0	0.606
2	SSG	32	18	14	0	0.563
3	NC	32	17	15	0	0.531
3	두산	32	17	15	0	0.531
3	KT	32	17	15	0	0.531
3	LG	32	17	15	0	0.531
7	KIA	32	15	17	0	0.469
8	키움	33	15	18	0	0.455
9	한화	32	13	19	0	0.406
10	롯데	32	12	20	0	0.375

10.4 교차 검증

3절이 사용한 방법의 한계

- iris를 (7:3)으로 나누고 일반화 능력 측정
- 우연히 높은 정확률을 얻게 분할 될 수도 있고 그 반대 일 수도 있음
- 여러 번 반복하고 **평균을 취하면 우연을 줄이고 통계적 신뢰성을 높일 수 있음**

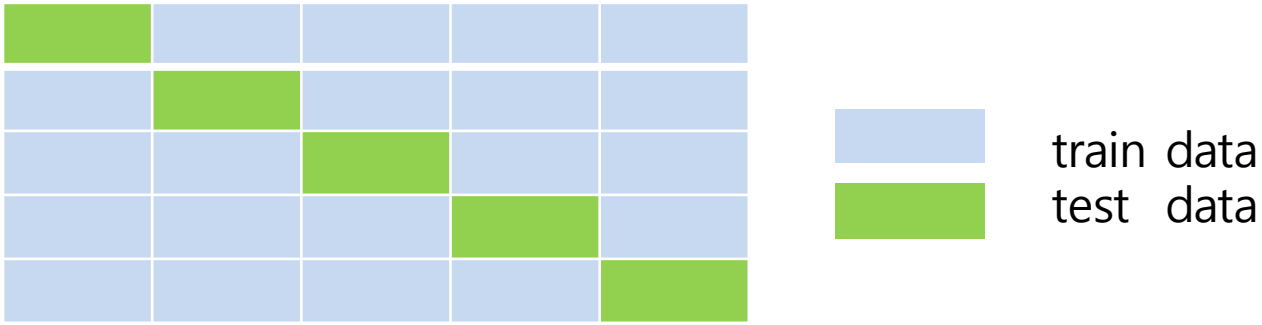
여러 번 반복하는 두 가지 방법

- 붓스트랩(bootstrap): 랜덤 샘플링과 평가를 k번 반복하고 평균을 취함
- 교차 검증: 붓스트랩 보다 교차 검증을 주로 사용

10.4 교차 검증

k-겹 교차 검증

- 데이터를 k개 부분 집합으로 분할 ([그림]은 5-겹 교차 검증)
- 학습과 평가를 k 번 반복하고, 평균을 취함



10.4 교차 검증

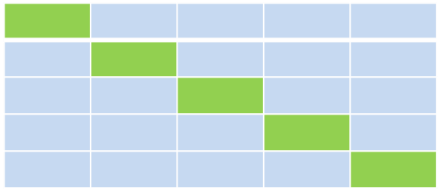
- k-겹 교차 검증을 알고리즘

1. data를 섞는다
2. data를 k개로 등분
3. accuracy = 0
4. for i = 1 to k
5. i번째를 test 집합으로
6. i외를 train 집합으로
7. train 집합으로 학습
8. test 집합으로 accuracy 측정
9. accuracy = accuracy + accuracyi
10. average_accuracy = accuracy / k

10.4 교차 검증

k-겹 교차 검증을 알고리즘 : for 문을 이용한 코드

```
14  # K-겹 교차 검증 #
15  library(caret)
16  data = iris[sample(nrow(iris)), ]   # iris data 순서 섞음.
17  head(data)
18  k = 5
19  q = nrow(data)/k          # k개 등분시 부분집합 크기
20  q
```



Console Jobs

C:/RSources/

```
> data = iris[sample(nrow(iris)), ]     # iris data 순서 섞음.
> head(data)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
7            4.6         3.4          1.4         0.3     setosa
32           5.4         3.4          1.5         0.4     setosa
98           6.2         2.9          4.3         1.3 versicolor
133          6.4         2.8          5.6         2.2  virginica
132          7.9         3.8          6.4         2.0  virginica
11           5.4         3.7          1.5         0.2     setosa
> data = iris[sample(nrow(iris)), ]     # iris data 순서 섞음.
> head(data)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
18           5.1         3.5          1.4         0.3     setosa
81           5.5         2.4          3.8         1.1 versicolor
146          6.7         3.0          5.2         2.3  virginica
129          6.4         2.8          5.6         2.1  virginica
34           5.5         4.2          1.4         0.2     setosa
105          6.5         3.0          5.8         2.2  virginica
> k = 5
> q = nrow(data)/k                      # k개 등분시 부분집합 크기
> q
[1] 30
```

10.4 교차 검증

- *k*-겹 교차 검증을 알고리즘 : for 문을 이용한 코드

Console Jobs

C:/RSources/

```
> data = iris[sample(nrow(iris)), ]      # iris data 순서 섞음.
> k = 5
> q = nrow(data)/k                       # k개 등분시 부분집합 크기
> l = 1:nrow(data)
> accuracy = 0
> for(i in 1:k) {
+   test_list = ((i-1)*q+1):(i*q)          # i번째를 test집합으로 설정
+   testData = data[test_list, ]
+   train_list = setdiff(l, test_list)  # i번째를 제외하고 train 집합으로 설정
+   trainData = data[train_list, ]
+   f = train(Species~., data = trainData, method = 'rf') #모델학습(randomforest)
+   p = predict(f, newdata = testData)
+   t = table(p, testData$Species)
+   accuracy=accuracy+(t[1, 1]+t[2, 2]+t[3, 3])/length(test_list) # 정확률&누적
+ }
> (average_accuracy = accuracy/k)                # 평균 정확률
[1] 0.9533333
```

[1] 0.9466667

[1] 0.94

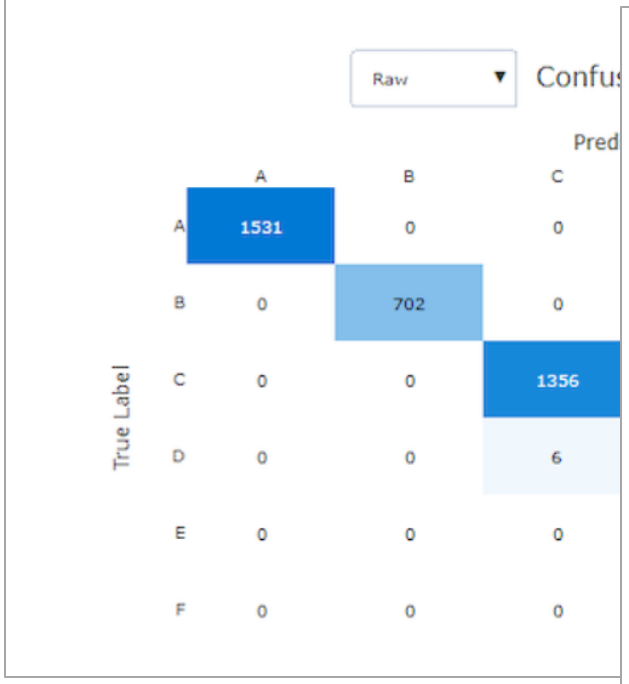
[1] 0.9533333

10.4 교차 검증

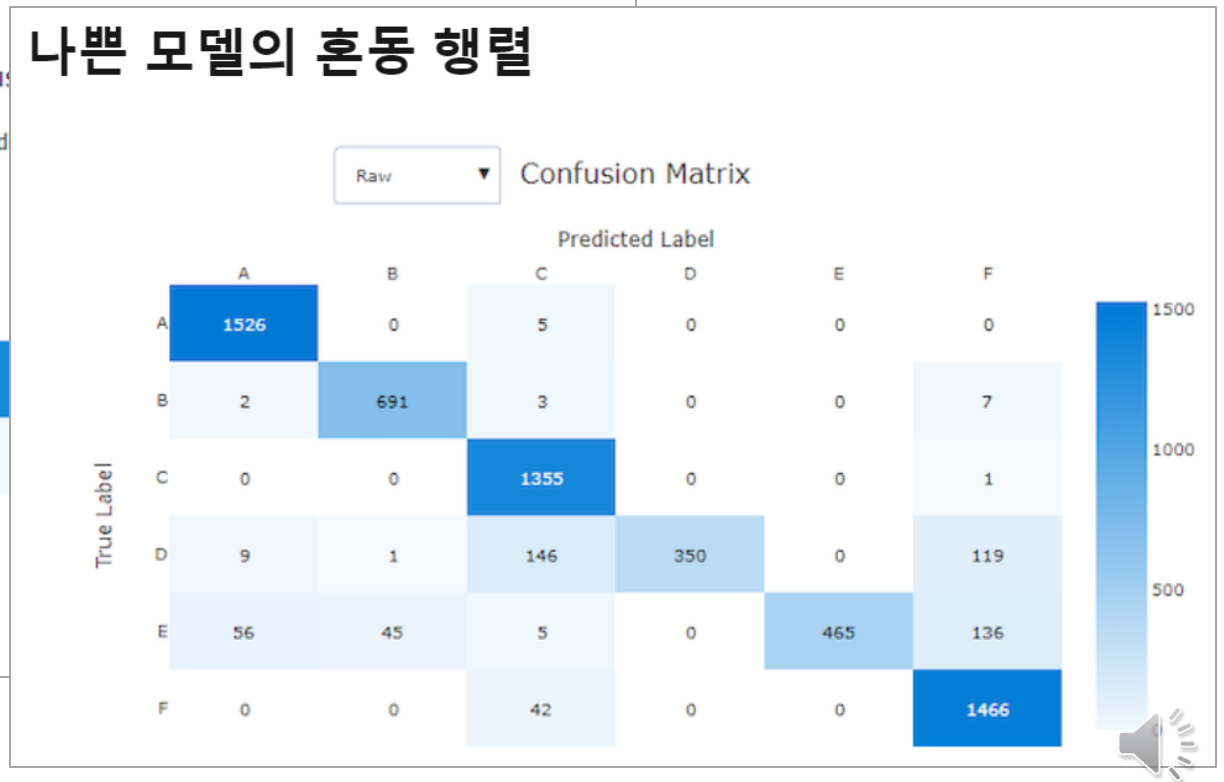
- *k*-겹 교차 검증을 알고리즘 : for 문을 이용한 코드

```
>accuracy=accuracy+(t[1, 1]+t[2, 2]+t[3, 3])/length(test_list) # 정확률&누적
```

좋은 모델의 혼동 행렬



나쁜 모델의 혼동 행렬



10.4 교차 검증

- caret 라이브러리를 이용한 코드

Console C:/RSources/

Console C:/RSources/

5-겹 교차 검증

```
> # caret 라이브러리로 정확률 교차 검증
> control = trainControl(method = 'cv', number = 5)
> f = train(Species~., data = iris, method = 'rf', metric = 'Accuracy',
  trControl = control)
> confusionMatrix(f)
Cross-Validated (5 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

            Reference
Prediction   setosa versicolor virginica
  setosa       33.3        0.0       0.0
  versicolor    0.0       31.3       2.7
  virginica     0.0        2.0      30.7

 Accuracy (average) : 0.9533
```

Thank you


Africa
Americas
Asia
Europe
Oceania
pop
2.50e+08
5.00e+08
7.50e+08
1.00e+09
1.25e+09
gdpPercap