[R] R에서 교차검증을 위한 데이터 셋 분리방법 3가지 — 나무늘보의 개발 블로그

노트북:	blog
만든 날짜:	2020-10-06 오후 4:46
URL:	https://continuous-development.tistory.com/56?category=793392

R

[R] R에서 교차검증을 위한 데이터 셋 분리방법 3가지

2020. 8. 7. 17:29 수정 삭제 공개

교차검증을 위한 데이터셋 분리에는 3가지 방법이 있습니다.

1. 단순 임의 추출

2.K-Fold 방식

3.Hold - Out방식

이 세 가지를 R을 통해 예제를 보며 진행하겠습니다.

```
#교차 검증을 위한 데이터셋 분리 방법
#1.sample()
#2.K-Fold 방식
#3.Hold-Out 방식


#1.단순임의 추출
#

#비복원 추출
sample(1:10,5)

#복원추출
sample(1:10,replace = T)
```

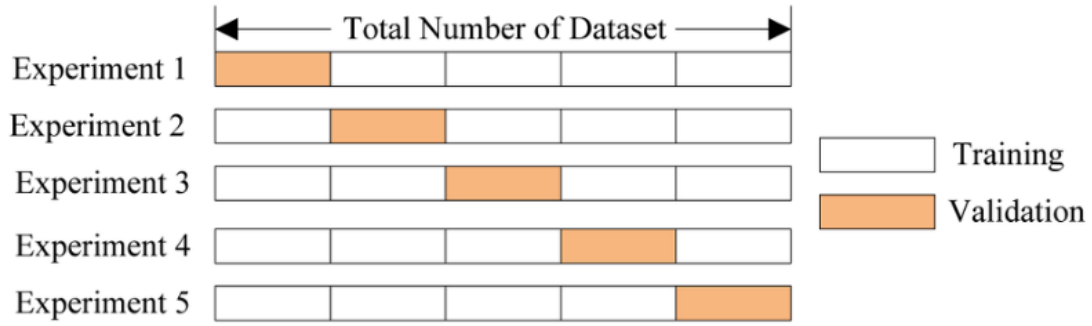
1. 단순 임의 추출

sample을 사용해서 복원/비 복원 추출을 한다. 이 작업을 통해 데이터셋을 만든다. 별로 추천하지 않는 형태이다. 말 그대로 임의 추출이기 때문에 데이터가 한쪽으로 몰릴 수도 있다.

```
> #비복원 추출
> sample(1:10,5)
[1]  9  5  6 10  7
> #복원추출
> sample(1:10,replace = T)
 [1] 10  2  1  6  6  4  4  5 10  8
>
```

2.k-Fold 교차방식

k개의 fold를 만들어서 진행하는 교차검증이다.



이런 방식으로 모든 데이터를 한 번씩 테스트 셋으로 사용되게 하는 기법이다.
가장 추천하는 방법이다.
이 방법을 통해 여러가지 정확도를 얻게 되고 그걸 평균으로 값을 내거나 또는 각각의 케이스에 따른 경우를 가져갈 때도 있다.

패키지를 설치한다.

```
#2.K-Fold 방식
install.packages("cvTools")
library(cvTools)
```

#cvFolds(n , k , r)

-n개의 관찰을 K겹 교차 검증의 R회 반복으로 분할한다.

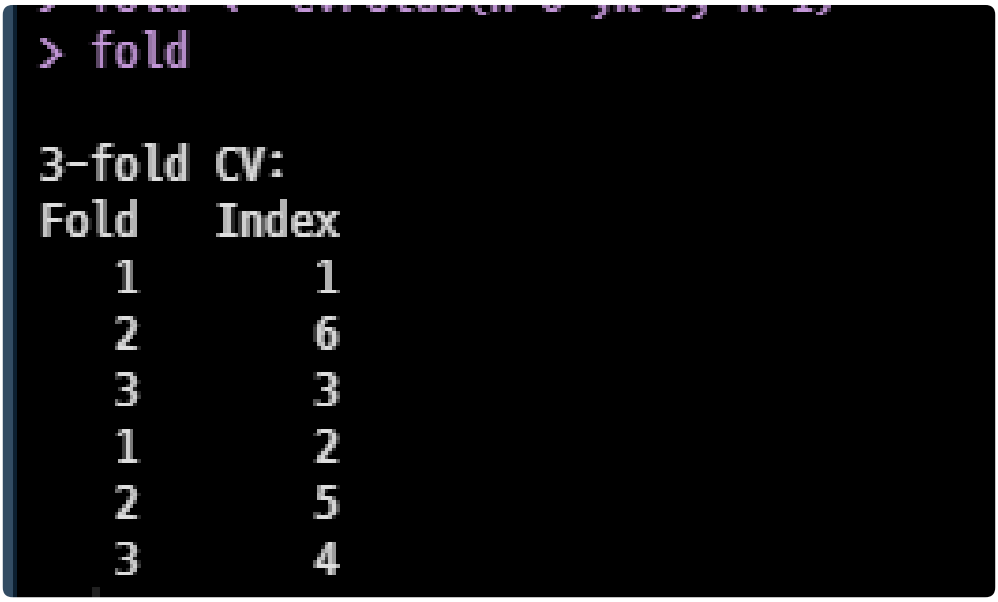
아래에서는 수치는 6까지고 k-fold는 3까지이다. 그리고 반복은 1이다

```
#수치는 6개였고 k-fold값은 3까지였고 re는 1이다.
fold ← cvFolds(n=6 ,K=3, R=1)
fold


str(fold)
fold$subsets #데이터
fold$which   #index값
fold$subsets[fold$which==1,1] #index값이 1인 첫번째 값들
fold$subsets[fold$which==2,1]
fold$subsets[fold$which==3,1]

```

fold에서 subsets는 데이터 값 / which는 fold의 인덱스 값을 나타낸다.



```
> fold

3-fold CV:
Fold   Index
   1       1
   2       6
   3       3
   1       2
   2       5
   3       4
```

```
> fold$subsets #데이터
     [,1]
[1,]    1
[2,]    6
[3,]    3
[4,]    2
[5,]    5
[6,]    4
> fold$which   #index값
[1] 1 2 3 1 2 3
> fold$subsets[fold$which==1,1] #index값이 1인 첫번째 값들
[1] 1 2
> fold$subsets[fold$which==2,1]
[1] 6 5
```

예제

```
> irisFold ← cvFolds(n=nrow(iris), K=3, R=2)
> irisFold

Repeated 3-fold CV with 2 replications:
Fold      1   2
   1     96 139
   2    137  11
   3    133  44
   1     75  17
   2      3  93
   3     43  58
   1     57  65
   2     97 148
   3     63  39
   1    122  56
   2     29  52
   3    119 137
```

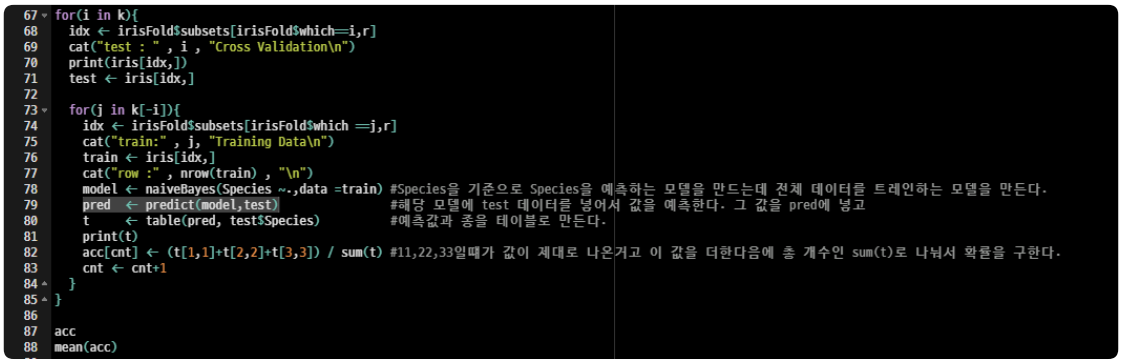
```
   2     136 125
   3      44 142
> irisFold$subsets[fold$which==1,1]
 [1]  96  75  57 122  68  69  80  79  40 129  76  99 1
[46]  94  48  46  55 104
```

이런것들 하나하나가 데이터 셋이다. 여기서는 k가 3이면 3가지의 데이터 셋이 있다.
이렇게 데이터셋을 분리할 수 있다.

이 3가지 데이터셋을 통해 naiveBayes 알고리즘을 사용해 분류를 해보자.

```
47
48  # classification - naiveBayes 알고리즘
49   #텍스트마이닝에 많이쓰는 알고리즘이다 스팸에 많이 사용된다.
50  # 목표변수가 범주형
51
52
53  acc ← numeric()
54  cnt ← 1
55  r   ← 1
56  k   ← 1:3
57  k[-1]
```

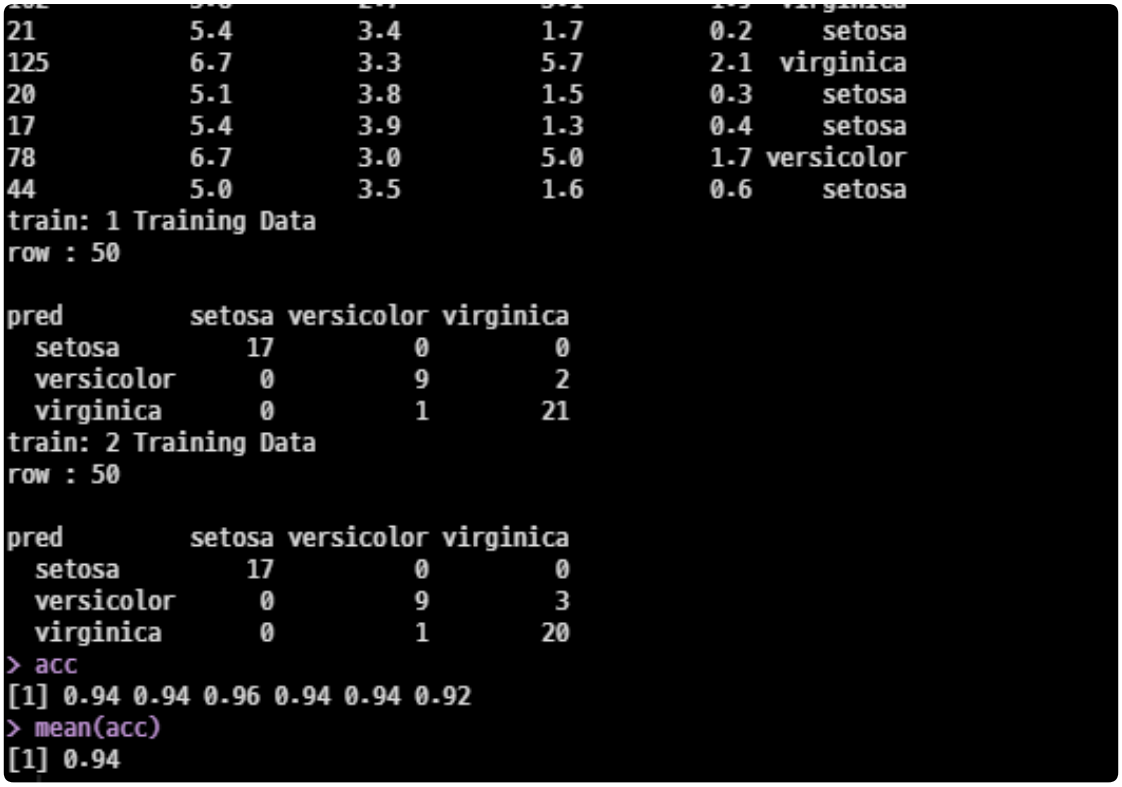
간단하게 설명하면 첫 번째 for문은 iris의 i번째 데이터 셋을 돌리고 두 번째 for문은 i가 아닌 나머지 데이터셋을 넣고 돌린다.
그렇게 i 번째 데이터 셋을 test 데이터 셋으로 i가 아닌 나머지 데이터 셋을 train 데이터셋으로 만들고
train 데이터셋을 통해 모델을 만들고 그 모델과 test 데이터셋으로 예측값 pred를 만든다.
그리고 나온 값들로 이 모델의 정확성을 파악한다.



```
for(i in k){
  idx <- irisFold$subsets[irisFold$which==i,r]
  cat("test : " , i , "Cross Validation\n")
  print(iris[idx,])
  test <- iris[idx,]

  for(j in k[-i]){
    idx <- irisFold$subsets[irisFold$which ==j,r]
    cat("train:" , j, "Training Data\n")
    train <- iris[idx,]
    cat("row :" , nrow(train) , "\n")
    model <- naiveBayes(Species ~.,data =train) #Species을 기준으로 Species을 예측하는 모델을 만드는데 전체 데이터를 트레인하는 모델을 만든다.
    pred  <- predict(model,test)                #해당 모델에 test 데이터를 넣어서 값을 예측한다. 그 값을 pred에 넣고
    t     <- table(pred, test$Species)          #예측값과 종을 테이블로 만든다.
    print(t)
    acc[cnt] <- (t[1,1]+t[2,2]+t[3,3]) / sum(t) #11,22,33일때가 값이 제대로 나온거고 이 값을 더한다음에 총 개수인 sum(t)로 나눠서 확률을 구한다.
    cnt <- cnt+1
  }
}

acc
mean(acc)
```



```
21           5.4         3.4          1.7         0.2     setosa
125          6.7         3.3          5.7         2.1  virginica
20           5.1         3.8          1.5         0.3     setosa
17           5.4         3.9          1.3         0.4     setosa
78           6.7         3.0          5.0         1.7 versicolor
44           5.0         3.5          1.6         0.6     setosa
train: 1 Training Data
row : 50

pred         setosa versicolor virginica
  setosa         17          0         0
  versicolor      0          9         2
  virginica       0          1        21
train: 2 Training Data
row : 50

pred         setosa versicolor virginica
  setosa         17          0         0
  versicolor      0          9         3
  virginica       0          1        20
> acc
[1] 0.94 0.94 0.96 0.94 0.94 0.92
> mean(acc)
[1] 0.94
```

이런 식으로 하면 총 6개의 값이 나온다.

#Hold - Out(교차검증)

데이터셋을 훈련 셋과 테스트 셋으로 분리한다.
createDataPartition(데이터, p=?) p는 퍼센트를 나타낸다. 여기서는 80프로로 데이터셋을 만든다.

```
# Hold - Out 교차검증 - 데이터셋을 훈련셋과 테스트셋으로 분리
install.packages("caret")
library(caret)

# createDataParititon()

set.seed(300)
?createDataPartition
hold_out_train ← createDataPartition(iris$Species,p=.8) #80퍼센트의 트레인 데이터를 만든다.
names(hold_out_train)
hold_out_train$Resample1
```

```
> hold_out_train ← createDataPartition(iris$Species,p=.8)
> hold_out_train
$Resample1
  [1]   1   2   3   5   6   7   8   9  11  12  13  14  15  16  17  19  21  22  24  25  26  27  28  29  30  31  32  35  36  38  39  40  41  42  43  44  46  47  48  49  51  52  53  54  55
 [46]  56  59  60  61  62  63  64  68  69  71  73  74  76  77  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98 100 101 102 103 104 105 106 107 109 110 111
 [91] 112 113 116 117 118 119 120 121 122 123 124 125 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143 144 146 147

> ?createDataPartition
> names(hold_out_train)
[1] "Resample1"
> hold_out_train$Resample1
  [1]   1   2   3   5   6   7   8   9  11  12  13  14  15  16  17  19  21  22  24  25  26  27  28  29  30  31  32  35  36  38  39  40  41  42  43  44  46  47  48  49  51  52  53  54  55
 [46]  56  59  60  61  62  63  64  68  69  71  73  74  76  77  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98 100 101 102 103 104 105 106 107 109 110 111
 [91] 112 113 116 117 118 119 120 121 122 123 124 125 128 129 130 131 132 133 134 135 136 137 138 139 141 142 143 144 146 147
```

이런 식으로 train data와 test data를 분류할 수 있다.

```
train_iris ←iris[hold_out_train$Resample1,]
test_iris ←iris[-hold_out_train$Resample1,]
```

이 train 데이터를 통해 모델을 만들고 예측값을 구한다. 그다음 데이터를 가져와보면

```
model ← naiveBayes(Species ~.,data =train_iris) #Species을 기준으로 Species을 예측하는 모델을 만드는데 전체 데이터를 트레인하는 모델을 만든다.
pred ← predict(model, test_iris)
t ← table(pred, test_iris$Species)
print(t)
```

```
> t ← table(pred, test_iris$Species)
> print(t)

pred         setosa versicolor virginica
  setosa         10          0         0
  versicolor      0          9         0
  virginica       0          1        10
```

위와 같이 나온다. 이것의 값을 구하면

```
virginica       0          1         10
> acc ← (t[1,1]+t[2,2]+t[3,3]) /sum(t)
> acc
[1] 0.9666667
```

#분류 실습

```
120
121 # --- 분류실습(Naive Bayes Classifier)
122 # 텍스트 분류
123 # 문서를 여러 범주 나누어서 판단하는 알고리즘
124 # 조건부 확률
125 # 10개의 메일중 , 3개는 스팸메일
126 # 그리고 그와 상관없이 free라는 단어를 포함하는 메일이 4개다.
127 # 문제는 free(A)라는 메일이 와 있을때
128 # 그것이 스팸메일(B) 인지 아닌지를 구분해야한다면
129 # 공식 : P(B/A) = P(B) *P(A/B) / P(A)
130 # 1.스팸메일이 오는 확률 :3/10
131 # 2.FREE를 포함하는 메일의 확류 :4:0
132 # 3. 스팸메일주에 포함된 FREE포함 메일 : 2/3
133 P(SPAM/FREE) = P(BSPAM) *P(FREE/SPAM) / P(FREE)
134
135
136
137 iris
138 install.packages("klaR")
139 library(klaR)
140
141 train ← sample(nrow(iris) , 100)
142 naive_mode ← NaiveBayes(Species ~. , data=iris, subset=train)
```

```
> 1-sum(tt[row(tt) == col(tt)] / sum(tt))
[1] 0
> pred ←predict(naive_model, iris[-train,])
> names(pred)
[1] "class"     "posterior"
>
```

```
> table(iris$Species[-train],
+ predict(naive_mode , iris[-train,] ) $class)

             setosa versicolor virginica
  setosa         18          0         0
  versicolor      0         17         0
  virginica       0          0        15
> tt←table(iris$Species[-train],
+ predict(naive_mode , iris[-train,] ) $class)
> sum(tt[row(tt) == col(tt)] / sum(tt))
[1] 1
> 1-sum(tt[row(tt) == col(tt)] / sum(tt))
[1] 0
```

```r

install.packages("ggplot2")
library(ggplot2)


test<- iris[-train,]
test$pred <-predict(naive_model, iris[-train,] )$class


ggplot(test,
       aes(Species,pred, col=Species))+
  geom_jitter(width=0.2, height =0.1, size=2)+
  labs(x= "Species",
       y= "Predict")
```

'**R**' 카테고리의 다른 글□

[R]R을 활용한 상관분석과 회귀분석 - 2□

[R] R을 활용한 크롤링 - 로또 1등 당첨 배출점 크롤링 하기□

[R] R에서 교차검증을 위한 데이터 셋 분리방법 3가지□

[R] R을 활용한 상관분석과 회귀분석 - 1□

[R] R을 통한 텍스트마이닝에서 워드클라우드 까지□

[R] R로 하는 비정형 데이터 처리 (facebook 데이터를 통한 긍정/부정 나누기)□

Hold out 교차검증 k-fold K-Fold 검증 R 교차검증 교차검증 단순임의추출

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.