Ch7 EDA와 data정제

데이터 전처리(data preprocessing) 과정

1. EDA란?

탐색적 자료분석(Exploratory Data Analysis):
- 수집한 자료를 다양한 각도에서 관찰하고 이해하는 과정
- 그래프나 통계적 방법을 이용하여 자료를 직관적으로 파악하는 과정

1.1 EDA 필요성

자료의 분포와 통계 파악 → 자료의 특성 이해
잠재적인 문제 발견 → 기존의 가설 수정 또는 새로운 방향의 가설 설정

1.2 EDA 과정

단계별 EDA 수행과정
1. 분석의 목적과 변수의 특징 확인
2. 자료 확인 및 전처리: 결측치, 이상치
3. 자료의 각 변수 관찰: 통계조사, 시각화
4. 변수 간의 관계에 초점을 맞춰 패턴 발견: 상관관계, 시각화 도구로 변수간의 패턴 발견

2. 수집자료 이해

수집된 자료를 이해하는 단계
예) 자료구조, 관측치의 길이, 변수 구성, 각 변수의 의미, 측정 방법, 척도 유형, 명세서와 동일하게 코딩되었는지 확인

2.1 데이터 셋 보기

데이터의 분포 현황을 통해 데이터의 유형과 결측치(NA), 극단치(outlier)등의 데이터를 발견
결측치: 응답자의 회피와 응답할 수 없는 상황(예, 여성의 경우 군필 항목, 남성의 출산여부 항목)에서 주로 발생
극단치: 데이터의 수집과 입력과정에서의 실수로 발생

데이터셋 전체를 볼 수 있는 함수: print()함수, View()함수
print()함수: console창으로 데이터 표시
View()함수: 별도의 데이터 뷰어 창을 통해 전체 데이터를 테이블 양식으로 출력

실습 (실습용 데이터 가져오기)

```
getwd()
setwd("C:/Rwork/")
dataset <- read.csv("dataset.csv", header = T)
dataset
```

(전체 데이터 보기)

```
print(dataset)
View(dataset)
```

print()함수, View()함수 사용

실습 (데이터의 앞부분과 뒷부분 보기)

head(dataset)
tail(dataset)

head()함수, tail()함수 사용

[표 7.1] 'dataset.csv"데이터 셋의 변수(컬럼) 구성

2.2 데이터 셋 구조 보기

데이터 셋의 구조를 확인하는 함수: names(), attributes(), str()함수

names()함수: 데이터 셋의 컬럼명 조회
attributes(): 열과 행 이름 및 자료구조 정보
str()함수: 자료구조, 관측치, 칼럼명과 자료형을 동시에 확인

실습 (데이터 셋 구조 보기)

```
names(dataset)
attributes(dataset)
str(dataset)
```

2.3 데이터셋 조회

데이터 셋에 포함된 특정 변수의 내용을 조회하는 방법

데이터프레임을 데이터 셋으로 구성한 경우 특정 변수에 접근하기 위해서 '$'기호를 사용하여
"객체$변수" 형식 사용

실습 (다양한 방법으로 데이터 셋 조회하기)

1단계: 데이터 셋에서 특정 변수 조회

```
dataset$age
dataset$resident
length(dataset$age)
```

2단계: 특정 변수의 조회 결과를 변수에 저장

```
x <- dataset$gender
y <- dataset$price

x
y
```

3단계: 산점도 그래프로 변수 조회

```
plot(dataset$price)
```

4단계: 컬럼명을 사용하여 특정 변수 조회

```
dataset["컬럼명"]

dataset["gender"]
dataset["price"]
```

5단계: index를 사용하여 특정 변수 조회

```
dataset[2]
dataset[6]
dataset[3, ]
dataset[ , 3]
```

6단계: 2개 이상의 컬럼 조회

```
dataset[c("job", "price")]
dataset[c(2, 6)]
dataset[c(1, 2, 3)]
dataset[c(2, 4:6, 3, 1)]
```

7단계: 특정 행/열을 조회

```
dataset[ , c(2:4)] # 2-4열의 모든 행 조회
dataset[c(2:4), ]
dataset[-c(1:100), ] #1-100행 제외한 나머지 행의 모든 열 조회
```

3. 결측치 처리

결측치 항목의 최대 자리수 만큼 숫자 9를 채워 부호화
하이픈(-)으로 해당 항목을 채워 놓음

결측치를 제거한 후 유효한 자료만을 대상으로 연산: na.rm속성, na.omit()함수

결측치 처리 방법

1) 결측치를 제거
2) 다른값으로 대체

Ex. kNN모델에서 결측치가 존재하는 경우 값이 왜곡되는 현상 발생 → 결측치가 포함된 관측치 제거

결측치를 포함한 관측치를 제거하면 해당 정보가 손실되므로 결측치를 0이나 평균으로 대체하는 방법 고려

3.1 결측치 확인

summary()함수를 이용하여 특정 변수의 결측치 확인
sum(), mean()함수에 결측치가 포함된 경우 'NA'가 출력

실습 (summary()함수를 사용하여 결측치 확인)

```
summary(dataset$price)
sum(dataset$price)
```

NA 개수 출력

3.2 결측치 제거

결측치 제거를 위하여 함수의 속성을 이용하거나 결측치 제거 함수를 사용

실습 (sum()함수의 속성을 이용하여 결측치 제거)

```
sum(dataset$price, na.rm = T)
```

na.rm = T 속성 적용

실습 (결측치 제거 함수를 이용하여 결측치 제거)

```
price2 <- na.omit(dataset$price)
sum(price2)
length(price2)
```

na.omit()_함수는 특정 칼럼의 결측치를 제거

3.3 결측치 대체

결측치를 포함한 관측치를 유지하기 위한 벙법:

1) 0으로 대체
2) 평균으로 대체

실습 (결측치를 0으로 대체)

```
x <- dataset$price
x[1:30]
dataset$price2 = ifelse(!is.na(x), x, 0)
dataset$price2[1:30]
```

실습 (결측치를 평균으로 대체)

```
x <- datset$price
x[1:30]
dataset$price3 = ifelse(!is.na(x), x, round(mean(x, na.rm = TRUE), 2))
dataset$price3[1:30]
dataset[c('price', 'price2', 'price3')]
```

결측치, 결측치를 0으로 대체, 결측치를 평균값으로 대체한 컬럼 3개 확인

4. 극단치 처리

극단치(outlier): 정상적인 분포에서 벗어난 값

예, 나이의 분포가 0~100세 사이의 분포인데 -2 또는 250과 같은 비정상적인 수치

4.1 범주형 변수 극단치 처리

명목척도 같은 범주형 변수

실습 (범주형 변수의 극단치 처리)

```
table(dataset$gender)
pie(table(dataset$gender))
```

subset()함수: 데이터 셋의 특정 변수를 대상으로 조건식에 해당하는 레코드(행) 추출
형식: subset(데이터프레임, 조건식)

실습 (subset()함수를 사용하여 데이터 정제)

```
dataset <- subset(dataset, gender == 1 | gender == 2)
dataset
length(dataset$gender)
pie(table(dataset$gender))
pie(table(dataset$gender), col = c("red", "blue"))
```

4.2 연속형 변수의 극단치 처리

연속된 데이터를 갖는 변수들을 대상으로 극단치 확인하고 데이터 정제

실습 (연속형 변수의 극단치 보기)

```
dataset <- read.csv("dataset.csv", header = T)
dataset$price
length(dataset$price)
plot(dataset$price)
summary(dataset$price)
```

산점도 또는 summary()에서 제공되는 요약통계량을 통해 극단치 처리 방법 결정

실습 (price 변수의 데이터 정제와 시각화)

```
dataset2 <- subset(dataset, price >= 2 & price <= 8)
length(dataset2$price)
stem(dataset2$price)
```

stem()함수를 사용하여 정보를 줄기와 잎 형태로 도표화

실습 (age 변수의 데이터 정제와 시각화)

```
# age 변수에서 NA 발견
summary(dataset2$age)
length(dataset2$age)

# age 변수 정제(20 ~ 69)
dataset2 <- subset(dataset2, age >= 20 & age <= 69)
length(dataset2)
```

```
# box 플로팅으로 평균연령 분석
boxplot(dataset2$age)
```

boxplot()함수: 정제된 결과를 상자 그래프로 시각화

4.3 극단치를 찾기 어려운 경우

범주형 변수는 극단치 발견이 상대적 쉬움.
연속형 변수는 극단치 찾기가 어려울 수 있음.
➔ boxplot과 통계 이용하여 극단치 찾기

실습 (boxplot과 통계를 이용한 극단치 처리하기)
변수 상/하위 0.3%를 극단치로 설정

```
# boxplot로 price의 극단치 시각화
boxplot(dataset$price)

# 극단치 통계 확인
boxplot(dataset$price)$stats

# 극단치를 제거한 서브 셋 만들기
dataset_sub <- subset(dataset, price >= 2 & price <= 7.9)
summary(dataset_sub$price)
```

5. 코딩 변경

코딩 변경: 최초 코딩 내용을 용도에 맞게 변경하는 작업
코딩 변경 목적: 데이터의 가독성, 척도 변경, 역 코딩

5.1 가독성을 위한 코딩 변경

일반적으로 데이터는 디지털화하기 위해서 숫자로 코딩
예, 서울:1, 인천:2 등

이러한 코딩 결과를 대상으로 기술통계분석을 수행하면 1과 2의 숫자를 실제 거주지명으로 표현해야 한다.
이를 위해 코딩 변경 작업이 필요

실습 (가독성을 위해 resident 컬럼을 대상으로 코딩 변경)

```
dataset2$resident2[dataset2$resident == 1] <- '1.서울특별시'
dataset2$resident2[dataset2$resident == 2] <- '2.인천광역시'
dataset2$resident2[dataset2$resident == 3] <- '3.대전광역시'
dataset2$resident2[dataset2$resident == 4] <- '4.대구광역시'
dataset2$resident2[dataset2$resident == 5] <- '5.시구군'

# 코딩 변경 전과 변경 후의 칼럼 보기
dataset2[c("resident", "resident2")]

# 실습: 가독성을 위해 job 칼럼을 대상으로 코딩 변경하기
dataset2$job2[dataset2$job == 1] <- '공무원'
dataset2$job2[dataset2$job == 2] <- '회사원'
dataset2$job2[dataset2$job == 3] <- '개인사업'

# 코딩 변경 전과 변경 후의 칼럼 보기
dataset2[c("job", "job2")]
```

5.2 척도 변경을 위한 코딩 변경

나이 같은 연속형 변수를 20대, 30대 같이 범주형 변수로 변경

실습 (나이를 나타내는 age컬럼을 대상으로 코딩 변경하기)

```
dataset2$age2[dataset2$age <= 30] <- "청년층"
dataset2$age2[dataset2$age > 30 & dataset2$age <= 55] <- "중년층"
dataset2$age2[dataset2$age > 55 ] <- "장년층"
head(dataset2)
```

상관관계 분석이나 회귀분석: 연속형 변수가 적합
빈도분석이나 교차분석: 범주형 변수가 적합

5.3 역 코딩을 위한 코딩 변경

만족도 평가를 위해 설문지 문항을 5점 척도인 (1)매우만족, (2)만족, (3)보통, (4),불만족, (5) 매우 불만족 형태로 작성된 경우 이를 역순으로 변경해야 한다

역코딩(inverse coding): 순서를 역순으로 변경

예, 만족도 컬럼을 대상으로 1~5순서로 코딩된 값을 5~1순서로 역코딩을 하기 위해 '6-현재값' 형식으로 수식 적용

실습 (만족도를 긍정순서로 역코딩)

```
survey <- dataset2$survey
csurvey <- 6 - survey
csurvey

dataset2$survey <- csurvey
head(dataset2)
```

6. 변수 간의 관계 분석

척도별로 시각화하여 데이터의 분포형태를 분석

명목척도와 서열척도의 범주형 변수와 비율척도의 연속형 변수간의 탐색적 분석 위주

6.1 범주형 vs. 범주형

명목척도 또는 서열척도 같은 범주형 변수를 대상으로 시각화하여 컬럼 간의 데이터 분포형태 파악

실습 (범주형 vs 범주형 데이터 분포 시각화)

1단계: 데이터 가져오기

```
setwd("C:/Rwork/ ")
new_data <- read.csv("new_data.csv", header = TRUE)
str(new_data)
```

2단계: 코딩 변경된 거주지역(resident) 컬럼과 성별(gender) 컬럼을 대상으로 빈도수 구하기

```
resident_gender <- table(new_data$resident2, new_data$gender2)
resident_gender
gender_resident <- table(new_data$gender2, new_data$resident2)
gender_resident
```

3단계: 성별(gender)에 따른 거주지역(resident)의 분포 현황 시각화

```
barplot(resident_gender, beside = T, horiz = T,
        col = rainbow(5),
```

```
        legend = row.names(resident_gender),
        main = '성별에 따른 거주지역 분포 현황')
```

4단계: 거주지역(resident)에 따른 성별(gender)의 분포 현황 시각화

```
barplot(gender_resident, beside = T,
        col = rep(c(2, 4), 5), horiz = T,
        legend = c("남자", "여자"),
        main = '거주지역별 성별 분포 현황')
```

6.2 연속형 vs. 범주형

연속형 변수(나이)와 범주형 변수(직업 유형)를 대상으로 시각화하여 컬럼 간의 데이터 분포 형태 파악

실습 (연속형 vs 범주형 데이터의 시각화)

1단계: lattice 패키지 설치와 메모리 로딩 및 데이터 준비

```
install.packages("lattice")
library(lattice)
```

lattice패키지: 고급 시각화 분석에서 사용되는 패키지. Ch8 고급 시각화 분석에서 해당 패키지의 특징과 관련 함수에 대해 설명

2단계: 직업 유형에 따른 나이 분포 현황

```
densityplot(~ age, data = new_data,
            groups = job2,
            # plot.points = T: 밀도, auto.key = T: 범례)
            plot.points = T, auto.key = T)
```

6.3 연속형 vs. 범주형 vs. 범주형

연속형 변수(구매비용), 범주형 변수(성별), 범주형 변수(서열)을 대상으로 시각화하여 컬럼 간의 데이터 분포 형태 파악

실습 (연속형 vs 범주형 vs 범주형 데이터 분포 시각화)

1단계: 성별에 따른 직급별 구매비용 분석

```
densityplot(~ price | factor(gender),
            data = new_data,
            groups = position2,
            plot.points = T, auto.key = T)
```

densityplot()함수 사용
Where 속성
factor(gender2): 격자를 만들어주는 컬럼을 지정하는 속성(성별로 격자 생성)
groups = position2: 하나의 격자에서 그룹을 지정하는 속성(직급으로 그룹 생성)

2단계: 직급에 따른 성별 구매비용 분석

```
densityplot(~ price | factor(position2),
            data = new_data,
            groups = gender2,
            plot.points = T, auto.key = T)
```

6.4 연속형(2개) vs. 범주형(1개)

연속형 변수 2개(구매비용, 나이)와 범주형 변수 1개(성별)을 대상으로 시각화하여 칼럼 간의 데이터분포형태 파악

실습 (연속형(2개) vs 범주형(1개) 데이터 분포 시각화)

```
xyplot(price ~ age | factor(gender2),
       data = new_data)
```

xyplot()함수: 산점도 사용

7. 파생변수

파생변수: 코딩된 데이터를 대상으로 분석에 이용하기 위해 만들어진 새로운 변수

파생변수 생성 방법:

1) 사칙연산을 이용: ex. 총점, 평균 컬럼 생성
2) 1:1관계로 나열하는 방법: 본 교재내 설명

[표 7.2] 3개의 컬럼을 갖는 테이블 구조의 데이터 셋

여기서 주거환경의 컬럼은 주택, 빌라, 아파트, 오피스텔의 4가지 범주를 갖는 컬럼
고객의 주거환경을 독립변수로 사용하기 위해서 1:N관계(개인 아이디에 4가지 범주를 갖는 주거환경)를 1:1관계(개인 아이디에 4가지 범주를 모두 나열하는 방식)로 변수를 나열하여 [표7.3]과 같이 파생변수를 생성

주거환경을 고객의 아이디와 1:1관계로 데이터 셋의 구조를 변경하면 컬럼 수는 늘어나지만 다양한 분석 방법에서 이용 가능

7.1 파생변수 생성을 위한 테이블 구조

[그림 7.2] 파생변수 생성을 위한 테이블 구조
고객정보 – 지불정보, 반품정보 등 1:N관계

7.2 더미 형식으로 파생변수 생성

더미(dummy) : 특정 컬럼을 명목상 두가지 상태(0과 1)로 범주화 하여 나타내는 형태

여기서 1:N 관계를 갖는 고객정보 테이블의 주거환경 컬럼을 대상으로 '주택유형'(단독주택과 다세대주택)과 '아파트유형'(아파트와 오피스텔)의 두가지 상태로 더미(dummy)화하여 파생변수 생성

실습 (파생변수 생성하기)

1단계: 데이터 파일 가져오기

```
setwd("C:/Rwork/ ")
user_data <- read.csv("user_data.csv", header = T)
head(user_data)
table(user_data$house_type)
```

2단계: 더미변수 생성

단독주택 or 다세대 주택이면 0, 아파트 or 오피스텔이면 1

```
house_type2 <- ifelse(user_data$house_type == 1 |
                              user_data$house_type == 2, 0 , 1)
house_type2[1:10]
```

3단계: 파생변수 추가

```
user_data$house_type2 <- house_type2
head(user_data)
```

7.3 1:1관계로 파생변수 생성

지불정보(pay_data)데이블의 고객식별번호(user_id)와 상품 유형(product_type)테이블의 고객식별번호(user_id) 그리고 지불방식(pay_method)간의 1:N관계를 1:1관계로 변수를 나열하여 파생변수를 생성

실습 (1:N관계를 1:1 관계로 파생변수 생성하기)

1단계: 데이터 파일 가져오기

```
pay_data <- read.csv("pay_data.csv", header = T)
head(pay_data, 10)
table(pay_data$product_type)
```

2단계: 고객별 상품 유형에 따른 구매금액과 합계를 나타내는 파생변수 생성

```
library(reshape2)
product_price <- dcast(pay_data, user_id ~ product_type,
                       sum, na.rm = T)
head(product_price, 3)
```

dcast()함수를 사용하여 user_id를 행으로 지정, product_type을 열로 지정하여 고객별로 구매한 상품 유형에 따라서 구매금액의 합계를 계산하여 파생변수 생성

3단계: 컬럼명 수정

```
names(product_price) <- c('user_id', '식표품(1)', '생필품(2)',
                          '의류(3)', '잡화(4)', '기타(5)')
head(product_price)
```

가독성 향상을 위해 컬럼명을 추가

실습 (고객식별번호(user_id)에 대한 지불유형(pay_method)의 파생변수 생성)

1단계: 고객별 지불유형에 따른 구매상품 개수를 나타내는 파생변수 생성

```
pay_price <- dcast(pay_data, user_id ~ pay_method, length)
head(pay_price, 3)
```

dcast()함수 사용 user_id은 행에 pay_method는 컬럼에 지정하여 고객별로 지불유형에 따른 구매상품 개수를 파생변수로 생성

2단계: 컬럼명 변경

```
names(pay_price) <- c('user_id', '현금(1)', '직불카드(2)',
                      '신용카드(3)', '상품권(4)')
head(pay_price, 3)
```

7.4 파생변수 합치기

고객정보 테이블에 파생변수를 추가하여 새로운 형태의 데이터프레임을 생성

실습 (고객정보(user_data)테이블에 파생변수 추가)

1단계: 고객정보 테이블과 고객별 상품 유형에 따른 구매금액 합계 병합하기

```
library(plyr)
user_pay_data <- join(user_data, product_price, by = 'user_id')
head(user_pay_data, 10)
```

join()함수 사용하여 고객식별번호(user_id)를 기준으로 고객정보 테이블(user_data)과 고객별 상품 유형에 따른 구매금액 합계(product_price)를 하나의 데이터프레임으로 병합

2단계: 고객별 지불유형에 따른 구매상품 개수 병합하기

```
user_pay_data <- join(user_pay_data, pay_price, by = 'user_id')
user_pay_data[c(1:10), c(1, 7:15)]
```

join()함수 사용하여 고객식별번호(user_id)를 기준으로 1단계에서 병합된 데이터프레임에 고객별 지불유형에 따른 구매상품 개수를 추가하여 하나의 데이터프레임으로 병합

실습 (사칙연산으로 총 구매금액 파생변수 생성)

1단계: 고객별 구매금액의 합계(총 구매금액) 계산

```
user_pay_data$총구매금액 <- user_pay_data$`식표품(1)` +
  user_pay_data$`생필품(2)` +
  user_pay_data$`의류(3)` +
  user_pay_data$`잡화(4)` +
  user_pay_data$`기타(5)`
```

2단계: 고객별 상품 구매 총금액 컬럼 확인

```
user_pay_data[c(1:10), c(1, 7:11, 16)]
```

8. 표본추출

샘플링(sampling): 정제한 데이터셋에서 표본으로 사용할 데이터를 추출

8.1 정제 데이터 저장

실습 (정제된 데이터 저장)

```
print(user_pay_data)

setwd("C:/Rwork/ ")
write.csv(user_pay_data, "cleanData.csv", quote = F, row.names = F)

data <- read.csv("cleanData.csv", header = TRUE)
data
```

8.2 표본 샘플링

표본 샘플링: 정제된 데이터를 대상으로 원하는 행(레코드) 수 만큼 임의로 데이터 추출

실습 (표본 추출)

```
# 표본 추출하기
nrow(data)
choice1 <- sample(nrow(data), 30)
choice1

# 50 ~ (data 길이) 사이에서 30개 행을 무작위 추출
choice2 <- sample(50:nrow(data), 30)
choice2

# 50~100 사이에서 30개 행을 무작위 추출
choice3 <- sample(c(50:100), 30)
choice3

# 다양한 범위를 지정하여 무작위 샘플링
choice4 <- sample(c(10:50, 80:150, 160:190), 30)
choice4
```

2단계: 샘플링 데이터로 표본추출

```
data[choice1, ]
```

* sample()함수에 의해서 추출된 결과는 관측치 기준이 아니라 관측치를 추출할 수 있는 행 번호기준으로 무작위(random) 추출됨

실습 (iris 데이터 셋을 대상으로 7:3 비율로 데이터 셋 생성)

1단계: iris 데이터 셋의 관측치와 컬럼 수 확인

```
data("iris")
dim(iris)
```

2단계: 학습 데이터(70%), 검정 데이터(30%)비율로 데이터 셋 구성

```
idx <-sample(1:nrow(iris), nrow(iris) * 0.7)
training <- iris[idx, ]
testing <- iris[-idx, ]
dim(training)
```

8.3 교차 검정 샘플링

전통적인 검정방식(hold-out): 학습데이터와 검정데이터를 7:3 비율로 구성하여 학습데이터로 모델을 생성하고 검정데이터로 모델을 평가

교차검정: 동일한 데이터 셋을 N등분하여 N-1개의 학습데이터로 모델을 생성하고 나머지 1개를 검정데이터로 이용하여 모델을 평가하는 방식

1) Cross-Validation:
1~n개의 데이터를 랜덤(무작위)하게 n등분하여, 데이터를 Training/Validation으로 나눈 다음 교차하여 확인하는 방법

- 전체 데이터 셋을 동일한 크기를 가진 2 개의 집합으로 분할하여 training set, validation set 을 만듭니다.
- 영향력이 큰 관측지가 어느 set 에 속하느냐에 따라 MSE 가 달라집니다.
- 관측치의 일부만 train 에 속하여 높은 bias 를 갖습니다.

2) K-Fold Cross Validation:
데이터를 데이터를(Random)로 섞은 후 K등분한것중 하나를 검정(Validaton) Set으로사용하는방법

- 전체 데이터 셋을 k 개의 그룹으로 분할하여 한 그룹은 validation set, 나머지 그룹은 train set 으로 사용합니다.
- k 번 fit 을 진행하여 k 개의 MSE 를 평균 내어 최종 MSE 를 계산합니다.
- LOOCV 보다 연산량이 낮습니다.
- 중간 정도의 bias 와 variance 를 갖습니다.

3) LOOCV(Leave-One-Out Cross-Validation):
데이터 중 하나만을 검정(Validation) Set으로 두고, 나머지를 학습(Training) Set으로 모델에 적합시키는 방법. 자료가n개인경우, 위 과정을 n번 반복후 결과치들의 평균을 도출하여 사용함

- n 번 fitting 을 진행하고, n 개의 MSE 를 평균하여 최종 MSE 를 계산합니다.
- n-1 개 관측값을 train 에 사용하므로 bias 가 낮습니다.
- overfitting 되어 높은 variance 를 갖습니다.
- n 번 나누고 n 번 fit 하므로 랜덤성이 없습니다.
- n 번 fit 을 진행하므로 expensive 합니다.

K겹 교차 검정 데이터 셋 생성 알고리즘 사용
Where
K겹: K겹의 회수만큼 모델을 평가

K겹 교차 검정 데이터 셋 생성 알고리즘 사용
1) K개로 데이터를 분할(D1, D2, ..., Dk)하여 D1은 검정데이터, 나머지는 학습데이터 생성
2) 검정데이터의 위치를 하나씩 변경하고, 나머지 데이터를 학습데이터로 생성
3) 위의 1단계와 2단계의 과정을 K번 만큼 반복

예시)
[표 7.7] K=3인 경우 교차 검정 데이터 셋 구성

K-fold	검정	학습
K=1,	D1	D2, D3
K=2	D2	D1, D3
K=3	D3	D1, D2

실습 (데이터 셋을 대상으로 K겹 교차 검정 데이터 셋 생성)

1단계: 데이터프레임 생성

```
name <- c('a', 'b','c', 'd', 'e', 'f')
score <- c(90, 85, 99, 75, 65, 88)
df <- data.frame(Name = name, Score = score)
```

2단계: 교차 검정을 위한 패키지 설치

```
install.packages("cvTools")
library(cvTools)
```

cvTools 패키지 설치

cvFolds()함수: K겹 교차 검정 데이터 셋을 생성

형식: cvFolds(n, K=5, R=1, type=c("random", "consecutive", "interleaved")
Where
n: 데이터의 크기
K: K겹 교차 검증
R: R회 반복

3단계: K겹 교차 검정 데이터 셋 생성

```
cross <- cvFolds(n = 6, K = 3, R = 1, type = "random")
cross
```

4단계: K겹 교차 검정 데이터 셋 구조 보기

```
str(cross)
cross$which
```

K겹 교차 검정 데이터 셋의 구조:
5개의 key로 구성된 List자료구조
결과에서 which는 Fold의 결과를 vector형태로 보관
subsets는 index의 결과를 matrix형태로 보관

5단계: subsets 데이터 참조하기

```
cross$subsets[cross$which == 1, 1]
cross$subsets[cross$which == 2, 1]
cross$subsets[cross$which == 3, 1]
```

실제 관측치의 행 번호를 가지고 있는 subsets의 데이터는which를 이용하여 접근 가능

6단계: 데이터프레임의 관측치 적용

```
r = 1
K = 1:3
for(i in K) {
```

```
    datas_idx <- cross$subsets[cross$which == i, r]
    cat('K = ', i, '검정데이터 ₩n')
    print(df[datas_idx, ])

    cat('K = ', i, '훈련데이터 ₩n')
    print(df[-datas_idx, ])
}
```

'df[-datas_idx,]는 검정데이터를 제외한 나머지 균등분할 데이터를 이용하여 학습데이터를 생성

연습문제 풀기