## K Nearest Neighbor

#### K Nearest Neighbor Algorithm(kNN 알고리즘, k 최근접 이웃 알고리즘)

분류 분석(Classification)은 지도 학습(Supervised Learning) 중 하나로, 데이터가 어떤 카테고리에 속할 지 예측하는 기법이다. 그 중에서 kNN 알고리즘은 어떤 데이터에 대해 근접해 있는 개의 데이터를 기반으로 해당 데이터를 분류하는 알고리즘이다. kNN 알고리즘은 매우 단순하지만 실용적이고, 다음과 같은 영역에서 사용된 바 있다.

* 안면 인식 및 글자 인식 등의 컴퓨터 비전 분야
* 개인화된 추천
* 특정 단백질이나 질병을 추출하는 데 사용되는 유전자 데이터의 패턴 식별

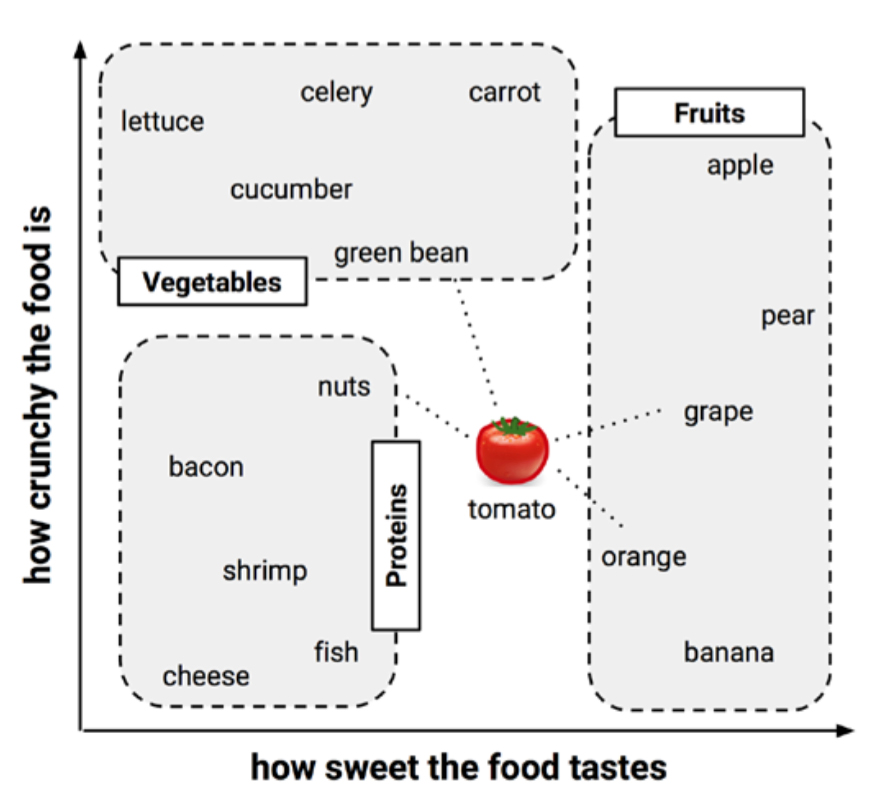
다음의 이미지가 kNN 알고리즘을 잘 나타내고 있다.

두 이미지는 kNN 알고리즘의 예제이다. 첫 번째 이미지는 인 경우인데, 가장 가까운 다섯 개의 데이터를 기반으로 해당 데이터를 분류한다. 이 경우, 파란색 세 개, 회색 두 개이므로 해당 데이터는 파란색으로 분류된다. 하지만 가 달라진다면 해당 데이터는 다른 결과를 얻을 수 있다. 오른쪽 이미지는 인 경우다. 파란색 세 개, 회색 여섯 개이므로 해당 데이터는 회색으로 분류된다.[[1]](#footnote-23) 위 이미지에서 보듯이 kNN 알고리즘은 이해하기 쉬운 알고리즘이지만, 값에 굉장히 민감함을 알 수 있다. 최적의 값을 선택하는 것은 굉장히 데이터에 의존적이다. 또한 닫힌 형태(Closed form)로 식이 존재하기보다는 보통 선험적(heuristic)으로 값을 선택하게 된다. 이 때, 값이 너무 작거나 큰 경우 다음과 같은 문제가 발생한다.

* 값이 매우 작을 때 : 일반적으로 로, 1NN인 경우다. 이 경우 모델이 과적합(overfitting)[[2]](#footnote-24)이 될 가능성이 존재한다.
* 값이 매우 클 때 : 데이터의 구조를 파악하기 어려워진다. 쉽게 말해서, 기존에 더 많이 분류되어 있는 쪽으로 편향된다.

가장 고전적이지만, 손쉽게 값을 선택할 수 있는 방법은 어림잡기(Rule of Thumb)다. 이 법칙은 데이터 전체 개수를 이라고 할 때, 로 둔다. 이 방법이 최선의 방법인지는 실제 알고리즘을 실행한 후에야 알 수 있을 것이다.

다음의 이미지를 통해 kNN 알고리즘을 살펴보도록 하자.



토마토를 중심으로 각 데이터들의 거리를 구해서 가까운 개의 데이터를 기반으로 토마토를 분류한다. 위 이미지에서 로 정의한다면, 토마토는 과일로 분류된다. 이처럼 거리를 구할 때는 여러 가지 척도가 사용될 수 있다. 그 중에서 가장 보편적으로 사용되는 것이 *유클리드 거리(Euclidean distance)*이다. 유클리드 거리는 가장 짧은 일직선의 경로값을 의미한다. 수식으론 다음과 같다.

또한 많이 쓰이는 거리 개념으로는 *맨해튼 거리(Manhattan distance)*가 있다. 수식으론 다음과 같다.

식에서 알 수 있듯이, 각 좌표 사이의 절대값을 모두 더한 값이다. 유클리드 거리와 맨해튼 거리의 비교는 다음의 이미지로 대체한다.

이외에도 코사인 유사도(Cosine similarity), 피어슨 상관계수(Pearson correaltion) 등이 사용된다. 본 실습에서는 간단하게 유클리드 거리를 사용하도록 한다. 다른 거리 개념에 대해서는 추후에 다시 다루도록 한다.

kNN 알고리즘은 대표적인 게으른 학습(lazy learning)이다. 여기서 게으른 학습의 반대는 열정적인 학습(eager learning)이다. 두 학습을 비교하자면 다음과 같다.

* 게으른 학습(lazy learning) : 학습 데이터(training data)를 단순하게 저장하고 테스트 데이터(test data)가 올 때까지 기다렸다가 데이터가 input되면 그 때 모델을 실행하는 경우. 학습 시간보다 예측 시간이 더 걸린다.
* 열정적인 학습(eager learning) : 학습 데이터가 주어지면 테스트 데이터가 input되기 전부터 바로 모델을 생성하는 경우.

kNN 알고리즘의 장단점을 비교하면 다음과 같다.

|  |  |
| --- | --- |
| 장점 | 단점 |
| - 단순하며 효율적이다. - 데이터 분산에 대한 추정을 만들 필요가 없다. - 훈련 단계가 빠르다. | - 특별한 모델을 생성하지 않는다. - 느린 분류 단계 - 많은 메모리가 필요하다. - 명목형 데이터와 결측치에 대해서는 추가적인 처리가 필요하다. |

#### 실습 1 : 유방암 진찰

##### 1단계. 데이터 수집

실습할 데이터는 [UCI 기계 학습 저장소](http://archive.ics.uci.edu/ml/)에서 'Breast Cancer Wisconsin Diagnostic' 데이터셋이다. 디지털화된 유방 덩어리 미세 침흡인물의 이미지로부터의 측정치가 포함되어 있다. 우선 데이터를 로드하도록 하자.

##### 2단계. 데이터 준비와 탐구

wbcd <- read.csv("./data/wisc\_bc\_data.csv", header = TRUE, stringsAsFactors = FALSE)  
  
# 첫 여섯 개의 데이터를 살펴보자.  
head(wbcd)

## id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean  
## 1 87139402 B 12.32 12.39 78.85 464.1  
## 2 8910251 B 10.60 18.95 69.28 346.4  
## 3 905520 B 11.04 16.83 70.92 373.2  
## 4 868871 B 11.28 13.39 73.00 384.8  
## 5 9012568 B 15.19 13.21 97.65 711.8  
## 6 906539 B 11.57 19.04 74.20 409.7  
## smoothness\_mean compactness\_mean concavity\_mean points\_mean  
## 1 0.10280 0.06981 0.03987 0.03700  
## 2 0.09688 0.11470 0.06387 0.02642  
## 3 0.10770 0.07804 0.03046 0.02480  
## 4 0.11640 0.11360 0.04635 0.04796  
## 5 0.07963 0.06934 0.03393 0.02657  
## 6 0.08546 0.07722 0.05485 0.01428  
## symmetry\_mean dimension\_mean radius\_se texture\_se perimeter\_se area\_se  
## 1 0.1959 0.05955 0.2360 0.6656 1.670 17.43  
## 2 0.1922 0.06491 0.4505 1.1970 3.430 27.10  
## 3 0.1714 0.06340 0.1967 1.3870 1.342 13.54  
## 4 0.1771 0.06072 0.3384 1.3430 1.851 26.33  
## 5 0.1721 0.05544 0.1783 0.4125 1.338 17.72  
## 6 0.2031 0.06267 0.2864 1.4400 2.206 20.30  
## smoothness\_se compactness\_se concavity\_se points\_se symmetry\_se  
## 1 0.008045 0.011800 0.01683 0.012410 0.01924  
## 2 0.007470 0.035810 0.03354 0.013650 0.03504  
## 3 0.005158 0.009355 0.01056 0.007483 0.01718  
## 4 0.011270 0.034980 0.02187 0.019650 0.01580  
## 5 0.005012 0.014850 0.01551 0.009155 0.01647  
## 6 0.007278 0.020470 0.04447 0.008799 0.01868  
## dimension\_se radius\_worst texture\_worst perimeter\_worst area\_worst  
## 1 0.002248 13.50 15.64 86.97 549.1  
## 2 0.003318 11.88 22.94 78.28 424.8  
## 3 0.002198 12.41 26.44 79.93 471.4  
## 4 0.003442 11.92 15.77 76.53 434.0  
## 5 0.001767 16.20 15.73 104.50 819.1  
## 6 0.003339 13.07 26.98 86.43 520.5  
## smoothness\_worst compactness\_worst concavity\_worst points\_worst  
## 1 0.1385 0.1266 0.12420 0.09391  
## 2 0.1213 0.2515 0.19160 0.07926  
## 3 0.1369 0.1482 0.10670 0.07431  
## 4 0.1367 0.1822 0.08669 0.08611  
## 5 0.1126 0.1737 0.13620 0.08178  
## 6 0.1249 0.1937 0.25600 0.06664  
## symmetry\_worst dimension\_worst  
## 1 0.2827 0.06771  
## 2 0.2940 0.07587  
## 3 0.2998 0.07881  
## 4 0.2102 0.06784  
## 5 0.2487 0.06766  
## 6 0.3035 0.08284

# 데이터 전체 구조 확인  
str(wbcd)

## 'data.frame': 569 obs. of 32 variables:  
## $ id : int 87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 ...  
## $ diagnosis : chr "B" "B" "B" "B" ...  
## $ radius\_mean : num 12.3 10.6 11 11.3 15.2 ...  
## $ texture\_mean : num 12.4 18.9 16.8 13.4 13.2 ...  
## $ perimeter\_mean : num 78.8 69.3 70.9 73 97.7 ...  
## $ area\_mean : num 464 346 373 385 712 ...  
## $ smoothness\_mean : num 0.1028 0.0969 0.1077 0.1164 0.0796 ...  
## $ compactness\_mean : num 0.0698 0.1147 0.078 0.1136 0.0693 ...  
## $ concavity\_mean : num 0.0399 0.0639 0.0305 0.0464 0.0339 ...  
## $ points\_mean : num 0.037 0.0264 0.0248 0.048 0.0266 ...  
## $ symmetry\_mean : num 0.196 0.192 0.171 0.177 0.172 ...  
## $ dimension\_mean : num 0.0595 0.0649 0.0634 0.0607 0.0554 ...  
## $ radius\_se : num 0.236 0.451 0.197 0.338 0.178 ...  
## $ texture\_se : num 0.666 1.197 1.387 1.343 0.412 ...  
## $ perimeter\_se : num 1.67 3.43 1.34 1.85 1.34 ...  
## $ area\_se : num 17.4 27.1 13.5 26.3 17.7 ...  
## $ smoothness\_se : num 0.00805 0.00747 0.00516 0.01127 0.00501 ...  
## $ compactness\_se : num 0.0118 0.03581 0.00936 0.03498 0.01485 ...  
## $ concavity\_se : num 0.0168 0.0335 0.0106 0.0219 0.0155 ...  
## $ points\_se : num 0.01241 0.01365 0.00748 0.01965 0.00915 ...  
## $ symmetry\_se : num 0.0192 0.035 0.0172 0.0158 0.0165 ...  
## $ dimension\_se : num 0.00225 0.00332 0.0022 0.00344 0.00177 ...  
## $ radius\_worst : num 13.5 11.9 12.4 11.9 16.2 ...  
## $ texture\_worst : num 15.6 22.9 26.4 15.8 15.7 ...  
## $ perimeter\_worst : num 87 78.3 79.9 76.5 104.5 ...  
## $ area\_worst : num 549 425 471 434 819 ...  
## $ smoothness\_worst : num 0.139 0.121 0.137 0.137 0.113 ...  
## $ compactness\_worst: num 0.127 0.252 0.148 0.182 0.174 ...  
## $ concavity\_worst : num 0.1242 0.1916 0.1067 0.0867 0.1362 ...  
## $ points\_worst : num 0.0939 0.0793 0.0743 0.0861 0.0818 ...  
## $ symmetry\_worst : num 0.283 0.294 0.3 0.21 0.249 ...  
## $ dimension\_worst : num 0.0677 0.0759 0.0788 0.0678 0.0677 ...

살펴본 바로는 569명의 환자들의 데이터가 존재하고, 유방암 진단 결과(diagnosis), 세포핵의 반지름(radius), 텍스쳐(texture), 둘레(perimeter), 면적(area), 평활도(smoothness), 다짐도(compactness), 요면(concavity), 요면점(points), 대칭(symmetry), 프랙탈 차원(dimension) 등의 데이터가 포함되어 있다. 이 때 id는 환자 식별자로 유방암 진찰 분류를 하는데에 필요가 없기 때문에 버리도록 한다.

wbcd <- wbcd[-1]

예상하고자 하는 결과는 diagnosis 변수에 포함되어 있으므로, table() 함수를 이용해 살펴보자.

table(wbcd$diagnosis)

##   
## B M   
## 357 212

양성인 경우 357개, 음성인 경우 212개임을 알 수 있다. 하지만 테이블의 레이블이 B와 M으로 나타나있어, 어떤 의미를 갖는지 확실히 알 수 없다. 각각은 양성(Benign), 음성(Malignant)을 나타내므로, factor() 함수를 이용해서 레이블을 붙여주도록 하자.

wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),  
 labels = c("Benign", "Malignant"))

이번엔 결과를 비율로 보도록 하자.

round(prop.table(table(wbcd$diagnosis)) \* 100, digits = 1)

##   
## Benign Malignant   
## 62.7 37.3

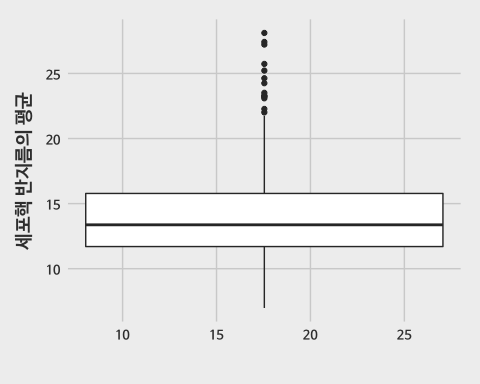
나머지 변수들은 모두 수치이며, 위에서 언급한 10개 특성이 각각 다른 방식으로 수치화되어 있다. 그 중에서 반지름, 면적, 평활도를 자세히 들여다보도록 하자.

summary(wbcd[c("radius\_mean", "area\_mean", "smoothness\_mean")])

## radius\_mean area\_mean smoothness\_mean   
## Min. : 6.981 Min. : 143.5 Min. :0.05263   
## 1st Qu.:11.700 1st Qu.: 420.3 1st Qu.:0.08637   
## Median :13.370 Median : 551.1 Median :0.09587   
## Mean :14.127 Mean : 654.9 Mean :0.09636   
## 3rd Qu.:15.780 3rd Qu.: 782.7 3rd Qu.:0.10530   
## Max. :28.110 Max. :2501.0 Max. :0.16340

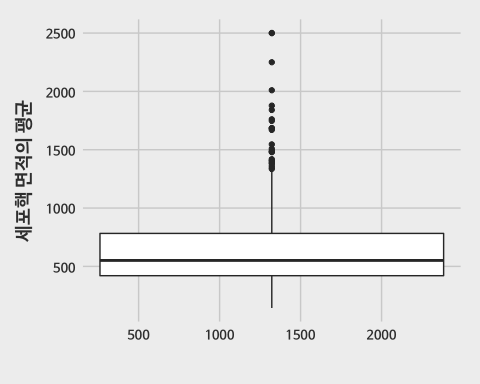
# Boxplot을 그린다.  
qplot(radius\_mean, radius\_mean, data = wbcd, geom = "boxplot", xlab = "",  
 ylab = "세포핵 반지름의 평균")

## Warning: Continuous x aesthetic -- did you forget aes(group=...)?



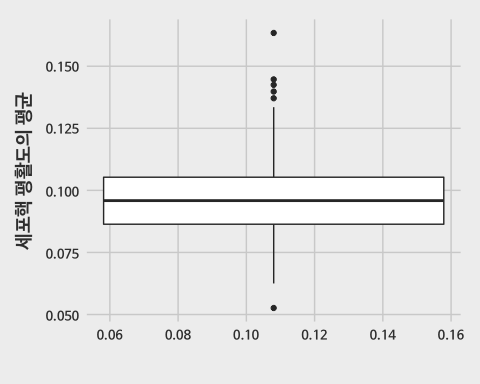
qplot(area\_mean, area\_mean, data = wbcd, geom = "boxplot", xlab = "",  
 ylab = "세포핵 면적의 평균")

## Warning: Continuous x aesthetic -- did you forget aes(group=...)?



qplot(smoothness\_mean, smoothness\_mean, data = wbcd, geom = "boxplot",  
 xlab = "", ylab = "세포핵 평활도의 평균")

## Warning: Continuous x aesthetic -- did you forget aes(group=...)?

 상자그림(boxplot)을 통해 각 데이터가 어떤 분포를 가지고 있는지, 이상치(outlier)가 얼마나 있는지 확인할 수 있다. 또한 측정 범위가 매우 다르기 때문에 이를 정규 범위로 재조정하기 위해 *정규화(Normalization)* 과정을 거쳐야 한다.

normalize <- function(x) {  
 return ((x - min(x)) / (max(x) - min(x)))  
}

위와 같이 정규화 함수를 만들고, 수치적인 데이터를 모두 정규화시켜주기 위해, lapply() 함수를 이용해 일괄적으로 데이터를 정규화하도록 한다. 이 때 laaply()는 결과값이 list로 반환되기 때문에 data frame으로 변환시켜주기 위해 as.data.frame()을 이용한다.

wbcd\_n <- as.data.frame(lapply(wbcd[2:31], normalize))  
summary(wbcd\_n)

## radius\_mean texture\_mean perimeter\_mean area\_mean   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2233 1st Qu.:0.2185 1st Qu.:0.2168 1st Qu.:0.1174   
## Median :0.3024 Median :0.3088 Median :0.2933 Median :0.1729   
## Mean :0.3382 Mean :0.3240 Mean :0.3329 Mean :0.2169   
## 3rd Qu.:0.4164 3rd Qu.:0.4089 3rd Qu.:0.4168 3rd Qu.:0.2711   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## smoothness\_mean compactness\_mean concavity\_mean points\_mean   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.3046 1st Qu.:0.1397 1st Qu.:0.06926 1st Qu.:0.1009   
## Median :0.3904 Median :0.2247 Median :0.14419 Median :0.1665   
## Mean :0.3948 Mean :0.2606 Mean :0.20806 Mean :0.2431   
## 3rd Qu.:0.4755 3rd Qu.:0.3405 3rd Qu.:0.30623 3rd Qu.:0.3678   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## symmetry\_mean dimension\_mean radius\_se texture\_se   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.2823 1st Qu.:0.1630 1st Qu.:0.04378 1st Qu.:0.1047   
## Median :0.3697 Median :0.2439 Median :0.07702 Median :0.1653   
## Mean :0.3796 Mean :0.2704 Mean :0.10635 Mean :0.1893   
## 3rd Qu.:0.4530 3rd Qu.:0.3404 3rd Qu.:0.13304 3rd Qu.:0.2462   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## perimeter\_se area\_se smoothness\_se compactness\_se   
## Min. :0.00000 Min. :0.00000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.04000 1st Qu.:0.02064 1st Qu.:0.1175 1st Qu.:0.08132   
## Median :0.07209 Median :0.03311 Median :0.1586 Median :0.13667   
## Mean :0.09938 Mean :0.06264 Mean :0.1811 Mean :0.17444   
## 3rd Qu.:0.12251 3rd Qu.:0.07170 3rd Qu.:0.2187 3rd Qu.:0.22680   
## Max. :1.00000 Max. :1.00000 Max. :1.0000 Max. :1.00000   
## concavity\_se points\_se symmetry\_se dimension\_se   
## Min. :0.00000 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.03811 1st Qu.:0.1447 1st Qu.:0.1024 1st Qu.:0.04675   
## Median :0.06538 Median :0.2070 Median :0.1526 Median :0.07919   
## Mean :0.08054 Mean :0.2235 Mean :0.1781 Mean :0.10019   
## 3rd Qu.:0.10619 3rd Qu.:0.2787 3rd Qu.:0.2195 3rd Qu.:0.12656   
## Max. :1.00000 Max. :1.0000 Max. :1.0000 Max. :1.00000   
## radius\_worst texture\_worst perimeter\_worst area\_worst   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.1807 1st Qu.:0.2415 1st Qu.:0.1678 1st Qu.:0.08113   
## Median :0.2504 Median :0.3569 Median :0.2353 Median :0.12321   
## Mean :0.2967 Mean :0.3640 Mean :0.2831 Mean :0.17091   
## 3rd Qu.:0.3863 3rd Qu.:0.4717 3rd Qu.:0.3735 3rd Qu.:0.22090   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.00000   
## smoothness\_worst compactness\_worst concavity\_worst points\_worst   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.3000 1st Qu.:0.1163 1st Qu.:0.09145 1st Qu.:0.2231   
## Median :0.3971 Median :0.1791 Median :0.18107 Median :0.3434   
## Mean :0.4041 Mean :0.2202 Mean :0.21740 Mean :0.3938   
## 3rd Qu.:0.4942 3rd Qu.:0.3025 3rd Qu.:0.30583 3rd Qu.:0.5546   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## symmetry\_worst dimension\_worst   
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.1851 1st Qu.:0.1077   
## Median :0.2478 Median :0.1640   
## Mean :0.2633 Mean :0.1896   
## 3rd Qu.:0.3182 3rd Qu.:0.2429   
## Max. :1.0000 Max. :1.0000

다시 주어진 데이터를 학습 데이터와 테스트 데이터로 나눠주어야 한다. 현재 가지고 있는 데이터는 테스트 데이터가 없다. 따라서 데이터를 나눠서 테스트 데이터를 만들어줘야 하는데, 아까 앞에서 데이터 구조를 확인할 때, id가 임의로 주어져 있음을 확인했다. 따라서 데이터를 슬라이스(slice)해도 일반성에 문제가 없다.

wbcd\_train <- wbcd\_n[1:469, ]  
wbcd\_test <- wbcd\_n[470:569, ]  
  
# 만들어놓은 데이터에 대하여 각각 레이블을 저장해놓는다.  
wbcd\_train\_labels <- wbcd[1:469, 1]  
wbcd\_test\_labels <- wbcd[470:569, 1]

##### 3단계. 데이터에 적용해 모델 훈련

kNN 알고리즘을 사용하기 위해 class 패키지를 불러온다. class 패키지는 가장 고전적이고 표준적인 kNN 알고리즘을 제공한다.

library(class)  
  
# knn() 함수의 사용법 확인  
?knn  
  
# knn(train, test, class, k)

train은 학습 데이터를, test는 테스트 데이터를, class는 학습 데이터를 분류하는 팩터 벡터를, k에는 kNN 알고리즘에서 최근접 이웃의 갯수 를 의미한다. 우리가 가지고 있는 학습 데이터는 wbcd\_train 이고 테스트 데이터는 wbcd\_test, 팩터 벡터는 wbcd\_train\_labels이고, 값은 학습 데이터 수의 제곱근의 근사치인 21로 둔다. 여기서 알아두어야 할 점은 본 데이터와 같이 분류가 바이너리(binary)한 경우, 를 홀수로 잡아주는 것이 좋다. 짝수로 둘 경우, 동률이 발생할 수 있기 때문이다. 코드는 다음과 같다.

wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 21)

##### 4단계. 모델 성능 평가

우리는 kNN 알고리즘을 이용해 wbcd\_test\_pred 라는 예측값을 얻어냈다. 이제 이 결과물이 얼마나 성능이 좋은지 확인하는 일만 남았다. 원래 가지고 있던 wbcd\_test\_labels과 비교를 해야 한다. 이를 편하게 하기 위해 교차표(Cross table)을 사용한다. 이를 위해 gmodels 패키지를 불러온다.

library(gmodels)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.968 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 2 | 37 | 39 |   
## | 0.051 | 0.949 | 0.390 |   
## | 0.032 | 1.000 | |   
## | 0.020 | 0.370 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 63 | 37 | 100 |   
## | 0.630 | 0.370 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

교차표를 확인하면 크게 네 개의 셀로 구분된다. 왼쪽 위부터 시계 방향으로 TP(True Positive), FP(False Positive), TN(True Negative), FN(False Negative) 으로 부르도록 하자. TP은 실제도 양성이었는데, 알고리즘의 결과도 양성인 경우다. 이처럼 FP는 실제로는 양성이지만 알고리즘의 결과가 음성인 경우, TN는 실제도 음성이었고 결과도 음성인 경우, 마지막으로 FN은 실제로는 음성이지만 결과가 양성인 경우다. 이를 평가하기 위해 다음의 세 개의 척도(measure)를 소개한다. 각각 *정밀도(Precision)*, *재현율(Recall)*, *정확도(Accuracy)*이다.[[3]](#footnote-35)

위의 척도를 사용하면 정밀도 100%, 재현율 96.8%, 정확도 98%이다. 다시 말해서 100개의 케이스 중, 두 개의 케이스가 오분류되었다. 굉장히 인상적인 결과지만, 모델 성능을 높이기 위해 다른 시도를 해보도록 하자.

##### 5단계. 모델 성능 높이기

kNN 알고리즘에서 데이터를 정규화하는 경우는 굉장히 많지만, 항상 옳은 방법은 아니다. 모든 데이터를 0부터 1까지로 압축하기 때문이다. 최댓값과 최솟값을 정해놓기 때문에 극단값이 중심을 향한다. 이런 경우 대안으로 *z 표준화*를 할 수 있다. 데이터의 평균 , 표준편차 에 대하여

로 나타낼 수 있다. z-표준화를 할 경우, 각 데이터들이 평균에서 먼 정도를 표준편차를 기준으로 나타내는데, 이상치에 대해 더 나은 결과를 보여줄 수 있다. R에서는 기본 함수인 scale()을 이용해서 표준화할 수 있다.

wbcd\_z <- as.data.frame(scale(wbcd[-1]))  
summary(wbcd\_z)

## radius\_mean texture\_mean perimeter\_mean area\_mean   
## Min. :-2.0279 Min. :-2.2273 Min. :-1.9828 Min. :-1.4532   
## 1st Qu.:-0.6888 1st Qu.:-0.7253 1st Qu.:-0.6913 1st Qu.:-0.6666   
## Median :-0.2149 Median :-0.1045 Median :-0.2358 Median :-0.2949   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.4690 3rd Qu.: 0.5837 3rd Qu.: 0.4992 3rd Qu.: 0.3632   
## Max. : 3.9678 Max. : 4.6478 Max. : 3.9726 Max. : 5.2459   
## smoothness\_mean compactness\_mean concavity\_mean points\_mean   
## Min. :-3.10935 Min. :-1.6087 Min. :-1.1139 Min. :-1.2607   
## 1st Qu.:-0.71034 1st Qu.:-0.7464 1st Qu.:-0.7431 1st Qu.:-0.7373   
## Median :-0.03486 Median :-0.2217 Median :-0.3419 Median :-0.3974   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.63564 3rd Qu.: 0.4934 3rd Qu.: 0.5256 3rd Qu.: 0.6464   
## Max. : 4.76672 Max. : 4.5644 Max. : 4.2399 Max. : 3.9245   
## symmetry\_mean dimension\_mean radius\_se texture\_se   
## Min. :-2.74171 Min. :-1.8183 Min. :-1.0590 Min. :-1.5529   
## 1st Qu.:-0.70262 1st Qu.:-0.7220 1st Qu.:-0.6230 1st Qu.:-0.6942   
## Median :-0.07156 Median :-0.1781 Median :-0.2920 Median :-0.1973   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.53031 3rd Qu.: 0.4706 3rd Qu.: 0.2659 3rd Qu.: 0.4661   
## Max. : 4.48081 Max. : 4.9066 Max. : 8.8991 Max. : 6.6494   
## perimeter\_se area\_se smoothness\_se compactness\_se   
## Min. :-1.0431 Min. :-0.7372 Min. :-1.7745 Min. :-1.2970   
## 1st Qu.:-0.6232 1st Qu.:-0.4943 1st Qu.:-0.6235 1st Qu.:-0.6923   
## Median :-0.2864 Median :-0.3475 Median :-0.2201 Median :-0.2808   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.2428 3rd Qu.: 0.1067 3rd Qu.: 0.3680 3rd Qu.: 0.3893   
## Max. : 9.4537 Max. :11.0321 Max. : 8.0229 Max. : 6.1381   
## concavity\_se points\_se symmetry\_se dimension\_se   
## Min. :-1.0566 Min. :-1.9118 Min. :-1.5315 Min. :-1.0960   
## 1st Qu.:-0.5567 1st Qu.:-0.6739 1st Qu.:-0.6511 1st Qu.:-0.5846   
## Median :-0.1989 Median :-0.1404 Median :-0.2192 Median :-0.2297   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.3365 3rd Qu.: 0.4722 3rd Qu.: 0.3554 3rd Qu.: 0.2884   
## Max. :12.0621 Max. : 6.6438 Max. : 7.0657 Max. : 9.8429   
## radius\_worst texture\_worst perimeter\_worst area\_worst   
## Min. :-1.7254 Min. :-2.22204 Min. :-1.6919 Min. :-1.2213   
## 1st Qu.:-0.6743 1st Qu.:-0.74797 1st Qu.:-0.6890 1st Qu.:-0.6416   
## Median :-0.2688 Median :-0.04348 Median :-0.2857 Median :-0.3409   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.5216 3rd Qu.: 0.65776 3rd Qu.: 0.5398 3rd Qu.: 0.3573   
## Max. : 4.0906 Max. : 3.88249 Max. : 4.2836 Max. : 5.9250   
## smoothness\_worst compactness\_worst concavity\_worst points\_worst   
## Min. :-2.6803 Min. :-1.4426 Min. :-1.3047 Min. :-1.7435   
## 1st Qu.:-0.6906 1st Qu.:-0.6805 1st Qu.:-0.7558 1st Qu.:-0.7557   
## Median :-0.0468 Median :-0.2693 Median :-0.2180 Median :-0.2233   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.5970 3rd Qu.: 0.5392 3rd Qu.: 0.5307 3rd Qu.: 0.7119   
## Max. : 3.9519 Max. : 5.1084 Max. : 4.6965 Max. : 2.6835   
## symmetry\_worst dimension\_worst   
## Min. :-2.1591 Min. :-1.6004   
## 1st Qu.:-0.6413 1st Qu.:-0.6913   
## Median :-0.1273 Median :-0.2163   
## Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.4497 3rd Qu.: 0.4504   
## Max. : 6.0407 Max. : 6.8408

표준화한 데이터를 다시 학습 데이터와 트레이닝 데이터로 나누고 kNN 알고리즘을 실행한 후 결과를 보도록 하자.

wbcd\_train <- wbcd\_z[1:469, ]  
wbcd\_test <- wbcd\_z[470:569, ]  
wbcd\_train\_labels <- wbcd[1:469, 1]  
wbcd\_test\_labels <- wbcd[470:569, 1]  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 21)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.924 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 5 | 34 | 39 |   
## | 0.128 | 0.872 | 0.390 |   
## | 0.076 | 1.000 | |   
## | 0.050 | 0.340 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 66 | 34 | 100 |   
## | 0.660 | 0.340 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

정밀도는 100%로 같지만, 재현율이 92.4%로 떨어지고 정확도도 95%로 떨어졌다. z-표준화를 이용하더라도 모델의 성능을 더 높이는 것은 힘들다는 결과를 얻을 수 있다.

추가적으로 값에 따라 정밀도, 재현율, 정확도가 모두 다른 것을 밑의 코드에서 확인할 수 있다.

wbcd\_train <- wbcd\_n[1:469, ]  
wbcd\_test <- wbcd\_n[470:569, ]  
wbcd\_train\_labels <- wbcd[1:469, 1]  
wbcd\_test\_labels <- wbcd[470:569, 1]  
  
# k = 1  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 1)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 58 | 3 | 61 |   
## | 0.951 | 0.049 | 0.610 |   
## | 0.983 | 0.073 | |   
## | 0.580 | 0.030 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 1 | 38 | 39 |   
## | 0.026 | 0.974 | 0.390 |   
## | 0.017 | 0.927 | |   
## | 0.010 | 0.380 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 59 | 41 | 100 |   
## | 0.590 | 0.410 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

# k = 5  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 5)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.968 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 2 | 37 | 39 |   
## | 0.051 | 0.949 | 0.390 |   
## | 0.032 | 1.000 | |   
## | 0.020 | 0.370 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 63 | 37 | 100 |   
## | 0.630 | 0.370 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

# k = 11  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 11)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.953 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 3 | 36 | 39 |   
## | 0.077 | 0.923 | 0.390 |   
## | 0.047 | 1.000 | |   
## | 0.030 | 0.360 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 64 | 36 | 100 |   
## | 0.640 | 0.360 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

# k = 15  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 15)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.953 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 3 | 36 | 39 |   
## | 0.077 | 0.923 | 0.390 |   
## | 0.047 | 1.000 | |   
## | 0.030 | 0.360 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 64 | 36 | 100 |   
## | 0.640 | 0.360 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

# k = 27  
wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,  
 cl = wbcd\_train\_labels, k = 27)  
CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 100   
##   
##   
## | wbcd\_test\_pred   
## wbcd\_test\_labels | Benign | Malignant | Row Total |   
## -----------------|-----------|-----------|-----------|  
## Benign | 61 | 0 | 61 |   
## | 1.000 | 0.000 | 0.610 |   
## | 0.938 | 0.000 | |   
## | 0.610 | 0.000 | |   
## -----------------|-----------|-----------|-----------|  
## Malignant | 4 | 35 | 39 |   
## | 0.103 | 0.897 | 0.390 |   
## | 0.062 | 1.000 | |   
## | 0.040 | 0.350 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 65 | 35 | 100 |   
## | 0.650 | 0.350 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

#### 실습 2 : 당뇨병 데이터

##### 1. 데이터 수집

실습할 데이터는 MASS 패키지에 포함되어 있는 Pima.tr 데이터다. MASS 패키지에는 실습을 위한 학습 데이터와 테스트 데이터가 모두 존재하는데, Pima.tr 데이터는 그 중에서 학습 데이터다. 본 데이터는 애리조나 피닉스 지역의 피마 인디언들의 당뇨병을 WHO 기준으로 기록한 데이터다.

##### 2. 데이터 탐구

학습 데이터와 테스트 데이터를 합쳐서 532명의 데이터가 존재하고, 총 여덟 개의 변수를 포함하고 있다. 각 변수에 대한 설명은 다음과 같다.

|  |  |
| --- | --- |
| 변수명 | 설명 |
| npreg | 임신 횟수 |
| glu | 글루코스 부하 검사 결과[[4]](#footnote-40) |
| bp | 확장기 혈압[[5]](#footnote-42) |
| skin | 삼두근 피부 주름 두께[[6]](#footnote-44) |
| bmi | BMI 수치 (몸무게 / 키^2) |
| ped | 가족력을 기반으로 당뇨병이 생길 확률[[7]](#footnote-45) |
| age | 나이 |
| type | 당뇨병 여부 |

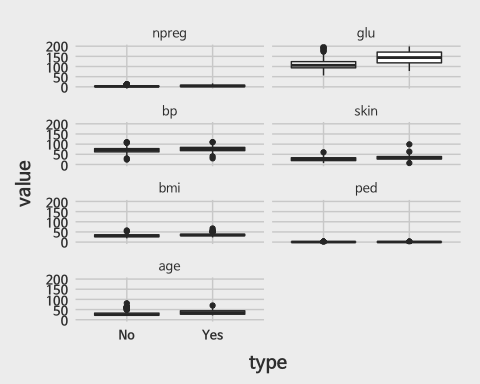
만약을 위해 기존의 데이터들을 합쳐서 다시 샘플링하여 학습 데이터와 테스트 데이터를 새로 생성하도록 한다. 우선 데이터를 합치고 데이터의 분포나 내부 구조를 들여다보자.

library(MASS)  
  
pima <- rbind(Pima.tr, Pima.te)  
str(pima)

## 'data.frame': 532 obs. of 8 variables:  
## $ npreg: int 5 7 5 0 0 5 3 1 3 2 ...  
## $ glu : int 86 195 77 165 107 97 83 193 142 128 ...  
## $ bp : int 68 70 82 76 60 76 58 50 80 78 ...  
## $ skin : int 28 33 41 43 25 27 31 16 15 37 ...  
## $ bmi : num 30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...  
## $ ped : num 0.364 0.163 0.156 0.259 0.133 ...  
## $ age : int 24 55 35 26 23 52 25 24 63 31 ...  
## $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...

탐색적 데이터 분석을 위해 reshape2 패키지의 melt() 함수를 이용해 데이터의 형태를 바꿔준다. 바꾼 데이터를 각각의 변수를 기준으로 상자그림(boxplot)을 그려준다.

library(reshape2)  
library(ggplot2)  
pima.melt <- melt(pima, id.var = "type")  
ggplot(data = pima.melt, aes(x = type, y = value)) +  
 geom\_boxplot() + facet\_wrap(~variable, ncol = 2)

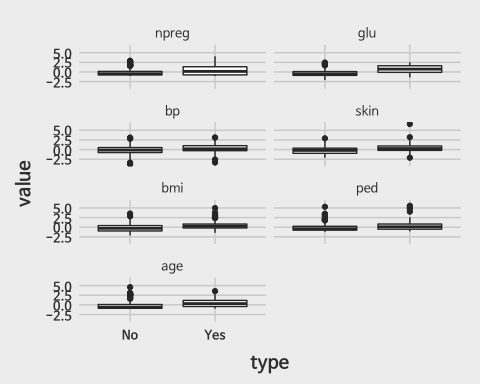


상자그림을 보면 실습 1과 같이 모든 데이터의 측정 범위가 상이하다. 따라서 표준화를 해주는데, 이번에는 z-표준화를 이용하도록 하자. 표준화 후, 위의 과정을 다시 한 번 진행한다.

pima.scale <- as.data.frame(scale(pima[, -8]))  
str(pima.scale)

## 'data.frame': 532 obs. of 7 variables:  
## $ npreg: num 0.448 1.052 0.448 -1.062 -1.062 ...  
## $ glu : num -1.13 2.386 -1.42 1.418 -0.453 ...  
## $ bp : num -0.285 -0.122 0.852 0.365 -0.935 ...  
## $ skin : num -0.112 0.363 1.123 1.313 -0.397 ...  
## $ bmi : num -0.391 -1.132 0.423 2.181 -0.943 ...  
## $ ped : num -0.403 -0.987 -1.007 -0.708 -1.074 ...  
## $ age : num -0.708 2.173 0.315 -0.522 -0.801 ...

pima.scale$type = pima$type  
  
pima.scale.melt <- melt(pima.scale, id.var = "type")  
ggplot(data = pima.scale.melt, aes(x = type, y = value)) +  
 geom\_boxplot() + facet\_wrap(~variable, ncol = 2)



각 변수에 대하여 kNN 알고리즘의 경우 서로 간의 상관도가 모델링에 있어 문제가 없지만, 선형 회귀법 등의 방법에서는 문제가 될 수 있으므로 확인하는 습관을 갖도록 하자.

cor(pima.scale[, -8])

## npreg glu bp skin bmi ped  
## npreg 1.000000000 0.1253296 0.204663421 0.09508511 0.008576282 0.007435104  
## glu 0.125329647 1.0000000 0.219177950 0.22659042 0.247079294 0.165817411  
## bp 0.204663421 0.2191779 1.000000000 0.22607244 0.307356904 0.008047249  
## skin 0.095085114 0.2265904 0.226072440 1.00000000 0.647422386 0.118635569  
## bmi 0.008576282 0.2470793 0.307356904 0.64742239 1.000000000 0.151107136  
## ped 0.007435104 0.1658174 0.008047249 0.11863557 0.151107136 1.000000000  
## age 0.640746866 0.2789071 0.346938723 0.16133614 0.073438257 0.071654133  
## age  
## npreg 0.64074687  
## glu 0.27890711  
## bp 0.34693872  
## skin 0.16133614  
## bmi 0.07343826  
## ped 0.07165413  
## age 1.00000000

상관계수를 확인하면 임신 횟수와 나이, 삼두근 피부 주름 두께와 bmi 가 제법 강한 양의 상관관계를 보여주고 있다.

이제 표준화한 데이터를 학습 데이터와 테스트 데이터로 나눈다. 실습 1과는 다르게 샘플링해서 분리하도록 하자. 이 때의 비율은 7:3으로 하도록 한다.

##### 3. 데이터 샘플링

set.seed(502)  
  
index <- sample(2, nrow(pima.scale), replace = TRUE, prob = c(0.7, 0.3))  
train <- pima.scale[index == 1, ]  
test <- pima.scale[index == 2, ]  
str(train)

## 'data.frame': 385 obs. of 8 variables:  
## $ npreg: num 0.448 0.448 -0.156 -0.76 -0.156 ...  
## $ glu : num -1.42 -0.775 -1.227 2.322 0.676 ...  
## $ bp : num 0.852 0.365 -1.097 -1.747 0.69 ...  
## $ skin : num 1.123 -0.207 0.173 -1.253 -1.348 ...  
## $ bmi : num 0.4229 0.3938 0.2049 -1.0159 -0.0712 ...  
## $ ped : num -1.007 -0.363 -0.485 0.441 -0.879 ...  
## $ age : num 0.315 1.894 -0.615 -0.708 2.916 ...  
## $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 2 1 1 1 ...

str(test)

## 'data.frame': 147 obs. of 8 variables:  
## $ npreg: num 0.448 1.052 -1.062 -1.062 -0.458 ...  
## $ glu : num -1.13 2.386 1.418 -0.453 0.225 ...  
## $ bp : num -0.285 -0.122 0.365 -0.935 0.528 ...  
## $ skin : num -0.112 0.363 1.313 -0.397 0.743 ...  
## $ bmi : num -0.391 -1.132 2.181 -0.943 1.513 ...  
## $ ped : num -0.403 -0.987 -0.708 -1.074 2.093 ...  
## $ age : num -0.7076 2.173 -0.5217 -0.8005 -0.0571 ...  
## $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 1 2 1 1 1 ...

##### 4. kNN 알고리즘에서 최적의 k 값 찾기

kNN 알고리즘의 적절한 값을 찾기 위해 다음의 과정을 거친다. 우선 부터 까지의 모든 경우를 테스트 하기 위해 expand.grid() 함수를 이용해 하나의 데이터 프레임을 만들어준다. 그 후, 학습 데이터를 제어하는 함수인 trainControl() 함수를 사용하는데, caret 패키지를 불러와야 사용 가능하다.

library(caret)

## Loading required package: lattice

grid1 <- expand.grid(.k = seq(2, 20, by = 1))  
control <- trainControl(method = "cv")

설정은 끝났고, 최적의 값만 찾는 일만 남았다.

library(e1071)  
set.seed(247)  
knn.train <- train(type~., data = train, method = "knn",   
 trControl = control, tuneGrid = grid1)  
knn.train

## k-Nearest Neighbors   
##   
## 385 samples  
## 7 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 347, 347, 347, 347, 347, 346, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa Accuracy SD Kappa SD   
## 2 0.7633367 0.4359009 0.05868938 0.13458253  
## 3 0.7660999 0.4315378 0.05083423 0.13027426  
## 4 0.7684008 0.4388530 0.05141061 0.13029596  
## 5 0.7634717 0.4268443 0.05440797 0.13412861  
## 6 0.7764238 0.4526894 0.04374589 0.11838938  
## 7 0.7555094 0.3969638 0.04917908 0.12940518  
## 8 0.7555061 0.3878549 0.06049806 0.16274000  
## 9 0.7530094 0.3850345 0.06338557 0.15729902  
## 10 0.7713630 0.4288802 0.04424808 0.12737021  
## 11 0.7686640 0.4233736 0.05333827 0.13770960  
## 12 0.7686640 0.4115415 0.04410195 0.13205313  
## 13 0.7710931 0.4144033 0.04207121 0.12714304  
## 14 0.7737922 0.4226427 0.04027685 0.11083768  
## 15 0.7918758 0.4697041 0.04243009 0.12320597  
## 16 0.7788529 0.4397692 0.03547550 0.10155165  
## 17 0.7868151 0.4583073 0.03776342 0.10825429  
## 18 0.7789204 0.4410321 0.03925586 0.10619862  
## 19 0.7712213 0.4177538 0.04302834 0.12939354  
## 20 0.7712922 0.4126024 0.02802213 0.09819693  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 15.

최적의 값은 15로 나왔다. 최적의 값은 정확도와 *카파 계수(Cohen's Kappa coefficient)*를 통해서 얻어진다. 카파 계수에 대한 내용은 [위키피디아 문서](https://en.wikipedia.org/wiki/Cohen%27s_kappa)를 확인하자. 최적의 값을 이용해 kNN 알고리즘을 실행하면 다음의 결과를 얻을 수 있다.

##### 5. 모델 성능 평가

knn.test <- knn(train[, -8], test[, -8], train[, 8], k = 15)  
CrossTable(x = test[, 8], y = knn.test, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 147   
##   
##   
## | knn.test   
## test[, 8] | No | Yes | Row Total |   
## -------------|-----------|-----------|-----------|  
## No | 76 | 17 | 93 |   
## | 0.817 | 0.183 | 0.633 |   
## | 0.738 | 0.386 | |   
## | 0.517 | 0.116 | |   
## -------------|-----------|-----------|-----------|  
## Yes | 27 | 27 | 54 |   
## | 0.500 | 0.500 | 0.367 |   
## | 0.262 | 0.614 | |   
## | 0.184 | 0.184 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 103 | 44 | 147 |   
## | 0.701 | 0.299 | |   
## -------------|-----------|-----------|-----------|  
##   
##

kNN 알고리즘 모델의 정밀도는 81.7%, 재현율은 73.8%, 정확도는 70.0%이다.

#### Reference

1. Lanz, Brett. *Machine Learning with R, 2nd Edition*. PACKT, 2015
2. <http://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/>
3. <http://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/>
4. <https://www.datacamp.com/community/tutorials/machine-learning-in-r>

1. 일반적으로 명목형 데이터를 다루지만, 수치형 데이터의 경우 개 데이터의 평균을 이용해 데이터를 분류한다. [↑](#footnote-ref-23)
2. 과적합이란 주어진 데이터로부터 보장되는 것 이상으로 모델을 만들 때 발생한다. 다시 말해서, 데이터 자체에 너무 충실한 나머지, 일반적인 구조를 표현하지 못하는 상황을 말한다. 만약 다른 데이터가 모델에 input으로써 들어올 때, 모델이 그 변화에 굉장히 민감해진다. [↑](#footnote-ref-24)
3. FP와 FN을 각각 제 1종 오류, 제 2종 오류라고 한다. [↑](#footnote-ref-35)
4. <http://terms.naver.com/entry.nhn?docId=488146&cid=55558&categoryId=55558> [↑](#footnote-ref-40)
5. <http://terms.naver.com/entry.nhn?docId=484340&cid=50364&categoryId=50364> [↑](#footnote-ref-42)
6. 피하지방 저장량의 크기를 추정할 수 있다. 환자의 영양 상태를 확인할 때 자주 사용한다. [↑](#footnote-ref-44)
7. <http://www.personal.kent.edu/~mshanker/personal/Zip_files/sar_2000.pdf> [↑](#footnote-ref-45)