데이터마이닝(DataMining)

Chapter 5.2. k-인접이웃분류

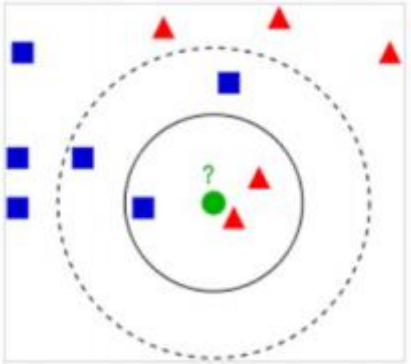
- k-인접이웃(k-nearest neighbor, 이하 k-NN)분류 모형은 새로운 데이터(설명변수값)에 대해 이와 가장 유사한(거리가 가까운) k-개의 과거 자료(설명변수값)의 결과(반응변수: 집단)를 이용 하여 다수결(majority vote)로 분류
- 과거자료를 이용하여 미리 분류모형을 수립하는 것이 아니라, 과거 데이터를 저장만 해두고 필요시 비교를 수행하는 방식
- k 값의 선택에 따라 새로운 데이터에 대한 분류결과가 달라짐에 유의
- k-NN은 반응변수가 범주형인 경우에는 분류(classification)의 목적으로, 반응변수가 연속형인 경우에는 회귀(regression)의 목적으로 사용

k-인접이웃분류

- k-NN은 기계학습 분야에서 가장 단순한 알고리즘
- 지역 정보만으로 근사 되며, 모든 계산이 이루어진 후에 분류가 이루어지는 특징으로 인해 사례-기반 학습(instancebased learning) 또는 게으른 학습(lazy learning)의 한 유형
- 사례-기반 학습은 메모리에 저장되어 있는 과거 훈련자료(training data)로부터 직접 결과가 도출되므로(분류가 수행되므로) 메모리-기반 학습(memory-based learning)이라고도 함

k-인접이웃분류

- 검증용 자료(중앙의 초록색 점)는 1그룹(푸른색 사각형) 또는 2그룹(붉은색 삼각 형)으로 분류
- 만약 $k = 3$(실선의 원)이면 이 자료는 2그룹으로 분류되며, $k = 5$(점선의 원) 이면 1그룹으로 분류



k-인접이웃분류

- k-NN은, 분류와 회귀 모두에서, 주변 값들의 기여도에 가중(weight)을 부여할 수 있음
- 더 가까운 주변일수록 더 큰 가중을 부여
- 예) 각 주변 점에 대해 새로운 점과의 거리의 역수(1/d)를 주변 점의 가중치로 하는 방법
- k-NN의 단점으로는 데이터의 지역 구조(local structure)에 민감함

k-인접이웃분류

- 통계적 모형을 적합하지 않는 메모리 기반의 방법
- $N(x)$를 x와 거리가 가장 가까운 k개의 훈련자료들의 집합인 k-근방이라하면

$$\hat{y}(x) = \arg \max_{I \in \{-1,+1\}} \sum_{x_j \in N(x)} I(y_j = I)$$

- 동점이 발생하면 랜덤하게 출력변수값을 예측
- k가 증가하면 편의는 커지고 분산은 감소
- 점근적으로 1-근방 분류법의 오분류율은 최적 베이즈 오분류율의 2배 이내

k-인접이웃분류

- 패키지 {class}의 knn() 함수를 이용하여 k-NN 분류를 수행

```
> data(iris3)    # 3차원 배열 자료(50×4×3)
> train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])    # 행렬 객체임
> test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
> cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
> knn(train, test, cl, k = 3, prob=TRUE)
```

k-인접이웃분류

```
[1] s s s s s s s s s s s s s s s s s s s s s s s s s c c v c c c c c v c
c c
[40] c c c c c c c c c c c c c v c c v v v v v v v v v v c v v v v v v v
v v v v v

attr(,"prob")
 [1] 1.000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [8] 1.000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[15] 1.000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 ...
[64] 0.666667 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[71] 1.000000 0.6666667 1.0000000 1.0000000 0.6666667
Levels: c s v
```

k-인접이웃분류

- 패키지 {DMwR}의 kNN() 함수를 이용하여 k-근접이웃분류를 수행
- kNN() 함수는 knn{class} 함수와 유사하나, 모형식 기반으로 수행되는 차이점이 있으며 자료에 대한 정규화 옵션(norm=)을 제공

```
> data(iris)
> idxs <- sample(1:nrow(iris), as.integer(0.7*nrow(iris)))
> trainIris <- iris[idxs,]
> testIris <- iris[-idxs,]
```

k-인접이웃분류

```
> nn3 <- kNN(Species ~ ., trainIris, testIris, norm=FALSE, k=3)
> table(testIris[,'Species'], nn3)
            nn3
             setosa    versicolor    virginica
 setosa          19             0            0
 versicolor       0             7            1
 virginica        0             1           17
```

k-인접이웃분류

```
> nn3 <- kNN(Species ~ ., trainIris, testIris, norm=FALSE, k=3)
> table(testIris[,'Species'], nn3)
            nn3
             setosa    versicolor    virginica
 setosa          19             0            0
 versicolor       0             7            1
 virginica        0             1           17
```

```
> nn5 <- kNN(Species ~ ., trainIris, testIris, norm=TRUE, k=5)
            nn5
             setosa    versicolor    virginica
 setosa          19             0            0
 versicolor       0             8            0
 virginica        0             1           17
```

k-인접이웃분류

```
> nn3 <- kNN(Species ~ ., trainIris, testIris, norm=FALSE, k=3)
> table(testIris[,'Species'], nn3)
            nn3
             setosa    versicolor    virginica
 setosa          19             0            0
 versicolor       0             7            1
 virginica        0             1           17
```

```
> nn5 <- kNN(Species ~ ., trainIris, testIris, norm=TRUE, k=5)
            nn5
             setosa    versicolor    virginica
 setosa          19             0            0
 versicolor       0             8            0
 virginica        0             1           17
```

k-인접이웃분류

- {kknn}의 kknn() 함수를 이용하여 k-근접이웃분류를 수행
- kknn() 함수는 가중(weighted) k-NN 분류를 제공

```
> library(kknn)
> data(iris)
> m <- dim(iris)[1]
> val <- sample(1:m, size=round(m/3), replace=FALSE,
                prob=rep(1/m, m))
> iris.learn <- iris[-val,]
> iris.valid <- iris[val,]
```

k-인접이웃분류

- knnn() 함수는 검증용 저료의 각 열에 대해 k-인접이웃을 민코우스키 거리에 기반
- distance= 옵션은 민코우스키 거리의 모수(parameter)를 지정하며, distance=2 는 유클리드거리에 해당

$$d(X,Y) = \left[\sum_{i=1}^{p}(x_i - y_i)^m\right]^{\frac{1}{m}}$$

- kernel= 옵션은 이웃점들의 가중치를 부여하는 방법을 지정
- kernel= 옵션에는 "rectangular"(이는 가중을 고려하지 않은 표준 k-NN과 동일), "triangular", "epanechnikov"(또는 beta(2,2)), "biweight"(또는 beta(3,3)), "triweight" (또는 beta(4,4)), "cos", "inv", "gaussian", "rank" 와 "optimal"이 있음
- 가중 k-NN 분류는 커널밀도함수의 합이 최대인 군집으로 분류를 수행

k-인접이웃분류

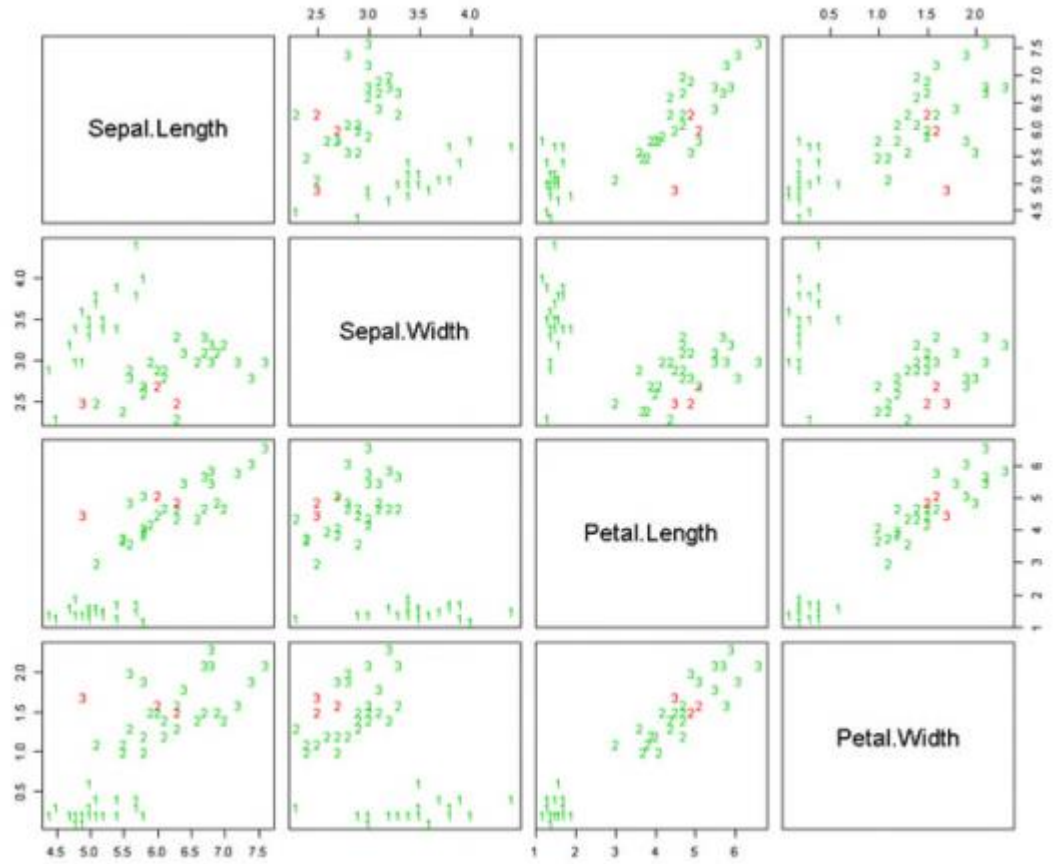
```
> iris.kknn <- kknn(Species~., iris.learn, iris.valid, distance=1,
                    kernel="triangular")
> summary(iris.kknn)
 Call:
 kknn(formula=Species~.,train=iris.learn,test=iris.valid,
 distance=1, kernel="triangular")    # distance=1은 시가거리

 Response: "nominal"
          fit prob.setosa prob.versicolor prob.virginica
 1  virginica           0      0.01512849     0.98487151
 2     setosa           1      0.00000000     0.00000000
 3     setosa           1      0.00000000     0.00000000
 4     setosa           1      0.00000000     0.00000000
 5 versicolor           0      1.00000000     0.00000000
```

k-인접이웃분류

```
> fit <- fitted(iris.kknn)
> table(iris.valid$Species, fit)
           fit
            setosa versicolor virginica
 setosa         21          0         0
 versicolor      0         17         2
 virginica       0          1         9
```

```
> pcol <- as.character(as.numeric(iris.valid$Species))
> pairs(iris.valid[1:4], pch=pcol,
        col=c("green3", "red")[(iris.valid$Species != fit)+1])
```

k-인접이웃분류



k-인접이웃분류

- 미 프로야구 선수 6명에 대해 두 시즌간의 기록(lag1, lag2)이 다음 해의 득점(runs)에 미친 영 향을 알아보기 위해 k-NN 회귀를 수행

```
> full <- data.frame(name=c("McGwire,Mark", "Bonds,Barry",
                            "Helton,Todd", "Walker,Larry",
                            "Pujols,Albert", "Pedroia,Dustin"),
                    lag1=c(100,90,75,89,95,70),
                    lag2=c(120,80,95,79,92,90),
                    Runs=c(65,120,105,99,65,100))
> full
             name lag1 lag2 Runs
1   McGwire,Mark  100  120   65
2    Bonds,Barry   90   80  120
3    Helton,Todd   75   95  105
4   Walker,Larry   89   79   99
5  Pujols,Albert   95   92   65
6 Pedroia,Dustin   70   90  100
```

k-인접이웃분류

- 5명의 선수 자료를 훈련용으 로하고, 한 명의 선수("Bonds,Barry")를 검증용으로 하여 예측을 수행

```
> library(kknn)
> train <- full[full$name!="Bonds,Barry",]
> test <- full[full$name=="Bonds,Barry",]
> k <- kknn(Runs~lag1+lag2, train=train, test=test, k=2, distance=1)
> fit <- fitted(k)
> fit
 [1] 90.5
```

k-인접이웃분류

- CL은 k-근접이웃의 class의 행렬, W는 k-근접이웃의 가중치의 행렬, D는 k-근접 이웃의 거리행렬, C는 k-근접이웃의 위치(indices) 등을 나타냄

```
> names(k)
 [1] "fitted.values" "CL"     "W"        "D"
 [5] "C"             "prob"   "response" "distance"
 [9] "call"          "terms"
```

```
> k$fitted.values
 [1] 90.5
```

k-인접이웃분류

- Bonds와 가장 가까운 2개의 인접값(득점)으로 99와 65를 얻었다. 두 값의 가 중치는 각각 0.75와 0.25이며, Bonds의 예측치 90.5는 가중평균((99*3+65)/4=90.5)으로 부터 구해진 값

```
> k$CL
     [,1] [,2]
[1,]   99   65
```

```
> k$W
     [,1] [,2]
[1,] 0.75 0.25
```

k-인접이웃분류

- 두 개의 인접값 중 99는 한명의 선수에만 해당되지만, 65는 두 명의 선수 (McGwire와 Pujols)가 이 값을 가짐

```
> k$C
 [1] 3 4
```

```
> train[c(k$C),]
            name lag1 lag2 Runs
4  Walker,Larry   89   79   99
5 Pujols,Albert   95   92   65
```

- 인접이웃의 위치가 훈련용 자료에서 3, 4번째임을 알려주며, 이를 출력하면 Walker와 Pujols가 Bonds의 인접이웃임을 알 수 있음

k-인접이웃분류

- {FNN} 패키지는 훈련용 자료에 대해 원하는 질의(query)를 통해 필요한 결과를 얻게 해 줌
- get.knnx() 함수를 통해 인접이웃을 구하는 과정

```
> library(FNN)
> get.knnx(data=train[,c("lag1","lag2")],
query=test[,c("lag1","lag2")], k=2)
 $nn.index    # 인접이웃의 index가 3, 4번임
      [,1] [,2]
 [1,]    3    4

 $nn.dist
          [,1] [,2]
 [1,] 1.414214   13
```

- Bonds와 가까운 인접이웃은 Walker와 Pujols

k-인접이웃분류

- {caret}을 이용하여 k-NN 분석을 수행

```
> library(ISLR)
> library(caret)
```

- {caret} 패키지의 createDataPartition() 함수를 이용하여 훈련용 자료와 검증용 자료로 분할

```
> set.seed(100)
> indxTrain <- createDataPartition(y = Smarket$Direction, p = 0.75,
                                   list = FALSE)
> training <- Smarket[indxTrain,]
> testing <- Smarket[-indxTrain,]
```

k-인접이웃분류

- 원 자료와 분할된 자료의 분포를 비교

```
> prop.table(table(training$Direction)) * 100
    Down       Up
48.18763 51.81237
```

```
> prop.table(table(testing$Direction)) * 100
    Down       Up
48.07692 51.92308
```

```
> prop.table(table(Smarket$Direction)) * 100
 Down    Up
48.16 51.84
```

- createDataParition()은, 자료 분할을 위해 이전에 사용되었던 방식보다, 매우 편리하게 자료 를 분할

k-인접이웃분류

- k-NN 분류를 수행하기 위해서는 변수의 정규화 또는 척도화가 필요
- preProcess() 함수를 통해 변수에 대한 중심화와 척도화를 수행

```
> trainX <- training[,names(training) != "Direction"]    # 반응변수를 제외
> preProcValues <- preProcess(x = trainX,
                              method = c("center", "scale"))
> preProcValues
 Created from 938 samples and 8 variables
 Pre-processing:
  - centered (8)
  - ignored (0)
  - scaled (8)
```

k-인접이웃분류

```
> set.seed(200)
> ctrl <- trainControl(method="repeatedcv", repeats = 3)
> # 추가 가능 옵션: classProbs=TRUE,summaryFunction = twoClassSummary
> knnFit <- train(Direction ~ ., data = training, method = "knn",
                  trControl = ctrl,
                  preProcess = c("center","scale"), tuneLength = 20)
> # k-NN 적합 결과
> knnFit
 k-Nearest Neighbors
```

k-인접이웃분류

```
938 samples
  8 predictor
  2 classes: 'Down', 'Up'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 845, 844, 845, 845, 844, 843, ...
Resampling results across tuning parameters:

  k  Accuracy     Kappa
  5 0.8674133 0.7338303
  7 0.8698920 0.7387770
  9 0.8745213 0.7479976
 11 0.8762378 0.7513700
 13 0.8826588 0.7642091
```

k-인접이웃분류

```
 17 0.8915318 0.7820211
 19 0.8915129 0.7818643
 21 0.8918674 0.7826024
 23 0.8893849 0.7774538
 25 0.8961421 0.7910932
 27 0.8922335 0.7832304
 29 0.8939992 0.7866896
 31 0.8897627 0.7782300
 33 0.8890345 0.7766604
 35 0.8858504 0.7702838
 37 0.8851447 0.7688566
 39 0.8855107 0.7696275
 41 0.8880235 0.7746903
 43 0.8855332 0.7696039

Accuracy was used to select the optimal model using the largest
value.
 The final value used for the model was k = 25.
```

k-인접이웃분류

- 인접이웃(k)의 크기에 따라, 교차타당법에 기초하여, 정확도를 구하기

```
> plot(knnFit)
```

- 인접 이웃의 크기(k)가 25일 때, 정확도가 제일 높음
- 훈련용 자료에 대해 적합된 최적 모형은 자동으로 이 값을 선택

k-인접이웃분류

```
> knnPredict <- predict(knnFit, newdata = testing )
> confusionMatrix(knnPredict, testing$Direction )
 Confusion Matrix and Statistics

           Reference
 Prediction Down  Up
       Down  128   7
         Up   22 155

               Accuracy : 0.9071    # (128+155)/2
                 95% CI : (0.8692, 0.9369)
    No Information Rate : 0.5192    # 162/312(자료에서 0과 1의 비율 중 큰 값)
    P-Value [Acc > NIR] : < 2e-16
```

k-인접이웃분류

```
                    Kappa : 0.8131    # (0.9071-0.5026)/(1-0.5026)
Mcnemar's Test P-Value : 0.00933   # 0.5026은 독립성하의 기대 정확도임
            Sensitivity : 0.8533
(재현율)    Specificity : 0.9568
(정밀도) Pos Pred Value : 0.9481    # 128/135(예측1 내에서 실제1의 비율)
        Neg Pred Value : 0.8757    # 155/177(예측0 내에서 실제0의 비율)
            Prevalence : 0.4808    # 150/312(자료에서 1의 비율)
        Detection Rate : 0.4103    # 128/312(전체에서 1을 1로 예측한 비율)
  Detection Prevalence : 0.4327    # 135/312(전체에서 1로 예측한 비율)
     Balanced Accuracy : 0.9051    # (민감도+특이도)/2

      'Positive' Class : Down
```

```
> mean(knnPredict == testing$Direction)
 [1] 0.9070513
```

- 검증용 자료에 대한 정확도가 90.7%
- 훈련용 자료보다도 더 높게 나타나, 과적 합의 문제는 없는 것으로 판단

k-인접이웃분류

- summary=twoClassSummary를 지정하면 두 집단 문제에서 AUC, 민감도, 특이도 등의 성능 측도를 제공
- classProbs=TRUE는 이러한 측도의 계산에 필요
- tuneLength=20은 조절모수의 격자를 조절하는 값

```
> # 2 클래스 요약함수 확인
> set.seed(200)
> ctrl <- trainControl(method="repeatedcv", repeats = 3,
classProbs=TRUE, summaryFunction = twoClassSummary)
> knnFit <- train(Direction ~ ., data = training, method = "knn",
trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20)
Warning message:
In train.default(x, y, weights = w, ...) :
  The metric "Accuracy" was not in the result set. ROC will be used
instead.
```

k-인접이웃분류

```
> # k-NN 적합 결과
> knnFit
k-Nearest Neighbors

938 samples
  8 predictor
  2 classes: 'Down', 'Up'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 845, 844, 845, 845, 844, 843, ...
Resampling results across tuning parameters:

  k       ROC       Sens      Spec
  5 0.9405061 0.8316747 0.9006094
```

k-인접이웃분류

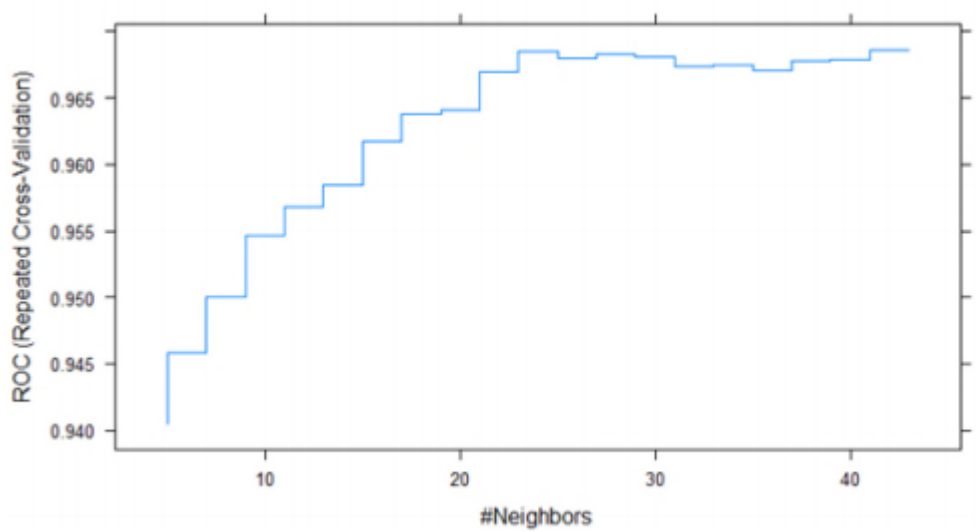
7	0.9457679	0.8331723	0.9039824
9	0.9500242	0.8338808	0.9121740
11	0.9546494	0.8287279	0.9203798
13	0.9568002	0.8338647	0.9279478
15	0.9584242	0.8398390	0.9334325
17	0.9617724	0.8383575	0.9409722
19	0.9637870	0.8324316	0.9464427
21	0.9641127	0.8361031	0.9437358
23	0.9670206	0.8228824	0.9512472
25	0.9685227	0.8338647	0.9540249
27	0.9680233	0.8287279	0.9513039
29	0.9682626	0.8257488	0.9574405
31	0.9680949	0.8250403	0.9499433
33	0.9673842	0.8184058	0.9546910
35	0.9674677	0.8176490	0.9492063

k-인접이웃분류

```
  37    0.9670697    0.8147343    0.9505527
  39    0.9677627    0.8162319    0.9499008
  41    0.9679091    0.8199356    0.9512755
  43    0.9685804    0.8110950    0.9547336

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 43.
```

```
> # 이웃의 수에 대한 정확도 그림(반복된 교차타당법에 의한)
> plot(knnFit, print.thres = 0.5, type="S")
```

k-인접이웃분류



- 적합모형은 인접이웃으로 k=43을 선택(AUC 기준 사용)
- k=25도 훌륭한 대안으로 생각

k-인접이웃분류

- 적합된 모형을 이용하여 검증용 자료에 대해 예측을 수행
- 정분류율 관점에서 모형의 성능이 다소간 향상되었음을 확인

```
> # 검증용 자료에 대한 예측
> knnPredict <- predict(knnFit, newdata = testing )
> confusionMatrix(knnPredict, testing$Direction )    # 정오분류행렬
 Confusion Matrix and Statistics

           Reference
 Prediction Down  Up
       Down  128   4
         Up   22 158
```

실제값(Reference)

예측값(Prediction)	Y	N
Y	True Positive(TP)	False Positive(FP)
N	False Negative(FN)	True Negative(TN)

지표	계산식	의미
Precision	$\frac{TP}{TP+FP}$	Y로 예측된 것 중 실제로도 Y인 경우의 비율
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	전체 예측에서(예측이 Y이든 N이든 무관하게) 옳은 예측의 비율
Recall (Sensitivity, TP Rate, Hit Rate)	$\frac{TP}{TP+FN}$	실제로 Y인 것들 중 예측이 Y로 된 경우의 비율
Specificity	$\frac{TN}{FP+TN}$	실제로 N인 것들 중 예측이 N으로 된 경우의 비율
FP Rate (False Alarm Rate)	$\frac{FP}{FP+TN}$	실제로 N인데 Y로 예측된 비율, 1-Specificity와 같은 값
Pos Pred Value	$\frac{TP}{TP+FP}$	Y로 예측한 것들 중에 실제 Y인 비율
Neg Pred Value	$\frac{TN}{TN+FN}$	N로 예측한 것들 중에 실제 N인 비율
Prevalence	$\frac{FN+TP}{TP+FP+FN+TN}$	전체 중 실제 Y가 나타난 비율
Detection Rate	$\frac{TP}{TP+FP+FN+TN}$	전체 중 모델이 실제 Y를 예측한 비율
Detection Prevalence	$\frac{TP+FP}{TP+FP+FN+TN}$	전체 중 모델이 Y로 예측한 비율
Balanced Accuracy	(Recall+Specificity)/2	(Recall+Specificity)/2
F-1 Score	$2\times\frac{1}{\frac{1}{Precision}+\frac{1}{recall}}$	Precision과 Recall의 조화평균 (Precision과 Recall 중 한쪽만 클 때보다 두 값이 골고루 클 때 큰 값을 가짐. 0~1 사이의 값을 가짐)
Kappa	$\frac{Accuracy-P(e)}{1-P(e)}$	일반적으로 코헨의 카파를 말하며 두 평가자의 평가가 얼마나 일치하는 지를 의미함. 0~1사이의 값을 가짐. P(e)는 두 평가자가 우연히 일치할 확률을 의미함.

k-인접이웃분류

```
               Accuracy : 0.9167
                 95% CI : (0.8803, 0.9448)
    No Information Rate : 0.5192
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8323
 Mcnemar's Test P-Value : 0.0008561

            Sensitivity : 0.8533
            Specificity : 0.9753
         Pos Pred Value : 0.9697
         Neg Pred Value : 0.8778
             Prevalence : 0.4808
         Detection Rate : 0.4103
   Detection Prevalence : 0.4231
```

k-인접이웃분류

```
Balanced Accuracy : 0.9143

 'Positive' Class : Down
```

```
> mean(knnPredict == testing$Direction)
 [1] 0.9166667
```

k-인접이웃분류

```
> # ROC 곡선 그리기
> library(pROC)
> knnPredict <- predict(knnFit, newdata = testing , type="prob")
> knnPredict
     Down      Up
 1   0.06    0.93
 2   0.69    0.30
  …
> knnROC <- roc(testing$Direction, knnPredict[,"Down"],
                levels = levels(testing$Direction))
> # 밑줄 부분 출력: [1] “Down”  “Up” (문자형 벡터)
```

k-인접이웃분류

```
> knnROC

 Call:
 roc.default(response = testing$Direction, predictor = knnPredict[,
"Down"], levels = levels(testing$Direction))

 Data: knnPredict[, "Down"] in 150 controls (testing$Direction
Down) > 162 cases (testing$Direction Up).
 Area under the curve: 0.9817
```

```
> plot(knnROC, type="S", print.thres= 0.5)    # 기준값 0.5일 때의 결과를 표시
```