R 기반 의학통계 및 머신러닝

박 승

충북대학교
CHUNGBUK NATIONAL UNIVERSITY

CHAPTER

07

ROC 곡선과 모델 평가

07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

(1) ROC(Receiver Operationg Characteristic) Curve

- ROC 곡선은 분류 모델의 성능을 평가하는 그래프

  모델이 얼마나 효과적으로 양성(Positive)과 음성(Negative)을 구별하는지 나타냄

(2) AUC (Area Under Curve)

- ROC 곡선 아래 면적을 나타내는 값으로, 모델의 전반적인 성능을 수치화

  AUC값 해석

  0.5 ⇒ 완전 랜덤 추측 (무의미한 모형)

  0.7 ~ 0.8 ⇒ 보통 수준의 모형

  0.8 ~ 0.9 ⇒ 좋은 성능의 모형

  0.9 ~ 0.99 ⇒ 아주 좋은 성능의 모형. (과적합이 의심됨)

  0.99 ~ 1 ⇒ 완벽한 모형. (결과변수에 직접적으로 연관되는 변수가 모형에 포함되지는 않았는지 검증이 필요)

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬 (Confusion Matrix) 과 성능 평가 지표

정확도 (Accuracy) $Accuracy = \frac{TP+TN}{Tp+TN+FP+FN}$

재현율 (Recall, Sensitivity, TPR) $Sensitivity = \frac{TP}{Tp+FN}$

특이도 (Specificity, 1-FPR) $Specificity = \frac{TN}{TN+FP}$

정밀도 (Precision, PPV) $Precision = \frac{TP}{TP+FP}$

F1-score $F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}$

실제 \ 예측	Positive	Negative
Positive	True Positive(TP)	False Negative(FN)
Negative	False Positive(FP)	True Negative(TN)

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

- 아래와 같은 자료를 고려하자

  predict는 어떤 로지스틱 회귀 모델의 결과이다.

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

- ROC를 직접 그리는 방법은 여러가지가 있으나, threshold 조절을 통한 방법을 소개한다

  1) threshold를 1부터 0까지, sequential 하게 감소시킨다.

  2) predict에 걸리기 직전 지점에서의 specificity, sensitivity를 계산한다.

  3) (0,0)을 시작점으로 둔다.

  4) X축을 1-Specificity로, Y축을 Sensitivity로 점들을 순서대로 이어 ROC를 작성한다.

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

1) threshold = 0.95 (시작점)

confusion matrix는 다음과 같다.

이때의 1-specificity는 0, sensitivity는 00이다. (0, 0)

		예측 0	예측 1
실제	0	3	0
	1	6	0

2) threshold = 0.85

confusion matrix는 다음과 같다.

이때의 1-specificity는 0, sensitivity는 0.16670이다. (0, 0.1667)

		예측 0	예측 1
실제	0	3	0
	1	5	1

⋮

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

3) threshold = 0.55

confusion matrix는 다음과 같다.

이때의 1-specificity는 0, sensitivity는 0.66670이다. (0, 0.6667)

		예측 0	예측 1
실제	0	3	0
	1	2	4

4) threshold = 0.45

confusion matrix는 다음과 같다.

이때의 1-specificity는 0.3333, sensitivity는 0.66670이다. (0.3333, 0.6667)

		예측 0	예측 1
실제	0	3	0
	1	5	1

⋮

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

5) threshold = 0.35

confusion matrix는 다음과 같다.

이때의 1-specificity는 0.3333, sensitivity는 0.83330이다. (0.3333, 0.8333)

실제 \ 예측	0	1
0	2	1
1	1	5

6) threshold = 0.25

confusion matrix는 다음과 같다.

이때의 1-specificity는 0.6667, sensitivity는 0.83330이다. (0.6667, 0.8333)

실제 \ 예측	0	1
0	1	2
1	1	5

⋮

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

7) threshold = 0.15

confusion matrix는 다음과 같다.

이때의 1-specificity는 0.6667, sensitivity는 10이다. (0.6667, 1)

		예측 0	예측 1
실제	0	1	2
	1	0	6

8) threshold = 0.05

confusion matrix는 다음과 같다.

이때의 1-specificity는 1, sensitivity는 10이다. (1, 1)

		예측 0	예측 1
실제	0	0	3
	1	0	6

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

실제 label	0	1	0	1	0	1	1	1	1
predict	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

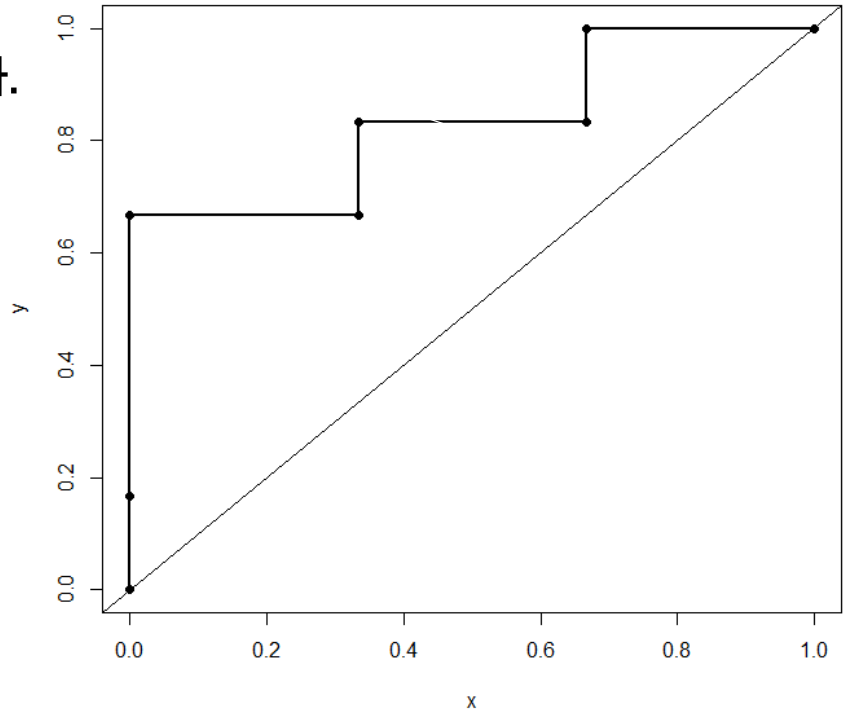
9) 얻어진 모든 점을 순서대로 이어 ROC 곡선을 그린다.

(0, 0), (0, 0.1667),

(0, 0.6667), (0.3333, 0.6667),

(0.3333, 0.8333), (0.6667, 0.8333),

(0.6667, 1), (1, 1)



ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

AUC는 곡선 아래의 넓이를 통해 구할 수 있다.

마름모꼴 또는 사각형 넓이로

계산하면 쉽다.

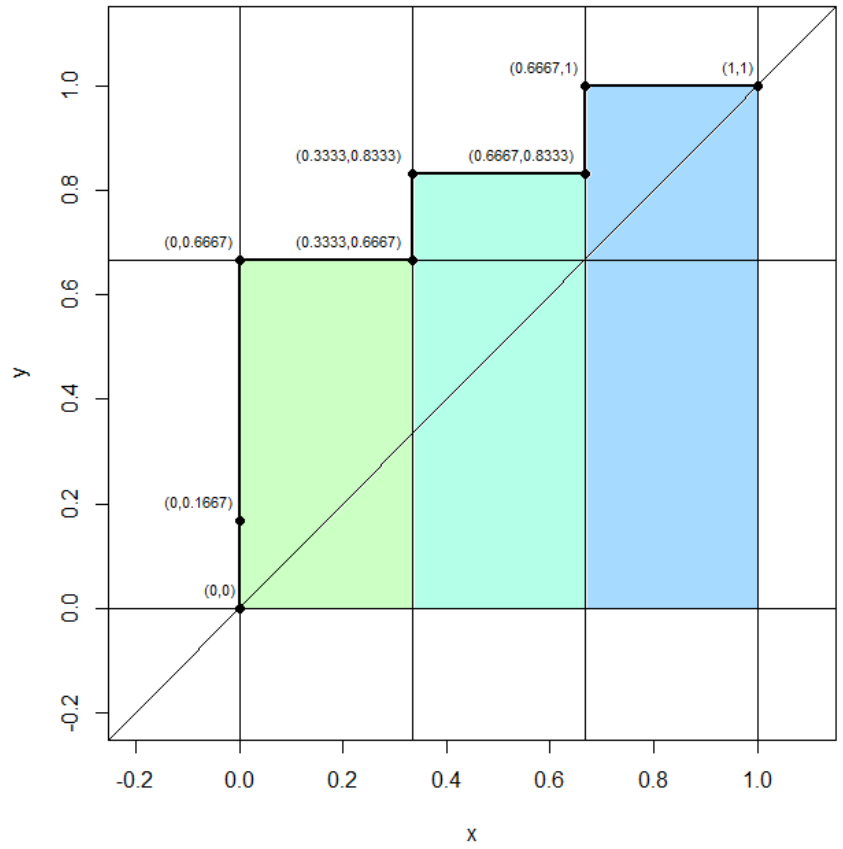
$$0.3333 \times 0.6667 = 0.2222$$

$$0.3334 \times 0.8333 = 0.2778$$

$$0.3334 \times 1 = 0.3333$$

ROC 아래의 넓이 AUC는

$$0.2222 + 0.2778 + 0.3333 = \mathbf{0.8333}$$



ROC 곡선과 AUC(Area Under Curve)

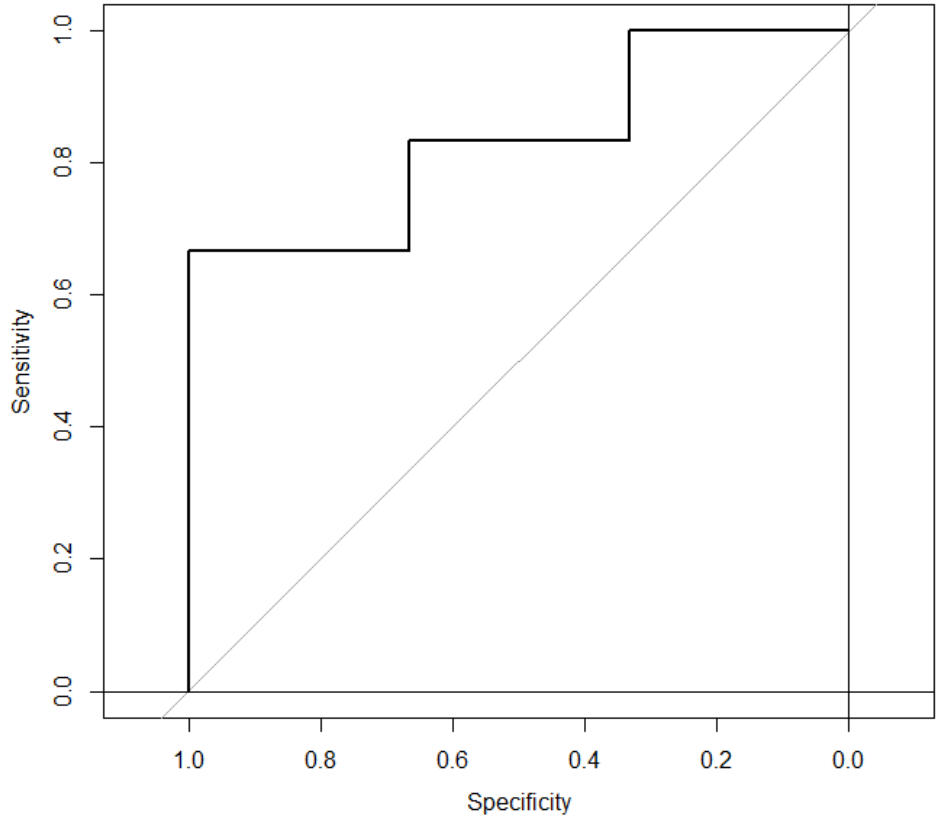
(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

R 결과와 비교

```
library(pROC)
p<-c(0.1, 0.2, 0.3, 0.4, 0.5,
     0.6, 0.7, 0.8, 0.9)
r<-c(0, 1, 0, 1, 0,
     1, 1, 1, 1)
dat1<-data.frame(cbind(p,r))
auc(dat1$r, dat1$p)
plot(roc(dat1$r, dat1$p))
abline(v=0)
abline(h=0)
```

abline에 v나 h 옵션을 사용하면
세로선이나 가로선을 그림

```
> auc(dat1$r, dat1$p)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
Area under the curve: 0.833
```



ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

참고) 사용된 코드

```
p<-c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)
r<-c(0,1,0,1,0,1,1,1,1)
dat1<-data.frame(cbind(p,r))
threshold<-0.05
dat1$p2<-ifelse(dat1$p>threshold,1,0)

t2<-table(factor(dat1$r, levels=c(0,1)),
          factor(dat1$p2, levels=c(0,1)))

sens<-t2[2,2]/(t2[2,2]+t2[2,1])
spec<-t2[1,1]/(t2[1,1]+t2[1,2])

print(t2)
print(threshold)
print(c(1-spec,sens))
```

#threshold를 0.95에서 0.1씩 빼 가면서 계산해보자

#범주가 하나로만 나오는 경우를 방지하기 위한 factor 설정

#혼동되지 않도록 주의

#혼동행렬, threshold, 결과지표 출력

		예측 0	예측 1
실제	0	TN	FP
실제	1	FN	TP

ROC 곡선과 AUC(Area Under Curve)

(3) 혼동행렬과 평가지표 – ROC 곡선 그려보기

참고) 사용된 코드

```
x <- c(0, 0, 0, 0.3333, 0.3333, 0.6667, 0.6667, 1)
y <- c(0, 0.1667, 0.6667, 0.6667, 0.8333, 0.8333, 1, 1)

plot(x, y, type = "l", lwd = 2,# xlab = "x", ylab = "y",
     xlim = c(-0.2, 1.1), ylim = c(-0.2, 1.1))
points(x, y, pch = 19)
text(x, y, labels = paste0("(",x,",",y,")"),
     adj=c(1.1,-1.1), cex = 0.7)
abline(0,1)
abline(0,0)
abline(v=0)
abline(v=1)
abline(v=0.3333)
abline(v=0.6667)
abline(h=0.6667)
```

x와 y의 값을 그대로 붙여넣는 코드

#점 좌표 위치. (1,-1)은 좌상단에 찍힌다.
값을 바꾸어 가며 실행해보자

#abline은 기본적으로 a+bx의 직선을 그리지만,
v나 h는 수직선 또는 수평선을 그려줌

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

train_data로 만든 모형에 test_data를 적용하는 과정 ##상술한 Contrast 문제로 dataset 변수는 제외

```
data <- read.csv("heart_disease_uci.csv")
data$target <- ifelse(data$num > 0, 1, 0)
data2 <- filter(data, if_all(everything(), ~!is.na(.) & .!=""))
data2sub <- subset(data2, select=c(-id, -num, -dataset))
set.seed(42)
train_index <- createDataPartition(data2sub$target, p = 0.8, list = FALSE)
train_data <- data2sub[train_index, ]
test_data <- data2sub[-train_index, ]


## 로지스틱
logit_model <- glm(target ~ ., data = train_data, family = binomial)
logit_pred <- predict(logit_model, newdata=test_data, type = "response")
                                                    #예측값을 확률형태로 반환
## 랜덤포레스트
train_data$target <- as.factor(train_data$target)
rf_model <- randomForest(target ~ ., data = train_data, importance = TRUE, ntree = 500)
rf_pred <- predict(rf_model, newdata=test_data, type = "prob")[,2]
                                                    #예측값을 확률형태로 반환
```

07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

```
## XGboost
train_matrix <- model.matrix(target ~ . - 1, data = train_data)
train_label <- as.numeric(as.character(train_data$target))
dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  tree_method = "auto"
)
xgb_model <- xgb.train(params, dtrain, nrounds = 100, verbose = 0)

test_matrix <- model.matrix(target ~ . - 1, data = test_data)
test_label <- as.numeric(as.character(test_data$target))
dtest <- xgb.DMatrix(data = test_matrix, label = test_label)
xgb_pred <- predict(xgb_model, dtest)
```

```
> xgb_pred <- predict(xgb_model, dtest)
predict.xgb.Booster(xgb_model, dtest)에서 다음과 같은 에러가 발생했습니다:
  Feature names stored in `object` and `newdata` are different!
```

07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

```
>  xgb_pred <- predict(xgb_model, dtest)
predict.xgb.Booster(xgb_model, dtest)에서 다음과 같은 에러가 발생했습니다:
  Feature names stored in `object` and `newdata` are different!
```

⇒ 해당 오류는 train 자료와 test 자료의 구성이 다르기 때문에 발생

또 비슷한 오류

같은 자료로 split 했고, 똑같은 코드로 matrix 구성을 했는데 또 왜?

07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

```
>  colnames(train_matrix)
 [1] "age"                       "sexFemale"
 [3] "sexMale"                   "cpatypical angina"
 [5] "cpnon-anginal"             "cptypical angina"
 [7] "trestbps"                  "chol"
 [9] "fbsTRUE"                   "restecgnormal"
[11] "restecgst-t abnormality"   "thalch"
[13] "exangTRUE"                 "oldpeak"
[15] "slopeflat"                 "slopeupsloping"
[17] "ca"                        "thalnormal"
[19] "thalreversable defect"
```

```
>  colnames(test_matrix)
 [1] "age"                       "sexFemale"
 [3] "sexMale"                   "cpatypical angina"
 [5] "cpnon-anginal"             "cptypical angina"
 [7] "trestbps"                  "chol"
 [9] "fbsTRUE"                   "restecgnormal"
[11] "thalch"                    "exangTRUE"
[13] "oldpeak"                   "slopeflat"
[15] "slopeupsloping"            "ca"
[17] "thalnormal"                "thalreversable defect"
```

⇒ one-hot-encoding 과정에서, train data에서는 restecg 변수에는 있는 범주가 test data에는 없었기 때문

```
>  table(train_data$restecg)

  lv hypertrophy           normal st-t abnormality
             116              120                4
>  table(test_data$restecg)

lv hypertrophy         normal
            30             29
```

```
>  levels(factor(train_data$restecg))
[1] "lv hypertrophy"   "normal"           "st-t abnormality"
>  levels(factor(test_data$restecg))
[1] "lv hypertrophy" "normal"
```

07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

```
test_data$restecg <- factor(test_data$restecg, levels = levels(factor(train_data$restecg)))
>  table(test_data$restecg)

  lv hypertrophy           normal st-t abnormality
               30               29                0
```

⇒ train_data의 restecg 변수의 factor level을 test data의 restecg 변수에 옮겨 적용

```
test_matrix <- model.matrix(target ~ . - 1, data = select(test_data, c(-dataset)))
test_label <- as.numeric(as.character(select(test_data, c(-dataset))$target))
dtest <- xgb.DMatrix(data = test_matrix, label = test_label)
xgb_pred <- predict(xgb_model, dtest)

>  xgb_pred <- predict(xgb_model, dtest)
>
```

다시 모형 적합 후 오류 발생 없는 것을 확인

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

Confusion matrix와 AUC 구하기

threshold를 0.5로 설정

#ROC 함수

```
logit_auc <- roc(test_data$target, logit_pred)
logit_pred_target<- ifelse(logit_pred>0.5, 1,0)
table(test_data$target, logit_pred_target)
auc(logit_auc)

rf_auc <- roc(test_data$target, rf_pred)
rf_pred_target<- ifelse(rf_pred>0.5, 1,0)
table(test_data$target, rf_pred_target)
auc(rf_auc)

xgb_auc <- roc(test_data$target, xgb_pred)
xgb_pred_target<- ifelse(xgb_pred>0.5, 1,0)
table(test_data$target, xgb_pred_target)
auc(xgb_auc)
```

```
>  table(test_data$target, logit_pred_target)
   logit_pred_target
     0  1
  0 35  3
  1  5 16
>  auc(logit_auc)
Area under the curve: 0.9048
```

```
>  table(test_data$target, rf_pred_target)
   rf_pred_target
     0  1
  0 34  4
  1  7 14
>  auc(rf_auc)
Area under the curve: 0.8872
```

```
>  table(test_data$target, xgb_pred_target)
   xgb_pred_target
     0  1
  0 33  5
  1  6 15
>  auc(xgb_auc)
Area under the curve: 0.8897
```

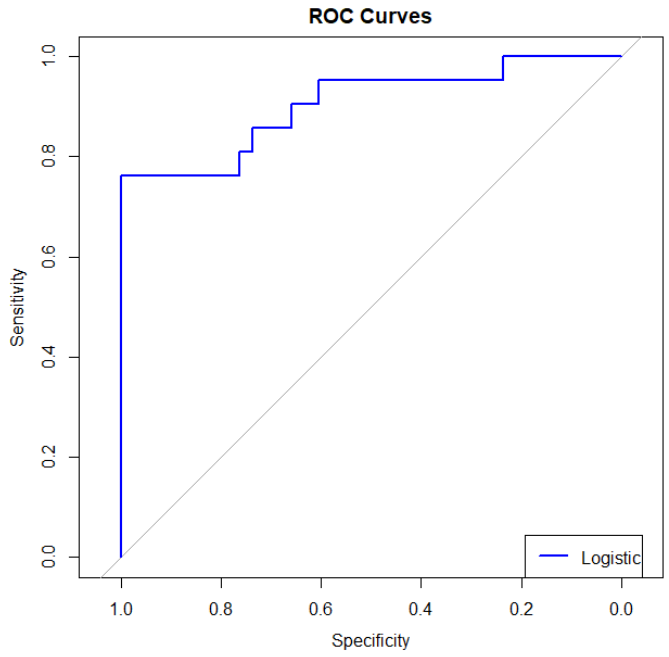
07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

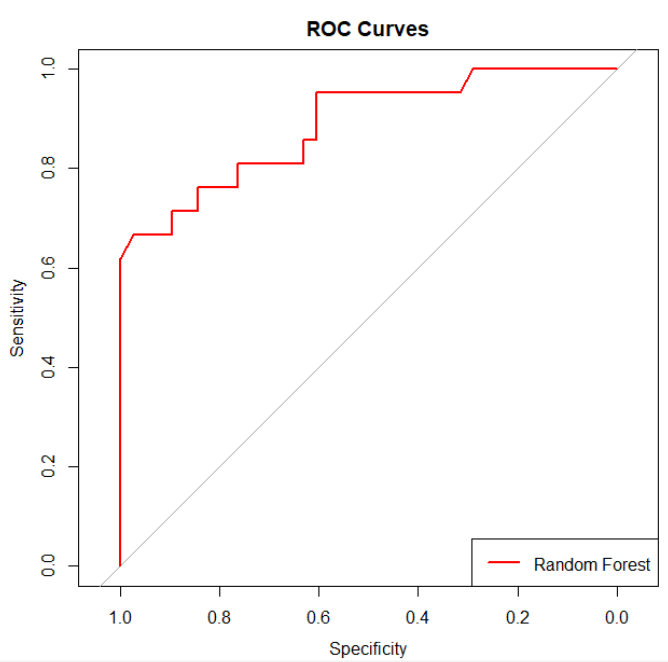
(4) 검증 데이터(Test data)를 통한 모형적합

ROC 곡선 그리기

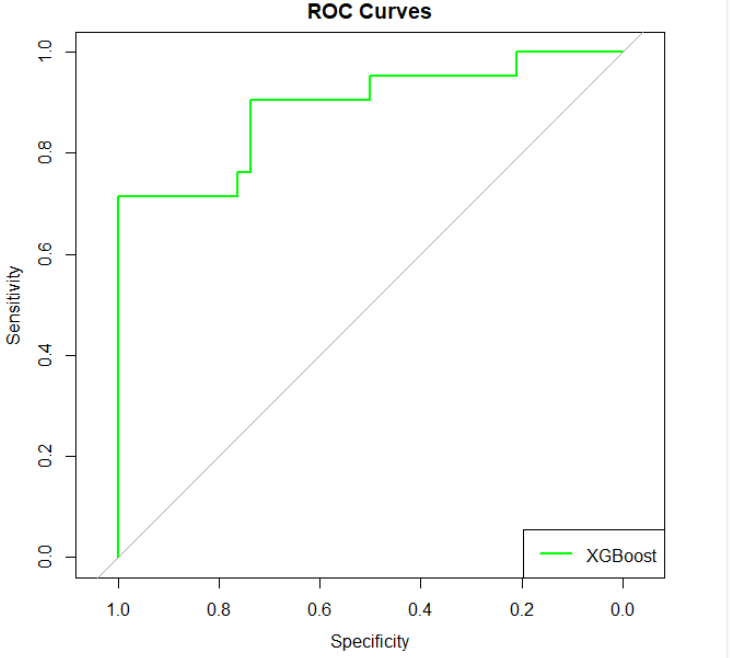
```
plot(logit_auc, col = "blue", main = "ROC Curves")
legend("bottomright", legend = c("Logistic"),
       col = c("blue"), lwd = 2)
```




```
plot(rf_auc, col = "red", main = "ROC Curves")
legend("bottomright", legend = c("Random Forest"),
       col = c("red"), lwd = 2)
```




```
plot(xgb_auc, col = "green", main = "ROC Curves")
legend("bottomright", legend = c("XGBoost"),
       col = c("green"), lwd = 2)
```



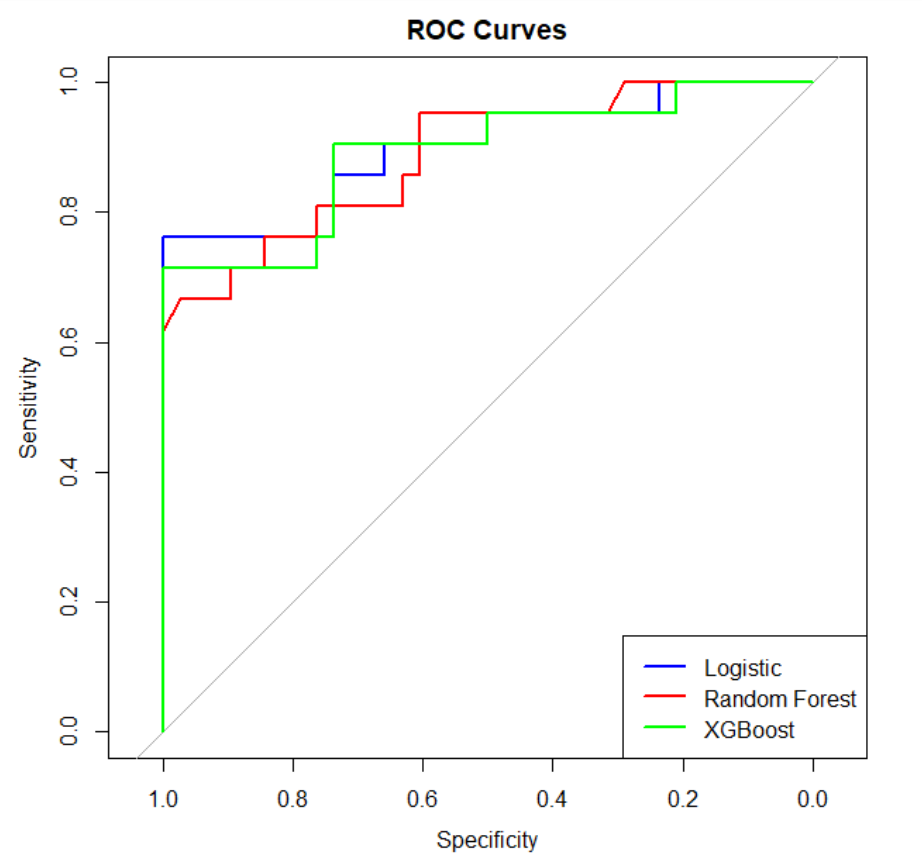
07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

세 그래프를 한번에 그리기

```
plot(logit_auc, col = "blue", main = "ROC Curves")
plot(rf_auc, col = "red", add = TRUE)
plot(xgb_auc, col = "green", add = TRUE)
legend("bottomright",
       legend = c("Logistic", "Random Forest", "XGBoost"),
       col = c("blue", "red", "green"), lwd = 2)
```



ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

#본 예제의 경우 실제값을 앞에, 예측값을 뒤에 놓음

```
TN<-table(test_data$target, logit_pred_target)[1,1]
FP<-table(test_data$target, logit_pred_target)[1,2]
FN<-table(test_data$target, logit_pred_target)[2,1]
TP<-table(test_data$target, logit_pred_target)[2,2]
```

#Confusion matrix의 각 위치를 TN, FP, FN, TP에 할당하고 성능 지표를 계산

```
accuracy <- (TP + TN) / (TN+FP+FN+TP)
sensitivity <- TP / (TP + FN)
specificity <- TN / (TN + FP)
precision <- TP / (TP + FP)
f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)
```

```
print(c(accuracy, sensitivity,specificity,precision, f1_score))

#또는

performance_metrics<- data.frame(
    Accuracy= accuracy,
    Sensitivity=sensitivity,
    Specificity=specificity,
    Precision=precision,
    F1_score=f1_score
    )
```

```
>  performance_metrics
   Accuracy Sensitivity Specificity Precision  F1_score
1 0.8135593   0.6666667   0.8947368 0.7777778 0.7179487
```

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

실습) threshold 조정

우리는 기존 코드에서, 0.5를 기준으로 0과 1을 나누었다.

threshold를 0.5가 아니라 다른 값으로 설정한다면, 그 결과는 어떻게 될까?

각 threshold별로 성과지표를 출력하는 코드를 for 문을 통해 작성해보자.

사용자 정의 함수를 사용할 수도 있다.

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

(hint)

첫번째. 빈 데이터프레임을 만든다.

매트릭스의 형태로 만든 다음 변수 명을 지정하고, 데이터프레임화 할 수도 있다.

두번째. 각 평가 지표를 생성하고, 데이터프레임 또는 행렬의 원하는 위치에 들어가도록 지정한다.

세번째. 각 평가 지표를 생성하는 과정을 반복할 수 있도록 설정한다.

** 물론, 다른 방법을 이용해도 문제 없다.

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

```
threshold<-seq(0.1, 0.9, by=0.1)
logit_result<-matrix(NA,length(threshold),6)
colnames(logit_result)<-c("threshold","Accuracy","Sensitivity",
                          "Specificity","Precision","F1_score")
logit_result<-data.frame(logit_result)

for ( i in 1:length(threshold))
 {
 logit_pred_target<- ifelse(logit_pred>threshold[i],1,0)
 TN<-table(test_data$target, logit_pred_target)[1,1]
 FP<-table(test_data$target, logit_pred_target)[1,2]
 FN<-table(test_data$target, logit_pred_target)[2,1]
 TP<-table(test_data$target, logit_pred_target)[2,2]

 accuracy <- (TP + TN) / (TN+FP+FN+TP)
 sensitivity <- TP / (TP + FN)
 specificity <- TN / (TN + FP)
 precision <- TP / (TP + FP)
 f1_score <- 2 * (precision * sensitivity) /
                 (precision + sensitivity)

 logit_result$threshold[i]<-threshold[i]
 logit_result$Accuracy[i]<-accuracy
 logit_result$Sensitivity[i]<-sensitivity
 logit_result$Specificity[i]<-specificity
 logit_result$Precision[i]<-precision
 logit_result$F1_score[i]<-f1_score
 }

 logit_result
```

```
>   logit_result
  threshold  Accuracy Sensitivity Specificity Precision  F1_score
1       0.1 0.7288136   0.9523810   0.6052632 0.5714286 0.7142857
2       0.2 0.7457627   0.8571429   0.6842105 0.6000000 0.7058824
3       0.3 0.7627119   0.7619048   0.7631579 0.6400000 0.6956522
4       0.4 0.7966102   0.7619048   0.8157895 0.6956522 0.7272727
5       0.5 0.8644068   0.7619048   0.9210526 0.8421053 0.8000000
6       0.6 0.8813559   0.7619048   0.9473684 0.8888889 0.8205128
7       0.7 0.8983051   0.7142857   1.0000000 1.0000000 0.8333333
8       0.8 0.8813559   0.6666667   1.0000000 1.0000000 0.8000000
9       0.9 0.8305085   0.5238095   1.0000000 1.0000000 0.6875000
```

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

```
>   logit_result
  threshold  Accuracy Sensitivity Specificity Precision  F1_score
1       0.1 0.7288136   0.9523810   0.6052632 0.5714286 0.7142857
2       0.2 0.7457627   0.8571429   0.6842105 0.6000000 0.7058824
3       0.3 0.7627119   0.7619048   0.7631579 0.6400000 0.6956522
4       0.4 0.7966102   0.7619048   0.8157895 0.6956522 0.7272727
5       0.5 0.8644068   0.7619048   0.9210526 0.8421053 0.8000000
6       0.6 0.8813559   0.7619048   0.9473684 0.8888889 0.8205128
7       0.7 0.8983051   0.7142857   1.0000000 1.0000000 0.8333333
8       0.8 0.8813559   0.6666667   1.0000000 1.0000000 0.8000000
9       0.9 0.8305085   0.5238095   1.0000000 1.0000000 0.6875000
```

threshold를 낮게 설정할 수록

⇒ 쉽게 양성판정을 하게 되므로, Sensitivity ↑ Specifity ↓

threshold를 높게 설정할 수록

⇒ 양성판정을 하기 어려워지므로, Sensitivity ↓ Specifity ↑

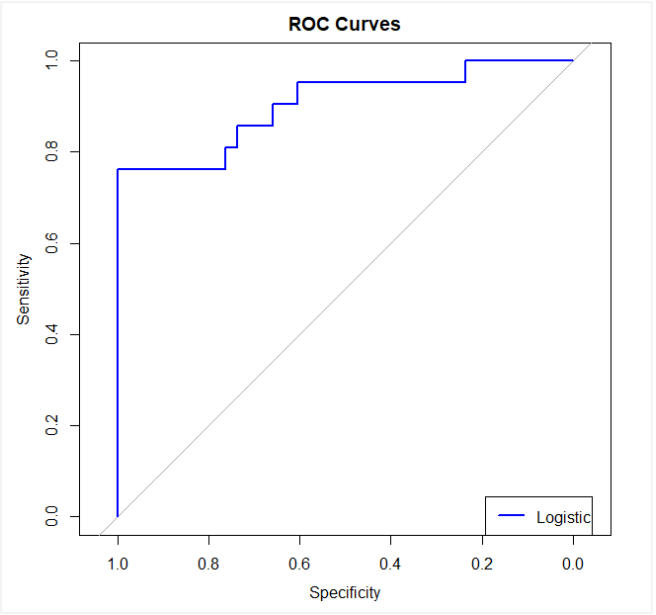
07 ROC 곡선과 모델 평가

ROC 곡선과 AUC(Area Under Curve)

(4) 검증 데이터(Test data)를 통한 모형적합

```
>   logit_result
  threshold  Accuracy Sensitivity Specificity Precision  F1_score
1       0.1 0.7288136   0.9523810   0.6052632 0.5714286 0.7142857
2       0.2 0.7457627   0.8571429   0.6842105 0.6000000 0.7058824
3       0.3 0.7627119   0.7619048   0.7631579 0.6400000 0.6956522
4       0.4 0.7966102   0.7619048   0.8157895 0.6956522 0.7272727
5       0.5 0.8644068   0.7619048   0.9210526 0.8421053 0.8000000
6       0.6 0.8813559   0.7619048   0.9473684 0.8888889 0.8205128
7       0.7 0.8983051   0.7142857   1.0000000 1.0000000 0.8333333
8       0.8 0.8813559   0.6666667   1.0000000 1.0000000 0.8000000
9       0.9 0.8305085   0.5238095   1.0000000 1.0000000 0.6875000
```

ROC curve는 threshold를 조정해가며 변하는 Sensitivity와 Specificity간의 그래프이므로 사회적 비용 및 연구 목적을 고려한 최적 지점을 찾는 것이 중요



본 ROC curve의 경우, 높은 Sensitivity가 중요할 경우 threshold를 0.1 미만으로 낮게 두는 것이 유리

균형적인 성능을 원하는 경우에는 F1-score가 높은 threshold 0.6~0.7 구간을 선택하는 것이 유리

높은 Specificity가 더 중요한 경우 threshold를 0.7 이상으로 설정하는 것이 합리적

감사합니다

Q&A

충북대학교
CHUNGBUK NATIONAL UNIVERSITY