채널 G
2016125077 최재혁
2015125043 어성준
2015125080 표정진

데이터 사이언스

- 9주차 팀 과제 (다양한 모델들의 ROC, AUC 비교)

목 차

데이터 셋 선택, 전처리

- 데이터 셋 결정

  지난 주에 사용했던 heart data set의 독립 변수간에 독립성이 비교적 높게 나타나, 분류모델을 만드는데 괜찮을 것으로 보여 선택함

- 데이터 전처리
  - na.omit() 함수를 이용하여 컬럼에 존재하는 결측치 제거
  - 명목형 변수인 경우, factor()함수를 통하여 순서형 변수로 변환
  - 정수 값은, numeric()함수를 통하여 num type으로 변환

```
> ht <- na.omit(ht[,-1])
```

<사진 1> 결측 값 제거

```
> sex <- as.numeric(ht$Sex)
> cp <- factor(ht$ChestPain, levels = c("typical", "nontypical" ,
+                                       "asymptomatic", "nonanginal"))
> ca <- as.numeric(ht$Ca)
> thal <- factor(ht$Thal, levels = c("fixed", "normal", "reversable"))
> AHD <- factor(ht$AHD, levels = c("No", "Yes"))
```

<사진 3> 형 변환

```
#age - 나이
#sex - (1 = 남성; 0 = 여성)
#cp - 가슴 통증 유형(0, 1, 2, 3, 4)
#trestbps - 안정 혈압(병원 입원시 mm Hg)
#chol - 혈청 콜레스테롤(mg/dl)
#fbs - (공복 혈당 > 120 mg/dl)(1 = true; 0 = false)
#restecg - 안정 심전도 결과(0, 1, 2)
#thalach - 최대 심박동수
#exang - 협심증 유발 운동(1 = yes; 0 = no)
#oldpeak - 비교적 안정되기까지 운동으로 유발되는 ST depression
#slope - 최대 운동 ST segment의 기울기
#ca - 형광 투시된 주요 혈관의 수(0-3)
#thal - (3 = 보통; 6 = 해결된 결함; 7 = 해결가능한 결함)
#target - 심장병 진단(1 = true; 0 = false)
```

<사진 2> 각 속성 값에 대한 설명

유효한 독립변수 선택

- 유의한 변수 추출
  - 다변량 회귀 분석을 통하여 (lm 함수 사용) 모델을 생성하고 확인한 결과, p-value 값이 2.2e-16보다 작기 때문에 회귀 계수들은 유의하며, Pr(>|t|) t통계량의 값이 0.05 보다 작은 4가지 변수를 선택
  - 이를 가지고 새로운 데이터 셋을 구성

```
> #회귀분석을 통해서 유의한 변수 추출
> model0 <- lm(target~.,data = heart2)
> summary(model0)

Call:
lm(formula = target ~ ., data = heart2)

Residuals:
     Min       1Q   Median       3Q      Max
-0.94748 -0.21270  0.06608  0.25022  0.93509

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.8288987  0.2929344   2.830 0.004987 **
age         -0.0008204  0.0026962  -0.304 0.761129
sex         -0.1959956  0.0471429  -4.157 4.24e-05 ***
cp           0.1127034  0.0223816   5.036 8.40e-07 ***
trestbps    -0.0019910  0.0012573  -1.583 0.114407
chol        -0.0003535  0.0004217  -0.838 0.402545
fbs          0.0173736  0.0596669   0.291 0.771125
restecg      0.0498480  0.0399228   1.249 0.212819
thalach      0.0030193  0.0011304   2.671 0.007988 **
exang       -0.1440459  0.0513689  -2.804 0.005387 **
oldpeak     -0.0587887  0.0229269  -2.564 0.010847 *
slope        0.0789788  0.0423896   1.863 0.063453 .
ca          -0.1006022  0.0218565  -4.603 6.25e-06 ***
thal        -0.1190392  0.0356550  -3.339 0.000952 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3542 on 289 degrees of freedom
Multiple R-squared:  0.5175,    Adjusted R-squared:  0.4958
F-statistic: 23.85 on 13 and 289 DF,  p-value: < 2.2e-16
```

<사진 1> 회귀 분석의 결과

```
> str(ht)
'data.frame':   297 obs. of  5 variables:
 $ sex : num  1 1 1 1 0 1 0 0 1 1 ...
 $ cp  : Factor w/ 4 levels "typical","nontypical",..: 1 3 3 4 2 2 3 3 3 3
 $ ca  : num  0 3 2 0 0 0 2 0 1 0 ...
 $ thal: Factor w/ 3 levels "fixed","normal",..: 1 2 3 2 2 2 2 2 3 3 ...
 $ AHD : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 2 1 2 2 ...
```

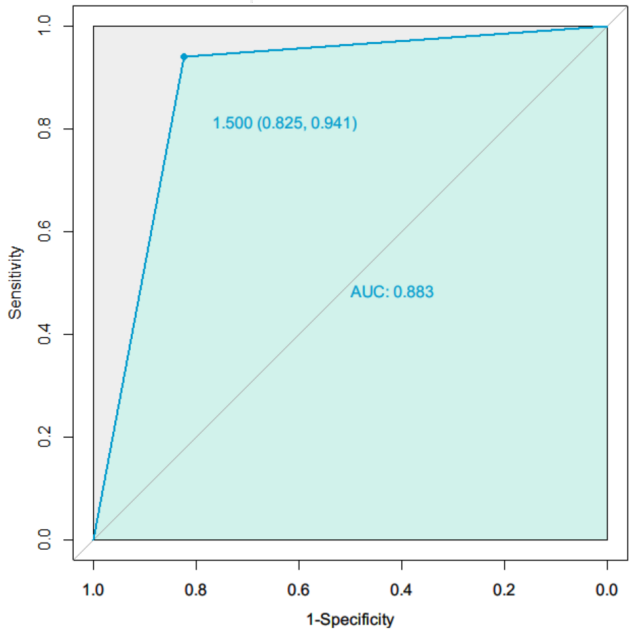
<사진 2> 전처리 후 Data set 의 특성

인공신경망 모델

- 인공신경망 모델 생성
  - 지난주 사용했던 모델을 가지고 nnet 모델을 생성하였다

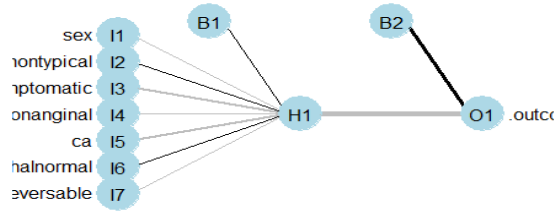
- ROC 함수 & AUC

```
> nn_pred1 <- ROCR::prediction(as.numeric(predict(nnet, newdata = test, type="raw")), as.numer
ic(test$AHD))
> nn_model1.roc <- performance(nn_pred1, "tpr", "fpr")
> win.graph()
> plot(nn_model1.roc, main='ROC of Test Data')
> performance(nn_pred1, measure = "auc")
```



<사진 3> 모델의 ROC / AUC

```
> library(devtools)
에러: package or namespace load failed for 'devtools' in loadNamespace(j <- i[[1L]], c(lib.loc,
 .libPaths()), versionCheck = vI[[j]]):
 'digest'이라고 불리는 패키지가 없습니다
추가정보: 경고메시지(들):
패키지 'devtools'는 R 버전 4.0.3에서 작성되었습니다
> source('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516
c34f1a4684a5/nnet_plot_update.r')
> plot.nnet(nnet)
```



<사진 1> 신경망 모델

```
> confusionMatrix(predict(nnet, newdata = test, type="raw"), test$AHD)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  34   4
       Yes  6  30

               Accuracy : 0.8649
                 95% CI : (0.7655, 0.9332)
    No Information Rate : 0.5405
    P-Value [Acc > NIR] : 2.913e-09

                  Kappa : 0.7291

 Mcnemar's Test P-Value : 0.7518

            Sensitivity : 0.8500
            Specificity : 0.8824
         Pos Pred Value : 0.8947
         Neg Pred Value : 0.8333
             Prevalence : 0.5405
         Detection Rate : 0.4595
   Detection Prevalence : 0.5135
      Balanced Accuracy : 0.8662

       'Positive' Class : No
```

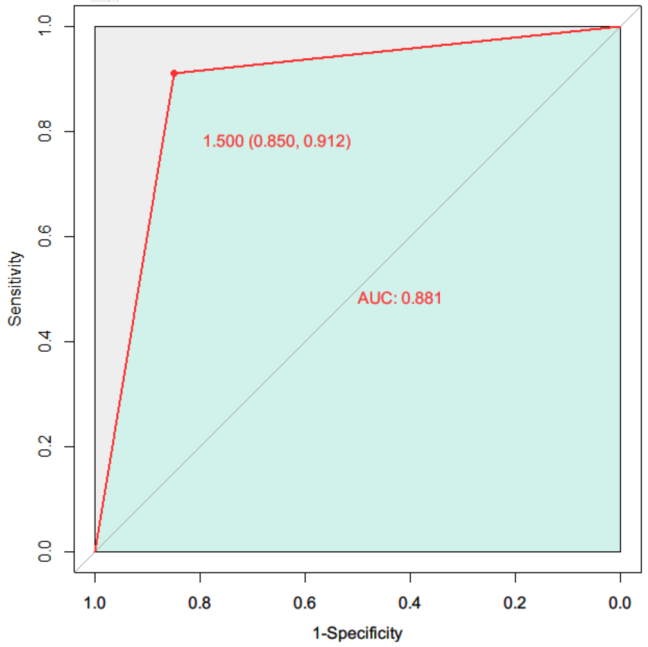
<사진 2> 모델의 혼동행렬

의사결정나무 모델

- 의사결정나무 모델 생성
  - Rpart2 메서드를 이용하여 의사결정나무 생성, 최적의 깊이로 tuning

- ROC 함수 & AUC

```
> nn_pred2 <- ROCR::prediction(as.numeric(predict(dt, newdata = test, type="class")),as.numeri
c(test$AHD))
> nn_pred2
A prediction instance
  with 74 data points
> nn_model2.roc <- performance(nn_pred2, "tpr", "fpr")
> win.graph()
> plot(nn_model2.roc, main='ROC of Test Data')
```



<사진 3> 모델의 ROC / AUC

```
> #결정나무
> dt_model <- train(AHD~.,
+                   data = train,
+                   method = "rpart2",
+                   metric = "ROC",
+                   tuneLength = 10,
+                   trControl = fitControl)
note: only 3 possible values of the max tree depth from the initial fit.
 Truncating the grid to 3 .

> bestdepth <- dt_model$results[which.max(dt_model$results[,"ROC"]),"maxdepth"]
> bestdepth
[1] 6
```

<사진 1> 결정나무 모델 생성 depth tuning

```
> dt <- rpart(AHD~., data = train, maxdepth = bestdepth)
> confusionMatrix(predict(dt, newdata = test,type="class"),test$AHD)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  34   3
       Yes  6  31

               Accuracy : 0.8784
                 95% CI : (0.7816, 0.9429)
    No Information Rate : 0.5405
    P-Value [Acc > NIR] : 5.148e-10

                  Kappa : 0.7568

 Mcnemar's Test P-Value : 0.505

            Sensitivity : 0.8500
            Specificity : 0.9118
         Pos Pred Value : 0.9189
         Neg Pred Value : 0.8378
             Prevalence : 0.5405
         Detection Rate : 0.4595
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8809

       'Positive' Class : No
```

<사진 2> 모델의 혼동 행렬

랜덤포레스트 모델

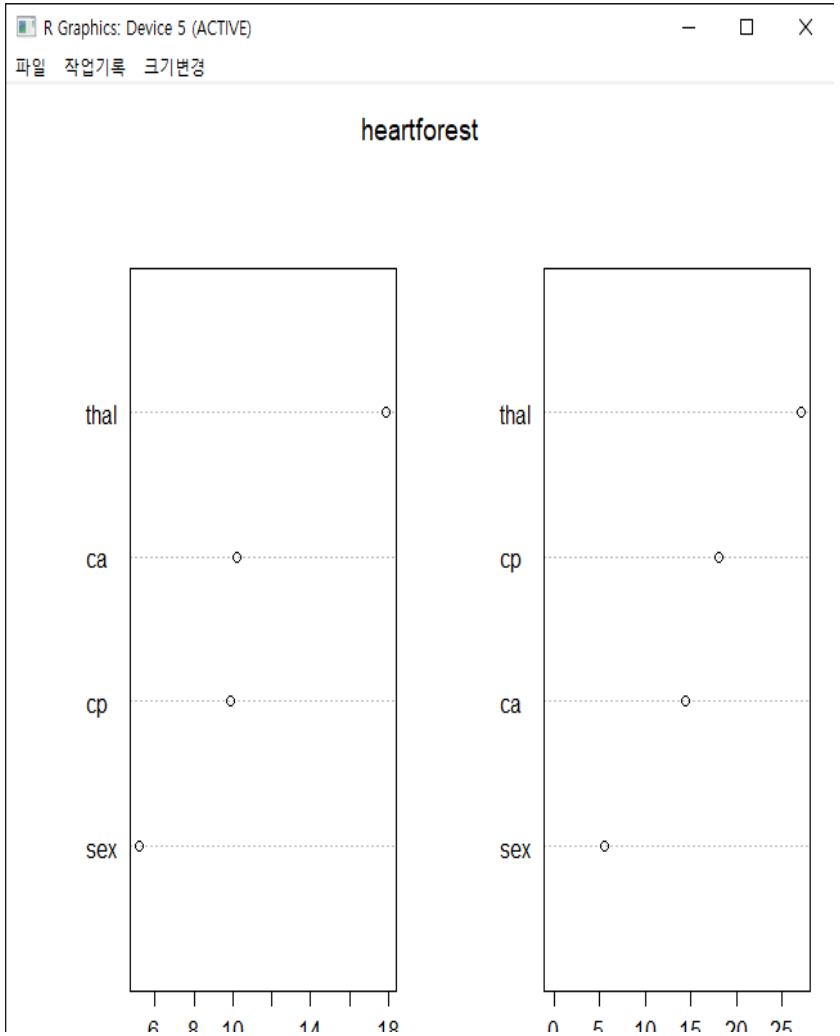
- 랜덤포레스트 모델 생성
  - 앙상블 기법 중의 하나인 랜덤 포레스트 모델을 생성
  - 변수의 중요도를 그래프로 표현
  - 그래프 이외에도 model$importance 옵션으로 출력
  - 트리의 총 개수는 100개로 설정

```
> #randomforest
> heartforest<-randomForest(AHD ~ ., data=train, ntree=100, importance = T)
> heartforest

Call:
 randomForest(formula = AHD ~ ., data = train, ntree = 100, importance = T)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 18.83%
Confusion matrix:
     No Yes class.error
No  106  14   0.1166667
Yes  28  75   0.2718447
> heartforest$importance
             No        Yes MeanDecreaseAccuracy MeanDecreaseGini
sex  0.01401152 0.01989906           0.01680768         5.521083
cp   0.03601287 0.05077731           0.04226401        18.077355
ca   0.05693120 0.05304975           0.05477826        14.369448
thal 0.12490716 0.10048334           0.11236952        26.995531
> windows()
> varImpPlot(heartforest)
```

<사진 1> random forest 모델과 변수의 중요도



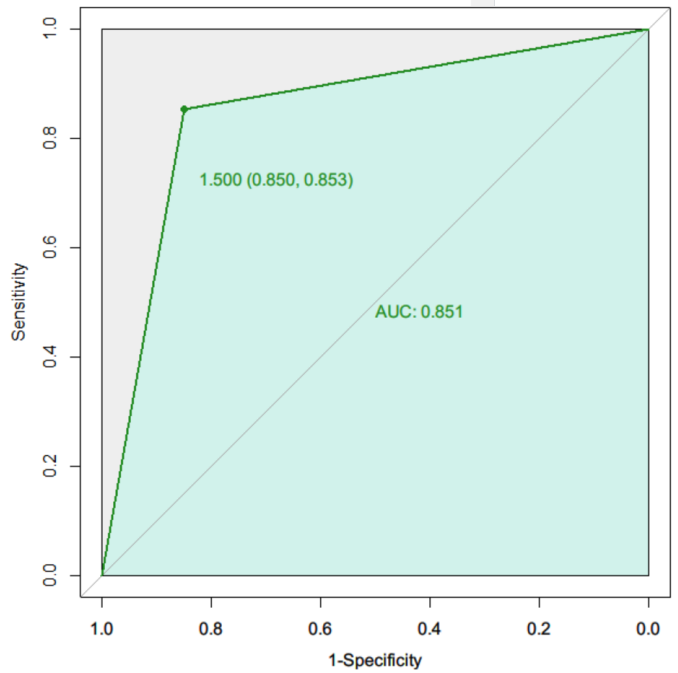
<사진 2> 변수의 중요도 그래프

랜덤 포레스트 모델 예측 과 ROC

- 랜덤 포레스트 모델 예측
  - Test 데이터 셋을 비교하여 정확도를 측정

- ROC 함수 & AUC

```
> heartforest.roc<-roc(train$AHD, heartforest$votes[,2], direction = "<",levels = c(control =
  "No", case = "Yes"))
> levels(train$AHD)
[1] "No"  "Yes"
> plot(heartforest.roc,colorize=TRUE)
```



<사진 2> 모델의 ROC / AUC

```
> confusionMatrix(predict(heartforest,newdata = test,type = "class"),test$AHD)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  35   5
       Yes  5  29

               Accuracy : 0.8649
                 95% CI : (0.7655, 0.9332)
    No Information Rate : 0.5405
    P-Value [Acc > NIR] : 2.913e-09

                  Kappa : 0.7279

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8750
            Specificity : 0.8529
         Pos Pred Value : 0.8750
         Neg Pred Value : 0.8529
             Prevalence : 0.5405
         Detection Rate : 0.4730
   Detection Prevalence : 0.5405
      Balanced Accuracy : 0.8640

       'Positive' Class : No
```

<사진 1> random forest 모델 혼동행렬

결론 및 고찰

- 결론

  세 모델 간의 성능비교

- 고찰

  다양한 랜덤 포레스트와 성능비교

세 모델 간의 성능 비교

- ROC 곡선을 기반으로 모델 간의 특성 비교

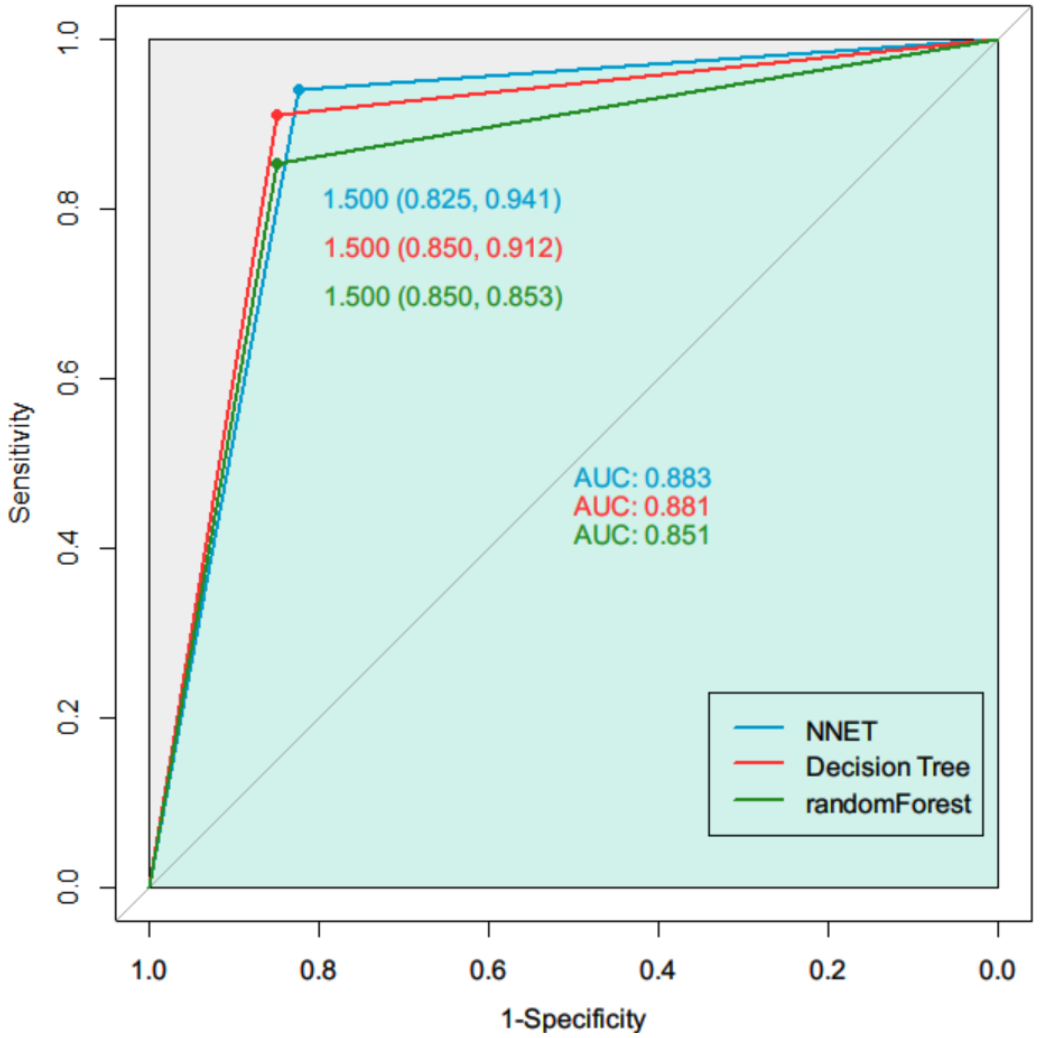
  선이 꺾이는 점이 [1,1]과 가까울수록 성능이 좋다

  인공신경망 모델 : $\sqrt{(1-0.825)^2+(1-0.941)^2} = 0.185$
  의사결정나무 모델 : $\sqrt{(1-0.850)^2+(1-0.912)^2} = 0.174$
  랜덤포레스트 모델 : $\sqrt{(1-0.850)^2+(1-0.853)^2} = 0.210$

  의사결정나무 모델이 가장 좋은 것으로 판단 가능하다.

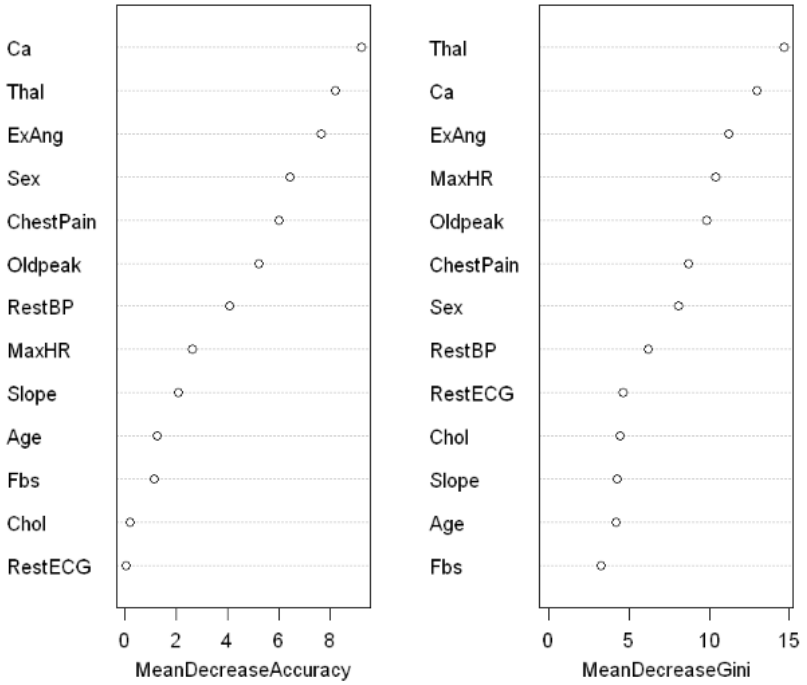
- AUC를 기반으로 모델 간의 성능 비교

  AUC (Area under ROC Curve)가 클수록 성능이 좋다.
  AUC 값이 0.883으로 가장 큰 인공신경망 모델의 성능이
  가장 좋은 것으로 판단 가능하다.

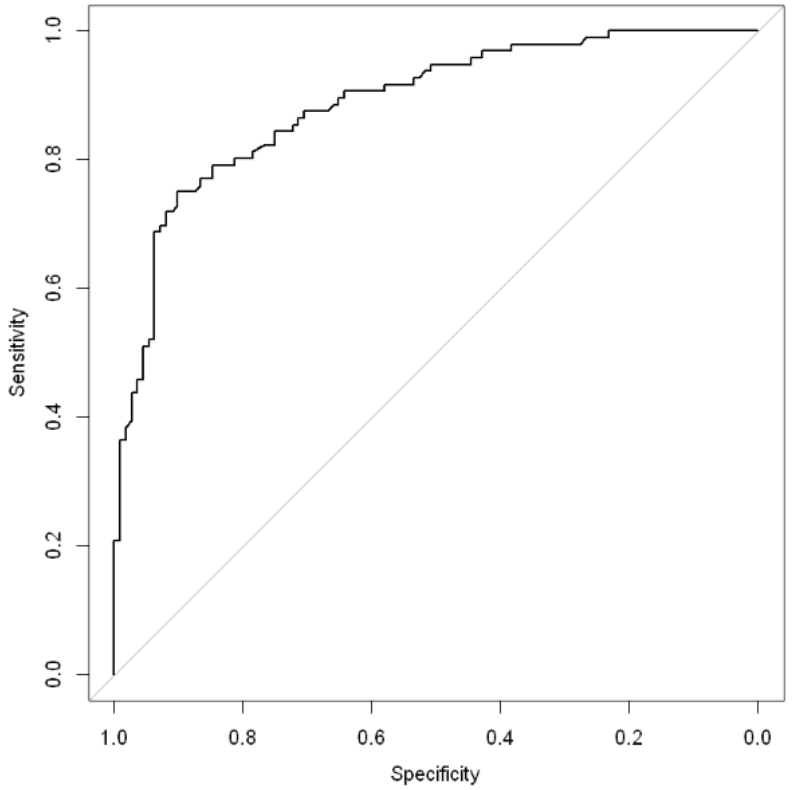


고찰

- 랜덤 포레스트 : 1) 모든 변수를 포함



<사진 1> 변수의 중요도 그래프

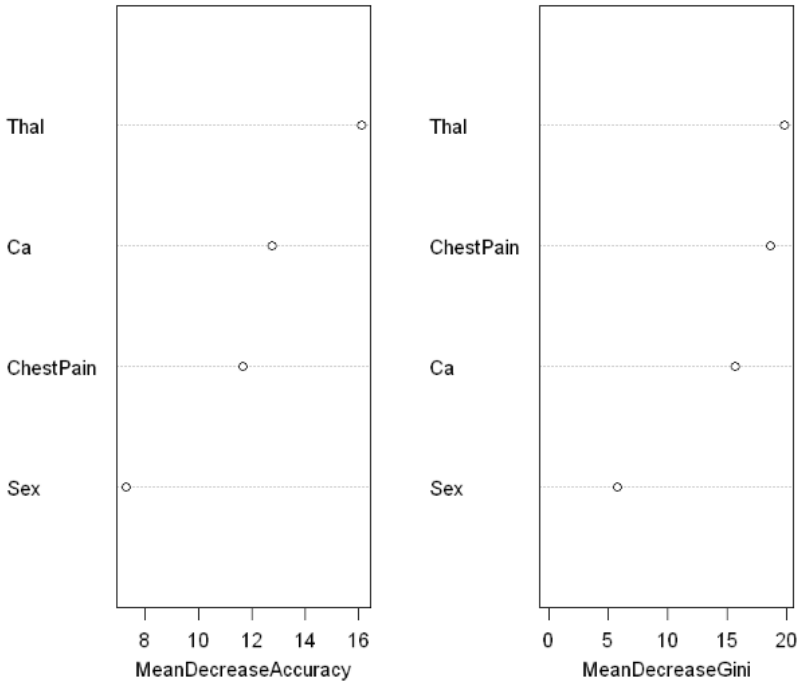


<사진 2> 모델의 ROC

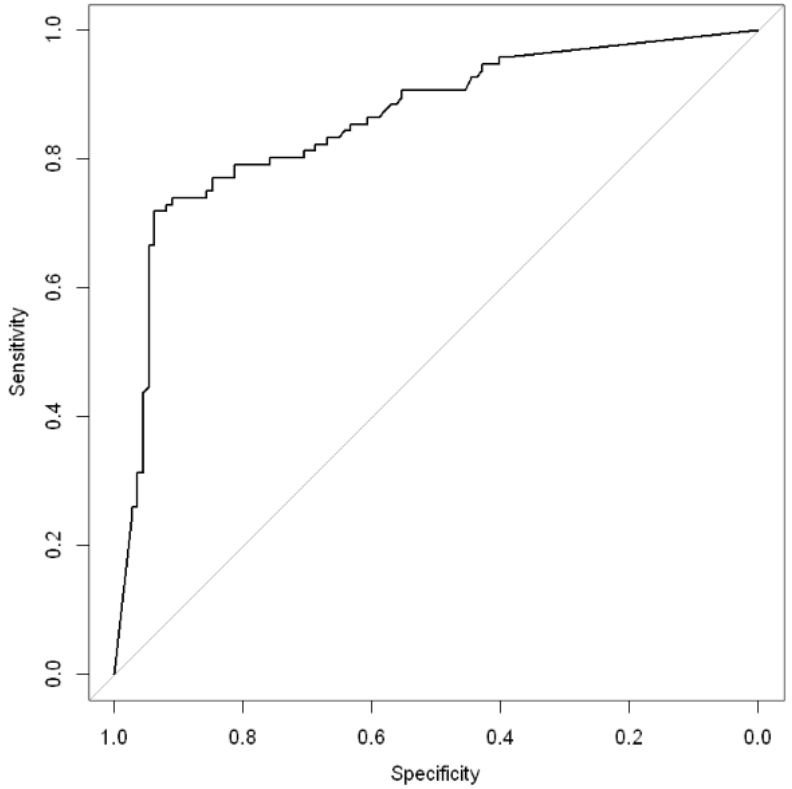
```
Accuracy : 0.8315
Sensitivity : 0.8333
Specificity : 0.8293
AUC: 0.886997767857143
```

고찰

- 랜덤 포레스트 : 2) 다변량 회귀 분석을 통한 변수 선택



<사진 1> 변수의 중요도 그래프

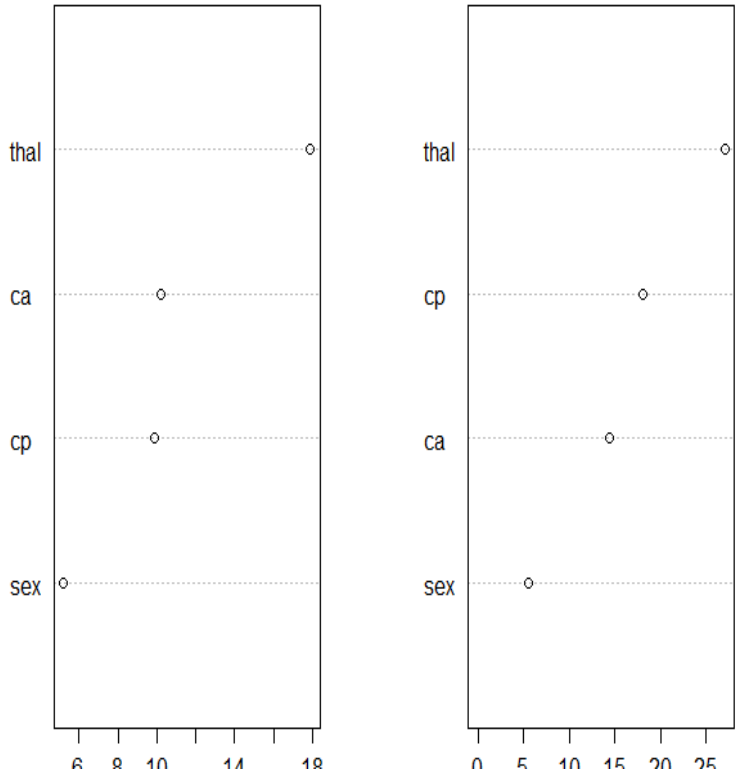


<사진 2> 모델의 ROC

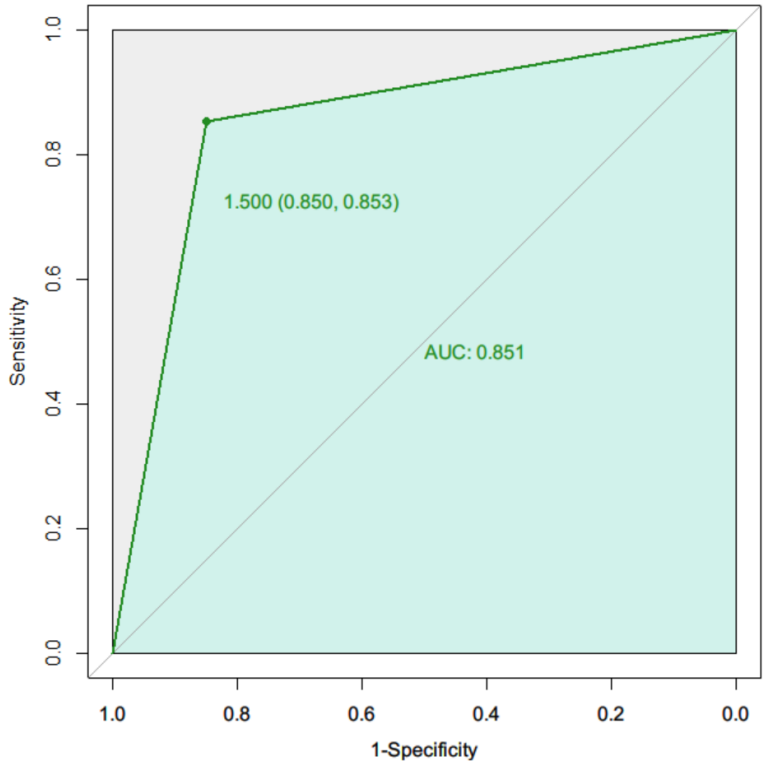
```
Accuracy : 0.8652
Sensitivity : 0.8750
Specificity : 0.8537
AUC: 0.858258928571429
```

고찰

- 랜덤 포레스트 : 3) train(…, method="rf")



<사진 1> 변수의 중요도 그래프



<사진 2> 모델의 ROC

```
Accuracy : 0.8649
Sensitivity : 0.8750
Specificity : 0.8529
AUC: 0.851
```

고찰

- 결과 비교

	AUC	Accuracy	Sensitivity	Specificity
모든 변수	0.8870	0.8315	0.8333	0.8293
다변량 회귀	0.8583	0.8652	0.8750	0.8537
train(...)	0.851	0.8649	0.8750	0.8529

<표 1> 랜덤 포레스트 유형에 따른 성능비교

일반적으로 AUC 가 높을수록 이상적인 성능을 가진다고 할 수 있습니다.

그러나 모델 생성 후, testData를 넣고 여러 성능 척도를 비교한 결과 AUC값이 낮은 모델에서 더 높은 성능을 보였습니다.