Heart Failure Prediction

데이터 마이닝 – 기말프로젝트

제출일	2021. 12. 02	학과	정보통계·보험수리학과
과목명	데이터 마이닝	담당 교수	정원일 교수님
학번	20171421	이름	김성연
학번	20171442	이름	손형락

목차

1. 탐색적 데이터 분석(EDA)

1.1 Dataset 탐색

<CODE>

```
# 데이터 로드
heart <- read.csv("heart.csv")
head(heart)
```

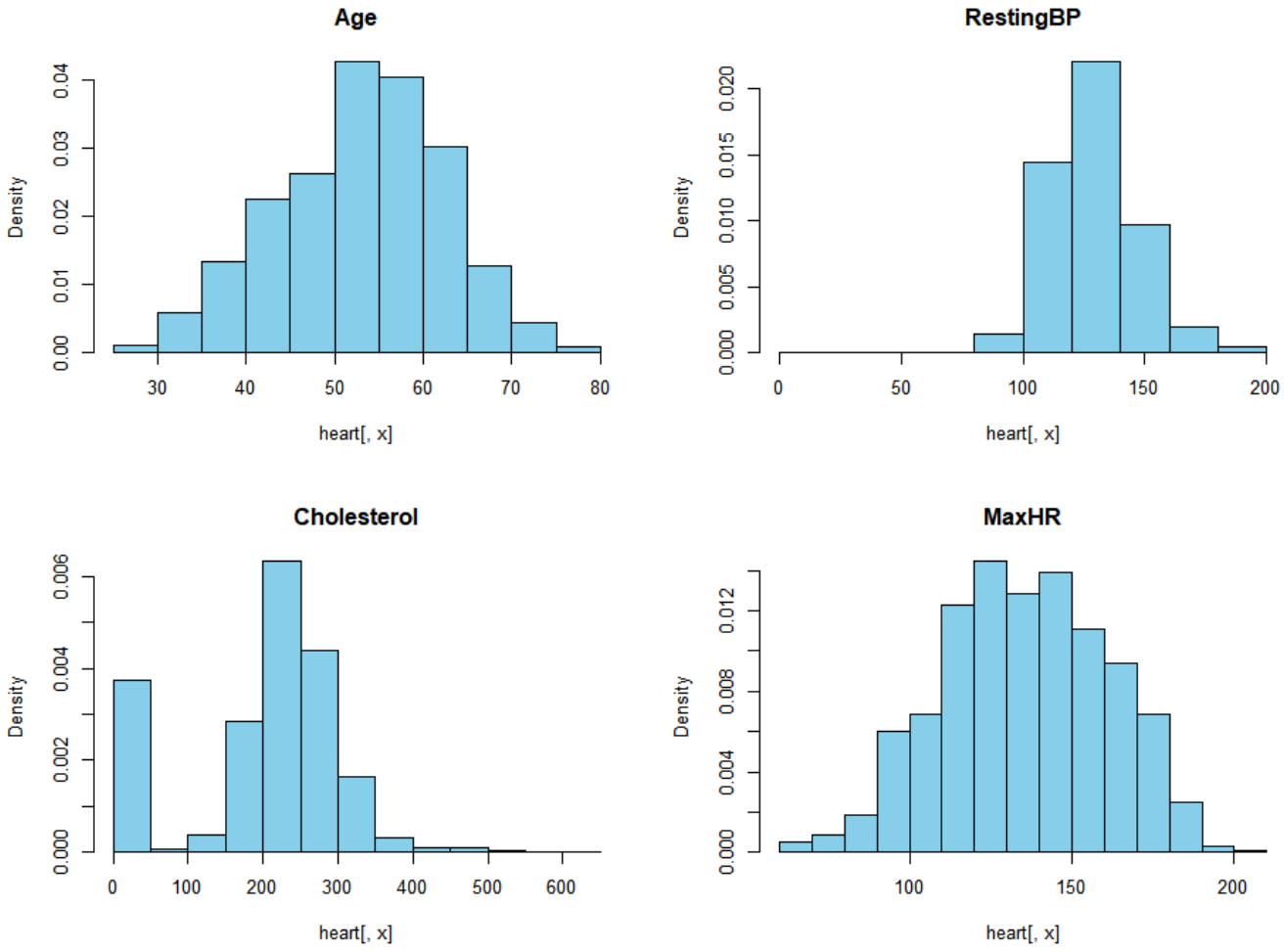
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR
1	40	M	ATA	140	289	0	Normal	172
2	49	F	NAP	160	180	0	Normal	156
3	37	M	ATA	130	283	0	ST	98
4	48	F	ASY	138	214	0	Normal	108
5	54	M	NAP	150	195	0	Normal	122
6	39	M	NAP	120	339	0	Normal	170

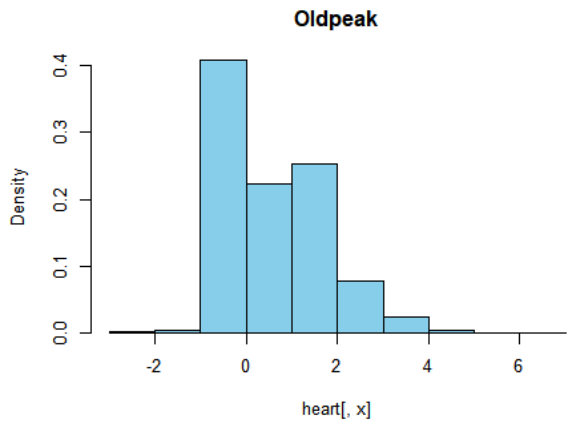
	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
1	N	0.0	Up	0
2	N	1.0	Flat	1
3	N	0.0	Up	0
4	Y	1.5	Flat	1
5	N	0.0	Up	0
6	N	0.0	Up	0

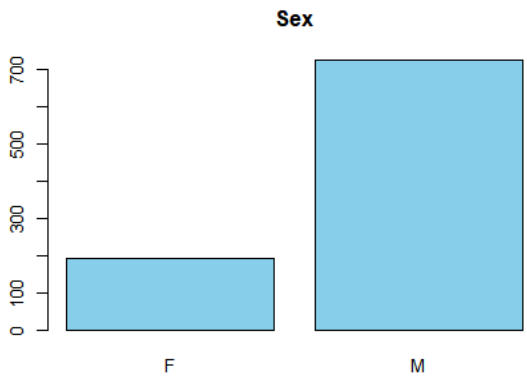
```
# 이산형 자료(문자형)를 요인형으로 변경한다.
heart$Sex <- factor(heart$Sex)
heart$ChestPainType <- factor(heart$ChestPainType)
heart$RestingECG <- factor(heart$RestingECG)
heart$ExerciseAngina <- factor(heart$ExerciseAngina)
heart$ST_Slope <- factor(heart$ST_Slope)

# 각 열에서 NA의 여부를 판단해본다. – 데이터 자체의 NA값은 존재하지 않는다.
sapply(names(heart), function(x) sum(is.na(heart$x)))
```

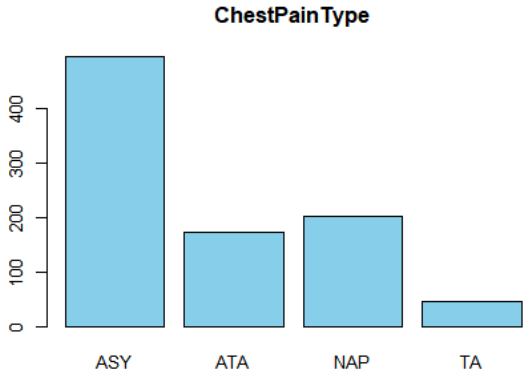
Age	Sex	ChestPainType	RestingBP	Cholesterol
0	0	0	0	0
FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak
0	0	0	0	0
ST_Slope	HeartDisease			
0	0			

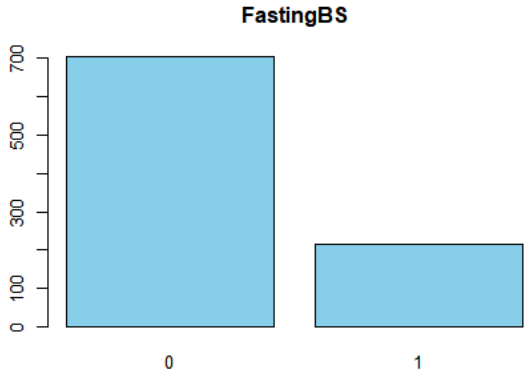
```
# 수치형 / 이산형 데이터로 분리
num_feature <- c("Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak")
qual_feature <- setdiff(names(heart), num_feature)

# 수치형 자료의 분포를 확인한다.
par(mfrow = c(2,2))
sapply(num_feature, function(x) {
  hist(heart[,x], main = x, freq = F, col = "skyblue")
  })

# 이산형 자료의 분포를 확인한다.
sapply(qual_feature, function(x) {
  barplot(table(heart[,x]), col = "skyblue", main = x)
})
```

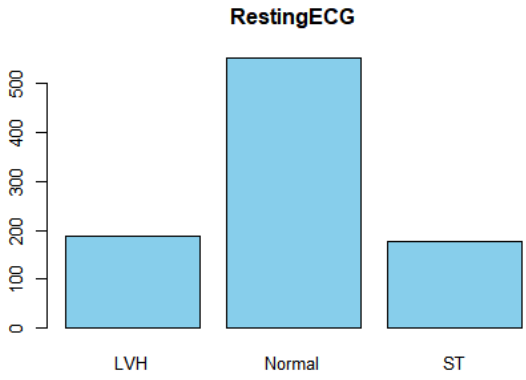


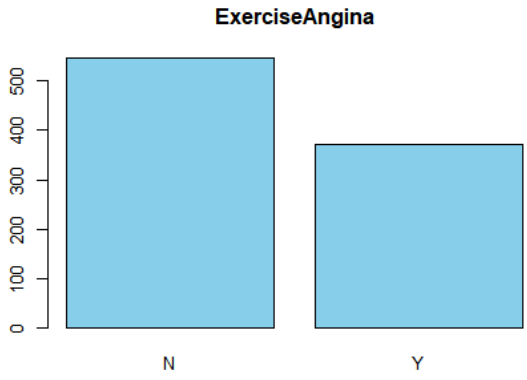

Oldpeak
Density
0.0
0.1
0.2
0.3
0.4
-2
0
2
4
6
heart[, x]

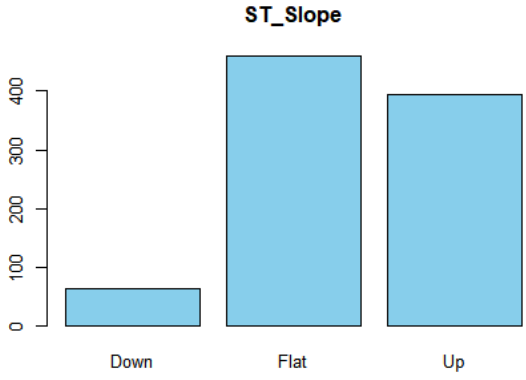

Sex
0
100
300
500
700
F
M

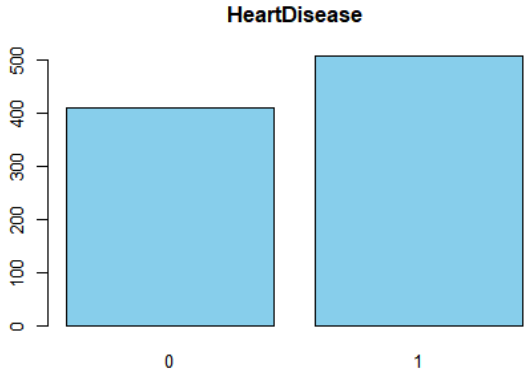

ChestPainType
0
100
200
300
400
ASY
ATA
NAP
TA


FastingBS
0
100
300
500
700
0
1


RestingECG
0
100
200
300
400
500
LVH
Normal
ST


ExerciseAngina
0
100
200
300
400
500
N
Y


ST_Slope
0
100
200
300
400
Down
Flat
Up


HeartDisease
0
100
200
300
400
500
0
1

-▶ 먼저 수치형 변수 히스토그램에서는 대체로 정규분포와 유사한 형태의 히스토그램이 그려진다. 정규성을 크게 의심하지 않아도 좋을 것 같다. Cholesterol 변수는 0근처에서 밀도가 높아지는 형상이 나타난다. 뒤에서 자세히 확인해보겠다. 범주형 변수에선 데이터 내 남자가 많은 것이 눈에 띄며 가슴 통증이 무증상인 사람이 많다. 공복 혈당 수치도 정상인 사람이 대부분이다. 또한 심장병 환자가 맞는 사람과 아닌 사람의 비율이 거의 비슷하다.

1.2 RestingBP, Cholesterol 변수의 심층적 탐색

 # 종합적 데이터 확인

```
summary(heart)
```

```
      Age          Sex         ChestPainType     RestingBP       Cholesterol       FastingBS      
 Min.   :28.00   F:193   ASY:496   Min.   :  0.0   Min.   :  0.0   Min.   :0.0000  
 1st Qu.:47.00   M:725   ATA:173   1st Qu.:120.0   1st Qu.:173.2   1st Qu.:0.0000  
 Median :54.00           NAP:203   Median :130.0   Median :223.0   Median :0.0000  
 Mean   :53.51           TA : 46   Mean   :132.4   Mean   :198.8   Mean   :0.2331  
 3rd Qu.:60.00                     3rd Qu.:140.0   3rd Qu.:267.0   3rd Qu.:0.0000  
 Max.   :77.00                     Max.   :200.0   Max.   :603.0   Max.   :1.0000  

   RestingECG        MaxHR          ExerciseAngina     Oldpeak        ST_Slope     HeartDisease   
 LVH   :188   Min.   : 60.0   N:547   Min.   :-2.6000   Down: 63   Min.   :0.0000  
 Normal:552   1st Qu.:120.0   Y:371   1st Qu.: 0.0000   Flat:460   1st Qu.:0.0000  
 ST    :178   Median :138.0           Median : 0.6000   Up   :395   Median :1.0000  
              Mean   :136.8           Mean   : 0.8874               Mean   :0.5534  
              3rd Qu.:156.0           3rd Qu.: 1.5000               3rd Qu.:1.0000  
              Max.   :202.0           Max.   : 6.2000               Max.   :1.0000  
```

-▶ 눈 여겨볼 변수들은 Cholesterol(콜레스테롤), RestingBP(안정기 혈압)이다. 최소값이 0인 경우가 존재한다. 안정기 혈압이 0의 의미는 일반적으로 맥박 순환이 없다는 것이므로 본 프로젝트에서는 이상값으로 판단하였다. 콜레스테롤이 0인경우도 확인해 보아야 할 것이다.

RestingBP 변수

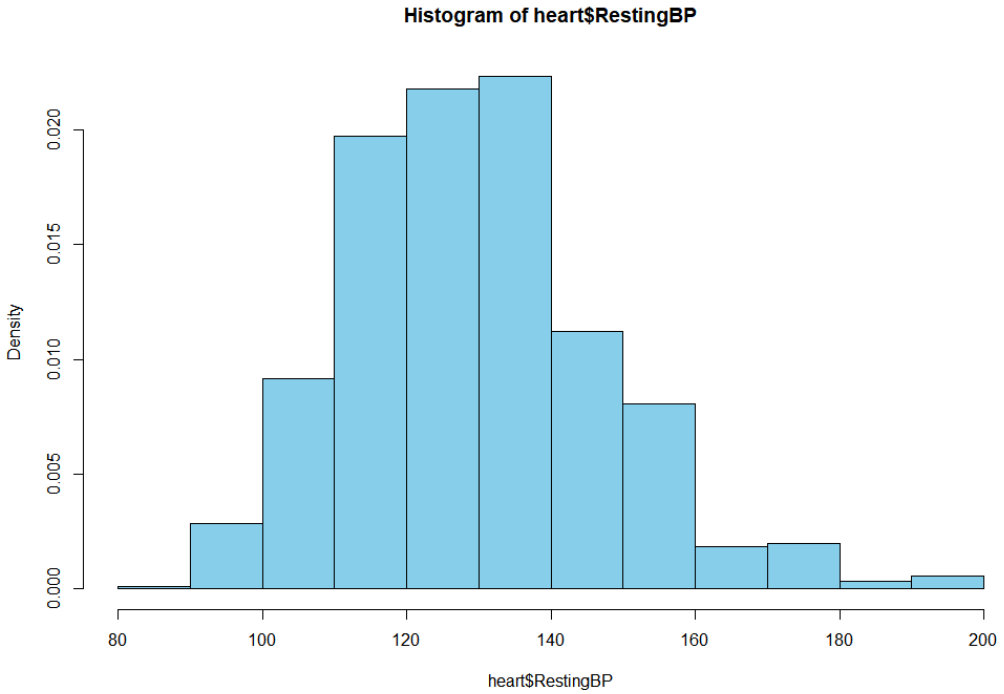
 # 안정기 혈압이 0인 행을 확인. (450번째 1개 행 존재)

```
heart[which(heart$RestingBP == 0),]
```

```
    Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR ExerciseAngina
450  55   M          NAP          0           0         0     Normal   155              N
    Oldpeak ST_Slope HeartDisease
450     1.5     Flat            1
```

 # 안정기 혈압이 0인 행을 삭제한다.

```
heart <- heart[-which(heart$RestingBP == 0),]
# RestingBP 이상치 제거 후 분포
par(mfrow = c(1,1))
hist(heart$RestingBP, col = "skyblue", freq = F)
```



-▶ 이상치를 제거한 후 정규분포에 더 가까워진 모습을 보인다.

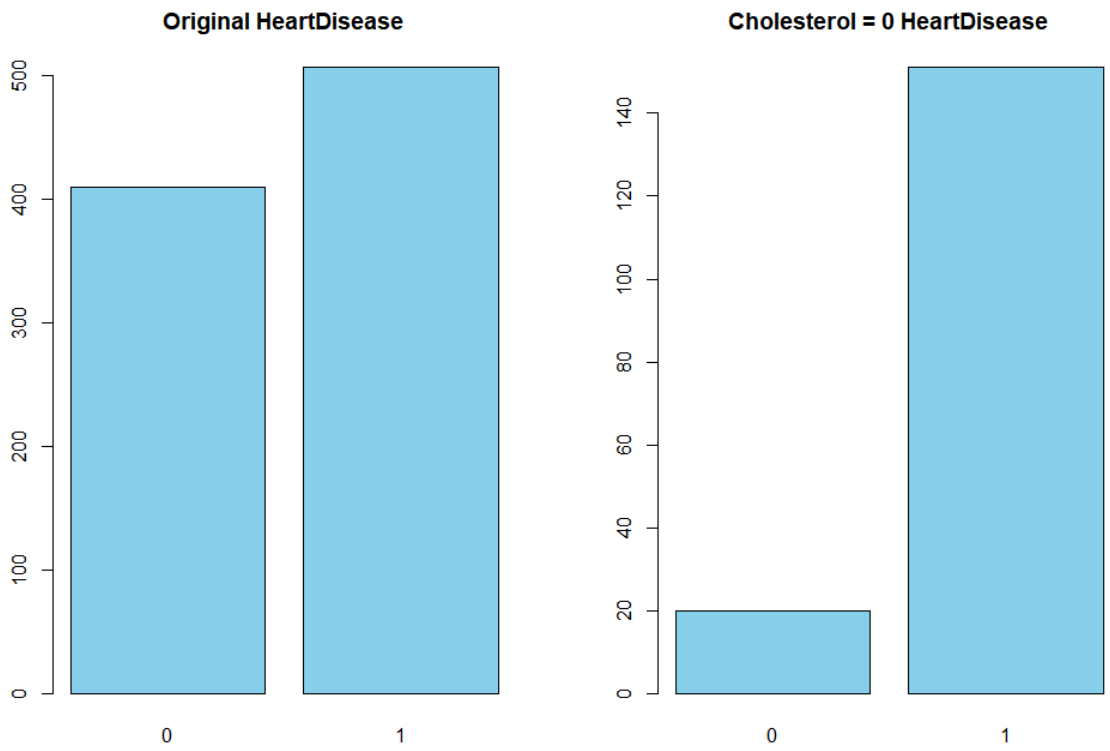
Cholesterol 변수

```
sum(heart$Cholesterol == 0)
```

[1] 171

-▶ 콜레스테롤이 0인경우는 일반적으로는 받아들이기 어려운 수치이다. 하지만 데이터의 약 15%정도를 차지 하기 때문에 함부로 제거하게 되면 예측에 문제가 발생할 수도 있다.

```
par(mfrow = c(1,2))
barplot(table(heart$HeartDisease), col = "skyblue", main = "Original HeartDisease")
barplot(table(heart[heart$Cholesterol == 0,]$HeartDisease), main = "Cholesterol = 0 HeartDisease",col = "skyblue")
par(mfrow = c(1,1))
```



-▶ 왼쪽 barplot은 original 데이터의 심장병 변수 빈도를 나타낸 것이고 오른쪽 barplot은 Cholesterol 변수가 0인 데이터 171개의 심장병 변수 빈도를 나타낸 것이다. Cholesterol 변수가 0일때 확연히 심장병일 확률이 높은 것을 확인해 볼 수 있다. 데이터가 어떻게 구성되었는지 정확히 모르기 때문에 함부로 이상치로 판단할 수 없다. 분류모델에서 자연스럽게 이 부분을 판단해줄 것으로 기대하고 제거하지 않는다.

2. Classification Model 적합 - 1

2.1 Dataset 분리 / K-Fold Dataset 분리

```
# 훈련데이터와 검정데이터로 분리한다.
set.seed(45)
train <- sample(1:nrow(heart), nrow(heart) * 0.7)
test <- setdiff(1:nrow(heart), train)

y_test <- heart$HeartDisease[test]

# K-fold CV는 Random하게 섞어서 분리한다.
set.seed(45)
kfoldseed <- sample(1:nrow(heart), nrow(heart))

k <- 10
kfold <- split(kfoldseed, f = ceiling(seq_along(kfoldseed)/round(nrow(heart)/k)))
```

2.2 Bagging Classification

```
# ------------------- (1) bagging -------------------------
library(randomForest)
set.seed(45)
bagg <- randomForest(HeartDisease ~., heart, subset = train, mtry = ncol(heart) - 3, ntree = 1000)

# 예측
pred_bagg <- predict(bagg, heart[test, ])
prob_bagg <- predict(bagg, heart[test, ], type = "prob")

# Confusion Matrix
(confu <- table(y_test, pred_bagg))
```

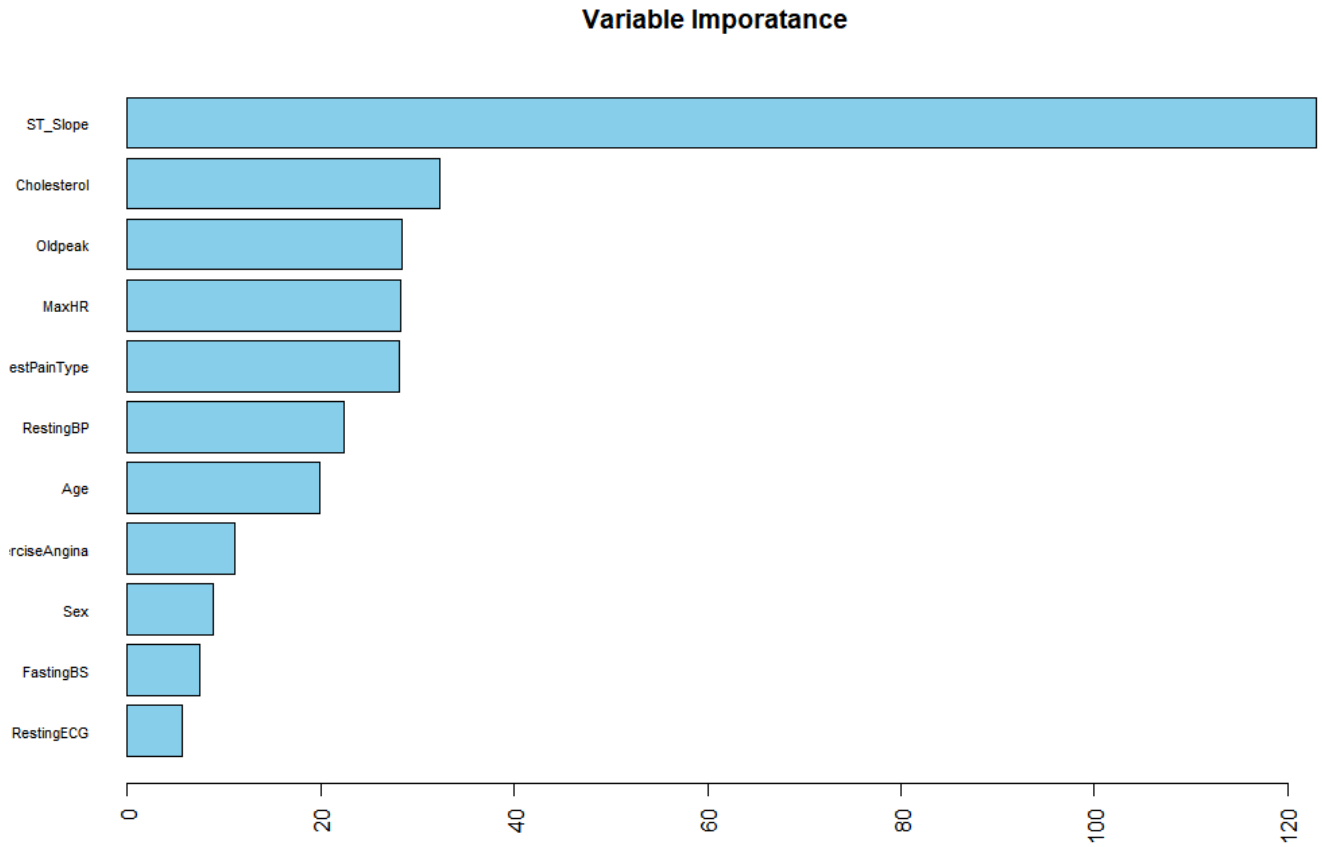
```
        pred_bagg
y_test    0    1
     0 110   17
     1   21 128
```

```
# Accuracy + sensitivity + specificity
(acc_bagg <- sum(confu[col(confu) == row(confu)]) / sum(confu))
(sens_bagg <- confu[2,2] / sum(confu[,2]))
(spec_bagg <- confu[1,1] / sum(confu[,1]))
```

Accuracy = 0.8623188
Sensitivity = 0.8827586
Specificity = 0.8396947

 # 변수 중요도 확인

```
imp <- importance(bagg)
barplot(sort(imp[,1]), horiz = T, las = 2, main = 'Variable Imporatance', cex.names = 0.7)
```



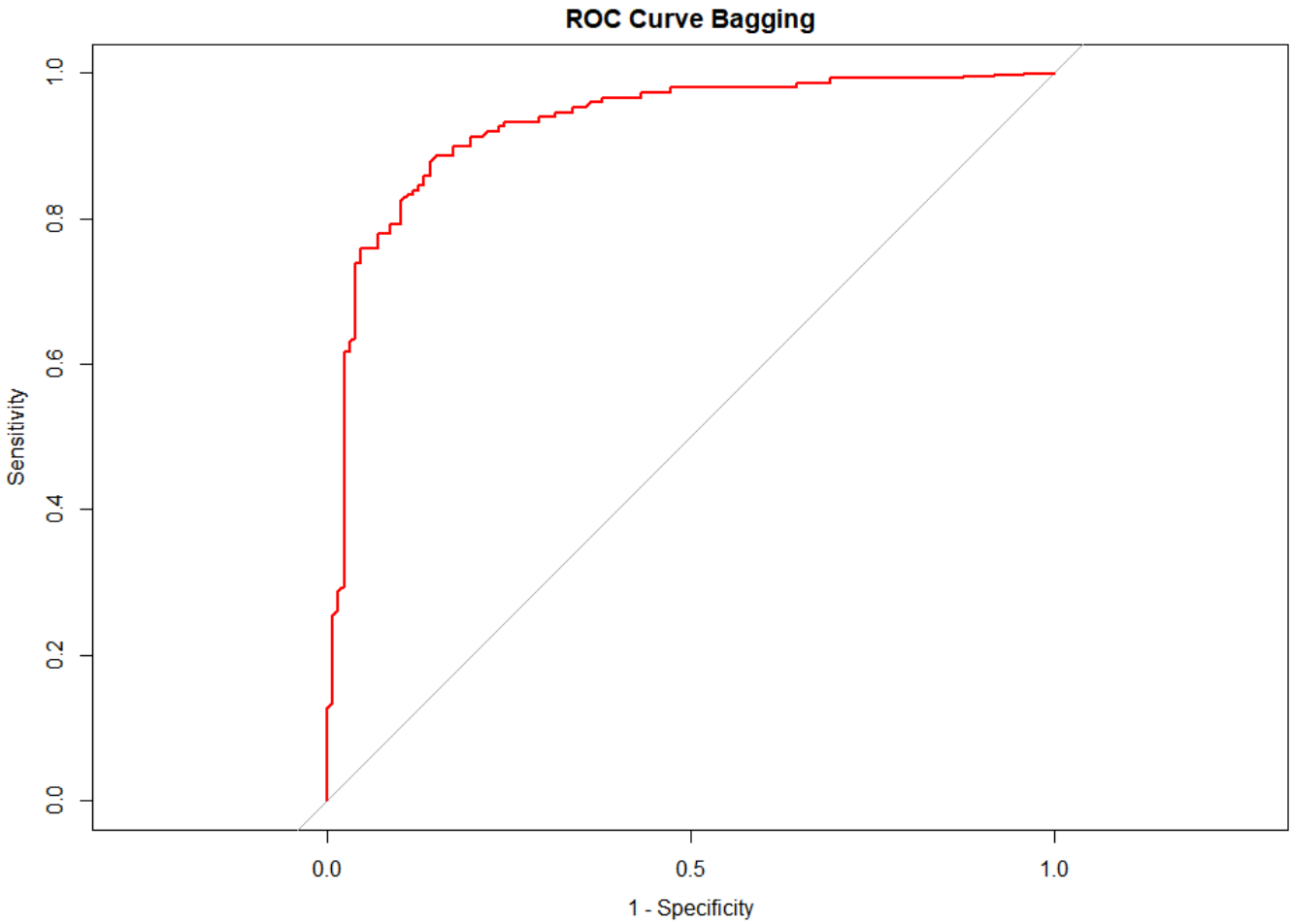
-▶ST_Slope 변수가 가장 중요한 영향을 미치고 있으며 이후 Cholesterol, Oldpeak, MaxHR 변수가 중요함을 알 수 있다.

 # ROC Curve

```
library(pROC)
curv <- roc(as.numeric(y_test), as.numeric(prob_bagg[,2]))

curv$auc
```

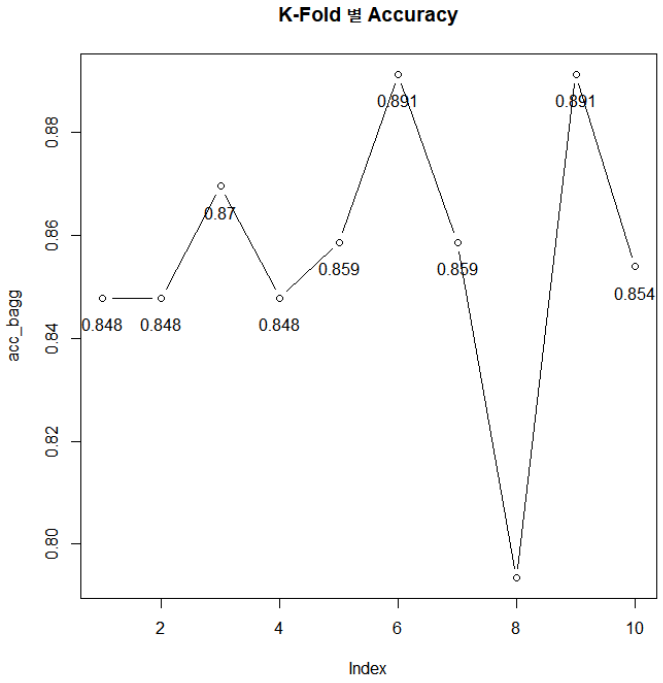
Area under the curve: 0.9291

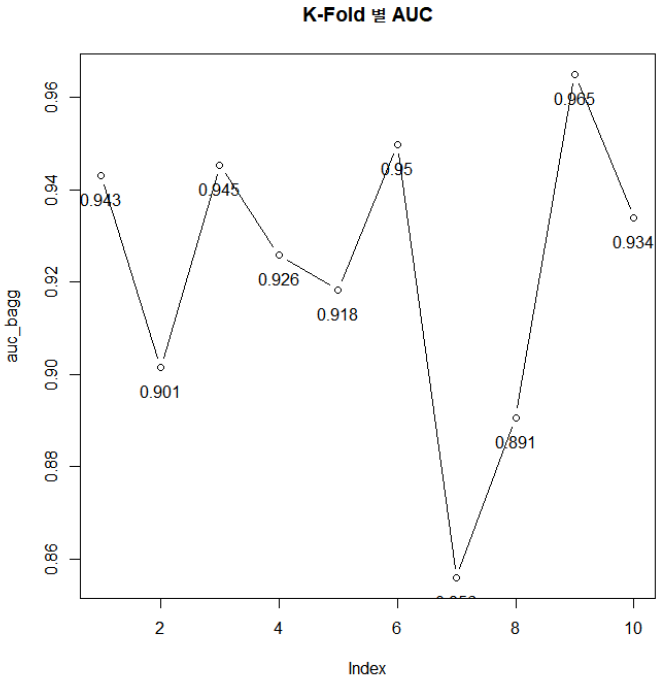
```
plot.roc(curv,legacy.axes = T,col ="red",main = "ROC Curve Bagging")
```




```
# K-Fold cv = 10
acc_bagg =auc_bagg = numeric()

for(k in 1:10){
  k_y_test <- heart[kfold[[k]], "HeartDisease"]

  bagg <- randomForest(HeartDisease ~., heart[-kfold[[k]], ], mtry = ncol(heart) - 1, ntree = 1000)

  pred_bagg <- predict(bagg, heart[kfold[[k]], ])
  prob_bagg <- predict(bagg, heart[kfold[[k]], ], type = "prob")

  # Confusion Matrix
  confu <- table(k_y_test, pred_bagg)

  # Accuracy
  acc_bagg[k] <- sum(confu[col(confu) == row(confu)]) / sum(confu)

  # AUC
  curv <- roc(as.numeric(k_y_test), as.numeric(prob_bagg[,2]))
  auc_bagg[k] <- curv$auc
```

```
}
# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_bagg, type = "b", main = "K-Fold 별 Accuracy")
text(acc_bagg - 0.005, labels = round(acc_bagg,3))

plot(auc_bagg, type = "b", main = "K-Fold 별 AUC")
text(auc_bagg - 0.005, labels = round(auc_bagg,3))
par(mfrow = c(1,1))
```






```
# 평균 정확도
mean(acc_bagg)
```

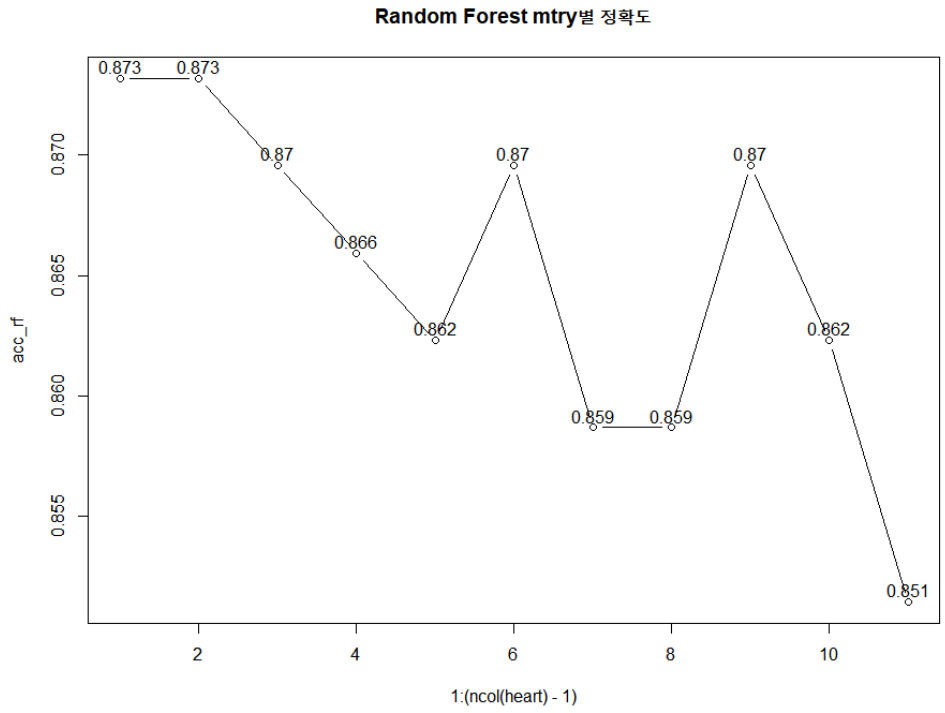
[1] 0.8560454

```
# 평균 AUC
mean(auc_bagg)
```

[1] 0.9229279

2.3 RandomForest Classification

```
# ------------------- (2) RandomForest -------------------------
library(randomForest)

# ntree는 조정하지 않고, mtry의 수에 따라서 정확도가 어떻게 변화하는지 확인.
acc_rf <- numeric()

set.seed(45)
for(i in 1:(ncol(heart)-1)) {
  rf <- randomForest(HeartDisease ~., heart, subset = train, mtry = i)

  pred_rf <- predict(rf, newdata = heart[test,])

  # Confusion Matrix
  confu <- table(y_test, pred_rf)

  # Accuracy
  acc_rf[i] <- sum(confu[col(confu) == row(confu)]) / sum(confu)
}

# mtry? 일반적으로 sqrt(12) = m으로 사용
plot(1:(ncol(heart)-1), acc_rf, type = "b", main = "Random Forest mtry별 정확도")
text(1:(ncol(heart)-1), acc_rf + 0.0005,label = round(acc_rf,3))
```



-▶ mtry가 3, 6일 때 높은 정확도를 보인다. 너무 적은 수의 변수는 좋지 않다고 판단하여. 6을 이용한다.

```
set.seed(45)
rf <- randomForest(HeartDisease ~., heart, subset = train, mtry = 6, ntree = 800)

# 예측
pred_rf <- predict(rf, newdata = heart[test,])
prob_rf <- predict(rf, newdata = heart[test,], type = "prob")

# Confusion Matrix
(confu <- table(y_test, pred_rf))
```

y_test	pred_rf 0	1
0	109	18
1	23	126

```
# Accuracy + sensitivity + specificity
(acc_rf <- sum(confu[col(confu) == row(confu)]) / sum(confu))
(sens_bagg <- confu[2,2] / sum(confu[,2]))
(spec_bagg <- confu[1,1] / sum(confu[,1]))
```

Accuracy = 0.8514493
Sensitivity = 0.875
Specificity = 0.8257576

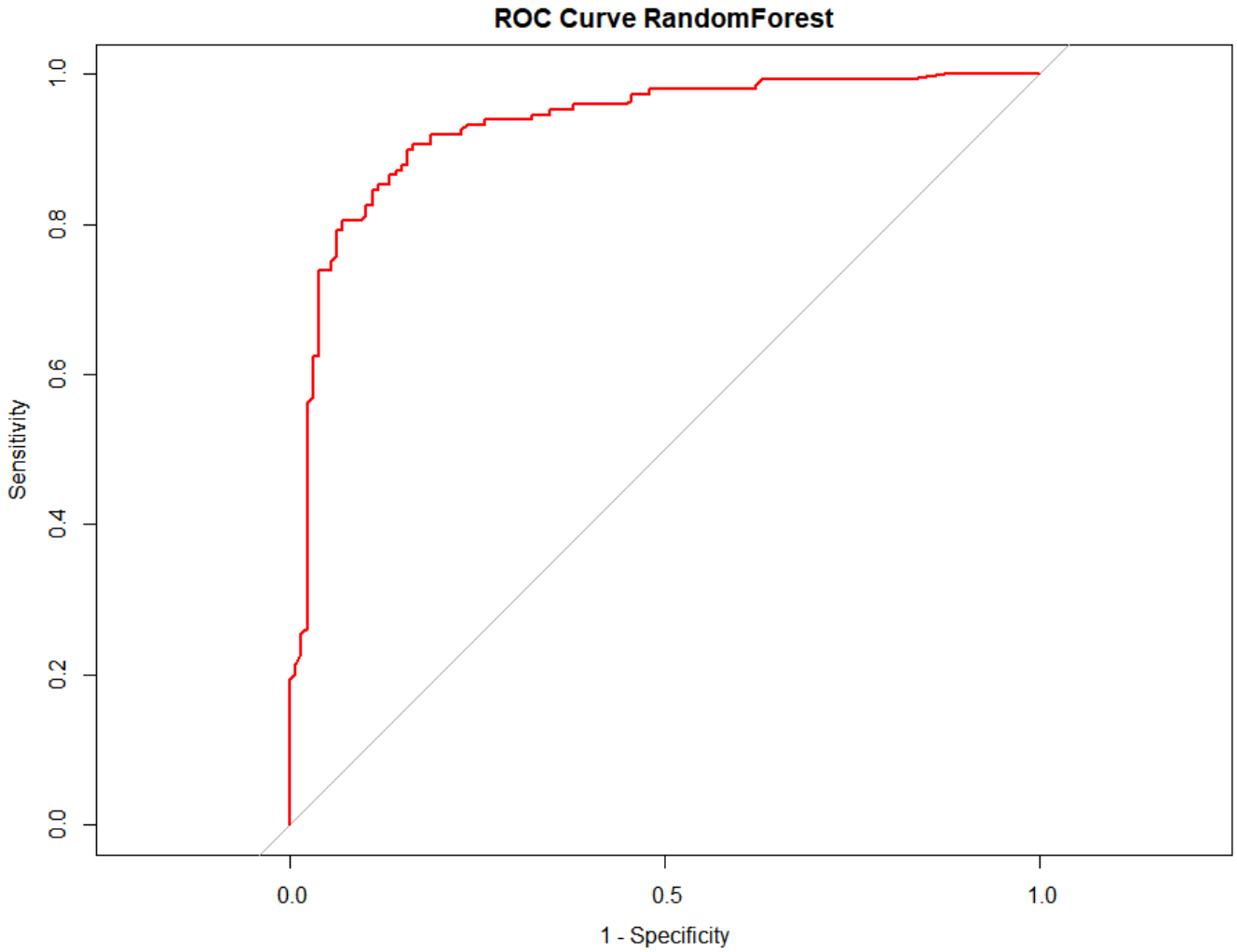
```
# 변수 중요도 확인
imp <- importance(rf)
barplot(sort(imp[,1]), horiz = T, las = 2, main = 'Variable Imporatance', cex.names = 0.7)
```

-▶Bagging에서 얻은 결과와 동일하다. (Bagging 중요도 Plot 확인)

```
# ROC Curve
library(pROC)
curv <- roc(as.numeric(y_test), as.numeric(prob_bagg[,2]))

curv$auc
```

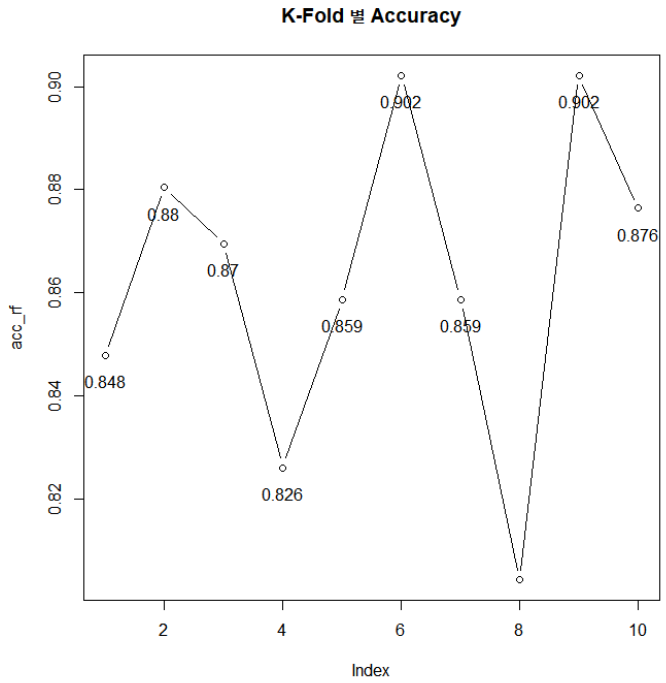
Area under the curve: 0.9306

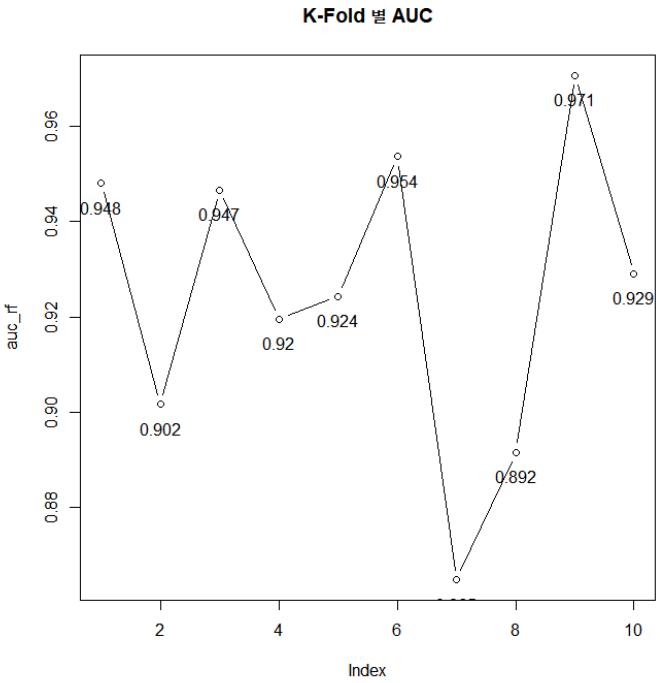
```
plot.roc(curv,legacy.axes = T,col ="red",main = "ROC Curve RandomForest")
```




```
# K-Fold cv = 10
acc_rf = auc_rf = numeric()

for(k in 1:10){
  k_y_test <- heart[kfold[[k]], "HeartDisease"]

  rf <- randomForest(HeartDisease ~., heart[-kfold[[k]], ], mtry = 5, ntree = 800)

  pred_rf <- predict(rf, heart[kfold[[k]], ])
  prob_rf <- predict(rf, newdata = heart[kfold[[k]],], type = "prob")

  # Confusion Matrix
  confu <- table(k_y_test, pred_rf)

  # Accuracy
  acc_rf[k] <- sum(confu[col(confu) == row(confu)]) / sum(confu)
```

```
  # AUC
  curv <- roc(as.numeric(k_y_test), as.numeric(prob_rf[,2]))
  auc_rf[k] <- curv$auc
}

# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_rf, type = "b", main = "K-Fold 별 Accuracy")
text(acc_rf - 0.005, labels = round(acc_rf,3))

plot(auc_rf, type = "b", main = "K-Fold 별 AUC")
text(auc_rf - 0.005, labels = round(auc_rf,3))
par(mfrow = c(1,1))
```






```
# 평균 정확도
mean(acc_bagg)
```

[1] 0.8637274

```
# 평균 AUC
mean(auc_bagg)
```

[1] 0.9266294

2.4 Logistic Regression

 # ---------- (3) Logistic Regression ----------

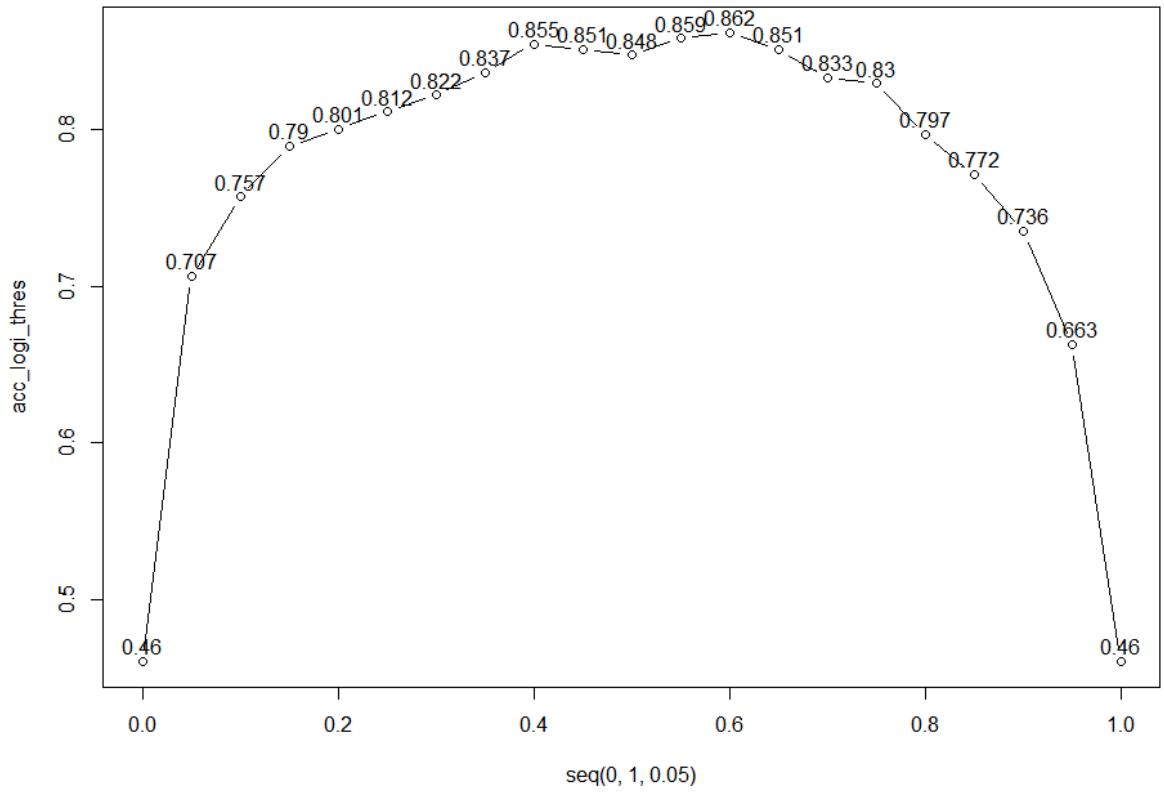
logi <- glm(HeartDisease ~., heart, subset = train, family = binomial) # 모든 변수 적합한 모델

summary(logi)

```
Call:
glm(formula = HeartDisease ~ ., family = binomial, data = heart,
    subset = train)

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        -1.448570   1.772844  -0.817 0.413878
Age                -0.002849   0.016296  -0.175 0.861221
SexM                1.244135   0.336735   3.695 0.000220 ***
ChestPainTypeATA   -2.274265   0.418767  -5.431 5.61e-08 ***
ChestPainTypeNAP   -1.494461   0.320316  -4.666 3.08e-06 ***
ChestPainTypeTA    -1.370512   0.519808  -2.637 0.008375 **
RestingBP           0.012740   0.008199   1.554 0.120223
Cholesterol        -0.004284   0.001278  -3.352 0.000801 ***
FastingBS           1.391605   0.341998   4.069 4.72e-05 ***
RestingECGNormal   -0.442960   0.344821  -1.285 0.198929
RestingECGST       -0.313705   0.432754  -0.725 0.468511
MaxHR              -0.002652   0.006049  -0.438 0.661095
ExerciseAnginaY     1.210475   0.303427   3.989 6.63e-05 ***
Oldpeak             0.368116   0.146905   2.506 0.012217 *
ST_SlopeFlat        1.695542   0.512543   3.308 0.000939 ***
ST_SlopeUp         -0.742814   0.531246  -1.398 0.162039
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 879.82  on 640  degrees of freedom
Residual deviance: 399.24  on 625  degrees of freedom
AIC: 431.24

Number of Fisher Scoring iterations: 6
```

-▶ Logistic 회귀계수의 유의성 검정에 따라 유의하지 않은 변수들을 제거하게 되면 정확도, AUC값이 떨어지는 경향이 있다. 모든 변수를 이용했을 때 가장 좋은 결론이 도출되어 변수를 제외하지 않는다.

```
# 높은 정확도를 갖는 Threshold를 확인해 본다.
acc_logi_thres <- numeric()
prob_logi <- predict(logi, newdata = heart[test,], type = "response")

for(k in seq(0,1,0.05)) {
  pred_logi <- ifelse(prob_logi > k,1,0)

  # Confusion Matrix
  confu <- table(y_test, pred_logi)

  # Accuracy
  acc_logi_thres <- c(acc_logi_thres, sum(confu[col(confu) == row(confu)]) / sum(confu))
}

plot(seq(0,1,0.05), acc_logi_thres, type = "b")
text(seq(0,1,0.05), acc_logi_thres + 0.01, round(acc_logi_thres,3))
```




```
# Threshold 0.6에서 좋은 결과가 도출되었다.
seq(0,1,0.05)[which.max(acc_logi_thres)]
```

[1] 0.6

```
# threshold 0.6를 이용한다.
pred_logi <- ifelse(prob_logi > 0.6,1,0)

# Confusion Matrix
(confu <- table(y_test, pred_logi))
```

```
        pred_logi
y_test   0   1
     0 109  18
     1  20 129
```

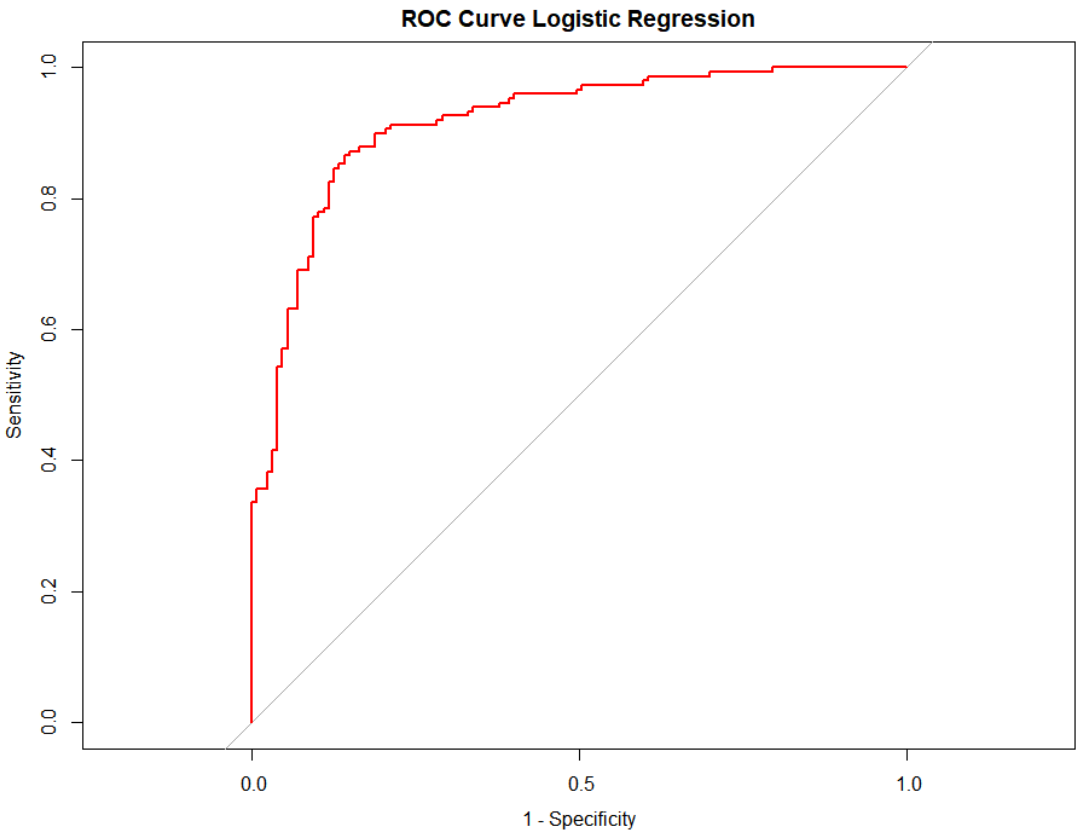
```
# Accuracy + sensitivity + specificity
(acc_logi <- sum(confu[col(confu) == row(confu)]) / sum(confu))
(sens_logi <- confu[2,2] / sum(confu[,2]))
(spec_logi <- confu[1,1] / sum(confu[,1]))
```

Accuracy = 0.8623188
Sensitivity = 0.877551
Specificity = 0.8449612

```
# ROC Curve
library(pROC)
curv <- roc(as.numeric(y_test), as.numeric(prob_logi))

curv$auc
```

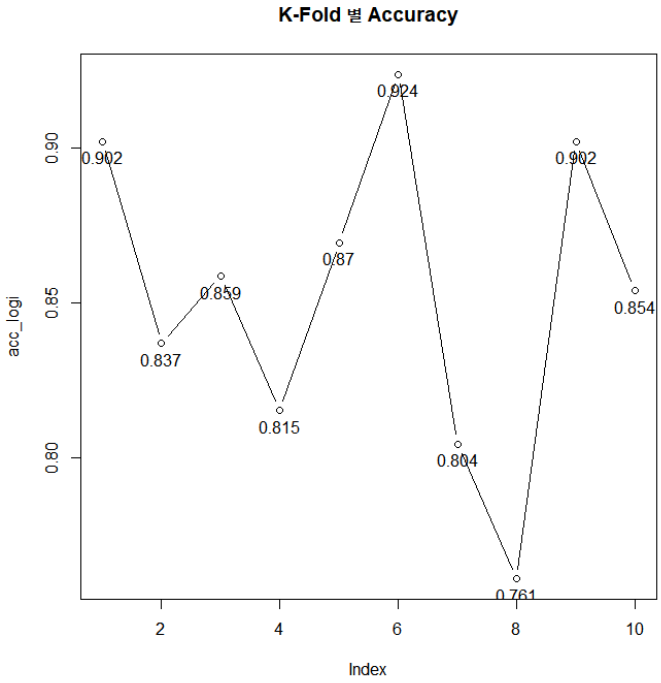
Area under the curve: 0.9155
-▶ 트리 기반 모델보다 성능이 약간 떨어지는 모습을 보여준다.

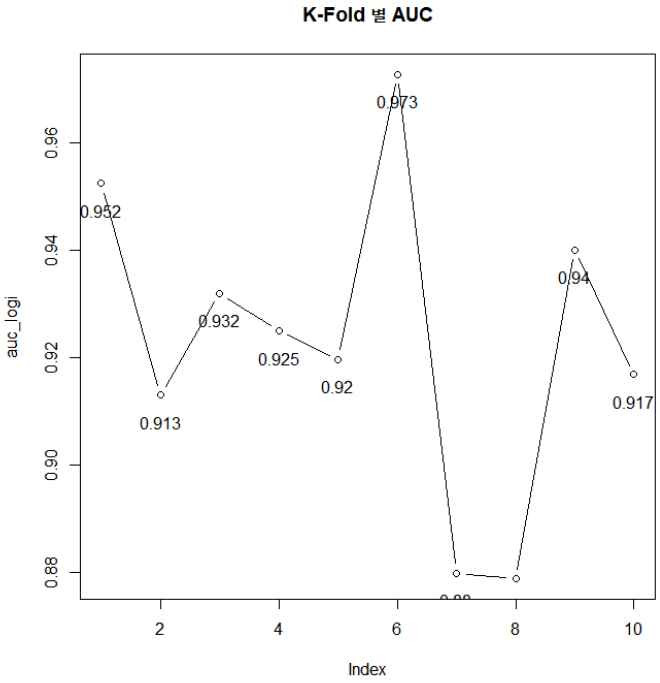
```
plot.roc(curv,legacy.axes = T,col ="red",main = "ROC Curve Logistic Regression")
```




```
# K-Fold cv = 10
acc_logi = auc_logi = numeric()

for(k in 1:10){
  k_y_test <- heart[kfold[[k]], "HeartDisease"]

  logi <- glm(HeartDisease ~., heart[-kfold[[k]],], subset = train, family = binomial)

  prob_logi <- predict(logi, heart[kfold[[k]], ])
  pred_logi <- ifelse(prob_logi > 0.6,1,0)

  # Confusion Matrix
  confu <- table(k_y_test, pred_logi)

  # Accuracy
  acc_logi[k] <- sum(confu[col(confu) == row(confu)]) / sum(confu)

  # AUC
  curv <- roc(as.numeric(k_y_test), as.numeric(prob_logi))
  auc_logi[k] <- curv$auc
}
```

```
# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_logi, type = "b", main = "K-Fold 별 Accuracy")
text(acc_logi - 0.005, labels = round(acc_logi,3))

plot(auc_logi, type = "b", main = "K-Fold 별 AUC")
text(auc_logi - 0.005, labels = round(auc_logi,3))
par(mfrow = c(1,1))
```






```
# 평균 정확도
mean(acc_logi)
```

[1] 0.8527846

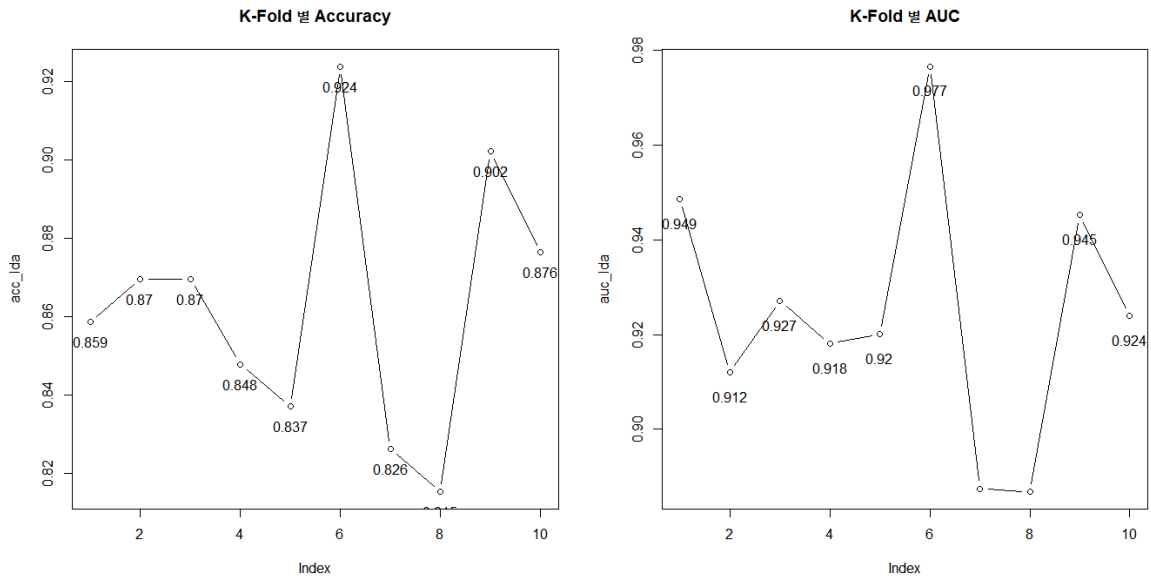
```
# 평균 AUC
mean(auc_logi)
```

[1] 0.9230032

3. Classification Model 적합 - 2

3.1 LDA Classification

```
## LDA 방법 적용
library(MASS)
library(pROC)
acc_lda <- numeric(k)
auc_lda <- numeric(k)

# K-fold cv
for (i in 1:k){
  lda.fit <- lda(HeartDisease~., data=heart[-kfold[[i]],])
  pred <- predict(lda.fit, newdata=heart[kfold[[i]],])
  real <- heart$HeartDisease[kfold[[i]]]
  confu <- table(real, pred$class)
  acc_lda[i] <- sum(confu[col(confu) == row(confu)])/ sum(confu)
  auc_lda[i] <- roc(as.numeric(real), as.numeric(pred$posterior[,2]))$auc
}

# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_lda, type = "b", main = "K-Fold 별 Accuracy")
text(acc_lda - 0.005, labels = round(acc_lda,3))

plot(auc_lda, type = "b", main = "K-Fold 별 AUC")
text(auc_lda - 0.005, labels = round(auc_lda,3))
par(mfrow = c(1,1))
```



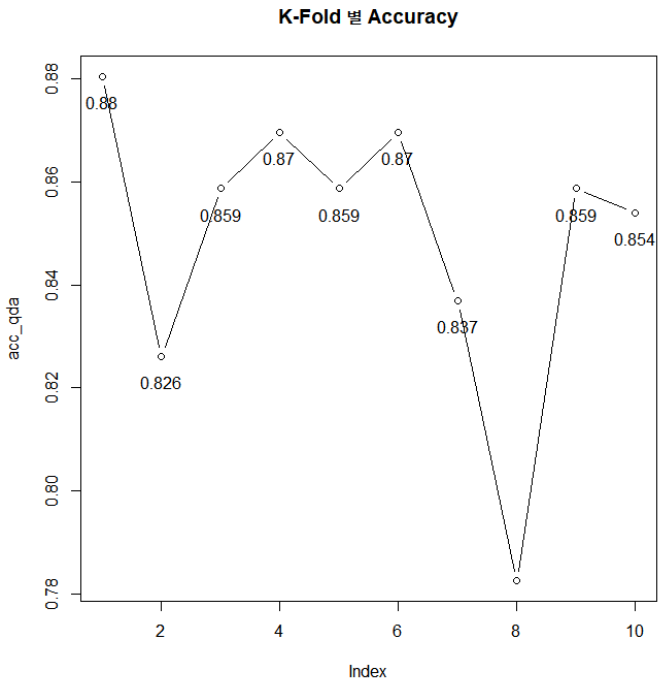
```
# 평균 정확도
mean(acc_lda)
```
[1] 0.8626404

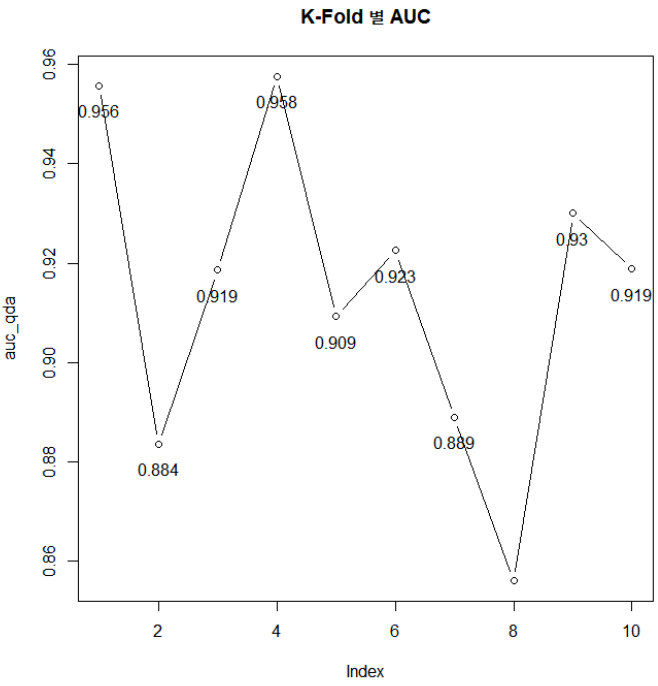
```
# 평균 AUC
mean(auc_lda)
```
[1] 0.9246064

-▶ LDA 모델을 적용시킨경우 모델 평균 정확도가 0.863, 모델 평균 AUC 점수가 0.925정도 나온다.

3.2 QDA Classification

```
## QDA 방법 적용
library(MASS)
library(pROC)
acc_qda <- numeric(k)
auc_qda <- numeric(k)

# K-fold cv
for (i in 1:k){
  qda.fit <- qda(HeartDisease~., data=heart[-kfold[[i]],])
  pred <- predict(qda.fit, newdata=heart[kfold[[i]],])
  real <- heart$HeartDisease[kfold[[i]]]
  confu <- table(real, pred$class)
  acc_qda[i] <- sum(confu[col(confu) == row(confu)])/ sum(confu)
  auc_qda[i] <- roc(as.numeric(real), as.numeric(pred$posterior[,2]))$auc
}

# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_qda, type = "b", main = "K-Fold 별 Accuracy")
text(acc_qda - 0.005, labels = round(acc_qda,3))

plot(auc_qda, type = "b", main = "K-Fold 별 AUC")
text(auc_qda - 0.005, labels = round(auc_qda,3))
par(mfrow = c(1,1))
```





 # 평균 정확도

```
mean(acc_qda)
```

[1] 0.8495237

 # 평균 AUC

```
mean(auc_qda)
```

[1] 0.9141044

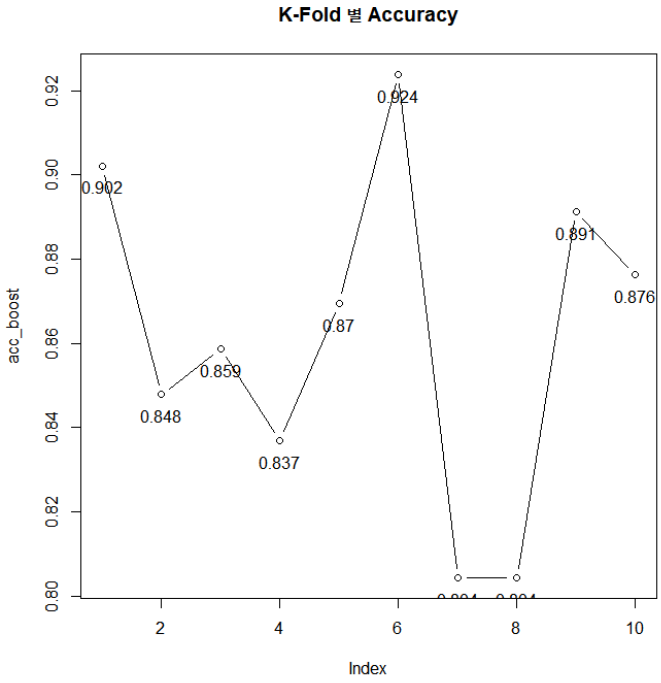
-▶ QDA 모델을 적용시킨 경우 모델 평균 정확도가 0.850, 모델 평균 AUC 점수가 0.914정도 나온다.

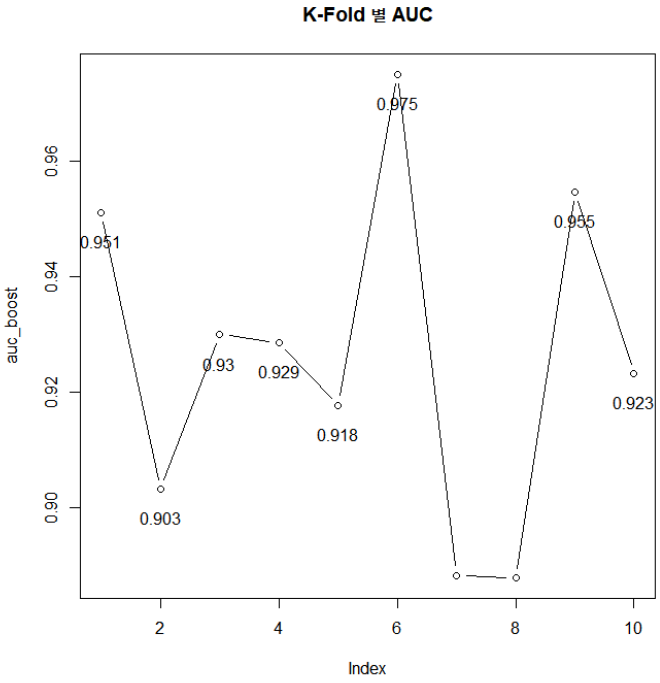
3.3 Boosting

Boosting 방법 적용

```
library(gbm)
set.seed(45)
acc_boost <- numeric(k)
auc_boost <- numeric(k)

# K-fold cv
for (i in 1:k){
  boost.fit <- gbm(HeartDisease~., data=heart[-kfold[[i]],],
                   distribution ="bernoulli")
  pred <- predict(boost.fit, newdata=heart[kfold[[i]],])
  predvalue <- ifelse(pred > 0.5, 1, 0)
  real <- heart$HeartDisease[kfold[[i]]]
```

```
  confu <- table(real, predvalue)
  acc_boost[i] <- sum(confu[col(confu) == row(confu)])/ sum(confu)
  auc_boost[i] <- roc(as.numeric(real), as.numeric(pred))$auc
}

# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_boost, type = "b", main = "K-Fold 별 Accuracy")
text(acc_boost - 0.005, labels = round(acc_boost,3))

plot(auc_boost, type = "b", main = "K-Fold 별 AUC")
text(auc_boost - 0.005, labels = round(auc_boost,3))
par(mfrow = c(1,1))
```






```
# 평균 정확도
mean(acc_boost)
```

[1] 0.8615535

```
# 평균 AUC
mean(auc_boost)
```

[1] 0.9259812

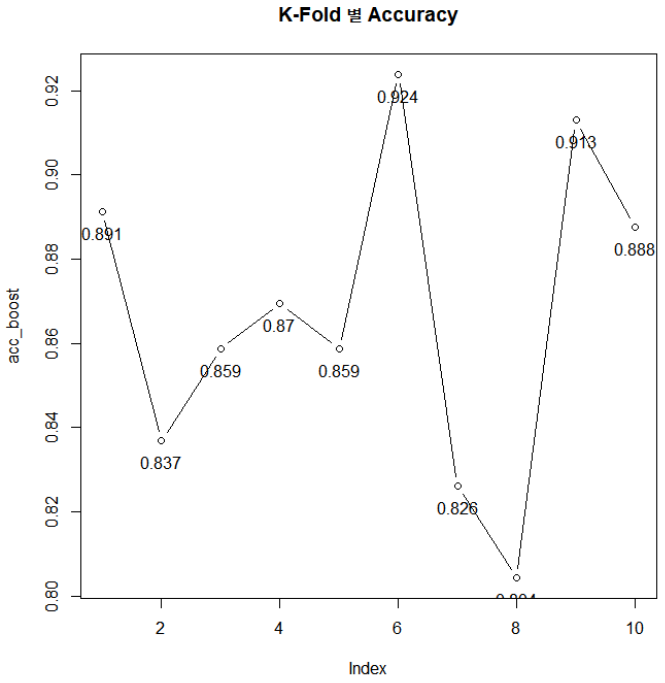
-▶ Boosting 기법을 적용시킨 경우 평균 정확도가 0.862, 모델 평균 AUC 점수가 0.926정도 나온다.

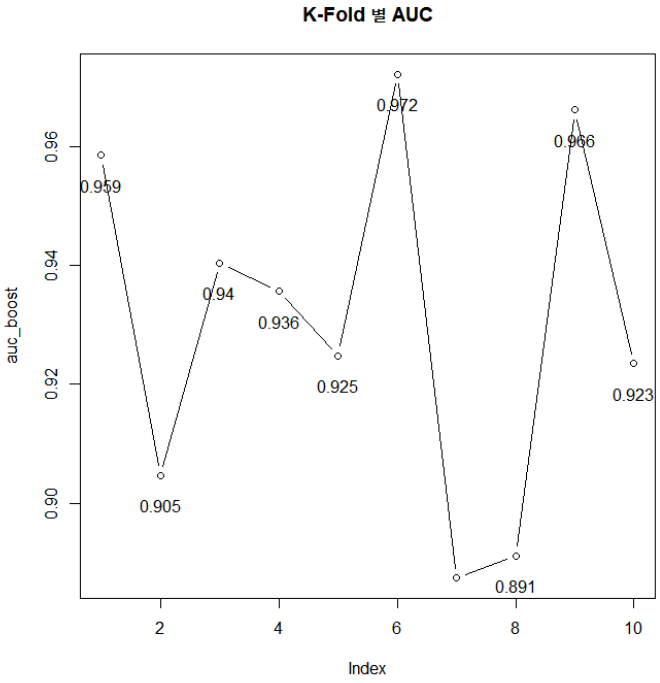
```
## 부스팅 방법 적용(+ 하이퍼 파라미터 interaction.depth 조정)
boostfun <- function(depth = 5){
  set.seed(45)
  auc_boost <- numeric(k)
  for (i in 1:k){
    boost.fit <- gbm(HeartDisease~., data=heart[-kfold[[i]],],
                     distribution ="bernoulli",
                     interaction.depth = depth)
    pred <- predict(boost.fit, newdata=heart[kfold[[i]],])
    predvalue <- ifelse(pred > 0.5, 1, 0)
    real <- heart$HeartDisease[kfold[[i]]]
    auc_boost[i] <- roc(as.numeric(real), as.numeric(pred))$auc
  }
  return(mean(auc_boost))
}

sapply(c(2,3,5,7,10),function(x) boostfun(depth = x))
[1] 0.9304616 0.9293532 0.9250370 0.9255766 0.9248607
```

-▶ interaction.depth = 2에서 최적의 AUC 점수를 보인다.

```
## 부스팅 방법 최종 적용
set.seed(45)
acc_boost <- numeric(k)
auc_boost <- numeric(k)

for (i in 1:k){
  boost.fit <- gbm(HeartDisease~., data=heart[-kfold[[i]],],
                   distribution ="bernoulli",
                   interaction.depth = 2)
  pred <- predict(boost.fit, newdata=heart[kfold[[i]],])
  predvalue <- ifelse(pred > 0.5, 1, 0)
  real <- heart$HeartDisease[kfold[[i]]]
  confu <- table(real, predvalue)
  acc_boost[i] <- sum(confu[col(confu) == row(confu)])/ sum(confu)
  auc_boost[i] <- roc(as.numeric(real), as.numeric(pred))$auc
}

# K-fold cv 평균 정확도 / 평균 AUC값
par(mfrow = c(1,2))
plot(acc_boost, type = "b", main = "K-Fold 별 Accuracy")
```

```
text(acc_boost - 0.005, labels = round(acc_boost,3))

plot(auc_boost, type = "b", main = "K-Fold 별 AUC")
text(auc_boost - 0.005, labels = round(auc_boost,3))
par(mfrow = c(1,1))
```





 # 평균 정확도

```
mean(acc_boost)
```

[1] 0.8670249

 # 평균 AUC

```
mean(auc_boost)
```

[1] 0.9304616

-▶ 최종 Boosting 모델 평균 정확도가 0.867, 모델 평균 AUC 점수가 0.930정도 나온다.

4. 결과 및 역할 ----------

4.1 AUC 값 기준 가장 예측력이 잘 나오는 모델

10 K-fold CV로 데이터를 분류하고 여러 모델을 적합시켜 보았다. 모델 별 AUC 값을 비교해보겠다.
Bagging 0.923, Random Forest 0.927, Logistic Regression 0.923, LDA 0.925, QDA 0.914, Boosting 0.930 AUC 값 기준으로 부스팅 모델이 가장 예측력이 잘 나오는 모델이라고 할 수 있겠다.

4.2 조원별 역할

김성연 :
1.2 RestingBP, Cholesterol 변수의 심층적 탐색
2.1 Dataset 분리 / K-Fold Dataset 분리
3.1 LDA Classification
3.2 QDA Classification
3.3 Boosting

손형락 :
1.1 Dataset 탐색
2.2 Bagging Classification
2.3 RandomForest Classification
2.4 Logistic Regression Classification
프로젝트 워드 파일 정리 및 작성