Homework 3

21500131 김은총

**Single Variable Model for Regression**

**1. bikeJuly data frame을 사용하여 목적 변수(cnt)를 가장 잘 예측하는 (단일 변수 모델)**

**single variable model(Best Model)을 찾으시오(RMSE 기준). 가장 성능이 좋은 모델을 찾아가는 과정을 보고서에 함께 기술하시오.**

**1) atemp.range**

# range of atemp

bikesJuly$atemp.range <- ifelse(bikesJuly$atemp <= 0.6, "step1",

ifelse(bikesJuly$atemp<=0.7, "step2",

ifelse(bikesJuly$atemp<=0.8, "step3",

ifelse(bikesJuly$atemp <= 0.9, "step4","step5"))))

sv\_reg\_atemp.range <- tapply(bikesJuly$cnt, bikesJuly$atemp.range, mean)

sv\_reg\_atemp.range # regression model

bikesJuly$pred\_atemp.range <- sv\_reg\_atemp.range[bikesJuly$atemp.range] # train model

# ereor = real - pred

bikesJuly$error <- bikesJuly$atemp - bikesJuly$pred\_atemp.range

#MSE, RMSE

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 79402.99

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 281.7854**

**2) weathersit**

sv\_reg\_weathersit <- tapply(bikesJuly$cnt, bikesJuly$weathersit, mean)

sv\_reg\_weathersit # regression model

bikesJuly$pred\_weathersit <- sv\_reg\_weathersit[bikesJuly$weathersit] # train model

# ereor = real – pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_weathersit

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 45457.63

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 213.2079**

**3) hum**

bikesJuly$hum.range <- ifelse(bikesJuly$hum <= 0.6, "step1",

ifelse(bikesJuly$hum<=0.65, "step2",

ifelse(bikesJuly$hum<=0.7, "step3",

ifelse(bikesJuly$hum <= 0.75, "step4",

ifelse(bikesJuly$hum <= 0.8,"step5",

ifelse(bikesJuly$hum <= 0.85,"step6","step7"))))))

sv\_reg\_hum.range <- tapply(bikesJuly$cnt, bikesJuly$hum.range, mean)

sv\_reg\_hum.range # regression model

## train model

bikesJuly$pred\_hum.range <- sv\_reg\_hum.range[bikesJuly$hum.range]

# ereor = real - pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_hum.range

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 38025.8

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 195.002**

**4) hr**

bikesJuly$hr <- as.numeric(bikesJuly$hr)

bikesJuly$hr.range <- ifelse(bikesJuly$hr ==1 , "dawn",

ifelse(bikesJuly$hr <= 6, "dawn",

ifelse(bikesJuly$hr <= 10, "ealy morning",

ifelse(bikesJuly$hr <= 11, "late in the morning",

ifelse(bikesJuly$hr <= 15, "ealry afternoon",

ifelse(bikesJuly$hr <= 17, "afternoon",

ifelse(bikesJuly$hr <= 22, "night", "deep in the night" )))))))

sv\_reg\_hr.range <- tapply(bikesJuly$cnt, bikesJuly$hr.range, mean)

sv\_reg\_hr.range # regression model## train model

bikesJuly$pred\_hr.range <- sv\_reg\_hr.range[bikesJuly$hr.range]

# ereor = real - pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_hr.range

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 20706.14

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 143.8963**

**5) holiday**

sv\_reg\_holiday <- tapply(bikesJuly$cnt, bikesJuly$holiday, mean)

sv\_reg\_holiday # regression model

## train model

bikesJuly$pred\_holiday <- ifelse(bikesJuly$holiday=="FALSE",272.5056, 308.4583 )

# ereor = real - pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_holiday

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 45761.85

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 213.9202**

**6) windspeed**

bikesJuly$windspeed.range <- ifelse(bikesJuly$windspeed <= 0.1, "step1",

ifelse(bikesJuly$windspeed<=0.2, "step2",

ifelse(bikesJuly$windspeed<=0.3, "step3",

ifelse(bikesJuly$windspeed <= 0.4, "step4",

ifelse(bikesJuly$windspeed <= 0.5, "step5",

ifelse(bikesJuly$windspeed <= 0.6, "step6", "step7"))))))

sv\_reg\_windspeed.range <- tapply(bikesJuly$cnt, bikesJuly$windspeed.range, mean)

sv\_reg\_windspeed.range # regression model

## train model

bikesJuly$pred\_windspeed.range <- sv\_reg\_windspeed.range[bikesJuly$windspeed.range]

# ereor = real – pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_windspeed.range

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 43085.7

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 207.5709**

**7) temp**

bikesJuly$temp.range <- ifelse(bikesJuly$temp <= 0.6, "step1",

ifelse(bikesJuly$temp<=0.65, "step2",

ifelse(bikesJuly$temp<=0.7, "step3",

ifelse(bikesJuly$temp <= 0.75, "step4",

ifelse(bikesJuly$temp <= 0.8,"step5",

ifelse(bikesJuly$temp <= 0.85,"step6",

ifelse(bikesJuly$temp <= 0.9,"step7",

ifelse(bikesJuly$temp <= 1,"step8"))))))))

sv\_reg\_temp.range <- tapply(bikesJuly$cnt, bikesJuly$temp.range, mean)

sv\_reg\_temp.range # regression model

## train model

bikesJuly$pred\_temp.range <- sv\_reg\_temp.range[bikesJuly$temp.range]

# ereor = real - pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_temp.range

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 36598.32

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 191.3069**

**7) workingday**

sv\_reg\_workingday <- tapply(bikesJuly$cnt, bikesJuly$workingday, mean)

sv\_reg\_workingday # regression model

## train model

bikesJuly$pred\_workingday <- ifelse(bikesJuly$workingday=="FALSE",252.4875 ,283.7500 )

# ereor = real - pred

bikesJuly$error <- bikesJuly$cnt - bikesJuly$pred\_workingday

#제곱한 에러의 평균 = MSE : 작을수록 좋음

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 45588.63

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 213.5149**

답 : hr 변수의 RMSE 값이 143.8963으로 가장 작다.

**2. 1번에서 찾은 모델의 RMSE와 R square 값을 계산하시오.**

MSE\_train <- mean(bikesJuly$error\*\*2)

MSE\_train # 20706.14

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 143.8963**

RSS = sum(bikesJuly$error \*\* 2)

SStot = sum((bikesJuly$cnt - mean(bikesJuly$cnt))\*\*2)

**1 - (RSS/SStot) # 0.5479225**

답 : 1) RMSE : `143.8963

2) R2 = 0.5479225

**3. 1번에서 찾은 모델을 bikesAugust data frame에 대해서 테스트하시오.**

**테스트 RMSE와 R square의 값은 어떻게 계산되나요?**

bikesAugust$hr <- as.numeric(bikesAugust$hr)

bikesAugust$hr.range <- ifelse(bikesAugust$hr ==1 , "dawn",

ifelse(bikesAugust$hr <= 6, "dawn",

ifelse(bikesAugust$hr <= 10, "ealy morning",

ifelse(bikesAugust$hr <= 11, "late in the morning",

ifelse(bikesAugust$hr <= 15, "ealry afternoon",

ifelse(bikesAugust$hr <= 17, "afternoon",

ifelse(bikesJuly$hr <= 22, "night", "deep in the night" )))))))

bikesAugust$pred\_hr.range <- sv\_reg\_hr.range[bikesAugust$hr.range]

#RMSE on test

RMSE\_test <- sqrt(mean((bikesAugust$cnt - bikesAugust$pred\_hr.range) \*\* 2))

**RMSE\_test # 150.6723**

sd(bikesAugust$cnt) #227.875

bikesAugust$error <- bikesAugust$cnt - bikesAugust$pred\_hr.range

RSS = sum(bikesAugust$error \*\* 2)

SStot = sum((bikesAugust$cnt - mean(bikesAugust$cnt))\*\*2)

**1 - (RSS/SStot) # 0.5622186**

답 : 1) RMSE : 150.6723

2) R2 : 0.5622186

**4. 찾은 모델은 overfitting(과적합)이라고 할 수 있나요? 할 수 있거나 혹은 할 수 없거나 왜**

**그런지 이유를 설명하시오.**

답 : RMSE는 train data 보다 test data 가 좀 더 크지만, test data의 R2 은 0.5622186이고, train data의 R2은 0.5479225 이므로 test data에서 R2이 높기 때문에 과적합이 아니라는 판단이 든다.

**5. 여러분이 찾은 모델의 성능을 Test 데이터 기준으로 다른 동료와 비교하시오. 여러분의**

**모델이 동료의 모델의 성능보다 낮다면 왜 그런지 생각해보고, 여러분의 모델을 개선하시**

**오. 비교 과정/결과와 개선 방법/개선 결과를 보고서에 기술하시오.**

sv\_reg\_hr <- tapply(bikesJuly2$cnt, bikesJuly2$hr, mean)

sv\_reg\_hr # regression model

bikesJuly2 <- bikesJuly[-13]

bikesJuly2 <- bikesJuly2[-13]

### train data ###

str(bikesJuly)

bikesJuly2$pred\_hr<- sv\_reg\_hr[bikesJuly2$hr]

bikesJuly2$error <- bikesJuly2$cnt - bikesJuly2$pred\_hr # error

MSE\_train <- mean(bikesJuly2$error\*\*2)

MSE\_train # 13244.28

RMSE\_train <- sqrt(MSE\_train)

**RMSE\_train # 115.0838**

RSS = sum(bikesJuly$error \*\* 2)

SStot = sum((bikesJuly$cnt - mean(bikesJuly$cnt))\*\*2)

**1 - (RSS/SStot) # 0.7108375**

### test data ###

bikesAugust$pred\_hr <- sv\_reg\_hr[bikesAugust$hr]

#RMSE on test

RMSE\_test <- sqrt(mean((bikesAugust$cnt - bikesAugust$pred\_hr) \*\* 2))

**RMSE\_test # 113.3978**

sd(bikesAugust$cnt) # 227.875

bikesAugust$error <- bikesAugust$cnt - bikesAugust$pred\_hr

RSS = sum(bikesAugust$error \*\* 2)

SStot = sum((bikesAugust$cnt - mean(bikesAugust$cnt))\*\*2)

**1 - (RSS/SStot) # 0.7520293**

답 : 나는 hr를 구간으로 묶은 뒤 분석을 했고, 다른 조원들은 0~24시 각각을 하나의 factor로 취급하여 구간을 나누지 않고 분석을 했다.

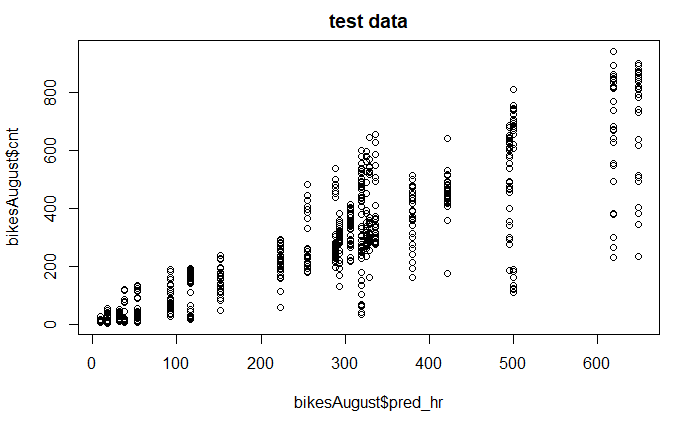
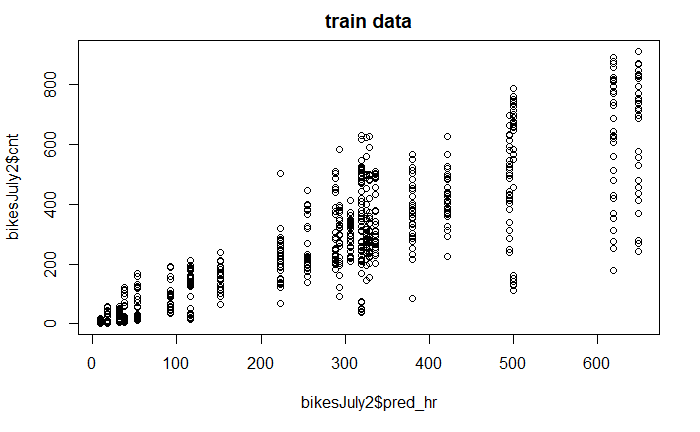
|  |  |  |
| --- | --- | --- |
| Test data | 구간 나눔 | 구간 없음 |
| RMSE | 150.6723 | 113.3978 |
| R2 | 0.5622186 | 0.7520293 |

비교 결과 구간으로 나눠서 시간을 묶은 뒤 분석을 하는 것 보다 0~24시 각각을 구간처럼 생각해서 분석을 하는 것이 성능이 더 좋다. 시간을 큰 구간으로 나눠서 분석을 하는 것 보다 세밀하게 나눠서 모델을 만들고 예측하는 것이 성능이 더 좋다. 단일 변수 모델이기 때문에 과적합이 발생하는 가능성도 적기 때문에 구간을 나누는 것도 크게 문제가 되지 않는 것 같다.

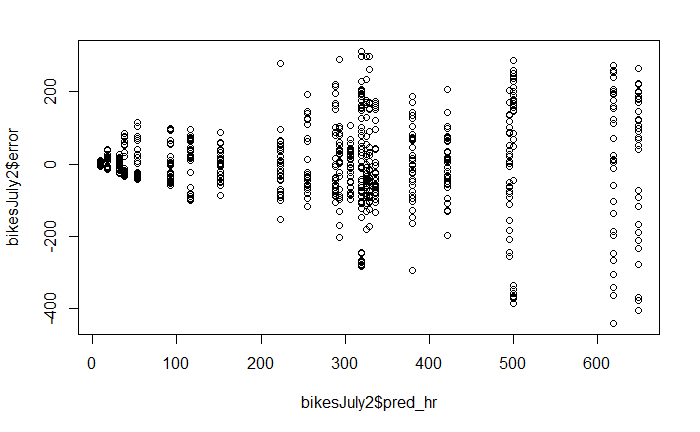
**6. 학습 데이터와 테스트 데이터에서, 각각 y축에 목적변수, x축에 목적변수에 대한 모델의**

**예측 값을 그린 산점도(scatter plot)을 그리시오. 그래프로부터 알 수 있는 것이 있다면**

**설명해 보시오**



답 : train data, test data 모두 값이 커질수록 흩어진 정도가 큰 경향을 보인다.



예측값과 에러 사이의 그래프를 추가로 그려본 결과 y 축의 0을 기준으로 x축의 값이 커질수록 에러가 증가하는 경향이 있다는 것을 확인할 수 있다.

**Single Variable Model for Classification**

**1. 학습데이터(train data frame)를 사용하여 목적 변수를 가장 잘 예측하는 (단일 변수 모델)**

**single variable model을 찾으시오 (AUC 기준). 목적변수가 범주형이므로 classification**

**model이며 단일 변수 모델이므로 하나의 입력 변수만을 모델에 사용해야 합니다. 가장**

**성능이 좋은 모델을 찾아가는 과정을 보고서에 함께 기술하시오.**

**어떤 변수를 사용하였나요? AUC는 얼마가 나오나요?**

**(코드 예시)**

**Relationship (Best model)**

# Building a Single Vatiable Model\_workclass

tble\_relationship <- table(train$relationship, train$income)

table(train$relationship)

tble\_relationship

prop.table(tble\_relationship , margin = 1)

sv\_model\_relationship <- prop.table(tble\_relationship, margin = 1)[,2]

# Prediction on Training Dataset

train$est\_prob\_relationship <- sv\_model\_relationship[train$relationship]

threshold <- 0.45

train$prediction\_relationship <- train$est\_prob\_relationship > threshold

# Accuracy : 맞춘 갯수 / total observation

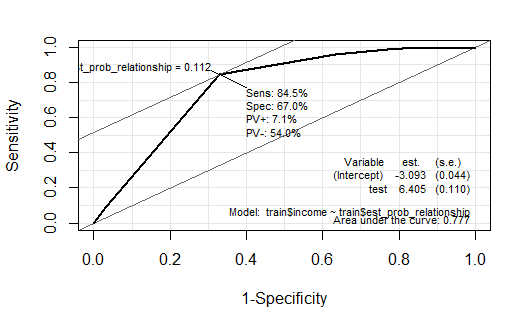
conf.train.table\_relationship <- table( actual = train$income,pred = train$prediction\_relationship)

(conf.train.table\_relationship[1,1]+conf.train.table\_relationship[2,2]) / sum(conf.train.table\_relationship) # 71.36%

# making Plot

library(ROCR)

ROC(train$est\_prob\_relationship, train$income) **# 0.777**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable | workclass | capital gain | race | relationship | marital.status |
| AUC | 0.57 | 0.570 | 0.539 | 0.777 | 0.766 |
| Note. |  | ~ mean  : low  Mean~ [(mean + max)/2]  : middle  ~ max  : high |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| hours.per.week | occupation | log10(capital.gain) | log10(capital.gain), yes or no |
| 0.630 | 0.725 | 0.583 | 0.583 |
| Min ~ 1st.Qu  : short  1st.Qu. ~ 1st Qu. + [(max-1st Qu.)/2]  : middle  1st Qu. + [(max-1st Qu.)/2] ~ max  : long |  | -Inf  : none  ~ 1st Qu.  : low  ~ 3rd Qu.  : middle  ~max  : high | -Inf : none  Other : have gain |

답 : relationship 일 때 AUC가 0.777로 가장 높다.

**2. 1번에서 찾은 모델을 test data frame에 대해서 test해서 AUC를 계산하시오.**

# Prediction on Test Data : overfitting

test$est\_prob\_relationship <- sv\_model\_relationship[test$relationship]

test$prediction\_relationship <- test$est\_prob\_relationship > 0.45

conf.test.table\_relationship <- table(actual = test$income,pred = test$prediction\_relationship)

# get\_arruracy

print(paste("arruracy on test set",

get\_acuuracy(test$prediction\_relationship, test$income))) **# 71.30%**

# get\_precision

print(paste("precision on test set",

get\_precision(test$prediction\_relationship, test$income))) **# 45.8%**

## get\_recall

print(paste("recall on test set",

get\_recall(test$prediction\_relationship, test$income))) **# 85.8%**

# making Plot

library(Epi)

ROC(test$est\_prob\_relationship, test$income)

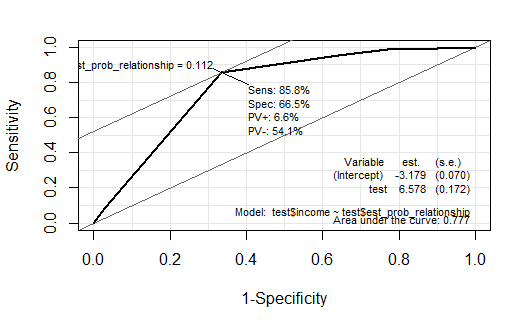
pred <- prediction(test$est\_prob\_relationship, test$income)

# cutoff 값에 따른 Accuracy의 변화

plot(performance(pred, "acc", "cutoff"))

# AUC

**performance(pred, "auc") # 0.777**



답 : Test AUC는 0.7770221 이다.

Slot "y.values":

[[1]]

[1] 0.7770221

**3. 찾은 모델은 overfitting(과적합)이라고 할 수 있나요? 할 수 있거나 혹은 할 수 없거나 왜**

**그런지 이유를 설명하시오.**

답 : 과적합이라고 할 수 없다. Train과 test 모두 AUC가 약 0.777 이므로 train과 test에서 성능이 비슷하기 때문이다.

**4. Best model에서 Threshold를 변화시킬 때, precision과 recall 값이 어떻게 변화하는지 그**

**래프로 표현하시오. 그래프를 바탕으로 어떤 threshold을 선택하는 것이 좋을지 여러분의**

**의견을 이유와 함께 제시하시오. (학습 데이터 테스트 데이터 각각에 대해)**

**5. Trade-off 관계에 있는 precision과 recall을 하나의 measure로 보기 위해서 F1 Measure라**

**는 것을 사용하기도 합니다. F1 Measure를 계산하는 수식은 아래와 같습니다. F1 값이 가장**

**크게 되는 threshold는 얼마인가요? (학습 데이터 테스트 데이터 각각에 대해)**

threshold <- seq(0.05,0.48,0.001)

# train dataset

graph\_precision\_train <- function(threshold){

train$prediction <-train$est\_prob\_relationship >threshold

a <-get\_precision(train$prediction,train$income)

return(a)

}

graph\_recall\_train <- function(threshold){

train$prediction <-train$est\_prob\_relationship >threshold

a <-get\_recall(train$prediction,train$income)

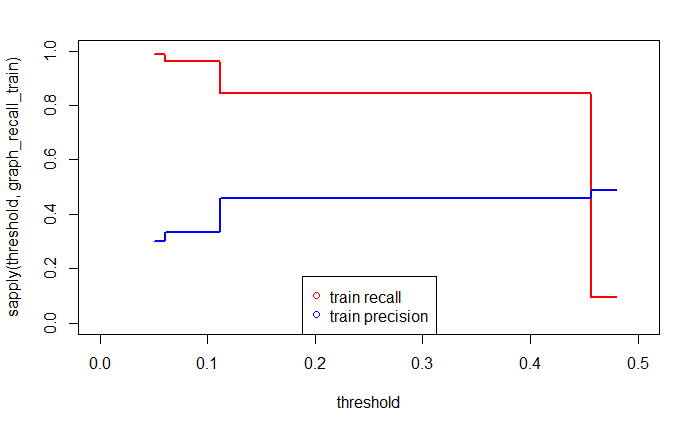
return(a)

}

plot(threshold, sapply(threshold, graph\_recall\_train) , type = "l", col="red", xlim=c(0,0.5), ylim=c(0,1),lwd="2")

lines(threshold, sapply(threshold,graph\_precision\_train), type = "l", col="blue", xlim=c(0,0.5), ylim=c(0,1),lwd="2")

legend("bottom", c("train recall", "train precision"), col=c("red", "blue"), pch = c(1,1))



# test dataset

graph\_precision <- function(threshold){

test$prediction <-test$est\_prob\_rel >threshold

a <-get\_precision(test$prediction,test$income)

return(a)

}

graph\_recall <- function(threshold){

test$prediction <-test$est\_prob\_rel >threshold

a <-get\_recall(test$prediction,test$income)

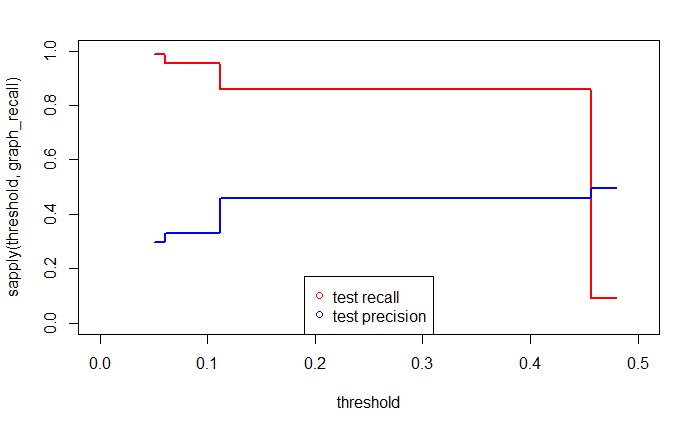
return(a)

}

plot(threshold, sapply(threshold, graph\_recall) , type = "l", col="red", xlim=c(0,0.5), ylim=c(0,1),lwd="2")

lines(threshold, sapply(threshold,graph\_precision), type = "l", col="blue", xlim=c(0,0.5), ylim=c(0,1),lwd="2")

legend("bottom", c("test recall", "test precision"), col=c("red", "blue"), pch = c(1,1))

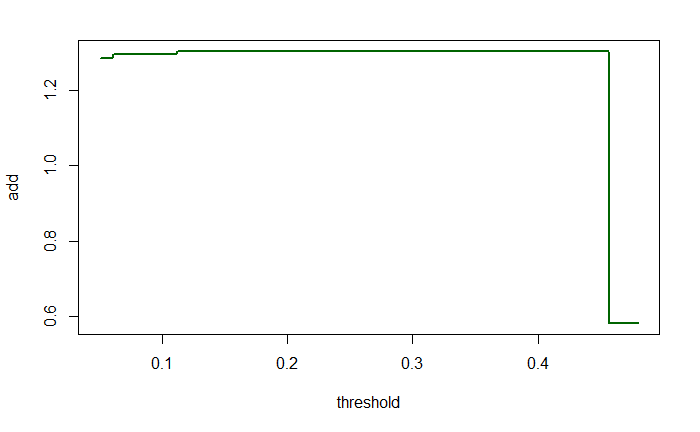


threshold가 0.49부터는 train data가 모두 소득이 50k 이하라고 판단하기 때문에 그래프를 0.48까지 그릴 수 있었다.

만약 소득이 50만불 이상인 사람에게 ‘고소득자에게 적합한 신용카드 가입 권유 메일’을 발송하여 가입율과 회사의 이익을 높이는 데에 분석 목적이 갖는다고 가정해보자.

"실제로 고소득자인데 저소득자로 판단 한 사건" 과 "실제는 저소득인데 고소득으로 판단 한 사건" 의 기회비용을 따지면 "실제로 고소득자인데 저소득자로 판단하여 메일을 보내지 않는다" 의 결과가 더 손실이 클 것이다.

따라서 recall값이 precision보다 더 중요하다고 판단된다. 그러므로 threshold가 약 0.46 이상인 곳 부터는 recall 값이 급격히 감소하므로 가정한 상황에서는 0.41~0.45 정도의 threshold 값을 가지는 것이 적절하다는 판단이 든다.



train\_th <- sapply(threshold,graph\_recall\_train)

train\_precision <- sapply(threshold,graph\_precision\_train)

add <- train\_th + train\_precision

plot(threshold, add, type="l", col = "dark green",lwd="2")

또한 precision과 recall값을 더한 add라는 변수를 만들어서 그래프를 그려본 결과 마찬가지로 threshold가 약 0.13 부0.46 부근까지 가장 높다. 따라서 이 구간 내에서 precision과 recall값의 비율을 잘 선택하는 것이 옳다는 판단이 든다.

5. Trade-off 관계에 있는 precision과 recall을 하나의 measure로 보기 위해서 F1 Measure라는 것을 사용하기도 합니다. F1 Measure를 계산하는 수식은 아래와 같습니다. F1 값이 가장 크게 되는 threshold는 얼마인가요?

#train

graph\_f1\_train <- function(threshold){

a <-graph\_recall\_train(threshold)

b <- graph\_precision\_train(threshold)

c <- 2\*(b\*a)/(b+a)

return(c)

}

# test

graph\_f1\_test <- function(threshold){

a <-graph\_recall(threshold)

b <- graph\_precision(threshold)

c <- 2\*(b\*a)/(b+a)

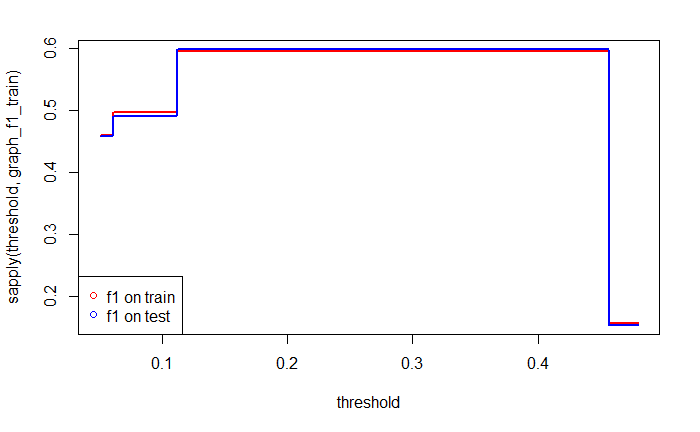
return(c)

}

plot(threshold,sapply(threshold,graph\_f1\_train ),type="l", col="red",lwd = 2)

lines(threshold,sapply(threshold,graph\_f1\_test),type="l", col="blue", lwd = 2)

legend("bottomleft", c("f1 on train", "f1 on test"), col=c("red", "blue"), pch = c(1,1))

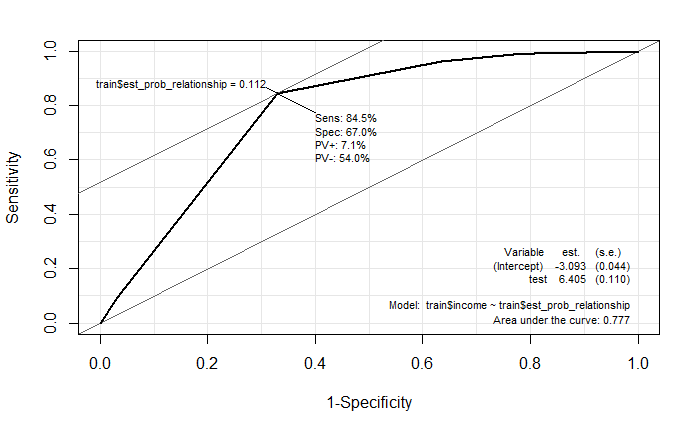


답 : train과 test data 모두 threshold가 0.14 부근부터 0.46부근까지 F1 값이 약 0.59로 가장 높다.

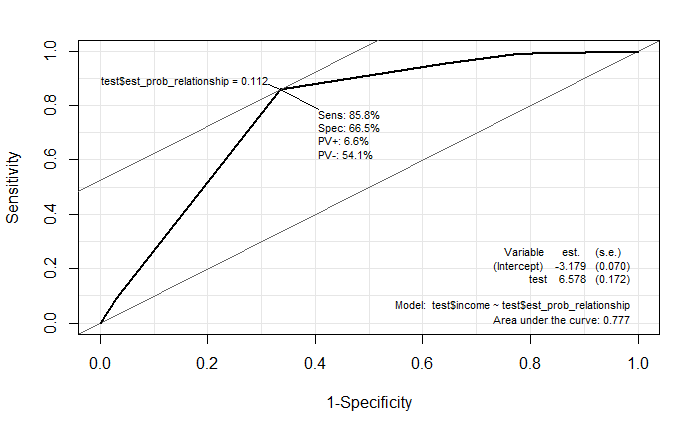
**6. 학습데이터와 테스트 데이터 각각에 대해서 여러분 모델의 ROC 커브를 그려보시오. 그래**

**프를 통해서 알 수 있는 점이 있다면 설명해보시오.**

1. **Train**



1. **Test**



답 : train data와 test data 비슷한 모양의 ROC 커브가 그려지는 것으로 보아 성능이 거의 같다고 할 수 있다.

**7. Accuracy가 가장 크게 되는 threshold는 얼마인가요? (학습 데이터 테스트 데이터 각각에**

**대해)**

threshold <- seq(0.05,0.48,0.001)

graph\_accuracy\_train <- function(threshold){

train$prediction <-train$est\_prob\_relationship > threshold

a <-get\_acuuracy(train$prediction,train$income)

return(a)

}

set.seed(1234)

accuracy\_train <- sapply(threshold, graph\_accuracy\_train)

plot(threshold,accuracy\_train, type = "l", xlim=c(0,0.5), ylim=c(0,1),col="red", lwd =2)

graph\_accuracy\_test <- function(threshold){

test$prediction <- test$est\_prob\_relationship > threshold

a <-get\_acuuracy(test$prediction,test$income)

return(a)

}

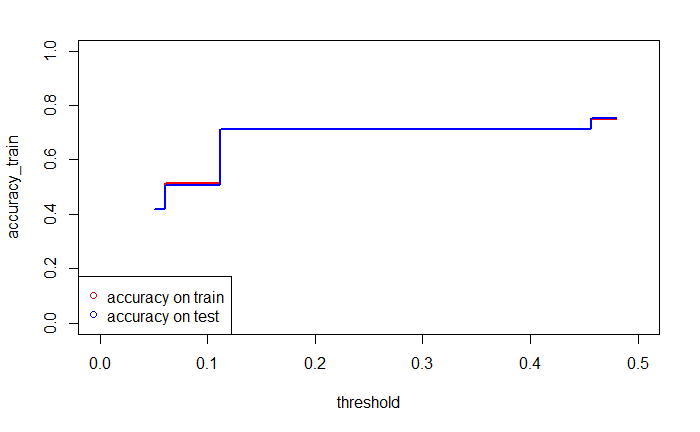
set.seed(1234)

accuracy\_test <- sapply(threshold, graph\_accuracy\_test)

plot(threshold,accuracy\_train, type = "l", xlim=c(0,0.5), ylim=c(0,1),col="red", lwd = 2)

lines(threshold,accuracy\_test, type = "l", xlim=c(0,0.5), ylim=c(0,1),col="blue", lwd = 2)

legend("bottomleft", c("accuracy on train", "accuracy on test"), col=c("red", "blue"), pch = c(1,1))



답 : train data와 test data 모두 threshold가 약 0.46 이상일때부터 0.48까지 accuracy가 최고치를 기록한다.

**8. 여러분이 찾은 모델의 성능을 Test 데이터 기준으로 다른 동료와 비교하시오. 여러분의**

**모델이 동료의 모델의 성능보다 낮다면 왜 그런지 생각해보고, 여러분의 모델을 개선하시**

**오. 비교 과정/결과와 개선 방법/개선 결과를 보고서에 기술하시오.**

답 : 나의 동료도 relationship으로 변수를 선택했으며, threshold값도 0.44 로 설정한 뒤 모델을 만들었기 때문에 성능이 매우 비슷했다. (컴퓨터 상에서는 accuracy, recall, precision, AUC 값 동일하다고 나와있지만 반올림을 반영한 것 일 수도 있다. 그러나 그것을 감안 하더라도 매우 비슷하다고 할 수있다.)