ABF\_Week6\_HW

AI MBA 이태환

2024-02-03

# data 불러오기  
data <- read.csv("GermanCredit.csv", header = T)

data[,c(1,2,4,5,7,8,10,11,13,15,16,18,20,21)] <- lapply( data[,c(1,2,4,5,7,8,10,11,13,15,16,18,20,21)], as.factor)  
data$good.bad <- as.factor(as.numeric(data$good.bad) - 1)  
str(data)

## 'data.frame': 1000 obs. of 21 variables:  
## $ good.bad: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 2 1 1 ...  
## $ checking: Factor w/ 4 levels "1","2","3","4": 2 1 4 4 1 4 2 1 4 2 ...  
## $ duration: int 9 18 6 12 18 36 18 40 12 26 ...  
## $ history : Factor w/ 5 levels "1","2","3","4",..: 5 3 5 3 3 5 3 5 3 3 ...  
## $ purpose : Factor w/ 10 levels "1","2","3","4",..: 3 1 4 1 1 2 4 7 1 2 ...  
## $ amount : int 1919 1216 1382 1101 2249 10477 1113 5998 2133 7966 ...  
## $ savings : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 2 5 1 1 5 1 ...  
## $ employed: Factor w/ 5 levels "1","2","3","4",..: 4 2 3 3 4 5 3 3 5 2 ...  
## $ installp: int 4 4 1 3 4 2 4 4 4 2 ...  
## $ martial : Factor w/ 4 levels "1","2","3","4": 3 2 2 4 3 3 2 3 2 3 ...  
## $ coapp : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 3 1 1 1 ...  
## $ resident: int 3 3 1 2 3 4 4 3 4 3 ...  
## $ property: Factor w/ 4 levels "1","2","3","4": 3 3 3 1 3 4 1 4 4 3 ...  
## $ age : int 35 23 28 27 30 42 26 27 52 30 ...  
## $ other : Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 1 3 3 ...  
## $ housing : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 3 2 2 3 2 ...  
## $ existcr : int 1 1 2 2 1 2 1 1 1 2 ...  
## $ job : Factor w/ 4 levels "1","2","3","4": 3 3 3 3 4 3 2 3 4 3 ...  
## $ depends : int 1 1 1 1 2 1 2 1 1 1 ...  
## $ telephon: Factor w/ 2 levels "1","2": 2 2 2 2 2 1 1 2 2 1 ...  
## $ foreign : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...

data$amount <- as.factor(ifelse(data$amount <= 2500, '0-2500', ifelse(data$amount <= 5000,'2500-5000','5000+')))  
head(data)

## good.bad checking duration history purpose amount savings employed installp  
## 1 0 2 9 5 3 0-2500 1 4 4  
## 2 1 1 18 3 1 0-2500 1 2 4  
## 3 0 4 6 5 4 0-2500 1 3 1  
## 4 0 4 12 3 1 0-2500 1 3 3  
## 5 0 1 18 3 1 0-2500 2 4 4  
## 6 0 4 36 5 2 5000+ 5 5 2  
## martial coapp resident property age other housing existcr job depends  
## 1 3 1 3 3 35 3 1 1 3 1  
## 2 2 1 3 3 23 3 1 1 3 1  
## 3 2 1 1 3 28 3 2 2 3 1  
## 4 4 1 2 1 27 3 2 2 3 1  
## 5 3 1 3 3 30 3 2 1 4 2  
## 6 3 1 4 4 42 3 3 2 3 1  
## telephon foreign  
## 1 2 1  
## 2 2 1  
## 3 2 1  
## 4 2 1  
## 5 2 1  
## 6 1 1

# 데이터 분리1

library(sampling)

n <- nrow(data) #data의 row 수를 n에 넣기기  
d <- sort(sample(n, n\*0.6)) #전체 행 중에 60%의 행을 랜덤하게 선택하여 d에 넣기  
train <- data[d,] # d에 해당하는 data 추출하여 train에 넣기  
test <- data[-d,] # d에 해당하지 않는 data 추출하여 test에 넣기

# 데이터 분리2

set.seed(123457)  
#good.bad == "0"인 행을 추출하여 수치형으로 변환 후 good\_idx에 넣기  
good\_idx <- as.numeric(row.names(data[data$good.bad == "0",]))   
#good.bad == "1"인 행을 추출하여 수치형으로 변환 후 bad\_idx에 넣기  
bad\_idx <- as.numeric(row.names(data[data$good.bad == "1",]))  
good\_n <- length(good\_idx)  
bad\_n <- length(bad\_idx)  
  
Train\_good\_idx <- sample(good\_idx, 0.6\*good\_n) #good\_idx 개수 중 60% 랜덤추출하여 Train\_good\_idx넣기  
Test\_good\_idx <- setdiff(good\_idx,Train\_good\_idx) #good\_idx에서 Train\_good\_idx를 뺀 나머지를 Test\_good\_idx에 넣기  
Train\_bad\_idx <- sample(bad\_idx, 0.6\*bad\_n)  
Test\_bad\_idx <- setdiff(bad\_idx,Train\_bad\_idx)  
#Train\_good\_idx, Train\_bad\_idx 합치고 오름차순으로 정렬하여 Train\_idx에 넣기  
Train\_idx <- sort(union(Train\_good\_idx, Train\_bad\_idx))   
train <- data[Train\_idx,]  
test <- data[-Train\_idx,]

# 데이터 분리3

library(splitstackshape)

set.seed(123457)  
#good.bad의 good/bad를 각각 60%씩 균일하게 추출하여 train1에 넣기  
train1 <- stratified(data, 'good.bad', 0.6, keep.rownames = TRUE)  
#추출된 train1의 rn열을 오름차순으로 정렬 후, data 에서 train1$rn을 제외한 행을 추출하여 test1에 넣기  
test1 <- data[-sort(as.numeric(train1$rn)), ]  
head(test1)

## good.bad checking duration history purpose amount savings employed installp  
## 1 0 2 9 5 3 0-2500 1 4 4  
## 3 0 4 6 5 4 0-2500 1 3 1  
## 8 1 1 40 5 7 5000+ 1 3 4  
## 9 0 4 12 3 1 0-2500 5 5 4  
## 11 0 1 24 3 3 0-2500 1 2 4  
## 14 0 2 12 5 5 0-2500 1 4 4  
## martial coapp resident property age other housing existcr job depends  
## 1 3 1 3 3 35 3 1 1 3 1  
## 3 2 1 1 3 28 3 2 2 3 1  
## 8 3 1 3 4 27 1 2 1 3 1  
## 9 2 1 4 4 52 3 3 1 4 1  
## 11 3 2 1 2 24 3 2 1 2 1  
## 14 3 1 3 2 26 3 2 1 3 1  
## telephon foreign  
## 1 2 1  
## 3 2 1  
## 8 2 1  
## 9 2 1  
## 11 1 2  
## 14 1 1

# logistic regression

Full\_model <- glm(good.bad~., data=train, family=binomial())  
step(Full\_model, direction="backward")

## Start: AIC=610.64  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + martial + coapp + resident +   
## property + age + other + housing + existcr + job + depends +   
## telephon + foreign  
##   
## Df Deviance AIC  
## - job 3 511.06 605.06  
## - martial 3 512.04 606.04  
## - property 3 512.96 606.96  
## - depends 1 510.93 608.93  
## - age 1 511.26 609.26  
## - foreign 1 511.36 609.36  
## - telephon 1 511.61 609.61  
## - resident 1 511.67 609.67  
## - employed 4 517.81 609.81  
## - coapp 2 514.25 610.25  
## - existcr 1 512.58 610.58  
## <none> 510.64 610.64  
## - installp 1 512.94 610.94  
## - housing 2 515.78 611.78  
## - duration 1 513.98 611.98  
## - other 2 516.48 612.48  
## - savings 4 520.52 612.52  
## - amount 2 517.93 613.93  
## - purpose 9 536.94 618.94  
## - history 4 530.48 622.48  
## - checking 3 556.80 650.80  
##   
## Step: AIC=605.06  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + martial + coapp + resident +   
## property + age + other + housing + existcr + depends + telephon +   
## foreign  
##   
## Df Deviance AIC  
## - martial 3 512.39 600.39  
## - property 3 513.33 601.33  
## - depends 1 511.28 603.28  
## - age 1 511.78 603.78  
## - foreign 1 511.79 603.79  
## - resident 1 512.09 604.09  
## - telephon 1 512.34 604.34  
## - employed 4 518.41 604.41  
## - coapp 2 514.65 604.65  
## - existcr 1 513.05 605.05  
## <none> 511.06 605.06  
## - installp 1 513.39 605.39  
## - housing 2 516.09 606.09  
## - duration 1 514.59 606.59  
## - savings 4 520.62 606.62  
## - other 2 516.75 606.75  
## - amount 2 518.25 608.25  
## - purpose 9 537.59 613.59  
## - history 4 531.04 617.04  
## - checking 3 556.86 644.86  
##   
## Step: AIC=600.39  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + resident + property +   
## age + other + housing + existcr + depends + telephon + foreign  
##   
## Df Deviance AIC  
## - property 3 514.40 596.40  
## - depends 1 512.46 598.46  
## - foreign 1 513.16 599.16  
## - age 1 513.22 599.22  
## - resident 1 513.28 599.28  
## - telephon 1 513.55 599.55  
## - coapp 2 516.16 600.16  
## - installp 1 514.36 600.36  
## - existcr 1 514.36 600.36  
## <none> 512.39 600.39  
## - employed 4 520.59 600.59  
## - duration 1 515.81 601.81  
## - savings 4 521.89 601.89  
## - other 2 517.93 601.93  
## - housing 2 518.31 602.31  
## - amount 2 519.46 603.46  
## - purpose 9 538.76 608.76  
## - history 4 531.91 611.91  
## - checking 3 559.77 641.77  
##   
## Step: AIC=596.4  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + resident + age +   
## other + housing + existcr + depends + telephon + foreign  
##   
## Df Deviance AIC  
## - depends 1 514.45 594.45  
## - foreign 1 515.30 595.30  
## - telephon 1 515.38 595.38  
## - resident 1 515.48 595.48  
## - age 1 515.52 595.52  
## <none> 514.40 596.40  
## - existcr 1 516.61 596.61  
## - installp 1 516.69 596.69  
## - employed 4 522.72 596.72  
## - coapp 2 519.46 597.46  
## - savings 4 524.05 598.05  
## - duration 1 518.32 598.32  
## - other 2 520.40 598.40  
## - amount 2 521.74 599.74  
## - housing 2 522.29 600.29  
## - purpose 9 542.58 606.58  
## - history 4 535.40 609.40  
## - checking 3 562.16 638.16  
##   
## Step: AIC=594.45  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + resident + age +   
## other + housing + existcr + telephon + foreign  
##   
## Df Deviance AIC  
## - foreign 1 515.32 593.32  
## - telephon 1 515.47 593.47  
## - resident 1 515.54 593.54  
## - age 1 515.57 593.57  
## <none> 514.45 594.45  
## - installp 1 516.71 594.71  
## - existcr 1 516.71 594.71  
## - employed 4 522.76 594.76  
## - coapp 2 519.46 595.46  
## - savings 4 524.05 596.05  
## - duration 1 518.35 596.35  
## - other 2 520.53 596.53  
## - amount 2 521.78 597.78  
## - housing 2 522.31 598.31  
## - purpose 9 542.62 604.62  
## - history 4 535.74 607.74  
## - checking 3 562.21 636.21  
##   
## Step: AIC=593.32  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + resident + age +   
## other + housing + existcr + telephon  
##   
## Df Deviance AIC  
## - telephon 1 516.32 592.32  
## - resident 1 516.40 592.40  
## - age 1 516.50 592.50  
## <none> 515.32 593.32  
## - employed 4 523.70 593.70  
## - installp 1 517.78 593.78  
## - existcr 1 517.87 593.87  
## - coapp 2 520.82 594.82  
## - savings 4 525.20 595.20  
## - other 2 521.42 595.42  
## - duration 1 519.89 595.89  
## - amount 2 522.23 596.23  
## - housing 2 523.53 597.53  
## - purpose 9 543.05 603.05  
## - history 4 536.85 606.85  
## - checking 3 563.27 635.27  
##   
## Step: AIC=592.32  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + resident + age +   
## other + housing + existcr  
##   
## Df Deviance AIC  
## - resident 1 517.38 591.38  
## - age 1 517.70 591.70  
## <none> 516.32 592.32  
## - employed 4 524.63 592.63  
## - existcr 1 518.65 592.65  
## - installp 1 518.67 592.67  
## - coapp 2 521.79 593.79  
## - savings 4 526.60 594.60  
## - amount 2 522.61 594.61  
## - other 2 522.66 594.66  
## - duration 1 520.79 594.79  
## - housing 2 524.34 596.34  
## - purpose 9 544.92 602.92  
## - history 4 538.00 606.00  
## - checking 3 565.63 635.63  
##   
## Step: AIC=591.38  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + age + other + housing +   
## existcr  
##   
## Df Deviance AIC  
## - age 1 518.52 590.52  
## - employed 4 525.17 591.17  
## <none> 517.38 591.38  
## - installp 1 519.80 591.80  
## - existcr 1 519.97 591.97  
## - coapp 2 522.83 592.83  
## - other 2 523.25 593.25  
## - savings 4 527.37 593.37  
## - amount 2 523.44 593.44  
## - duration 1 522.07 594.07  
## - housing 2 527.41 597.41  
## - purpose 9 545.51 601.51  
## - history 4 538.70 604.70  
## - checking 3 566.61 634.61  
##   
## Step: AIC=590.52  
## good.bad ~ checking + duration + history + purpose + amount +   
## savings + employed + installp + coapp + other + housing +   
## existcr  
##   
## Df Deviance AIC  
## <none> 518.52 590.52  
## - installp 1 520.77 590.77  
## - existcr 1 520.88 590.88  
## - employed 4 527.18 591.18  
## - other 2 524.16 592.16  
## - coapp 2 524.21 592.21  
## - amount 2 524.29 592.29  
## - savings 4 528.91 592.91  
## - duration 1 523.92 593.92  
## - housing 2 528.48 596.48  
## - purpose 9 546.14 600.14  
## - history 4 540.45 604.45  
## - checking 3 568.32 634.32

##   
## Call: glm(formula = good.bad ~ checking + duration + history + purpose +   
## amount + savings + employed + installp + coapp + other +   
## housing + existcr, family = binomial(), data = train)  
##   
## Coefficients:  
## (Intercept) checking2 checking3 checking4   
## 0.90963 -0.57647 -0.72327 -2.01092   
## duration history2 history3 history4   
## 0.02886 0.61349 -0.94671 -0.99412   
## history5 purpose2 purpose3 purpose4   
## -1.75951 -1.68045 -0.95181 -0.70174   
## purpose5 purpose6 purpose7 purpose8   
## -0.87285 1.24055 0.08993 1.10662   
## purpose9 purpose10 amount2500-5000 amount5000+   
## -0.89027 -2.25096 -0.07531 0.83856   
## savings2 savings3 savings4 savings5   
## -0.38650 0.12875 -1.23181 -0.92614   
## employed2 employed3 employed4 employed5   
## 0.80042 0.43952 -0.27933 0.24602   
## installp coapp2 coapp3 other2   
## 0.17123 0.63332 -1.01892 -0.53169   
## other3 housing2 housing3 existcr   
## -0.74834 -0.83584 -0.17506 0.39900   
##   
## Degrees of Freedom: 599 Total (i.e. Null); 564 Residual  
## Null Deviance: 733   
## Residual Deviance: 518.5 AIC: 590.5

# 참고

library(dplyr)

library(MASS) # Cars93 dataframe을 얻기 위함 . 자동차에 관한 27 개 변수 93 개 관측값으로 구성

Cars93 데이터프레임에서 %>% (2) 제조생산국 (Origin), 차종 (Type), 실린더개수 (Cylinders) 별로 %>%

차 가격 (Price) 과 고속도로 연비 (MPG.highway) 변수에 대하여 결측값은 제외하고 ) 평균 을 구하는데 , %>% (4) 단 , 가격 평균은 10 을 넘거나 또는 고속도로 연비는 25 를 넘는 것만 선택 출력

dataframe %>% group\_by() %>% summarise() %>% filter()의 순서

Cars93 %>% group\_by(Origin, Type, Cylinders) %>%  
 summarise(P\_m = mean(Price, na.rm = T),  
 M\_m = mean(MPG.highway, na.rm = T)) %>%  
 filter(P\_m > 10 | M\_m > 25)

## `summarise()` has grouped output by 'Origin', 'Type'. You can override using  
## the `.groups` argument.

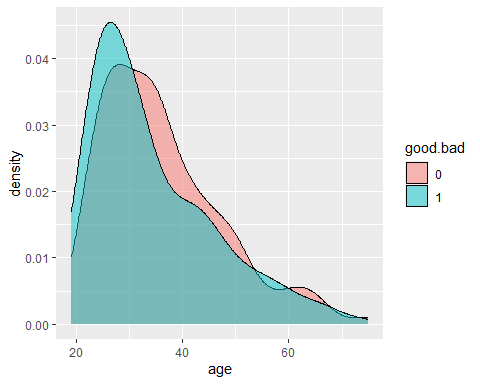
## # A tibble: 25 × 5  
## # Groups: Origin, Type [11]  
## Origin Type Cylinders P\_m M\_m  
## <fct> <fct> <fct> <dbl> <dbl>  
## 1 USA Compact 4 12.8 30.6  
## 2 USA Large 6 22.4 27.3  
## 3 USA Large 8 27.6 25.8  
## 4 USA Midsize 4 15.9 29.5  
## 5 USA Midsize 6 22.8 27.2  
## 6 USA Midsize 8 40.1 25   
## 7 USA Small 4 10.0 33.9  
## 8 USA Sporty 4 14.6 28.8  
## 9 USA Sporty 6 19.5 26.7  
## 10 USA Sporty 8 38 25   
## # ℹ 15 more rows

# 변수별 우량/불량에 대한 분포

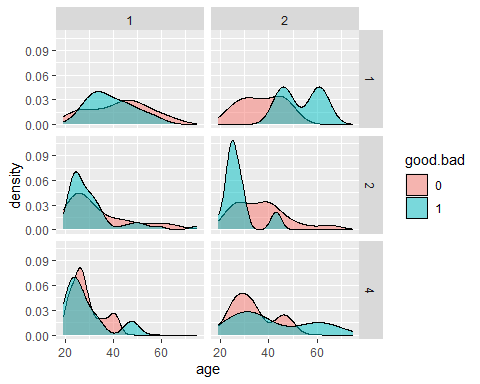
library(dplyr)  
library(ggplot2)

## Warning: 패키지 'ggplot2'는 R 버전 4.3.2에서 작성되었습니다

train %>% ggplot(aes(age, fill = good.bad)) + geom\_density(alpha = 0.5)



train %>% filter(martial %in% c('1', '2', '4'), existcr %in% c('1', '2')) %>%  
 ggplot(aes(age, fill = good.bad)) + geom\_density(alpha = 0.5) +  
 facet\_grid(martial ~ existcr)

### 첫 번째 코드 라인

train %>% ggplot(aes(age, fill = good.bad)) + geom\_density(alpha = 0.5)

train %>%: dplyr 패키지의 파이프 연산자 %>%를 사용하여 train 데이터 프레임을 다음 함수로 전달합니다.

ggplot(aes(age, fill = good.bad)): ggplot2 패키지의 ggplot 함수를 사용하여 데이터의 시각화를 시작합니다. aes 함수는 미적 요소를 정의하는데 사용되며, 여기서는 age를 x축에 매핑하고, good.bad 변수의 값에 따라 색상을 다르게 하여 밀도 그래프를 채우도록 설정합니다.

geom\_density(alpha = 0.5): 데이터의 밀도를 표시하는 그래프를 추가합니다. alpha = 0.5는 그래프의 투명도를 조절하여 겹치는 부분의 구분을 용이하게 합니다.

### 두 번째 코드 라인

train %>% filter(martial %in% c('1', '2', '4'), existcr %in% c('1', '2')) %>%  
 ggplot(aes(age, fill = good.bad)) + geom\_density(alpha = 0.5) +  
 facet\_grid(martial ~ existcr)

train %>% filter(martial %in% c('1', '2', '4'), existcr %in% c('1', '2')): filter 함수를 사용하여 train 데이터 프레임에서 martial 변수의 값이 ‘1’, ‘2’, 또는 ‘4’인 행과, existcr 변수의 값이 ’1’ 또는 ’2’인 행만을 선택합니다.

ggplot(aes(age, fill = good.bad)) + geom\_density(alpha = 0.5): 첫 번째 코드 라인과 동일하게 age 변수를 기반으로 한 밀도 그래프를 생성하고, good.bad 변수의 값에 따라 색상을 구분합니다.

facet\_grid(martial ~ existcr): facet\_grid를 사용하여 그래프를 martial 변수의 값에 따라 행으로, existcr 변수의 값에 따라 열로 분할하여 여러 패널로 나누어 시각화합니다. 이는 데이터의 서브그룹 간 비교를 용이하게 합니다.

이 코드들은 데이터의 분포를 시각적으로 탐색하고, 특정 조건을 만족하는 서브그룹 내에서 age 변수의 분포가 good.bad 변수의 값에 따라 어떻게 다른지 비교하는 데 사용됩니다.

# 최종 모형 설정 및 계수 특징 확인

Logistic\_Model <-  
 glm(  
 good.bad ~ checking + duration + history + purpose + amount + savings +  
 employed + installp + coapp + other + housing + existcr,  
 family = binomial(),  
 data = train  
 )  
summary(Logistic\_Model)

##   
## Call:  
## glm(formula = good.bad ~ checking + duration + history + purpose +   
## amount + savings + employed + installp + coapp + other +   
## housing + existcr, family = binomial(), data = train)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.90963 1.03071 0.883 0.377492   
## checking2 -0.57647 0.28931 -1.993 0.046310 \*   
## checking3 -0.72327 0.47809 -1.513 0.130320   
## checking4 -2.01092 0.31029 -6.481 9.13e-11 \*\*\*  
## duration 0.02886 0.01244 2.320 0.020331 \*   
## history2 0.61349 0.73645 0.833 0.404823   
## history3 -0.94671 0.55628 -1.702 0.088783 .   
## history4 -0.99412 0.62853 -1.582 0.113726   
## history5 -1.75951 0.58607 -3.002 0.002680 \*\*   
## purpose2 -1.68045 0.50279 -3.342 0.000831 \*\*\*  
## purpose3 -0.95181 0.35713 -2.665 0.007696 \*\*   
## purpose4 -0.70174 0.31301 -2.242 0.024967 \*   
## purpose5 -0.87285 1.25453 -0.696 0.486577   
## purpose6 1.24055 0.78906 1.572 0.115907   
## purpose7 0.08993 0.57909 0.155 0.876589   
## purpose8 1.10662 1.81645 0.609 0.542379   
## purpose9 -0.89027 0.43360 -2.053 0.040056 \*   
## purpose10 -2.25096 0.99272 -2.267 0.023361 \*   
## amount2500-5000 -0.07531 0.31384 -0.240 0.810355   
## amount5000+ 0.83856 0.43125 1.944 0.051836 .   
## savings2 -0.38650 0.36859 -1.049 0.294358   
## savings3 0.12875 0.51313 0.251 0.801884   
## savings4 -1.23181 0.71830 -1.715 0.086363 .   
## savings5 -0.92614 0.36793 -2.517 0.011830 \*   
## employed2 0.80042 0.58036 1.379 0.167835   
## employed3 0.43952 0.55587 0.791 0.429122   
## employed4 -0.27933 0.59460 -0.470 0.638518   
## employed5 0.24602 0.56095 0.439 0.660963   
## installp 0.17123 0.11482 1.491 0.135878   
## coapp2 0.63332 0.63461 0.998 0.318296   
## coapp3 -1.01892 0.51345 -1.984 0.047203 \*   
## other2 -0.53169 0.56003 -0.949 0.342416   
## other3 -0.74834 0.31271 -2.393 0.016709 \*   
## housing2 -0.83584 0.28945 -2.888 0.003881 \*\*   
## housing3 -0.17506 0.43139 -0.406 0.684893   
## existcr 0.39900 0.26376 1.513 0.130343   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 733.04 on 599 degrees of freedom  
## Residual deviance: 518.52 on 564 degrees of freedom  
## AIC: 590.52  
##   
## Number of Fisher Scoring iterations: 5

predict(Logistic\_Model, type='response', test[1:3,]) #test 맨위 3개 행 선택하여 예측(1, 3, 8행)

## 1 3 8   
## 0.11282710 0.02752324 0.87219304

# 결과는 확률로 출력됨

test[1:3, ]$good.bad #맨 위 3개 행의 good/bad는 bad, bad, good임

## [1] 0 0 1  
## Levels: 0 1

테스트 데이터를 전부 예측하여 test$phat에 저장

test$phat <- predict(Logistic\_Model, type='response', test)

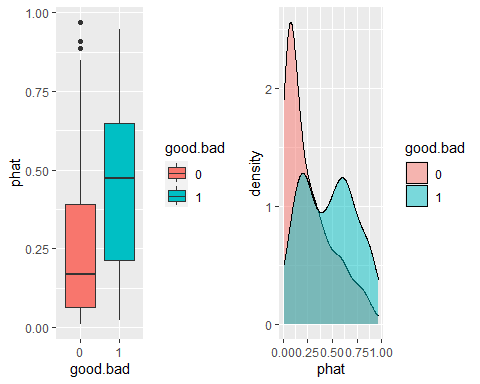
library(gridExtra)

## Warning: 패키지 'gridExtra'는 R 버전 4.3.2에서 작성되었습니다

##   
## 다음의 패키지를 부착합니다: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

F1 <- test %>% ggplot(aes(good.bad, phat, group=good.bad, fill=good.bad))+geom\_boxplot()  
F2 <- test %>% ggplot(aes(phat, fill=good.bad))+geom\_density(alpha=0.5)  
grid.arrange(F1,F2, ncol=2)



이 코드는 R 언어를 사용하여 데이터 시각화를 수행하는 과정을 나타냅니다. 여기서도 dplyr와 ggplot2 패키지, 그리고 gridExtra 패키지의 기능을 활용합니다. 각 부분의 작업을 설명하겠습니다.

### 첫 번째 그래프 (F1)

F1 <- test %>% ggplot(aes(good.bad, phat, group=good.bad, fill=good.bad)) + geom\_boxplot()

test %>%: dplyr의 파이프 연산자를 사용하여 test 데이터 프레임을 ggplot 함수로 전달합니다.

ggplot(aes(good.bad, phat, group=good.bad, fill=good.bad)): ggplot 함수로 박스플롯을 그릴 기본 설정을 합니다. 여기서는 x축에 good.bad 변수, y축에 phat 변수를 사용합니다. group=good.bad와 fill=good.bad는 박스플롯의 그룹화와 색상 채우기를 good.bad 변수의 값에 따라 다르게 설정합니다.

geom\_boxplot(): 박스플롯을 생성하는 명령입니다. 이는 phat 값의 분포를 good.bad 값에 따라 비교하기 위해 사용됩니다.

### 두 번째 그래프 (F2)

F2 <- test %>% ggplot(aes(phat, fill=good.bad)) + geom\_density(alpha=0.5)

ggplot(aes(phat, fill=good.bad)): 데이터의 밀도 그래프를 그리기 위한 기본 설정을 합니다. 여기서는 phat 값을 연속적인 x축 변수로 사용하고, good.bad 변수의 값에 따라 색상을 다르게 적용합니다.

geom\_density(alpha=0.5): 데이터의 밀도를 나타내는 그래프를 생성하며, alpha=0.5로 투명도를 조정해 겹치는 부분을 더 잘 볼 수 있게 합니다.

### 두 그래프 병렬 배치

grid.arrange(F1, F2, ncol=2)

grid.arrange(F1, F2, ncol=2): gridExtra 패키지의 grid.arrange 함수를 사용하여 F1과 F2라는 두 개의 그래프를 나란히 배치합니다. ncol=2는 두 그래프를 2열로 배치하라는 의미입니다.

이 코드는 test 데이터 세트에서 good.bad 변수에 따른 phat 값의 분포를 박스플롯과 밀도 그래프로 시각화하여, 이 두 변수 간의 관계를 탐색하는 데 사용됩니다. F1은 good.bad의 각 값에 대한 phat의 분포를 박스플롯으로 보여주고, F2는 phat 값의 전체적인 밀도 분포를 good.bad의 값에 따라 색으로 구분하여 보여줍니다.

# 모형 분석을 통한 승인 또는 거절 이유 확인

Terms=predict(Logistic\_Model, type='terms',test[1,])  
Tmp=abs(Terms) #절대값으로 변경  
Tmp

## checking duration history purpose amount savings employed  
## 1 0.4374457 0.3408454 0.7099754 0.3012243 0.130482 0.2505368 0.5796587  
## installp coapp other housing existcr  
## 1 0.1740859 0.04319091 0.1098881 0.6176889 0.1522841  
## attr(,"constant")  
## [1] -1.260773

print(order(Tmp, decreasing = TRUE))#Tmp값(절대값)을 내림차순으로 정렬 후, 해당값의 인덱스 번호를 나열

## [1] 3 11 7 1 2 4 6 8 12 5 10 9

Terms[,order(Tmp, decreasing = TRUE)][1:5] #Tmp값(절대값)을 내림차순으로 정렬 후, 해당값의 인덱스 번호로 상위 5개 데이터를 추출

## history housing employed checking duration   
## -0.7099754 0.6176889 -0.5796587 0.4374457 -0.3408454

res=names(Terms[,order(Tmp, decreasing = TRUE)][1:5]) # 열이름 추출  
print(res)

## [1] "history" "housing" "employed" "checking" "duration"

paste(res,collapse="; ",sep="")

## [1] "history; housing; employed; checking; duration"

R의 paste 함수는 벡터를 문자열로 변환한 후 연결하는 데 사용됩니다. ‘collapse’ 및 ‘sep’ 인수는 이러한 문자열이 결합되는 방식을 제어합니다. 제공한 코드 조각에서 다음을 수행합니다.

res는 단일 문자열로 연결하려는 요소의 벡터입니다.

collapse="; "는 res의 요소를 단일 문자열로 연결하고 각 요소를 ;로 구분하도록 지정합니다.(세미콜론 다음에 공백이 옵니다).

sep=""는 요소가 연결될 때 요소를 구분하는 데 사용되는 문자열을 정의하므로 이 맥락에서 다소 중복됩니다. 그러나 ’collapse’는 모든 요소를 단일 문자열로 병합하는 데 사용되므로 ’sep’의 역할은 ’collapse’에 의해 가려집니다. collapse가 없으면 sep은 res의 개별 요소가 결합되는 방식을 정의합니다.

test$phat[1]

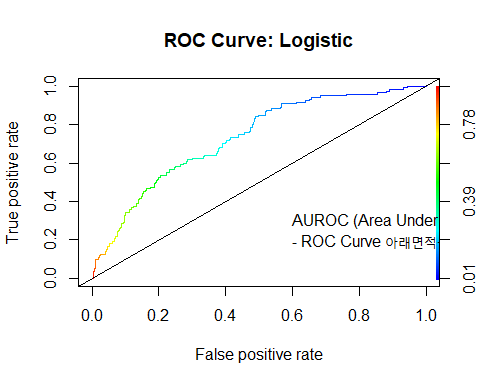
## [1] 0.1128271

# ROC Curve

library(ROCR)

## Warning: 패키지 'ROCR'는 R 버전 4.3.2에서 작성되었습니다

pred=prediction(test$phat, test$good.bad)  
perf=performance(pred,"tpr","fpr")  
plot(perf,colorize=TRUE, main='ROC Curve: Logistic')  
text(0.6,0.3,adj=0,labels="AUROC (Area Under the ROC Curve)")  
text(0.6,0.2,adj=0,labels="- ROC Curve 아래면적을 가르킴")  
abline(0,1) #절편이 0 기울기가 1인 라인 그리기



performance(pred,"auc")@y.values[[1]] # AUROC

## [1] 0.7301637

# Confusion Matrix 관련 지표

library(caret)

## Warning: 패키지 'caret'는 R 버전 4.3.2에서 작성되었습니다

## 필요한 패키지를 로딩중입니다: lattice

##   
## 다음의 패키지를 부착합니다: 'caret'

## The following object is masked from 'package:sampling':  
##   
## cluster

library(e1071)

## Warning: 패키지 'e1071'는 R 버전 4.3.2에서 작성되었습니다

confusionMatrix(data=as.factor(as.numeric(test$phat>0.5)), reference=test$good.bad)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 232 64  
## 1 48 56  
##   
## Accuracy : 0.72   
## 95% CI : (0.6732, 0.7635)  
## No Information Rate : 0.7   
## P-Value [Acc > NIR] : 0.2073   
##   
## Kappa : 0.3069   
##   
## Mcnemar's Test P-Value : 0.1564   
##   
## Sensitivity : 0.8286   
## Specificity : 0.4667   
## Pos Pred Value : 0.7838   
## Neg Pred Value : 0.5385   
## Prevalence : 0.7000   
## Detection Rate : 0.5800   
## Detection Prevalence : 0.7400   
## Balanced Accuracy : 0.6476   
##   
## 'Positive' Class : 0   
##

# Cost Marix를 고려한 Cut Off 설정

t <- data.frame(CutOff=0, Type11=0, Type12=0, Type51=0, Type52=0, Acc=0)  
t

## CutOff Type11 Type12 Type51 Type52 Acc  
## 1 0 0 0 0 0 0

for(i in seq(0.01, 0.99, 0.01)) {  
 CM = confusionMatrix(data = as.factor(as.numeric(test$phat > i)),  
 reference = test$good.bad)  
 t1 = CM$overall[[1]] # Accuracy  
 t2 = CM$table[1, 2] # Predict Good at Bad (Type 1) #good을 bad로 예측  
 t3 = CM$table[2, 1] # Predict Bad at Good (Type 2) #bad를 good으로 예측  
 tt = data.frame(  
 CutOff = i,  
 Type11 = 0.5 \* t2,  
 Type12 = 0.5 \* t3,  
 Type51 = (5 / 6) \* t2,  
 Type52 = t3 / 6,  
 Acc = t1  
 )  
 t = rbind(t, tt)  
}

## Warning in confusionMatrix.default(data = as.factor(as.numeric(test$phat > :  
## Levels are not in the same order for reference and data. Refactoring data to  
## match.  
  
## Warning in confusionMatrix.default(data = as.factor(as.numeric(test$phat > :  
## Levels are not in the same order for reference and data. Refactoring data to  
## match.  
  
## Warning in confusionMatrix.default(data = as.factor(as.numeric(test$phat > :  
## Levels are not in the same order for reference and data. Refactoring data to  
## match.

head(t)

## CutOff Type11 Type12 Type51 Type52 Acc  
## 1 0.00 0.0 0.0 0.000000 0.00000 0.0000  
## 2 0.01 0.0 138.5 0.000000 46.16667 0.3075  
## 3 0.02 0.0 133.0 0.000000 44.33333 0.3350  
## 4 0.03 1.0 129.0 1.666667 43.00000 0.3500  
## 5 0.04 2.0 119.5 3.333333 39.83333 0.3925  
## 6 0.05 2.5 114.0 4.166667 38.00000 0.4175

tail(t)

## CutOff Type11 Type12 Type51 Type52 Acc  
## 95 0.94 59 0.5 98.33333 0.1666667 0.7025  
## 96 0.95 60 0.5 100.00000 0.1666667 0.6975  
## 97 0.96 60 0.5 100.00000 0.1666667 0.6975  
## 98 0.97 60 0.0 100.00000 0.0000000 0.7000  
## 99 0.98 60 0.0 100.00000 0.0000000 0.7000  
## 100 0.99 60 0.0 100.00000 0.0000000 0.7000

t <- t[-1, ] #CutOff 0인 행 제거  
t

## CutOff Type11 Type12 Type51 Type52 Acc  
## 2 0.01 0.0 138.5 0.000000 46.1666667 0.3075  
## 3 0.02 0.0 133.0 0.000000 44.3333333 0.3350  
## 4 0.03 1.0 129.0 1.666667 43.0000000 0.3500  
## 5 0.04 2.0 119.5 3.333333 39.8333333 0.3925

… 생략….  
## 99 0.98 60.0 0.0 100.000000 0.0000000 0.7000  
## 100 0.99 60.0 0.0 100.000000 0.0000000 0.7000

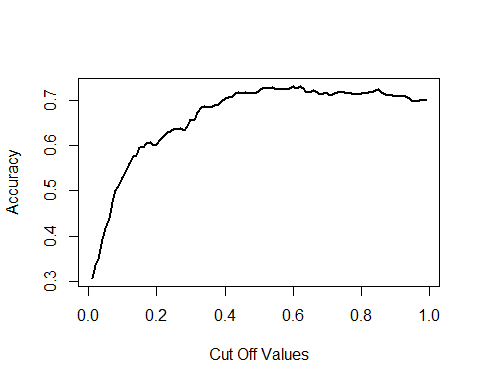
t$Type1 = t$Type11 + t$Type12  
t$Type5 = t$Type51 + t$Type52  
t[t$Type1 == min(t$Type1), ]

## CutOff Type11 Type12 Type51 Type52 Acc Type1 Type5  
## 61 0.60 38.5 15.5 64.16667 5.166667 0.73 54 69.33333  
## 63 0.62 40.0 14.0 66.66667 4.666667 0.73 54 71.33333

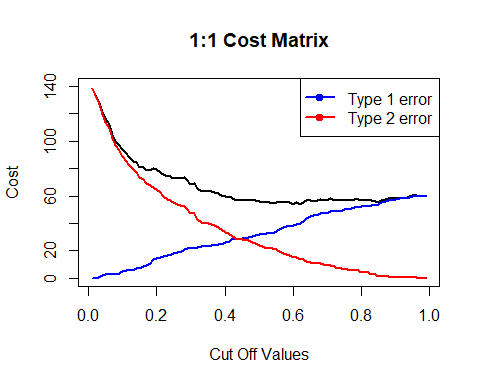
t[t$Type5==min(t$Type5),]

## CutOff Type11 Type12 Type51 Type52 Acc Type1 Type5  
## 14 0.13 5.5 79.5 9.166667 26.5 0.575 85 35.66667

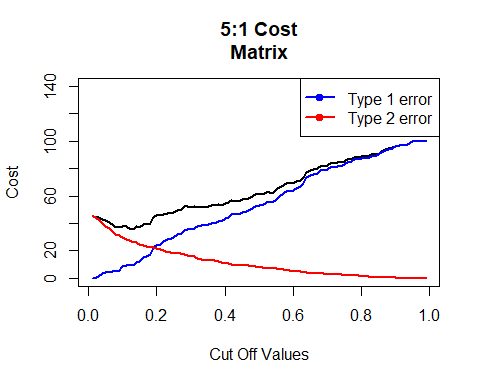
plot(t$CutOff,t$Acc, type="l", lwd=2, xlab="Cut Off Values", ylab="Accuracy")



plot(t$CutOff,t$Type1, type="l", lwd=2, xlab="Cut Off Values", ylab="Cost",main="1:1 Cost Matrix", ylim=c(0,140))  
lines(t$CutOff,t$Type11, type="l", lwd=2, col="blue")  
lines(t$CutOff,t$Type12,type="l", lwd=2, col="red")  
legend("topright", legend = c("Type 1 error","Type 2 error"), lwd=2, pch = 19, col = c("blue","red"))



plot(t$CutOff,t$Type5, type="l", lwd=2, xlab="Cut Off Values", ylab="Cost",main="5:1 Cost  
Matrix", ylim=c(0,140))  
lines(t$CutOff,t$Type51, type="l", lwd=2, col="blue")  
lines(t$CutOff,t$Type52,type="l", lwd=2, col="red")  
legend("topright", legend = c("Type 1 error","Type 2 error"), lwd=2, pch = 19, col = c("blue","red"))



# Tree 모델

library(rpart)

## Warning: 패키지 'rpart'는 R 버전 4.3.2에서 작성되었습니다

fit1=rpart(good.bad~., data=train)  
fit1

## n= 600   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 600 180 0 (0.7000000 0.3000000)   
## 2) checking=3,4 279 36 0 (0.8709677 0.1290323) \*  
## 3) checking=1,2 321 144 0 (0.5514019 0.4485981)   
## 6) history=3,4,5 273 106 0 (0.6117216 0.3882784)   
## 12) duration< 27.5 214 69 0 (0.6775701 0.3224299)   
## 24) purpose=2,3,4,7,9,10 151 36 0 (0.7615894 0.2384106) \*  
## 25) purpose=1,5,6,8 63 30 1 (0.4761905 0.5238095)   
## 50) duration< 11.5 16 3 0 (0.8125000 0.1875000) \*  
## 51) duration>=11.5 47 17 1 (0.3617021 0.6382979)   
## 102) martial=3 22 10 0 (0.5454545 0.4545455)   
## 204) employed=1,2,3,4 15 4 0 (0.7333333 0.2666667) \*  
## 205) employed=5 7 1 1 (0.1428571 0.8571429) \*  
## 103) martial=1,2,4 25 5 1 (0.2000000 0.8000000) \*  
## 13) duration>=27.5 59 22 1 (0.3728814 0.6271186)   
## 26) purpose=2,9,10 16 6 0 (0.6250000 0.3750000) \*  
## 27) purpose=1,3,4,6,7 43 12 1 (0.2790698 0.7209302) \*  
## 7) history=1,2 48 10 1 (0.2083333 0.7916667) \*

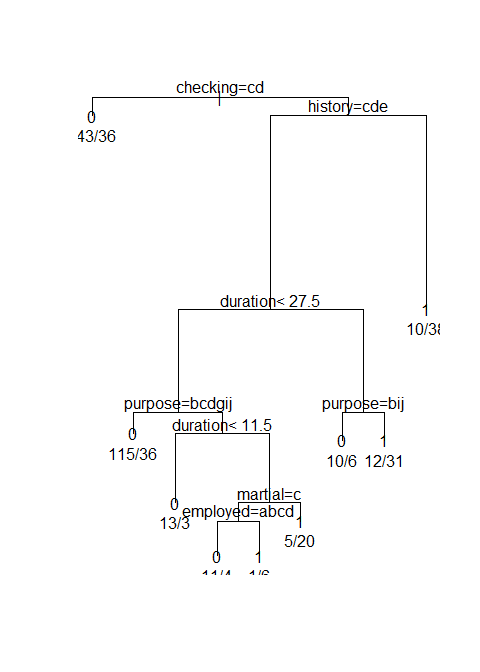
#결과를 자세하게 보고 싶은 경우  
printcp(fit1)

##   
## Classification tree:  
## rpart(formula = good.bad ~ ., data = train)  
##   
## Variables actually used in tree construction:  
## [1] checking duration employed history martial purpose   
##   
## Root node error: 180/600 = 0.3  
##   
## n= 600   
##   
## CP nsplit rel error xerror xstd  
## 1 0.077778 0 1.00000 1.00000 0.062361  
## 2 0.036111 3 0.76111 0.90000 0.060415  
## 3 0.022222 5 0.68889 0.86667 0.059691  
## 4 0.019444 6 0.66667 0.81667 0.058527  
## 5 0.010000 8 0.62778 0.80000 0.058119

summary(fit1)

## Call:  
## rpart(formula = good.bad ~ ., data = train)  
## n= 600   
##   
## CP nsplit rel error xerror xstd  
## 1 0.07777778 0 1.0000000 1.0000000 0.06236096  
## 2 0.03611111 3 0.7611111 0.9000000 0.06041523  
## 3 0.02222222 5 0.6888889 0.8666667 0.05969056  
## 4 0.01944444 6 0.6666667 0.8166667 0.05852745  
## 5 0.01000000 8 0.6277778 0.8000000 0.05811865  
##   
## Variable importance  
## checking history duration purpose employed savings age martial   
## 29 18 13 12 6 5 4 3   
## amount resident coapp housing foreign depends   
## 2 2 1 1 1 1   
##   
## Node number 1: 600 observations, complexity param=0.07777778  
## predicted class=0 expected loss=0.3 P(node) =1  
## class counts: 420 180  
## probabilities: 0.700 0.300   
## left son=2 (279 obs) right son=3 (321 obs)  
## Primary splits:  
## checking splits as RRLL, improve=30.486580, (0 missing)  
## history splits as RRLLL, improve=23.100650, (0 missing)  
## housing splits as RLR, improve=13.805750, (0 missing)  
## property splits as LLLR, improve= 9.604356, (0 missing)  
## purpose splits as RLLLLRRRRL, improve= 8.870926, (0 missing)  
## Surrogate splits:  
## savings splits as RRLLL, agree=0.613, adj=0.168, (0 split)  
## history splits as RRRLL, agree=0.600, adj=0.140, (0 split)  
## purpose splits as RLRLLRRLRR, agree=0.575, adj=0.086, (0 split)  
## age < 31.5 to the right, agree=0.567, adj=0.068, (0 split)  
## employed splits as RRRRL, agree=0.558, adj=0.050, (0 split)  
##   
## Node number 2: 279 observations  
## predicted class=0 expected loss=0.1290323 P(node) =0.465  
## class counts: 243 36  
## probabilities: 0.871 0.129   
##   
## Node number 3: 321 observations, complexity param=0.07777778  
## predicted class=0 expected loss=0.4485981 P(node) =0.535  
## class counts: 177 144  
## probabilities: 0.551 0.449   
## left son=6 (273 obs) right son=7 (48 obs)  
## Primary splits:  
## history splits as RRLLL, improve=13.285420, (0 missing)  
## property splits as LLRR, improve= 7.784637, (0 missing)  
## duration < 22.5 to the left, improve= 7.366218, (0 missing)  
## purpose splits as RLLLRRRRLL, improve= 6.699165, (0 missing)  
## housing splits as RLR, improve= 6.367960, (0 missing)  
##   
## Node number 6: 273 observations, complexity param=0.07777778  
## predicted class=0 expected loss=0.3882784 P(node) =0.455  
## class counts: 167 106  
## probabilities: 0.612 0.388   
## left son=12 (214 obs) right son=13 (59 obs)  
## Primary splits:  
## duration < 27.5 to the left, improve=8.587088, (0 missing)  
## purpose splits as RLLLRRRRLL, improve=6.822542, (0 missing)  
## property splits as LLRR, improve=6.175468, (0 missing)  
## coapp splits as RRL, improve=3.023874, (0 missing)  
## age < 25.5 to the right, improve=2.350006, (0 missing)  
## Surrogate splits:  
## amount splits as LLR, agree=0.835, adj=0.237, (0 split)  
## purpose splits as LLLLLLLLLR, agree=0.788, adj=0.017, (0 split)  
##   
## Node number 7: 48 observations  
## predicted class=1 expected loss=0.2083333 P(node) =0.08  
## class counts: 10 38  
## probabilities: 0.208 0.792   
##   
## Node number 12: 214 observations, complexity param=0.03611111  
## predicted class=0 expected loss=0.3224299 P(node) =0.3566667  
## class counts: 145 69  
## probabilities: 0.678 0.322   
## left son=24 (151 obs) right son=25 (63 obs)  
## Primary splits:  
## purpose splits as RLLLRRLRLL, improve=7.241664, (0 missing)  
## coapp splits as RRL, improve=3.090638, (0 missing)  
## duration < 8.5 to the left, improve=2.986240, (0 missing)  
## property splits as LLRR, improve=2.872278, (0 missing)  
## housing splits as RLR, improve=2.231328, (0 missing)  
## Surrogate splits:  
## age < 67.5 to the left, agree=0.71, adj=0.016, (0 split)  
## job splits as RLLL, agree=0.71, adj=0.016, (0 split)  
##   
## Node number 13: 59 observations, complexity param=0.02222222  
## predicted class=1 expected loss=0.3728814 P(node) =0.09833333  
## class counts: 22 37  
## probabilities: 0.373 0.627   
## left son=26 (16 obs) right son=27 (43 obs)  
## Primary splits:  
## purpose splits as RLRR-RR-LL, improve=2.790895, (0 missing)  
## installp < 3.5 to the left, improve=2.680778, (0 missing)  
## employed splits as LRRLR, improve=1.670515, (0 missing)  
## savings splits as RLRLR, improve=1.242200, (0 missing)  
## resident < 1.5 to the left, improve=1.176554, (0 missing)  
## Surrogate splits:  
## employed splits as LRRRR, agree=0.780, adj=0.188, (0 split)  
## martial splits as LRRR, agree=0.763, adj=0.125, (0 split)  
## job splits as LRRR, agree=0.746, adj=0.063, (0 split)  
## depends < 1.5 to the right, agree=0.746, adj=0.063, (0 split)  
##   
## Node number 24: 151 observations  
## predicted class=0 expected loss=0.2384106 P(node) =0.2516667  
## class counts: 115 36  
## probabilities: 0.762 0.238   
##   
## Node number 25: 63 observations, complexity param=0.03611111  
## predicted class=1 expected loss=0.4761905 P(node) =0.105  
## class counts: 30 33  
## probabilities: 0.476 0.524   
## left son=50 (16 obs) right son=51 (47 obs)  
## Primary splits:  
## duration < 11.5 to the left, improve=4.851444, (0 missing)  
## history splits as --RRL, improve=2.456704, (0 missing)  
## age < 25.5 to the right, improve=1.973187, (0 missing)  
## existcr < 1.5 to the right, improve=1.734558, (0 missing)  
## savings splits as RL-LR, improve=1.680320, (0 missing)  
## Surrogate splits:  
## foreign splits as RL, agree=0.794, adj=0.187, (0 split)  
##   
## Node number 26: 16 observations  
## predicted class=0 expected loss=0.375 P(node) =0.02666667  
## class counts: 10 6  
## probabilities: 0.625 0.375   
##   
## Node number 27: 43 observations  
## predicted class=1 expected loss=0.2790698 P(node) =0.07166667  
## class counts: 12 31  
## probabilities: 0.279 0.721   
##   
## Node number 50: 16 observations  
## predicted class=0 expected loss=0.1875 P(node) =0.02666667  
## class counts: 13 3  
## probabilities: 0.812 0.187   
##   
## Node number 51: 47 observations, complexity param=0.01944444  
## predicted class=1 expected loss=0.3617021 P(node) =0.07833333  
## class counts: 17 30  
## probabilities: 0.362 0.638   
## left son=102 (22 obs) right son=103 (25 obs)  
## Primary splits:  
## martial splits as RRLR, improve=2.793037, (0 missing)  
## employed splits as LLLLR, improve=1.739965, (0 missing)  
## age < 25.5 to the right, improve=1.739965, (0 missing)  
## installp < 2.5 to the left, improve=1.728443, (0 missing)  
## resident < 2.5 to the right, improve=1.472289, (0 missing)  
## Surrogate splits:  
## age < 29.5 to the right, agree=0.723, adj=0.409, (0 split)  
## history splits as --RRL, agree=0.702, adj=0.364, (0 split)  
## employed splits as LRRLL, agree=0.681, adj=0.318, (0 split)  
## depends < 1.5 to the right, agree=0.638, adj=0.227, (0 split)  
## amount splits as RLR, agree=0.617, adj=0.182, (0 split)  
##   
## Node number 102: 22 observations, complexity param=0.01944444  
## predicted class=0 expected loss=0.4545455 P(node) =0.03666667  
## class counts: 12 10  
## probabilities: 0.545 0.455   
## left son=204 (15 obs) right son=205 (7 obs)  
## Primary splits:  
## employed splits as LLLLR, improve=3.3281390, (0 missing)  
## installp < 2.5 to the left, improve=2.3757580, (0 missing)  
## checking splits as RL--, improve=0.8757576, (0 missing)  
## age < 39.5 to the left, improve=0.7757576, (0 missing)  
## property splits as RRLL, improve=0.4475524, (0 missing)  
## Surrogate splits:  
## resident < 3.5 to the left, agree=0.864, adj=0.571, (0 split)  
## coapp splits as LRR, agree=0.818, adj=0.429, (0 split)  
## age < 39.5 to the left, agree=0.773, adj=0.286, (0 split)  
## housing splits as RLL, agree=0.773, adj=0.286, (0 split)  
## history splits as --LLR, agree=0.727, adj=0.143, (0 split)  
##   
## Node number 103: 25 observations  
## predicted class=1 expected loss=0.2 P(node) =0.04166667  
## class counts: 5 20  
## probabilities: 0.200 0.800   
##   
## Node number 204: 15 observations  
## predicted class=0 expected loss=0.2666667 P(node) =0.025  
## class counts: 11 4  
## probabilities: 0.733 0.267   
##   
## Node number 205: 7 observations  
## predicted class=1 expected loss=0.1428571 P(node) =0.01166667  
## class counts: 1 6  
## probabilities: 0.143 0.857

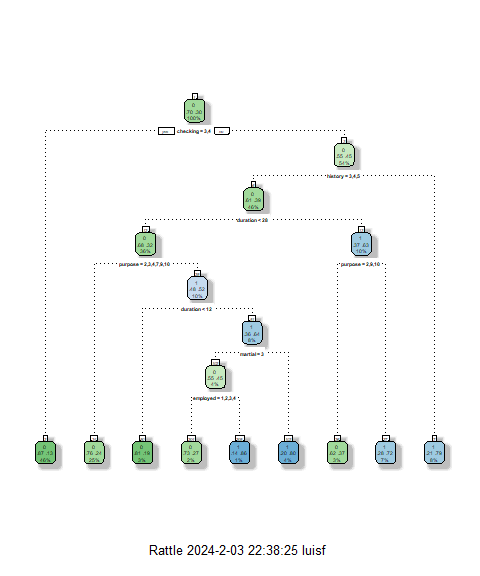
#패키지에서 제공하는 짜증나는 그림  
plot(fit1)  
text(fit1, use.n=TRUE)



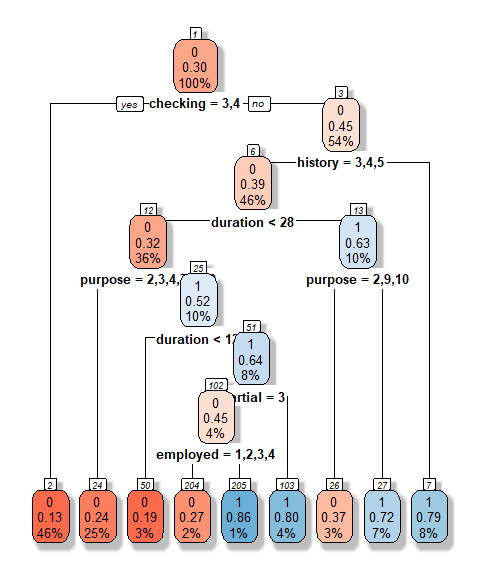
# 조금 이쁜 그림  
library(rattle)

library(rpart.plot)

library(RColorBrewer) # Color selection  
fancyRpartPlot(fit1)



#괜찮은 그림  
library(rpart.plot)  
rpart.plot(  
 fit1,  
 box.palette = "RdBu",  
 shadow.col = "gray",  
 cex = 0.8,  
 nn = TRUE  
)



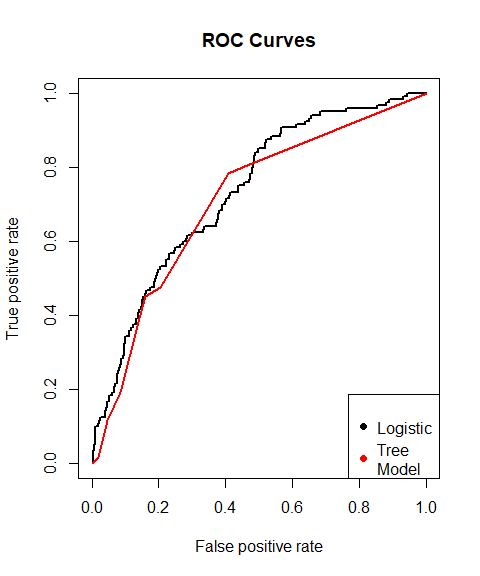
#score test data  
#logistic regression ROC curve  
pred = prediction(test$phat, test$good.bad)  
perf = performance(pred, "tpr", "fpr")  
performance(pred, "auc")@y.values[[1]]

## [1] 0.7301637

#tree ROC Curve  
test$phat1 = predict(fit1, test)[, 2]  
pred1 = prediction(test$phat1, test$good.bad)  
perf1 = performance(pred1, "tpr", "fpr")  
performance(pred1, "auc")@y.values[[1]]

## [1] 0.709122

plot(perf,  
 col = 'black',  
 lwd = 2,  
 main = 'ROC Curves')  
plot(perf1,  
 col = 'red',  
 lwd = 2,  
 add = TRUE)  
legend(  
 'bottomright',  
 legend = c('Logistic', 'Tree  
Model'),  
pch = 19,  
col = c('black', 'red')  
)



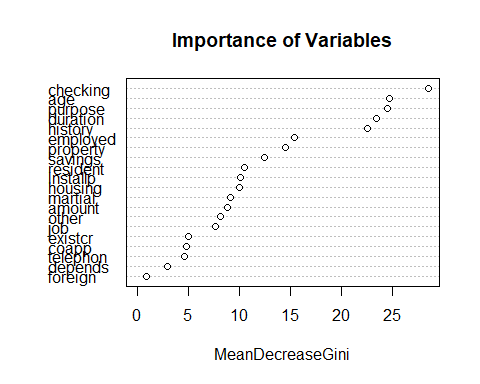
# RANDOM Forest

library(randomForest)

set.seed(123457)  
fit2=randomForest(good.bad~., train)  
fit2

##   
## Call:  
## randomForest(formula = good.bad ~ ., data = train)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 27%  
## Confusion matrix:  
## 0 1 class.error  
## 0 378 42 0.1000000  
## 1 120 60 0.6666667

varImpPlot(fit2, main="Importance of Variables")



Impor=importance(fit2)  
Impor[order(Impor, decreasing=TRUE)[1:5],]

## checking age purpose duration history   
## 28.54292 24.75003 24.55512 23.47530 22.60195

test$phat2=predict(fit2, type='prob',test)[,2]  
pred2=prediction(test$phat2,test$good.bad)  
perf2 =performance(pred2,"tpr","fpr")  
performance(pred,"auc")@y.values[[1]] #logistic regression

## [1] 0.7301637

performance(pred1,"auc")@y.values[[1]] #tree

## [1] 0.709122

performance(pred2,"auc")@y.values[[1]] #random forest

## [1] 0.7369196

plot(perf, col='black', lwd=2, main='ROC Curves') #logistic regression  
plot(perf1,col='red', lwd=2, add=TRUE) #tree  
plot(perf2,col='blue', lwd=2, add=TRUE) #random forest  
legend(  
 'bottomright',  
 inset = 0.1,  
 legend = c('Logistic', 'Tree Model', 'Random  
Forest'),  
lty = 1,  
col = c('black', 'red', 'blue')  
)

