빅데이터 제조 교재 : R 을 중심으로

- 필요 패키지 : ggplot2, psych, corrplot, MASS, car, leaps, ROCR, tree, monmlp, randomForest, RSNNS, glmnet, caret
- 필요 데이터 : autoparts.csv

학습목표

자동차 부품의 생산데이터로서 공정변수들과 함께 생산된 oil gasket 의 탕구 두께가 추출되어 기록된 자료이다.
먼저 기본적인 데이터 탐색을 통해 전체 자료 및 각 변수별 특성을 파악하는 방법을 학습하고, 이를 바탕으로 생산품 품질에 주된 영향을 미치는 공정변수 및 그 연관성 등을 파악하여 본다.
또한, 불량품 탐지를 위한 모형들을 구성하여 보고 그 성능을 비교한다.

여러 제품들 중 제품번호 45231-3B400 에 해당하는 oil gasket 에 대한 분석을 실시한다.
다른 제품군에 대해서도 비슷한 프로세스를 통해 분석이 가능하다.

본 분석에서는 탕구두께에 영향을 줄 것으로 생각되는 공정변수로 고정시간(fix_time), a 속도(a_speed), b 속도(b_speed), 실압력(mpa), 하중시간(load_time), 고압시간(highpressure_time) 등 6 개의 변수를 고려하였다.

탕구두께가 [21,27] 구간을 벗어나면 불량을 의심할 수 있는 것으로 보았다. 전체 자료의 약 11% 정도가 이에 해당한다.

A. 자료의 요약

변수의 개수 및 종류등 자료의 기본적인 특성을 파악한다.

요약통계량 출력 및 기초적인 시각화를 통해 각 변수별 특성 및 변수간의 연관성을 파악한다.

```
car <- read.csv('./Data/autoparts.csv', header=T, fileEncoding="UTF-8")
head(car)
```

```
##                 prod_date     prod_no  prod_name degree     mold prod
## 1 2014-05-01 오전 8:28:56 90784-76001 Oil Gasket      2 생산대기 생산
## 2 2014-05-01 오전 8:27:29 90784-76001 Oil Gasket      2 생산대기 생산
## 3 2014-05-01 오전 8:26:04 90784-76001 Oil Gasket      2 생산대기 생산
## 4 2014-05-01 오전 8:24:37 90784-76001 Oil Gasket      2 생산대기 생산
## 5 2014-05-01 오전 8:23:11 90784-76001 Oil Gasket      2 생산대기 생산
## 6 2014-05-01 오전 8:21:46 90784-76001 Oil Gasket      2 생산대기 생산
##     s_no fix_time a_speed b_speed separation s_separation rate_terms  mpa
## 1 892890     85.5   0.611   1.715      242.0        657.6         95 78.2
## 2 892889     86.2   0.606   1.708      244.7        657.1         95 77.9
## 3 892888     86.0   0.609   1.715      242.7        657.5         95 78.0
## 4 892887     86.1   0.610   1.718      241.9        657.3         95 78.2
## 5 892886     86.1   0.603   1.704      242.5        657.3         95 77.9
## 6 892885     86.3   0.606   1.707      244.5        656.9         95 77.9
##   load_time highpressure_time c_thickness
## 1      18.1                58        24.7
## 2      18.2                58        22.5
## 3      18.1                82        24.1
## 4      18.1                74        25.1
## 5      18.2                56        24.5
## 6      18.0                78        22.9
```

```
str(car)
```

```
## 'data.frame':    34139 obs. of  17 variables:
##  $ prod_date        : Factor w/ 34139 levels "2014-03-01 오전 10:00:33",..:
 34139 34138 34137 34136 34135 34134 34133 34132 34131 34130 ...
##  $ prod_no          : Factor w/ 6 levels "45231-3B400",..: 6 6 6 6 6 6 6 6
 6 6 ...
##  $ prod_name        : Factor w/ 1 level "Oil Gasket": 1 1 1 1 1 1 1 1 1 1
 ...
##  $ degree           : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ mold             : Factor w/ 3 levels "생산대기","승인대기",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ prod             : Factor w/ 1 level "생산": 1 1 1 1 1 1 1 1 1 1 ...
##  $ s_no             : int  892890 892889 892888 892887 892886 892885 89288
```

```
4 892883 892882 892881 ...
##  $ fix_time         : num  85.5 86.2 86 86.1 86.1 86.3 86.5 86.4 86.3 86
 ...
##  $ a_speed          : num  0.611 0.606 0.609 0.61 0.603 0.606 0.606 0.607
0.604 0.608 ...
##  $ b_speed          : num  1.72 1.71 1.72 1.72 1.7 ...
##  $ separation       : num  242 245 243 242 242 ...
##  $ s_separation     : num  658 657 658 657 657 ...
##  $ rate_terms       : int  95 95 95 95 95 95 95 95 95 95 ...
##  $ mpa              : num  78.2 77.9 78 78.2 77.9 77.9 78.2 77.5 77.8 77.5
 ...
##  $ load_time        : num  18.1 18.2 18.1 18.1 18.2 18 18.1 18.1 18 18.1
 ...
##  $ highpressure_time: int  58 58 82 74 56 78 55 57 50 60 ...
##  $ c_thickness      : num  24.7 22.5 24.1 25.1 24.5 22.9 24.3 23.9 22.2 19
 ...

car <- car[car$prod_no == "45231-3B400",]
car <- car[(car$c_thickness > 10)*(car$c_thickness < 50) ==1,]
```

- `read.csv()` 함수를 이용하여 csv 형태의 외부 데이터 화일을 읽어들임.

- `head()`, `str()` 함수를 이용하여 데이터의 기본적인 형태 (변수의 개수 및 읽어들인 object 의 특성 등)를 파악.

- 탕구두께(c_thickness)중 자료의 범위를 크게 벗어나는 점들(10 미만, 50 초과)이 탐지됨.
  - 이는 명백한 불량으로 볼 수 있으며 불량이 나타나는 프로세스 분석시 의미있는 정보를 제공할 수도 있으나,
  - 통계적 모형화를 실시할 때 지나치게 큰 영향력을 발휘할 수 있다고 판단되므로 제거하고 분석을 실시함.

현재 `car` 에는 17 개의 변수와 총 34139 개의 관측치가 존재함을 확인할 수 있다.
변수들 중 oil gasket 의 탕구 두께와 관계되는 변수들의 특성을 파악하여 본다.

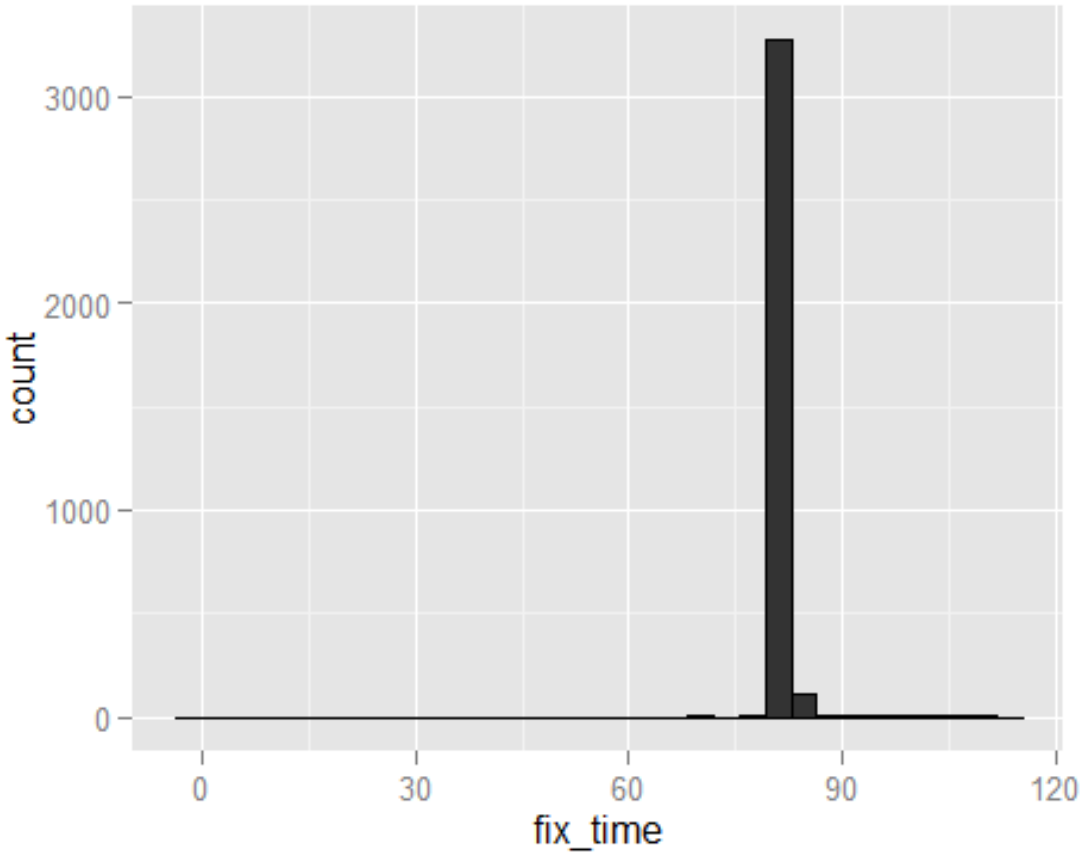
A.1 변수들의 분포 파악

변수들의 분포의 특성을 파악하기 위해서는 기초통계량을 출력하여 볼 수도 있으나, 히스토그램이나 상자그림등을 활용하여 시각화하는 것이 매우 효과적일 때가 많다.
탕구 두께 및 그에 영향을 미칠 것으로 생각되는 변수들에 대하여 시각화를 실행하여 보자.

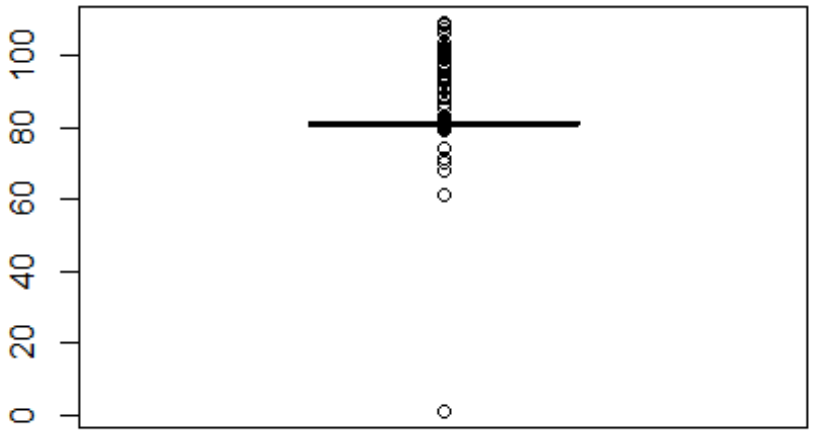
```
library(ggplot2)
library(psych)
```

- **ggplot** 은 데이터 시각화를 위해 최근 가장 폭넓게 사용되는 패키지임.
- **psych** 은 자료의 요약을 위한 패키지 중 하나임. `describe()` 함수의 사용을 위해 로딩함.

```
ggplot(car, aes(x=fix_time)) + geom_histogram(, colour = "black")
```



```r
boxplot(car$fix_time)
```

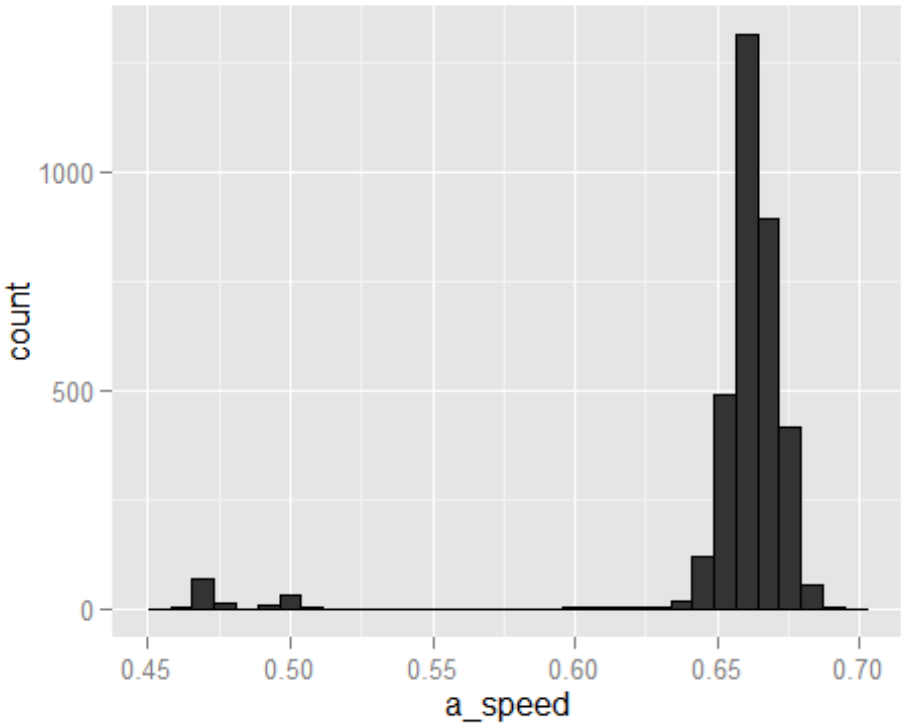


- ggplot() 함수는 데이터 시각화를 위한 함수이며, geom_histogram()과 함께 양적변수에 대한 히스토그램을 출력한다.

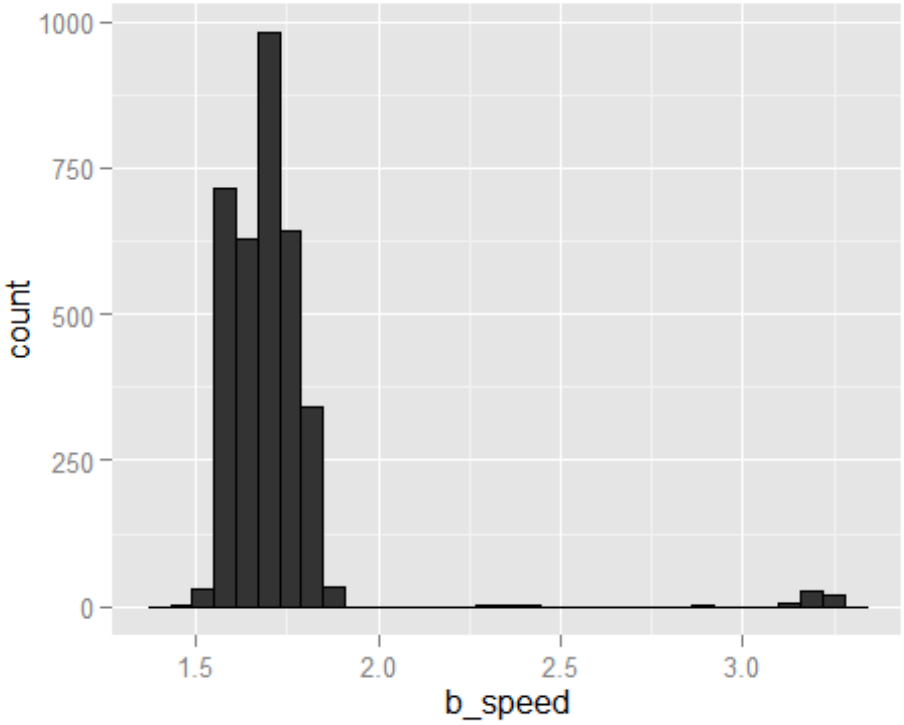
고정시간(fix_time)에 대한 히스토그램을 보면, 대부분이 80~90 사이에 몰려 있음을 알 수 있다.
하지만 일부 70 근처에서 형성되고 있는 값도 존재하며 히스토그램 상으로는 잘 보이지 않지만 아래 상자그림을 통해서 0 에 매우 가까운 값도 존재함을 볼 수 있다.

변수 분포의 중심에서 크게 벗어나는 자료들은 불량을 일으킬 가능성이 클 것으로 추측할 수 있다.
또한, 0 에 매우 가까운 값은 비정상적인 것으로 보인다.
이러한 값들이 공정과정에 나타날 경우, 완제품이 나오기 전에 미리 경고신호를 부여할 수도 있을 것이다.
단, 본 분석과정에서는 공정변수(탕구두께를 제외한 6 개의 변수)들이 탕구두께 혹은 불량률에 미치는 영향을 모형을 통해 파악하고자 하므로 이러한 값들을 제거하지 않고 포함시킨 채로 분석을 시행하도록 한다.

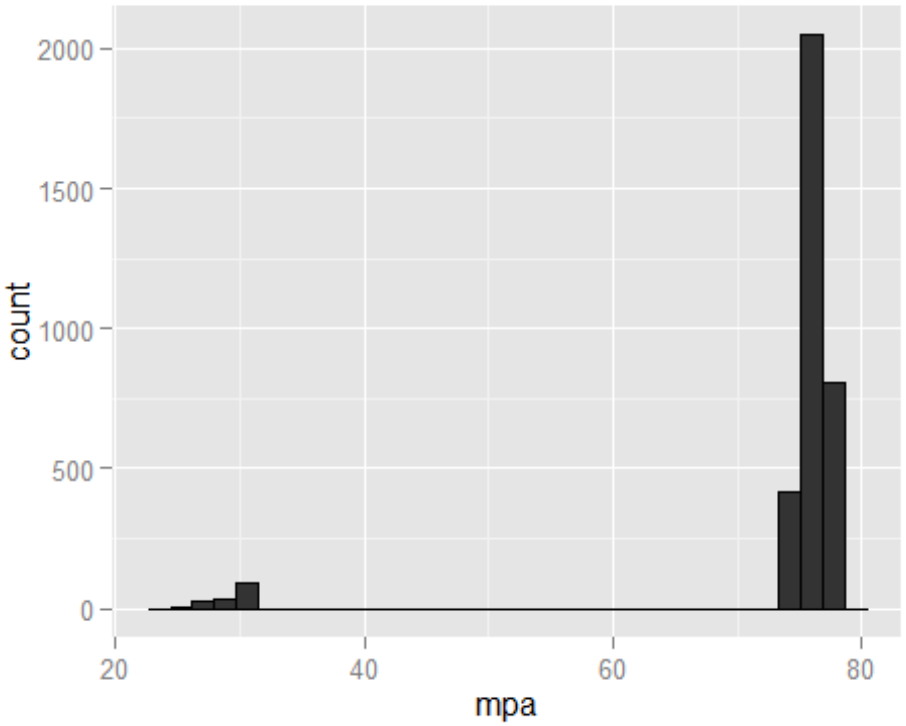
```
ggplot(car, aes(x=a_speed)) + geom_histogram(, colour = "black")
```



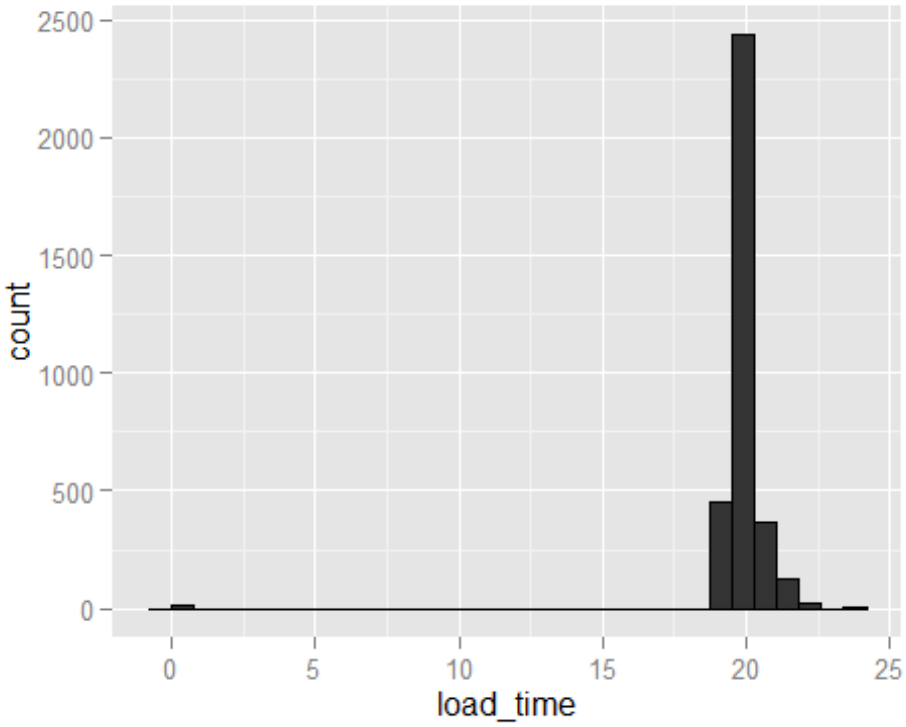

```
ggplot(car, aes(x=b_speed)) + geom_histogram(, colour = "black")
```



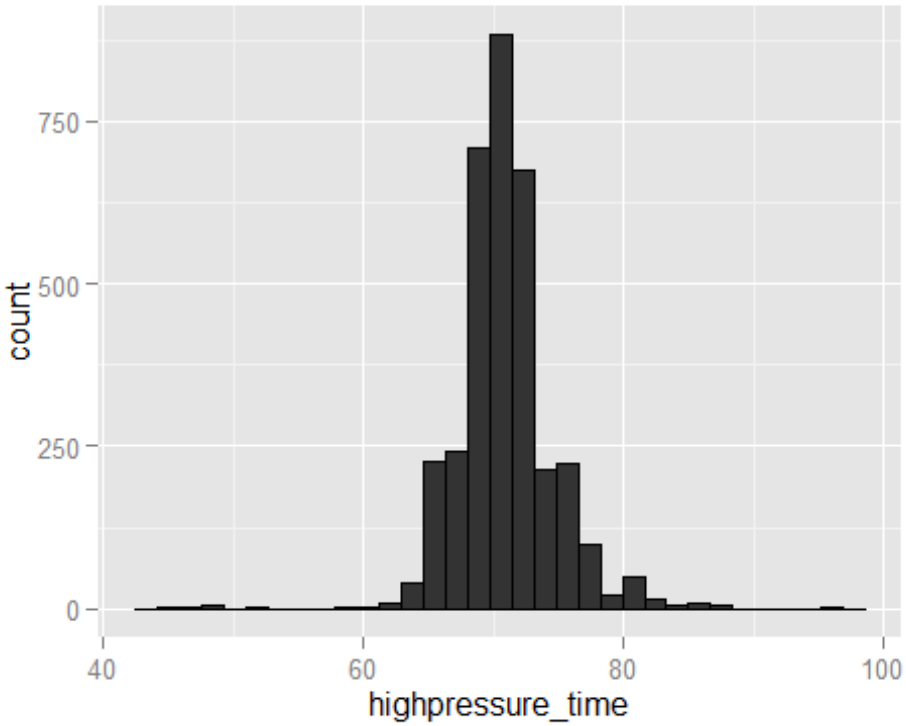
```
ggplot(car, aes(x=mpa)) + geom_histogram(, colour = "black")
```




```
ggplot(car, aes(x=load_time)) + geom_histogram(, colour = "black")
```



```
ggplot(car, aes(x=highpressure_time)) + geom_histogram(, colour = "black")
```



a 속도, b 속도, 실압력, 하중시간, 고압시간 등도 값이 특정구간에 집중되는 경향이 있으나 일부 자료들은 분포의 중심에서 떨어져 있다.
특히 대부분 공정변수의 값들이 크게 두 그룹으로 분리되어 있는 것처럼 보인다.

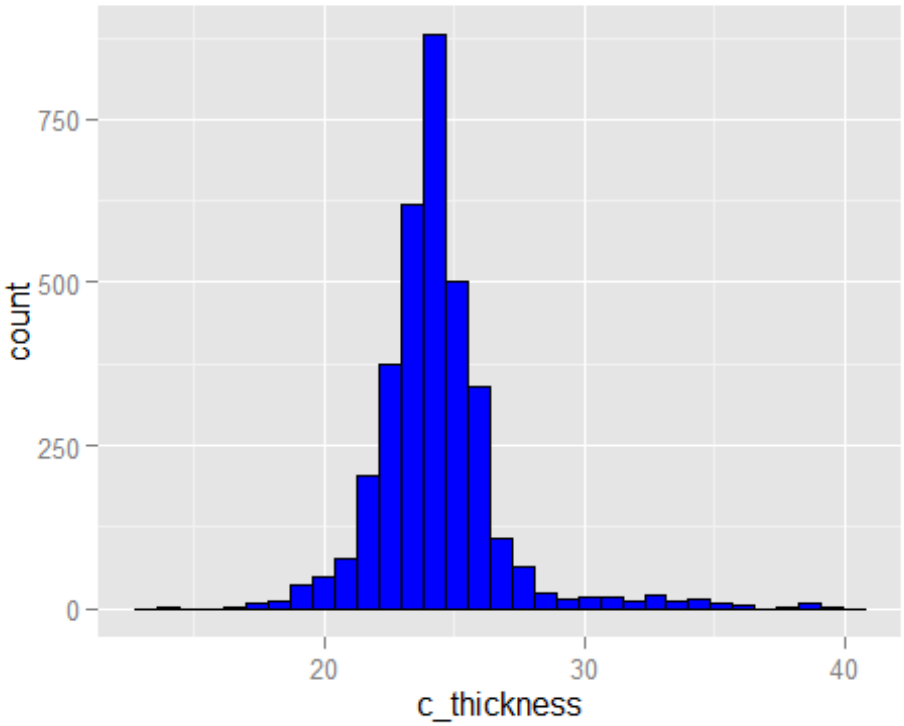
두 그룹 중 소수그룹이 불량을 일으키는 주된 원인이 될 것으로 추측해볼 수 있다.

원자료를 통해 파악해 보면, 특정 공정변수의 값이 다수그룹에 속해 있어도 탕구두께의 값이 매우 크거나 작은 경우가 있으며, 반대로 공정변수의 값이 다소 비정상적으로 보이더라도 완성된 oil gasket 의 탕구두께는 정상범위에 속하는 경우도 있다.

```
describe(car$c_thickness)
```

```
##    vars    n  mean   sd median trimmed  mad min  max range skew kurtosis
## 1     1 3432 24.24 2.39   24.1   24.08 1.48  14 39.5  25.5 1.71      7.6
##     se
## 1 0.04
```

```
ggplot(car, aes(x=c_thickness)) + geom_histogram(, fill="blue", colour = "bla
ck")
```



- `describe()` 함수는 자료의 개수, 중위수, 적률값 등 기초적인 요약통계량을 제공해 줌.

탕구두께에 대한 요약통계량 및 히스토그램을 출력하였다.
평균수준이 24.24 로 유지되고 있으며, 히스토그램에서 비교적 두꺼운 제품이 많이 발생하는 것을 볼 수 있다.
왜도(skewness) 또한 1.71 로 양수의 값을 가지므로 분포가 대칭이 아니라 오른쪽으로 약간 긴 꼬리를 가지는 것으로 생각할 수 있다.
즉, 불량 발생시 두께가 얇은 쪽 보다는 두꺼운 쪽으로 발생할 가능성이 높다고 여겨진다.

A.2 공정변수의 영향력 분석 - 각 공정변수의 수준에 따른 탕구두께의 분포

공정변수들이 탕구두께에 미치는 영향을 파악하기 위해 간단한 시각화를 통해 확인하여 본다.

변수들은 모두 연속형변수로 볼 수 있으며, 각 변수들을 임의로 3 개의 구간(low,middle,high = 1,2,3)으로 나누어 각 구간별로 탕구두께에 대한 상자그림을 출력한다.
이 때, 구간은 나누는 기준은 각 구간에 일정 비율 이상의 자료들이 포함된다는 원칙에 의해 결정되었다.
구간의 개수를 늘리거나 구간을 나누는 기준점을 변화시키며 그림을 그려보는 것도 좋다.

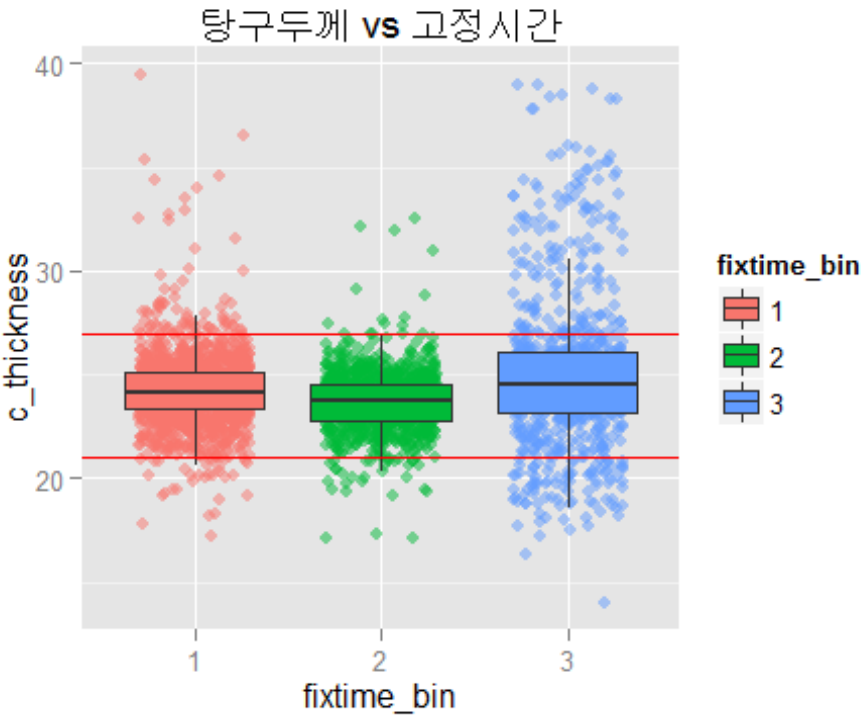
두께가 [21,27] 구간을 벗어나면 불량을 의심할 수 있다.

상자그림에 두께 21,27 에 해당하는 지시선(reference line)을 그어 확인한다.

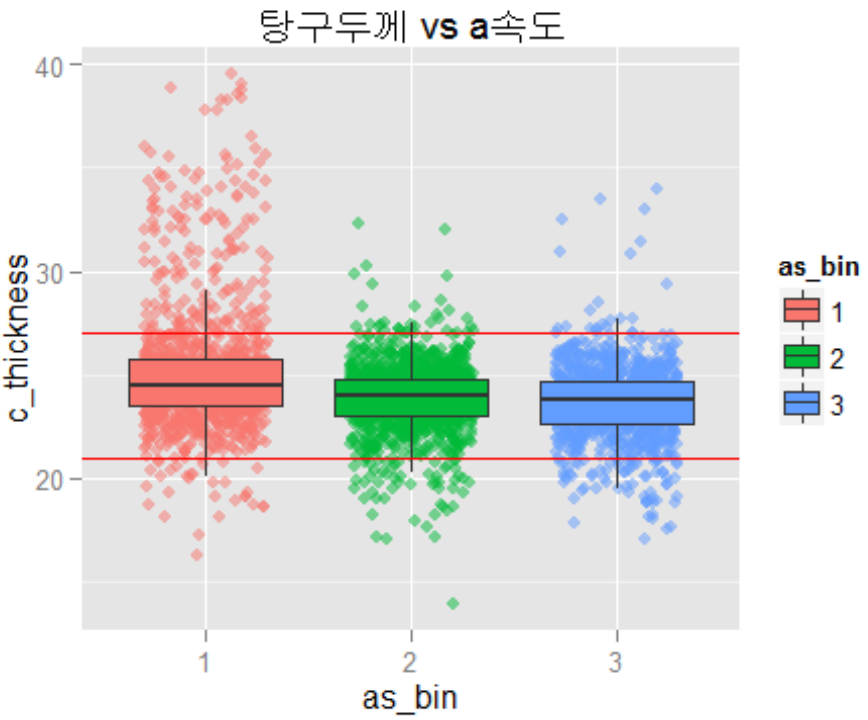
```
car1 <- car[,c(8,9,10,14,15,16,17)]
```

- 탕구두께및 공정변수 6 개만을 새로운 object 로 저장.

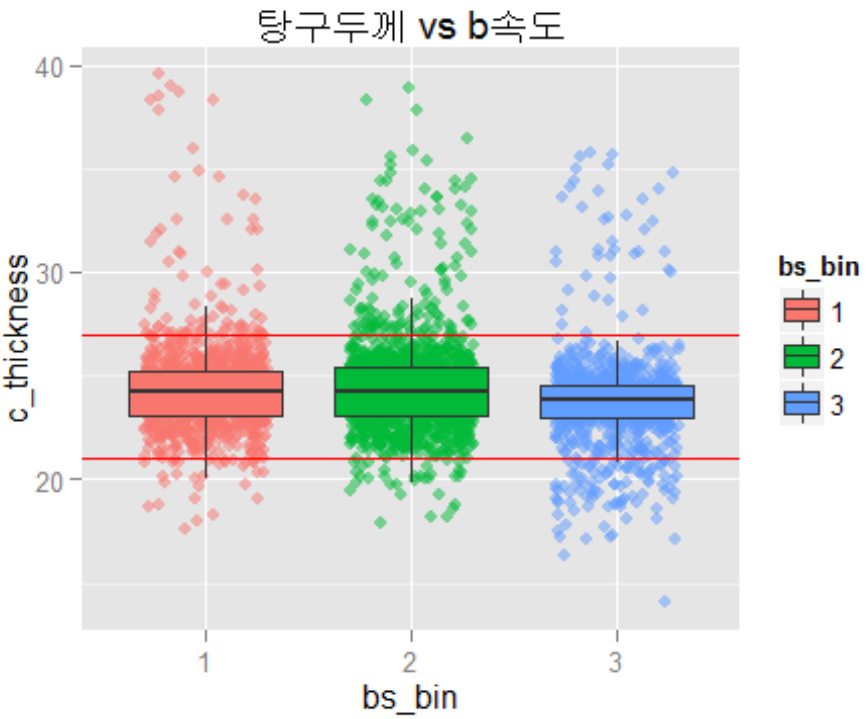
```
car1$fixtime_bin <- as.factor((car1$fix_time > 81.3) + (car1$fix_time > 81.1)
 + 1)
ggplot(car1, aes(x=fixtime_bin,c_thickness)) +
    geom_jitter(aes(colour = fixtime_bin), position = position_jitter(width =
 .3), alpha = 0.5) +
    geom_boxplot(aes(fill=fixtime_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red") + ggtitle("탕구두께 vs 고정시
간")
```



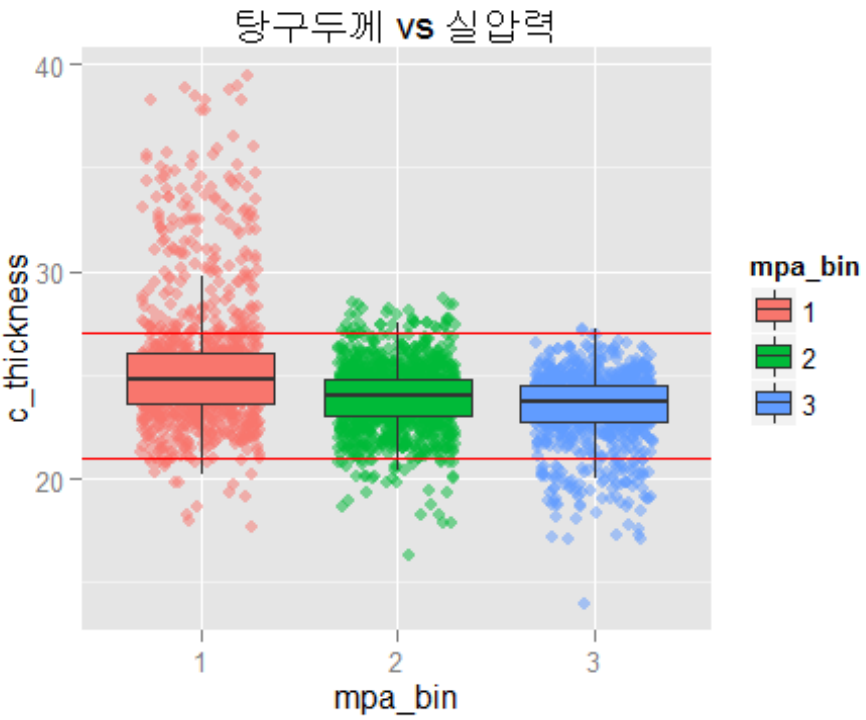

```
car1$as_bin <- as.factor((car1$a_speed > 0.659) + (car1$a_speed > 0.667) + 1)
ggplot(car1, aes(x=as_bin,c_thickness)) +
    geom_jitter(aes(colour = as_bin), position = position_jitter(width = .3),
 alpha = 0.5) +
    geom_boxplot(aes(fill=as_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red") + ggtitle("탕구두께 vs a 속도")
```



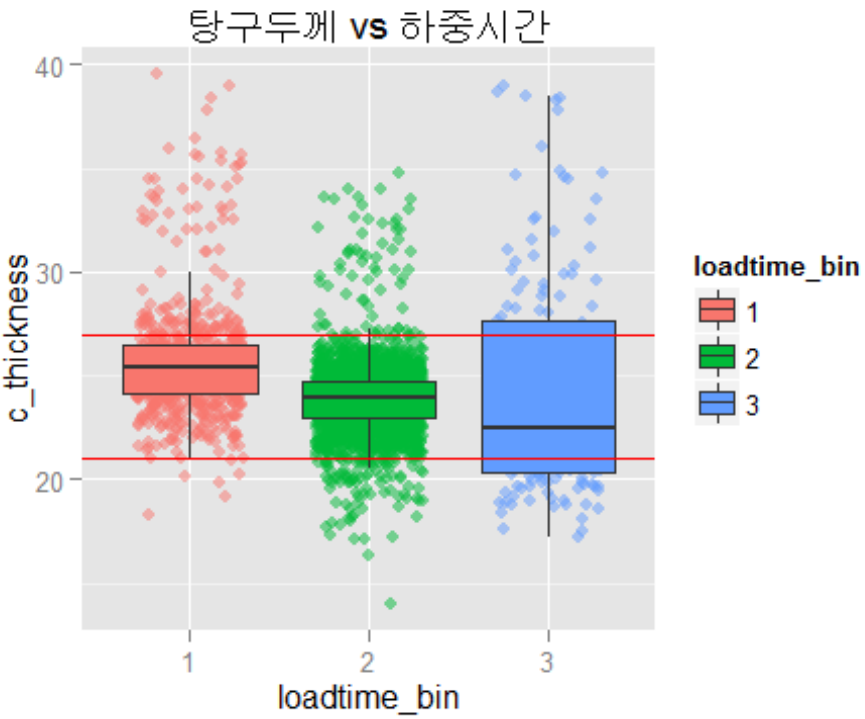

```
car1$bs_bin <- as.factor((car1$b_speed > 1.632) + (car1$b_speed > 1.732) + 1)
ggplot(car1, aes(x=bs_bin,c_thickness)) +
    geom_jitter(aes(colour = bs_bin), position = position_jitter(width = .3),
 alpha = 0.5) +
    geom_boxplot(aes(fill=bs_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red") + ggtitle("탕구두께 vs b 속도")
```



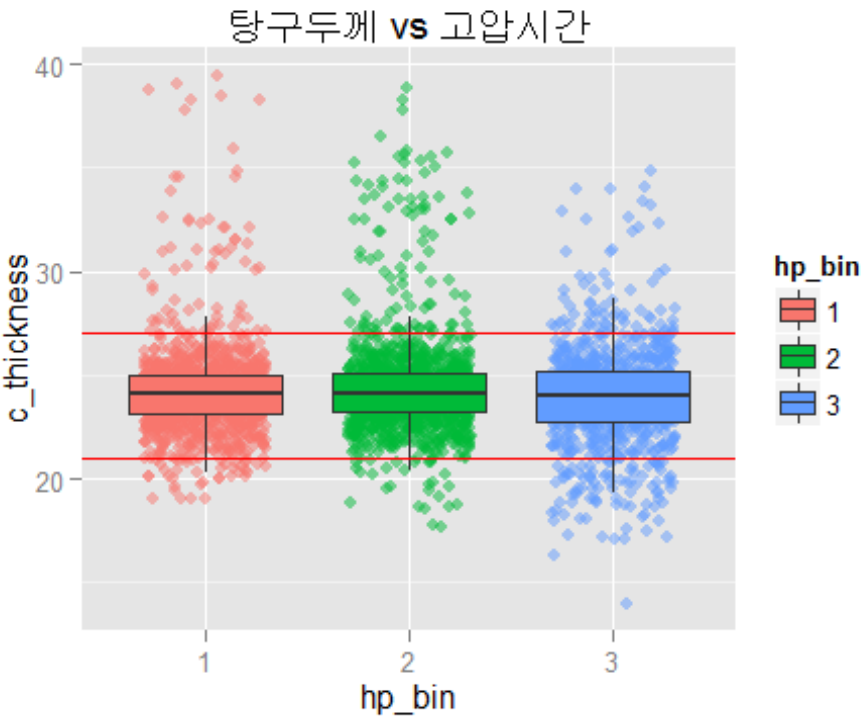

```
car1$mpa_bin <- as.factor((car1$mpa > 76.4) + (car1$mpa > 75.4) + 1)
ggplot(car1, aes(x=mpa_bin,c_thickness)) +
    geom_jitter(aes(colour = mpa_bin), position = position_jitter(width = .3),
 alpha = 0.5) +
    geom_boxplot(aes(fill=mpa_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red") + ggtitle("탕구두께 vs 실압력")
```




```
car1$loadtime_bin <- as.factor((car1$load_time > 20.1) + (car1$load_time > 20.
3) + 1)
ggplot(car1, aes(x=loadtime_bin,c_thickness)) +
    geom_jitter(aes(colour = loadtime_bin), position = position_jitter(width
= .3), alpha = 0.5) +
    geom_boxplot(aes(fill=loadtime_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red") + ggtitle("탕구두께 vs 하중시
간")
```




```
car1$hp_bin <- as.factor((car1$highpressure_time > 72) + (car1$highpressure_t
ime > 69) + 1)
ggplot(car1, aes(x=hp_bin,c_thickness)) +
    geom_jitter(aes(colour = hp_bin), position = position_jitter(width = .3),
 alpha = 0.5) +
    geom_boxplot(aes(fill=hp_bin),outlier.colour = NA) +
    geom_hline(yintercept=c(21,27),color="red")+ ggtitle("탕구두께 vs 고압시간
")
```



- `geom_boxplot()`은 상자그림을 출력해 주며, `geom_hline()`은 출력된 그림 위에 수평 지시선을 그어줌.

- `geom_jitter()`는 상자 바깥에 있는 점들을 겹치지 않게 흩뿌려서, 점들의 밀도를 보다 시각적으로 쉽게 파악할 수 있도록 해 주는 함수임.

- 이상점은 따로 구별하여 표시하지 않음. (`outlier.colour = NA`)

- `as.factor()`를 이용하여 각 변수들을 3 개의 범주를 가지는 범주형 변수로 변환.

- `ggtitle()`로 그림의 main title 을 설정.

모든 경우에 지시선을 벗어나는 점들이 다수 눈에 띄는 것으로 보아 불량으로 의심할 수 있는 제품들이 많다.

고정시간이 보통 수준보다 길면, 탕구두께의 변동도 커진다.

a 시간과 실압력의 경우 수준이 높아질 수록, 탕구두께가 전체적으로 얇아지는 경향성이 관측된다.

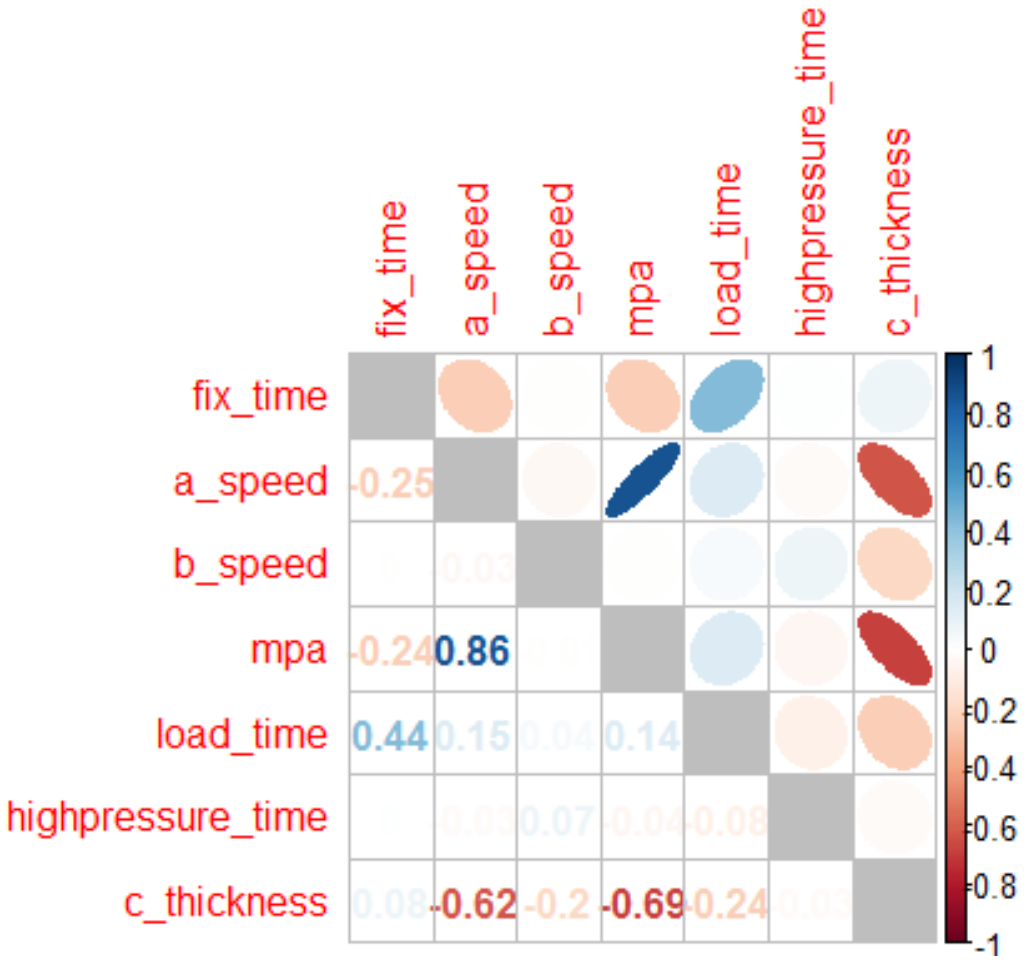
b 시간과 고압시간의 경우 뚜렷한 경향성은 관측되지 않는다.

본 상자그림에서 가장 눈에 띄는 현상은 하중시간에 따른 두께의 변화이다.
하중시간이 길수록 두께가 전반적으로 얇아지는 경향이 보이며, 특히 하중시간이 가장 높은 수준에 속할 때는 두께의 변동이 매우 크게 증가하는 것으로 보인다.

A.3 상관분석

탕구두께 및 공정변수들간의 상관계수를 계산하여 선형적 연관성의 정도를 파악한다.

```
car1 <- car[,c(8,9,10,14,15,16,17)]
library(corrplot)
corrplot.mixed(cor(car1),  upper = "ellipse" ,lower="number",tl.pos="lt",bg="
gray")
```



- **corrplot** 은 변수들간의 상관계수를 계산하여 다양한 형태의 그림으로 시각화 시킬 수 있는 패키지임.

- `corrplot.mixed()`를 이용하여 상관계수 뿐 아니라 상관계수의 크기를 타원의 모양과 색깔로 시각화.
  - 타원의 모양이 원에 가까울 수록 약한 상관, 직선의 형태에 가까울 수록 강한 상관관계를 나타내며, 색깔이 진할수록 상관관계가 강함.

- 그림의 위쪽은 타원의 모양 및 색깔의 진하기로 상관의 정도를 표현하고, 대각 아래쪽은 숫자로 표시함.

실압력과 a 속도간의 상관관계가 가장 강하게 나타남.

탕구두께와 가장 강한 선형적 연관성을 가지는 변수는 a 속도와 실압력으로 파악되며 둘 모두 음의 상관임. 즉, a 속도나 실압력이 커질 수록 탕구두께는 얇아지는 경향성이 있는 것으로 파악됨.

B. 선형회귀모형 (Linear model)

B.1 선형회귀모형의 적합 및 변수선택

변수들간에 어느정도 연관성이 존재하는 것으로 파악되므로 선형회귀모형을 이용하여 변수들간의 연관성 및 각 변수의 유의성을 살펴본다.

다중선형회귀모형에서 필수적인 변수선택의 절차를 수행한다.

```
lm1 <- lm(c_thickness~.,data=car1)
summary(lm1)
```

```
##
## Call:
## lm(formula = c_thickness ~ ., data = car1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6889 -0.8926  0.1050  0.9795  7.0233
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       55.134162   1.454324  37.910  < 2e-16 ***
## fix_time          -0.045671   0.014550  -3.139  0.00171 **
## a_speed           -9.398624   1.548467  -6.070 1.42e-09 ***
## b_speed           -2.320796   0.134285 -17.283  < 2e-16 ***
## mpa               -0.138710   0.005626 -24.656  < 2e-16 ***
## load_time         -0.226013   0.025295  -8.935  < 2e-16 ***
## highpressure_time -0.031567   0.007396  -4.268 2.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.618 on 3425 degrees of freedom
## Multiple R-squared:  0.5421, Adjusted R-squared:  0.5413
## F-statistic: 675.7 on 6 and 3425 DF,  p-value: < 2.2e-16
```

- `lm()`은 선형회귀모형을 적합하여 주는 함수.
  - 적합된 회귀모형을 새로운 object 로 저장후 summary 함수를 이용하면 적합된 모형에서의 회귀계수 추정치와 표준오차, 각 변수별 유의성 검정결과인 p-value 를 확인할 수 있음.

선형회귀모형 적합결과 모든 공정변수들이 유의수준 0.05 에서 유의하다.
즉, 모두 탕구두께에 유의미한 영향력을 가지고 있다 하겠다.

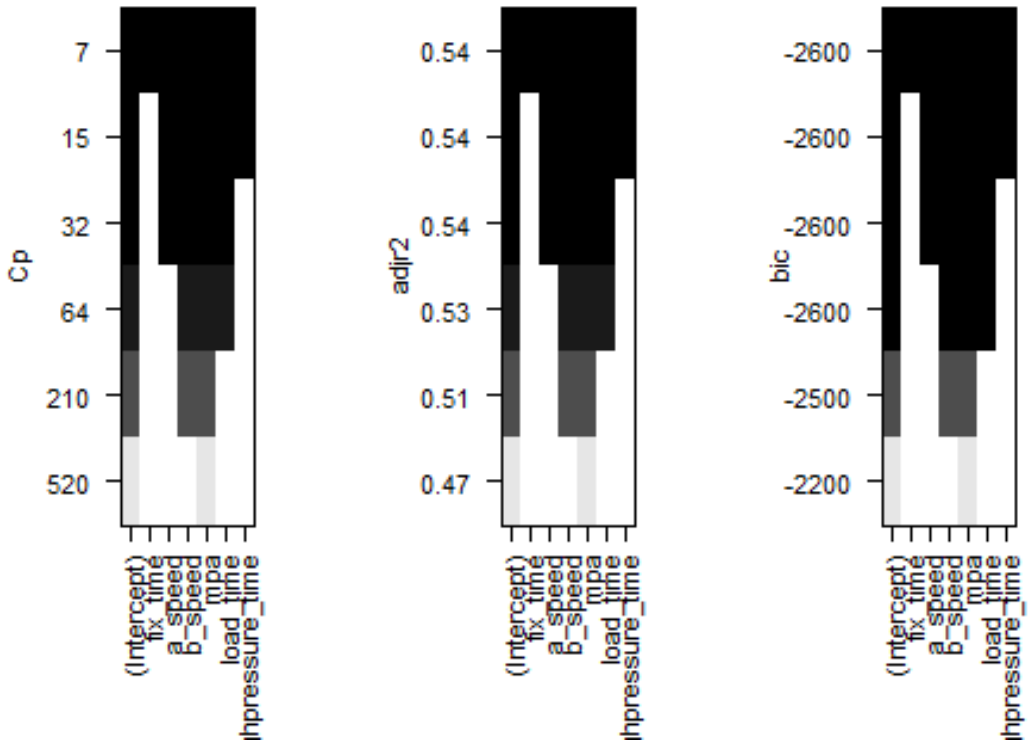
모든 변수들의 계수가 음수이므로, 공정변수들이 커질 때 탕구두께는 얇아지는 경향이 있음을 알 수 있다.

F 통계량 및 유의확률을 확인할 때, 모형 전체도 매우 유의함을 알 수 있다.

```
library(MASS)
library(leaps)
vs1 <- regsubsets(c_thickness~.,data=car1)
summary(vs1)
```

```
## Subset selection object
## Call: regsubsets.formula(c_thickness ~ ., data = car1)
## 6 Variables  (and intercept)
##                   Forced in Forced out
## fix_time              FALSE      FALSE
## a_speed               FALSE      FALSE
## b_speed               FALSE      FALSE
## mpa                   FALSE      FALSE
## load_time             FALSE      FALSE
## highpressure_time     FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: exhaustive
##          fix_time a_speed b_speed mpa load_time highpressure_time
## 1  ( 1 ) " "      " "     " "     "*" " "       " "
## 2  ( 1 ) " "      " "     "*"     "*" " "       " "
## 3  ( 1 ) " "      " "     "*"     "*" "*"       " "
## 4  ( 1 ) " "      "*"     "*"     "*" "*"       " "
## 5  ( 1 ) " "      "*"     "*"     "*" "*"       "*"
## 6  ( 1 ) "*"      "*"     "*"     "*" "*"       "*"
```

```
par(mfrow=c(1,3))
plot(vs1,scale="Cp")
plot(vs1,scale="adjr2")
plot(vs1,scale="bic")
```




```
vs2 <- stepAIC(lm1,direction="both")
```

```
## Start:  AIC=3311.89
## c_thickness ~ fix_time + a_speed + b_speed + mpa + load_time +
##     highpressure_time
##
##                      Df Sum of Sq     RSS    AIC
## <none>                             8971.6 3311.9
## - fix_time            1     25.81  8997.4 3319.7
## - highpressure_time   1     47.73  9019.3 3328.1
## - a_speed             1     96.50  9068.1 3346.6
## - load_time           1    209.12  9180.7 3389.0
## - b_speed             1    782.40  9754.0 3596.8
## - mpa                 1   1592.41 10564.0 3870.6
```

```
vs2$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## c_thickness ~ fix_time + a_speed + b_speed + mpa + load_time +
```

```
##     highpressure_time
##
## Final Model:
## c_thickness ~ fix_time + a_speed + b_speed + mpa + load_time +
##     highpressure_time
##
##
##   Step Df Deviance Resid. Df Resid. Dev      AIC
## 1                       3425   8971.614 3311.886
```

- **leaps** 는 regression subset selection 패키지임. 여기서는 `regsubsets()`함수를 위해 로딩.

- `regsubsets()` 함수는 best subset selection 을 수행하며, 모형의 크기(변수가 포함되는 개수) 순으로 가장 좋은 모형을 출력함. `plot()`함수에 의해 AIC, 수정결정계수, Cp 통계량 등을 기준으로 한 최적의 모형 및 각 변수의 포함여부를 시각화.

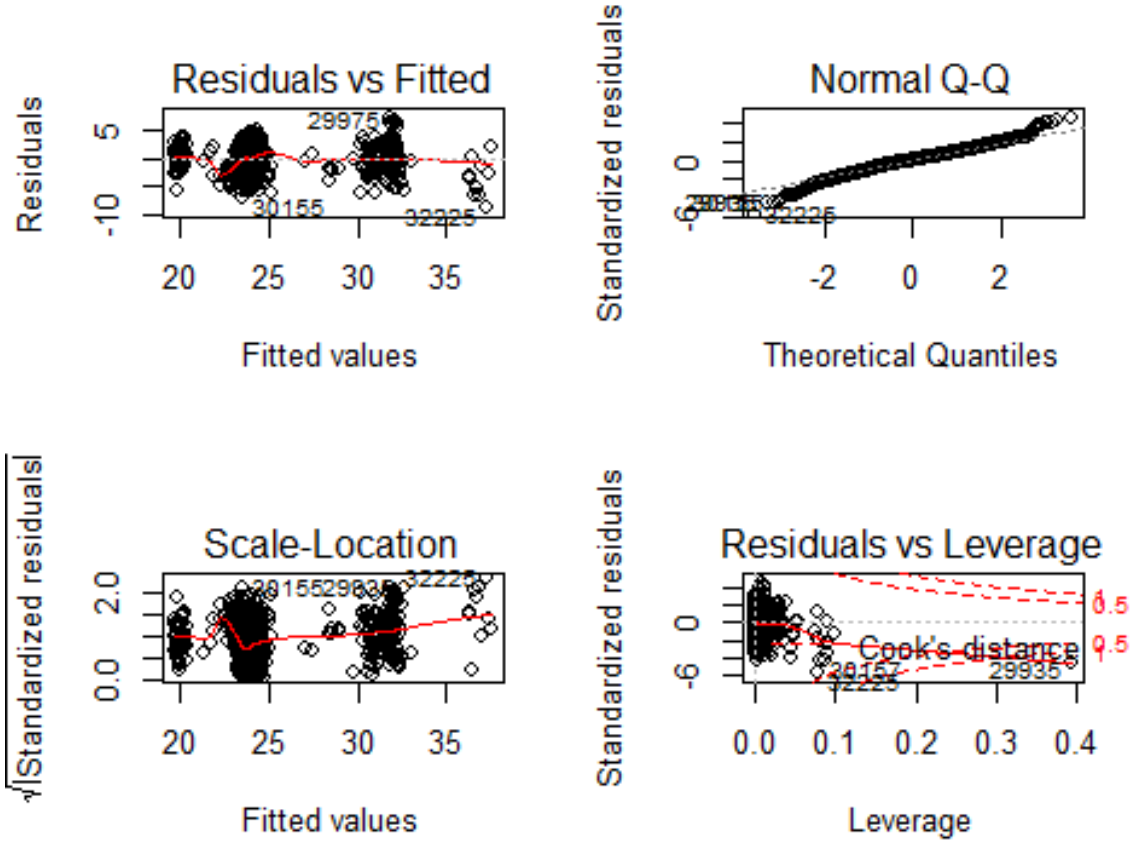
- `stepAIC()`는 AIC 를 기준으로 하여 stepwise selection 을 수행한 결과 선택된 최종모형을 확인할 수 있는 함수임. (**MASS** package 로드 필요)

변수 1 개만 모형에 포함하는 경우에는 실압력(mpa)이 선택됨.
즉, 가장 영향력이 큰 공정변수로 볼 수 있다.
두 개의 변수만 포함하는 경우에는 실압력 및 b 속도(b_speed)가 포함됨.
이는, 탕구두께와 개별적인 상관계수를 계산했을 때(실압력과 a 속도가 가장 큰 상관관계를 가짐) 와는 약간 다른 결과이다.

최적의 모형은 변수 6 개를 모두 고려하는 것임.
Cp, 수정결정계수, bic 를 기준으로 했을 때 모두 같은 결과를 준다.

`stepAIC()` 함수를 이용하여도 같은 모형이 선택된다.

```
library(car)
par(mfrow=c(2,2))
plot(lm1)
```




```
vif(lm1)
```

```
##          fix_time           a_speed           b_speed               mpa
##          1.423225          3.946969          1.011914          3.910896
##         load_time highpressure_time
##          1.371218          1.014465
```

- **car** 는 Companion to Applied Regression 의 약자로, 회귀분석과 연관된 다양한 함수를 제공하는 패키지임.
  - 여기서는, `vif()` 함수를 사용하기 위해서 로딩.

- `plot()`함수는 산점도 등을 그려주는 함수이나, 회귀모형이 적합된 object 에 적용하면 잔차도를 그려줌.

- `vif()`는 분산팽창계수를 출력해 주는 함수임.

잔차도를 보면, 오차항에 대한 정규성 가정이 의심되며, 영향점(influential points)도 존재하는 것으로 파악된다.
이는, 앞에서 이미 파악했듯이 공정변수들이 그룹화되는 경향이 있으며, 매우 극단적인 값을 가지는 경우가 다수 존재하기 때문으로 파악된다.

분산팽창계수가 전반적으로 4 를 넘지 않으므로 다중공선성이 심각하게 존재하지 않는 것으로 볼 수 있다.
(보통 분산팽창계수가 10 을 넘으면 심각한 것으로 보고 4~5 를 넘으면 의심해 볼 수 있는 것으로 알려져 있음.)

특히 모형을 통해 예측된 반응변수의 값에 몇 개의 군집이 형성되는 것처럼 보인다.
이러한 경우 선형회귀모형을 그 성능이 저하될 수 있다.

B.2 축소추정법

선형회귀모형에서 설명변수의 개수가 다수 존재하거나 변수들간의 상관성이 큰 경우 다중공선성 문제가 발생하며, 이는 추정량의 분산을 크게 하여 모형을 매우 불안정하게 만든다.

이 때, 축소추정법(shrinkage method)에 의해 모형의 성능을 개선할 있음이 알려져 있다.

LASSO 및 Ridge 추정량은 벌점화(penalization)에 기반한 축소추정량이며, LASSO 는 L1, Ridge 는 L2 type 벌점을 사용한다.
특히 LASSO 는 일부 추정량을 0 으로 만들어 추정과 변수선택을 동시에 할 수 있는 방법으로 알려져 있다.

본 단원에서는 위에서 적용했던 선형모형에 축수추정법을 적용해 볼 것이다.
이미 위 절에서 모든 변수들이 유의미한 것으로 판명되었고 다중공선성도 크게 의심할만 하지 않아 필수적인 절차라고는 할 수 없으나, 사실의 확인 및 연습을 위해 소개한다.

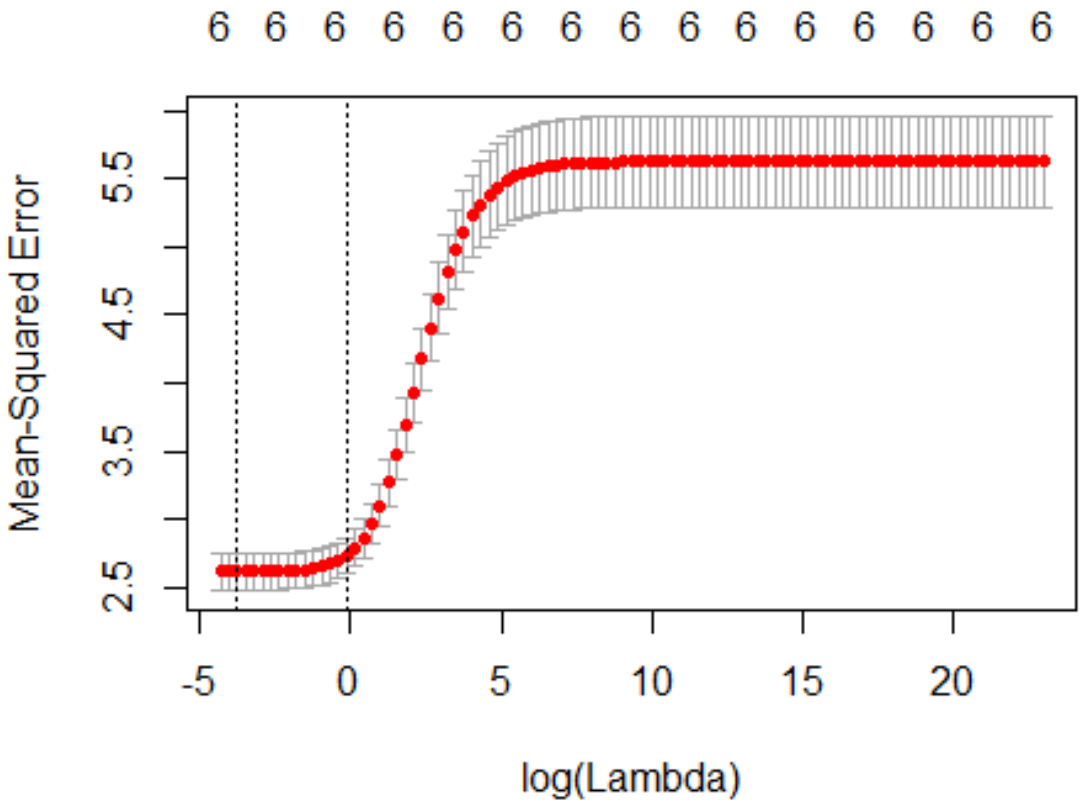
```
library(glmnet)
set.seed(1)
ind.train <- sample(1:nrow(car1),nrow(car1)*0.7)
car.train <- car1[ind.train,]
car.test <- car1[-ind.train,]
X <- as.matrix(car.train[,1:6])
Y <- as.matrix(car.train[,7])
nX <- as.matrix(car.test[,1:6])
nY <- as.matrix(car.test[,7])
```

- **glmnet** 은 축소추정에 의한 선형모형을 적합할 수 있는 패키지임.
- `set.seed()`는 모의실험의 동일 결과재현을 위해 seed number 를 설정하는 함수임.
- 자료를 7:3 의 비율로 훈련(training)자료와 검증(test)자료로 분할함.

- 훈련자료는 모형의 적합을 위해, 검증자료는 예측오차(prediction error)의 계산을 위해 사용됨.

B.2.1 능형(Ridge)회귀

```
cv.ridge <- cv.glmnet(X,Y,alpha=0,lambda=10^seq(10,-2,length=100))
plot(cv.ridge)
```




```
ridge.pred <- predict(glmnet(X,Y,alpha=0,lambda=cv.ridge$lambda.min),newx=nX)
mean((nY - ridge.pred)^2)
```

```
## [1] 2.730974
```

```
coef(glmnet(X,Y,alpha=0,lambda=cv.ridge$lambda.min))
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)       51.09135733
```

```
## fix_time           -0.01404767
## a_speed            -8.52058228
## b_speed            -2.13848411
## mpa                -0.13289647
## load_time          -0.25342045
## highpressure_time -0.02155629
```

- 축소추정을 위해서는 조절모수를 선택해야 하며, 이를 위해 교차검증(cross-validation)을 이용함. `cv.glmnet()`이 이러한 cv 를 수행해줌.

- Ridge 모형적합을 위해 `alpha=0` 으로 설정.

- `plot()`함수에 의해 조절모수(로그변환)에 따른 평균제곱오차를 시각화.

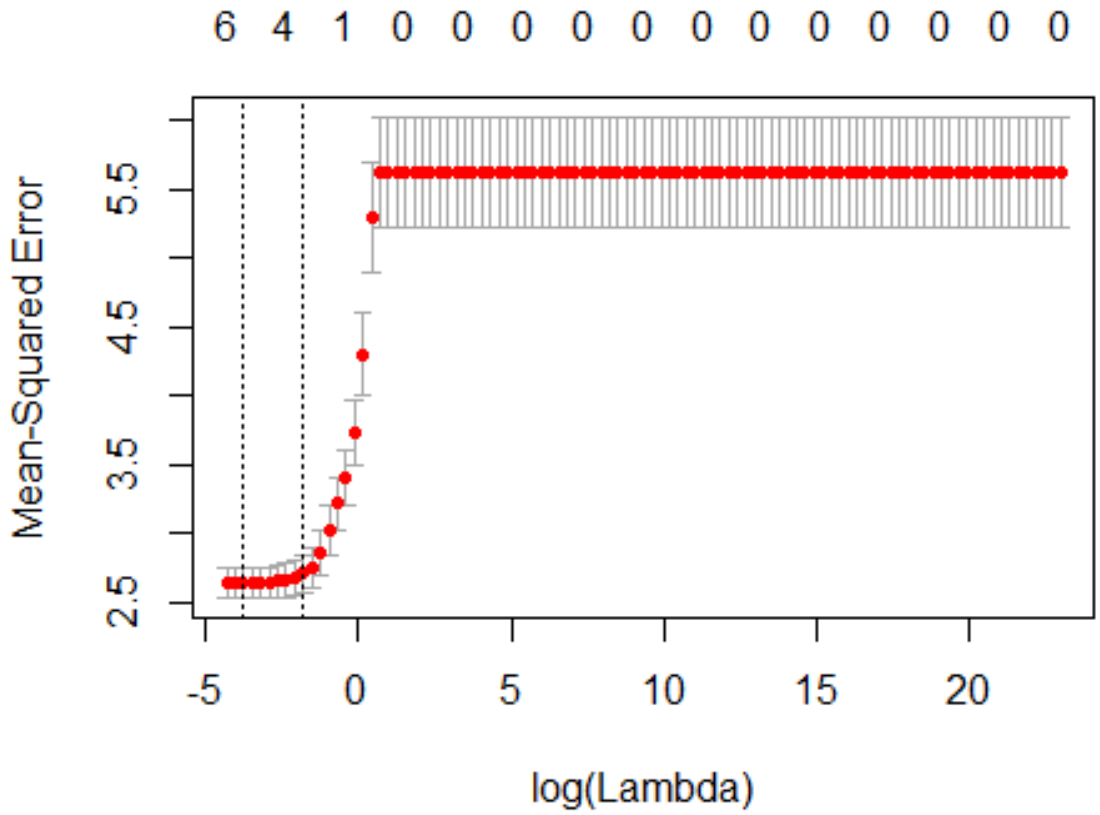
- cv 에 의해 선택된 조절모수값에 대한 모형을 적합하고 예측오차를 계산함.

그림에서 볼 수 있듯이 조절모수의 값은 작은 쪽에서 형성된다.
실제 선택된 값은 0.0534 이다.

예측오차는 2.7310 이다.

B.2.2 LASSO

```
cv.lasso <- cv.glmnet(X, Y, alpha=1, lambda=10^seq(10, -2, length=100))
plot(cv.lasso)
```




```
lasso.pred <- predict(glmnet(X,Y,alpha=1,lambda=cv.lasso$lambda.min),newx=nX)
mean((nY - lasso.pred)^2)

## [1] 2.75667

coef(glmnet(X,Y,alpha=1,lambda=cv.lasso$lambda.min))

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)       48.950408220
## fix_time          -0.003424949
## a_speed           -7.289834649
## b_speed           -2.053439660
## mpa               -0.135027127
## load_time         -0.250295531
## highpressure_time -0.015636951
```

- LASSO 모형적합을 위해 `alpha=1` 으로 설정.

- cv 에 의해 선택된 조절모수값에 대한 모형을 적합하고 예측오차를 계산함.

그림에서 볼 수 있듯이 조절모수의 값은 작은 쪽에서 형성된다.
실제 선택된 값은 0.01 로 현재 설정한 값들 중 최소에 해당하는 값이다.

추정된 값중 0 은 없다. 즉, 모형에서 빠지는 변수는 없다.

예측오차는 2.7567 이다.

```
lm.pred <- predict(lm(c_thickness~.,data=car.train),newdata=car.test[,1:6])
mean((car.test[,7] -lm.pred)^2)

## [1] 2.727557
```

- 선형회귀모형(축소추정법 X)에 의한 예측오차도 계산.

축소추정법을 적용하지 않은 선형회귀모형의 경우 예측오차가 2.7276 이다.
즉, Ridge(2.7310)나 LASSO(2.7567)에 비해 작은 값을 주었다.

앞에서 모두 변수가 유의하였던 점, 축소추정에 의해 변수가 모형에서 빠지지 않았던 점, 축소추정에 의해서 예측오차가 좋아지지 않았던 점, 분산팽창계수가 그리 크지 않았던 점 등을 종합하여 볼 때, 본 자료에서는 축소추정법이 필요하지 않은 것으로 보인다.

다만, 앞에서 보았듯이 선형모형 가정 자체에도 의심할 만한 여지가 있으며, 공정변수들이 두 그룹으로 완전히 분리되어 나타나는 경우가 많았던 점 등을 생각할 때 선형회귀 모형도 그 성능이 매우 뛰어나다고 보기는 어렵다.

C. 분류모형 (Classification) - 불량 vs 양품 분석

탕구두께가 [21,27] 구간을 벗어나는 것은 불량이 의심된다.

탕구두께가 [21,27] 구간에 속하는지 여부를 반응변수(불량품 or 양품)로 하여 다른 공정변수들이 어떤 영향을 미치는지 분석하여 볼 수 있다.

즉, 반응변수를 연속형이 아닌 이산형(binary)으로 변환하여 분류문제로 바꾸어 분석해 볼 수 있다.

일반화선형모형으로 로지스틱회귀모형을 고려할 수 있으며, 비선형 모형으로 의사결정나무 모형 및 신경망 모형등을 활용하여 분석해 보자.

여러 분류모형을 고려한 후, 오분류율, ROC curve, KS(kolmogorov-smirnov)통계량 등을 비교하여 모형의 성능 또한 비교해 볼 수 있다.

```
car1$failure <- as.factor((car1$c_thickness > 27)+(car1$c_thickness < 21))
car2 <- car1[,-7]
car2.train <- car2[ind.train,]
car2.test <- car2[-ind.train,]
```

- 위와 같이 불량이 의심되는 경우 1, 그렇지 않은 경우 0 의 값을 가지도록 변수를 재설정하고 기존 변수는 삭제.

- 모형의 성능평가를 위해 훈련자료와 검증자료로 분류

C.1 로지스틱(Logistic) 회귀모형

로지스틱 회귀모형은 선형모형의 확장된 형태이며, 본 예에서는 불량발생확률의 logit 변환을 공정변수들의 선형결합으로 모형화한 것이다.

선형회귀모형과 비슷한 함수들을 이용하여 적합 및 변수선택이 가능하다.

```
lm2 <- glm(failure~.,data=car2.train, family=binomial)
summary(lm2)
```

```
##
## Call:
## glm(formula = failure ~ ., family = binomial, data = car2.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.1421  -0.3727  -0.3199  -0.2755   3.3988
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)       -23.31800    7.81829  -2.982  0.00286 **
## fix_time            0.25662    0.11120   2.308  0.02102 *
## a_speed             2.85861    4.97567   0.575  0.56562
## b_speed             1.70299    0.23727   7.177 7.10e-13 ***
## mpa                -0.11668    0.01901  -6.137 8.42e-10 ***
## load_time          -0.23939    0.12350  -1.938  0.05258 .
## highpressure_time   0.12109    0.02126   5.696 1.23e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1646.9  on 2401  degrees of freedom
## Residual deviance: 1105.7  on 2395  degrees of freedom
## AIC: 1119.7
##
## Number of Fisher Scoring iterations: 6
```

```
vs3 <- stepAIC(lm2,direction="both")
```

```
## Start:  AIC=1119.71
## failure ~ fix_time + a_speed + b_speed + mpa + load_time + highpressure_ti
me
##
##                     Df Deviance    AIC
## - a_speed            1   1106.1 1118.1
## <none>                   1105.7 1119.7
## - load_time          1   1111.8 1123.8
## - fix_time           1   1114.5 1126.5
## - highpressure_time  1   1137.6 1149.6
## - b_speed            1   1151.3 1163.3
## - mpa                1   1204.2 1216.2
##
## Step:  AIC=1118.07
## failure ~ fix_time + b_speed + mpa + load_time + highpressure_time
##
##                     Df Deviance    AIC
## <none>                   1106.1 1118.1
## + a_speed            1   1105.7 1119.7
## - load_time          1   1112.0 1122.0
## - fix_time           1   1114.8 1124.8
## - highpressure_time  1   1137.8 1147.8
## - b_speed            1   1151.4 1161.4
## - mpa                1   1428.5 1438.5
```

```
vs3$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
```

```
## Initial Model:
## failure ~ fix_time + a_speed + b_speed + mpa + load_time + highpressure_ti
me
##
## Final Model:
## failure ~ fix_time + b_speed + mpa + load_time + highpressure_time
##
##
##        Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1                              2395   1105.710 1119.710
## 2 - a_speed  1 0.3574133      2396   1106.068 1118.068
```

- glm() 일반화선행모형 적합함수로 binomial 설정으로 로지스틱 회귀모형 적합.

- stepAIC 로 변수선택.

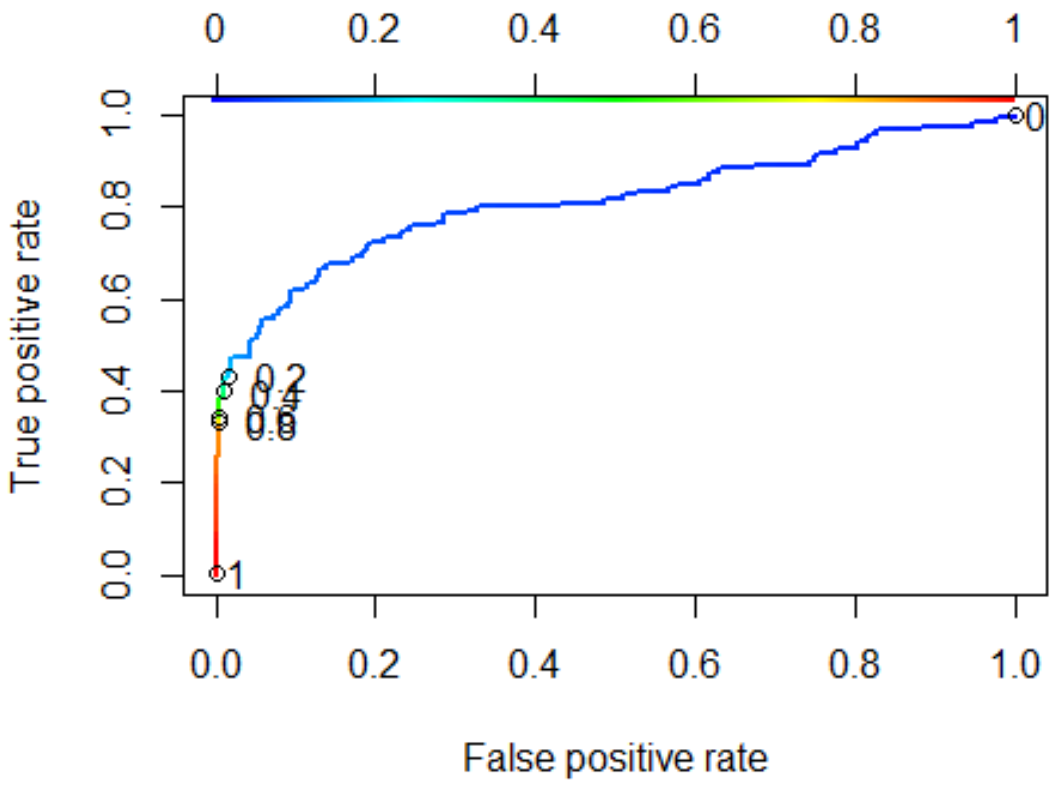
유의수준 0.05 에서 a 속도와 하중시간이 유의하지 않은 것으로 판명된다.

AIC 에 의한 변수선택결과 a 속도를 제외한 모형이 최적으로 판명되었다.

```
library(ROCR)
fit.log <- predict.glm(vs3, newdata=car2.test,type="response")
pred.log <- prediction(fit.log, car2.test$failure)
perf.log <- performance(pred.log, "tpr", "fpr")
```

- **ROCR** 는분류기의 성능을 시각화하기 위한 패키지임.
- predict.glm()에 의해 적합된 로지스틱 모형에서 예측값(예측확률:불량이 일어날 확률)을 계산.
- performace()함수에 의해 tpr(True positive rate:민감도)와 fpr(false positive rate:1-특이도)를 각각 계산.

```
plot(perf.log, colorize=T, colorkey.pos="top", print.cutoffs.at=seq(0, 1, by=
0.2), text.cex=1,
     text.adj=c(-0.5, 0.5), lwd=2)
```




```
auc <- performance(pred.log,"auc")
auc
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8094963
##
##
```

```
## Slot "alpha.values":
## list()
```

```
library(caret)
confusionMatrix(1*(fit.log>0.5), car2.test$failure)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 905  75
##          1   5  45
##
##                Accuracy : 0.9223
##                  95% CI : (0.9043, 0.9379)
##     No Information Rate : 0.8835
##     P-Value [Acc > NIR] : 2.697e-05
##
##                   Kappa : 0.4948
##  Mcnemar's Test P-Value : 1.215e-14
##
##             Sensitivity : 0.9945
##             Specificity : 0.3750
##          Pos Pred Value : 0.9235
##          Neg Pred Value : 0.9000
##              Prevalence : 0.8835
##          Detection Rate : 0.8786
##    Detection Prevalence : 0.9515
##       Balanced Accuracy : 0.6848
##
##        'Positive' Class : 0
##
```

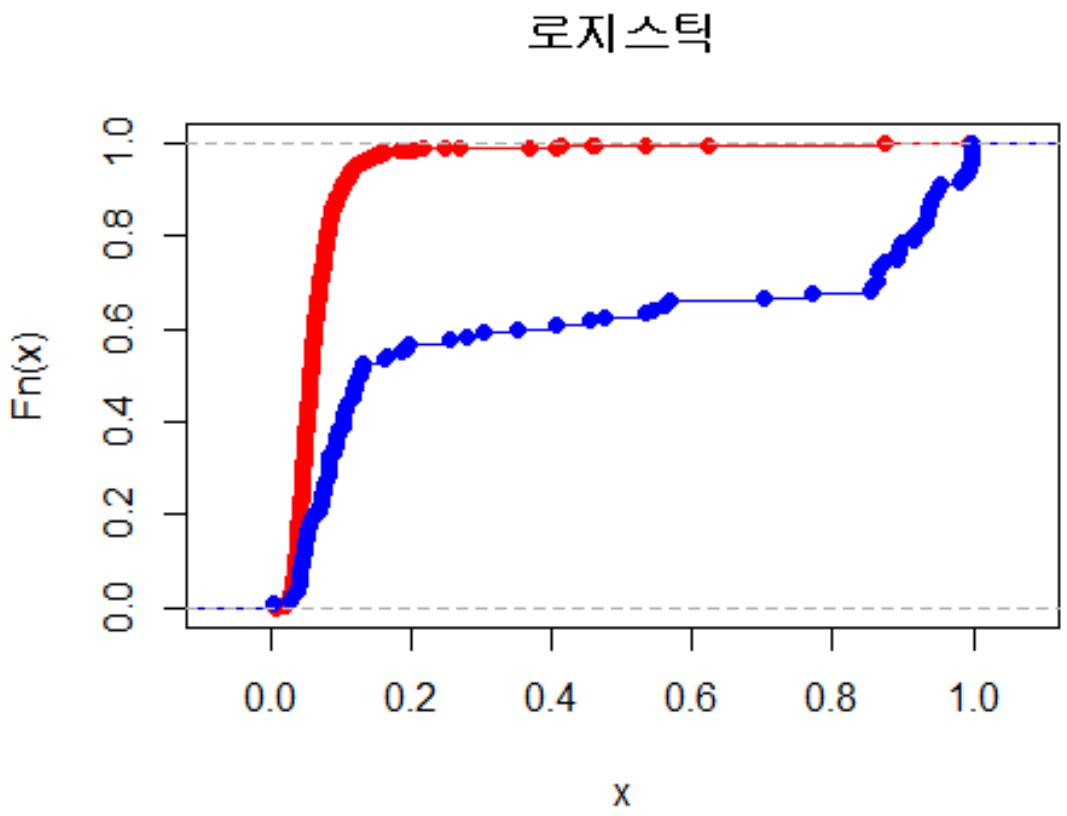
- ROC(Receiver Operating Characteristic) 곡선을 그림.
  - ROC 곡선은 cutoff 값의 변화에 따른 fpr 과 tpr 의 관계를 좌표평면상에 연결하여 나타낸 것으로 분류기의 성능을 평가하기 위해 쓰이는 대표적인 방법 중 하나임.
  - 곡선의 아래 면적 (AUC:Area Under Curve)가 1 에 가까울 수록 우수한 성능을 보임. 즉, 분류가 잘 된 것으로 볼 수 있음.
- `performance()` 함수에서 옵션을 `auc` 로 설정하여 AUC 값을 계산함.
- 분류행렬을 출력하고 오분류율을 계산.

로지스틱 모형에 의한 오분류율은 0.0777.

양품은 양품으로 대부분 잘 분류하였으나, 불량의심제품에 대한 분류성능이 약간 떨어지는 것으로 보인다.

AUC 값은 0.8095.

```
Label0.log <- fit.log[car2.test$failure==0]
Label1.log <- fit.log[car2.test$failure==1]
plot(ecdf(Label0.log), col="red",main="로지스틱")
plot(ecdf(Label1.log), col="blue",add=T)
```




```
ks.test(Label0.log, Label1.log)

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  Label0.log and Label1.log
## D = 0.537, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

- 양품과 불량의심제품의 두 그룹으로 나누어 각각 불량예측확률을 저장.

- `ecdf()`함수를 이용하여 각 그룹의 예측확률의 분포함수를 추정 (Empirical distribution function).

- `ks.test()`함수를 이용하여 KS(Kolmogorov-Smirnov) 통계량을 계산함.

KS 통계량 값이 클 수록 두 그룹이 잘 분리되었다는 의미이므로 분류기의 성능이 좋은 것으로 볼 수 있다.
여기서 KS 통계량 값은 0.537 이다.

양품은 0 근처에 예측확률이 집중되어 있으나, 불량의심품은 0 근처와 1 근처에 고르게 예측확률이 분포한다.
즉, 불량의심품의 분류에 어려움을 겪는다는 뜻으로 이는, 분류행렬에서 확인했던 바와 일치한다.

C.2 의사결정나무 (Decision tree)

의사결정나무는 (하나의 변수에 의한)분류규칙에 의해 자료를 이원화하는 과정을 반복하는 것으로 하위 노드의 불순도가 가능하면 커지는 방향으로 분류규칙이 결정된다.

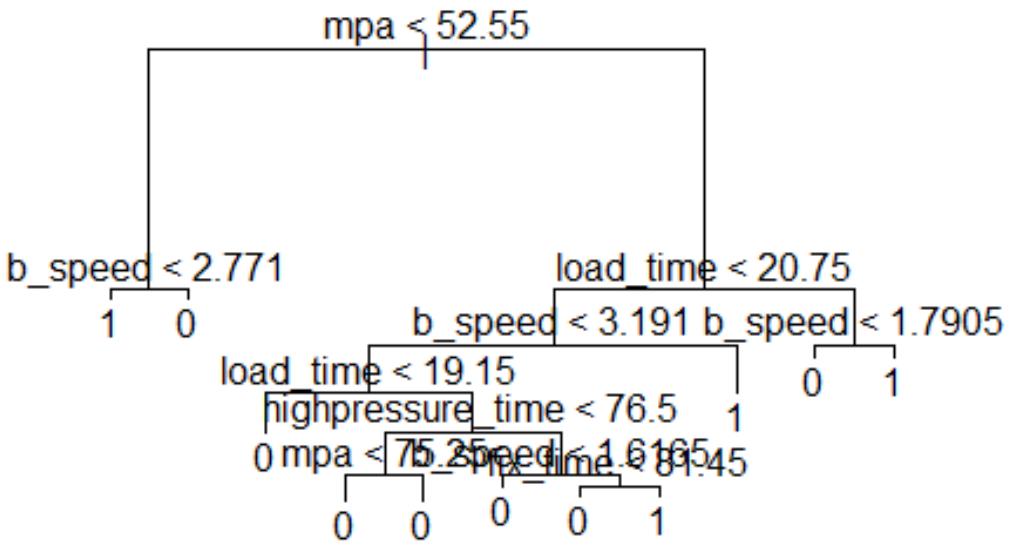
로지스틱 모형과 달리 비선형모형이다.
현 자료의 경우 공정변수들의 값이 이원화되어 있는 경우가 많았으므로 로지스틱 모형보다는 의사결정나무 모형에 의한 분류성능이 더 좋을 것으로 기대해 볼 수 있다.

직관적으로 모형의 이해가 가능하고 해석하기 좋으며 시각화가 잘 된다는 장점이 있다.

```
library(tree)
tree.car <- tree(failure~., data=car2.train)
summary(tree.car)
```

```
##
## Classification tree:
## tree(formula = failure ~ ., data = car2.train)
## Variables actually used in tree construction:
## [1] "mpa"               "b_speed"           "load_time"
## [4] "highpressure_time" "fix_time"
## Number of terminal nodes:  11
## Residual mean deviance:  0.3052 = 729.7 / 2391
## Misclassification error rate: 0.04829 = 116 / 2402
```

```
plot(tree.car)
text(tree.car)
```



- `tree()`를 이용하여 의사결정나무를 적합하고 결과를 그림.

- `text()`는 분류규칙을 나무 위에 표시.

의사결정나무 적합결과 실제 나무적합을 위해 쓰인 변수는 실압력, b 속도, 고압시간, 하중시간, 고정시간이다.
a 속도는 사용되지 않았다.

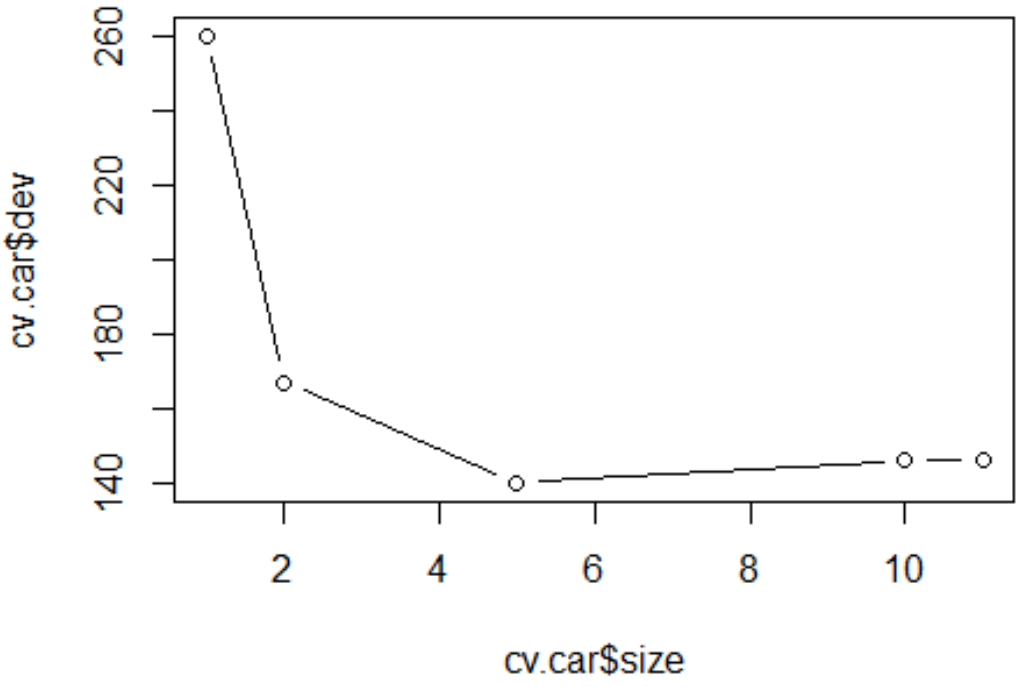
보통 큰 나무모형은 과적합(overfitting) 문제가 있는 것으로 알려져 있다.
적절한 가지치기(pruning)를 통해 나무의 크기를 줄여 예측 정확도를 향상시켜 볼 것이다.

```
set.seed(3)
cv.car <- cv.tree(tree.car, FUN=prune.misclass)
cv.car
```

```
## $size
## [1] 11 10  5  2  1
##
## $dev
## [1] 146 146 140 167 260
```

```
##
## $k
## [1]     -Inf  0.00000  3.00000 11.33333 95.00000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

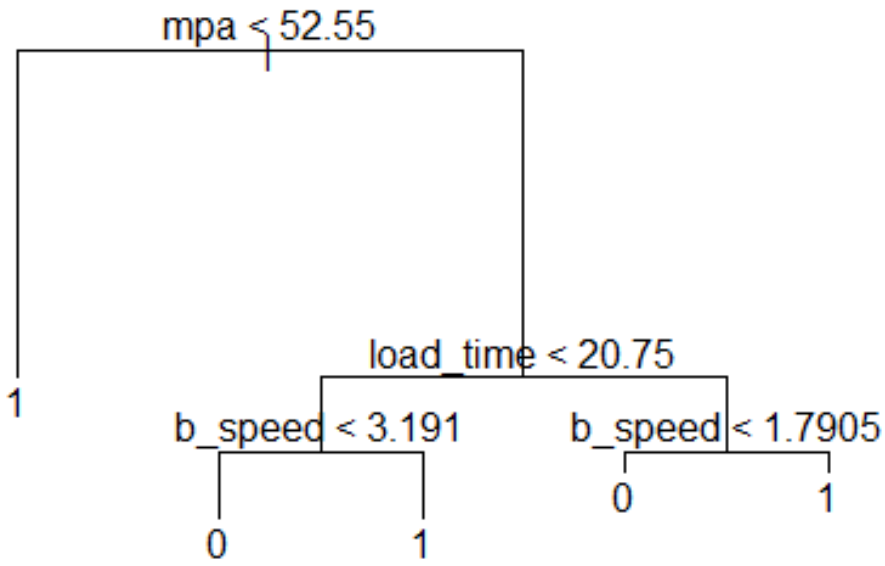
```
plot(cv.car$size,cv.car$dev, type="b")
```



- 가지치기를 위해 cross-validation 에 의해 deviance 를 최소로 하는 terminal node 의 개수를 찾음. 여기서 deviance 는 평균제곱오차(MSE)의 확장된 개념으로 일종의 오차임.

크기가 5 인 경우 deviance 가 최소가 된다.
따라서, 크기가 5 인 모형을 선택한다.

```
prune.car <- prune.misclass(tree.car, best=5)
plot(prune.car)
text(prune.car)
```



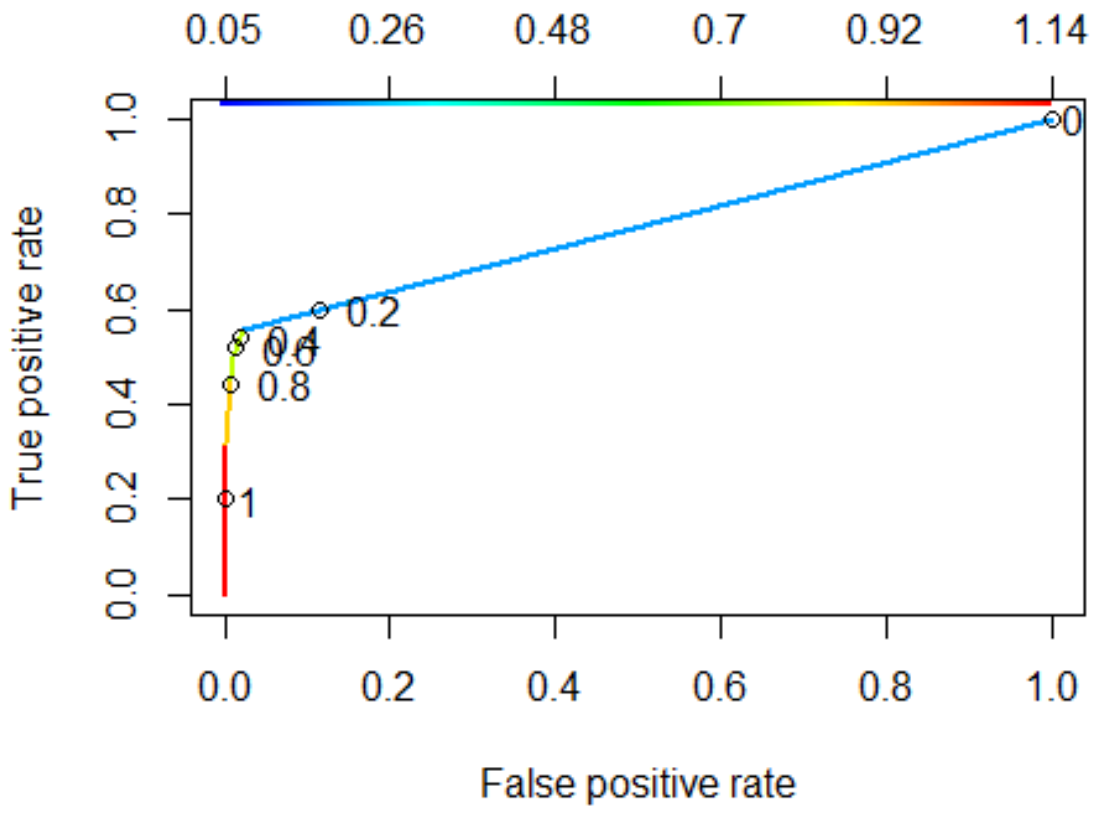

```
prune.car
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2402 1647.00 0 ( 0.89176 0.10824 )
##    2) mpa < 52.55 113   62.81 1 ( 0.07965 0.92035 ) *
##    3) mpa > 52.55 2289 1139.00 0 ( 0.93185 0.06815 )
##      6) load_time < 20.75 2218  939.10 0 ( 0.94545 0.05455 )
##       12) b_speed < 3.191 2194  825.20 0 ( 0.95351 0.04649 ) *
##       13) b_speed > 3.191 24   24.56 1 ( 0.20833 0.79167 ) *
##      7) load_time > 20.75 71   98.41 0 ( 0.50704 0.49296 )
##       14) b_speed < 1.7905 37   38.63 0 ( 0.78378 0.21622 ) *
##       15) b_speed > 1.7905 34   34.57 1 ( 0.20588 0.79412 ) *
```

나무가 단순해진 것을 확인할 수 있으며, 분류를 위해 쓰인 변수는 실압력, 하중시간, b 속도 세 개이다.

각 터미널(terminal) 노드와 해당 노드에서의 분류비율을 확인할 수 있다.

```
fit.tree <- predict(prune.car, newdata=car2.test,type="vector")
pred.tree <- prediction(fit.tree[,2], car2.test$failure)
perf.tree <- performance(pred.tree, "tpr", "fpr")
plot(perf.tree, colorize=T, colorkey.pos="top", print.cutoffs.at=seq(0,1,by=0.
2), text.cex=1, text.adj=c(-0.5, 0.5), lwd=2)
```




```
perf.tree1 <- performance(pred.tree, "auc")
perf.tree1
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
```

```
##
## Slot "y.values":
## [[1]]
## [1] 0.7717628
##
##
## Slot "alpha.values":
## list()
```

```
confusionMatrix(1*(fit.tree[,2]>0.5), car2.test$failure)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 901  60
##          1   9  60
##
##                Accuracy : 0.933
##                  95% CI : (0.916, 0.9475)
##     No Information Rate : 0.8835
##     P-Value [Acc > NIR] : 7.123e-08
##
##                   Kappa : 0.601
##  Mcnemar's Test P-Value : 1.752e-09
##
##             Sensitivity : 0.9901
##             Specificity : 0.5000
##          Pos Pred Value : 0.9376
##          Neg Pred Value : 0.8696
##              Prevalence : 0.8835
##          Detection Rate : 0.8748
##    Detection Prevalence : 0.9330
##       Balanced Accuracy : 0.7451
##
##        'Positive' Class : 0
##
```

오분류율은 0.067 이다.

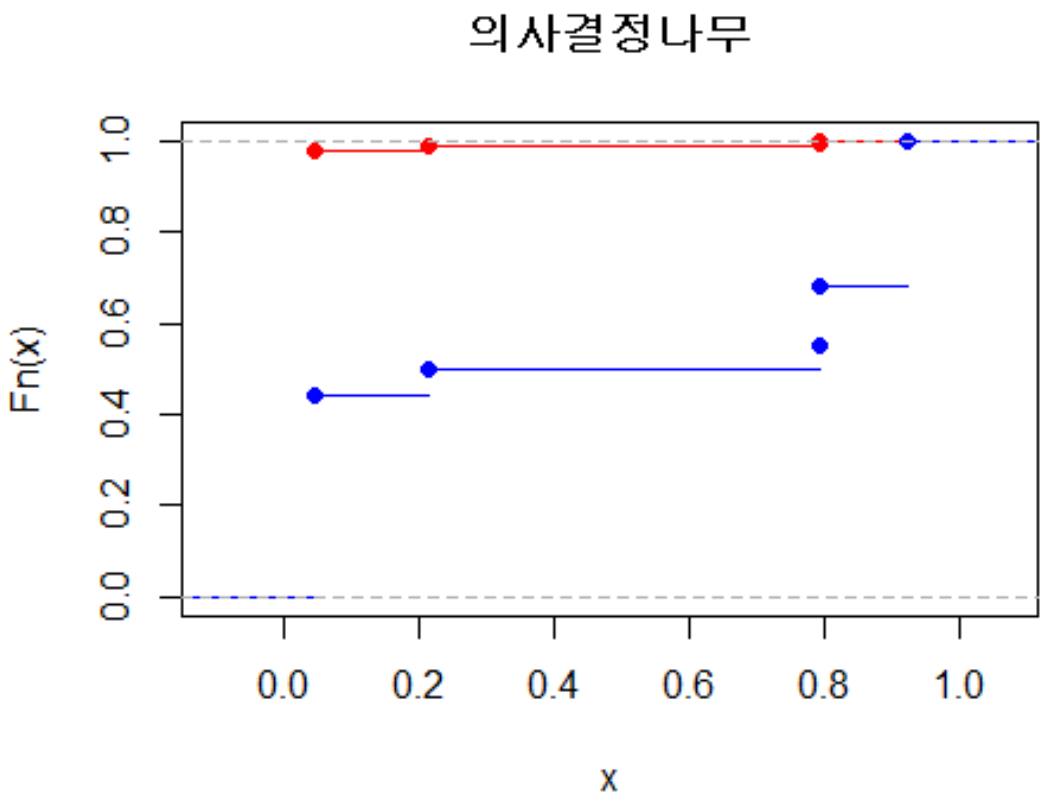
로지스틱 모형에 비해서, 불량의심품을 분류해 내는 성능이 다소 개선되었음을 확인할 수 있다,

AUC 는 0.7718 로 로지스틱 모형에 비해 작다.

즉, 양품과 불량의심품을 구분해 내는 데에는 상대적으로 성공적이었으나 전체적인 분류성능이 매우 좋다고 할 수는 없다.

```
Label0.tree <- fit.tree[,2][car2.test$failure==0]
Label1.tree <- fit.tree[,2][car2.test$failure==1]
```

```r
plot(ecdf(Label0.tree),col="red",main="의사결정나무")
plot(ecdf(Label1.tree),col="blue",add=T)
```




```r
ks.test(Label0.tree,Label1.tree)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  Label0.tree and Label1.tree
## D = 0.53526, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

KS 통계량 값은 0.5353 으로 로지스틱 모형과 큰 차이가 없다.

실제 분류를 위해 사용된 변수는 실압력, 하중시간, b 속도 3 개의 변수이다.

특히, b 속도는 커질 수록 불량으로 판명될 가능성이 전반적으로 크다 할 수 있다.

C.3 앙상블(Ensemble) 모형

의사결정나무모형은 여러 장점을 가지고 있으나 분산이 크다는 단점이 있다.
즉, 주어진 자료가 조금만 달라지더라도 전혀 다른 모형을 생성하는 경우가 많다.

이를 보완하기 위해 여러 개의 나무를 생성한 후 결과로 나온 예측값을 종합하는 방법을 고려할 수 있는데, 이를 앙상블(Ensemble) 모형이라 한다.

Averaging 을 통해 분산을 줄이는 효과를 내는 것으로 Bootstrap sampling 에 기반하여 여러 개의 나무를 생성한다.

C.3.1 Bagging (Bootstrap aggregating)

B 개의 부스트랩 표본에 대해 나무를 구성하고 종합하는 방식

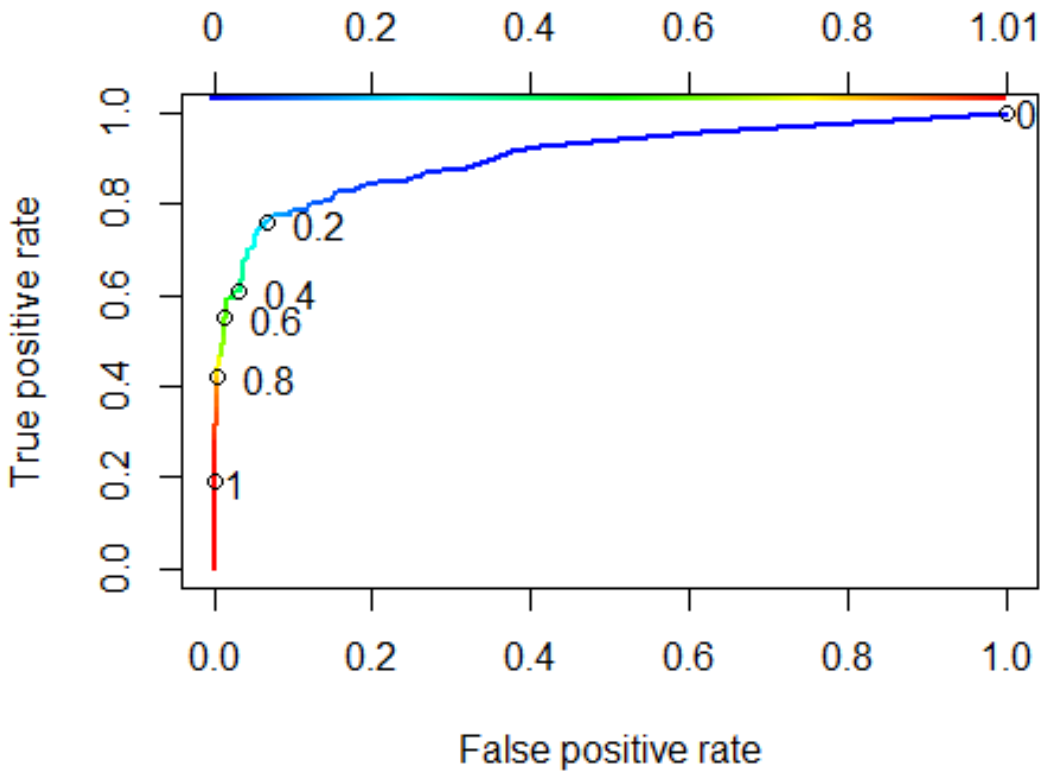
```
library(randomForest)
bag.car <- randomForest(failure~., data=car2.train, mtry=6)
bag.car
```

```
##
## Call:
##  randomForest(formula = failure ~ ., data = car2.train, mtry = 6)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 5.91%
## Confusion matrix:
##      0   1 class.error
## 0 2101  41  0.01914099
## 1  101 159  0.38846154
```

- **randomForest** : 앙상블모형을 적합하는 패키지.

- mtry=6, 즉 모든 변수를 개별나무모형 생성에 포함하여 Bagging 을 수행.

```
fit.bag <- predict(bag.car, newdata=car2.test,type="prob")
pred.bag <- prediction(fit.bag[,2], car2.test$failure)
perf.bag <- performance(pred.bag, "tpr", "fpr")
plot(perf.bag, colorize=T, colorkey.pos="top", print.cutoffs.at=seq(0,1,by=0.
2), text.cex=1,text.adj=c(-0.5, 0.5), lwd=2)
```




```
perf.bag1 <- performance(pred.bag,"auc")
perf.bag1
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9034386
##
##
```

```
## Slot "alpha.values":
## list()
```

```
confusionMatrix(1*(fit.bag[,2]>0.5), car2.test$failure)
```

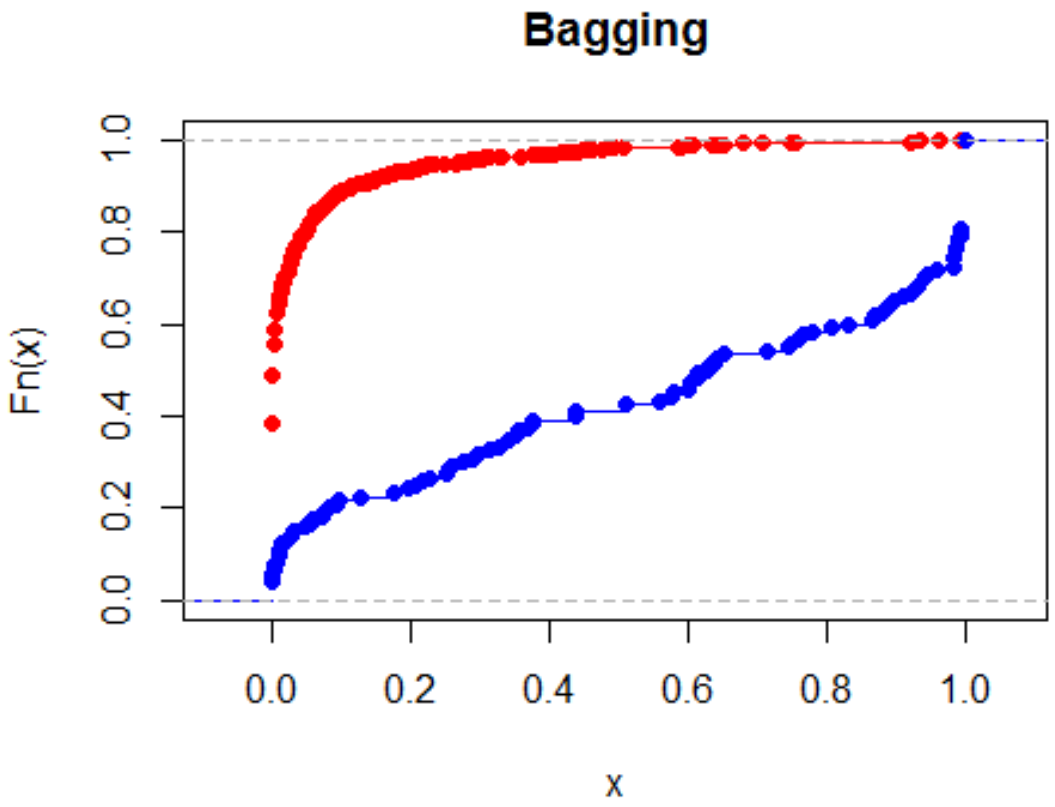
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 894  49
##          1  16  71
##
##                Accuracy : 0.9369
##                  95% CI : (0.9203, 0.951)
##     No Information Rate : 0.8835
##     P-Value [Acc > NIR] : 5.302e-09
##
##                   Kappa : 0.6519
##  Mcnemar's Test P-Value : 7.214e-05
##
##             Sensitivity : 0.9824
##             Specificity : 0.5917
##          Pos Pred Value : 0.9480
##          Neg Pred Value : 0.8161
##              Prevalence : 0.8835
##          Detection Rate : 0.8680
##    Detection Prevalence : 0.9155
##       Balanced Accuracy : 0.7870
##
##        'Positive' Class : 0
##
```

오분류율은 0.0631 로 의사결정나무모형에 비해 소폭 감소하였다.

AUC 는 0.9034 로 로지스틱모형(약 0.8)이나 의사결정나무모형(약 0.77)에 비해 매우 크게 향상되었음을 알 수 있다.

즉, 분류가 이전보다 더 확실하게 되었다.

```
Label0.bag <- fit.bag[,2][car2.test$failure==0]
Label1.bag <- fit.bag[,2][car2.test$failure==1]
plot(ecdf(Label0.bag),col="red",main="Bagging")
plot(ecdf(Label1.bag),col="blue",add=T)
```




```
ks.test(Label0.bag,Label1.bag)

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  Label0.bag and Label1.bag
## D = 0.70247, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

KS 통계량 또한 0.7025 로 0.5353 이었던 의사결정나무모형에 비해 크게 좋아졌다.

C.3.2 랜덤포레스트 (Random forest)

부스트랩 표본에 근거하여 여러 개의 나무를 생성하여 종합한다는 면에서는 Bagging 과 같으나, 나무 생성시 모든 변수를 사용하지 않고 랜덤하게 일부분의 변수만을 선택하여 사용한다는 점이 다르다.
이는, 상관관계가 큰 변수들이 모두 모형에 포함되어 모형의 분산을 높이는 것을 방지할 수 있는 효과가 있다.

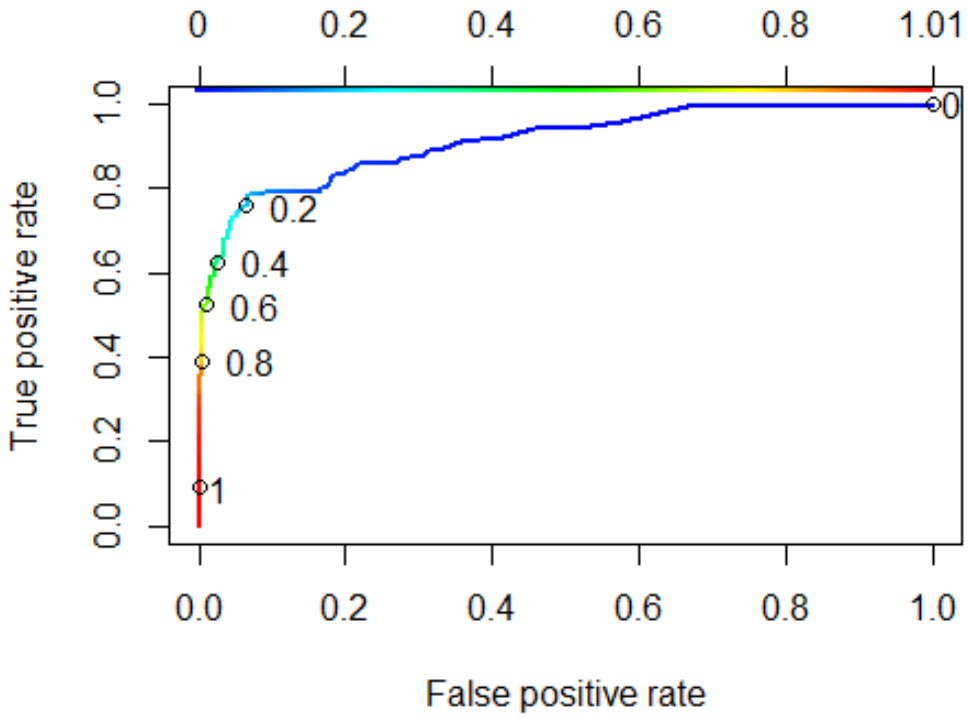
개별 나무들간의 연관성을 약화시킴으로써 결합된 모형의 분산을 감소시킬 수 있다.

```
library(randomForest)
rf.car <- randomForest(failure~.,data=car2.train,mtry=3)
rf.car

##
## Call:
##  randomForest(formula = failure ~ ., data = car2.train, mtry = 3)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.66%
## Confusion matrix:
##      0   1 class.error
## 0 2105  37  0.01727358
## 1   99 161  0.38076923
```

- mtry=3, 6 개의 변수 중 3 개의 변수만 개별나무 생성에 기여하도록 설정.

```
fit.rf <- predict(rf.car,newdata=car2.test,type="prob")
pred.rf <- prediction(fit.rf[,2], car2.test$failure)
perf.rf <- performance(pred.rf,"tpr","fpr")
plot(perf.rf, colorize=T, colorkey.pos="top", print.cutoffs.at=seq(0,1,by=0.
2), text.cex=1,text.adj=c(-0.5, 0.5), lwd=2)
```



```
perf.rf1 <- performance(pred.rf,"auc")
perf.rf1
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9110165
##
##
## Slot "alpha.values":
## list()
```

```
confusionMatrix(1*(fit.rf[,2]>0.5),car2.test$failure)
```

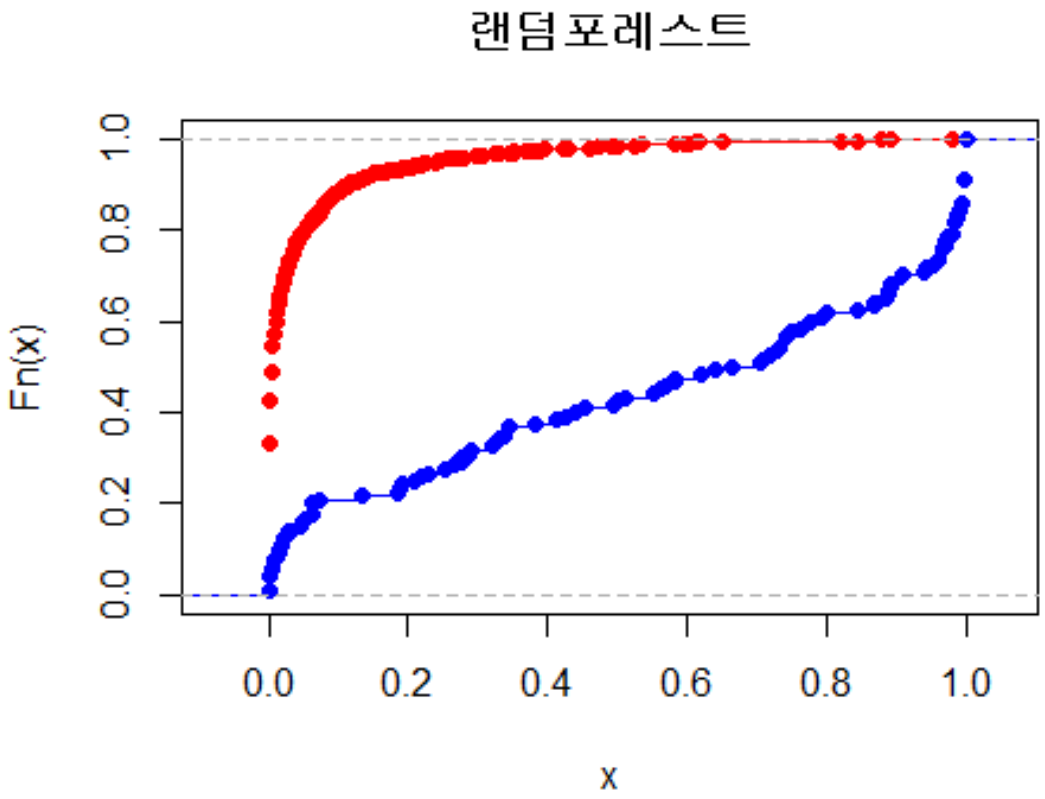
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 895  50
##          1  15  70
##
##                Accuracy : 0.9369
##                  95% CI : (0.9203, 0.951)
##     No Information Rate : 0.8835
##     P-Value [Acc > NIR] : 5.302e-09
##
##                   Kappa : 0.649
##  Mcnemar's Test P-Value : 2.474e-05
##
##             Sensitivity : 0.9835
##             Specificity : 0.5833
##          Pos Pred Value : 0.9471
##          Neg Pred Value : 0.8235
##              Prevalence : 0.8835
##          Detection Rate : 0.8689
##    Detection Prevalence : 0.9175
##       Balanced Accuracy : 0.7834
##
```

```
##         'Positive' Class : 0
##
```

오분류율은 0.0631 로 bagging 과 동일하였다.

AUC 는 0.9110 으로 bagging(0.9034)에 비해 소폭 상승하였다.

```
Label0.rf <- fit.rf[,2][car2.test$failure==0]
Label1.rf <- fit.rf[,2][car2.test$failure==1]
plot(ecdf(Label0.rf),col="red",main="랜덤포레스트")
plot(ecdf(Label1.rf),col="blue",add=T)
```




```
ks.test(Label0.rf,Label1.rf)

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  Label0.rf and Label1.rf
## D = 0.7152, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

KS 통계량 또한 0.7152 로 bagging 에 비해 소폭 상승하여, 더 분류가 잘 되었다고 할 수 있다.

C.4 신경망모형 (Neural network model)

신경망모형은 생물학의 신경망을 본따 만든 학습 알고리즘으로 비선형 모형이다.

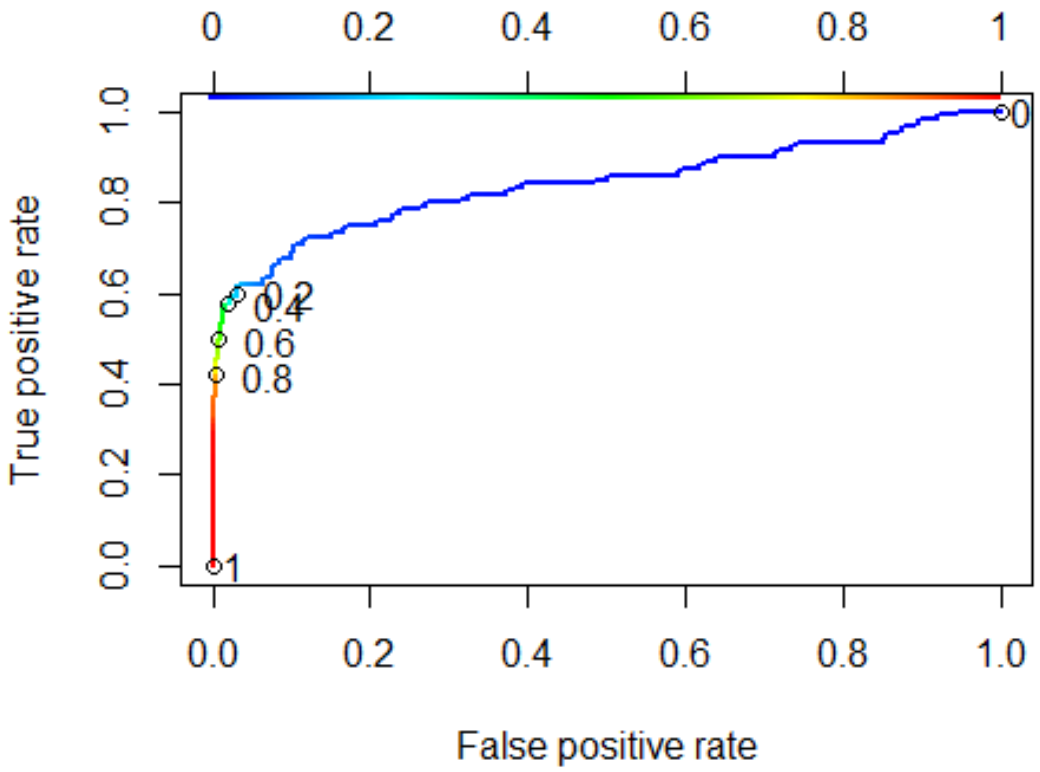
굉장히 다양한 모형이 존재하지만 여기서는 가장 잘 알려져 있는 것 중 하나인 다층퍼셉트론(Multilayer perceptron) 모형을 이용한 적합을 소개한다.

신경망모형의 적합을 위해서는 설명변수들을 표준화(normalization 하는 것이 좋은 것으로 알려져 있다.

```
library(RSNNS)
Y <- decodeClassLabels(car2.train$failure)
X <- normalizeData(car2.train[,-7])
test.Y <- decodeClassLabels(car2.test$failure)
test.X <- normalizeData(car2.test[,-7])
```

- **RSNNS** : 신경망모형 적합을 위한 패키지.
- `normalizeData()`를 이용하여 표준화.
- 훈련자료와 검증자료 분류

```
nn.car <- mlp(X,Y,size=5,maxit=50)
fit.nn <- predict(nn.car,test.X)
pred.nn <- prediction(fit.nn[,2], car2.test$failure)
perf.nn <- performance(pred.nn,"tpr","fpr")
plot(perf.nn, colorize=T, colorkey.pos="top", print.cutoffs.at=seq(0,1,by=0.
2), text.cex=1,text.adj=c(-0.5, 0.5), lwd=2)
```




```
perf.nn1 <- performance(pred.nn,"auc")
perf.nn1
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8393681
##
##
```

```
## Slot "alpha.values":
## list()
```

```
library(caret)
confusionMatrix(1*(fit.nn[,2]>0.5),car2.test$failure)
```

```
##          predictions
## targets   1   2
##       0 899  56
##       1  11  64
```

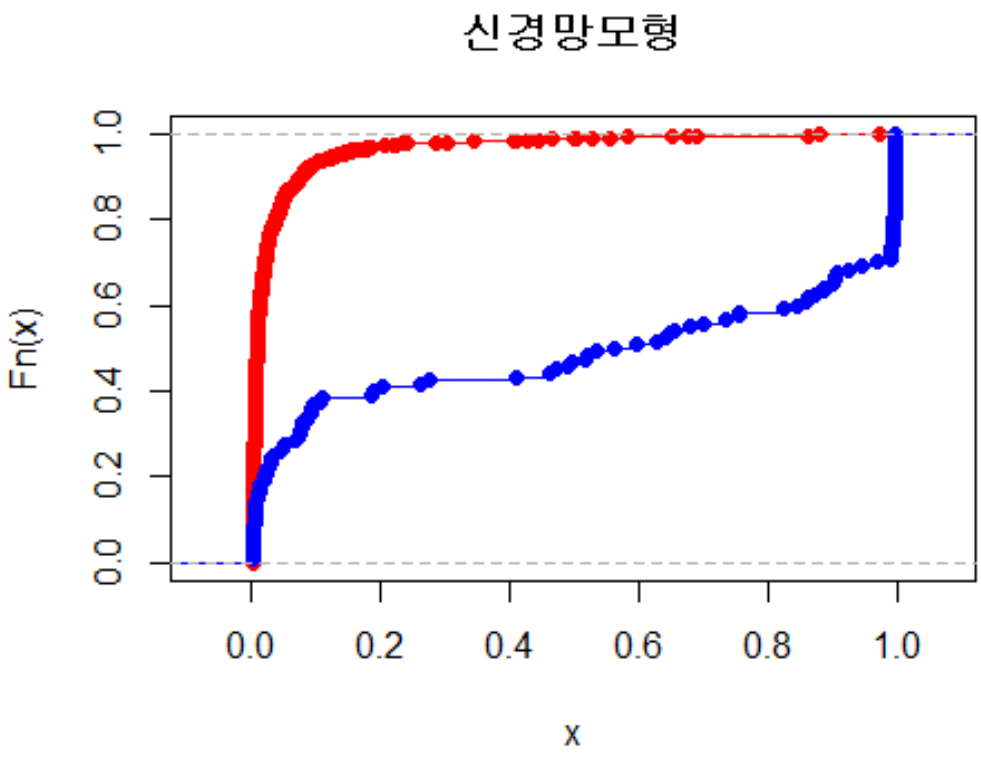
- 은닉층의 개수를 5 개로 설정 (size=5)

- 학습을 위한 최대 반복수를 50 으로 설정

오분류율이 0.0696 으로 로지스틱모형에 비해서는 좋으나 다른 모형들에 비해서는 다소 떨어진다.

AUC 가 0.8394 으로 로지스틱 및 의사결정나무에 비해서 큰 값을 가진다.

```
Label0.nn <- fit.nn[,2][car2.test$failure==0]
Label1.nn <- fit.nn[,2][car2.test$failure==1]
plot(ecdf(Label0.nn),col="red",main="신경망모형")
plot(ecdf(Label1.nn),col="blue",add=T)
```



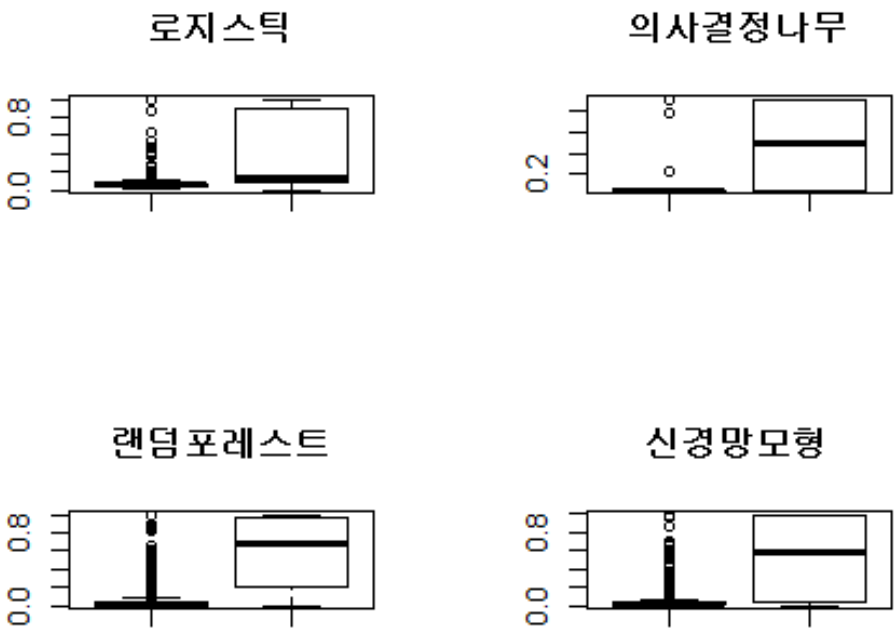
```r
ks.test(Label0.nn,Label1.nn)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  Label0.nn and Label1.nn
## D = 0.60632, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

- KS 통계량이 0.6063 으로 로지스틱과 의사결정나무에 비해서 크며, 앙상블모형에 비해서는 작다.

C.5 모형성능비교

```r
par(mfrow=c(2,2))
boxplot(Label0.log,Label1.log,main="로지스틱")
boxplot(Label0.tree,Label1.tree,main="의사결정나무")
boxplot(Label0.rf,Label1.rf,main="랜덤포레스트")
boxplot(Label0.nn,Label1.nn,main="신경망모형")
```



위 그림은 양품(좌), 불량의심품(우) 그룹에서 각 모형에 의한 예측확률을 상자그림으로 표현한 것이다.
즉, 좌측 상자는 0 에 가깝고 우측상자는 1 에 가까울 수록, 또한 두 상자의 폭이 좁고 서로 멀리 떨어져 있을 수록 분류가 잘 된 것으로 볼 수 있다.

약간씩의 차이가 있기는 하지만 양품은 불량확률을 0 에 매우 가깝게 대부분 예측하고 있음을 알 수 있다.
다만, 양품을 양품으로 가장 잘 예측하는 것은 의사결정나무모형인 것으로 보인다.

불량의심품의 경우에는 모든 분류모형이 저하된 성능을 보였다.
상대적으로는 로지스틱이 가장 좋지 않은 성능을 보이고 있으며, 랜덤포레스트(Random forest)가 가장 좋은 성능을 보여주고 있다.

앞서 보았듯이 오분류율, AUC, KS 통계량의 관점에서도 비슷한 결론을 내릴 수 있었다.