2020년 데이터마이닝 기말 프로젝트 및 빅데이터 경진대회 문제

정보통계보험수리학과
20161187 임찬묵
20161184 최석현

주어진 white wine quality data set에는 white wine의 quality에 대한 정보를 담고 있다. white wine 데이터에는 아래와 같은 11개의 설명변수와 'quality'라는 반응변수 그리고 4898개의 샘플이 포함되어 있다. 설명변수는 pH값과 같은 wine의 객관적인 정보와 관련되어 있고 반응변수는 3명 이상의 wine 전문가에 의해서 평가된 점수의 중간값을 기초로 작성되어 있다.

Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

이 데이터는 기본적으로 regression 문제이지만 white wine의 quality score를 good이나 bad로 나누게 되면 classification 문제로 바꾸어서 생각해 볼 수도 있다. 데이터마이닝 수업시간에 배운 10-fold cross-validation을 활용하여 white wine의 quality score를 예측하는 모델을 여러 가지 (eg. linear model, LDA, tree, random forest, boosting, SVM) 만들어 보고 어떤 모델의 예측력이 좋은지 비교해 보아라.

```
# R Script
###### Data Mining - Project R Script ######
  library(tidyverse)      # data science package
  library(tm)             # Corpus
  library(ggplot2)        # Graphics for data.frame
  library(ggiraph)        # visualize linear Regression
  library(ggiraphExtra)   # visualize linear Regression
  library(car)            # KNN method
  library(magrittr)       # pipe
  library(nnet)           # multinom
  library(MASS)           # LDA, QDA
  library(cvTools)        # cvFolds
  library(tree)           # tree
  library(randomForest) # randomForest
```

Exploratory Data Analysis (EDA)

* Variable Explanation

변수명	의역	변수 설명
fixed acidity	고정 산도	대부분의 산도는 wine/fixed/nonvolatile 와 관련
volatile acidity	휘발성 산도	와인에 함유된 아세트산의 양
citric acid	구연산	와인에 맛과 풍미를 더할 수 있는 산(acid)
residual sugar	비정재 설탕	발효가 멈춘 후 남은 설탕의 양
chlorides	화학물질	와인에 있는 설탕의 양
free sulfur dioxide	자유이산화황	미생물 성장과 와인의 산화를 방지 (SO2 자유형)
total sulfur dioxide	총이산화황양	자유형SO2 + 결합형SO2의 양 (저농도는 주로 검출X)
density	밀도	알코올 및 설탕 함량 비율에 따라 밀도가 다름
pH	산성도	0~14의 값을 가지며 와인은 보통 3~4의 값을 가진다.
sulphates	황산염	이황산가스를 유발할수 있는 물질로 향균 역할
alcohol	알코올	와인에 있는 알코올의 비율(%)
quality	품질	세명이상의 전문가 평가의 중앙값으로 0~10의 값을 가진다.

```
data <- read.csv("winequality-white.csv", header = T)
names(data) %<>% str_replace_all(" ", "_")          # 편리성을 위해 변수사이 공백제거
head(data, 10)
```

```
> head(data, 10)
# A tibble: 10 x 12
   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides free_sulfur_dioxide total_sulfur_dioxide density    pH sulphates alcohol quality
           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>              <dbl>               <dbl>   <dbl> <dbl>     <dbl>   <dbl>   <dbl>
 1           7              0.27        0.36           20.7     0.045                 45                 170    1.00  3            0.45     8.8       6
 2           6.3            0.3         0.34            1.6     0.049                 14                 132    0.994 3.3          0.49     9.5       6
 3           8.1            0.28        0.4             6.9     0.05                  30                  97    0.995 3.26         0.44    10.1       6
 4           7.2            0.23        0.32            8.5     0.058                 47                 186    0.996 3.19         0.4      9.9       6
 5           7.2            0.23        0.32            8.5     0.058                 47                 186    0.996 3.19         0.4      9.9       6
 6           8.1            0.28        0.4             6.9     0.05                  30                  97    0.995 3.26         0.44    10.1       6
 7           6.2            0.32        0.16            7       0.045                 30                 136    0.995 3.18         0.47     9.6       6
 8           7              0.27        0.36           20.7     0.045                 45                 170    1.00  3            0.45     8.8       6
 9           6.3            0.3         0.34            1.6     0.049                 14                 132    0.994 3.3          0.49     9.5       6
10           8.1            0.22        0.43            1.5     0.044                 28                 129    0.994 3.22         0.45    11         6
```

```
str(data) # 4898 * 12 var
```

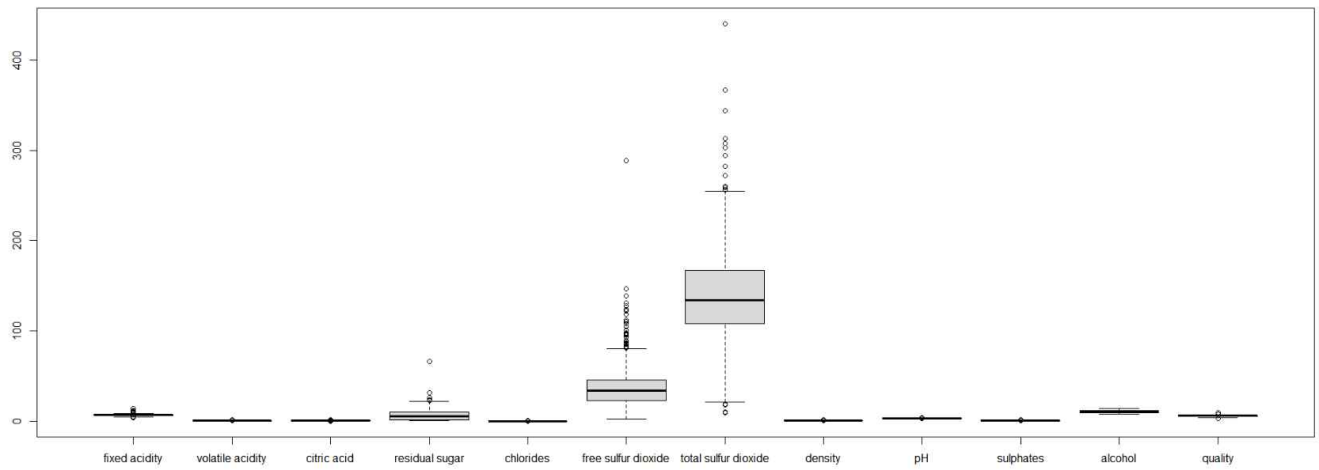
```
> str(data)
tibble [4,898 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ fixed_acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile_acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric_acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual_sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free_sulfur_dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
 $ total_sulfur_dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
 - attr(*, "spec")=
  .. cols(
  ..   `fixed acidity` = col_double(),
  ..   `volatile acidity` = col_double(),
  ..   `citric acid` = col_double(),
  ..   `residual sugar` = col_double(),
  ..   chlorides = col_double(),
  ..   `free sulfur dioxide` = col_double(),
  ..   `total sulfur dioxide` = col_double(),
  ..   density = col_double(),
  ..   pH = col_double(),
  ..   sulphates = col_double(),
  ..   alcohol = col_double(),
  ..   quality = col_double()
  .. )
```

summary(data)

```
> summary(data)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides       free.sulfur.dioxide
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   :  2.00
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600   1st Qu.: 23.00
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Median :0.04300   Median : 34.00
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean   :0.04577   Mean   : 35.31
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Qu.:0.05000   3rd Qu.: 46.00
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.   :0.34600   Max.   :289.00
 total.sulfur.dioxide    density             pH           sulphates         alcohol          quality
 Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
 1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
 Median :134.0        Median :0.9937   Median :3.180   Median :0.4700   Median :10.40   Median :6.000
 Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
 3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
 Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

✓ NA값은 없으나 일부 변수에서 정상 범주를 넘어서는 이상값들이 있는 것으로 보임

boxplot(data)



✓ 대다수의 변수에서 이상치가 발견되었으나 데이터 정제작업(Cleansing)은 후에 각 모델의 특성에 맞게 할 예정.

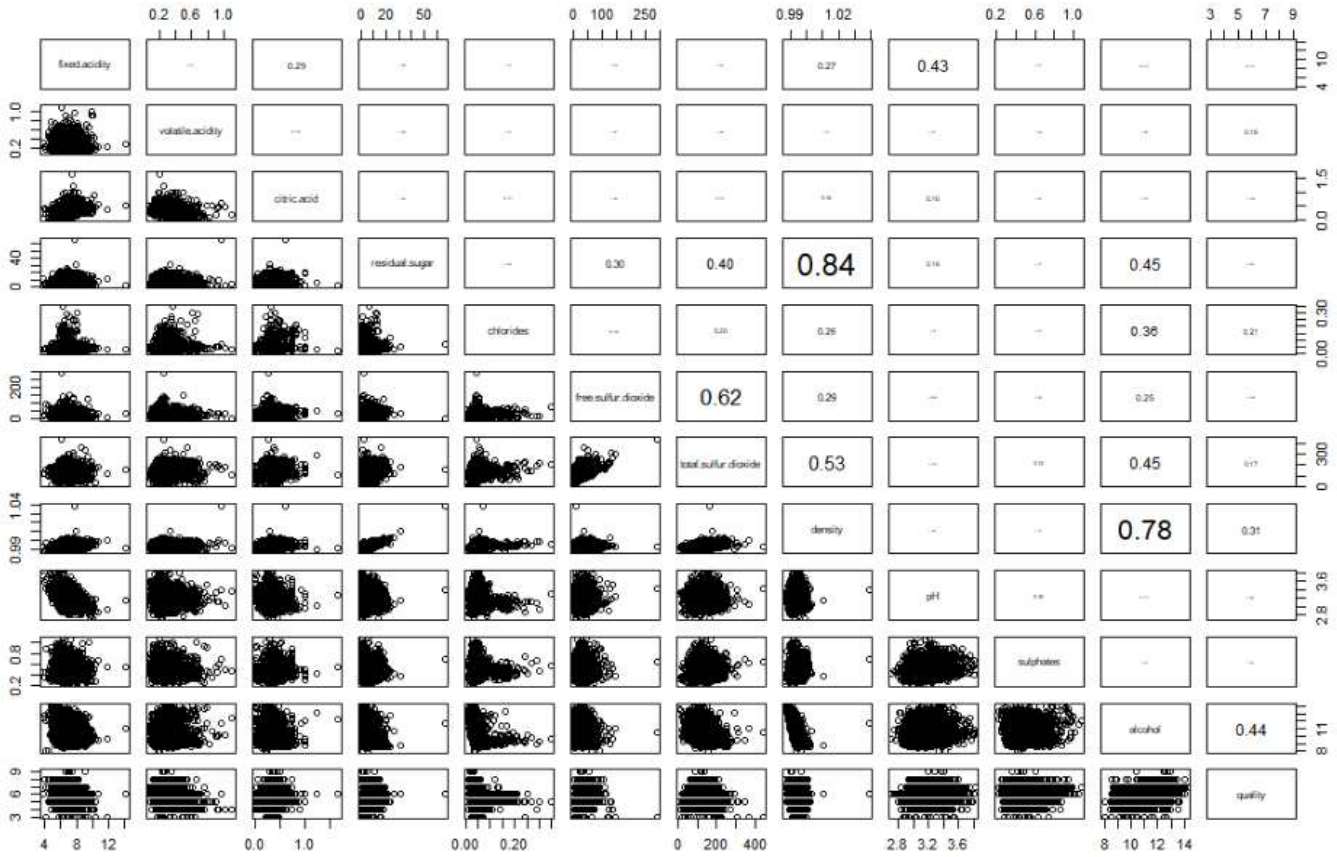
test of normalize

sapply(data, function(x) shapiro.test(x)$p.value)

```
> sapply(data, function(x) shapiro.test(x)$p.value)
       fixed acidity     volatile acidity          citric acid       residual sugar
        1.150151e-27         4.586797e-48         1.013179e-44         2.820710e-51
           chlorides  free sulfur dioxide total sulfur dioxide              density
        2.140584e-75         3.857845e-40         4.383453e-19         1.780895e-36
                  pH            sulphates              alcohol              quality
        6.505521e-20         1.821979e-37         2.569014e-36         1.340111e-50
```

✓ 만일 데이터가 다변량 정규분포에서 온 자료라면 분석 시에 용이하므로 각 변수에 shapiro함수를 적용하자. 이때 모든 변수가 유의수준 0.01에서 귀무가설을 기각하므로 데이터의 각 변수는 정규분포를 따르지 않는다. 또한 다변량 정규분포에서의 각 변수는 정규분포를 따르므로 해당 데이터는 다변량 정규분포라고 볼 수 없다.

```
# Visualize Correlation of Variable

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(data, upper.panel = panel.cor)
```



✓ 일부 변수사이에 높은 상관성 확인 및 산점도 그림에서 일부 이상치들이 있음을 확인할 수 있음

✓ 산점도 그림에서 대체로 타원형의 모양을 가지지 않고 있어서 다변량 정규분포에서 나왔다고 볼 수 없음

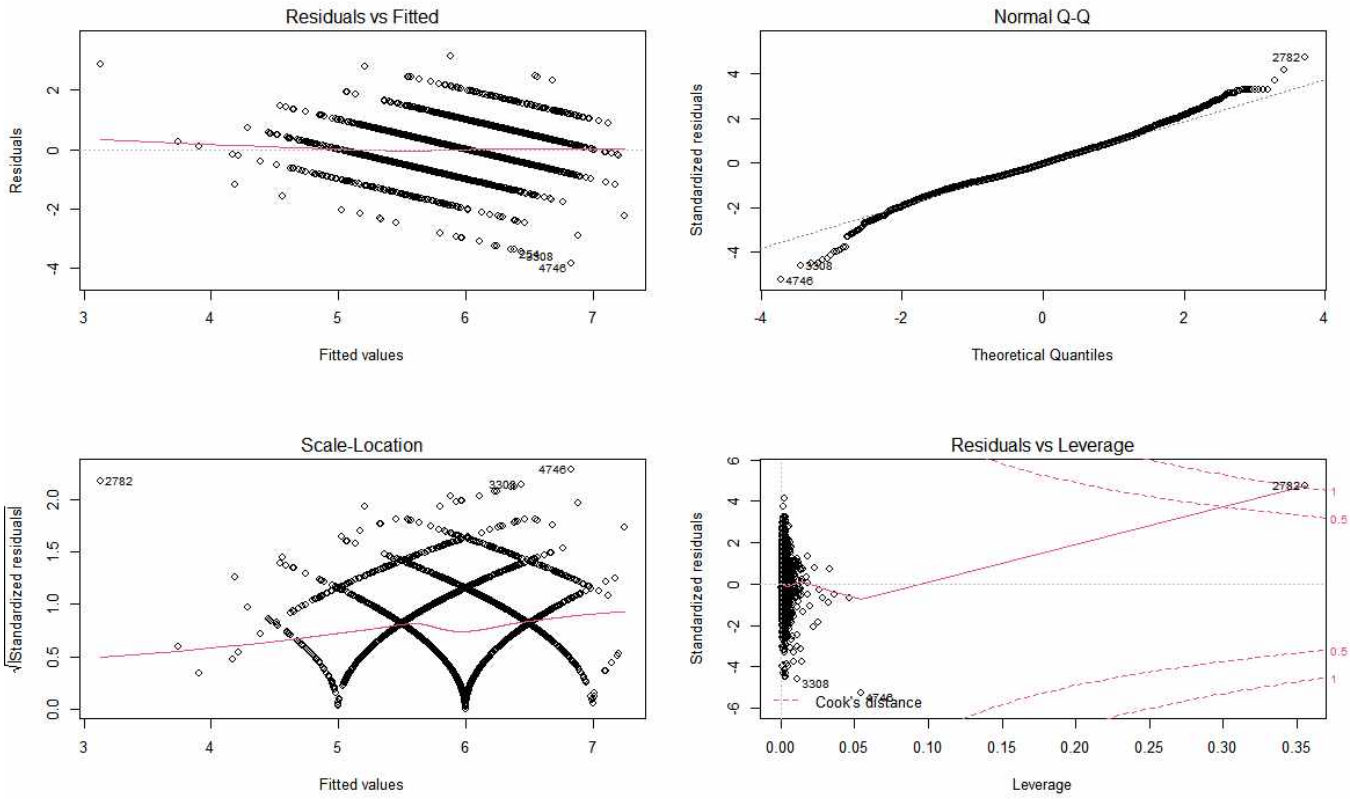
✓ (alcohol, density), (density, residual.sugar), (density, free sulfur dioxide)
(free sulfur dioxide, total sulfur dioxide) 의 상관성이 높음에 유의

다중선형회귀분석 (Multiple linear regression Analysis)

Multiple linear Regression

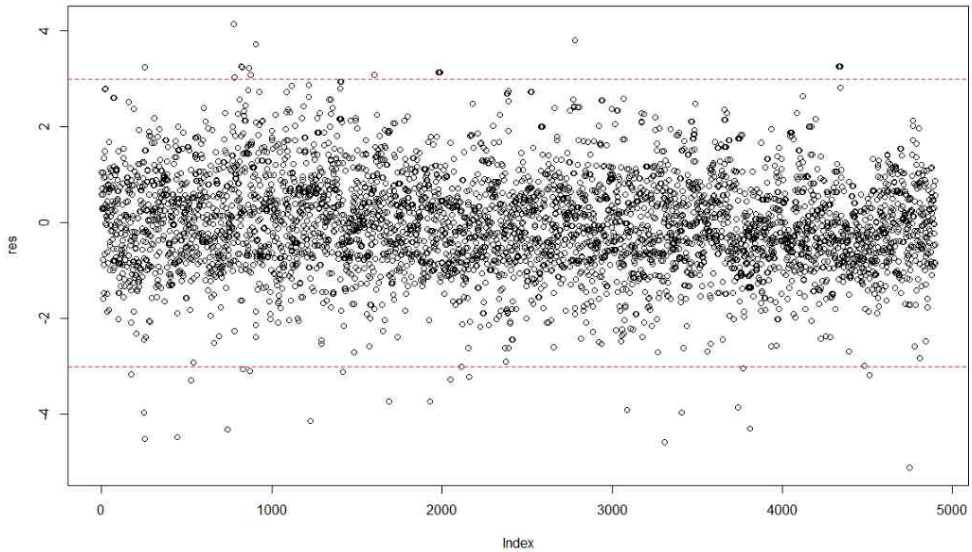
```
par(mfrow=c(2,2))
lm.fit <- lm(quality~., data=data)
summary(lm.fit)
plot(lm.fit)
```

```
> summary(lm.fit)       # adjust R : 0.2803, RES = 0.7514

Call:
lm(formula = quality ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8348 -0.4934 -0.0379  0.4637  3.1143

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.502e+02  1.880e+01   7.987 1.71e-15 ***
fixed_acidity         6.552e-02  2.087e-02   3.139  0.00171 **
volatile_acidity     -1.863e+00  1.138e-01 -16.373  < 2e-16 ***
citric_acid           2.209e-02  9.577e-02   0.231  0.81759
residual_sugar        8.148e-02  7.527e-03  10.825  < 2e-16 ***
chlorides            -2.473e-01  5.465e-01  -0.452  0.65097
free_sulfur_dioxide   3.733e-03  8.441e-04   4.422 9.99e-06 ***
total_sulfur_dioxide -2.857e-04  3.781e-04  -0.756  0.44979
density              -1.503e+02  1.907e+01  -7.879 4.04e-15 ***
pH                    6.863e-01  1.054e-01   6.513 8.10e-11 ***
sulphates             6.315e-01  1.004e-01   6.291 3.44e-10 ***
alcohol               1.935e-01  2.422e-02   7.988 1.70e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7514 on 4886 degrees of freedom
Multiple R-squared:  0.2819,    Adjusted R-squared:  0.2803
F-statistic: 174.3 on 11 and 4886 DF,  p-value: < 2.2e-16
```

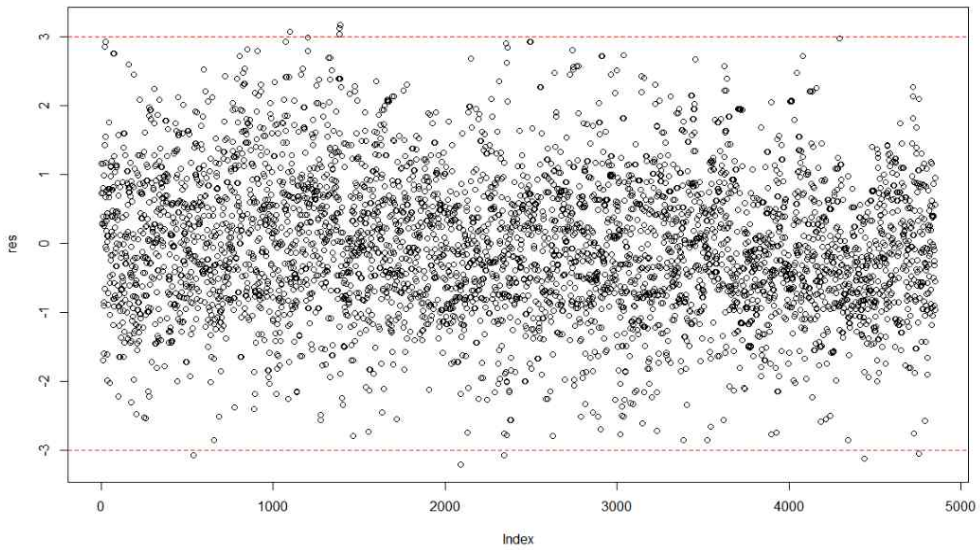


✓ 이분산성과 오차항의 상관성은 없으나 이상치가 존재하고 일부 관측치에서 레버지지값이 높음

```
### cleansing data
# outlier (studentized resiual check)
student_resid <- function(lm.fit, plot=F, limit=3){ # 스튜던트화 잔차
  res <- lm.fit$residuals/summary(lm.fit)$sigma
  attributes(res)$names <- NULL
  if(plot==T){
    plot(res)
    abline(h=c(-limit,limit), col="red", lty=2)
  }
  return(res)
}
par(mfrow=c(1,1))
student_resid(lm.fit, plot=T)
```

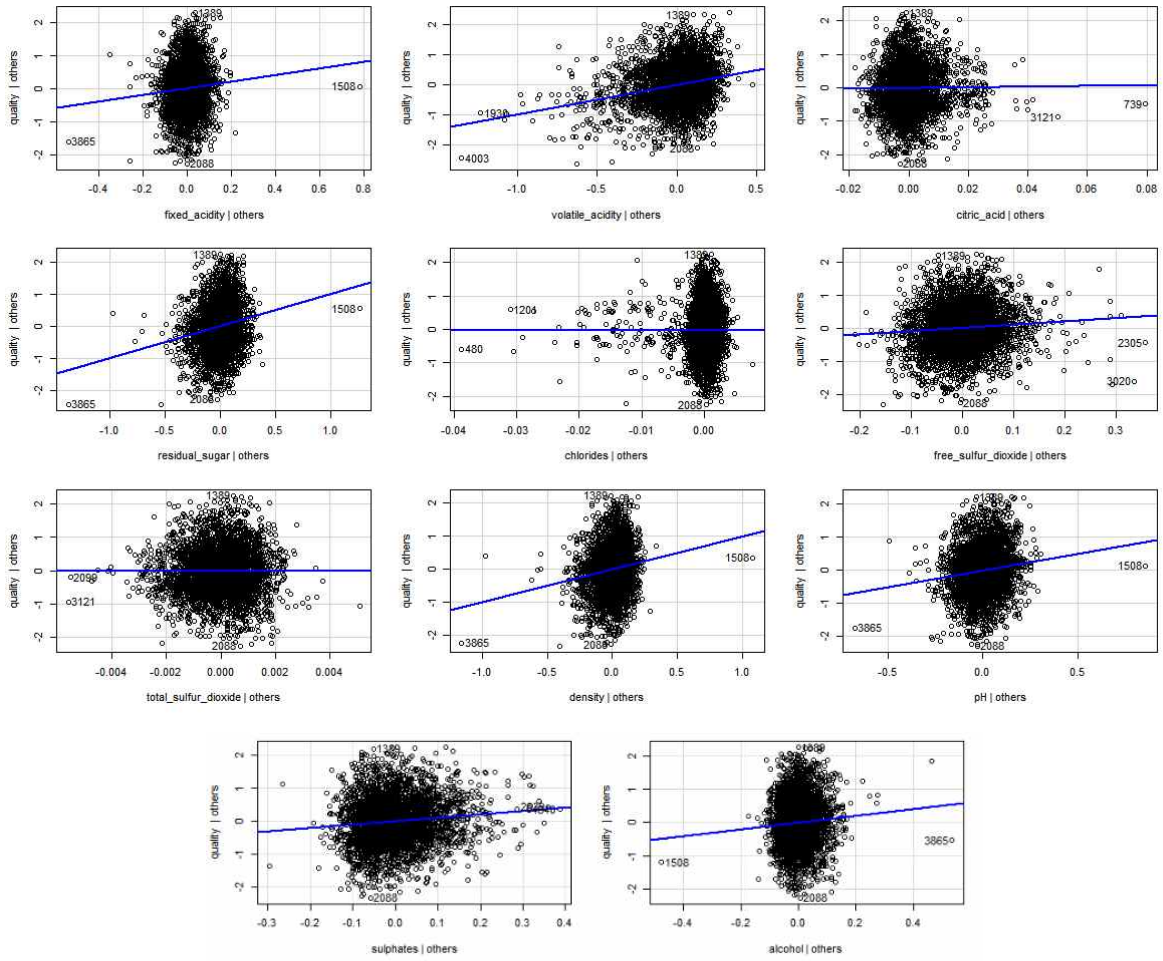



```
ind <- abs(student_resid(lm.fit))<=3; head(ind)
data_clean <- data[ind,]
student_resid(lm(quality~., data=data_clean), plot=T)
```



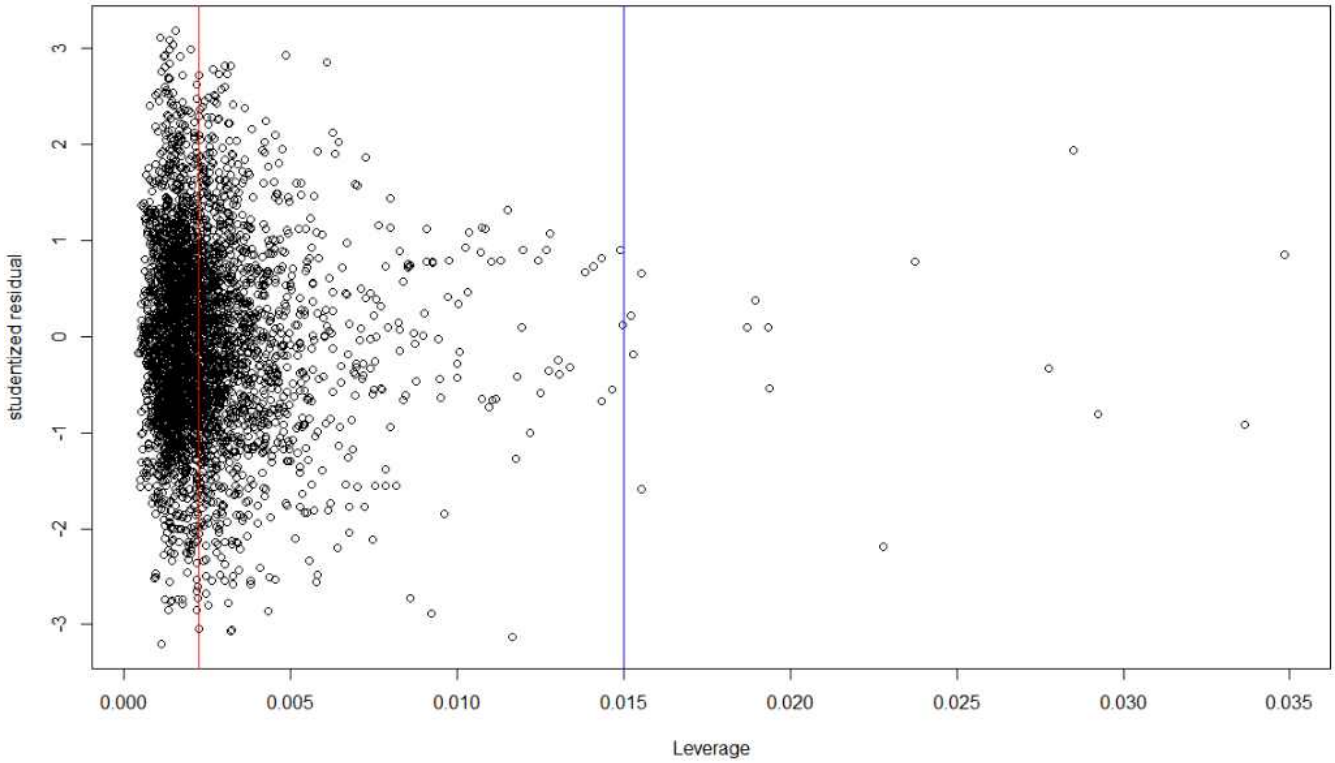
✓ 스튜던트화 잔차가 3이상인 값을 이상치로 간주하고 제거한후 이상치가 많이 제거되었다

```
# Leverage
lm.fit <- lm(quality~., data=data_clean)
leveragePlots(lm.fit)                          # 레버리지가 높은 값 식별
```




```
# leverage 플롯에서 각 변수마다 가질 수 있는 범주를 크게 벗어난 값을 수작업 제외
data_clean <- data_clean[-c(1508, 3865, 2305, 3020, 3121, 480), ]

par(mfrow=c(1,1))
lm.fit <- lm(quality~., data=data_clean)

# 레버리지 식별 함수
Leverage.High.index <- function(lm.fit, plot=F, limit=NULL){
  leverstat <- hatvalues(lm.fit)
  resid_stu <- student_resid(lm.fit, plot=F)
  line = (ncol(data_clean)-1)/nrow(data_clean)
  if(is.null(limit)) limit = line*4.5
  if(plot==T){
    plot(leverstat, resid_stu, xlab="Leverage",
         ylab="studentized residual")
    abline(v=c(line, limit), col=c("red", "blue"))
  }
  return(leverstat>=limit)
}
```

ind <- Leverage.High.index(lm.fit, plot=T, limit=0.015)



data_clean <- data_clean[!ind,]

Variance inflation factor(VIF)
vif(lm(quality~., data=data_clean)) # residual_sugar, density, alcohol
vif(lm(quality~.-density, data=data_clean))

변수명	density 제거 전 VIF	density 제거 후 VIF
fixed_acidity	3.464767	1.364229
volatile_acidity	1.124454	1.120335
citric_acid	1.173999	1.169529
residual_sugar	17.386302	1.466148
chlorides	1.269087	1.225856
free_sulfur_dioxide	1.787737	1.736469
total_sulfur_dioxide	2.294144	2.124300
density	43.053473	
pH	2.616953	1.337348
sulphates	1.177277	1.056517
alcohol	11.912997	1.684698

✓ 상관행렬을 보아 density는 VIF값이 높은 residual_sugar, alcohol, fixed_acidity와 correlation이 높음
✓ 두변수와 상관성이 높은 변수(density) 제거후 공산성 문제가 해결됨을 볼 수 있음

data_clean <- data[,-8]

linear regression

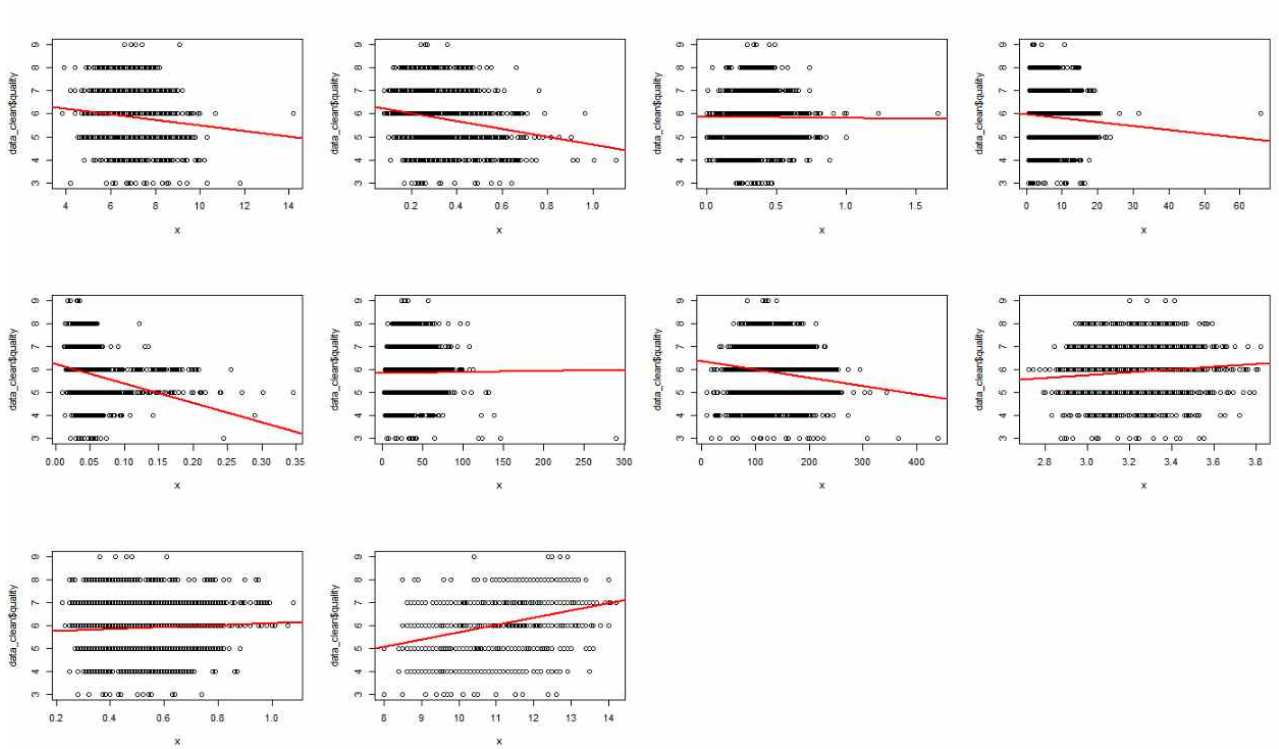
```
lm.fit <- lm(quality~., data=data_clean)
summary(lm.fit)
```

```
Residual standard error: 0.756 on 4887 degrees of freedom
Multiple R-squared:  0.2727,    Adjusted R-squared:  0.2713
F-statistic: 183.3 on 10 and 4887 DF,  p-value: < 2.2e-16
```

```
lm.fit_prod <- lm(quality~.*., data=data_clean)
summary(lm.fit_prod)
```

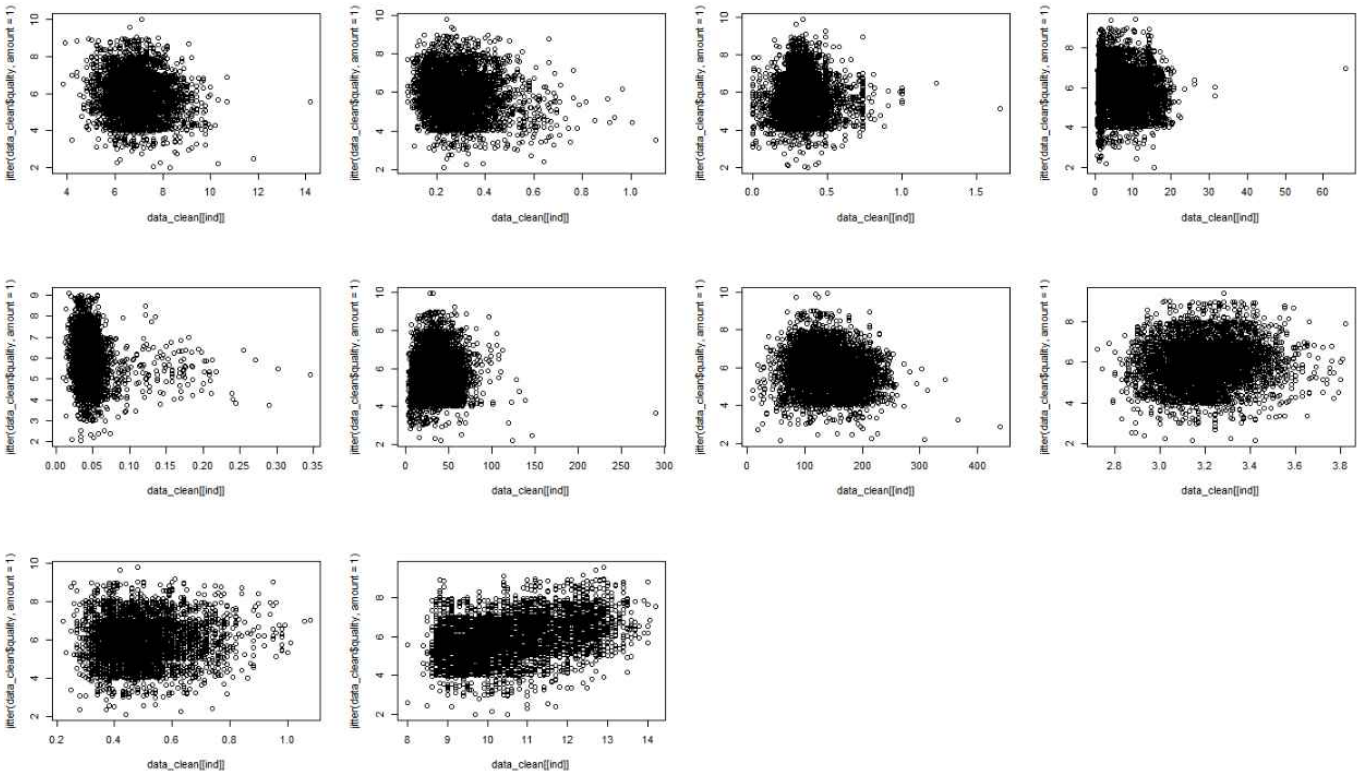
```
Residual standard error: 0.7227 on 4842 degrees of freedom
Multiple R-squared:  0.3415,    Adjusted R-squared:  0.3341
F-statistic: 45.66 on 55 and 4842 DF,  p-value: < 2.2e-16
```

```
# plot quality vs another variables
par(mfrow=c(3,4))
lapply(data_clean[,1:10], function(x){ #비선형 변환할 변수가 마땅히 보이지 않음
  lm.fit <- lm(data_clean$quality~x)
  plot(x,data_clean$quality)
  abline(lm.fit, col="red", lwd=2)
})
```



✓ Simple linear Regression

```
par(mfrow=c(3,4))
lapply(1:10, function(ind){
  plot(data_clean[[ind]], jitter(data_clean$quality,amount = 1))})
par(mfrow=c(1,1))
```

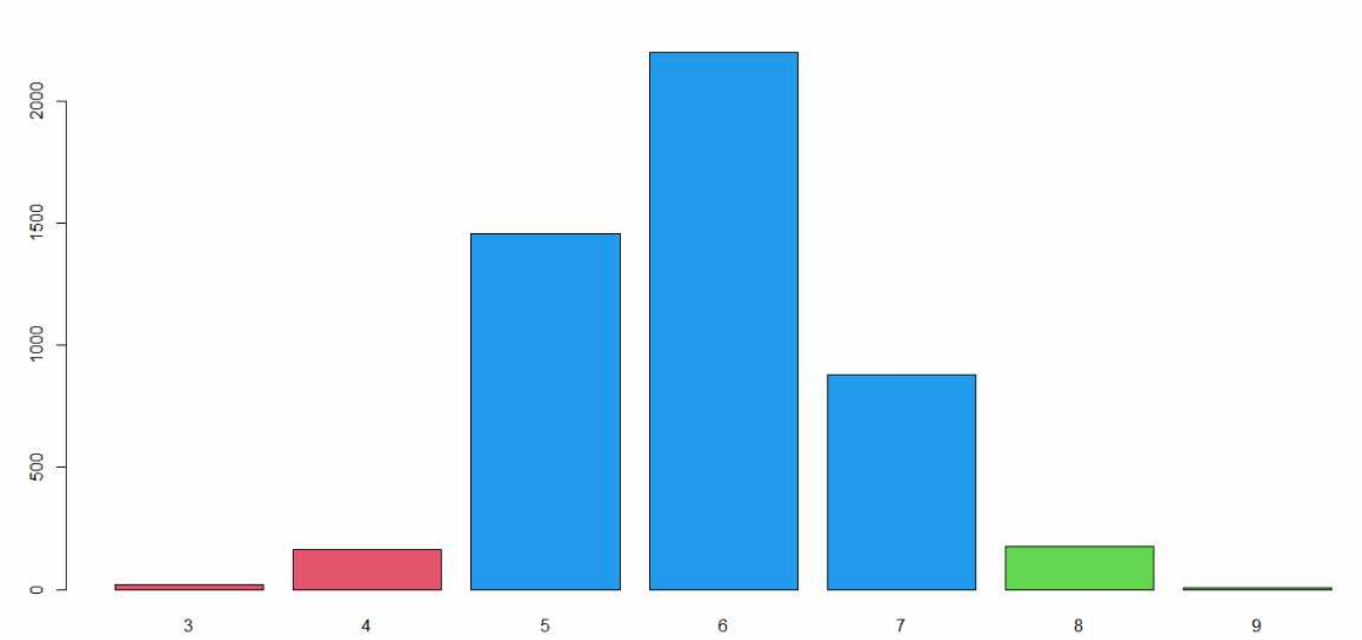


✓ 종속변수와 반응변수 사이에 특정 패턴이 없어서 비선형 변환은 하지 않는다.

✓ 이상치, 고(高) 레버리지 제거 및 공산성 제거 후에도 분산 설명 비율(R^2)이 작고 상호작용항을 넣은 모델역시 분산 설명 비율이 40% 아래이다. 이는 선형회귀보다는 다른 비모수적 방법이 더 설명력이 높을 것으로 보이므로 교차검증 및 추가 작업은 하지 않는다.

분류 (Classification) : Logistic Regression vs LDA vs QDA

```
data <- read_csv("winequality-white.csv")
names(data) %<>% str_replace_all(" ", "_") # for convenient
barplot(round(table(data$quality)),2, col=c(2,2,4,4,4,3,3), axes=T)
```



✓ 1~10의 정수 값을 가진 quality의 변수를 세가지 범주로 재조정하여 분류기법에 사용할 수 있는 방법 적용
✓ 이때 3~4의 값을 “low”, 5~7의 값을 “Middle”, 8~9의 값을 “High”로 설정한다.

```
ind3 <- data$quality<=4
ind2 <- data$quality>=5 & data$quality<=7
ind1 <- data$quality>=8

data$quality[ind3] <- "low"
data$quality[ind2] <- "middle"
data$quality[ind1] <- "high"
data$quality <- as.factor(data$quality)
```

```
> data %>% head(10)
# A tibble: 10 x 12
   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides free_sulfur_dioxi~ total_sulfur_dio~ density    pH sulphates alcohol quality
           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>            <dbl>            <dbl>   <dbl> <dbl>     <dbl>   <dbl> <fct>
 1           7              0.27        0.36           20.7     0.045               45              170    1.00   3          0.45     8.8 middle
 2           6.3            0.3         0.34            1.6     0.049               14              132    0.994  3.3        0.49     9.5 middle
 3           8.1            0.28        0.4             6.9     0.05                30               97    0.995  3.26       0.44    10.1 middle
 4           7.2            0.23        0.32            8.5     0.058               47              186    0.996  3.19       0.4      9.9 middle
 5           7.2            0.23        0.32            8.5     0.058               47              186    0.996  3.19       0.4      9.9 middle
 6           8.1            0.28        0.4             6.9     0.05                30               97    0.995  3.26       0.44    10.1 middle
 7           6.2            0.32        0.16            7       0.045               30              136    0.995  3.18       0.47     9.6 middle
 8           7              0.27        0.36           20.7     0.045               45              170    1.00   3          0.45     8.8 middle
 9           6.3            0.3         0.34            1.6     0.049               14              132    0.994  3.3        0.49     9.5 middle
10           8.1            0.22        0.43            1.5     0.044               28              129    0.994  3.22       0.45    11   middle
```

```
classification_score <- function(formula, train, test, label){

  multinom.fit <- multinom(formula = formula, data=train)
  lda.fit <- lda(formula=formula, data=train)
  qda.fit <- qda(formula=formula, data=train)

  multinom.pred <- predict(multinom.fit, test, type="class")
  lda.pred <- predict(lda.fit, test)$class
  qda.pred <- predict(qda.fit, test)$class

  multinom.score = mean(multinom.pred == label)
  lda.score = mean(lda.pred == label)
  qda.score = mean(qda.pred == label)

  return(c("Logistic"=multinom.score, "LDA"=lda.score, "QDA"=qda.score))
}

# split train, test data [0 : test, 1 : train]
ind <- sample(c(0,1), size=nrow(data), replace = T, prob=c(3,7))
train = data[ind==1, ]
test = data[ind==0, ]
classification_score(formula=quality~., train=train, test=test, label=test$quality)
```

```
> classification_score(formula=quality~., train=train, test=test, label=test$quality)
# weights:  39 (24 variable)
initial  value 3756.155415
iter  10 value 1234.605520
iter  20 value 932.699141
iter  30 value 920.917335
iter  40 value 920.104432
iter  50 value 919.985155
iter  60 value 919.974047
iter  70 value 919.930552
iter  80 value 919.863353
iter  90 value 918.912156
iter 100 value 918.080817
final  value 918.080817
stopped after 100 iterations


 Logistic       LDA       QDA
0.9263016 0.9168357 0.8803245
```

✓ Shapiro.test에서 각 클래스의 관측치들이 가우스 분포를 따르지 않으므로 예측대로 LDA와 QDA의 예측력은 로지스틱 방법보다 떨어졌다. 또 각 클래스마다 서로 다른 공분산행렬을 갖지않으므로 QDA보다 LDA의 예측력이 더 좋다.

● 참고1 : 각 클래스마다 공분산 행렬으로 상당히 유사함을 알 수 있다

```
round(cov(data[data$quality=="high",1:11]),2)
round(cov(data[data$quality=="middle",1:11]),2)
round(cov(data[data$quality=="low",1:11]),2)
```

```
# K-folds CV : 검정오차율 계산
CV_ErrorRate <- function(formula, data, K=5, labelname, plot=F){

  MSE <- matrix(0, nrow=K, ncol=3)
  data <- as.data.frame(data)
  colnames(MSE) <- c("Logistic", "LDA", "QDA")
  rownames(MSE) <- stringr::str_c("Fold[", 1:K, "]", sep="")
  ind <- cvFolds(n=nrow(data), K=K, R=1, type = "random")

  for(i in 1:K){
    train <- ind$subsets[ind$which!=i]
    test <- ind$subsets[ind$which==i]
    train <- data[train,]
    test <- data[test,]

    multinom.fit <- multinom(formula = formula, data=train)
    lda.fit <- lda(formula=formula, data=train)
    qda.fit <- qda(formula=formula, data=train)

    multinom.pred <- predict(multinom.fit, test, type="class")
    lda.pred <- predict(lda.fit, test)$class
    qda.pred <- predict(qda.fit, test)$class

    multinom.score = mean(multinom.pred != test[[labelname]])
    lda.score = mean(lda.pred != test[[labelname]])
    qda.score = mean(qda.pred != test[[labelname]])

    MSE[i,1] <- multinom.score
    MSE[i,2] <- lda.score
    MSE[i,3] <- qda.score
  }
  cat("\n\n")
  res <- colMeans(MSE)
  names(res) <- c("Logistic", "LDA", "QDA")

  if(plot==T){
    plot_data <- data.frame(melt(MSE))
    plot(plot_data %>% ggplot(mapping=aes(x=X1, y=value)) +
           geom_line(aes(group=X2, color=X2), size=1.2)  +
           xlab(label="fold") + ylab(label="Error_rate"))
  }

  return(list("Error_rate"=res, "ER_fold"=MSE))
}
```

Test of Data : Error Rate

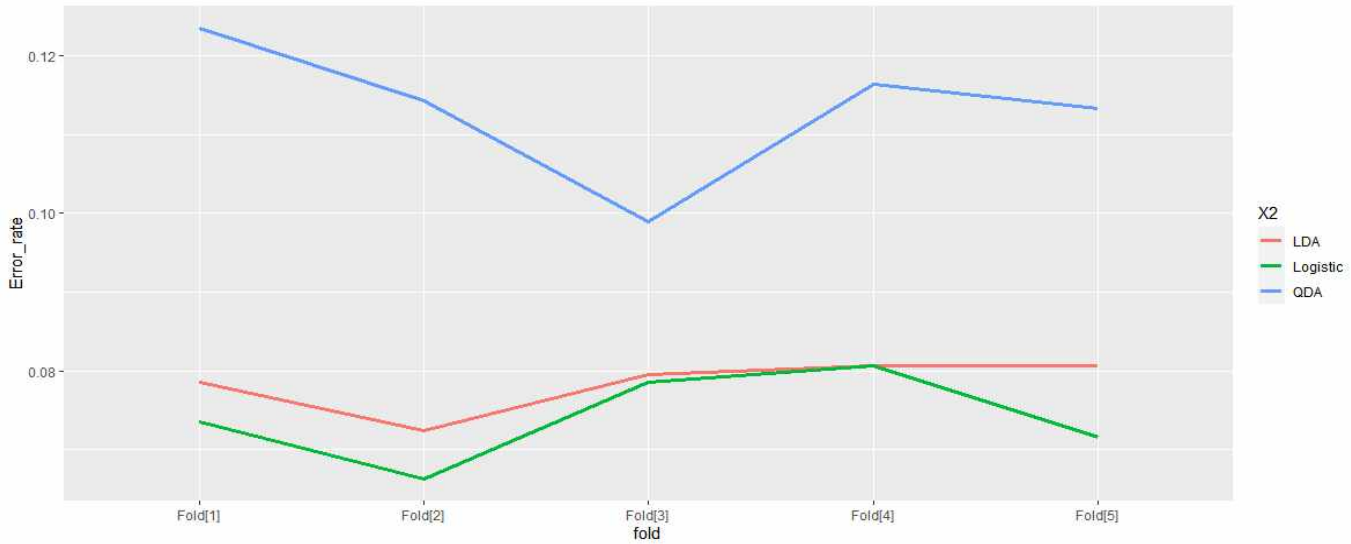
```
CV_ErrorRate(formula=quality~., data=data, K=5, labelname="quality", plot=T)   # K=5
CV_ErrorRate(formula=quality~., data=data, K=10, labelname="quality", plot=T)  # K=10
```

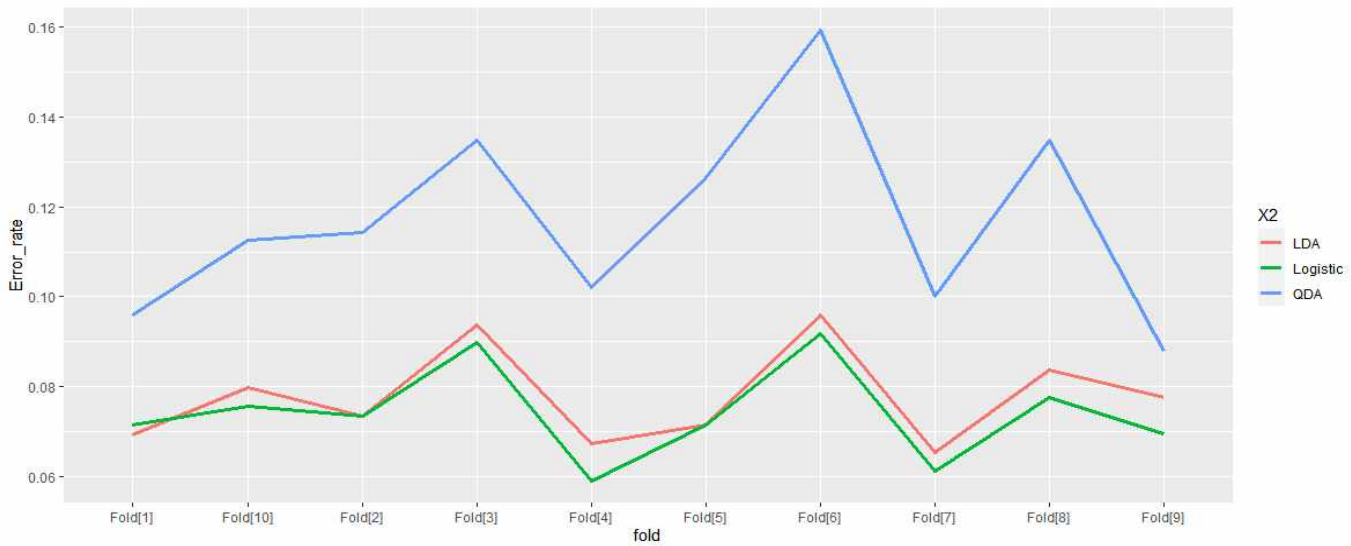
```
$Error_rate
  Logistic        LDA        QDA
0.07411269 0.07840028 0.11331221

$ER_fold
          Logistic        LDA        QDA
Fold[1] 0.07346939 0.07857143 0.12346939
Fold[2] 0.06632653 0.07244898 0.11428571
Fold[3] 0.07857143 0.07959184 0.09897959
Fold[4] 0.08069459 0.08069459 0.11644535
Fold[5] 0.07150153 0.08069459 0.11338100
```

```
$Error_rate
  Logistic        LDA        QDA
0.07411126 0.07778724 0.11677559

$ER_fold
           Logistic        LDA        QDA
Fold[1]  0.07142857 0.06938776 0.09591837
Fold[2]  0.07346939 0.07346939 0.11428571
Fold[3]  0.08979592 0.09387755 0.13469388
Fold[4]  0.05918367 0.06734694 0.10204082
Fold[5]  0.07142857 0.07142857 0.12653061
Fold[6]  0.09183673 0.09591837 0.15918367
Fold[7]  0.06122449 0.06530612 0.10000000
Fold[8]  0.07755102 0.08367347 0.13469388
Fold[9]  0.06952965 0.07770961 0.08793456
Fold[10] 0.07566462 0.07975460 0.11247444
```

✓ K=5와 K=10 일때의 오차율 : K의 수와 상관없이 비슷한 오차율을 나타낸다.

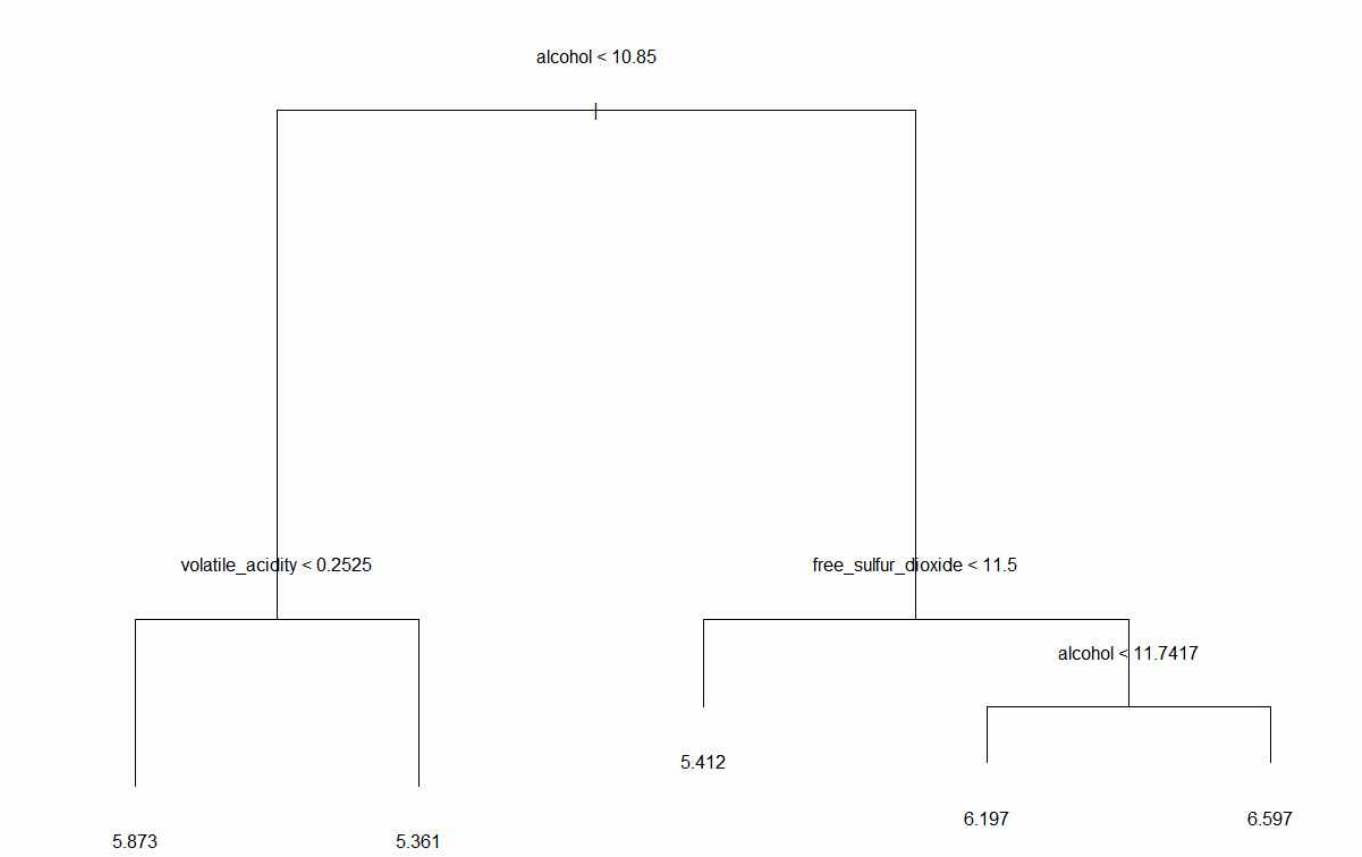
✓ 다만 오차율(Error Rate) 가 Logistic < LDA < QDA 이므로 앞서 보았던 것처럼 로지스틱을 사용하는 것이 좋아보인다.





트리 기반 모델 (Tree)

```
# Tree base
data <- read_csv("winequality-white.csv")
names(data) %<>% str_replace_all(" ", "_") # for convenient
tree.fit <- tree(quality~., data=data)
summary(tree.fit)
plot(tree.fit)
text(tree.fit)
```



✓ 전체데이터를 이용한 결과 점수(quality)를 결정하는데 alcohol, volatile_acidtiy, free_sulfur_dioxide 세 변수가 중요한 역할을 하는 것을 알 수 있다. 우선적으로 높은 점수를 받기위해서 알코올의 비율이 높을수록 유리하다고 볼 수 있다.

```
# Score of the prediction Using tree-based techniques for Classification

classification_score <- function(formula, train, test, label){

  tree.fit <<- tree(formula = formula ,data = train)
  bag.fit <<- randomForest(formula = formula ,data =train ,  mtry = ncol(train)-1)
  rf.fit <<- randomForest(formula = formula ,data =train ,  mtry = round(sqrt(ncol(train))))

  tree.pred <- predict(tree.fit ,test, type = 'class')
  bag.pred <- predict(bag.fit, test, type = 'class')
  rf.pred <- predict(rf.fit, test, type = 'class')

  tree.score = mean(tree.pred == label)
  bag.score = mean(bag.pred == label)
  rf.score = mean(rf.pred == label)

  return(c("Tree" = tree.score,
           "Bag" = bag.score,
           "Randomfrest" = rf.score))
}

# Split train, test data (0:test, 1:train)
lapply(1:10, function(num){
  ind <- sample(c(0,1), size=nrow(data), replace = T, prob=c(3,7))
  train = data[ind==1, ]
  test = data[ind==0, ]
  classification_score(formula=quality~., train=train, test=test, label=test$quality)
})
```

```
                [,1]      [,2]      [,3]      [,4]      [,5]
Tree       0.9226402 0.9219381 0.9310585 0.9322152 0.9346667
Bag        0.9425124 0.9421265 0.9428969 0.9440950 0.9460000
Randomfrest 0.9375444 0.9421265 0.9449861 0.9461915 0.9513333
                [,6]      [,7]      [,8]      [,9]     [,10]
Tree       0.9194769 0.9167245 0.9246717 0.9221843 0.9159262
Bag        0.9359945 0.9278279 0.9398756 0.9317406 0.9323308
Randomfrest 0.9373710 0.9271339 0.9371113 0.9365188 0.9323308
```

```
> table(data$quality)/nrow(data)

      high        low     middle
0.03674969 0.03736219 0.92588812
```

✓ 전체 middle의 비율이 92.59% 이므로 Tree를 제외한 NULL 분류기 (모든 예측을 middle)로 할 때보다는 좋은 성능을 보여준다.

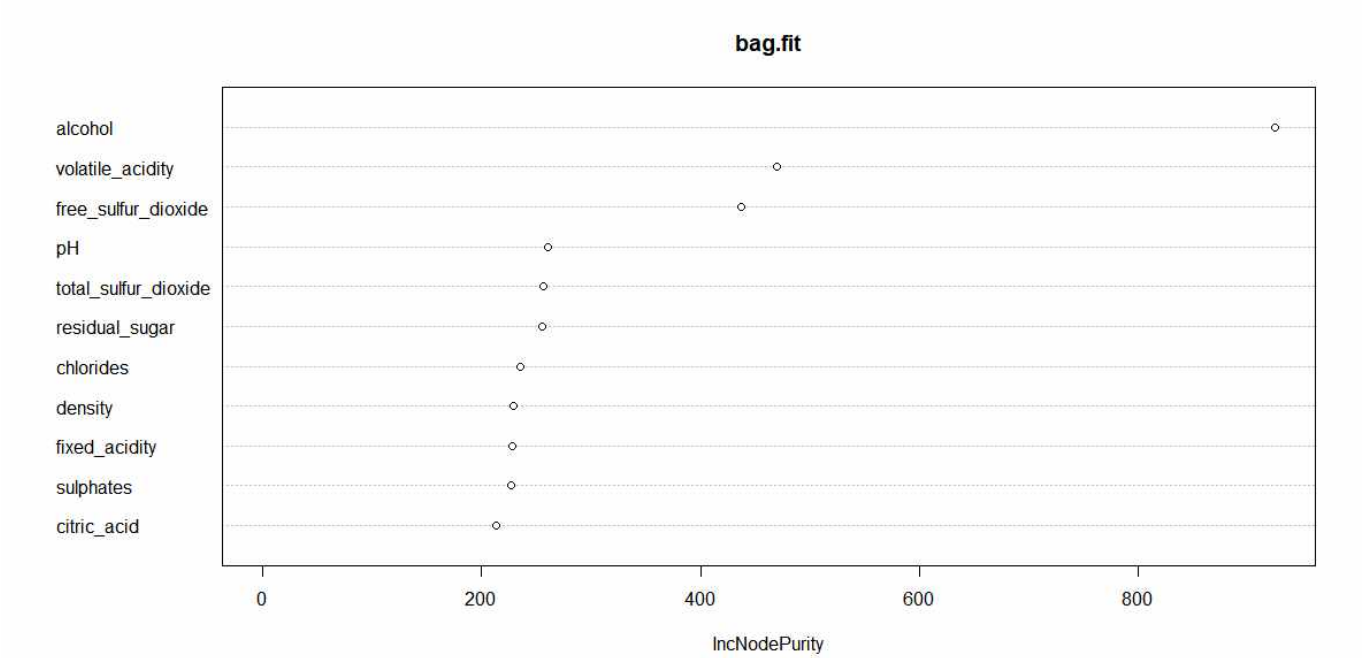
```
CV_ErrorRate <- function(formula, data, K=5, labelname, plot=F){

  MSE <- matrix(0, nrow=K, ncol=3)
  data <- as.data.frame(data)
  colnames(MSE) <- c("Tree","Bagging","Randomforest")
  rownames(MSE) <- stringr::str_c("Fold[", 1:K, "]", sep="")
  ind <- cvFolds(n=nrow(data), K=K, R=1, type = "random")

  for(i in 1:K){
    train <- ind$subsets[ind$which!=i]
    test <- ind$subsets[ind$which==i]
    train <- data[train,]
    test <- data[test,]

    tree.fit <- tree(formula = formula ,data = train)
    bag.fit <- randomForest(formula = formula ,data =train ,  mtry = ncol(train)-1)
    rf.fit = randomForest(formula = formula ,data =train ,  mtry = round(sqrt(ncol(train))))

    tree.pred <- predict(tree.fit ,test, type = 'class')
    bag.pred <- predict(bag.fit, test, type = 'class')
    rf.pred <- predict(rf.fit, test, type = 'class')

    tree.score = mean(tree.pred != test[[labelname]])
    bag.score = mean(bag.pred != test[[labelname]])
    rf.score = mean(rf.pred != test[[labelname]])

    MSE[i,1] <- tree.score
    MSE[i,2] <- bag.score
    MSE[i,3] <- rf.score
  }
  cat("\n\n")
  res <- colMeans(MSE)
  names(res) <- c("Tree","Bagging","Randomforest")

  if(plot==T){
    plot_data <- data.frame(melt(MSE))
    plot(plot_data %>% ggplot(mapping=aes(x=X1, y=value)) +
           geom_line(aes(group=X2, color=X2), size=1.2)  +
           xlab(label="fold") + ylab(label="Error_rate"))
  }

  return(list("Error_rate"=res, "ER_fold"=MSE))
}
```

CV_ErrorRate(formula=quality~., data=data, K=10, labelname="quality", plot=T)

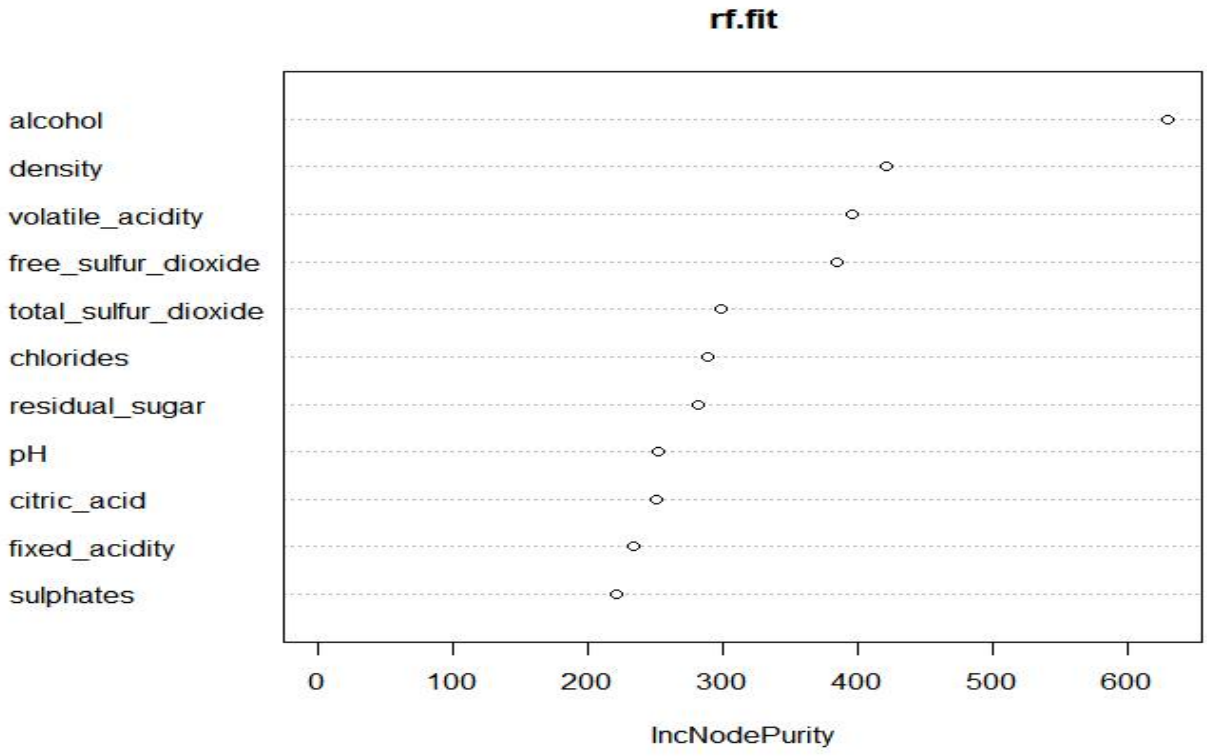
```
> CV_ErrorRate(formula=quality~., data=data, K=10, labelname="quality", plot=T)

$Error_rate
        Tree      Bagging Randomforest
  0.07451943   0.05573557   0.05430491

$ER_fold
                Tree    Bagging Randomforest
Fold[1]  0.06734694 0.04897959   0.04693878
Fold[2]  0.06938776 0.05306122   0.05306122
Fold[3]  0.08367347 0.06938776   0.06938776
Fold[4]  0.06326531 0.04693878   0.04489796
Fold[5]  0.09387755 0.06734694   0.06734694
Fold[6]  0.09183673 0.06326531   0.06122449
Fold[7]  0.06326531 0.04285714   0.04489796
Fold[8]  0.06734694 0.06122449   0.06122449
Fold[9]  0.06134969 0.04294479   0.03476483
Fold[10] 0.08384458 0.06134969   0.05930470
```

✓ 단일 Tree 기법은 NULL 분류기의 오차율(7.41%) 보다 높은 이유로 해당 기법으로 추정하는 것은 권장되지 않는다.

```
bag.fit = randomForest(formula = quality~. ,data =data ,  mtry = ncol(data)-1)
varImpPlot(bag.fit)
```




```
rf.fit = randomForest(quality~. ,data =data ,  mtry = round(sqrt(ncol(data))))
varImpPlot(rf.fit)
```



✓ 배깅에서는 alcohol, volatile_acidty가 모델 적합시 중요한 변수이며 랜덤포레스트에서는 alcohol, density 가 중요한 변수 역할을 하는 알 수 있으며 이것은 이전의 결과와 상당히 유사함을 알 수 있다.

```
# 부스팅 (boosting)
data <- read_csv("winequality-white.csv")
names(data) %<>% str_replace_all(" ", "_") # for convenient
shrinkage = c(1,0.1,0.01,0.001)
interaction.depth = c(1,2,4)
RSS <- matrix(0, 4, 3)
for(i in 1:4){
  for(j in 1:3){
    total <- 0
    for(k in 1:5){
      ind <- sample(c(0,1), size=nrow(data), replace = T, prob=c(3,7))
      train = data[ind==1, ]
      test = data[ind==0, ]
      boost.fit = gbm(quality~.,data = train, distribution = "gaussian",
                      n.tree = 15000, shrinkage = shrinkage[i],
                      interaction.depth = interaction.depth[j])
      boost.pred <- predict(boost.fit, test, type = 'response')
      total <- total + sum(boost.pred-test$quality)^2
    }
    RSS[i,j] <-total/5
  }
}
rownames(RSS) <- str_c("strinkage ", shrinkage, sep="")
colnames(RSS) <- str_c("depth", interaction.depth, sep="")
RSS
```

```
> RSS
                  depth1    depth2   depth4
shrinkage 1     1219.0950 2660.0479 2759.509
shrinkage 0.1    316.9381 2633.8778 1321.416
shrinkage 0.01   656.9854 1181.1664 1213.801
shrinkage 0.001  322.8502  405.6023  676.112
```

✓ 비교적 트리 깊이가 작고 shrinkage 가 작을수록 좋은 성능을 보여준다

서포트 백터 머신 (Support Vector Machine; SVM)

```
tuning = tune(svm, quality~., data = data , kernel = 'radial',
              ranges =  list(gamma = c(0.5,1,2,3),cost = c(10^(-1:3))))

tuning$best.parameters
tuning$best.model

bestmodel = tuning$best.model

svm.pred = predict(bestmodel,test)
mean(svm.pred == test$quality)
```

✓ 감마값과 비용값을 여러개 설정하여 최적화

```
> tuning$best.parameters
   gamma cost
10     1   10
> tuning$best.model

Call:
best.tune(method = svm, train.x = quality ~ ., data = data, ranges = list(gamma = c(0.5,
    1, 2, 3), cost = c(10^(-1:3))), kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  4054

>
> bestmodel = tuning$best.model
>
> svm.pred = predict(bestmodel,test)
> mean(svm.pred == test$quality)
[1] 0.9986667
```

✓ gamma = 1, cost = 10인 경우에 교차검증 오차율이 가장 낮았다.

✓ 해당 파라미터로 모델을 적합하고 테스트 데이터에 적용했을 때 99.87%의 적중률로

✓ 분류에 사용한 모든 기법 대비 매우 높은 적중률을 보여주었다.