# Breast Cancer Prediction from Cytopathology Data

## 소개

- The Breast Cancer (Wisconsin) Diagnosis dataset 은 세포핵의 30 가지 특성을 나타나는 특성치와 진단 데이터를 포함.
- 각각의 세포핵에 대해서 계산되어진 실수값을 갖는 10 가지 변수들
  - radius (중심에서 중심까지의 거리의 평균);
  - texture (그레이 스케일 값의 표준 편차);
  - perimeter( 둘레 );
  - area;
  - smoothness (반경 길이의 국부적 인 변화);
  - compactness (둘레 ^ 2 / area - 1.0);
  - concavity (윤곽의 오목한 부분의 정도);
  - concave points (윤곽의 오목한 부분의 수);
  - symmetry;
  - fractal dimension ( “해안선 근사”-1).
- 10 개의 변수에서 평균(mean), 표준오차(se), worst 또는 largest 3 가지를 계산해서 총 30 개의 특성치를 생성.
- 진단값을 위한 예측값들을 이해하기 위해서 특성들을 분석해보고, 여러 가지 다른 알고리즘으로 진단을 예측하는 모델 을 생성

## 준비물

- ggbiplot for representation of data grouped on category in the PCA transform components plane;
- ggplot2g for representation of data grouped on category in the plane defined by pairs of features;

```
#this Kernel uses ggbiplot from https://github.com/vqv/ggbiplot/

#
#  ggscreeplot.r
```

```
#
#  Copyright 2011 Vincent Q. Vu.
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301,
 USA.
#

#' Screeplot for Principal Components
#'
#' @param pcobj          an object returned by prcomp() or princomp()
#' @param type           the type of scree plot.  'pev' corresponds proportio
n of explained variance, i.e. the eigenvalues divided by the trace. 'cev' cor
responds to the cumulative proportion of explained variance, i.e. the partial
 sum of the first k eigenvalues divided by the trace.
#' @export
#' @examples
#'   data(wine)
#'   wine.pca <- prcomp(wine, scale. = TRUE)
#'   print(ggscreeplot(wine.pca))
#'
ggscreeplot <- function(pcobj, type = c('pev', 'cev'))
{
  type <- match.arg(type)
  d <- pcobj$sdev^2
  yvar <- switch(type,
                 pev = d / sum(d),
                 cev = cumsum(d) / sum(d))

  yvar.lab <- switch(type,
                     pev = 'proportion of explained variance',
                     cev = 'cumulative proportion of explained variance')

  df <- data.frame(PC = 1:length(d), yvar = yvar)

  ggplot(data = df, aes(x = PC, y = yvar)) +
    xlab('principal component number') + ylab(yvar.lab) +
```

```
    geom_point() + geom_path()
}

#
#  ggbiplot.r
#
#  Copyright 2011 Vincent Q. Vu.
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301,
 USA.
#

#' Biplot for Principal Components using ggplot2
#'
#' @param pcobj           an object returned by prcomp() or princomp()
#' @param choices         which PCs to plot
#' @param scale           covariance biplot (scale = 1), form biplot (scale =
 0). When scale = 1, the inner product between the variables approximates the
 covariance and the distance between the points approximates the Mahalanobis
distance.
#' @param obs.scale       scale factor to apply to observations
#' @param var.scale       scale factor to apply to variables
#' @param pc.biplot       for compatibility with biplot.princomp()
#' @param groups          optional factor variable indicating the groups that
 the observations belong to. If provided the points will be colored according
 to groups
#' @param ellipse         draw a normal data ellipse for each group?
#' @param ellipse.prob    size of the ellipse in Normal probability
#' @param labels          optional vector of labels for the observations
#' @param labels.size     size of the text used for the labels
#' @param alpha           alpha transparency value for the points (0 = transp
arent, 1 = opaque)
#' @param circle          draw a correlation circle? (only applies when prcom
p was called with scale = TRUE and when var.scale = 1)
#' @param var.axes        draw arrows for the variables?
#' @param varname.size    size of the text for variable names
#' @param varname.adjust  adjustment factor the placement of the variable nam
```

```r
es, >= 1 means farther from the arrow
#' @param varname.abbrev  whether or not to abbreviate the variable names
#'
#' @return                a ggplot2 plot
#' @export
#' @examples
#'   data(wine)
#'   wine.pca <- prcomp(wine, scale. = TRUE)
#'   print(ggbiplot(wine.pca, obs.scale = 1, var.scale = 1, groups = wine.cla
ss, ellipse = TRUE, circle = TRUE))
#'
ggbiplot <- function(pcobj, choices = 1:2, scale = 1, pc.biplot = TRUE,
                     obs.scale = 1 - scale, var.scale = scale,
                     groups = NULL, ellipse = FALSE, ellipse.prob = 0.68,
                     labels = NULL, labels.size = 3, alpha = 1,
                     var.axes = TRUE,
                     circle = FALSE, circle.prob = 0.69,
                     varname.size = 3, varname.adjust = 1.5,
                     varname.abbrev = FALSE, ...)
{
  library(ggplot2)
  library(plyr)
  library(scales)
  library(grid)

  stopifnot(length(choices) == 2)

  # Recover the SVD
 if(inherits(pcobj, 'prcomp')){
    nobs.factor <- sqrt(nrow(pcobj$x) - 1)
    d <- pcobj$sdev
    u <- sweep(pcobj$x, 2, 1 / (d * nobs.factor), FUN = '*')
    v <- pcobj$rotation
  } else if(inherits(pcobj, 'princomp')) {
    nobs.factor <- sqrt(pcobj$n.obs)
    d <- pcobj$sdev
    u <- sweep(pcobj$scores, 2, 1 / (d * nobs.factor), FUN = '*')
    v <- pcobj$loadings
  } else if(inherits(pcobj, 'PCA')) {
    nobs.factor <- sqrt(nrow(pcobj$call$X))
    d <- unlist(sqrt(pcobj$eig)[1])
    u <- sweep(pcobj$ind$coord, 2, 1 / (d * nobs.factor), FUN = '*')
    v <- sweep(pcobj$var$coord,2,sqrt(pcobj$eig[1:ncol(pcobj$var$coord),1]),F
UN="/")
  } else if(inherits(pcobj, "lda")) {
      nobs.factor <- sqrt(pcobj$N)
      d <- pcobj$svd
      u <- predict(pcobj)$x/nobs.factor
      v <- pcobj$scaling
```

```r
    d.total <- sum(d^2)
} else {
  stop('Expected a object of class prcomp, princomp, PCA, or lda')
}

# Scores
choices <- pmin(choices, ncol(u))
df.u <- as.data.frame(sweep(u[,choices], 2, d[choices]^obs.scale, FUN='*'))

# Directions
v <- sweep(v, 2, d^var.scale, FUN='*')
df.v <- as.data.frame(v[, choices])

names(df.u) <- c('xvar', 'yvar')
names(df.v) <- names(df.u)

if(pc.biplot) {
  df.u <- df.u * nobs.factor
}

# Scale the radius of the correlation circle so that it corresponds to
# a data ellipse for the standardized PC scores
r <- sqrt(qchisq(circle.prob, df = 2)) * prod(colMeans(df.u^2))^(1/4)

# Scale directions
v.scale <- rowSums(v^2)
df.v <- r * df.v / sqrt(max(v.scale))

# Change the labels for the axes
if(obs.scale == 0) {
  u.axis.labs <- paste('standardized PC', choices, sep='')
} else {
  u.axis.labs <- paste('PC', choices, sep='')
}

# Append the proportion of explained variance to the axis labels
u.axis.labs <- paste(u.axis.labs,
                     sprintf('(%0.1f%% explained var.)',
                             100 * pcobj$sdev[choices]^2/sum(pcobj$sdev^2)))

# Score Labels
if(!is.null(labels)) {
  df.u$labels <- labels
}

# Grouping variable
if(!is.null(groups)) {
  df.u$groups <- groups
```

```
}

# Variable Names
if(varname.abbrev) {
  df.v$varname <- abbreviate(rownames(v))
} else {
  df.v$varname <- rownames(v)
}

# Variables for text label placement
df.v$angle <- with(df.v, (180/pi) * atan(yvar / xvar))
df.v$hjust = with(df.v, (1 - varname.adjust * sign(xvar)) / 2)

# Base plot
g <- ggplot(data = df.u, aes(x = xvar, y = yvar)) +
        xlab(u.axis.labs[1]) + ylab(u.axis.labs[2]) + coord_equal()

if(var.axes) {
  # Draw circle
  if(circle)
  {
    theta <- c(seq(-pi, pi, length = 50), seq(pi, -pi, length = 50))
    circle <- data.frame(xvar = r * cos(theta), yvar = r * sin(theta))
    g <- g + geom_path(data = circle, color = muted('white'),
                       size = 1/2, alpha = 1/3)
  }

  # Draw directions
  g <- g +
    geom_segment(data = df.v,
                 aes(x = 0, y = 0, xend = xvar, yend = yvar),
                 arrow = arrow(length = unit(1/2, 'picas')),
                 color = muted('red'))
}

# Draw either labels or points
if(!is.null(df.u$labels)) {
  if(!is.null(df.u$groups)) {
    g <- g + geom_text(aes(label = labels, color = groups),
                       size = labels.size)
  } else {
    g <- g + geom_text(aes(label = labels), size = labels.size)
  }
} else {
  if(!is.null(df.u$groups)) {
    g <- g + geom_point(aes(color = groups), alpha = alpha)
  } else {
    g <- g + geom_point(alpha = alpha)
```

```
    }
  }

  # Overlay a concentration ellipse if there are groups
  if(!is.null(df.u$groups) && ellipse) {
    theta <- c(seq(-pi, pi, length = 50), seq(pi, -pi, length = 50))
    circle <- cbind(cos(theta), sin(theta))

    ell <- ddply(df.u, 'groups', function(x) {
      if(nrow(x) <= 2) {
        return(NULL)
      }
      sigma <- var(cbind(x$xvar, x$yvar))
      mu <- c(mean(x$xvar), mean(x$yvar))
      ed <- sqrt(qchisq(ellipse.prob, df = 2))
      data.frame(sweep(circle %*% chol(sigma) * ed, 2, mu, FUN = '+'),
                 groups = x$groups[1])
    })
    names(ell)[1:2] <- c('xvar', 'yvar')
    g <- g + geom_path(data = ell, aes(color = groups, group = groups))
  }

  # Label the variable axes
  if(var.axes) {
    g <- g +
    geom_text(data = df.v,
              aes(label = varname, x = xvar, y = yvar,
                  angle = angle, hjust = hjust),
              color = 'darkred', size = varname.size)
  }
  # Change the name of the legend for groups
  # if(!is.null(groups)) {
  #   g <- g + scale_color_brewer(name = deparse(substitute(groups)),
  #                               palette = 'Dark2')
  # }

  # TODO: Add a second set of axes

  return(g)
}

boxplot2g = function(x,y=NULL, groups = NULL, smooth = loess, smooth.args = l
ist(span = 0.1),  colv = NULL, alpha = 1, n = 360,...){

  prbs <- c(0.25,0.5,0.75)

  if(is.null(y)){
    stopifnot(ncol(x)==2)
```

```
  data <- as.data.frame(x)
}else{
  data <- as.data.frame(cbind(x,y))
}

if(is.null(groups)){
  data$groups <- as.factor(0)
}else{
  data$groups <- as.factor(groups)
}

labs <- names(data)
names(data) <- c("x","y","groups")
DM <- data.matrix(data)
#require(ggplot2)


# initiate the smoother
if(is.logical(smooth)){
  do.smooth <- smooth
}else{
  do.smooth <- TRUE
}
if(do.smooth){
  poss.args <- names(formals(smooth))
  spec.args <- names(smooth.args)


  ind <- match(spec.args, poss.args)
  for(i in seq_along(ind)){
    formals(smooth)[ind[i]] <- smooth.args[[i]]
  }
  if("span" %in% poss.args){
    formals(smooth)$span <- formals(smooth)$span/3
  }
}else{
  smooth <- NULL
}

phi = seq(360/n, 360, 360/n)/180*pi
e1 <- new.env()
e1$vectors <- cbind(sin(phi),cos(phi))

ntv <- nlevels(data$groups)
if(is.null(colv)){
  #print(ntv)
  if(ntv == 1){
    colv = 1
```

```
    }else{
      colv <- rainbow(ntv)
    }
  }

  e1$colv <- colv
  e1$lvls <- levels(data$groups)
  #colv <- colv[match(groups,levels(as.factor(data$groups)))]
  #e1$gp <- qplot(data$x, data$y, colour = data$groups)
  e1$gp <- ggplot(data=data,aes(x=x,y=y,colour=groups))+geom_point(alpha=alph
a)
  #print(formals(smooth))
  if(ntv == 1){
    groupbox2d(x=data,env=e1,prbs=prbs,smooth=smooth,do.smooth)
  }else{
    by(data,groups, groupbox2d, env= e1, prbs = prbs, smooth = smooth)
  }
  #e1$gp <- e1$gp  + opts(legend.position = "none")
  return(e1$gp)

}
groupbox2d = function( x, env, prbs, past, smooth){

  grp <- x[1,3]

  colid <- match(grp, env$lvls)

  if(any(colid)){
    colv <- env$colv[]
  }else{
    colv <- env$col[1]
  }
  xs <- x[,1:2]
  mm <- apply(xs,2,mean)
  xs <-  data.matrix(xs) - rep(mm,each=nrow(xs))

  S <- cov(xs)

  if (requireNamespace("MASS", quietly = TRUE)) {
    Sinv <- MASS::ginv(S)
    SSinv <- svd(Sinv)
    SSinv <- SSinv$u %*% diag(sqrt(SSinv$d))
    SS <- MASS::ginv(SSinv)
  }else{
    Sinv <- solve(S)
    SSinv <- svd(Sinv)
    SSinv <- SSinv$u %*% diag(sqrt(SSinv$d))
    SS <- solve(SSinv)
```

```
  }

  xs <- xs %*% SSinv


  prj <- xs %*% t(env$vectors)

  qut <- t(apply(prj,2, function(z){
    quarts <- quantile(z, probs = prbs)
    iqr <- quarts[3]-quarts[1]
    w1 <- min(z[which(z >= quarts[1] - 1.5*iqr)])
    #w2 <- max(z[which(z <= quarts[3] + 1.5*iqr)])
    #return(c(w1,quarts,w2))
    return(c(w1,quarts))
  }))
  #print(formals(smooth))
  if( !is.null(smooth) ){
    n <- nrow(qut)
    qut <- apply(qut,2,function(z){
      x <- 1:(3*n)
      z <- rep(z,3)
      ys <- predict(smooth(z~x))
      return(ys[(n+1):(2*n)])
    })
    #print(dim(qut))
  }


  ccBox <- env$vectors*qut[,2]
  md <- data.frame((env$vectors*qut[,3])%*%SS)
  md <- sapply(md,mean)+mm

  md[3] <- grp

  ccWsk <- env$vectors*qut[,1]

  ccc <- data.frame(rbind(ccBox,ccWsk) %*% SS + rep(mm,each=2*nrow(ccBox)))

  ccc$grp <- as.factor(rep(c("box","wsk"),each=nrow(ccBox)))
  ccc$groups <- factor(grp)
  md <- data.frame(md[1],md[2],grp)
  names(md) <- names(ccc)[-3]
  X1 <- NULL
  X2 <- NULL
  groups <- NULL
  #env$gp <- env$gp + geom_point(x=md[1],y=md[2],colour=md[3])
  env$gp <- env$gp + geom_point(data=md,aes(x=X1,y=X2, colour = groups),size=
5) +  scale_colour_manual(values = colv)
```

```
  env$gp <- env$gp + geom_path(data=ccc, aes(x=X1,y=X2,group=grp, colour = gr
oups), alpha = 1/8)
  env$gp <- env$gp + geom_polygon(data=ccc,aes(x=X1,y=X2,group=grp, colour =
groups, fill = groups), alpha = 1/8)
  env$gp <- env$gp + geom_point(data=md,aes(x=X1,y=X2),size=3,alpha=1,colour=
"white")
  env$gp <- env$gp + geom_point(data=md,aes(x=X1,y=X2),size=1,alpha=1)
  return( invisible(TRUE) )

}
```

## 데이터 읽기

```
setwd('D:/Work_Git/DeepMenia/part02/week1_180608')
raw.data <- read.csv("Breast_Cancer_Wisconsin_(Diagnostic)_Data_Set.csv")
print(sprintf("Number of data rows: %d",nrow(raw.data)))
```

```
## [1] "Number of data rows: 569"
```

```
print(sprintf("Number of data columns: %d",ncol(raw.data)))
```

```
## [1] "Number of data columns: 33"
```

## 데이터 탐색

```
summary(raw.data)
```

```
##        id            diagnosis  radius_mean      texture_mean
##  Min.   :     8670   B:357     Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   M:212     1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024             Median :13.370   Median :18.84
##  Mean   : 30371831             Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129             3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502             Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave.points_mean symmetry_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619
##  Median :0.06154   Median :0.03350     Median :0.1792
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040
##  fractal_dimension_mean   radius_se        texture_se      perimeter_se
```

```
##  Min.   :0.04996        Min.   :0.1115   Min.   :0.3602   Min.   : 0.757
##  1st Qu.:0.05770        1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606
##  Median :0.06154        Median :0.3242   Median :1.1080   Median : 2.287
##  Mean   :0.06280        Mean   :0.4052   Mean   :1.2169   Mean   : 2.866
##  3rd Qu.:0.06612        3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357
##  Max.   :0.09744        Max.   :2.8730   Max.   :4.8850   Max.   :21.980
##     area_se         smoothness_se      compactness_se      concavity_se
##  Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000
##  1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509
##  Median : 24.530   Median :0.006380   Median :0.020450   Median :0.02589
##  Mean   : 40.337   Mean   :0.007041   Mean   :0.025478   Mean   :0.03189
##  3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205
##  Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600
##  concave.points_se   symmetry_se       fractal_dimension_se
##  Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
##  1st Qu.:0.007638   1st Qu.:0.015160   1st Qu.:0.0022480
##  Median :0.010930   Median :0.018730   Median :0.0031870
##  Mean   :0.011796   Mean   :0.020542   Mean   :0.0037949
##  3rd Qu.:0.014710   3rd Qu.:0.023480   3rd Qu.:0.0045580
##  Max.   :0.052790   Max.   :0.078950   Max.   :0.0298400
##   radius_worst   texture_worst   perimeter_worst    area_worst
##  Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
##  1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
##  Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
##  Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
##  3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness_worst  compactness_worst concavity_worst  concave.points_worst
##  Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
##  Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
##  Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
##  3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
##  symmetry_worst   fractal_dimension_worst    X
##  Min.   :0.1565   Min.   :0.05504         Mode:logical
##  1st Qu.:0.2504   1st Qu.:0.07146         NA's:569
##  Median :0.2822   Median :0.08004
##  Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.6638   Max.   :0.20750
```

- diagnosis 필드는 B (beningn) 또는 M (malignant) 값을 갖음. 이 카테고리별로 환자수를 확인

```
diagnostic <- plyr::count(raw.data$diagnosis)
print(sprintf("Malignant: %d | Benign: %d",diagnostic$freq[2],diagnostic$freq
[1]))
```

```
## [1] "Malignant: 212 | Benign: 357"
```

```
print(sprintf("Percent of malignant tumor: %.1f%%",round(diagnostic$freq[2]/n
row(raw.data)*100,1)))
```

```
## [1] "Percent of malignant tumor: 37.3%"
```

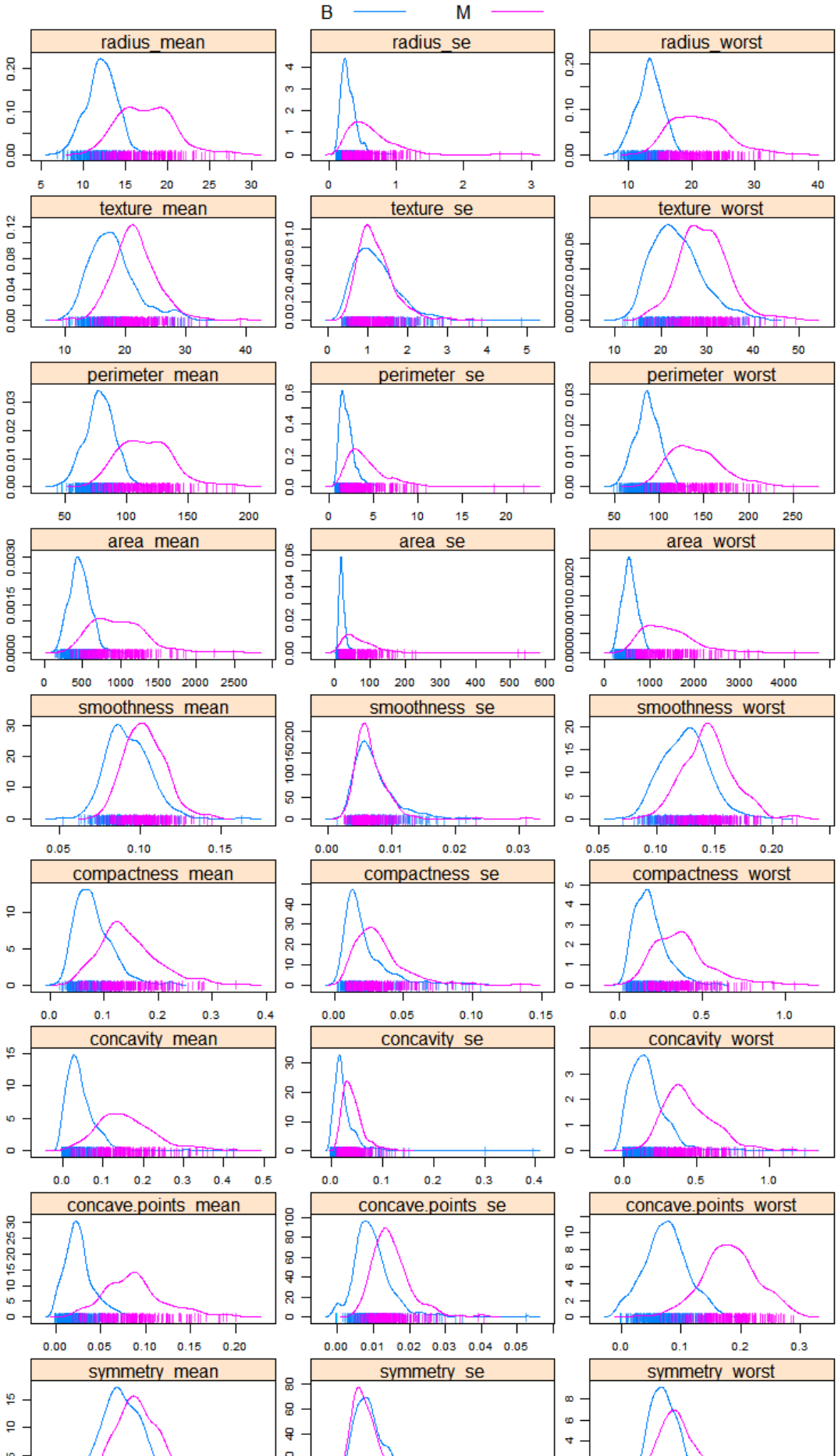
## Features plots

- id 와 diagnostic 필드는 제거
- NA 값이 많은 X 필드도 제거

```
newNames = c(
  "fractal_dimension_mean",  "fractal_dimension_se", "fractal_dimension_worst
",
  "symmetry_mean", "symmetry_se", "symmetry_worst",
  "concave.points_mean", "concave.points_se", "concave.points_worst",
  "concavity_mean","concavity_se", "concavity_worst",
  "compactness_mean", "compactness_se", "compactness_worst",
  "smoothness_mean", "smoothness_se", "smoothness_worst",
  "area_mean", "area_se", "area_worst",
  "perimeter_mean",  "perimeter_se", "perimeter_worst",
  "texture_mean" , "texture_se", "texture_worst",
  "radius_mean", "radius_se", "radius_worst"
)

bc.data = (raw.data[,newNames])
bc.diag = raw.data[,2]
```

### Feature density

```
if( !require(caret)  ) {
    install.packages('caret')
}
library(caret)

scales <- list(x=list(relation="free"),y=list(relation="free"), cex=0.6)
featurePlot(x=bc.data, y=bc.diag, plot="density",scales=scales,
            layout = c(3,10), auto.key = list(columns = 2), pch = "|")
```

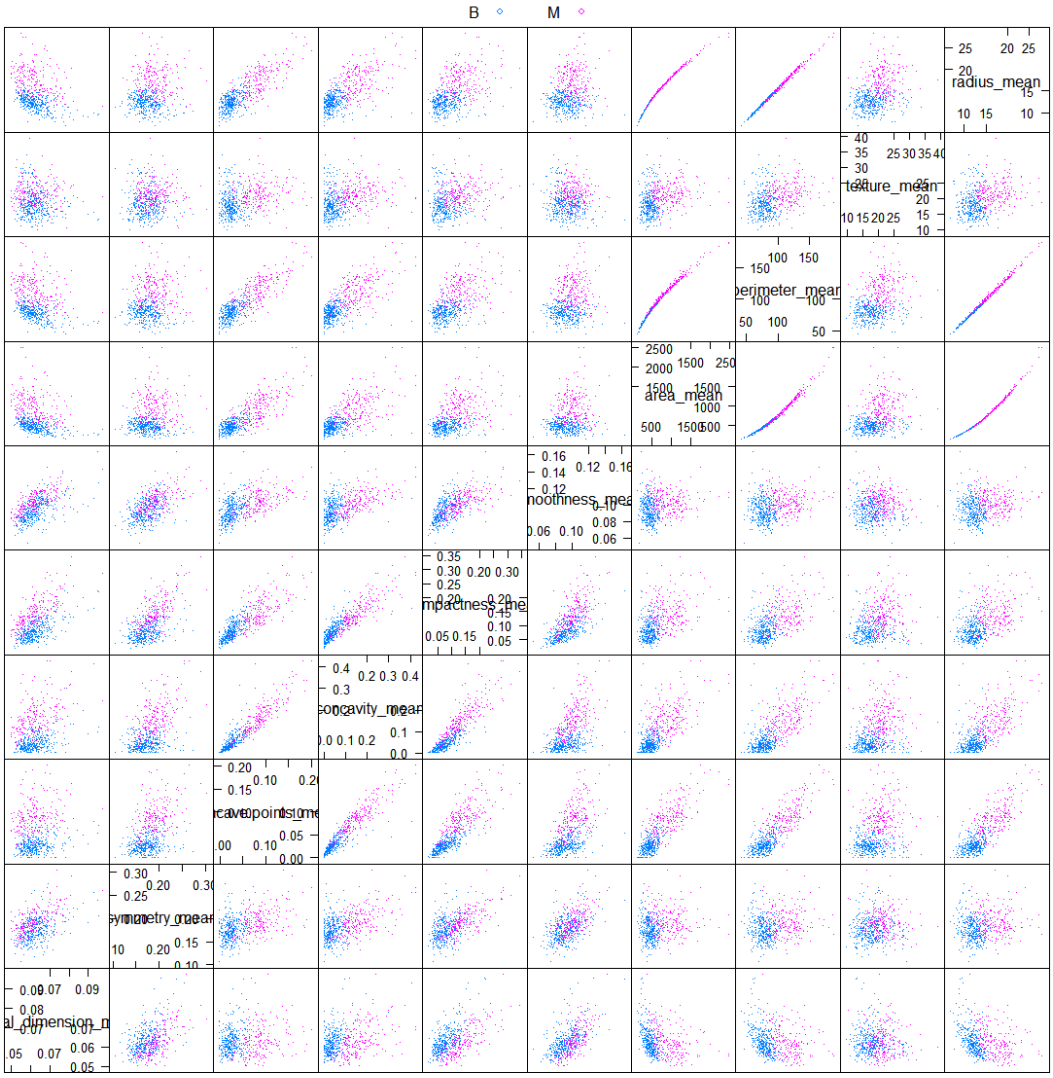

B
M
radius_mean
radius_se
radius_worst
texture_mean
texture_se
texture_worst
perimeter_mean
perimeter_se
perimeter_worst
area_mean
area_se
area_worst
smoothness_mean
smoothness_se
smoothness_worst
compactness_mean
compactness_se
compactness_worst
concavity_mean
concavity_se
concavity_worst
concave.points_mean
concave.points_se
concave.points_worst
symmetry_mean
symmetry_se
symmetry_worst

- concave.points_worst, concavity_worst, perimeter_worst, area_mean, perimeter_mean 변수가 종양 여부를 잘 구분.
- symmetry_se, smoothness_se 변수는 종양 여부를 구분하지 못함.

### Feature pairs

mean, se 및 worst 속성으로 그룹화된 Feature pairs 을 나타냅니다.

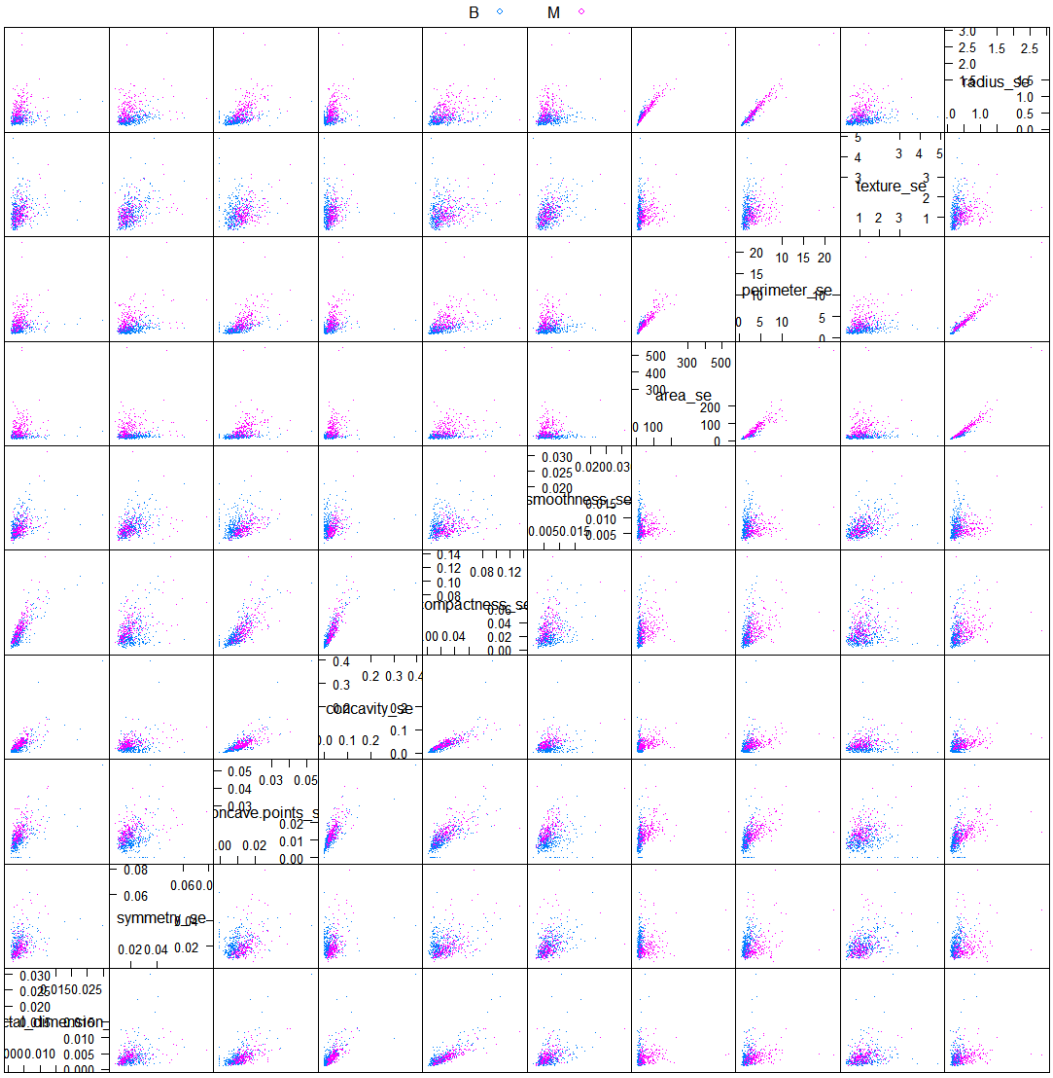
```
newNamesMean = c(
  "fractal_dimension_mean",
  "symmetry_mean",
  "concave.points_mean",
  "concavity_mean",
  "compactness_mean",
  "smoothness_mean",
  "area_mean",
  "perimeter_mean",
  "texture_mean" ,
  "radius_mean"
)

bcM.data = (raw.data[,newNamesMean])
bcM.diag = raw.data[,2]

scales <- list(x=list(relation="free"),y=list(relation="free"), cex=0.4)
featurePlot(x=bcM.data, y=bcM.diag, plot="pairs",scales=scales,
         auto.key = list(columns = 2), pch=".")
```




Scatter Plot Matrix

```
newNamesSE = c(
  "fractal_dimension_se",
  "symmetry_se",
  "concave.points_se",
  "concavity_se",
  "compactness_se",
  "smoothness_se",
  "area_se",
  "perimeter_se",
  "texture_se" ,
  "radius_se"
)

bcSE.data = (raw.data[,newNamesSE])
bcSE.diag = raw.data[,2]
```

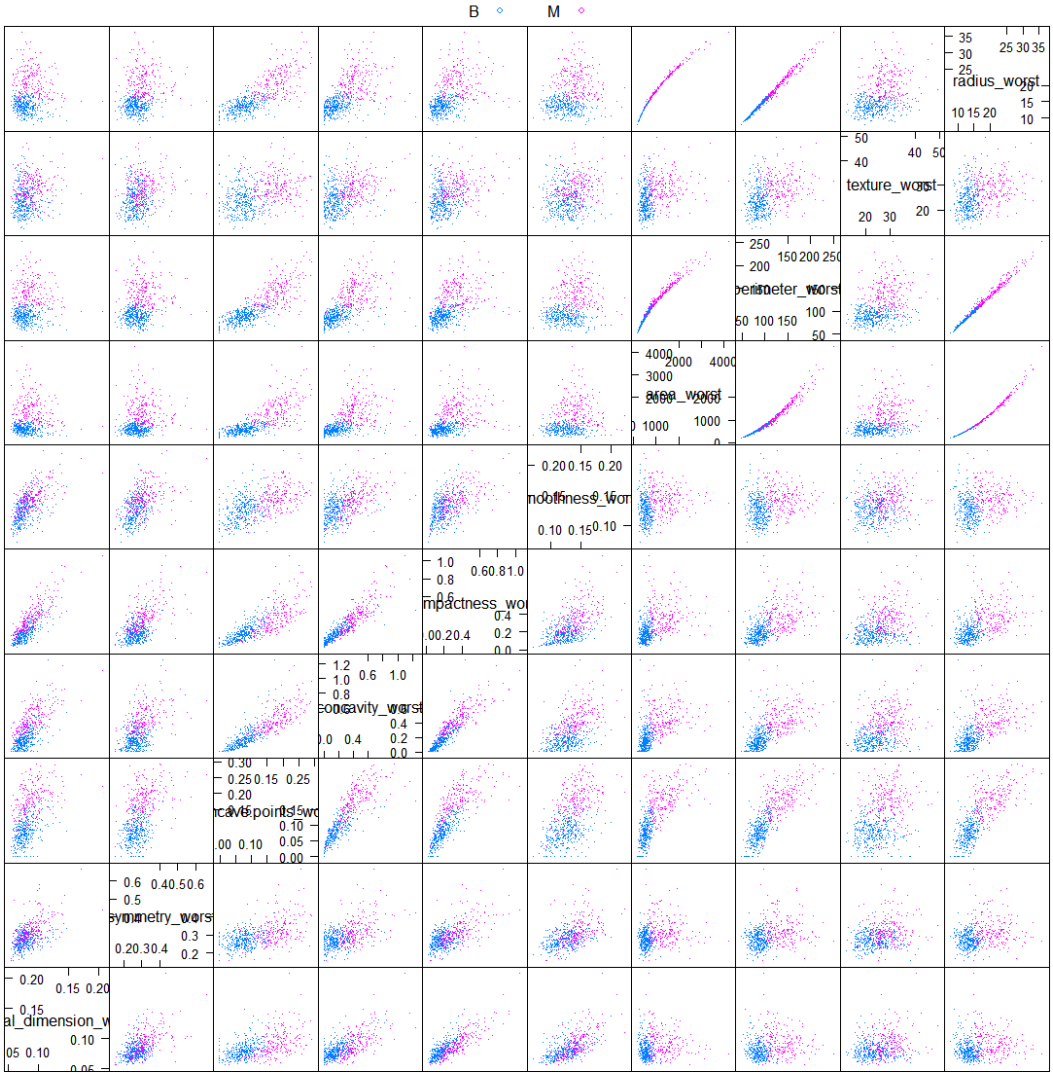
```
scales <- list(x=list(relation="free"),y=list(relation="free"), cex=0.4)
featurePlot(x=bcSE.data, y=bcSE.diag, plot="pairs",scales=scales,
         auto.key = list(columns = 2), pch=".")
```




Scatter Plot Matrix

```
newNamesW = c(
  "fractal_dimension_worst",
  "symmetry_worst",
  "concave.points_worst",
  "concavity_worst",
  "compactness_worst",
  "smoothness_worst",
  "area_worst",
  "perimeter_worst",
  "texture_worst" ,
  "radius_worst"
```

```
)

bcW.data = (raw.data[,newNamesW])
bcW.diag = raw.data[,2]

scales <- list(x=list(relation="free"),y=list(relation="free"), cex=0.4)
featurePlot(x=bcW.data, y=bcW.diag, plot="pairs",scales=scales,
         auto.key = list(columns = 2), pch=".")
```



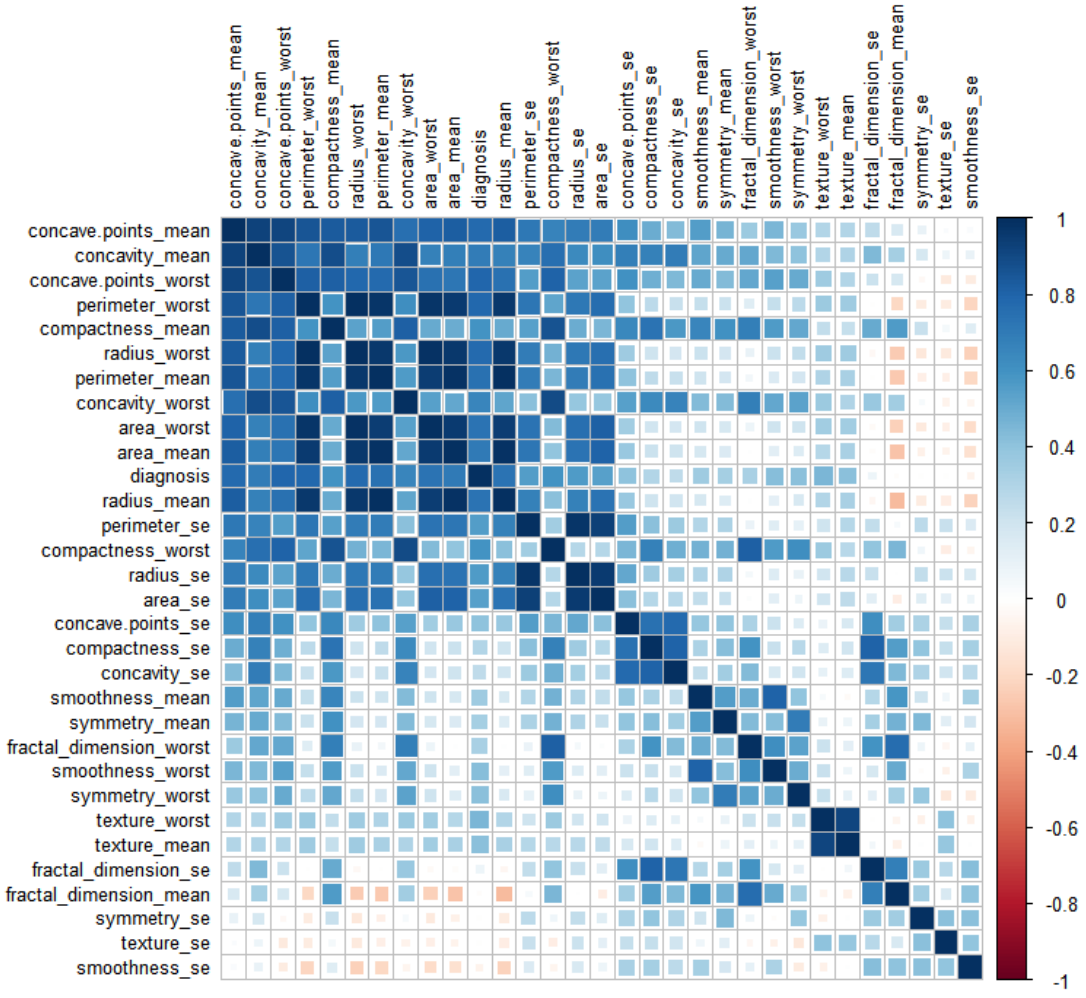

Scatter Plot Matrix

## Pearson correlation

피어슨 상관을 조사

```
nc=ncol(raw.data)
df <- raw.data[,3:nc-1]
```

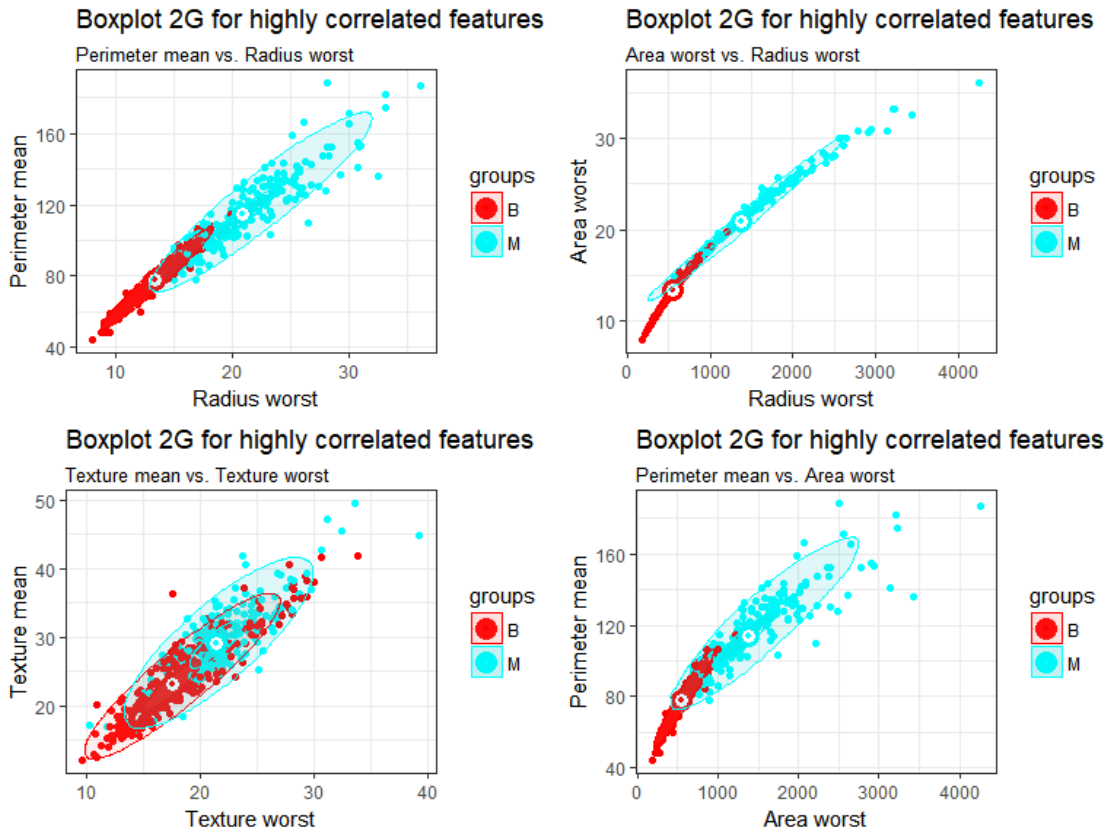
```
df$diagnosis <- as.integer(factor(df$diagnosis))-1
correlations <- cor(df,method="pearson")
corrplot(correlations, number.cex = .9, method = "square",
         hclust.method = "ward", order = "FPC",
         type = "full", tl.cex=0.8,tl.col = "black")
```




- 가장 높은 상관계수 are between:
  - perimeter_mean and radius_worst;
  - area_worst and radius_worst;
  - perimeter_worst and radius_worst, perimeter_mean, area_worst, area_mean, radius_mean;
  - texture_mean and texture_worst;

## Highly correlated pairs

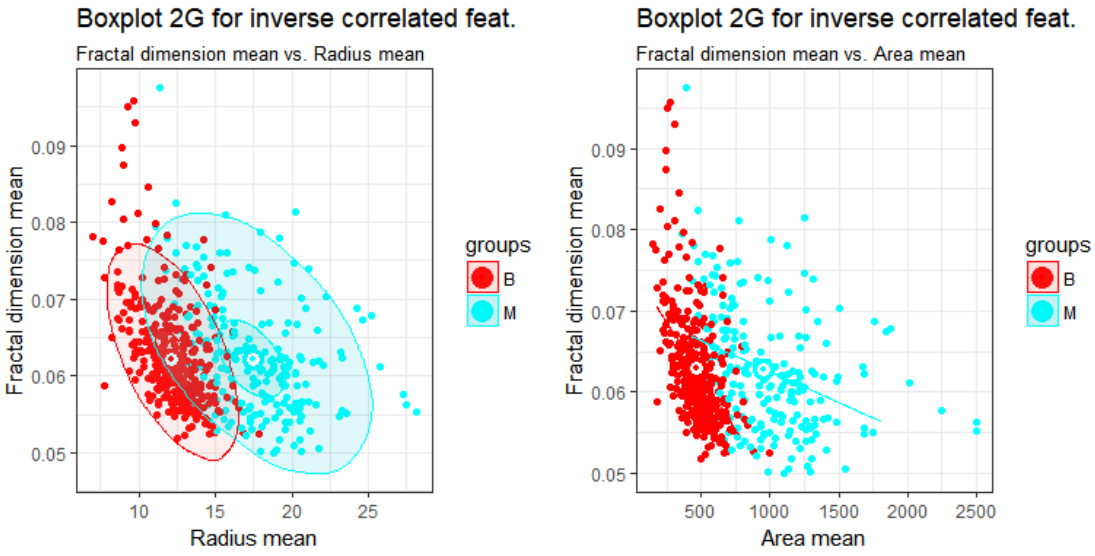
```
b1 <- boxplot2g(bc.data$radius_worst, bc.data$perimeter_mean, bc.diag, smooth
 = loess, NULL, NULL) +
  labs(title="Boxplot 2G for highly correlated features", subtitle = "Perimet
er mean vs. Radius worst", x="Radius worst", y="Perimeter mean") + theme_bw()

b2 <- boxplot2g(bc.data$area_worst, bc.data$radius_worst, bc.diag, smooth = l
oess, NULL, NULL) +
  labs(title="Boxplot 2G for highly correlated features", subtitle = "Area wo
rst vs. Radius worst", x="Radius worst", y="Area worst") + theme_bw()

b3 <- boxplot2g(bc.data$texture_mean, bc.data$texture_worst, bc.diag, smooth
= loess, NULL, NULL) +
  labs(title="Boxplot 2G for highly correlated features", subtitle = "Texture
 mean vs. Texture worst", x="Texture worst", y="Texture mean") + theme_bw()

b4 <- boxplot2g(bc.data$area_worst, bc.data$perimeter_mean, bc.diag, smooth =
 loess, NULL, NULL) +
  labs(title="Boxplot 2G for highly correlated features", subtitle = "Perimet
er mean vs. Area worst", x="Area worst", y="Perimeter mean") + theme_bw()

grid.arrange(b1, b2, b3, b4, ncol=2)
```



- 상관 관계가 있는 쌍중 일부가 종양의 양성 여부를 잘 구분하고 있음.

### Inverse correlated pairs

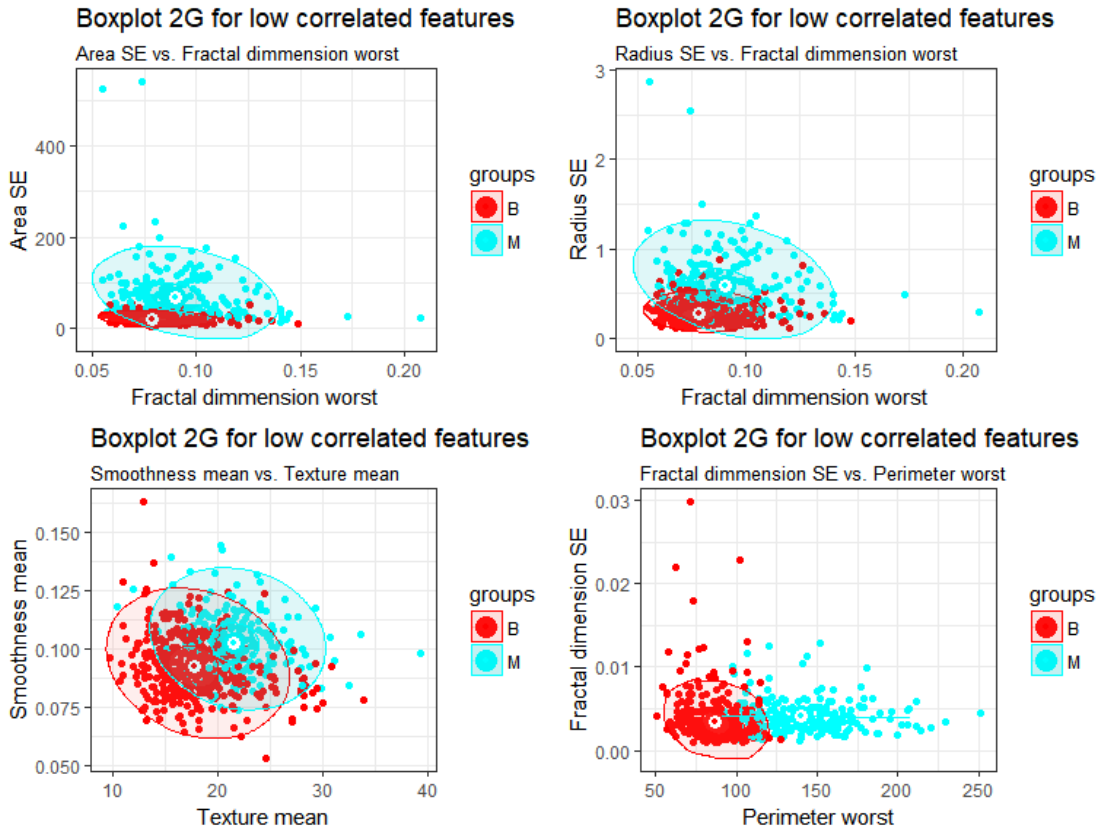
```
b5 <- boxplot2g(bc.data$radius_mean, bc.data$fractal_dimension_mean, bc.diag,
 smooth = loess, NULL, NULL) +
  labs(title="Boxplot 2G for inverse correlated feat.", subtitle = "Fractal d
imension mean vs. Radius mean", x="Radius mean", y="Fractal dimension mean")
+ theme_bw()

b6 <- boxplot2g(bc.data$area_mean, bc.data$fractal_dimension_mean, bc.diag, s
mooth = loess, NULL, NULL) +
  labs(title="Boxplot 2G for inverse correlated feat.", subtitle = "Fractal d
imension mean vs. Area mean", x="Area mean", y="Fractal dimension mean") + th
eme_bw()

grid.arrange(b5, b6, ncol=2)
```




### Low correlated pairs

```
b9 <- boxplot2g(bc.data$fractal_dimension_worst, bc.data$area_se, bc.diag, sm
ooth = loess, NULL, NULL) +
  labs(title="Boxplot 2G for low correlated features", subtitle = "Area SE vs.
 Fractal dimmension worst", x="Fractal dimmension worst", y="Area SE") + them
e_bw()

b10 <- boxplot2g(bc.data$fractal_dimension_worst, bc.data$radius_se, bc.diag,
 smooth = loess, NULL, NULL) +
  labs(title="Boxplot 2G for low correlated features", subtitle = "Radius SE
vs. Fractal dimmension worst", x="Fractal dimmension worst", y="Radius SE") +
 theme_bw()
```

```
b11 <- boxplot2g(bc.data$texture_mean, bc.data$smoothness_mean, bc.diag, smoo
th = loess, NULL, NULL) +
  labs(title="Boxplot 2G for low correlated features", subtitle = "Smoothness
 mean vs. Texture mean", x="Texture mean", y="Smoothness mean") + theme_bw()

b12 <- boxplot2g(bc.data$perimeter_worst, bc.data$fractal_dimension_se, bc.di
ag, smooth = loess, NULL, NULL) +
  labs(title="Boxplot 2G for low correlated features", subtitle = "Fractal di
mmension SE vs. Perimeter worst", x="Perimeter worst", y="Fractal dimension S
E") + theme_bw()

grid.arrange(b9, b10, b11, b12, ncol=2)
```



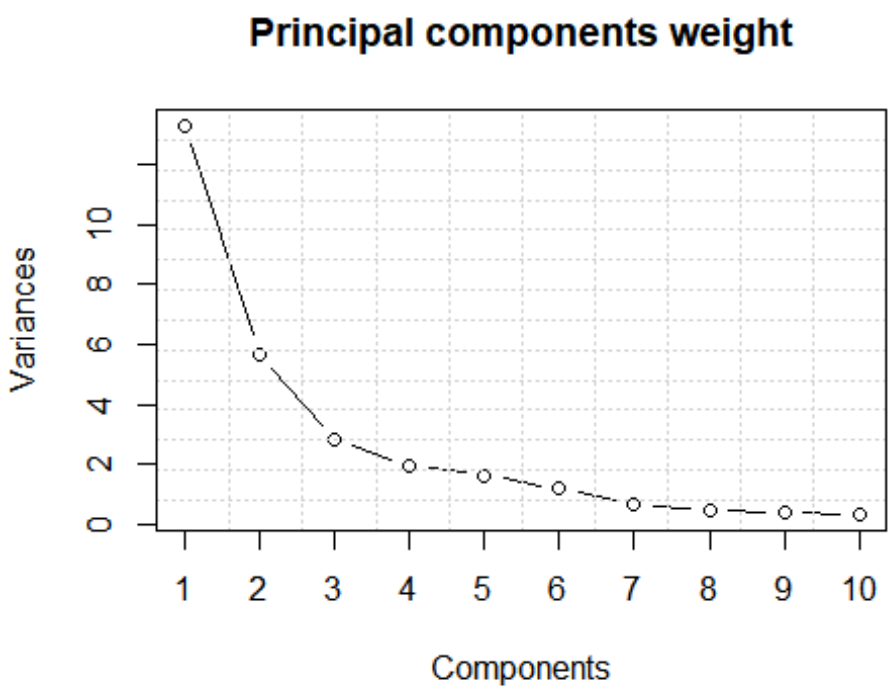

- 낮은 상관관계인 변수라도 종양 여부를 구분하지 못하는 경우가 있는 것을 확인.( fractal_dimension_worst <-> area_se)
- 종양여부를 잘 구분하는 경우 ( perimeter_worst <-> fractal_dimension_se )

## Principal Components Analysis (PCA) transform

- diagnosis 과 X 컬럼은 제거

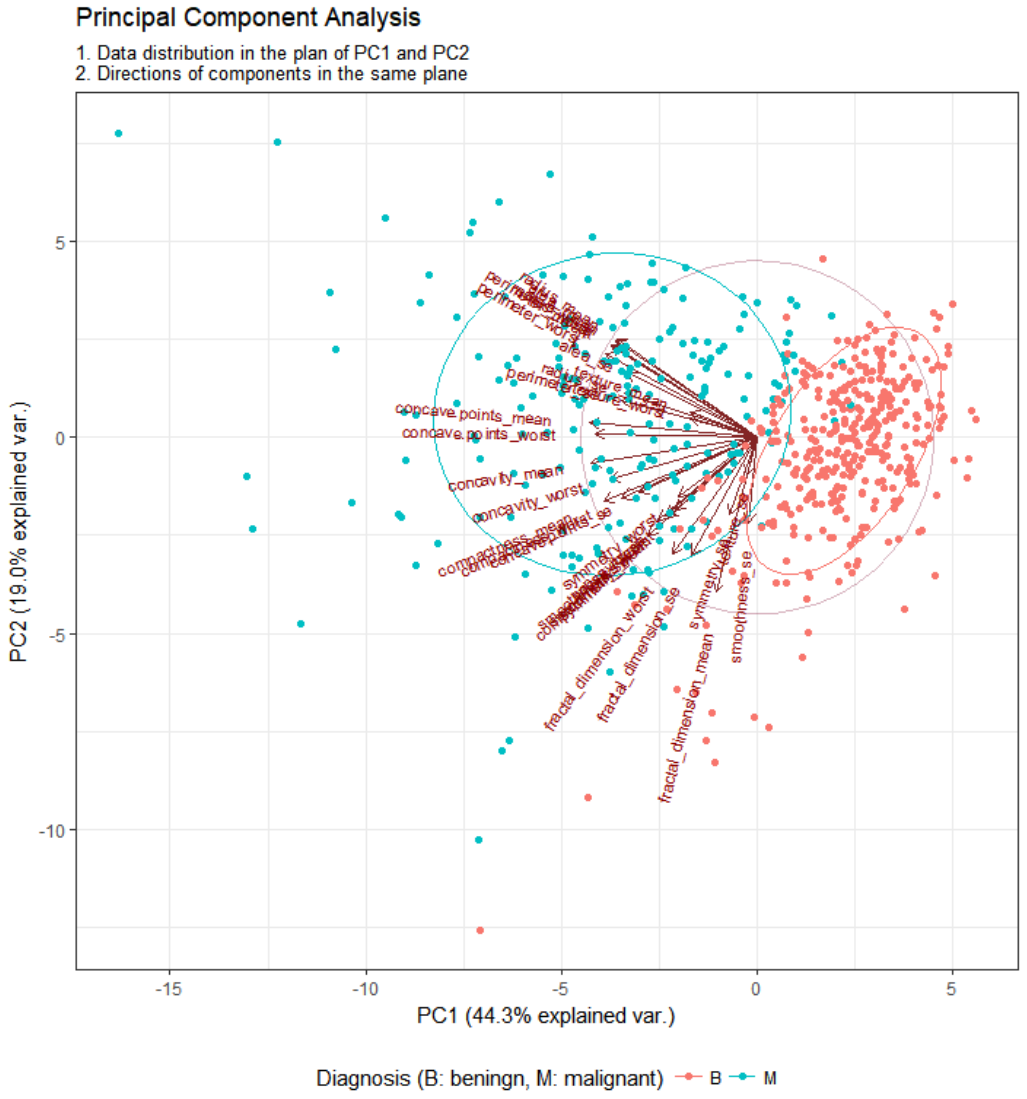
```
bc.pca <- prcomp(bc.data, center=TRUE, scale.=TRUE)
plot(bc.pca, type="l", main='')
```

```
grid(nx = 10, ny = 14)
title(main = "Principal components weight", sub = NULL, xlab = "Components")
box()
```




- 2 개의 주성분을 평면에 투영
- 두 개의 elips 는 진단의 두 그룹인 B 와 M 의 분포에 대해 0.68 확률 경계를 보여줌.
- A circle superposed over the scatter plot data helps to evaluate the relative ratio between the features in the most important principal components plane.
- 화살표로 그어진 피처는 가장 높은 분산을 가진 피처.

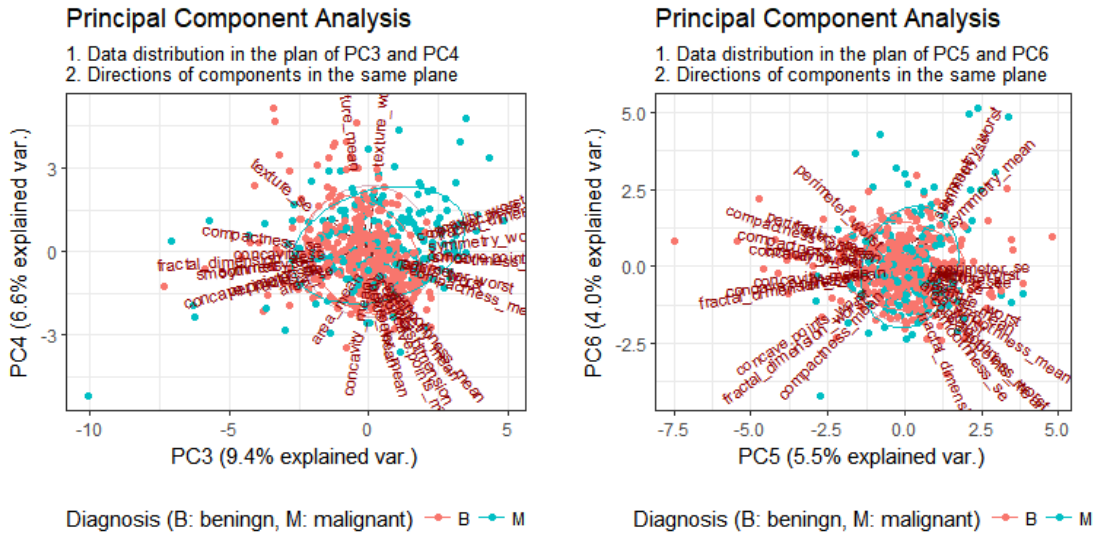
```
ggbiplot(bc.pca, choices=1:2, obs.scale = 1, var.scale = 1, groups = bc.diag,

  ellipse = TRUE, circle = TRUE, varname.size = 3, ellipse.prob = 0.68, circl
e.prob = 0.69) +
  scale_color_discrete(name = 'Diagnosis (B: beningn, M: malignant)') + theme
_bw() +
  labs(title = "Principal Component Analysis",
  subtitle = "1. Data distribution in the plan of PC1 and PC2\n2. Directions
of components in the same plane") +
  theme(legend.direction = 'horizontal', legend.position = 'bottom')
```




- 앞의 2 개의 주성분이 전체 변화의 63.3%설명
- {PC3, PC4} 와 {PC5, PC6} 주성분에서 대해서 투영.

```
pc34<- ggbiplot(bc.pca, choices=3:4, obs.scale = 1, var.scale = 1, groups = b
c.diag,
        ellipse = TRUE, circle = TRUE, varname.size = 3, ellipse.prob = 0.68,
 circle.prob = 0.69) +
        scale_color_discrete(name = 'Diagnosis (B: beningn, M: malignant)') +
 theme_bw() +
        labs(title = "Principal Component Analysis",
        subtitle = "1. Data distribution in the plan of PC3 and PC4\n2. Direc
tions of components in the same plane") +
        theme(legend.direction = 'horizontal', legend.position = 'bottom')

pc56<- ggbiplot(bc.pca, choices=5:6, obs.scale = 1, var.scale = 1, groups = b
```

```
c.diag,
        ellipse = TRUE, circle = TRUE, varname.size = 3, ellipse.prob = 0.68,
 circle.prob = 0.69) +
        scale_color_discrete(name = 'Diagnosis (B: beningn, M: malignant)') +
 theme_bw() +
        labs(title = "Principal Component Analysis",
        subtitle = "1. Data distribution in the plan of PC5 and PC6\n2. Direc
tions of components in the same plane") +
        theme(legend.direction = 'horizontal', legend.position = 'bottom')

grid.arrange(pc34, pc56, ncol=2)
```




- 주성분 PC3-PC6 이 25.5% 변화를 설명.
- PC3:PC6 주성분에서 유의미한 정렬을 관계되지 않음.

## t-SNE transform

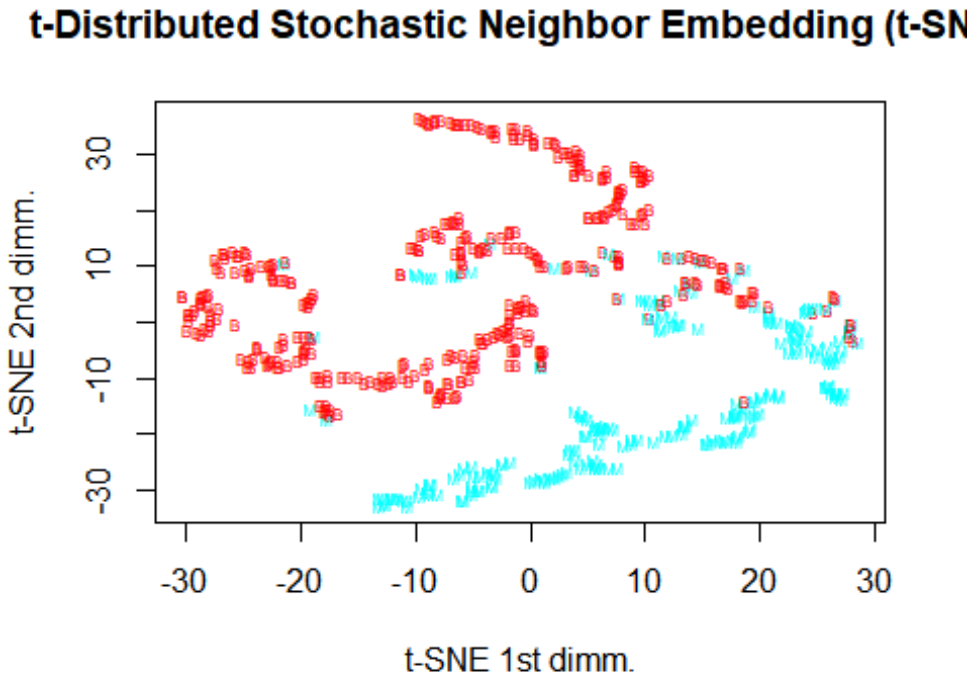
- PCA 는 선형 알고리즘이지만 t-SNE 는 비선형 차원 감소 알고리즘.
- 여러 기능이있는 데이터 요소의 유사성을 기반으로 관찰 된 클러스터를 식별하여 데이터의 패턴을 찾음.
- 클러스터링 알고리즘이 아니라 데이터 탐색 및 시각화 기술로 사용할 수있는 dimmensionality 감소 알고리즘.

```
library(Rtsne)
colors = rainbow(length(unique(bc.diag)))
names(colors) = unique(bc.diag)
```

```r
tsne <- Rtsne(bc.data, dims = 2, perplexity=10, verbose=TRUE, max_iter = 500)
```

```
## Read the 569 x 30 data matrix successfully!
## Using no_dims = 2, perplexity = 10.000000, and theta = 0.500000
## Computing input similarities...
## Normalizing input...
## Building tree...
##  - point 0 of 569
## Done in 0.05 seconds (sparsity = 0.064103)!
## Learning embedding...
## Iteration 50: error is 68.363539 (50 iterations in 0.22 seconds)
## Iteration 100: error is 58.906050 (50 iterations in 0.16 seconds)
## Iteration 150: error is 57.193580 (50 iterations in 0.21 seconds)
## Iteration 200: error is 56.521969 (50 iterations in 0.19 seconds)
## Iteration 250: error is 56.170191 (50 iterations in 0.19 seconds)
## Iteration 300: error is 0.741596 (50 iterations in 0.22 seconds)
## Iteration 350: error is 0.567157 (50 iterations in 0.22 seconds)
## Iteration 400: error is 0.534648 (50 iterations in 0.20 seconds)
## Iteration 450: error is 0.514638 (50 iterations in 0.17 seconds)
## Iteration 500: error is 0.505020 (50 iterations in 0.19 seconds)
## Fitting performed in 1.97 seconds.
```

```r
plot(tsne$Y, t='n', main="t-Distributed Stochastic Neighbor Embedding (t-SNE)
",
     xlab="t-SNE 1st dimm.", ylab="t-SNE 2nd dimm.")
text(tsne$Y, labels=bc.diag, cex=0.5, col=colors[bc.diag])
```




## Predictive models

- RandomForest (RF), Gradient Boosting Machine (GBM), Light Gradient Boosting Machine (lightGBM) and XGBoost : 4 가지 알고리즘 사용.

- lightGBM 은 설치를 못 함.

```
df <- raw.data[,2:32]
df$diagnosis = as.integer(factor(df$diagnosis))-1
nrows <- nrow(df)
set.seed(314)
indexT <- sample(1:nrow(df), 0.7 * nrows)

#separate train and validation set
trainset = df[indexT,]
testset =   df[-indexT,]

n <- names(trainset)
```

### Random Forest

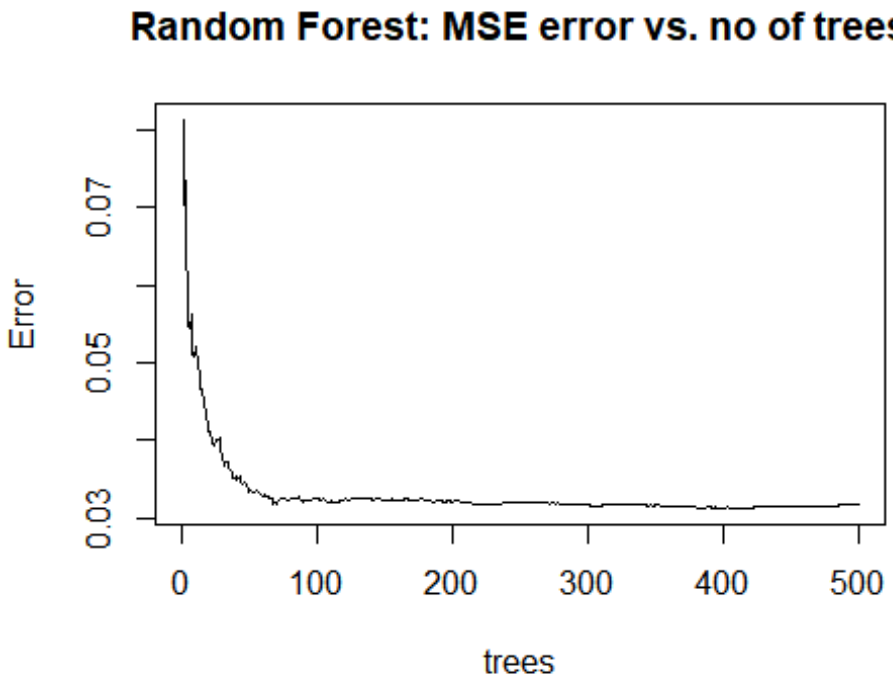
*Model using all features*

- tree 개수를 500 으로 설정.

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

- 에러와 tree 개수를 비교

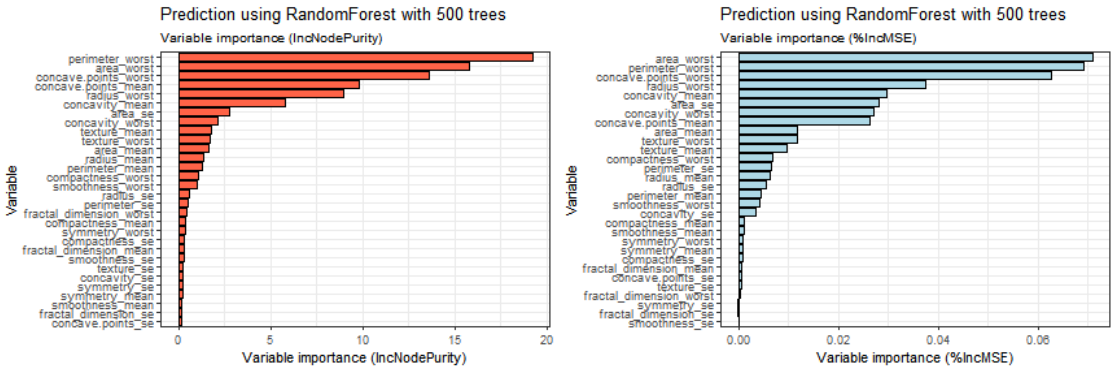
```
plot(trainset.rf, main="Random Forest: MSE error vs. no of trees")
```




- IncNodePurity 와 % IncMSE 의 두 가지 방법으로 변수의 중요성을 시각화
- IncNodePurity 는 모든 트리에 대해 평균화 된 변수에 대한 분할에서 지니 인덱스로 측정 한 노드 불순물의 총 감소량.
- IncMSE 는 변수 j 가 치환되는 결과 (임의로 섞인 값)로 예측의 mse 가 증가량.

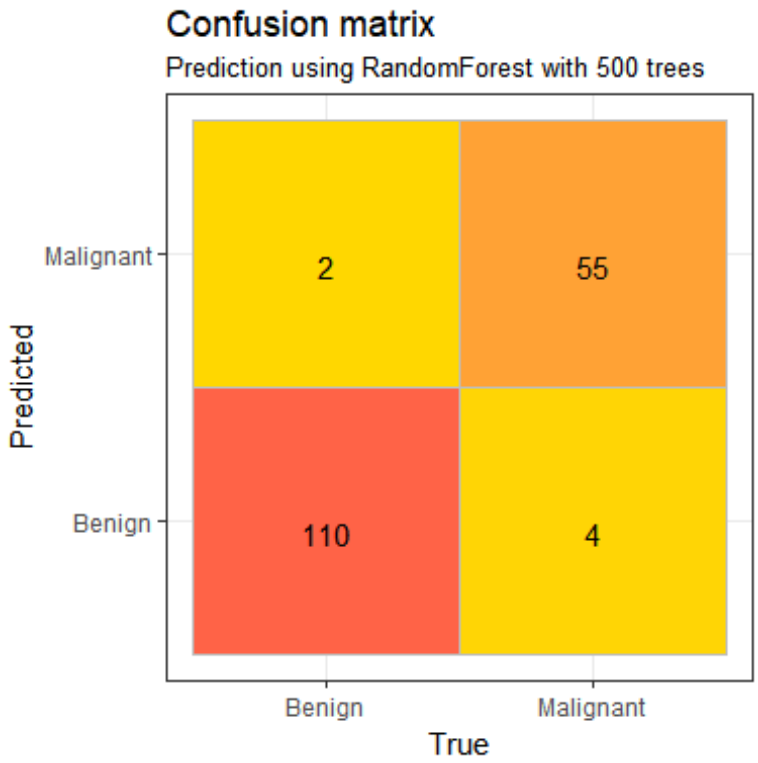
```
varimp <- data.frame(trainset.rf$importance)

  vi1 <- ggplot(varimp, aes(x=reorder(rownames(varimp),IncNodePurity), y=IncN
odePurity)) +
  geom_bar(stat="identity", fill="tomato", colour="black") +
  coord_flip() + theme_bw(base_size = 8) +
  labs(title="Prediction using RandomForest with 500 trees", subtitle="Variab
le importance (IncNodePurity)", x="Variable", y="Variable importance (IncNode
Purity)")

  vi2 <- ggplot(varimp, aes(x=reorder(rownames(varimp),X.IncMSE), y=X.IncMSE))
 +
```

```
  geom_bar(stat="identity", fill="lightblue", colour="black") +
  coord_flip() + theme_bw(base_size = 8) +
  labs(title="Prediction using RandomForest with 500 trees", subtitle="Variab
le importance (%IncMSE)", x="Variable", y="Variable importance (%IncMSE)")

grid.arrange(vi1, vi2, ncol=2)
```




- perimeter_worst, area_worst, concave.points_worst, radius_worst, concavity_mean, concavity_worst, area_se, concave.points_mean 가 중요한 변수
- 그들 대부분은 주요한 Principal Components 평면에서 상위 차원의 기능 목록에 있거나 주요한 Principal Component 인 PC1 과 정렬이 된 변수.

```
testset$predicted <- round(predict(trainset.rf ,testset),0)

plotConfusionMatrix <- function(testset, sSubtitle) {
    tst <- data.frame(testset$predicted, testset$diagnosis)
    opts <- c("Predicted", "True")
    names(tst) <- opts
    cf <- plyr::count(tst)
    cf[opts][cf[opts]==0] <- "Benign"
    cf[opts][cf[opts]==1] <- "Malignant"

    ggplot(data =  cf, mapping = aes(x = True, y = Predicted)) +
      labs(title = "Confusion matrix", subtitle = sSubtitle) +
      geom_tile(aes(fill = freq), colour = "grey") +
      geom_text(aes(label = sprintf("%1.0f", freq)), vjust = 1) +
      scale_fill_gradient(low = "gold", high = "tomato") +
      theme_bw() + theme(legend.position = "none")
}

plotConfusionMatrix(testset,"Prediction using RandomForest with 500 trees")
```



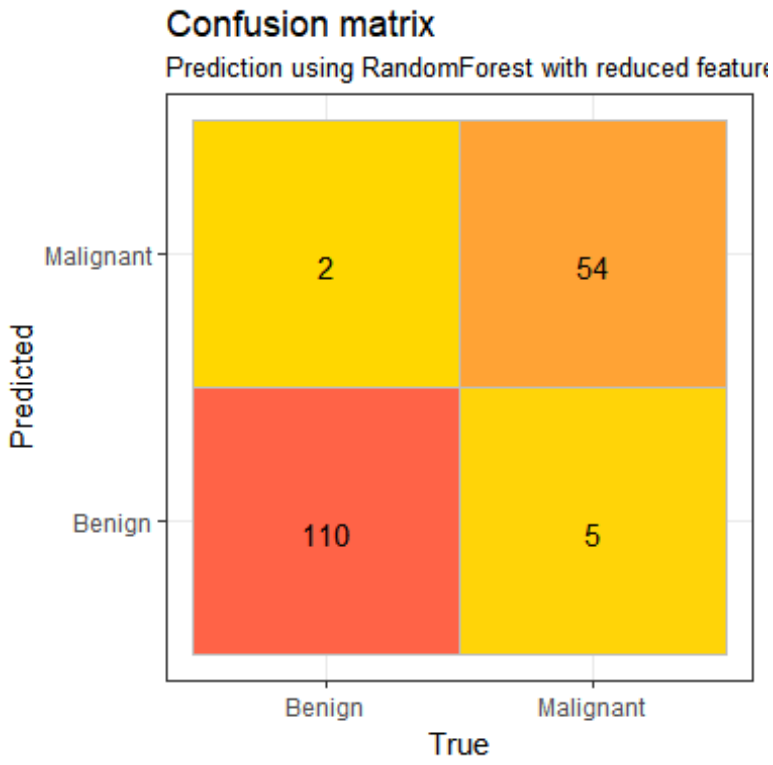

```
print(sprintf("Area under curve (AUC) : %.3f",auc(testset$diagnosis, testset
$predicted)))

## [1] "Area under curve (AUC) : 0.957"
```

*Model with reduced number of features*

- %IncMSE 값에 따라서 중요한 변수만 선택

```
features_list = c("perimeter_worst", "area_worst", "concave.points_worst", "r
adius_worst",
                  "concavity_mean", "concavity_worst","area_se", "concave.poi
nts_mean",
                  "texture_worst", "area_mean", "texture_mean", "area_mean",
                  "radius_mean", "radius_se", "perimeter_mean", "perimeter_se
",
                  "compactness_worst", "smoothness_worst", "concavity_se",
                  "fractal_dimension_worst", "symmetry_worst",  "diagnosis")

#define train and validation set
trainset_fl = trainset[,features_list]
testset_fl =   testset[,features_list]

#training
n <- names(trainset_fl)
rf.form <- as.formula(paste("diagnosis ~", paste(n[!n %in% "diagnosis"], coll
```

```
apse = " + ")))
trainset.rf <- randomForest(rf.form,trainset_fl,ntree=500,importance=T)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

#prediction
testset_fl$predicted <- round(predict(trainset.rf ,testset_fl),0)

plotConfusionMatrix(testset_fl,"Prediction using RandomForest with reduced fe
atures set")
```




```
print(sprintf("Area under curve (AUC) : %.3f",auc(testset_fl$diagnosis, tests
et_fl$predicted)))

## [1] "Area under curve (AUC) : 0.949"

print(sprintf("Area under curve (AUC) : %.3f",auc(testset_fl$diagnosis, tests
et_fl$predicted)))

## [1] "Area under curve (AUC) : 0.949"
```

- 변수 개수를 33 개에서 22 개로 줄임.
- 감소 된 특징 집합을 사용하면 진정한 양성 (TP) 수가 감소하고 거짓 음성 (FP) 수가 증가하는 반면 진정한 음성 및 위양성은 변하지 않습니다 .

- AUC 는 민감도의 감소와 선택성의 변화로 인해 0.949 으로 감소

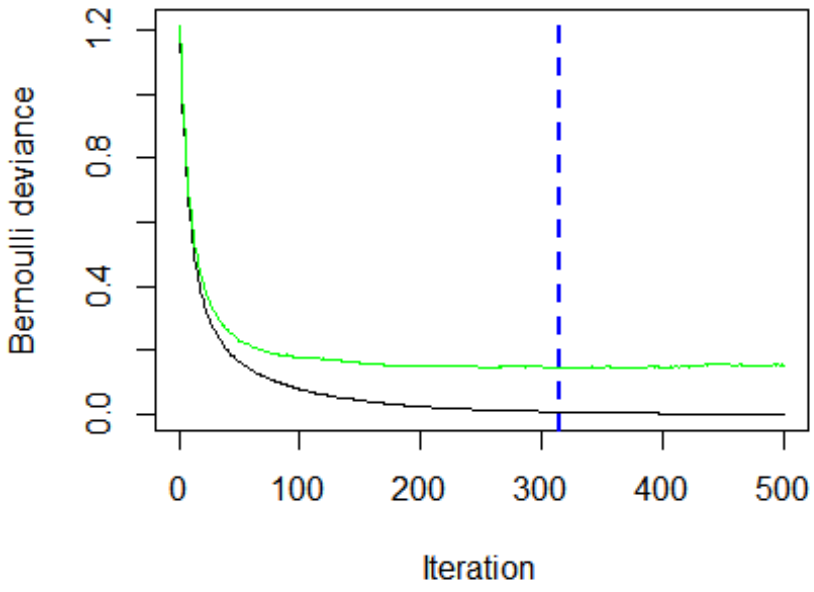
## Gradient Boosting Machine (GBM)

- 5 fold Cross validation 을 사용.

```
n<-names(trainset)
gbm.form <- as.formula(paste("diagnosis ~", paste(n[!n %in% "diagnosis"], col
lapse = " + ")))

gbmCV = gbm(formula = gbm.form,
                distribution = "bernoulli",
                data = trainset,
                n.trees = 500,
                shrinkage = .1,
                n.minobsinnode = 15,
                cv.folds = 5,
                n.cores = 1)
```

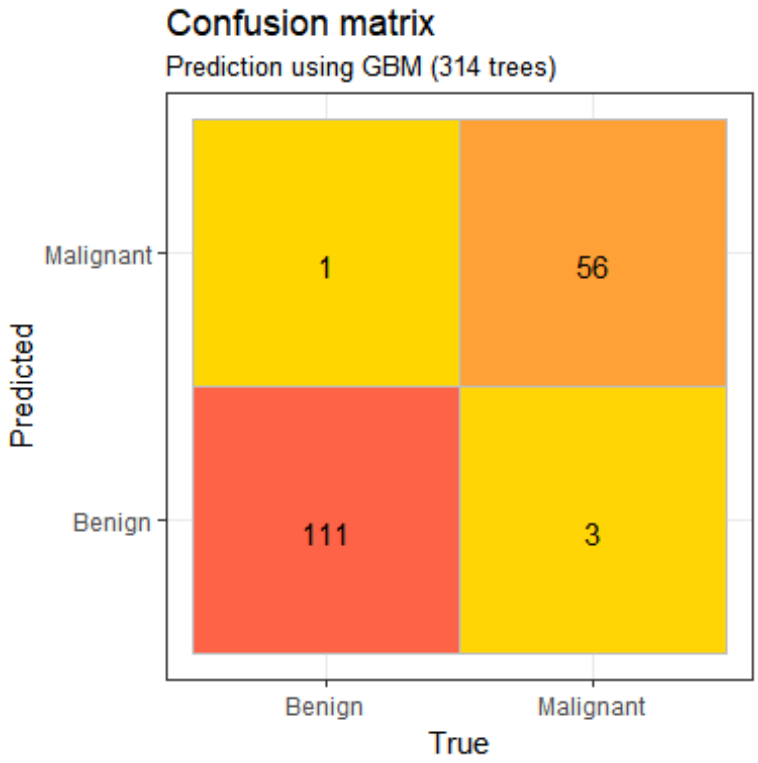
- 가지 개수를 최적화하기 위해서 gbm.perf()함수 사용.
- This function returns the optimal number of trees for prediction.

```
## Using cv method...
```




```
gbmTest = predict(object = gbmCV,
                              newdata = testset,
```

```
                              n.trees = optimalTreeNumberPredictionCV,
                              type = "response")
testset$predicted <- round(gbmTest,0)

plotConfusionMatrix(testset,sprintf("Prediction using GBM (%d trees)",optimal
TreeNumberPredictionCV))
```




```
print(sprintf("Area under curve (AUC) : %.3f",auc(testset$diagnosis, testset
$predicted)))

## [1] "Area under curve (AUC) : 0.970"
```

## Light Gradient Boosting Machines (lightGBM)

- 생략

## eXtreme Gradient Boost (XGBoost)

```
dMtrain <- xgb.DMatrix(as.matrix(trainset %>% select(-diagnosis)), label = tr
ainset$diagnosis)
dMtest <- xgb.DMatrix(as.matrix(testset %>% select(-diagnosis,-predicted)), l
abel = testset$diagnosis)
```

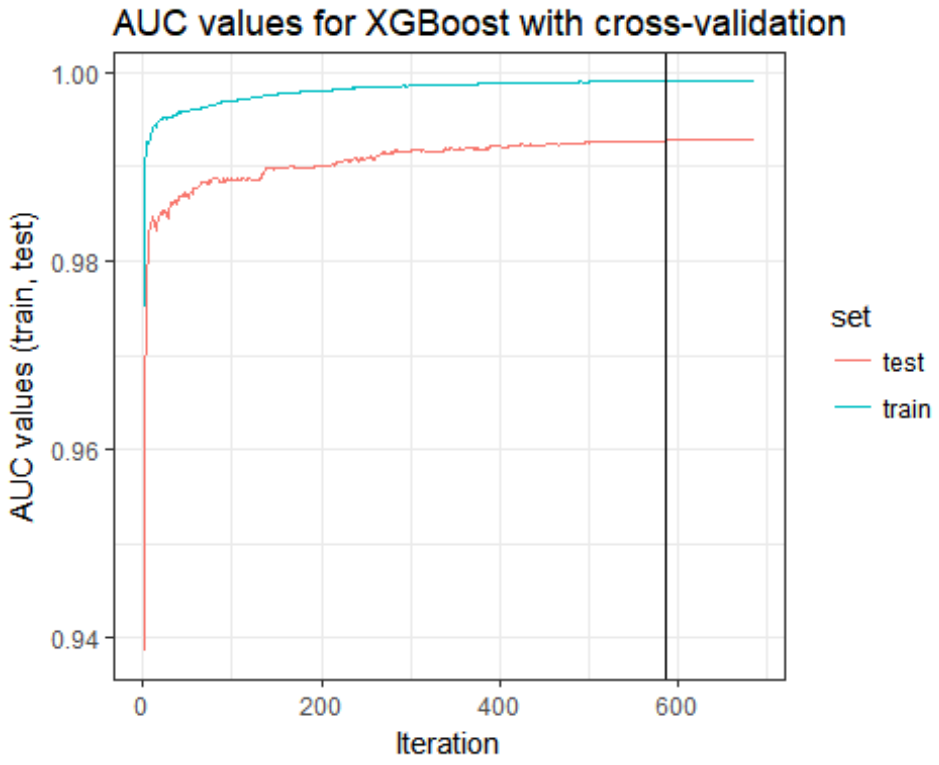
We set the XGBoost parameters for the model. We will use a binary logistic objective function. The evaluation metric will be AUC (Area under curve). We start with $\eta$ = 0.012, `subsample`=0.8, `max_depth`=8, `colsample_bytree`=0.9 and `min_child_weight`=5.

```
params <- list(
  "objective"           = "binary:logistic",
  "eval_metric"         = "auc",
  "eta"                 = 0.012,
  "subsample"           = 0.8,
  "max_depth"           = 8,
  "colsample_bytree"    =0.9,
  "min_child_weight"    = 5
)
```

- 5fold cross validation 사용.
- 5000 단계의 라운드수와 100 단계의 early stopping 기준을 갖음.
- 100 단계마다 부분 결과 출력.

```
nRounds <- 5000
earlyStoppingRound <- 100
printEveryN = 100

model_xgb.cv <- xgb.cv(params=params,
                       data = dMtrain,
                       maximize = TRUE,
                       nfold = 5,
                       nrounds = nRounds,
                       nthread = 1,
                       early_stopping_round=earlyStoppingRound,
                       print_every_n=printEveryN)

## [1]  train-auc:0.975069+0.001905 test-auc:0.938539+0.022705
## Multiple eval metrics are present. Will use test_auc for early stopping.
## Will train until test_auc hasn't improved in 100 rounds.
##
## [101]    train-auc:0.996977+0.000702 test-auc:0.988625+0.010420
## [201]    train-auc:0.998039+0.000626 test-auc:0.990007+0.010265
## [301]    train-auc:0.998606+0.000497 test-auc:0.991752+0.009836
## [401]    train-auc:0.998815+0.000395 test-auc:0.992137+0.008454
## [501]    train-auc:0.998982+0.000318 test-auc:0.992530+0.008792
## [601]    train-auc:0.999116+0.000219 test-auc:0.992801+0.008901
## Stopping. Best iteration:
## [586]    train-auc:0.999099+0.000237 test-auc:0.992801+0.008901

d <- model_xgb.cv$evaluation_log
n <- nrow(d)
v <- model_xgb.cv$best_iteration
df <- data.frame(x=rep(d$iter, 2), val=c(d$train_auc_mean, d$test_auc_mean),
                 set=rep(c("train", "test"), each=n))
ggplot(data = df, aes(x=x, y=val)) +
  geom_line(aes(colour=set)) +
  geom_vline(xintercept=v) +
```

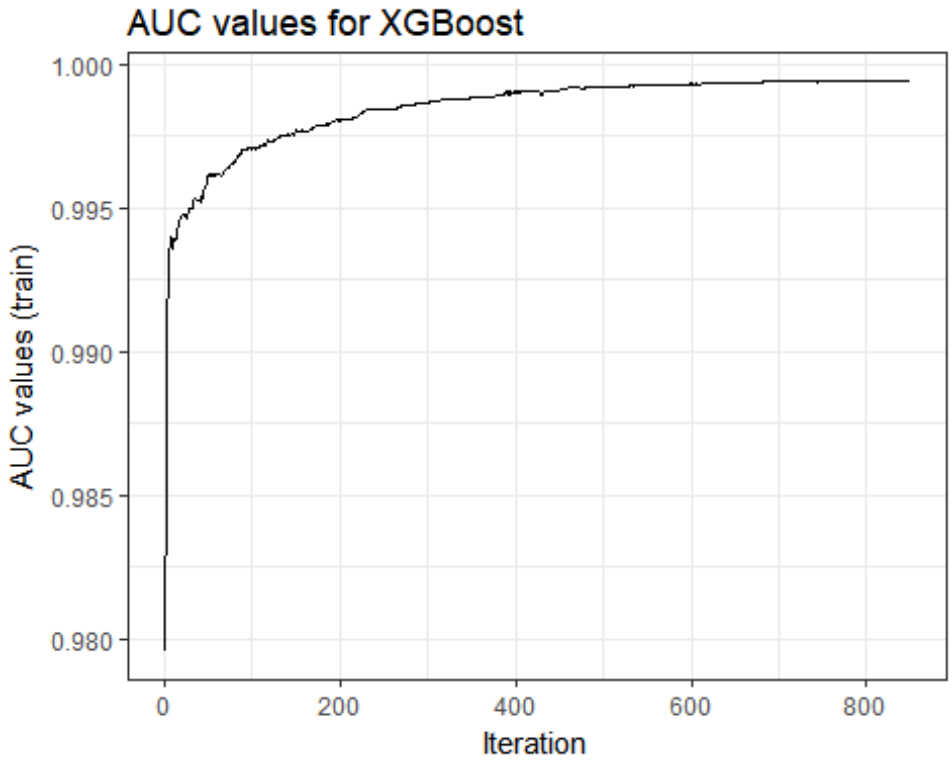
```
  theme_bw() +
  labs(title="AUC values for XGBoost with cross-validation", x="Iteration", y
="AUC values (train, test)")
```




- AUC : 0.99

```
model_xgb <- xgboost(params=params,
                     data = dMtrain,
                     maximize = TRUE,
                     nrounds = nRounds,
                     nthread = 1,
                     early_stopping_round=earlyStoppingRound,
                     print_every_n=printEveryN)

## [1]  train-auc:0.979565
## Will train until train_auc hasn't improved in 100 rounds.
##
## [101]    train-auc:0.997079
## [201]    train-auc:0.998053
## [301]    train-auc:0.998666
## [401]    train-auc:0.999013
## [501]    train-auc:0.999173
## [601]    train-auc:0.999333
## [701]    train-auc:0.999386
## [801]    train-auc:0.999440
## Stopping. Best iteration:
## [749]    train-auc:0.999440
```

```
d <- model_xgb$evaluation_log
n <- nrow(d)
df <- data.frame(x=rep(d$iter), val=d$train_auc)
ggplot(data = df, aes(x=x, y=val)) +
  geom_line() +
  theme_bw() +
  labs(title="AUC values for XGBoost", x="Iteration", y="AUC values (train)")
```

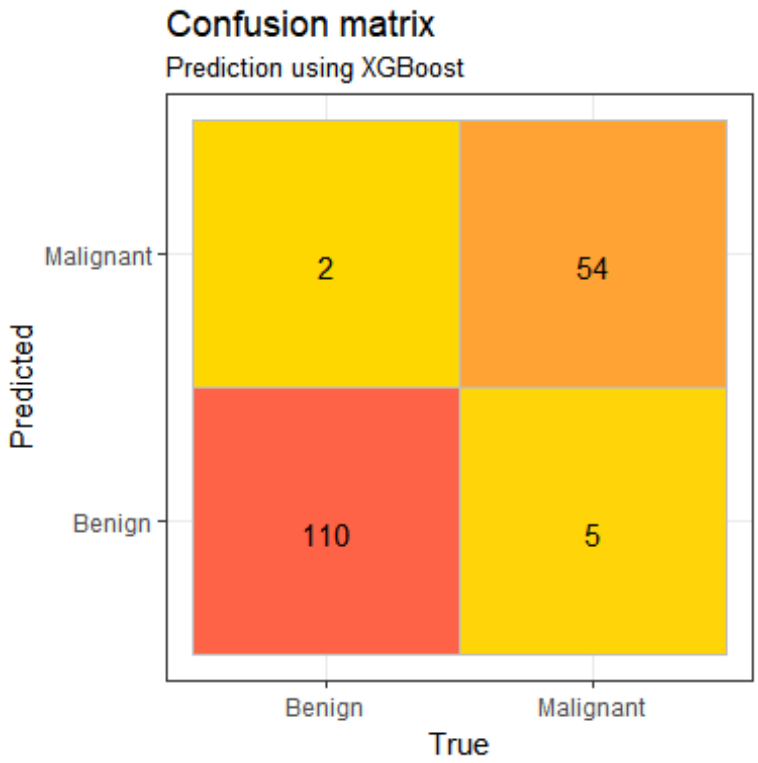



Let's use the model now to predict the test data:

```
testset$predicted <- round(predict(object = model_xgb ,newdata = dMtest),0)
```

Let's visualize the confusion matrix, to see how accurate are the results we obtained.

```
plotConfusionMatrix(testset,"Prediction using XGBoost")
```




Let's calculate as well the AUC for the prediction.

```
print(sprintf("Area under curve (AUC) : %.3f",auc(testset$diagnosis, testset
$predicted)))
```

```
## [1] "Area under curve (AUC) : 0.949"
```

## 결론

- feature analysis 으로 진단을 위한 좋은 예측 변수를 확인해봄.
- 주성분 분석을 통해서 `concave.ponts_worst`, `concavity_mean`, `concavity_worst`, `perimeter_worst`, `area_worst` 변수가 중요하다는 것을 파악.
- `Random Forest`, `Gradient Boosting Machine (GBM)`, `Light Gradient Boosting Machine (lightGBM)` and `XGBoost` 4 가지 모델을 사용
- `GBM`, `lightGBM` and `XGBoost` 모델을 cross validation 을 사용해서 최적 모델 결정.