A Comparative Analysis of Machine Learning Classification Methods Applied to Wisconsin Breast Cancer Database

Soodabeh Ramezani

Fall, 2019

```
#All packages
'Loading library corrplot in order to use corrplot function for
plotting dependency between input features using heat map'
library(corrplot)
```

```
## corrplot 0.84 loaded
```

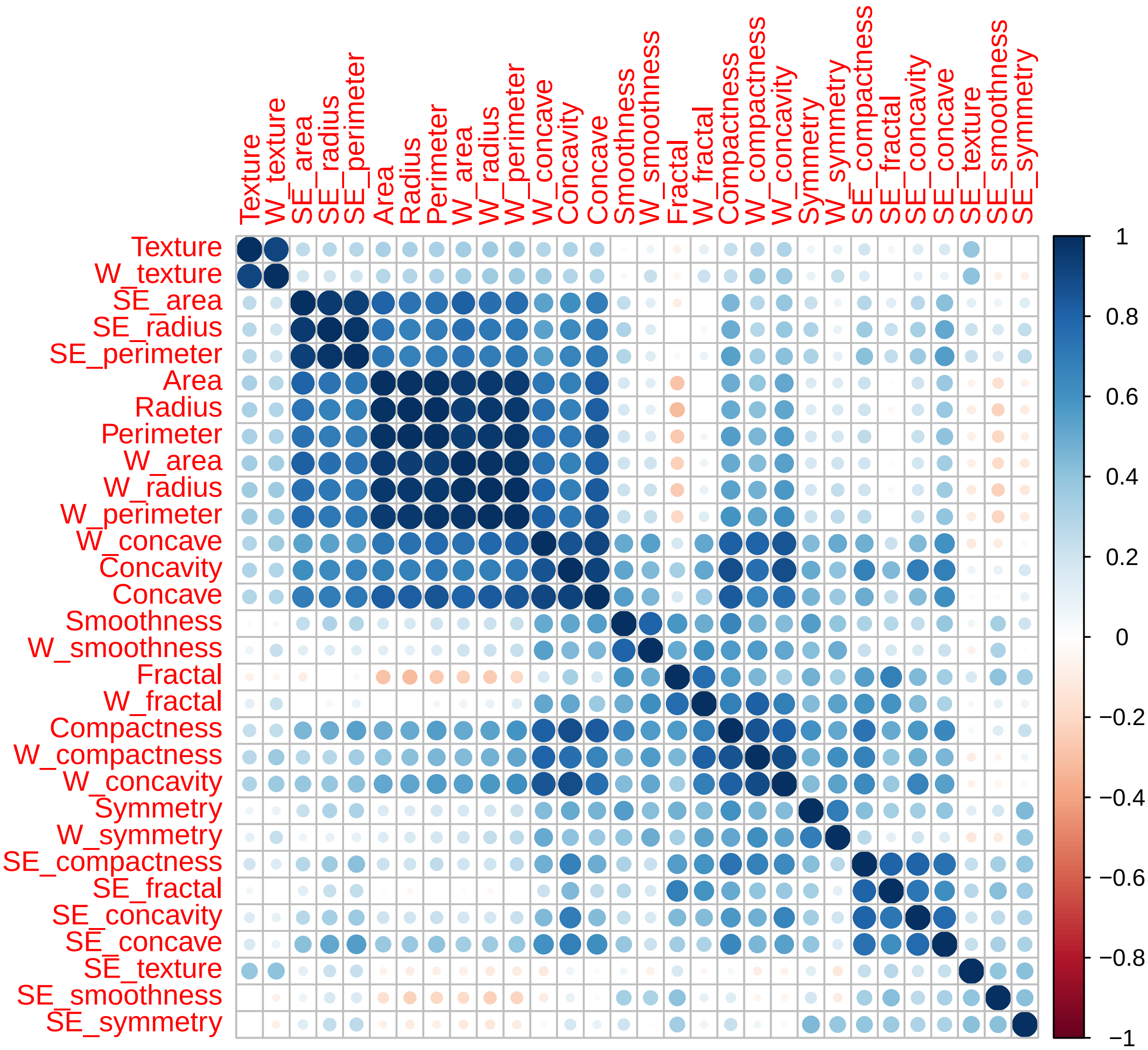
```
'Loading library ggplot2 in order to use ggplot function for
advance plotting with variety of visualization options'
library(ggplot2)

'Loading library tibble in order to use function as_tibble to
convert data to tibble format which makes it easier to handle
tabular data including data merging and printing'
library(tibble)

'Loading library GGally in order to use ggpairs function to make a
generalized pair plot for  numerical and categorial data'
library(GGally)
```

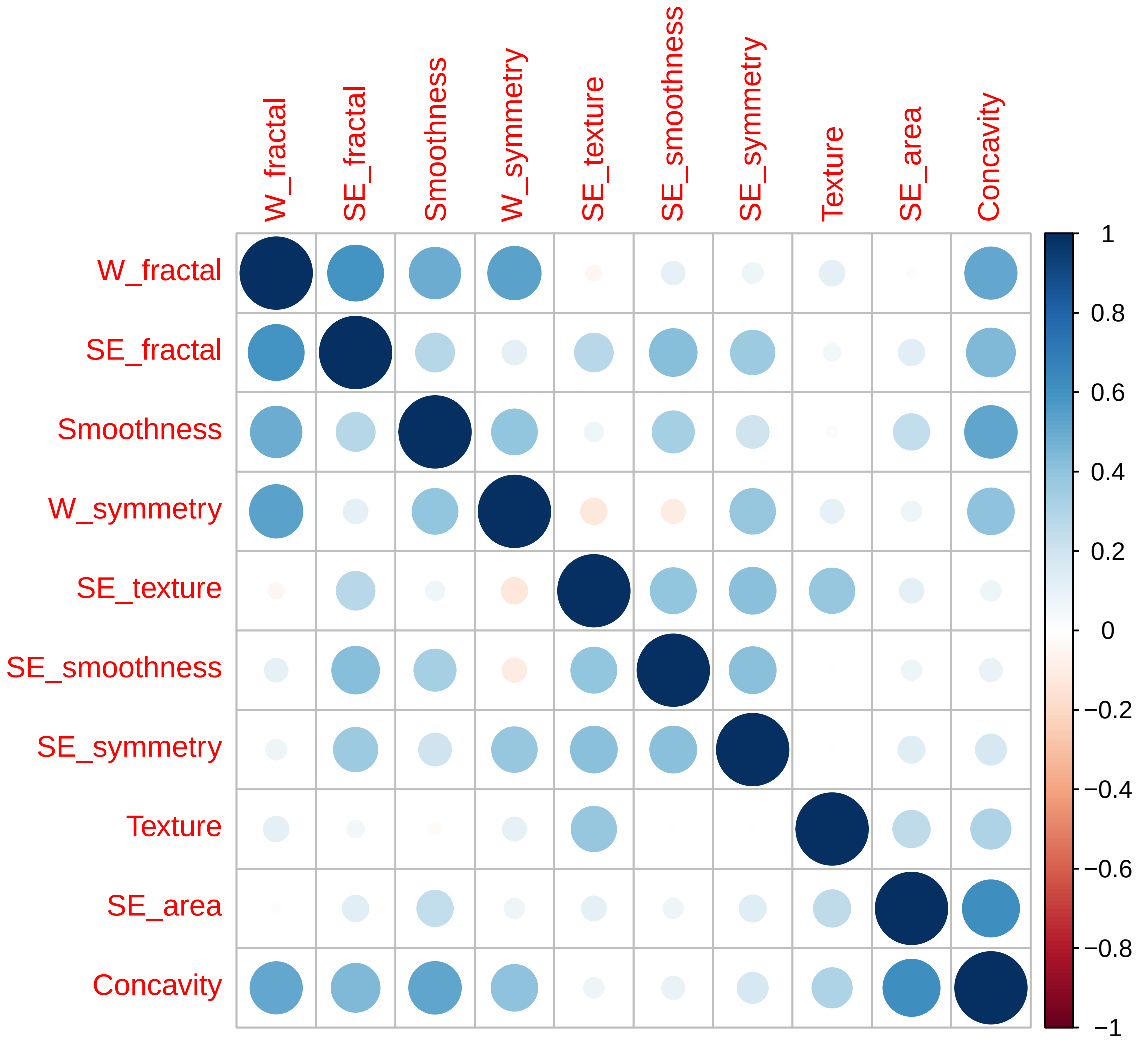
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

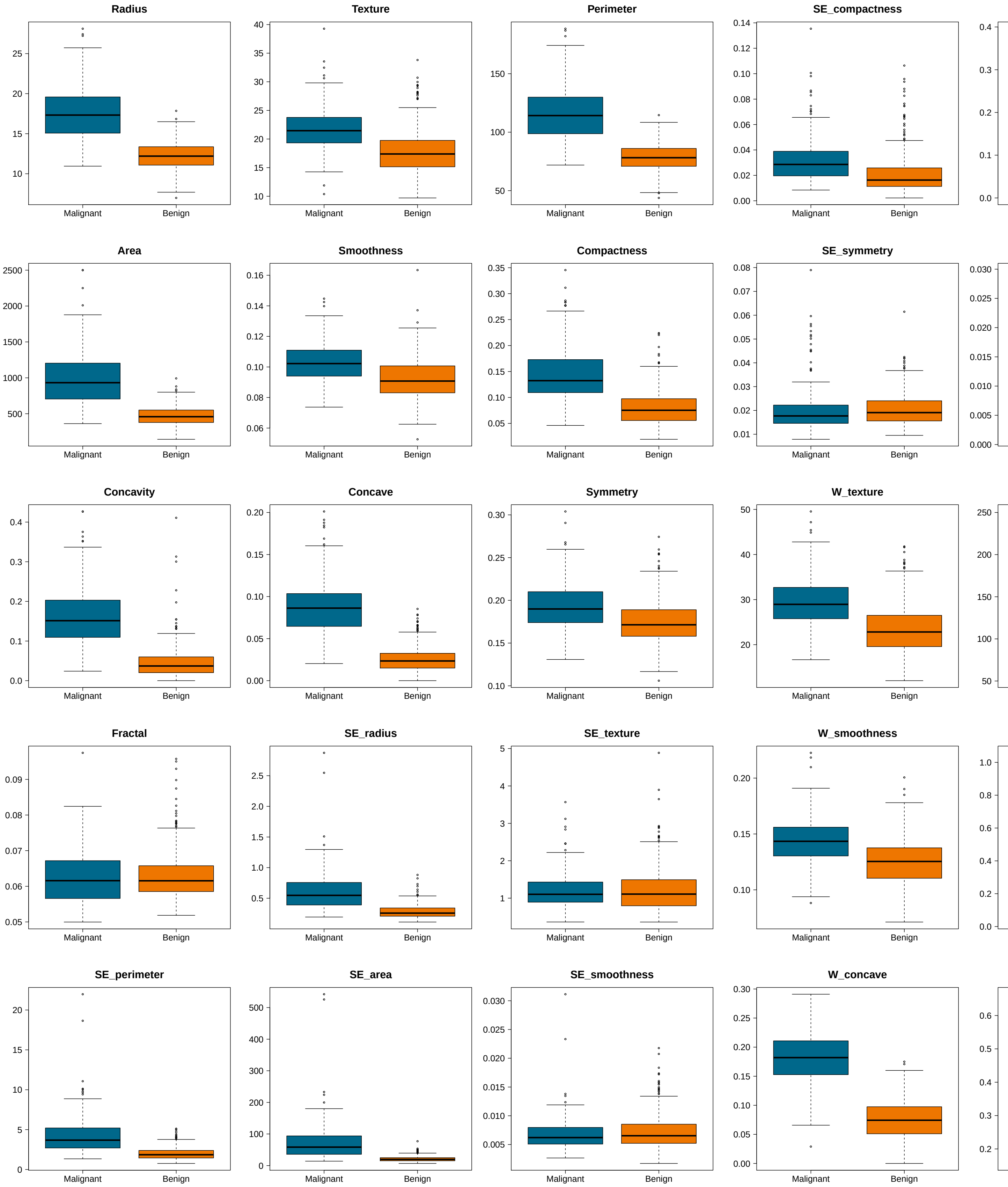
```
'Loading library gmodels in order to use CrossTable to
show/compare the performance of classification'
library(gmodels)

#Loading library mass in order to use lda function
library(MASS)

#Loading library class in order to use knn function
library(class)

'Loading caret library to use createFolds in order to
partition the data into k folds for cross-validation'
library(caret)
```

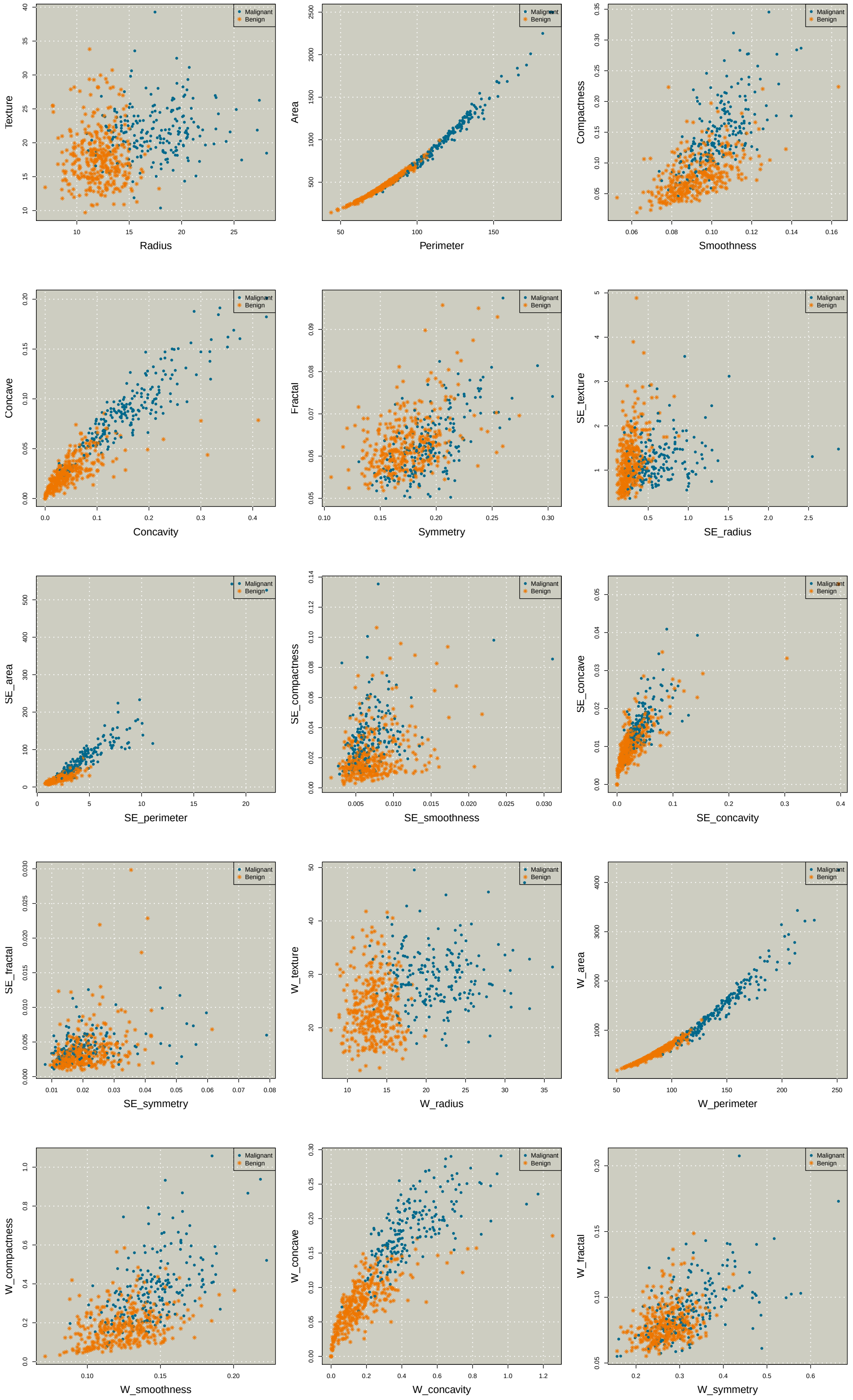
```
## Loading required package: lattice
```

```
#Loading library tree for using tree function
library(tree)
```


Texture
W_texture
SE_area
SE_radius
SE_perimeter
Area
Radius
Perimeter
W_area
W_radius
W_perimeter
W_concave
Concavity
Concave
Smoothness
W_smoothness
Fractal
W_fractal
Compactness
W_compactness
W_concavity
Symmetry
W_symmetry
SE_compactness
SE_fractal
SE_concavity
SE_concave
SE_texture
SE_smoothness
SE_symmetry
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


W_fractal
SE_fractal
Smoothness
W_symmetry
SE_texture
SE_smoothness
SE_symmetry
Texture
SE_area
Concavity
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


Radius
25
20
15
10
Malignant
Benign
Texture
40
35
30
25
20
15
10
Malignant
Benign
Perimeter
150
100
50
Malignant
Benign
SE_compactness
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Malignant
Benign
0.4
0.3
0.2
0.1
0.0
Area
2500
2000
1500
1000
500
Malignant
Benign
Smoothness
0.16
0.14
0.12
0.10
0.08
0.06
Malignant
Benign
Compactness
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Malignant
Benign
SE_symmetry
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
Malignant
Benign
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Concavity
0.4
0.3
0.2
0.1
0.0
Malignant
Benign
Concave
0.20
0.15
0.10
0.05
0.00
Malignant
Benign
Symmetry
0.30
0.25
0.20
0.15
0.10
Malignant
Benign
W_texture
50
40
30
20
Malignant
Benign
250
200
150
100
50
Fractal
0.09
0.08
0.07
0.06
0.05
Malignant
Benign
SE_radius
2.5
2.0
1.5
1.0
0.5
Malignant
Benign
SE_texture
5
4
3
2
1
Malignant
Benign
W_smoothness
0.20
0.15
0.10
Malignant
Benign
1.0
0.8
0.6
0.4
0.2
0.0
SE_perimeter
20
15
10
5
0
Malignant
Benign
SE_area
500
400
300
200
100
0
Malignant
Benign
SE_smoothness
0.030
0.025
0.020
0.015
0.010
0.005
Malignant
Benign
W_concave
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Malignant
Benign
0.6
0.5
0.4
0.3
0.2


Malignant
Benign
Texture
Radius
10 15 20 25
10 15 20 25 30 35 40
Area
Perimeter
50 100 150
500 1000 1500 2000 2500
Compactness
Smoothness
0.06 0.08 0.10 0.12 0.14 0.16
0.05 0.10 0.15 0.20 0.25 0.30 0.35
Concave
Concavity
0.0 0.1 0.2 0.3 0.4
0.00 0.05 0.10 0.15 0.20
Fractal
Symmetry
0.10 0.15 0.20 0.25 0.30
0.05 0.06 0.07 0.08 0.09
SE_texture
SE_radius
0.5 1.0 1.5 2.0 2.5
1 2 3 4 5
SE_area
SE_perimeter
0 5 10 15 20
0 100 200 300 400 500
SE_compactness
SE_smoothness
0.005 0.010 0.015 0.020 0.025 0.030
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14
SE_concave
SE_concavity
0.0 0.1 0.2 0.3 0.4
0.00 0.01 0.02 0.03 0.04 0.05
SE_fractal
SE_symmetry
0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
0.000 0.005 0.010 0.015 0.020 0.025 0.030
W_texture
W_radius
10 15 20 25 30 35
20 30 40 50
W_area
W_perimeter
50 100 150 200 250
1000 2000 3000 4000
W_compactness
W_smoothness
0.10 0.15 0.20
0.0 0.2 0.4 0.6 0.8 1.0
W_concave
W_concavity
0.0 0.2 0.4 0.6 0.8 1.0 1.2
0.00 0.05 0.10 0.15 0.20 0.25 0.30
W_fractal
W_symmetry
0.2 0.3 0.4 0.5 0.6
0.05 0.10 0.15 0.20

```
summary(Cancer$Perimeter)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   43.79   75.17   86.24   91.97  104.10  188.50
```

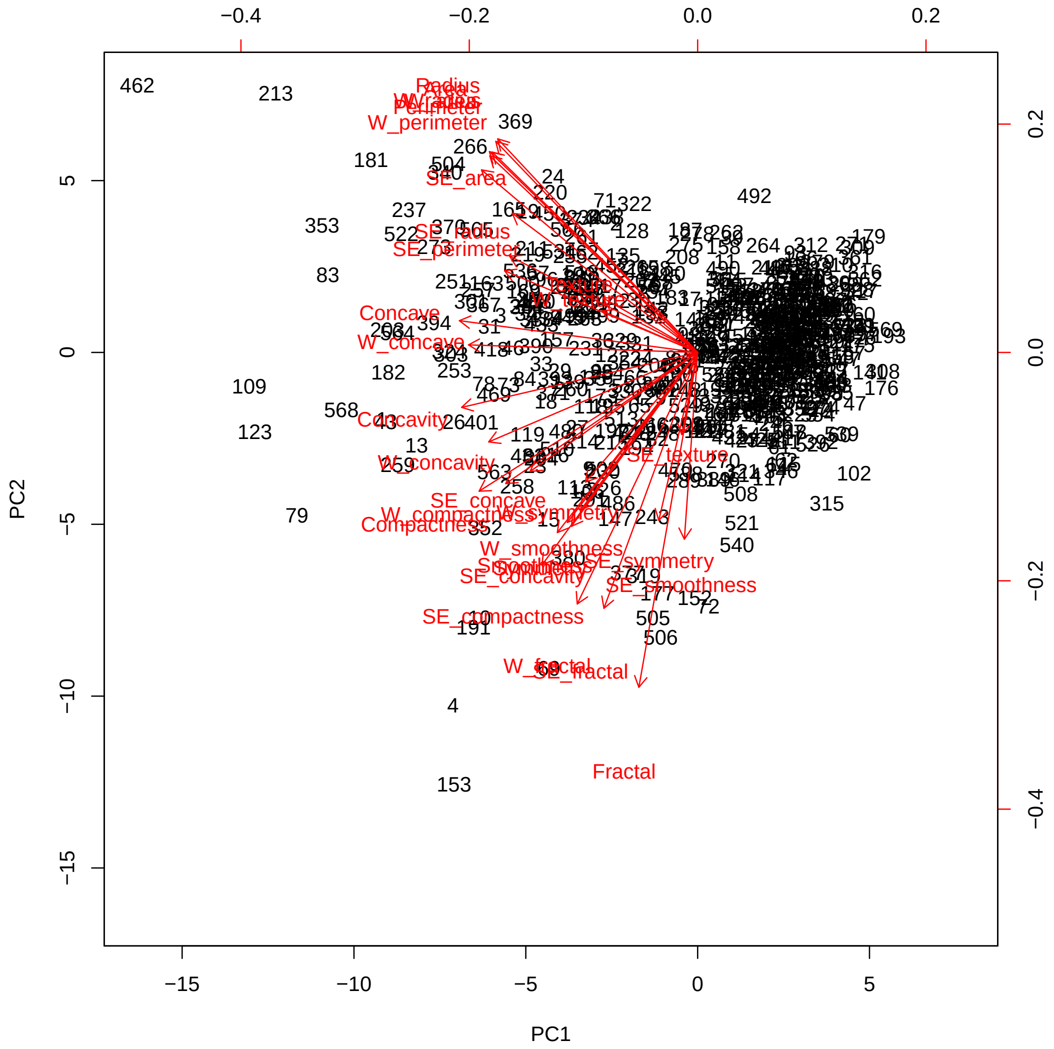
```
summary(Cancer$W_perimeter)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50.41   84.11   97.66  107.26  125.40  251.20
```

```
summary(Cancer$W_radius)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.93   13.01   14.97   16.27   18.79   36.04
```

```
#Use prcomp to compute principal components
pca_Cancer=prcomp(Cancer[, 3:32], scale=TRUE)
#------------------------------------------------------------
par(mfrow=c(1,1))
#use biplot to plot first two principal components
biplot(pca_Cancer, scale=0)
```



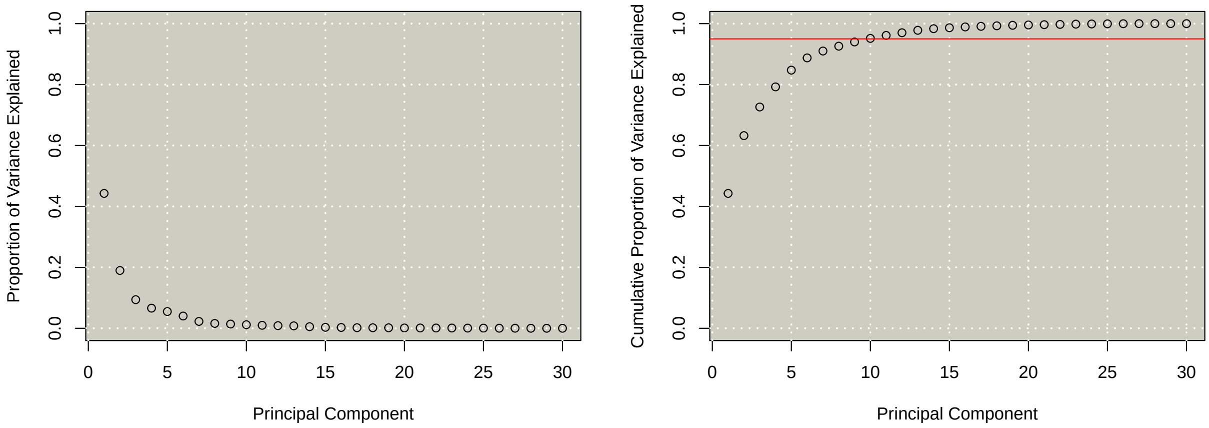
```
#Print the standard deviation of each principle component (PC)
pca_Cancer$sdev
```

```
##  [1] 3.64439401 2.38565601 1.67867477 1.40735229 1.28402903 1.09879780
##  [7] 0.82171778 0.69037464 0.64567392 0.59219377 0.54213992 0.51103950
## [13] 0.49128148 0.39624453 0.30681422 0.28260007 0.24371918 0.22938785
## [19] 0.22243559 0.17652026 0.17312681 0.16564843 0.15601550 0.13436892
## [25] 0.12442376 0.09043030 0.08306903 0.03986650 0.02736427 0.01153451
```

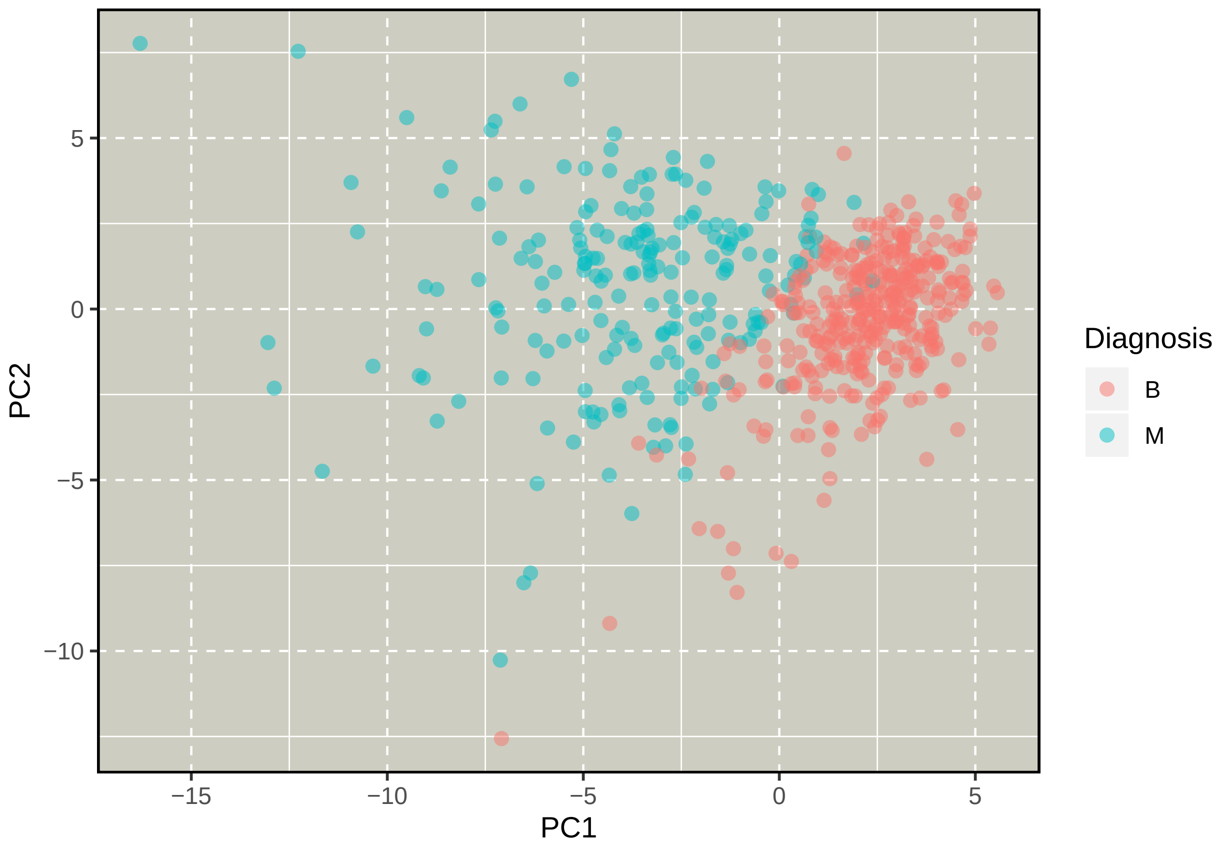
```
#Calculate the variance of each PC
pca_Cancer_var=pca_Cancer$sdev^2

#Compute proportion of variance explained by each PC
pve=pca_Cancer_var/sum(pca_Cancer_var)

#Print the variance explained by each PC
pve
```

```
##  [1] 4.427203e-01 1.897118e-01 9.393163e-02 6.602135e-02 5.495768e-02
##  [6] 4.024522e-02 2.250734e-02 1.588724e-02 1.389649e-02 1.168978e-02
## [11] 9.797190e-03 8.705379e-03 8.045250e-03 5.233657e-03 3.137832e-03
## [16] 2.662093e-03 1.979968e-03 1.753959e-03 1.649253e-03 1.038647e-03
## [21] 9.990965e-04 9.146468e-04 8.113613e-04 6.018336e-04 5.160424e-04
## [26] 2.725880e-04 2.300155e-04 5.297793e-05 2.496010e-05 4.434827e-06
```

```
#Plot the PVE (Proportion of Variance Explained) by each PC--------------------------------------------
par(mfrow=c(1,2))
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1),type='b')
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory3")
grid(col = "white", lty = "dotted",lwd =1.8)
points(pve)
#First compute the cumulative sum of elements for PVE vector
# Then plot the cumulative PVE for each PC -----------------------------
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained",
     ylim=c(0,1),type='b')
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory3")
grid(col = "white", lty = "dotted",lwd =1.8)
points(cumsum(pve))
abline(h=0.95, col="red")
```



```
#Convort the principal component score vectors to a dataframe
pca_x <- as.data.frame(pca_Cancer$x)


library(ggplot2)
#Show pairwise scatter plot of PC1 vs. PC2, color coded by image type (benign or malignant)
par(mfrow=c(1,1))
Diagnosis=Cancer$Diagnosis
ggplot(pca_x, aes(x=PC1, y=PC2,col=Diagnosis))+ geom_point(alpha=0.5, size=2) +
  theme(panel.background = element_rect(fill="ivory3"),
        panel.grid.major = element_line(size = 0.4, linetype = 2,colour = "white"),
  panel.border = element_rect(colour = "black", fill=NA, size=0.9))
```



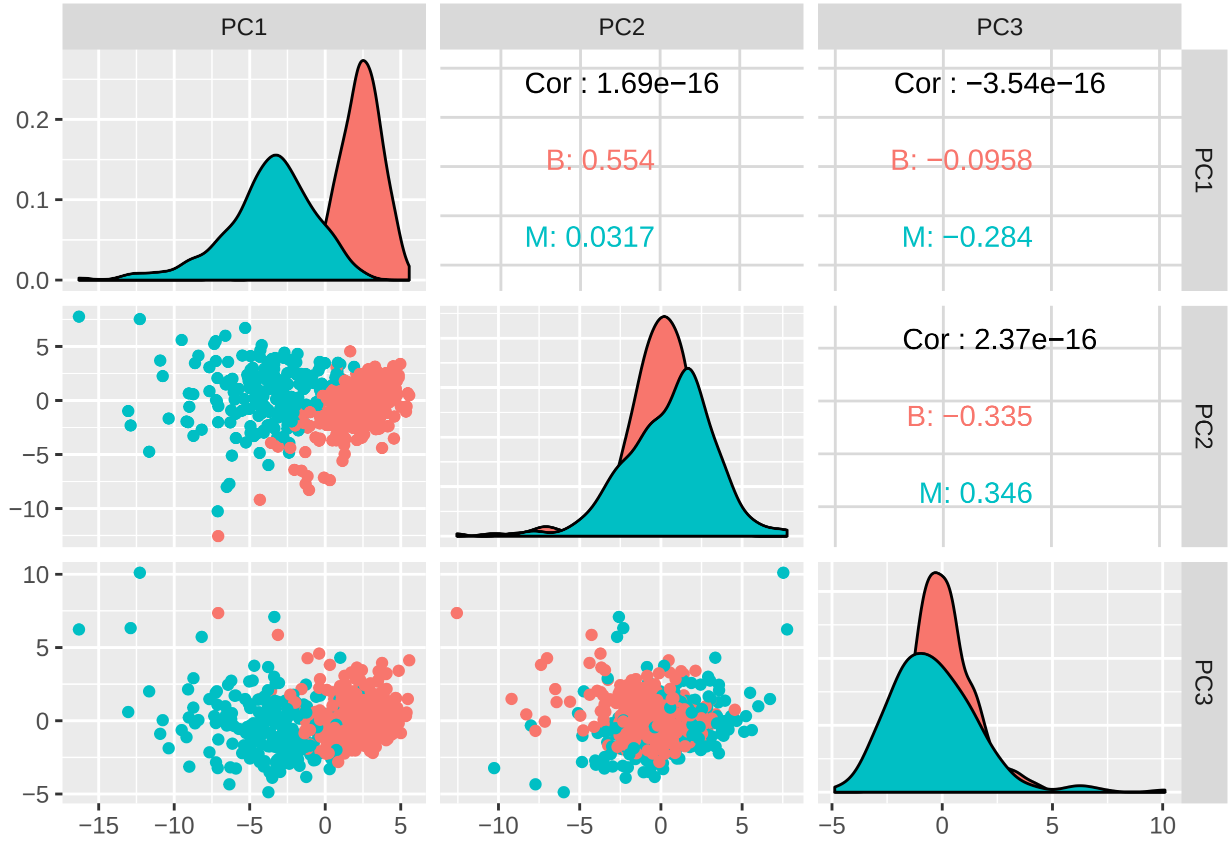
```
#-----------------------------------------------------------------------------------------------


library(tibble)
#Combin Diagnosis column with principal component score vectors
df_pcs <- cbind(as_tibble(Cancer$Diagnosis), as_tibble(pca_Cancer$x))
```

```
## Warning: Calling `as_tibble()` on a vector is discouraged, because the behavior is likely to change i
## This warning is displayed once per session.
```

```
library(GGally)

'Show pairwise scatter plots of the first three principle components after
apply pca function, color coded by image type (benign or malignant).
The distribution of rotated data is shown on the diagonal.'
```

```
## [1] "Show pairwise scatter plots of the first three principle components after\napply pca function, c
```

```
GGally::ggpairs(df_pcs, columns = 2:4, ggplot2::aes(color = value))
```



Split the data into training and test sets

```
#Split the sample into train and test sets in order to estimate the test error

#set a unique seed for random generator to ensure reproducibility of results
set.seed(123)
n<-dim(Cancer[1])

#Allocate 75% of data to training set with the remaining being the test set
samp<-sample(1:n,size=0.75*n, rep=FALSE)
```

```
## Warning in 1:n: numerical expression has 2 elements: only the first used
```

```
#Create train and test datasets from the original dataset------------------------
Cancer_train<-Cancer[samp,]
Cancer_test<-Cancer[-samp,]

#Compute principal components of the training set--------------------------------
pc <- prcomp(Cancer_train[, 3:32])

'Combine the Diagnosis column (Benign or Malignant) with the principal component score vectors
(pc in the previous commandline) to create training set'
```

```
## [1] "Combine the Diagnosis column (Benign or Malignant) with the principal component score vectors \r
```

```
Cancer_train_pc <- data.frame(Diagnosis = Cancer_train[,"Diagnosis"], pc$x)

#Rotate the test sample based on principle component loading vectors
Cancer_test_pc <- data.frame(predict(pc, newdata =Cancer_test[, 3:32]))
```

Logistic Regression

```
#Logistic Regression

#Use glm() function to fit logistic regression
glm_Cancer <- glm(Diagnosis ~Texture+SE_area+Concavity+Smoothness+W_fractal+W_symmetry+SE_fractal+
                    SE_texture+SE_smoothness+SE_symmetry, data =Cancer_train , family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_Cancer)
```

```
##
## Call:
## glm(formula = Diagnosis ~ Texture + SE_area + Concavity + Smoothness +
##     W_fractal + W_symmetry + SE_fractal + SE_texture + SE_smoothness +
##     SE_symmetry, family = binomial, data = Cancer_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.58877  -0.04959  -0.00628   0.00010   2.68816
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.579e+01  7.615e+00  -4.699 2.61e-06 ***
## Texture        5.235e-01  1.344e-01   3.895 9.81e-05 ***
## SE_area        2.836e-01  5.799e-02   4.890 1.01e-06 ***
## Concavity      4.928e+01  1.080e+01   4.562 5.07e-06 ***
## Smoothness     1.734e+01  3.851e+01   0.450  0.65244
## W_fractal      1.037e+02  5.418e+01   1.914  0.05566 .
## W_symmetry     3.852e+01  1.232e+01   3.128  0.00176 **
## SE_fractal    -1.473e+03  4.519e+02  -3.261  0.00111 **
## SE_texture    -1.116e+00  8.248e-01  -1.352  0.17622
## SE_smoothness  3.222e+02  1.528e+02   2.109  0.03495 *
## SE_symmetry   -2.829e+02  9.109e+01  -3.106  0.00190 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 551.501  on 425  degrees of freedom
## Residual deviance:  61.575  on 415  degrees of freedom
## AIC: 83.575
##
## Number of Fisher Scoring iterations: 10
```

```
#Use predict function to predict the probability of diagnosis being Malignant
glm_prob=predict(glm_Cancer,Cancer_test,type="response")

#Use contrasts function to see the dummy variable that R has created with response (M,B)
contrasts(Cancer_train$Diagnosis)
```

```
##   M
## B 0
## M 1
```

```
#Convert the predicted probabilities to class labels
glm_pred <- rep("B", nrow(Cancer_test))
glm_pred[glm_prob > 0.5] <- "M"

#Compute the misclassification rate
mm=mean(glm_pred==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.979021
```

```
1-mm
```

```
## [1] 0.02097902
```

```
library(gmodels)
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(glm_pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        80 |         3 |        83 |
##                   |     0.559 |     0.021 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         0 |        60 |        60 |
##                   |     0.000 |     0.420 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
```

```
##
```

Use the first two principle components of the data as inputs for logistic regression

```r
#Use the first two principle components of the data as inputs for logistic regression

#Use glm function to fit logistic regression
glm_Cancer <- glm(Cancer_train_pc$Diagnosis ~ PC1+PC2, data = Cancer_train_pc, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(glm_Cancer)
```

```
##
## Call:
## glm(formula = Cancer_train_pc$Diagnosis ~ PC1 + PC2, family = binomial,
##     data = Cancer_train_pc)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.96425  -0.27853  -0.11471   0.00433   2.96855
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.575027   0.291275   1.974   0.0484 *
## PC1         -0.011990   0.001430  -8.386  < 2e-16 ***
## PC2          0.030533   0.005604   5.448 5.09e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 551.50  on 425  degrees of freedom
## Residual deviance: 154.02  on 423  degrees of freedom
## AIC: 160.02
##
## Number of Fisher Scoring iterations: 8
```

```r
#Use predict function to predict the probability of diagnosis being Malignant
glm_prob <- predict(glm_Cancer, newdata = Cancer_test_pc, type = "response")


#Use contrasts function to see the dummy variable that R has created with response (M,B)
contrasts(Cancer_train_pc$Diagnosis)
```

```
##   M
## B 0
## M 1
```

```r
#Convert the predicted probabilities to class labels
glm_pred <- rep("B", nrow(Cancer_test_pc))
glm_pred[glm_prob > 0.5] <- "M"
```

```
#Compute the misclassification rate
mm=mean(glm_pred==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.9440559
```

```
1-mm
```

```
## [1] 0.05594406
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(glm_pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  143 
## 
## 
##                   | actual defaut 
## predicted default |         B |         M | Row Total | 
## ------------------|-----------|-----------|-----------|
##                 B |        78 |         6 |        84 | 
##                   |     0.545 |     0.042 |           | 
## ------------------|-----------|-----------|-----------|
##                 M |         2 |        57 |        59 | 
##                   |     0.014 |     0.399 |           | 
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 | 
## ------------------|-----------|-----------|-----------|
## 
## 
```

Linear Discriminant Analysis (LDA)

```
#Linear Discriminant Analysis (LDA)

library(MASS)
#Use lda() function to perform LDA on the data set
lda_Cancer <- lda(Diagnosis ~ Texture+SE_area+Concavity+Smoothness+W_fractal+W_symmetry+SE_fractal+
                    SE_texture+SE_smoothness+SE_symmetry,data =Cancer_train ,
                 family = binomial)
```

```
summary(lda_Cancer)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   20     -none- numeric
## scaling 10     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     4     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
#Use predict function to predict
lda_pred=predict(lda_Cancer,Cancer_test)
#Extract the LDA's prediction
lda_class=lda_pred$class

#Compute the misclassification rate
mm=mean(lda_class==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.9020979
```

```
1-mm
```

```
## [1] 0.0979021
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(lda_class,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        80 |        14 |        94 |
##                   |     0.559 |     0.098 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         0 |        49 |        49 |
##                   |     0.000 |     0.343 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

Use the first two principle components of the data as inputs for Linear Discriminant Analysis

```
#Use the first two principle components of the data as inputs for Linear Discriminant Analysis

#Use lda() function to perform LDA on the data set
lda_Cancer <- lda(Cancer_train_pc$Diagnosis ~ PC1+PC2, data =Cancer_train_pc, family = binomial)

summary(lda_Cancer)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
## means   4      -none- numeric
## scaling 2      -none- numeric
## lev     2      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    4      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

```
#Use predict function to predict
lda_pred=predict(lda_Cancer,Cancer_test_pc)
names(lda_pred)
```

```
## [1] "class"     "posterior" "x"
```

```
#Extract the LDA's prediction
lda_class=lda_pred$class

#Compute the misclassification rate
mm=mean(lda_class==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.8531469
```

```
1-mm
```

```
## [1] 0.1468531
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(lda_class,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
```

```
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        79 |        20 |        99 |
##                   |     0.552 |     0.140 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         1 |        43 |        44 |
##                   |     0.007 |     0.301 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

Quadratic Discriminant Analusis

```
#Quadratic Discriminant Analusis

#Use qda() function to fit Quadratic Discriminant Analusis
qda_Cancer <- qda(Diagnosis ~ Texture+SE_area+Concavity+Smoothness+W_fractal+W_symmetry+SE_fractal+
                    SE_texture+SE_smoothness+SE_symmetry,data =Cancer_train ,
                 family = binomial)

summary(qda_Cancer)
```

```
##         Length Class  Mode
## prior     2    -none- numeric
## counts    2    -none- numeric
## means    20    -none- numeric
## scaling 200    -none- numeric
## ldet      2    -none- numeric
## lev       2    -none- character
## N         1    -none- numeric
## call      4    -none- call
## terms     3    terms  call
## xlevels   0    -none- list
```

```
#Use predict function to predict
qda_pred=predict(qda_Cancer,Cancer_test)
#Extract the QDA's prediction
qda_class=qda_pred$class
```

```
#Compute the misclassification rate
mm=mean(qda_class==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.951049
```

```
1-mm
```

```
## [1] 0.04895105
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(qda_class,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  143 
## 
## 
##                    | actual defaut 
## predicted default |         B |         M | Row Total | 
## ------------------|-----------|-----------|-----------|
##                  B |        79 |         6 |        85 | 
##                    |     0.552 |     0.042 |           | 
## ------------------|-----------|-----------|-----------|
##                  M |         1 |        57 |        58 | 
##                    |     0.007 |     0.399 |           | 
## ------------------|-----------|-----------|-----------|
##       Column Total |        80 |        63 |       143 | 
## ------------------|-----------|-----------|-----------|
## 
## 
```

Use the first two principle components of the data as inputs for Quadratic Discriminant Analysis

```
#Use the first two principle components of the data as inputs for Quadratic Discriminant Analysis
#Use qda() function to fit Quadratic Discriminant Analusis
qda_Cancer <- qda( Cancer_train_pc$Diagnosis~ PC1+PC2, data =Cancer_train_pc ,
                 family = binomial)
summary(qda_Cancer)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
## means   4      -none- numeric
## scaling 8      -none- numeric
## ldet    2      -none- numeric
## lev     2      -none- character
## N       1      -none- numeric
## call    4      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

```
#Use predict function to predict
qda_pred=predict(qda_Cancer,Cancer_test_pc)
#Extract the QDA's prediction
qda_class=qda_pred$class

#Compute the misclassification rate
mm=mean(qda_class==Cancer_test$Diagnosis)
mm
```

```
## [1] 0.9440559
```

```
1-mm
```

```
## [1] 0.05594406
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(qda_class,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        78 |         6 |        84 |
##                   |     0.545 |     0.042 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         2 |        57 |        59 |
##                   |     0.014 |     0.399 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

K-Nearest Neighbors

```r
#K-Nearest Neighbors
#set a unique seed for random generator to ensure reproducibility of results
set.seed(123)

#Standardize test and training sets in order to use knn
standardized.train=scale(Cancer_train[,c("Texture","SE_area","Concavity","Smoothness","W_fractal",
                  "W_symmetry","SE_fractal","SE_texture","SE_smoothness","SE_symmetry")],scale =T)
standardized.test=scale(Cancer_test[,c("Texture","SE_area","Concavity","Smoothness","W_fractal",
                  "W_symmetry","SE_fractal","SE_texture","SE_smoothness","SE_symmetry")],scale =T)

#Compute the square root of the size of data as the first try in K-Nearest Neighbors  which is common
n=round(sqrt(dim(Cancer_train)[1]))


library(class)
#Using knn() function to perform KNN
knn.pred=knn(standardized.train, standardized.test, Cancer_train$Diagnosis, k=n)


#Compute the misclassification rate
mm=mean(knn.pred!=Cancer_test$Diagnosis)
1-mm
```

```
## [1] 0.8811189
```

```r
mm
```

```
## [1] 0.1188811
```

```r
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```r
CrossTable(knn.pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                  | actual defaut
## predicted default |         B |         M | Row Total |
## -------------------|-----------|-----------|-----------|
##                 B |        80 |        17 |        97 |
##                   |     0.559 |     0.119 |           |
## -------------------|-----------|-----------|-----------|
##                 M |         0 |        46 |        46 |
##                   |     0.000 |     0.322 |           |
## -------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## -------------------|-----------|-----------|-----------|
##
##
```

```
#--------------------------------------------------------------------------
```

Use the first two principle components of the data as inputs for K-Nearest Neighbors

```
set.seed(123)
#Use the first two principle components of the data as inputs for K-Nearest Neighbors
#Compute the square root of the size of data as the first try in K-Nearest Neighbors  which is common
n=round(sqrt(dim(Cancer_train_pc)[1]))
n
```

```
## [1] 21
```

```
#Using knn() function to perform KNN
knn.pred=knn(Cancer_train_pc[,2:3], Cancer_test_pc[,1:2], Cancer_train_pc$Diagnosis, k=n)

#Compute the misclassification rate
mm=mean(knn.pred!=Cancer_test$Diagnosis)
mm
```

```
## [1] 0.05594406
```

```
1-mm
```

```
## [1] 0.9440559
```

```
'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(knn.pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
```

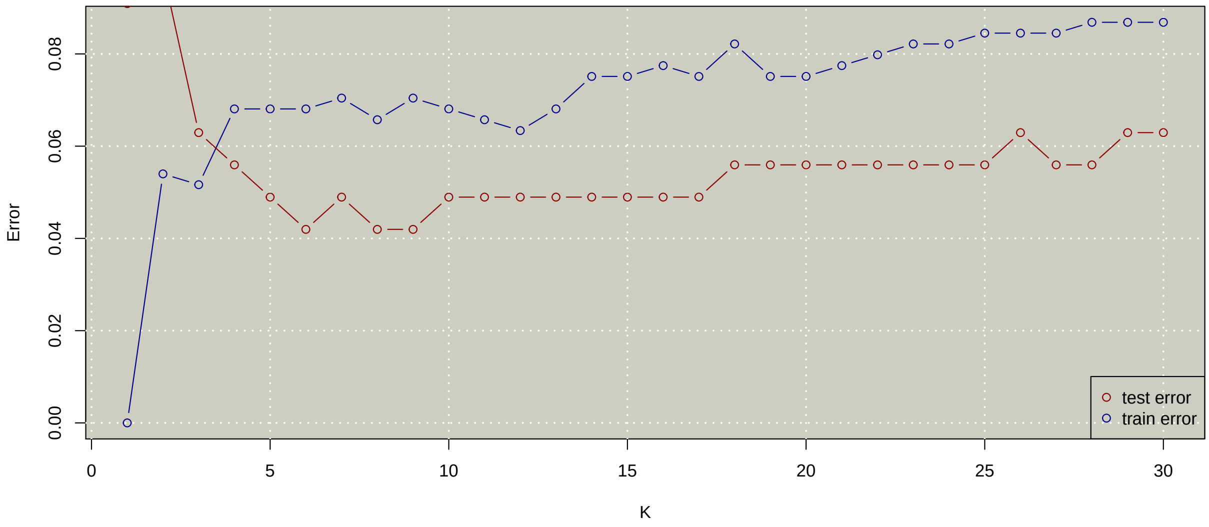
```
## Total Observations in Table:  143 
## 
##  
##                   | actual defaut 
## predicted default |         B |         M | Row Total | 
## -------------------|-----------|-----------|-----------|
##                 B |        78 |         6 |        84 | 
##                   |     0.545 |     0.042 |           | 
## -------------------|-----------|-----------|-----------|
##                 M |         2 |        57 |        59 | 
##                   |     0.014 |     0.399 |           | 
## -------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 | 
## -------------------|-----------|-----------|-----------|
## 
## 
```

```
#Try different k-------------------------------------------------------------------------------------
set.seed(123)
#Create two vectors for training error rate and test error rate
train.error=rep(0,30)
test.error=rep(0,30)
#Create a loop to quantify test and train errors for different k
for (k in 1:30){
  #Perform knn on train set and predict train accuracy
cancer_knn_tr=knn(Cancer_train_pc[,2:3], Cancer_train_pc[,2:3], Cancer_train_pc$Diagnosis, k=k)
  #Compute the missclasification
train.error[k] <- sum(cancer_knn_tr != Cancer_train$Diagnosis)/length(Cancer_train$Diagnosis)

  #Perform knn on train set and predict test accuracy
cancer_knn_ts=knn(Cancer_train_pc[,2:3], Cancer_test_pc[,1:2], Cancer_train_pc$Diagnosis, k=k)
test.error[k] <- sum(cancer_knn_ts != Cancer_test$Diagnosis)/length(Cancer_test$Diagnosis)
}
#Plot the computed errors
par(mfrow=c(2,1))
plot(1:30, train.error, xlab = "K", ylab="Error",col="darkorange", type="b")
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory3")
grid(col = "white", lty = "dotted",lwd =1.8)
points(1:30, train.error, col="darkblue", type="b")
points(1:30, test.error, col="darkred", type="b")
legend("bottomright", c("test error", "train error"), col = c("darkred", "darkblue"), pch = c(1,1) )

#K=3 -------------------------------------------------------------------------------------
knn.pred=knn(Cancer_train_pc[,2:3], Cancer_test_pc[,1:2], Cancer_train_pc$Diagnosis, k=3)

CrossTable(knn.pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        77 |         6 |        83 |
##                   |     0.538 |     0.042 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         3 |        57 |        60 |
##                   |     0.021 |     0.399 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

```r
#K=6 -------------------------------------------------------------------------------------------
knn.pred=knn(Cancer_train_pc[,2:3], Cancer_test_pc[,1:2], Cancer_train_pc$Diagnosis, k=6)

CrossTable(knn.pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        78 |         4 |        82 |
##                   |     0.545 |     0.028 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         2 |        59 |        61 |
##                   |     0.014 |     0.413 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```



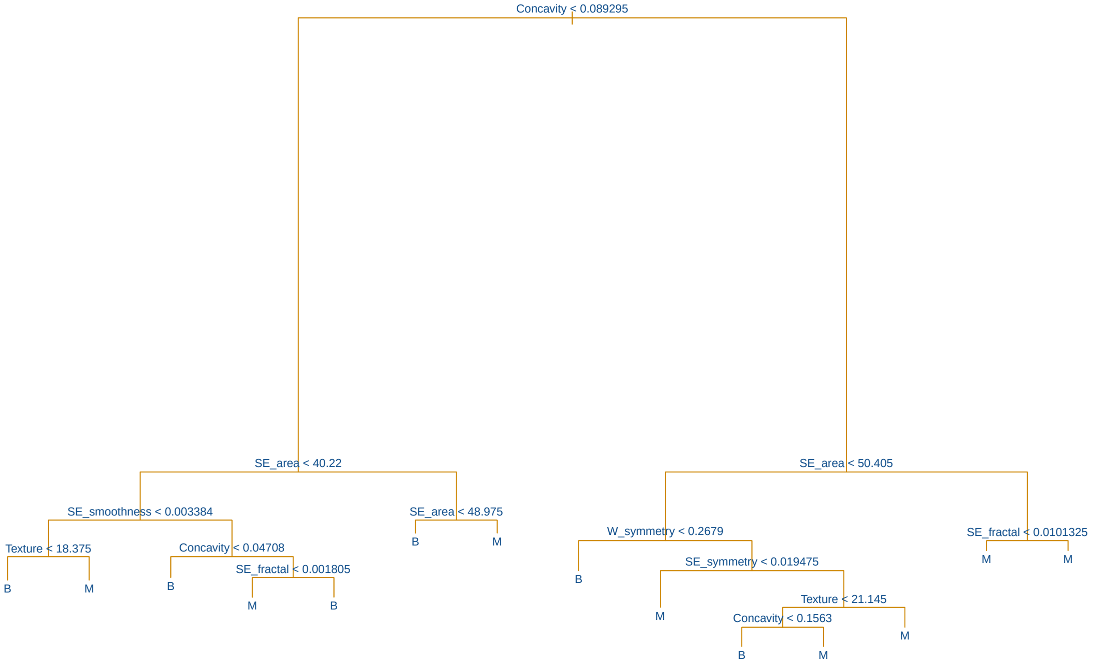
Tree-Based Method (TBM)

```
##Tree-Based Method (TBM)

library(tree)
# Use tree function to fit a tree-based classifier
TBM_Cancer=tree(Diagnosis ~ Texture+SE_area+Concavity+Smoothness+W_fractal+W_symmetry+SE_fractal+
                    SE_texture+SE_smoothness+SE_symmetry , data =Cancer_train)

summary(TBM_Cancer)
```

```
##
```

```
## Classification tree:
## tree(formula = Diagnosis ~ Texture + SE_area + Concavity + Smoothness +
##     W_fractal + W_symmetry + SE_fractal + SE_texture + SE_smoothness +
##     SE_symmetry, data = Cancer_train)
## Variables actually used in tree construction:
## [1] "Concavity"     "SE_area"       "SE_smoothness" "Texture"
## [5] "SE_fractal"    "W_symmetry"    "SE_symmetry"
## Number of terminal nodes:  14
## Residual mean deviance:  0.191 = 78.71 / 412
## Misclassification error rate: 0.03521 = 15 / 426
```

```
#Use plot function to display the tree structure
plot(TBM_Cancer, col = "orange3", lwd = 1)
#Use the text function to display the node labels
text(TBM_Cancer,offset = 1,cex = 0.8, col = "dodgerblue4", pretty = 0)
```



```
#Use predict function to predict the test set from the developed model
TBM_pred=predict(TBM_Cancer,Cancer_test,type="class")


'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(TBM_pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
```

```
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        73 |         9 |        82 |
##                   |     0.510 |     0.063 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         7 |        54 |        61 |
##                   |     0.049 |     0.378 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

Pruning the tree to improve the resault

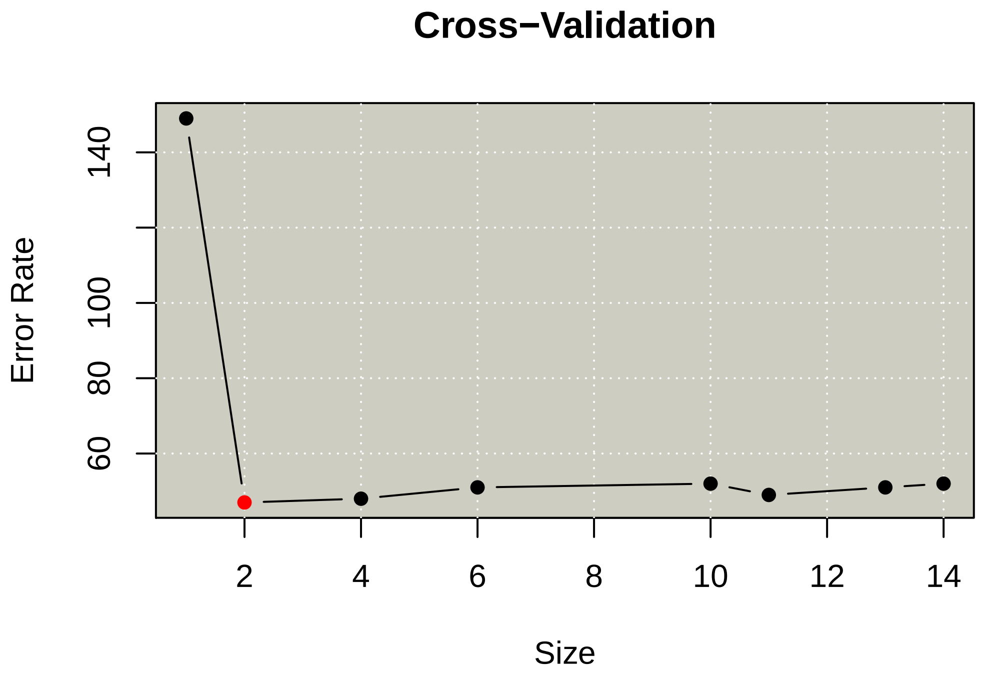
```
#Pruning the tree to improve the result
set.seed(123)

#Use cv.tree function to perform cross-validation
cv_Cancer=cv.tree(TBM_Cancer,FUN=prune.misclass)
names(cv_Cancer)
```

```
## [1] "size"   "dev"    "k"      "method"
```

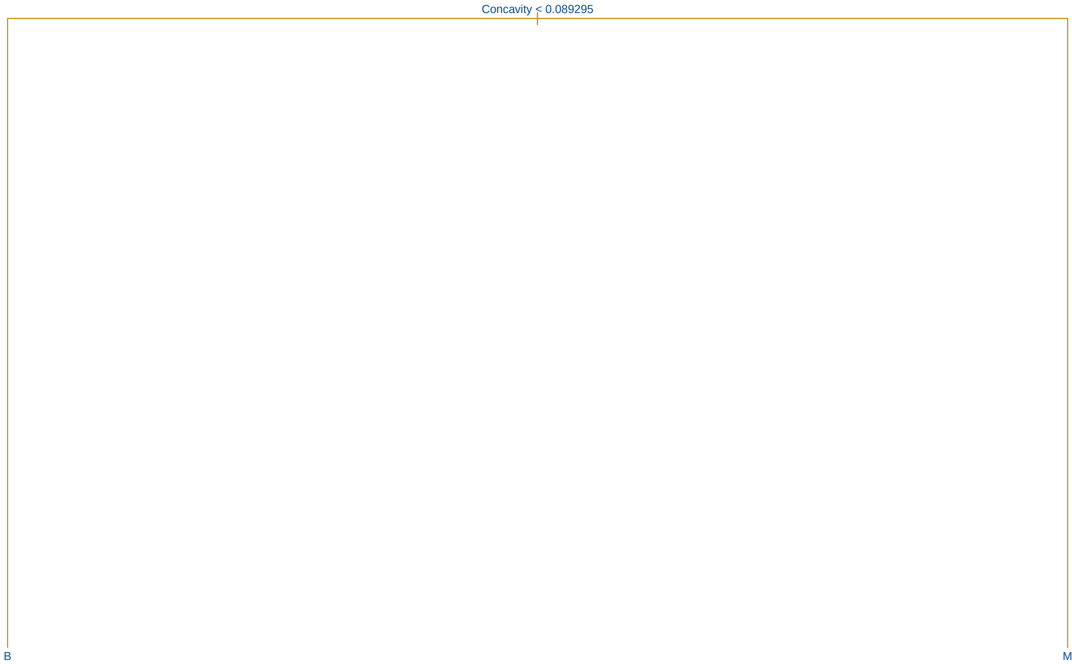
```
cv_Cancer
```

```
## $size
## [1] 14 13 11 10  6  4  2  1
##
## $dev
## [1]  52  51  49  52  51  48  47 149
##
## $k
## [1]  -Inf   0.0   0.5   2.0   2.5   4.5   5.0 102.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
#Plot the error rate as a function of size
m=min(cv_Cancer$dev)
Col=ifelse(cv_Cancer$dev==m ,2,1)
plot(cv_Cancer$size,cv_Cancer$dev,type="b", xlab ="Size", ylab="Error Rate",
     main="Cross-Validation")
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory3")
grid(col = "white", lty = "dotted",lwd =1)
points(cv_Cancer$size,cv_Cancer$dev,type="b", col=Col, pch=16)
```



```
#best=2-------------------------------------------------------
#Use prune.misclass function in order to prune the tree to obtain the two-node tree
Prune_Cancer=prune.misclass(TBM_Cancer,best=2)

#Use plot function to display the tree structure
#Use the text function to display the node labels
plot(Prune_Cancer,col = "orange3", lwd = 1)
text(Prune_Cancer,offset = 1,cex = 0.8, col = "dodgerblue4",pretty=0)
```



```
#Use predict function to predict the test set from the pruned model
Prune_pred=predict(Prune_Cancer,Cancer_test,type="class")

'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(Prune_pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F, prop.r = F,
           dnn=c('predicted default','actual defaut'))
```

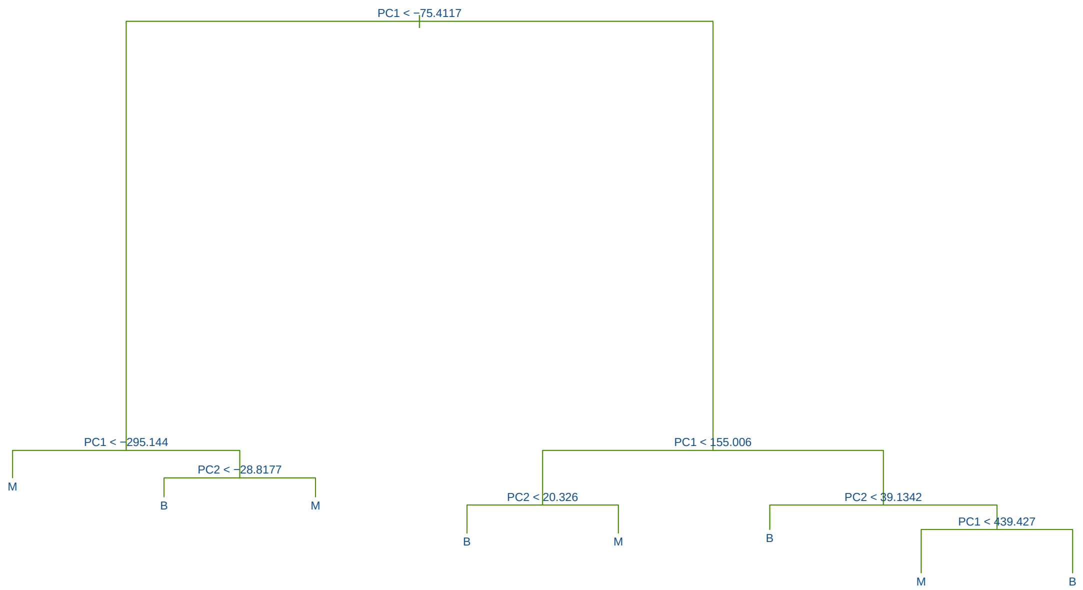
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        71 |        10 |        81 |
##                   |     0.497 |     0.070 |           |
## -------------------|-----------|-----------|-----------|
##                  M |         9 |        53 |        62 |
##                    |     0.063 |     0.371 |           |
## -------------------|-----------|-----------|-----------|
##       Column Total |        80 |        63 |       143 |
## -------------------|-----------|-----------|-----------|
##
##
```

Use the first two principle components of the data as inputs for Tree-Based Method (TBM)

```
#Use the first two principle components of the data as inputs for Tree-Based Method (TBM)
set.seed(123)

# Use tree function to fit a tree-based classifier
TBM_Cancer_pca=tree(Diagnosis ~PC1+PC2, data =Cancer_train_pc)
summary(TBM_Cancer_pca)
```

```
##
## Classification tree:
## tree(formula = Diagnosis ~ PC1 + PC2, data = Cancer_train_pc)
## Number of terminal nodes:  8
## Residual mean deviance:  0.2593 = 108.4 / 418
## Misclassification error rate: 0.0493 = 21 / 426
```

```
#Use plot function to display the tree structure
#Use the text function to display the node labels
plot(TBM_Cancer_pca, col = "chartreuse4", lwd = 1)
text(TBM_Cancer_pca,offset = 1,cex = 0.8, col = "dodgerblue4", pretty = 0)
```



```
#Use predict function to predict the test set from the developed model
TBM_pred_pca=predict(TBM_Cancer_pca,Cancer_test_pc,type="class")

'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```
CrossTable(TBM_pred_pca,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F,
           prop.r = F, dnn=c('predicted default','actual defaut'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                   | actual defaut
## predicted default |         B |         M | Row Total |
## ------------------|-----------|-----------|-----------|
##                 B |        78 |         9 |        87 |
##                   |     0.545 |     0.063 |           |
## ------------------|-----------|-----------|-----------|
##                 M |         2 |        54 |        56 |
##                   |     0.014 |     0.378 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```

Pruning the tree to improve the result

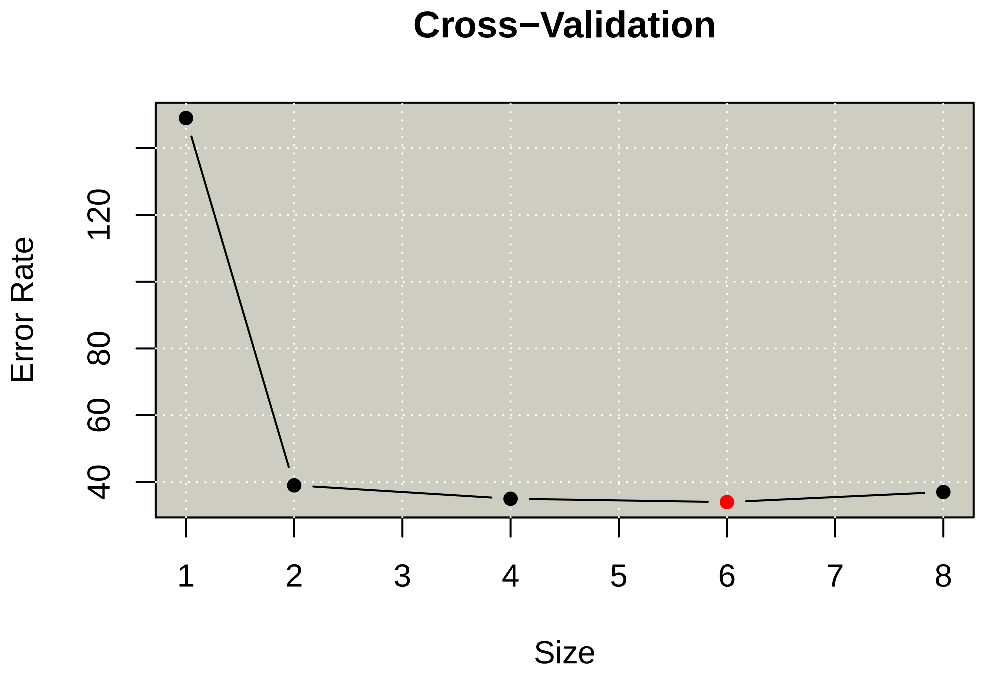
```
#Pruning the tree to improve the result
set.seed(123)

#Use cv.tree function to perform cross-validation
cv_Cancer_pca=cv.tree(TBM_Cancer_pca,FUN=prune.misclass)
names(cv_Cancer_pca)
```

```
## [1] "size"   "dev"    "k"      "method"
```

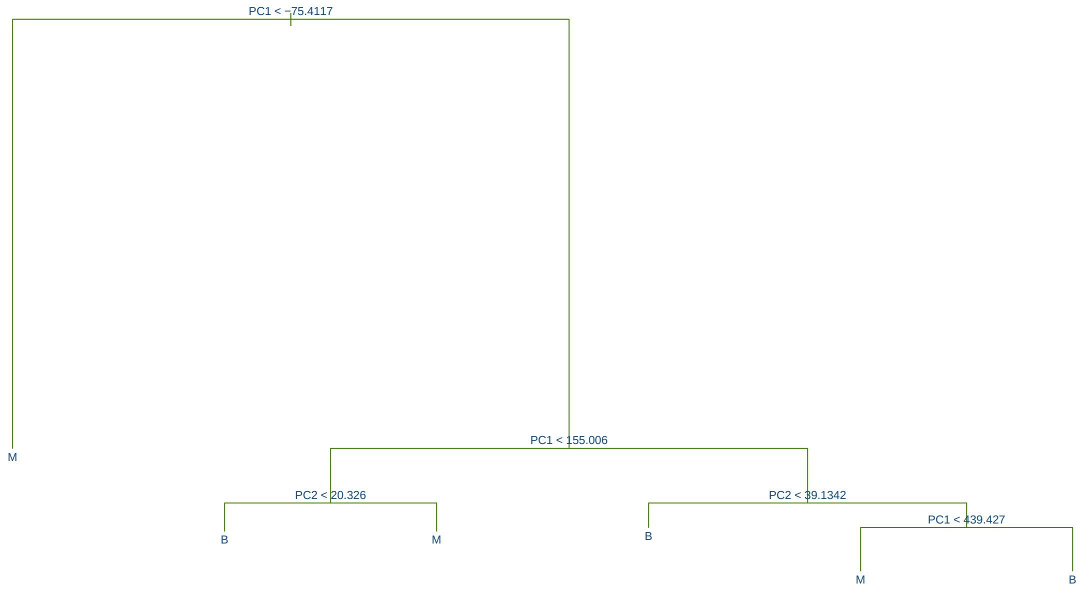
```
cv_Cancer_pca
```

```
## $size
## [1] 8 6 4 2 1
##
## $dev
## [1]  37  34  35  39 149
##
## $k
## [1]  -Inf   0.5   3.0   4.5 112.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
#Plot the error rate as a function of size
m=min(cv_Cancer_pca$dev)
Col=ifelse(cv_Cancer_pca$dev==m ,2,1)
plot(cv_Cancer_pca$size,cv_Cancer_pca$dev,type="b", xlab ="Size", ylab="Error Rate",
     main="Cross-Validation")
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory3")
grid(col = "white", lty = "dotted",lwd =1)
points(cv_Cancer_pca$size,cv_Cancer_pca$dev,type="b", col=Col, pch=16)
```



```
#best=6-------------------------------------------------------
#Use prune.misclass function in order to prune the tree to obtain the six-node tree
Prune_Cancer=prune.misclass(TBM_Cancer_pca,best=6)

#Use plot function to display the tree structure
#Use the text function to display the node labels
plot(Prune_Cancer,col = "chartreuse4", lwd = 1)
text(Prune_Cancer,offset = 1,cex = 0.8, col = "dodgerblue4",pretty=0)
```



```r
#Use predict function to predict the test set from the developed model
Prune_pred=predict(Prune_Cancer,Cancer_test_pc,type="class")

'Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,
determine how many observations are correctly classified and how many are misclassified'
```

```
## [1] "Use crossTable function to produce a confusion matrix for accuracy quantification (e.g.,\ndeterm
```

```r
CrossTable(Prune_pred,Cancer_test$Diagnosis,prop.chisq = F, prop.c = F, prop.r = F,
           dnn=c('predicted default','actual defaut'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  143 
## 
## 
##                   | actual defaut 
## predicted default |         B |         M | Row Total | 
## ------------------|-----------|-----------|-----------|
##                 B |        77 |         5 |        82 | 
##                   |     0.538 |     0.035 |           | 
## ------------------|-----------|-----------|-----------|
##                 M |         3 |        58 |        61 | 
##                   |     0.021 |     0.406 |           | 
## ------------------|-----------|-----------|-----------|
##      Column Total |        80 |        63 |       143 |
## ------------------|-----------|-----------|-----------|
##
##
```