R Final Project : Breast Cancer Classification :: Notebook 3.2

Utpal Mishra - 20207425

26 December 2020

Import Libraries

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(repr)
```

```
## Loading required package: repr
```

```
## Warning: package 'repr' was built under R version 3.6.3
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
library(gplots)
```

```
## Warning: package 'gplots' was built under R version 3.6.3
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.6.3
```

```
library(fitdistrplus)
```

```
## Warning: package 'fitdistrplus' was built under R version 3.6.3
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: survival
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v stringr 1.4.0
## v tidyr   1.1.2     v forcats 0.4.0
## v readr   1.3.1
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
## Warning: package 'tibble' was built under R version 3.6.3
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
## Warning: package 'purrr' was built under R version 3.6.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::%+%()   masks psych::%+%()
## x ggplot2::alpha() masks psych::alpha()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x MASS::select()   masks dplyr::select()
```

```
library(corpcor)
library("ggplot2", lib.loc="~/R/win-library/3.6")
library("GGally", lib.loc="~/R/win-library/3.6")
```

```
## Warning: package 'GGally' was built under R version 3.6.3
```

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

```
cat("IMPORTED LIBRARIES!!!")
```

```
## IMPORTED LIBRARIES!!!
```

Import Breast Cancer Data

```
library(readxl) #reading data using the function read.csv() from the library readxl

data <- read.csv("E:/UCD/Lectures/Semester 1/Data Programming with R/Final Project/breast-cancer-wisconsin_wdbc.csv")
data <- data[c(-1)]
head(data) #View(data) #fix(data) #display first 5 rows of the data
```

```
##   diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 1                                 M         17.99          10.38
## 2                                 M         20.57          17.77
## 3                                 M         19.69          21.25
## 4                                 M         11.42          20.38
## 5                                 M         20.29          14.34
## 6                                 M         12.45          15.70
##   perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 1           122.80      1001.0           0.11840            0.27760
## 2           132.90      1326.0           0.08474            0.07864
## 3           130.00      1203.0           0.10960            0.15990
## 4            77.58       386.1           0.14250            0.28390
## 5           135.10      1297.0           0.10030            0.13280
## 6            82.57       477.1           0.12780            0.17000
##   concavity..nucA. concave.points..nucA. symmetry..nucA.
## 1           0.3001               0.14710          0.2419
## 2           0.0869               0.07017          0.1812
## 3           0.1974               0.12790          0.2069
## 4           0.2414               0.10520          0.2597
## 5           0.1980               0.10430          0.1809
## 6           0.1578               0.08089          0.2087
##   fractal.dimension..nucA. radius..nucB. texture..nucB. perimeter..nucB.
## 1                  0.07871        1.0950         0.9053            8.589
## 2                  0.05667        0.5435         0.7339            3.398
## 3                  0.05999        0.7456         0.7869            4.585
## 4                  0.09744        0.4956         1.1560            3.445
## 5                  0.05883        0.7572         0.7813            5.438
## 6                  0.07613        0.3345         0.8902            2.217
##   area..nucB. smoothness..nucB. compactness..nucB. concavity..nucB.
## 1      153.40          0.006399            0.04904          0.05373
## 2       74.08          0.005225            0.01308          0.01860
## 3       94.03          0.006150            0.04006          0.03832
## 4       27.23          0.009110            0.07458          0.05661
## 5       94.44          0.011490            0.02461          0.05688
## 6       27.19          0.007510            0.03345          0.03672
##   concave.points..nucB. symmetry..nucB. fractal.dimension..nucB. radius..nucC.
## 1               0.01587         0.03003                 0.006193         25.38
## 2               0.01340         0.01389                 0.003532         24.99
## 3               0.02058         0.02250                 0.004571         23.57
## 4               0.01867         0.05963                 0.009208         14.91
## 5               0.01885         0.01756                 0.005115         22.54
## 6               0.01137         0.02165                 0.005082         15.47
##   texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 1          17.33           184.60      2019.0            0.1622
## 2          23.41           158.80      1956.0            0.1238
## 3          25.53           152.50      1709.0            0.1444
## 4          26.50            98.87       567.7            0.2098
## 5          16.67           152.20      1575.0            0.1374
## 6          23.75           103.40       741.6            0.1791
##   compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 1             0.6656           0.7119                0.2654          0.4601
## 2             0.1866           0.2416                0.1860          0.2750
## 3             0.4245           0.4504                0.2430          0.3613
## 4             0.8663           0.6869                0.2575          0.6638
## 5             0.2050           0.4000                0.1625          0.2364
## 6             0.5249           0.5355                0.1741          0.3985
##   fractal.dimension..nucC.
## 1                  0.11890
## 2                  0.08902
## 3                  0.08758
## 4                  0.17300
## 5                  0.07678
## 6                  0.12440
```

Feature Selection based on evaluations on previous notebooks

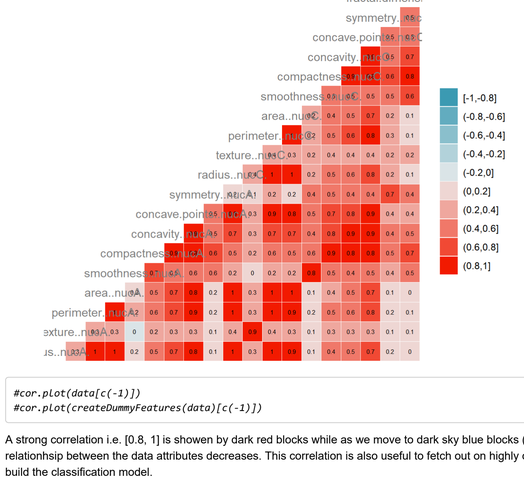
```
data = data[, c("diagnosis..M.malignant..B.benign.", "radius..nucA.", "texture..nucA.", "perimeter..nucA.", "area..nucA.",
"smoothness..nucA.", "compactness..nucA.", "concavity..nucA.", "concave.points..nucA.", "symmetry..nucA.", "radius..nucC.",
"texture..nucC.", "perimeter..nucC.", "area..nucC.", "smoothness..nucC.", "compactness..nucC.", "concavity..nucC.", "concav
e.points..nucC.", "symmetry..nucC.", "fractal.dimension..nucC.")]
head(data)
```

```
##   diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 1                                 M         17.99          10.38
## 2                                 M         20.57          17.77
## 3                                 M         19.69          21.25
## 4                                 M         11.42          20.38
## 5                                 M         20.29          14.34
## 6                                 M         12.45          15.70
##   perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 1           122.80      1001.0           0.11840            0.27760
## 2           132.90      1326.0           0.08474            0.07864
## 3           130.00      1203.0           0.10960            0.15990
## 4            77.58       386.1           0.14250            0.28390
## 5           135.10      1297.0           0.10030            0.13280
## 6            82.57       477.1           0.12780            0.17000
##   concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 1           0.3001              0.14710          0.2419         25.38
## 2           0.0869              0.07017          0.1812         24.99
## 3           0.1974              0.12790          0.2069         23.57
## 4           0.2414              0.10520          0.2597         14.91
## 5           0.1980              0.10430          0.1809         22.54
## 6           0.1578              0.08089          0.2087         15.47
##   texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 1          17.33           184.60      2019.0            0.1622
## 2          23.41           158.80      1956.0            0.1238
## 3          25.53           152.50      1709.0            0.1444
## 4          26.50            98.87       567.7            0.2098
## 5          16.67           152.20      1575.0            0.1374
## 6          23.75           103.40       741.6            0.1791
##   compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 1             0.6656           0.7119                0.2654          0.4601
## 2             0.1866           0.2416                0.1860          0.2750
## 3             0.4245           0.4504                0.2430          0.3613
## 4             0.8663           0.6869                0.2575          0.6638
## 5             0.2050           0.4000                0.1625          0.2364
## 6             0.5249           0.5355                0.1741          0.3985
##   fractal.dimension..nucC.
## 1                  0.11890
## 2                  0.08902
## 3                  0.08758
## 4                  0.17300
## 5                  0.07678
## 6                  0.12440
```

Correlation Plot

Finding correlation values between the features of the data to understand the degree of correleation.

```
ggcorr(data[c(-1)], nbreaks = 10, label = TRUE, label_size = 2, color = "grey50") #finding the correlation between the data features
```



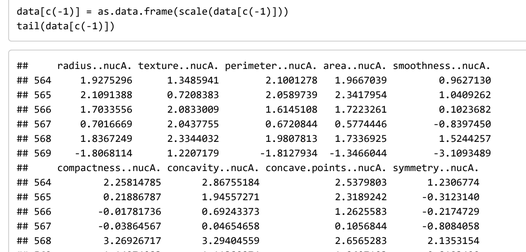

```
#cor.plot(data[c(-1)])
#cor.plot(createDummyFeatures(data)[c(-1)])
```

A strong correlation i.e. [0.8, 1] is shown by dark red blocks while as we move to dark sky blue blocks (lowest correlation), the strength of relationhsip between the data attributes decreases. This correlation is also useful to fetch out on highly correlated features, preprocess them and build the classification model.

Boxplot

Boxplot in an effective plot to visualize the presence of outliers in the data. As can be seen, from the plot there are 2 features nucA and nucC specifically that contains high number of outliers.

```
boxplot(data[c(-1)], col = "red", main = "Finding Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Calulated Values")  #using boxplot to find the outliers
```



As can be compared from the above two boxplots, the outliers for the columns nucA and nucC are removed in the later one with change in the y-scale from the multiple iterations

Standardizing the Data

```
tail(data[c(-1)])
```

```
##     radius..nucA. texture..nucA. perimeter..nucA. area..nucA. smoothness..nucA.
## 564         20.92          25.09           143.00      1347.0           0.10990
## 565         21.56          22.39           142.00      1479.0           0.11100
## 566         20.13          28.25           131.20      1261.0           0.09780
## 567         16.60          28.08           108.30       858.1           0.08455
## 568         20.60          29.33           140.10      1265.0           0.11780
## 569          7.76          24.54            47.92       181.0           0.05263
##     compactness..nucA. concavity..nucA. concave.points..nucA. symmetry..nucA.
## 564            0.22360          0.31740               0.14740          0.2149
## 565            0.11590          0.24390               0.13890          0.1726
## 566            0.10340          0.14400               0.09791          0.1752
## 567            0.10230          0.09251               0.05302          0.1590
## 568            0.27700          0.35140               0.15200          0.2397
## 569            0.04362          0.00000               0.00000          0.1587
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 564        24.290          29.41           179.10      1819.0           0.14070
## 565        25.450          26.40           166.10      2027.0           0.14100
## 566        23.690          38.25           155.00      1731.0           0.11660
## 567        18.980          34.12           126.70      1124.0           0.11390
## 568        25.740          39.42           184.60      1821.0           0.16500
## 569         9.456          30.37            59.16       268.6           0.08996
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 564            0.41860           0.6599                0.2542          0.2929
## 565            0.21130           0.4107                0.2216          0.2060
## 566            0.19220           0.3215                0.1628          0.2572
## 567            0.30940           0.3403                0.1418          0.2218
## 568            0.86810           0.9387                0.2650          0.4087
## 569            0.06444           0.0000                0.0000          0.2871
##     fractal.dimension..nucC.
## 564                  0.09873
## 565                  0.07115
## 566                  0.06637
## 567                  0.07820
## 568                  0.12400
## 569                  0.07039
```

```
data[c(-1)] = as.data.frame(scale(data[c(-1)]))
tail(data[c(-1)])
```

```
##     radius..nucA. texture..nucA. perimeter..nucA. area..nucA. smoothness..nucA.
## 564     1.9275296      1.3485941        2.1001278   1.9667039         0.9627130
## 565     2.1091388      0.7208383        2.0589739   2.3417954         1.0409262
## 566     1.7033556      2.0833009        1.6145108   1.7223261         0.1023682
## 567     0.7016669      2.0437755        0.6720844   0.5774446        -0.8397450
## 568     1.8367249      2.3344032        1.9807813   1.7336925         1.5244257
## 569    -1.8068114      1.2207179       -1.8127934  -1.3466044        -3.1093489
##     compactness..nucA. concavity..nucA. concave.points..nucA. symmetry..nucA.
## 564         2.25814785       2.86755184              2.5379803       1.2306774
## 565         0.21886787       1.94557271              2.3189242      -0.3123140
## 566        -0.01781736       0.69243373              1.2625583      -0.2174729
## 567        -0.03864567       0.04654658              0.1056844      -0.8084058
## 568         3.26926717       3.29404559              2.6565283       2.1353154
## 569        -1.14974083      -1.11389274             -1.2607103      -0.8193490
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 564     1.6595095      0.6073251        2.1378974   1.6482047         0.3648935
## 565     1.8995140      0.1175962        1.7510219   2.0135291         0.3780327
## 566     1.5353692      2.0455987        1.4206897   1.4936444        -0.6906227
## 567     0.5608679      1.3736451        0.5784916   0.4275294        -0.8088756
## 568     1.9595152      2.2359585        2.3015755   1.6517174         1.4291692
## 569    -1.4096522      0.7635178       -1.4314754  -1.0748672        -1.8573842
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 564          1.0444809        1.8584199              2.1236696      0.04565289
## 565         -0.2730774        0.6639281              1.6277189     -1.35896255
## 566         -0.3944733        0.2363652              0.7331821     -0.53138705
## 567          0.3504270        0.3264793              0.4137047     -1.10357792
## 568          3.9014151        3.1947936              2.2879723      1.91739590
## 569         -1.2064909       -1.3046827             -1.7435287     -0.04809589
##     fractal.dimension..nucC.
## 564                0.8185573
## 565               -0.7084673
## 566               -0.9731220
## 567               -0.3181292
## 568                2.2176840
## 569               -0.7505463
```

Feature Selection

```
#install.packages('Boruta')
library(Boruta)
```

```
## Warning: package 'Boruta' was built under R version 3.6.3
```

```
# Perform Boruta search
boruta_output <- Boruta(diagnosis..M.malignant..B.benign. ~ ., data=na.omit(data), doTrace=0)
#print(names(boruta_output))

boruta_signif <- getSelectedAttributes(boruta_output, withTentative = TRUE)
#print(boruta_signif)

roughFixMod <- TentativeRoughFix(boruta_output)
```

```
## Warning in TentativeRoughFix(boruta_output): There are no Tentative attributes!
## Returning original object.
```

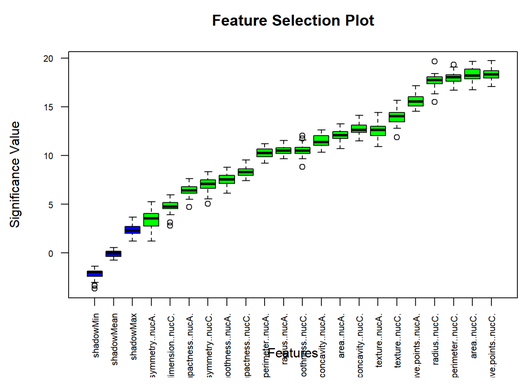
```
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)
```

```
##  [1] "radius..nucA."            "texture..nucA."
##  [3] "perimeter..nucA."         "area..nucA."
##  [5] "smoothness..nucA."        "compactness..nucA."
##  [7] "concavity..nucA."         "concave.points..nucA."
##  [9] "symmetry..nucA."          "radius..nucC."
## [11] "texture..nucC."           "perimeter..nucC."
## [13] "area..nucC."              "smoothness..nucC."
## [15] "compactness..nucC."       "concavity..nucC."
## [17] "concave.points..nucC."    "symmetry..nucC."
## [19] "fractal.dimension..nucC."
```

```
# Variable Importance Scores
imps <- attStats(roughFixMod)
imps2 = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]
head(imps2[order(-imps2$meanImp), ])  # descending sort
```

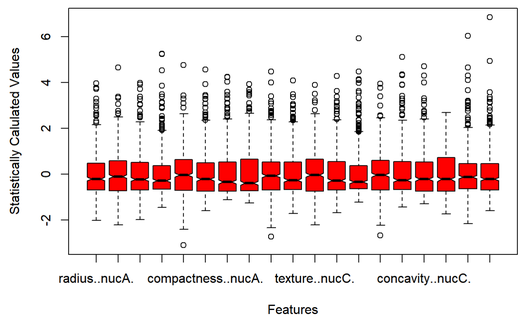
```
##                          meanImp  decision
## concave.points..nucC. 18.34227 Confirmed
## area..nucC.           18.30653 Confirmed
## perimeter..nucC.      17.97840 Confirmed
## radius..nucC.         17.65487 Confirmed
## concave.points..nucA. 15.59765 Confirmed
## texture..nucC.        13.98691 Confirmed
```

```
# Plot variable importance
plot(boruta_output, cex.axis=.7, las=2, xlab="Features", ylab = "Significance Value", main="Feature Selection Plot")
```



Removing Outliers

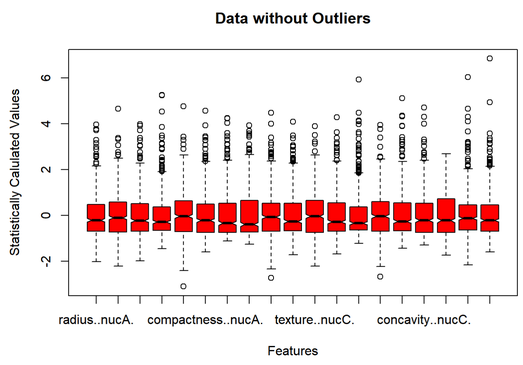
```
boxplot(data[c(-1)], col = "red", main = "Data witho Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Calul
ated Values")  #using boxplot to represent data with outliers
```




```
outliers <- function(x) {

  Q1 <- quantile(x, probs=.25)
  Q3 <- quantile(x, probs=.75)
  iqr = Q3-Q1

 upper_limit = Q3 + (iqr*1.5)
 lower_limit = Q1 - (iqr*1.5)

 x > upper_limit | x < lower_limit
}

remove_outliers <- function(data, cols = names(data)) {
  for (col in cols) {
    data <- data[!outliers(data[[col]]),]
  }
  head(data)
}

remove_outliers(data, c(names(data[, c(-1)]))) #function to remove outliers
```

```
##    diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 11                                 M     0.5370834      0.9184652
## 14                                 M     0.4888435      1.0835417
## 17                                 M     0.1568390      0.1953835
## 20                                 B    -0.1666526     -1.1461538
## 21                                 B    -0.2971842     -0.8322759
## 22                                 B    -1.3119261     -1.5925579
##    perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 11        0.4416221  0.40609593        -1.0167912        -0.71291456
## 14        0.4827761  0.36318774        -0.8781405        -0.07840879
## 17        0.1140363  0.08414239         0.1642277        -0.61237067
## 20       -0.1855647 -0.25173500         0.1016571        -0.43646621
## 21       -0.2608765 -0.38330119         0.7920661         0.42904436
## 22       -1.3016609 -1.08261951         0.4294414        -0.74642919
##    concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 11       -0.7000684           -0.40432978     -1.03456525     0.6043170
## 14        0.1327234            0.12166258      0.12906182     0.1181009
## 17       -0.1862688            0.09460271     -0.82299669     0.5794890
## 20       -0.2779650           -0.02858414      0.26767570    -0.2398369
## 21       -0.5408858           -0.45922267      0.56678987    -0.3660462
## 22       -0.7430941           -0.72569797      0.01233434    -1.2495113
##    texture..nucC. perimeter..nucC.  area..nucC. smoothness..nucC.
## 11      1.3345970        0.4921886  0.473194982      -0.624926668
## 14      0.3225990        0.1410247 -0.007171473      -0.843913439
## 17      0.8464951        0.4802847  0.452118574       0.614538459
## 20     -1.0440863       -0.2250191 -0.297498987       0.509424808
## 21     -0.8439645       -0.3324514 -0.439237827      -0.051181327
## 22     -1.6298085       -1.2538103 -0.993547345       0.001375499
##    compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 11         -0.6302737       -0.60533934           -0.22601086       0.0763637
## 14         -0.3932021       -0.19167703           -0.04117035      -0.1483101
## 17         -0.4268879        0.09208668            0.70427701       0.2072887
## 20         -0.4891748       -0.15908255            0.21593293       0.1232381
## 21          0.1483124       -0.39874785           -0.63555051       0.4578243
## 22         -0.8864126       -0.87966023           -0.79620203      -0.7285828
##    fractal.dimension..nucC.
## 11               0.03179084
## 14              -1.16690689
## 17              -0.09887552
## 20              -0.62873867
## 21              -0.11714666
## 22              -0.34415178
```

```
boxplot(data[c(-1)], col = "red", main = "Data without Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Cal
ulated Values")  #using boxplot to represent data without outliers
```




```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

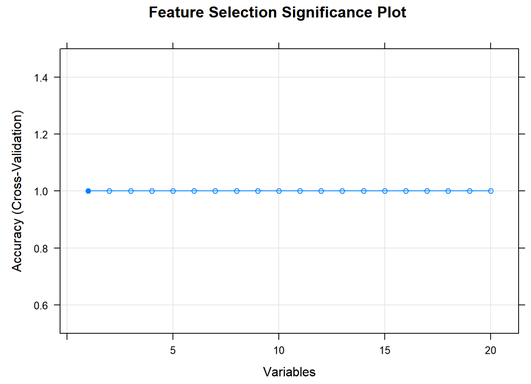
```
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(data[, c(1:dim(data)[2])], data[, c(1)], sizes=c(1:dim(data)[2]), rfeControl=control)
# summarize the results
print(results)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy Kappa AccuracySD KappaSD Selected
##          1        1     1          0       0        *
##          2        1     1          0       0
##          3        1     1          0       0
##          4        1     1          0       0
##          5        1     1          0       0
##          6        1     1          0       0
##          7        1     1          0       0
##          8        1     1          0       0
##          9        1     1          0       0
##         10        1     1          0       0
##         11        1     1          0       0
##         12        1     1          0       0
##         13        1     1          0       0
##         14        1     1          0       0
##         15        1     1          0       0
##         16        1     1          0       0
##         17        1     1          0       0
##         18        1     1          0       0
##         19        1     1          0       0
##         20        1     1          0       0
##
## The top 1 variables (out of 1):
##    diagnosis..M.malignant..B.benign.
```

```
# list the chosen features
predictors(results)
```

```
## [1] "diagnosis..M.malignant..B.benign."
```

```
# plot the results
plot(results, type=c("g", "o"), main = "Feature Selection Significance Plot")
```

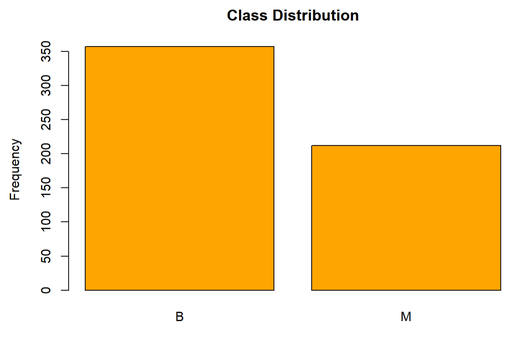


Resampling

```
prop.table(table(data$diagnosis..M.malignant..B.benign.))
```

```
##
##         B         M
## 0.6274165 0.3725835
```

```
barplot(table(data$diagnosis..M.malignant..B.benign.), col = "orange", xlab = "Dignosis", ylab = "Frequency", main = "Class
 Distribution")
```




```
#install.packages("ROSE")
require(ROSE)
```

```
## Loading required package: ROSE
```

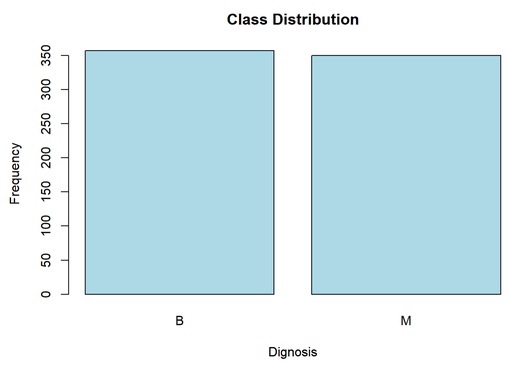
```
## Warning: package 'ROSE' was built under R version 3.6.3
```

```
## Loaded ROSE 0.0-3
```

```
data = ovun.sample(diagnosis..M.malignant..B.benign. ~ ., data = data, method = "over", seed = 1)$data

prop.table(table(ovun.sample(diagnosis..M.malignant..B.benign. ~ ., data = data, method = "over", seed = 1)$data$diagnosis..
M.malignant..B.benign.))
```

```
##
##         B         M
## 0.5049505 0.4950495
```

```
barplot(table(ovun.sample(diagnosis..M.malignant..B.benign. ~ ., data = data, method = "over", seed = 1)$data$diagnosis..M.m
alignant..B.benign.), col = "lightblue", xlab = "Dignosis", ylab = "Frequency", main = "Class Distribution")
```



Building Classfication Model

Spliting the Data into Training and Testing Data

```
library(caTools) #using caTools to split the data into training and testing sets

data[c(-1)] = scale(data[c(-1)])
#data$diagnosis..M.malignant..B.benign. = factor(data$diagnosis..M.malignant..B.benign., levels = c(0, 1))
sample.split(data$diagnosis..M.malignant..B.benign., SplitRatio = 0.80) -> split_data

subset(data, split_data == TRUE) -> train_data
subset(data, split_data == FALSE) -> test_data
```

Decision Tree

Fitting Model

```
library(rpart) #using rpart function to build a decision tree classification model

rpart(diagnosis..M.malignant..B.benign. ~., data = train_data) -> dtmodel #fitting the model
summary(dtmodel) #model summary
```

```
## Call:
## rpart(formula = diagnosis..M.malignant..B.benign. ~ ., data = train_data)
##   n= 566
##
##           CP nsplit  rel error    xerror       xstd
## 1 0.85357143      0 1.00000000 1.0821429 0.04237727
## 2 0.03035714      1 0.14642857 0.1571429 0.02275073
## 3 0.01785714      3 0.08571429 0.1178571 0.01990922
## 4 0.01000000      4 0.06785714 0.1107143 0.01933265
##
## Variable importance
##      perimeter..nucC.         radius..nucC.           area..nucC.
##                    18                    16                    16
##      perimeter..nucA.         radius..nucA.           area..nucA.
##                    15                    15                    14
##       texture..nucA.       symmetry..nucC.       texture..nucC.
##                     1                     1                     1
## concave.points..nucC.
##                     1
##
## Node number 1: 566 observations,    complexity param=0.8535714
##   predicted class=B  expected loss=0.4946996  P(node) =1
##     class counts:   286   280
##    probabilities: 0.505 0.495
##   left son=2 (275 obs) right son=3 (291 obs)
##   Primary splits:
##       perimeter..nucC.      < -0.2602409  to the left,  improve=207.2506, (0 missing)
##       area..nucC.           < -0.1905673  to the left,  improve=197.8728, (0 missing)
##       radius..nucC.         < -0.09050281 to the left,  improve=197.2428, (0 missing)
##       concave.points..nucC. < 0.2325323   to the left,  improve=195.1516, (0 missing)
##       concave.points..nucA. < -0.1682351  to the left,  improve=191.9079, (0 missing)
##   Surrogate splits:
##       radius..nucC.    < -0.281926   to the left,  agree=0.970, adj=0.938, (0 split)
##       area..nucC.      < -0.3428518  to the left,  agree=0.966, adj=0.931, (0 split)
##       perimeter..nucA. < -0.2659997  to the left,  agree=0.935, adj=0.865, (0 split)
##       radius..nucA.    < -0.1867334  to the left,  agree=0.928, adj=0.851, (0 split)
##       area..nucA.      < -0.2995772  to the left,  agree=0.926, adj=0.847, (0 split)
##
## Node number 2: 275 observations,    complexity param=0.01785714
##   predicted class=B  expected loss=0.05454545  P(node) =0.4858657
##     class counts:   260    15
##    probabilities: 0.945 0.055
##   left son=4 (268 obs) right son=5 (7 obs)
##   Primary splits:
##       concave.points..nucC. < 0.6988829   to the left,  improve=9.253828, (0 missing)
##       symmetry..nucC.       < 1.497247    to the left,  improve=7.970378, (0 missing)
##       compactness..nucC.    < 0.8405162   to the left,  improve=7.757576, (0 missing)
##       smoothness..nucC.     < 1.963936    to the left,  improve=6.972659, (0 missing)
##       concave.points..nucA. < 0.4058198   to the left,  improve=6.252762, (0 missing)
##   Surrogate splits:
##       symmetry..nucC.       < 1.497247    to the left,  agree=0.996, adj=0.857, (0 split)
##       concave.points..nucA. < 0.4058198   to the left,  agree=0.985, adj=0.429, (0 split)
##       compactness..nucC.    < 0.8405162   to the left,  agree=0.985, adj=0.429, (0 split)
##       concavity..nucC.      < 0.9145522   to the left,  agree=0.985, adj=0.429, (0 split)
##       concavity..nucA.      < 0.6832421   to the left,  agree=0.982, adj=0.286, (0 split)
##
## Node number 3: 291 observations,    complexity param=0.03035714
##   predicted class=M  expected loss=0.08934708  P(node) =0.5141343
##     class counts:    26   265
##    probabilities: 0.089 0.911
##   left son=6 (63 obs) right son=7 (228 obs)
##   Primary splits:
##       perimeter..nucC.      < 0.01224012  to the left,  improve=13.67475, (0 missing)
##       concave.points..nucA. < -0.1774817  to the left,  improve=13.49701, (0 missing)
##       concave.points..nucC. < 0.2431484   to the left,  improve=13.08955, (0 missing)
##       area..nucC.           < -0.1740817  to the left,  improve=12.31691, (0 missing)
##       concavity..nucA.      < -0.2650965  to the left,  improve=12.04480, (0 missing)
##   Surrogate splits:
##       radius..nucC.    < 0.02592988  to the left,  agree=0.942, adj=0.730, (0 split)
##       area..nucC.      < -0.1348657  to the left,  agree=0.935, adj=0.698, (0 split)
##       perimeter..nucA. < 0.02419194  to the left,  agree=0.890, adj=0.492, (0 split)
##       area..nucA.      < -0.2104802  to the left,  agree=0.876, adj=0.429, (0 split)
##       radius..nucA.    < 0.09922928  to the left,  agree=0.873, adj=0.413, (0 split)
##
## Node number 4: 268 observations
##   predicted class=B  expected loss=0.03358209  P(node) =0.4734982
##     class counts:   259     9
##    probabilities: 0.966 0.034
##
## Node number 5: 7 observations
##   predicted class=M  expected loss=0.1428571  P(node) =0.01236749
##     class counts:     1     6
##    probabilities: 0.143 0.857
##
## Node number 6: 63 observations,    complexity param=0.03035714
##   predicted class=M  expected loss=0.3809524  P(node) =0.1113074
##     class counts:    24    39
##    probabilities: 0.381 0.619
##   left son=12 (25 obs) right son=13 (38 obs)
##   Primary splits:
##       texture..nucA.           < -0.03561389 to the left,  improve=17.46797, (0 missing)
##       texture..nucC.           < -0.7177071  to the left,  improve=15.08929, (0 missing)
##       smoothness..nucC.        < 0.2924185   to the left,  improve=12.90090, (0 missing)
##       area..nucA.              < -0.2870605  to the right, improve=12.03008, (0 missing)
##       fractal.dimension..nucC. < 0.422162    to the left,  improve=11.17725, (0 missing)
##   Surrogate splits:
##       texture..nucC.    < -0.12453    to the left,  agree=0.921, adj=0.80, (0 split)
##       symmetry..nucC.   < 0.3607312   to the left,  agree=0.746, adj=0.36, (0 split)
##       smoothness..nucC. < 0.2924185   to the left,  agree=0.730, adj=0.32, (0 split)
##       radius..nucA.     < -0.1554137  to the right, agree=0.714, adj=0.28, (0 split)
##       perimeter..nucA.  < 0.005993145 to the right, agree=0.714, adj=0.28, (0 split)
##
## Node number 7: 228 observations
##   predicted class=M  expected loss=0.00877193  P(node) =0.4028269
##     class counts:     2   226
##    probabilities: 0.009 0.991
##
## Node number 12: 25 observations
##   predicted class=B  expected loss=0.16  P(node) =0.04416961
##     class counts:    21     4
##    probabilities: 0.840 0.160
##
## Node number 13: 38 observations
##   predicted class=M  expected loss=0.07894737  P(node) =0.06713781
##     class counts:     3    35
##    probabilities: 0.079 0.921
```

Predictions

```
library(caret) #using caret to make model predictions

predict(dtmodel, test_data, type = "class") -> dtresult
#table(test_data$diagnosis..M.malignant..B.benign., dtresult)
```

Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., dtresult)) #the maximum accuracy of the model is 94.33
```

```
## Confusion Matrix and Statistics
##
##    dtresult
##      B  M
##   B 66  5
##   M  4 66
##
##               Accuracy : 0.9362
##                 95% CI : (0.8823, 0.9704)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8723
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9429
##            Specificity : 0.9296
##         Pos Pred Value : 0.9296
##         Neg Pred Value : 0.9429
##             Prevalence : 0.4965
##         Detection Rate : 0.4681
##   Detection Prevalence : 0.5035
##      Balanced Accuracy : 0.9362
##
##       'Positive' Class : B
##
```

Tree Model

```
#install.packages("party")
library(party)
```

```
## Warning: package 'party' was built under R version 3.6.3
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 3.6.3
```

```
## Loading required package: modeltools
```

```
## Warning: package 'modeltools' was built under R version 3.6.3
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Warning: package 'strucchange' was built under R version 3.6.3
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 3.6.3
```

```
##
## Attaching package: 'strucchange'
```

```
## The following object is masked from 'package:stringr':
##
##     boundary
```

```
plot(ctree(diagnosis..M.malignant..B.benign. ~., data = train_data)) #tree model
```

Random Forest

Fitting Model

```
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

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

```
randomForest(formula = diagnosis..M.malignant..B.benign. ~., data = train_data) -> rfmodel
summary(rfmodel)
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted        566   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes           1132   matrix numeric
## oob.times        566   -none- numeric
## classes            2   -none- character
## importance        19   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                566   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

Predictions

```
predict(rfmodel, test_data, type = "class") -> rfresult #using caret to make model predictions
#table(test_data$diagnosis..M.malignant..B.benign., rfresult)
```

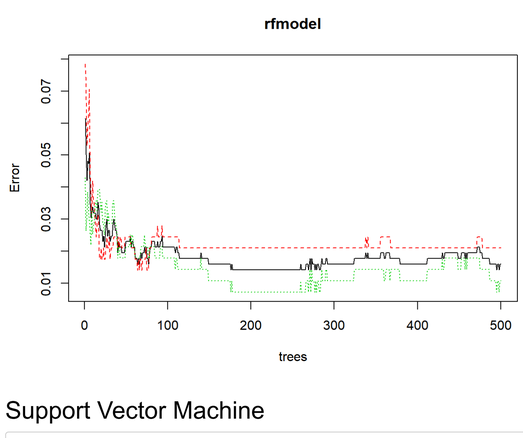
Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., rfresult)) #the maximum accuracy of the model is 98.58
```

```
## Confusion Matrix and Statistics
##
##    rfresult
##      B  M
##   B 70  1
##   M  2 68
##
##                Accuracy : 0.9787
##                  95% CI : (0.9391, 0.9956)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9574
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9722
##             Specificity : 0.9855
##          Pos Pred Value : 0.9859
##          Neg Pred Value : 0.9714
##              Prevalence : 0.5106
##          Detection Rate : 0.4965
##    Detection Prevalence : 0.5035
##       Balanced Accuracy : 0.9789
##
##        'Positive' Class : B
##
```

Error vs Model Plot

```
plot(rfmodel)
```



Support Vector Machine

```
#install.packages('e1071')
library(e1071) #using library e1071 to build a SVM classification model
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

Fitting Model

```
svm(diagnosis..M.malignant..B.benign. ~., data = train_data, type = 'C-classification', kernel = 'linear') -> svmmodel #fitt
ing the model
summary(svmmodel) #model summary
```

```
##
## Call:
## svm(formula = diagnosis..M.malignant..B.benign. ~ ., data = train_data,
##     type = "C-classification", kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  45
##
##  ( 22 23 )
##
##
## Number of Classes:  2
##
## Levels:
##  B M
```

Predictions

```
predict(svmmodel, test_data, type = "class") -> svmresult #using caret to make model predictions
#table(test_data$diagnosis..M.malignant..B.benign., svmresult)
```

Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., svmresult)) #the maximum accuracy of the model is 99.29
```

```
## Confusion Matrix and Statistics
##
##    svmresult
##      B  M
##   B 69  2
##   M  2 68
##
##                Accuracy : 0.9716
##                  95% CI : (0.929, 0.9922)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9433
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9714
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9714
##              Prevalence : 0.5035
##          Detection Rate : 0.4894
##    Detection Prevalence : 0.5035
##       Balanced Accuracy : 0.9716
##
##        'Positive' Class : B
##
```

Naive Bayes

```
#install.packages('e1071')
#library(e1071) #using library e1071 to build a Naive Bayes classification model
```

Fitting Model

```
naiveBayes(diagnosis..M.malignant..B.benign. ~., data = train_data, laplace = 1) -> nbmodel #fitting the model
summary(nbmodel) #model summary
```

```
##           Length Class  Mode
## apriori    2     table  numeric
## tables    19     -none- list
## levels     2     -none- character
## isnumeric 19     -none- logical
## call       4     -none- call
```

Predictions

```
predict(nbmodel, test_data, type = "class") -> nbresult #using caret to make model predictions
#table(test_data$diagnosis..M.malignant..B.benign., nbresult)
```

Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., nbresult)) #the maximum accuracy of the model is 92.20
```

```
## Confusion Matrix and Statistics
##
##    nbresult
##      B  M
##   B 65  6
##   M  5 65
##
##                Accuracy : 0.922
##                  95% CI : (0.8647, 0.9604)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.844
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9286
##             Specificity : 0.9155
##          Pos Pred Value : 0.9155
##          Neg Pred Value : 0.9286
##              Prevalence : 0.4965
##          Detection Rate : 0.4610
##    Detection Prevalence : 0.5035
##       Balanced Accuracy : 0.9220
##
##        'Positive' Class : B
##
```

KNN

```
# require(class)
#
# knn(train, test, cl = train$diagnosis..M.malignant..B.benign., k=3) -> knnmodel
# confusionMatrix(table(test$diagnosis..M.malignant..B.benign., knnmodel)) #the maximum accuracy of the model is 98.78
```

Neural Network: Model 1

```
#install.packages('neuralnet')
library(neuralnet) #using library neuralnet to build a neural network classification model
```

```
## Warning: package 'neuralnet' was built under R version 3.6.3
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
train = train_data #creating dummy training data
test = test_data #creating dummy testing data
```

Categorical Encoding

```
train$diagnosis..M.malignant..B.benign. <- ifelse(train$diagnosis..M.malignant..B.benign. %in% c("B", "B"), 0, 1) #encoding
 the categorical/ response variable in training data
tail(train)
```

```
##     diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 700                                 1    -0.5557615     0.59340586
## 701                                 1     0.9366915     0.02592065
## 703                                 1     0.7242621     1.09707747
## 704                                 1     1.1872494     1.12442615
## 706                                 1     1.1246099     0.32903524
## 707                                 1     1.3588270     0.38601166
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 700      -0.5933801  -0.5773660       -0.47289785        -1.00322816
## 701       0.9052508   0.8592222       -0.17424645        -0.04285865
## 703       0.7667818   0.7100991        0.57750598         0.91582894
## 704       1.4116520   1.0826376        0.02851443         2.51364161
## 706       1.0635012   1.1014799       -0.50656933        -0.15685465
## 707       1.3048330   1.3706551       -0.23280555         0.36079928
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 700      -0.6762943            -0.7303832      -0.8858445    -0.5453797
## 701       0.1403738             0.4592732      -0.1033538     1.1103326
## 703       1.4253281             1.2268752      -1.0618114     1.1616419
## 704       1.3082877             1.4168757       2.1542629     0.7314330
## 706       0.0818536             0.5988603      -0.8970765     1.5010728
## 707       0.3209149             0.7863274       1.0198386     1.6727617
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 700     1.11232436       -0.6271629  -0.5589975        0.35013361
## 701     0.19690785        1.1178567   1.0269589        0.45224492
## 703     0.94559351        1.1777740   1.1168809        0.15922985
## 704     0.56243318        1.0750587   0.5706884       -1.34580213
## 706     0.02376427        1.4802136   1.4599164       -0.65322104
## 707     0.35401962        1.6057546   1.6980430        0.05711854
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 700         -0.7602553       -0.4087068            -0.5013444      -0.1751346
## 701         -0.1028402        0.3578675             1.0135748       0.1991716
## 703          0.6800645        0.8854539             0.6935749      -1.2452666
## 704          0.7256403        0.2886993             0.7633379       0.3799264
## 706         -0.2526784       -0.1654256             0.7967028      -0.6198233
## 707          0.3085906        0.2782049             0.8437170       0.2103688
##     fractal.dimension..nucC.
## 700              -0.24109716
## 701              -0.07589979
## 703               0.43560834
## 704               0.94601881
## 706              -1.05226564
## 707              -0.13791741
```

```
test$diagnosis..M.malignant..B.benign. <- ifelse(test$diagnosis..M.malignant..B.benign. %in% c("B", "B"), 0, 1) #encoding th
e categorical/ response variable in testing data
tail(test)
```

```
##     diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 664                                 1     0.6697930     1.26572769
## 672                                 1     0.3920006    -0.08119503
## 694                                 1     1.0347359     0.08745519
## 696                                 1    -0.5394208     0.56149906
## 702                                 1     0.4873215     1.87195687
## 705                                 1    -0.1526903     0.52047603
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 664       0.62040017   0.5598993       -1.0028577       -0.007351707
## 672       0.51753743   0.2557313        0.4091486        1.457777099
## 694       1.00415729   1.0045768       -0.1910822       -0.087709537
## 696      -0.45332853  -0.5857104        0.8629816        1.282111145
## 702       0.45819355   0.3695924       -0.9120911       -0.168067366
## 705      -0.01181002  -0.2021357        0.5775060        1.672687574
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 664        0.2362473             0.03772548      -1.3276336    0.61697378
## 672        1.5212016             0.91375441       1.2332452    0.02691660
## 694        0.5923490             0.54312679      -0.4665194    0.80050327
## 696        0.8475967             0.32272622       1.1284139   -0.40526575
## 702       -0.1110141            -0.07222147      -0.8671247    0.34069282
## 705        1.3954454             0.77670072       0.4769623    0.08217279
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 664      1.4473893        0.5358031  0.49075780       0.425607191
## 672     -0.5421587        0.2305102 -0.08257776       0.163669472
## 694      0.1327806        0.7383806  0.68891909      -0.009475799
## 696      0.5961000       -0.2488281 -0.47173990       1.939518410
## 702      1.2325630        0.3617576  0.22432245      -0.892960646
## 705      0.4934964        0.1449141 -0.04960639       1.331290149
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 664          0.8623676        0.9550991             0.7208735      -0.7014028
## 672          1.0746384        1.3658151             0.7754706       0.6070695
## 694         -0.3894057        0.1966341             0.3432432      -0.9557391
## 696          1.6246695        1.5494686             1.0818213       0.7494338
## 702          0.2205606        0.1761222             0.2340489      -1.1524899
## 705          2.1890600        1.8547626             0.7906365      -0.1511406
##     fractal.dimension..nucC.
## 664               -0.3086031
## 672                0.3456005
## 694               -0.5956404
## 696                2.1534414
## 702               -0.3766578
## 705                1.5442418
```

Fitting Model

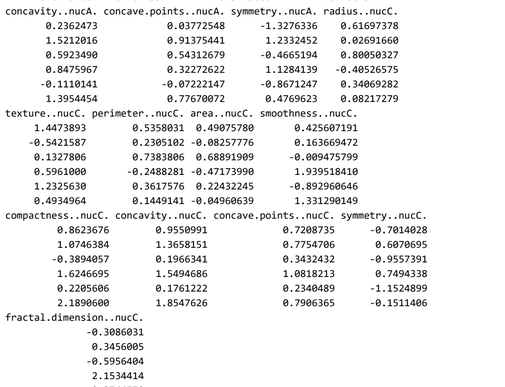
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, hidden = 5, err.fct = "ce", linear.output = FALSE, lifesign =
'full', rep = 1, algorithm = "rprop+", stepmax = 100000) -> nnmodel #fitting the model
```

```
## hidden: 5    thresh: 0.01    rep: 1/1    steps:    1000  min thresh: 0.0349646574984445
##                                          2000  min thresh: 0.0146846264066943
##                                          2311  error: 2.45059  time: 1.07 secs
```

```
summary(nnmodel) #model summary
```

```
##                     Length Class      Mode
## call                   10  -none-     call
## response              566  -none-     numeric
## covariate           10754  -none-     numeric
## model.list              2  -none-     list
## err.fct                 1  -none-     function
## act.fct                 1  -none-     function
## linear.output           1  -none-     logical
## data                   20  data.frame list
## exclude                 0  -none-     NULL
## net.result              1  -none-     list
## weights                 1  -none-     list
## generalized.weights     1  -none-     list
## startweights            1  -none-     list
## result.matrix         109  -none-     numeric
```

```
plot(nnmodel, rep = 1) #network architecture
```



Results

```
nnresults <- compute(nnmodel, test_data)
results <- data.frame(actual = test$diagnosis..M.malignant..B.benign., prediction = nnresults$net.result)
head(results)
```

```
##    actual   prediction
## 4       0 7.390170e-01
## 7       0 1.814224e-10
## 9       0 1.495226e-11
## 17      0 1.478257e-11
## 18      0 1.763344e-08
## 30      0 1.680649e-11
```

Prediction

```
predict(nnmodel, test_data, type = "class") -> nnresult  #using caret to make model predictions
#table(test_data$diagnosis..M.malignant..B.benign., nnresult)
```

Confusion Matrix

```
#confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., nnresult))
roundedresults <- sapply(results,round,digits = 0)
roundedresultsdata = data.frame(roundedresults)
attach(roundedresultsdata)
table(actual, prediction)
```

```
##       prediction
## actual  0  1
##      0 68  3
##      1  2 68
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 97.87
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 68  3
##      1  2 68
##
##               Accuracy : 0.9645
##                 95% CI : (0.9192, 0.9884)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9291
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9714
##            Specificity : 0.9577
##         Pos Pred Value : 0.9577
##         Neg Pred Value : 0.9714
##             Prevalence : 0.4965
##         Detection Rate : 0.4823
##   Detection Prevalence : 0.5035
##      Balanced Accuracy : 0.9646
##
##       'Positive' Class : 0
##
```

Neural Network: Model 2

Fitting Model

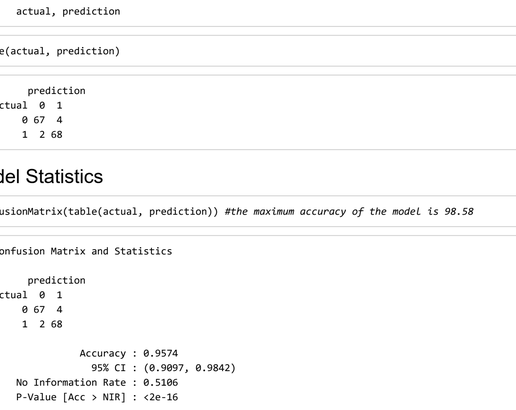
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, threshold = 0.03, hidden = 32, err.fct = "ce", linear.output =
FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> nnmodel
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.390086028175195
##                                           2000 min thresh: 0.197224766300759
##                                           3000 min thresh: 0.119454718020134
##                                           4000 min thresh: 0.0832128466357515
##                                           5000 min thresh: 0.0629035307148206
##                                           6000 min thresh: 0.0500721356022195
##                                           7000 min thresh: 0.0413087882817903
##                                           8000 min thresh: 0.0349845123289776
##                                           9000 min thresh: 0.0302284795904298
##                                           9056 error: 0.28481  time: 22.45 secs
```

```
summary(nnmodel) #model summary
```

```
##                   Length Class      Mode
## call                  13 -none-     call
## response             566 -none-     numeric
## covariate          10754 -none-     numeric
## model.list             2 -none-     list
## err.fct                1 -none-     function
## act.fct                1 -none-     function
## linear.output          1 -none-     logical
## data                  20 data.frame list
## exclude                0 -none-     NULL
## net.result             1 -none-     list
## weights                1 -none-     list
## generalized.weights    1 -none-     list
## startweights           1 -none-     list
## result.matrix        676 -none-     numeric
```

```
plot(nnmodel, rep = 1) #network architecture
```



Results

```
nnresults <- compute(nnmodel, test_data)
results <- data.frame(actual = test$diagnosis..M.malignant..B.benign., prediction = nnresults$net.result)
```

```
head(results)
```

```
##    actual   prediction
## 4       0 5.339353e-01
## 7       0 1.429233e-02
## 9       0 1.942662e-06
## 17      0 6.663307e-07
## 18      0 4.054958e-04
## 30      0 5.766854e-04
```

Prediction

```
predict(nnmodel, test_data, type = "class") -> nnresult  #using caret to make model predictions
#table(test_data$diagnosis..M.malignant..B.benign., nnresult)
```

Confusion Matrix

```
#confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., nnresult))
roundedresults <- sapply(results,round,digits = 0)
roundedresultsdata = data.frame(roundedresults)
attach(roundedresultsdata)
```

```
## The following objects are masked from roundedresultsdata (pos = 3):
##
##     actual, prediction
```

```
table(actual, prediction)
```

```
##       prediction
## actual  0  1
##      0 67  4
##      1  2 68
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 98.58
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 67  4
##      1  2 68
##
##               Accuracy : 0.9574
##                 95% CI : (0.9097, 0.9842)
##    No Information Rate : 0.5106
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9149
##
##  Mcnemar's Test P-Value : 0.6831
##
##            Sensitivity : 0.9710
##            Specificity : 0.9444
##         Pos Pred Value : 0.9437
##         Neg Pred Value : 0.9714
##             Prevalence : 0.4894
##         Detection Rate : 0.4752
##   Detection Prevalence : 0.5035
##      Balanced Accuracy : 0.9577
##
##       'Positive' Class : 0
##
```

Hybrid Models

Decision Tree and Random Forest

```
confusionMatrix(table(round((ifelse(dtresult %in% c("B", "B"), 0, 1) +
                            ifelse(rfresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 96.45
```

```
## Confusion Matrix and Statistics
##
##
##     0  1
##   0 70  4
##   1  1 66
##
##               Accuracy : 0.9645
##                 95% CI : (0.9192, 0.9884)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9291
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9859
##             Specificity : 0.9429
##          Pos Pred Value : 0.9459
##          Neg Pred Value : 0.9851
##              Prevalence : 0.5035
##          Detection Rate : 0.4965
##    Detection Prevalence : 0.5248
##       Balanced Accuracy : 0.9644
##
##        'Positive' Class : 0
##
```

Decision Tree and SVM

```
confusionMatrix(table(round((ifelse(dtresult %in% c("B", "B"), 0, 1) +
                            ifelse(svmresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the max
imum accuracy of the model is 96.45
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  4
##   1  2 66
##
##               Accuracy : 0.9574
##                 95% CI : (0.9097, 0.9842)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9149
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9718
##             Specificity : 0.9429
##          Pos Pred Value : 0.9452
##          Neg Pred Value : 0.9706
##              Prevalence : 0.5035
##          Detection Rate : 0.4894
##    Detection Prevalence : 0.5177
##       Balanced Accuracy : 0.9573
##
##        'Positive' Class : 0
##
```

Random Forest and SVM

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1) +
                               ifelse(svmresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the max
imum accuracy of the model is 99.29
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 70  2
##   1  1 68
##
##               Accuracy : 0.9787
##                 95% CI : (0.9391, 0.9956)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9574
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9859
##             Specificity : 0.9714
##          Pos Pred Value : 0.9722
##          Neg Pred Value : 0.9855
##              Prevalence : 0.5035
##          Detection Rate : 0.4965
##    Detection Prevalence : 0.5106
##       Balanced Accuracy : 0.9787
##
##        'Positive' Class : 0
##
```

Random Forest and Naive Bayes

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1) +
                               ifelse(nbresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 97.87
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 70  5
##   1  1 65
##
##               Accuracy : 0.9574
##                 95% CI : (0.9097, 0.9842)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9149
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.9859
##             Specificity : 0.9286
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.9848
##              Prevalence : 0.5035
##          Detection Rate : 0.4965
##    Detection Prevalence : 0.5319
##       Balanced Accuracy : 0.9572
##
##        'Positive' Class : 0
##
```

Random Forest and Neural Network

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1)*0.90 +
                               (ifelse(nnresult %in% c("B", "B"), 0, 1)*0.90))/2), test$diagnosis..M.malignant..B.benign.)) #th
e maximum accuracy of the model is 98.58
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 70  2
##   1  1 68
##
##               Accuracy : 0.9787
##                 95% CI : (0.9391, 0.9956)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9574
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9859
##             Specificity : 0.9714
##          Pos Pred Value : 0.9722
##          Neg Pred Value : 0.9855
##              Prevalence : 0.5035
##          Detection Rate : 0.4965
##    Detection Prevalence : 0.5106
##       Balanced Accuracy : 0.9787
##
##        'Positive' Class : 0
##
```

SVM and Naive Bayes

```
confusionMatrix(table(round((ifelse(dtresult %in% c("B", "B"), 0, 1) +
                               ifelse(nbresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 92.91
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 67  6
##   1  4 64
##
##               Accuracy : 0.9291
##                 95% CI : (0.8734, 0.9655)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8581
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9437
##             Specificity : 0.9143
##          Pos Pred Value : 0.9178
##          Neg Pred Value : 0.9412
##              Prevalence : 0.5035
##          Detection Rate : 0.4752
##    Detection Prevalence : 0.5177
##       Balanced Accuracy : 0.9290
##
##        'Positive' Class : 0
##
```

SVM and Neural Network

```
confusionMatrix(table(round((ifelse(svmresult %in% c("B", "B"), 0, 1) +
                               ifelse(nnresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 99.29
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  2
##   1  2 68
##
##               Accuracy : 0.9716
##                 95% CI : (0.929, 0.9922)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9433
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9714
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9714
##              Prevalence : 0.5035
##          Detection Rate : 0.4894
##    Detection Prevalence : 0.5035
##       Balanced Accuracy : 0.9716
##
##        'Positive' Class : 0
##
```

Naive Bayes and Neural Network

```
confusionMatrix(table(round((ifelse(nbresult %in% c("B", "B"), 0, 1) +
                               ifelse(nnresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 95.58
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 65  5
##   1  6 65
##
##               Accuracy : 0.922
##                 95% CI : (0.8647, 0.9604)
##    No Information Rate : 0.5035
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.844
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9155
##             Specificity : 0.9286
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.9155
##              Prevalence : 0.5035
##          Detection Rate : 0.4610
##    Detection Prevalence : 0.4965
##       Balanced Accuracy : 0.9220
##
##        'Positive' Class : 0
##
```

Random Forest, SVM and Neural Network

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1)*0.90 +
                              ifelse(svmresult %in% c("B", "B"), 0, 1)*0.85 +
                             (ifelse(nnresult %in% c("B", "B"), 0, 1)*0.90))/3), test$diagnosis..M.malignant..B.benign.)) #th
e maximum accuracy of the model is 98.58
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  2
##   1  2 68
##
##                Accuracy : 0.9716
##                  95% CI : (0.929, 0.9922)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9433
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9714
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9714
##              Prevalence : 0.5035
##          Detection Rate : 0.4894
##    Detection Prevalence : 0.5035
##       Balanced Accuracy : 0.9716
##
##       'Positive' Class : 0
##
```

Ensemble Model: Random Forest, SVM -> Neural Network

Creating Sample Datasets

```
rftrain <- train #creating dummy training data for random forest algorithm
rftest <- test #creating dummy training data for random forest algorithm

svmtrain <- train #creating dummy training data for svm algorithm
svmtest <- test #creating dummy testing data for svm algorithm

ensembletrain <- train #creating dummy training data for stacked ensemble model
ensembletest <- test #creating dummy testing data for stacked ensemble model
```

Prediction for training data using Random Forest and SVM

```
rftrain$diagnosis..M.malignant..B.benign. <- ifelse(predict(rfmodel, train_data, type = "class") %in% c("B", "B"), 0, 1) #en
coding the categorical/ response variable in training data for random forest
svmtrain$diagnosis..M.malignant..B.benign. <- ifelse(predict(svmmodel, train_data, type = "class") %in% c("B", "B"), 0, 1) #
encoding the categorical/ response variable in training data for svm

ensembletrain$diagnosis..M.malignant..B.benign. <-round((rftrain$diagnosis..M.malignant..B.benign. + svmtrain$diagnosis..M.m
alignant..B.benign.)/2) #encoding the categorical/ response variable in training data for stacked ensemble model
```

Predction for testing data using Random Forest and SVM

```
rftest$diagnosis..M.malignant..B.benign. <- ifelse(rfresult %in% c("B", "B"), 0, 1) #encoding the categorical/ response vari
able in testing data for random forest
svmtest$diagnosis..M.malignant..B.benign. <- ifelse(svmresult %in% c("B", "B"), 0, 1) #encoding the categorical/ response va
riable in testing data for svm

ensembletest$diagnosis..M.malignant..B.benign. <- round((rftest$diagnosis..M.malignant..B.benign. + svmtest$diagnosis..M.mal
ignant..B.benign.)/2) #encoding the categorical/ response variable in testing data for stacked ensemble model
```

Training the Neural Network

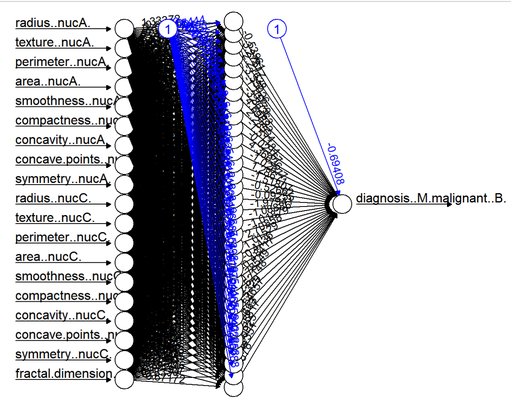
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = ensembletrain, threshold = 0.03, hidden = 32, err.fct = "ce", linear.
output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> ensemblemodel #fitting th
e model
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.215147225542229
##                                                     2000 min thresh: 0.108765685107557
##                                                     3000 min thresh: 0.0716017355797312
##                                                     4000 min thresh: 0.0529337640653415
##                                                     5000 min thresh: 0.0417763165870059
##                                                     6000 min thresh: 0.0343897010101897
##                                                     6817 error: 0.19817  time: 18.43 secs
```

```
summary(ensemblemodel) #model summary
```

```
##                     Length Class      Mode
## call                    13 -none-     call
## response               566 -none-     numeric
## covariate            10754 -none-     numeric
## model.list               2 -none-     list
## err.fct                  1 -none-     function
## act.fct                  1 -none-     function
## linear.output            1 -none-     logical
## data                    20 data.frame list
## exclude                  0 -none-     NULL
## net.result               1 -none-     list
## weights                  1 -none-     list
## generalized.weights      1 -none-     list
## startweights             1 -none-     list
## result.matrix          676 -none-     numeric
```

```
plot(ensemblemodel, rep = 1) # network architecture
```



Model Results

```
ensembleresults <- compute(ensemblemodel, ensembletest)
ensembleresults <- data.frame(actual = ensembletest$diagnosis..M.malignant..B.benign.,
                              prediction = ensembleresults$net.result)
head(ensembleresults)
```

```
##    actual   prediction
## 4       0 8.097231e-09
## 7       0 8.026651e-04
## 9       0 4.743610e-08
## 17      0 2.788038e-08
## 18      0 1.711124e-08
## 30      0 2.206554e-06
```

Prediction

```
predict(ensemblemodel, ensembletest, type = "class") -> ensembleresult #using caret to make model predictions
```

```
#confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., nnresult))
roundedresults <- sapply(ensembleresults,round,digits = 0)
roundedresultsdata = data.frame(roundedresults)
attach(roundedresultsdata)
```

```
## The following objects are masked from roundedresultsdata (pos = 3):
##
##     actual, prediction
```

```
## The following objects are masked from roundedresultsdata (pos = 4):
##
##     actual, prediction
```

```
#table(actual, prediction)
```

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 100.00
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 72  0
##      1  0 69
##
##                Accuracy : 1
##                  95% CI : (0.9742, 1)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5106
##          Detection Rate : 0.5106
##    Detection Prevalence : 0.5106
##       Balanced Accuracy : 1.0000
##
##       'Positive' Class : 0
##
```