R Final Project : Breast Cancer Classification :: Notebook 3.1

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

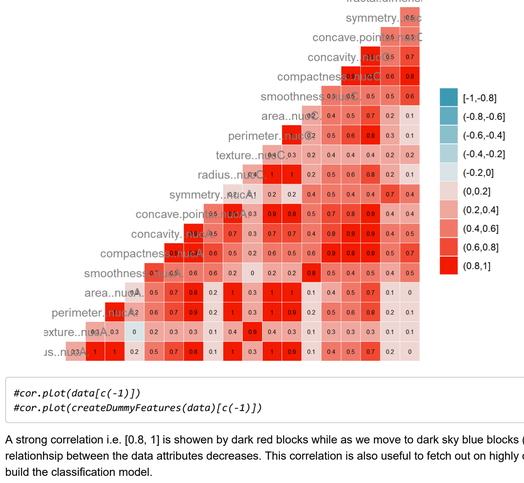
```
data = data[, c("diagnosis..M.malignant..B.benign.", "radius..nucA.", "texture..nucA.", "perimeter..nucA.", "area..nucA.",
"smoothness..nucA.", "compactness..nucA.", "concavity..nucA.", "concave.points..nucA.", "symmetry..nucA.", "radius..nucC.",
"texture..nucC.", "perimeter..nucC.", "area..nucC.", "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
```



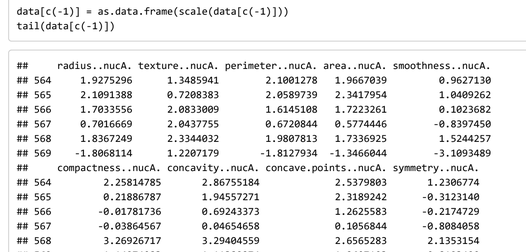

```
#cor.plot(data[c(-1)])
#cor.plot(createDummyFeatures(data)[c(-1)])
```

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.
```

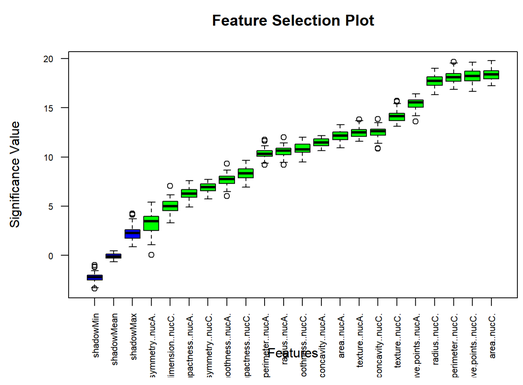
```
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)
```

```
##  [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
```

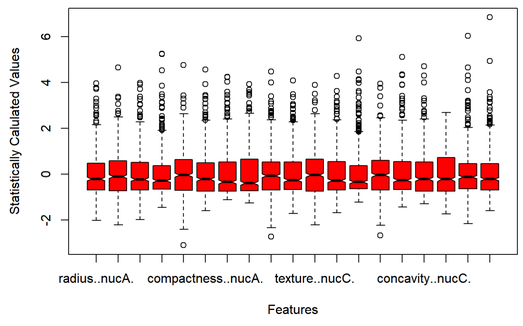
```
##                       meanImp  decision
## area..nucC.          18.37436 Confirmed
## concave.points..nucC. 18.24485 Confirmed
## perimeter..nucC.      18.05579 Confirmed
## radius..nucC.         17.66742 Confirmed
## concave.points..nucA. 15.43812 Confirmed
## texture..nucC.        14.15516 Confirmed
```

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



Removing Outliers

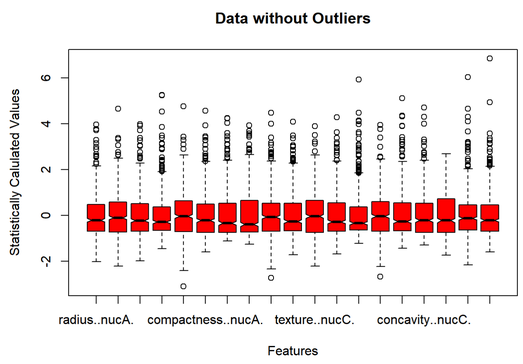
```
boxplot(data[c(-1)], col = "red", main = "Data with Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Calula
ted Values")  #using boxplot to represent data with outliers
```




```
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
```

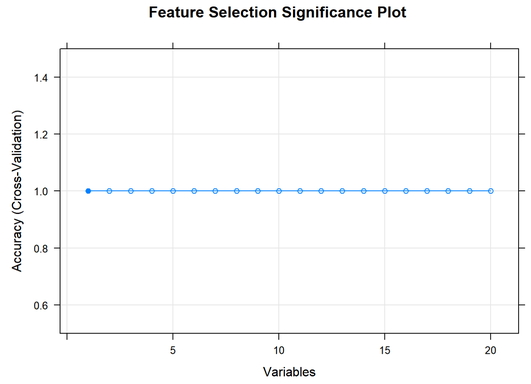
```
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(data[, c(1:dim(data)[2])], data[, c(1)], sizes=c(1:dim(data)[2]), rfeControl=control)
# summarize the results
print(results)
```

```
##
## 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")
```

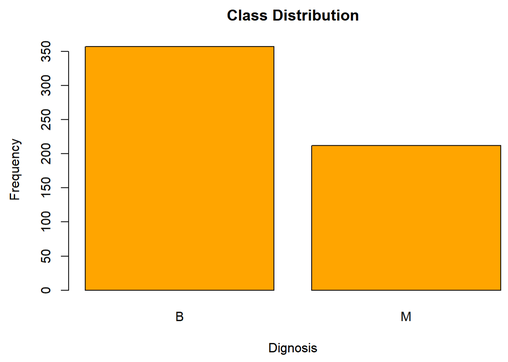


Resampling

```
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
```

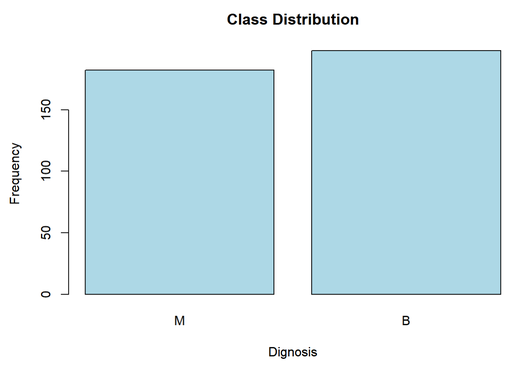
```
## Warning: package 'ROSE' was built under R version 3.6.3
```

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

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

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

```
##
##         M         B
## 0.4789474 0.5210526
```

```
barplot(table(ovun.sample(diagnosis..M.malignant..B.benign. ~ ., data = data, method = "under", seed = 1)$data$diagnosis..M.
malignant..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= 328
##
##           CP nsplit rel error    xerror       xstd
## 1 0.81645570      0 1.0000000 1.0000000 0.05727420
## 2 0.04430380      1 0.1835443 0.2405063 0.03668570
## 3 0.02531646      2 0.1392405 0.2278481 0.03583015
## 4 0.01000000      3 0.1139241 0.2151899 0.03493970
##
## Variable importance
## concave.points..nucA.      perimeter..nucC. concave.points..nucC.
##                    18                    15                    15
##      concavity..nucA.      concavity..nucC.    compactness..nucA.
##                    15                    13                    13
##          area..nucC.         radius..nucC.      perimeter..nucA.
##                     2                     2                     1
##          area..nucA.         radius..nucA.        texture..nucA.
##                     1                     1                     1
##       texture..nucC.
##                     1
##
## Node number 1: 328 observations,    complexity param=0.8164557
##   predicted class=M  expected loss=0.4817073  P(node) =1
##     class counts:   158   170
##    probabilities: 0.482 0.518
##   left son=2 (163 obs) right son=3 (165 obs)
##   Primary splits:
##       concave.points..nucA. < -0.1444417  to the left,  improve=111.0720, (0 missing)
##       perimeter..nucC.      < -0.3292055  to the left,  improve=110.9657, (0 missing)
##       concave.points..nucC. < -0.2544719  to the left,  improve=109.5763, (0 missing)
##       area..nucC.           < -0.4258789  to the left,  improve=106.3156, (0 missing)
##       radius..nucC.         < -0.33512    to the left,  improve=104.4683, (0 missing)
##   Surrogate splits:
##       concave.points..nucC. < 0.005787681 to the left,  agree=0.936, adj=0.871, (0 split)
##       concavity..nucA.      < -0.4175264  to the left,  agree=0.921, adj=0.840, (0 split)
##       perimeter..nucC.      < -0.3431236  to the left,  agree=0.887, adj=0.773, (0 split)
##       concavity..nucC.      < -0.4153455  to the left,  agree=0.875, adj=0.748, (0 split)
##       compactness..nucA.    < -0.2057054  to the left,  agree=0.872, adj=0.742, (0 split)
##
## Node number 2: 163 observations,    complexity param=0.0443038
##   predicted class=B  expected loss=0.1042945  P(node) =0.4969512
##     class counts:   146    17
##    probabilities: 0.896 0.104
##   left son=4 (150 obs) right son=5 (13 obs)
##   Primary splits:
##       radius..nucC.    < -0.1048036  to the left,  improve=12.49194, (0 missing)
##       area..nucC.      < -0.1773159  to the left,  improve=12.02337, (0 missing)
##       perimeter..nucC. < -0.2178603  to the left,  improve=11.20364, (0 missing)
##       radius..nucA.    < 0.03042683  to the left,  improve=10.31281, (0 missing)
##       area..nucA.      < -0.08559375 to the left,  improve=10.31281, (0 missing)
##   Surrogate splits:
##       area..nucC.      < -0.2041114  to the left,  agree=0.994, adj=0.923, (0 split)
##       perimeter..nucC. < -0.06476078 to the left,  agree=0.982, adj=0.769, (0 split)
##       radius..nucA.    < 0.04106354  to the left,  agree=0.975, adj=0.692, (0 split)
##       area..nucA.      < -0.07031348 to the left,  agree=0.975, adj=0.692, (0 split)
##       perimeter..nucA. < -0.0235404  to the left,  agree=0.969, adj=0.615, (0 split)
##
## Node number 3: 165 observations,    complexity param=0.02531646
##   predicted class=M  expected loss=0.07272727  P(node) =0.5030488
##     class counts:    12   153
##    probabilities: 0.073 0.927
##   left son=6 (8 obs) right son=7 (157 obs)
##   Primary splits:
##       texture..nucA.        < -1.360273   to the left,  improve=7.713144, (0 missing)
##       perimeter..nucC.      < -0.4867588  to the left,  improve=6.716084, (0 missing)
##       concave.points..nucC. < 0.3199685   to the left,  improve=6.336027, (0 missing)
##       area..nucC.           < -0.4725499  to the left,  improve=6.121954, (0 missing)
##       radius..nucC.         < -0.4870514  to the left,  improve=5.128749, (0 missing)
##   Surrogate splits:
##       texture..nucC.   < -1.351124   to the left,  agree=0.976, adj=0.500, (0 split)
##       perimeter..nucC. < -0.80451    to the left,  agree=0.964, adj=0.250, (0 split)
##       area..nucC.      < -0.8235476  to the left,  agree=0.964, adj=0.250, (0 split)
##       perimeter..nucA. < -0.9224359  to the left,  agree=0.958, adj=0.125, (0 split)
##       radius..nucC.    < -0.8383322  to the left,  agree=0.958, adj=0.125, (0 split)
##
## Node number 4: 150 observations
##   predicted class=B  expected loss=0.04666667  P(node) =0.4573171
##     class counts:   143     7
##    probabilities: 0.953 0.047
##
## Node number 5: 13 observations
##   predicted class=M  expected loss=0.2307692  P(node) =0.03963415
##     class counts:     3    10
##    probabilities: 0.231 0.769
##
## Node number 6: 8 observations
##   predicted class=B  expected loss=0.25  P(node) =0.02439024
##     class counts:     6     2
##    probabilities: 0.750 0.250
##
## Node number 7: 157 observations
##   predicted class=M  expected loss=0.03821656  P(node) =0.4786585
##     class counts:     6   151
##    probabilities: 0.038 0.962
```

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 98.78, 1.00, 97.67
```

```
## Confusion Matrix and Statistics
##
##    dtresult
##      B  M
##   B 39  1
##   M  0 42
##
##               Accuracy : 0.9878
##                 95% CI : (0.9339, 0.9997)
##    No Information Rate : 0.5244
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 1.0000
##            Specificity : 0.9767
##         Pos Pred Value : 0.9750
##         Neg Pred Value : 1.0000
##             Prevalence : 0.4756
##         Detection Rate : 0.4756
##   Detection Prevalence : 0.4878
##      Balanced Accuracy : 0.9884
##
##       '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
```

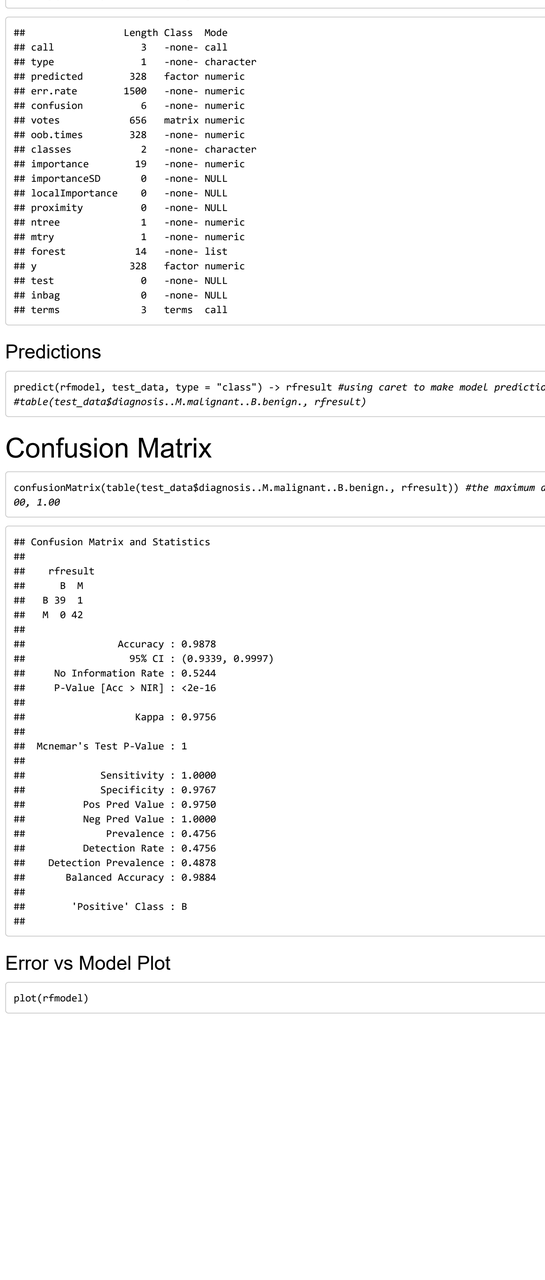
```
## Loading required package: sandwich
```

```
## 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) #using randomForest function to build a random forest classification model
```

```
## 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 #fitting the model
summary(rfmodel) #model summary
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted        328   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes            656   matrix numeric
## oob.times        328   -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                328   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)
```

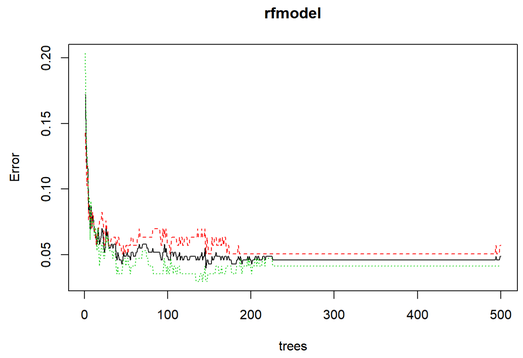
Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., rfresult)) #the maximum accuracy of the model is 1.00, 1.00, 1.00
```

```
## Confusion Matrix and Statistics
##
##    rfresult
##      B  M
##   B 39  1
##   M  0 42
##
##               Accuracy : 0.9878
##                 95% CI : (0.9339, 0.9997)
##    No Information Rate : 0.5244
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 1.0000
##            Specificity : 0.9767
##         Pos Pred Value : 0.9750
##         Neg Pred Value : 1.0000
##             Prevalence : 0.4756
##         Detection Rate : 0.4756
##   Detection Prevalence : 0.4878
##      Balanced Accuracy : 0.9884
##
##       '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 #fitting 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:  39
##
##  ( 18 21 )
##
##
## 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 98.08, 1.00, 96.43
```

```
## Confusion Matrix and Statistics
##
##    svmresult
##      B  M
##   B 38  2
##   M  1 41
##
##               Accuracy : 0.9634
##                 95% CI : (0.8968, 0.9924)
##    No Information Rate : 0.5244
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9267
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9744
##            Specificity : 0.9535
##         Pos Pred Value : 0.9500
##         Neg Pred Value : 0.9762
##             Prevalence : 0.4756
##         Detection Rate : 0.4634
##   Detection Prevalence : 0.4878
##      Balanced Accuracy : 0.9639
##
##       '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 97.56
```

```
## Confusion Matrix and Statistics
##
##    nbresult
##      B  M
##   B 38  2
##   M  0 42
##
##               Accuracy : 0.9756
##                 95% CI : (0.9147, 0.997)
##    No Information Rate : 0.5366
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9511
##
##  Mcnemar's Test P-Value : 0.4795
##
##            Sensitivity : 1.0000
##            Specificity : 0.9545
##         Pos Pred Value : 0.9500
##         Neg Pred Value : 1.0000
##             Prevalence : 0.4634
##         Detection Rate : 0.4634
##   Detection Prevalence : 0.4878
##      Balanced Accuracy : 0.9773
##
##       '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 99.12
```

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.
## 404                                 1   -0.15837477      0.5976541
## 405                                 1    0.09424708      2.4719687
## 407                                 1    1.78016556      0.5604930
## 408                                 1    1.39990319      1.9215194
## 409                                 1    0.46121357      1.8820357
## 410                                 1    1.52488453      2.1723571
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 404       -0.1306986 -0.25459099       0.422911726         0.02904345
## 405        0.2405005 -0.02708466       0.495055439         1.68829105
## 407        1.7283808  1.96353037       0.942346461         0.03793545
## 408        1.3120827  1.39411156      -0.009950554        -0.18436460
## 409        0.4293765  0.34173156      -0.965854754        -0.20392701
## 410        1.6551432  1.40455961       1.432923710         2.90293854
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 404        0.5048552            0.09511503       0.3598126   -0.40286010
## 405        1.8013301            0.88562811       1.0994973    0.02874116
## 407        1.6689394            1.95750618      -0.3725485    1.56353845
## 408        0.4774229            0.97260858      -0.2773416    1.22290248
## 409       -0.1367030           -0.10599726      -0.8705541    0.31131418
## 410        2.9511017            2.27226975       2.0845231    1.61966597
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 404     1.22982265       -0.2971938  -0.4434207         0.1004570
## 405     2.61587138        0.3820116  -0.1429564         0.2610045
## 407    -0.05048293        1.4230886   1.6466332         0.2306306
## 408     1.87729612        1.1141059   1.1702676        -0.8281149
## 409     1.20542038        0.3263390   0.1933963        -0.9452711
## 410     2.06763380        1.9380599   1.3151085         1.2720197
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 404        -0.04506497        0.5354318             0.1078785      -0.4279659
## 405         3.01495447        4.0142373             1.5371494       1.6670474
## 407        -0.41103952        0.4762557             1.3358436      -1.3888707
## 408        -0.52378329        0.0606256             0.4903594      -0.6177371
## 409         0.16802666        0.1482248             0.1884008      -1.1509037
## 410         3.46592956        2.9364879             1.9598915       1.6640352
##     fractal.dimension..nucC.
## 404              -0.1139548
## 405               2.8461100
## 407              -0.7514596
## 408              -0.9980027
## 409              -0.3878343
## 410               1.9744408
```

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

```
##     diagnosis..M.malignant..B.benign. radius..nucA. texture..nucA.
## 391                                 1     0.8361576      0.1354625
## 392                                 1     0.9637981     -0.5705990
## 399                                 1     0.9159329      0.1401077
## 400                                 1     1.3360829      0.0657854
## 403                                 1     1.5115886      0.2051396
## 406                                 1     1.6099782      1.1875871
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 391        0.8186924   0.7306603         0.1559800          0.2691275
## 392        0.9304761   0.8899931         0.2353381          0.3189227
## 399        0.9112030   0.8482009         0.6393429          0.1962131
## 400        1.2851004   1.2713470         0.4156974          0.3064739
## 403        1.5664871   1.5168762         0.4806267          1.0694077
## 406        1.7669269   1.6187447         0.8629884          1.9532727
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 391        0.1553915             0.4850855      1.05555568     0.8048492
## 392        0.5382511             0.8265198      0.01560286     1.0409719
## 399        0.6312824             0.8909142      0.11813342     0.8687185
## 400        0.4428344             0.8861087     -0.09425131     1.2306442
## 403        1.2467203             1.7965202      1.09583554     1.3409638
## 406        2.5455806             2.1617421      1.17639527     1.3390284
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 391    -0.10579474        0.7912050   0.6794180        -0.2076206
## 392    -0.62312279        0.8747138   0.9594437         0.2393089
## 399    -0.08301929        0.7578014   0.7872439         0.5864385
## 400    -0.24082061        1.2672054   1.0334734         0.2610045
## 403    -0.20015017        1.2588545   1.2957964        -0.3855245
## 406     0.43918921        1.7849603   1.3118898         0.2176133
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 391         -0.2847193       -0.2520289             0.2904916      0.40491861
## 392          0.1650753        0.2087987             0.5406859     -0.71111654
## 399         -0.1613504        0.3141041             0.3206874      0.13833531
## 400          0.2949372        0.5126001             0.4687909     -0.65237784
## 403          0.1922282        0.6281564             1.2380665      0.14285367
## 406          0.8126141        1.6374107             1.8045985     -0.08005213
##     fractal.dimension..nucC.
## 391              -0.4884115
## 392               0.4503092
## 399              -0.3656557
## 400               0.2909330
## 403              -0.5172952
## 406               0.6710633
```

Fitting Model

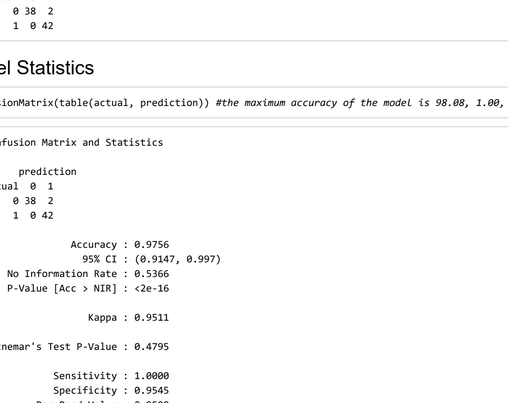
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, hidden = 5, err.fct = "ce", linear.output = FALSE, lifesign = 'full', rep = 1, algorithm = "rprop+", stepmax = 100000) -> nnmodel #fitting the model
```

```
## hidden: 5    thresh: 0.01    rep: 1/1    steps:     266  error: 0.02525  time: 0.14 secs
```

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

```
##                   Length Class      Mode
## call                10   -none-     call
## response           328   -none-     numeric
## covariate         6232   -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 1.472854e-07
## 10      0 2.267654e-05
## 27      0 5.049805e-07
## 30      0 1.555622e-07
## 34      0 1.662702e-07
## 36      0 2.711094e-07
```

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 38  2
##      1  0 42
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 98.08, 1.00, 96.43
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 38  2
##      1  0 42
##
##                Accuracy : 0.9756
##                  95% CI : (0.9147, 0.997)
##     No Information Rate : 0.5366
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9511
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9545
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4634
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 0.9773
##
##        'Positive' Class : 0
##
```

Neural Network: Model 2

Fitting Model

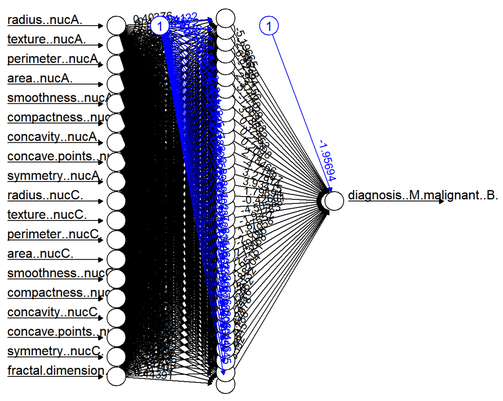
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, threshold = 0.03, hidden = 32, err.fct = "ce", linear.output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> nnmodel
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.394826871547688
##                                           2000 min thresh: 0.201454112716163
##                                           3000 min thresh: 0.128741182297861
##                                           4000 min thresh: 0.0910222181852623
##                                           5000 min thresh: 0.0691333630458698
##                                           6000 min thresh: 0.0551445011416677
##                                           7000 min thresh: 0.04556212292445
##                                           8000 min thresh: 0.0386459363871006
##                                           9000 min thresh: 0.0336081502750497
##                                           9916 error: 0.31591  time: 19.22 secs
```

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

```
##                   Length Class      Mode
## call                13   -none-     call
## response           328   -none-     numeric
## covariate         6232   -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 6.473206e-06
## 10      0 5.449688e-04
## 27      0 6.634890e-05
## 30      0 2.062124e-04
## 34      0 4.674457e-05
## 36      0 1.137803e-05
```

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 39  1
##      1  0 42
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 99.08, 1.00, 98.18
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 39  1
##      1  0 42
##
##               Accuracy : 0.9878
##                 95% CI : (0.9339, 0.9997)
##     No Information Rate : 0.5244
##     P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9767
##          Pos Pred Value : 0.9750
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4756
##          Detection Rate : 0.4756
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 0.9884
##
##        '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 maximum accuracy of the model is 94.69
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 40  0
##   1  0 42
##
##               Accuracy : 1
##                 95% CI : (0.956, 1)
##     No Information Rate : 0.5122
##     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 : 0.4878
##          Detection Rate : 0.4878
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 1.0000
##
##        '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 maximum accuracy of the model is 98.78
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 40  1
##   1  0 41
##
##               Accuracy : 0.9878
##                 95% CI : (0.9339, 0.9997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9762
##          Pos Pred Value : 0.9756
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4878
##          Detection Rate : 0.4878
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9881
##
##        '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 maximum accuracy of the model is 97.56
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 39  1
##   1  1 41
##
##               Accuracy : 0.9756
##                 95% CI : (0.9147, 0.997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9512
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9750
##             Specificity : 0.9762
##          Pos Pred Value : 0.9750
##          Neg Pred Value : 0.9762
##              Prevalence : 0.4878
##          Detection Rate : 0.4756
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 0.9756
##
##        '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 maximum accuracy of the model is 98.78
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 39  0
##   1  1 42
##
##               Accuracy : 0.9878
##                 95% CI : (0.9339, 0.9997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9750
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9767
##              Prevalence : 0.4878
##          Detection Rate : 0.4756
##    Detection Prevalence : 0.4756
##       Balanced Accuracy : 0.9875
##
##        '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.)) #the maximum accuracy of the model is 98.78
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 39  0
##   1  1 42
##
##                Accuracy : 0.9878
##                  95% CI : (0.9339, 0.9997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9756
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9750
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9767
##              Prevalence : 0.4878
##          Detection Rate : 0.4756
##    Detection Prevalence : 0.4756
##       Balanced Accuracy : 0.9875
##
##        '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 100.00
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 40  0
##   1  0 42
##
##                Accuracy : 1
##                  95% CI : (0.956, 1)
##     No Information Rate : 0.5122
##     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 : 0.4878
##          Detection Rate : 0.4878
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 1.0000
##
##        '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 98.23
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 38  1
##   1  2 41
##
##                Accuracy : 0.9634
##                  95% CI : (0.8968, 0.9924)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9267
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9500
##             Specificity : 0.9762
##          Pos Pred Value : 0.9744
##          Neg Pred Value : 0.9535
##              Prevalence : 0.4878
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4756
##       Balanced Accuracy : 0.9631
##
##        '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 97.56
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 38  0
##   1  2 42
##
##                Accuracy : 0.9756
##                  95% CI : (0.9147, 0.997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9511
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9500
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9545
##              Prevalence : 0.4878
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4634
##       Balanced Accuracy : 0.9750
##
##        '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 97.56
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 38  0
##   1  2 42
##
##                Accuracy : 0.9756
##                  95% CI : (0.9147, 0.997)
##     No Information Rate : 0.5122
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9511
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9500
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9545
##              Prevalence : 0.4878
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4634
##       Balanced Accuracy : 0.9750
##
##        '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

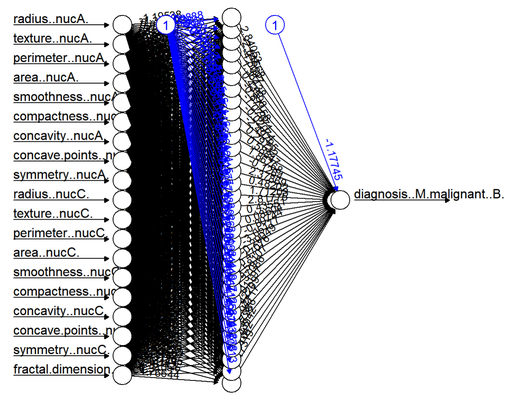
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = ensembletrain, threshold = 0.03, hidden = 32, err.fct = "ce", linear.
output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> ensemblemodel #fitting th
e model
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.189593100600767
##                                                     2000 min thresh: 0.0914126447955287
##                                                     3000 min thresh: 0.0591028924371781
##                                                     4000 min thresh: 0.0433835314196654
##                                                     5000 min thresh: 0.0341377453053527
##                                                     5640 error: 0.14907  time: 9.08 secs
```

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

```
##                    Length Class      Mode
## call                 13   -none-     call
## response            328   -none-     numeric
## covariate          6232   -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 1.044813e-07
## 10      0 6.802972e-05
## 27      0 5.453532e-07
## 30      0 2.135006e-07
## 34      0 1.038160e-05
## 36      0 3.377923e-07
```

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 98.23
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 40  0
##      1  0 42
##
##               Accuracy : 1
##                 95% CI : (0.956, 1)
##    No Information Rate : 0.5122
##    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 : 0.4878
##         Detection Rate : 0.4878
##   Detection Prevalence : 0.4878
##      Balanced Accuracy : 1.0000
##
##       'Positive' Class : 0
##
```

Ensemble Model: Random Forest, Decision Tree -> 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

dttrain <- train #creating dummy training data for decision tree algorithm
dttest <- test #creating dummy testing data for decision tree 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

```
rftest$diagnosis..M.malignant..B.benign. <- ifelse(rfresult %in% c("B", "B"), 0, 1) #encoding the categorical/ response variable in testing data for random forest
dttest$diagnosis..M.malignant..B.benign. <- ifelse(dtresult %in% c("B", "B"), 0, 1) #encoding the categorical/ response variable in testing data for decision tree

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

Predction for testing data using Random Forest and SVM

```
rftest$diagnosis..M.malignant..B.benign. <- ifelse(rfresult %in% c("B", "B"), 0, 1)
dttest$diagnosis..M.malignant..B.benign. <- ifelse(dtresult %in% c("B", "B"), 0, 1)

ensembletest$diagnosis..M.malignant..B.benign. <- round((rftest$diagnosis..M.malignant..B.benign. + svmtest$diagnosis..M.malignant..B.benign.)/2)
```

Training the Neural Network

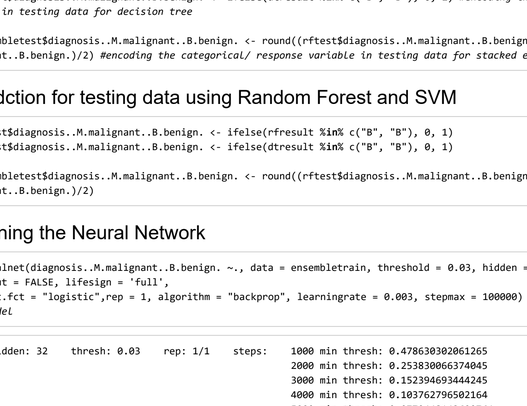
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = ensembletrain, threshold = 0.03, hidden = 32, err.fct = "ce", linear.output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> ensemblemodel #fitting the model
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.478630302061265
##                                             2000 min thresh: 0.253830066374045
##                                             3000 min thresh: 0.152394693444245
##                                             4000 min thresh: 0.103762796502164
##                                             5000 min thresh: 0.0779448142600764
##                                             6000 min thresh: 0.0616843631311151
##                                             7000 min thresh: 0.0505611234724957
##                                             8000 min thresh: 0.0425719798568568
##                                             9000 min thresh: 0.0366057179898938
##                                            10000 min thresh: 0.0320072319914558
##                                            10525 error: 0.32238  time: 13.37 secs
```

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

```
##                   Length Class      Mode
## call                13   -none-     call
## response           328   -none-     numeric
## covariate         6232   -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 4.607430e-06
## 10      0 5.865688e-03
## 27      0 7.421445e-05
## 30      0 7.062466e-06
## 34      0 2.691047e-04
## 36      0 2.511215e-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
```

```
## The following objects are masked from roundedresultsdata (pos = 5):
##
##     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 38  2
##      1  0 42
##
##                Accuracy : 0.9756
##                  95% CI : (0.9147, 0.997)
##     No Information Rate : 0.5366
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9511
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9545
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4634
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 0.9773
##
##        'Positive' Class : 0
##
```