R Final Project : Breast Cancer Classification :: Notebook 1.2

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
```

Statistical values about Data

```
summary(data) # summary of the data with IQR
```

```
##  diagnosis..M.malignant..B.benign. radius..nucA.    texture..nucA.
##  B:357                             Min.   : 6.981   Min.   : 9.71
##  M:212                             1st Qu.:11.700   1st Qu.:16.17
##                                    Median :13.370   Median :18.84
##                                    Mean   :14.127   Mean   :19.29
##                                    3rd Qu.:15.780   3rd Qu.:21.80
##                                    Max.   :28.110   Max.   :39.28
##  perimeter..nucA.  area..nucA.     smoothness..nucA. compactness..nucA.
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity..nucA.  concave.points..nucA. symmetry..nucA.
##  Min.   :0.00000   Min.   :0.00000       Min.   :0.1060
##  1st Qu.:0.02956   1st Qu.:0.02031       1st Qu.:0.1619
##  Median :0.06154   Median :0.03350       Median :0.1792
##  Mean   :0.08880   Mean   :0.04892       Mean   :0.1812
##  3rd Qu.:0.13070   3rd Qu.:0.07400       3rd Qu.:0.1957
##  Max.   :0.42680   Max.   :0.20120       Max.   :0.3040
##  fractal.dimension..nucA. radius..nucB.    texture..nucB.   perimeter..nucB.
##  Min.   :0.04996          Min.   :0.1115   Min.   :0.3602   Min.   : 0.757
##  1st Qu.:0.05770          1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606
##  Median :0.06154          Median :0.3242   Median :1.1080   Median : 2.287
##  Mean   :0.06280          Mean   :0.4052   Mean   :1.2169   Mean   : 2.866
##  3rd Qu.:0.06612          3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357
##  Max.   :0.09744          Max.   :2.8730   Max.   :4.8850   Max.   :21.980
##   area..nucB.      smoothness..nucB.  compactness..nucB. concavity..nucB.
##  Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000
##  1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509
##  Median : 24.530   Median :0.006380   Median :0.020450   Median :0.02589
##  Mean   : 40.337   Mean   :0.007041   Mean   :0.025478   Mean   :0.03189
##  3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205
##  Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600
##  concave.points..nucB. symmetry..nucB.    fractal.dimension..nucB.
##  Min.   :0.000000      Min.   :0.007882   Min.   :0.0008948
##  1st Qu.:0.007638      1st Qu.:0.015160   1st Qu.:0.0022480
##  Median :0.010930      Median :0.018730   Median :0.0031870
##  Mean   :0.011796      Mean   :0.020542   Mean   :0.0037949
##  3rd Qu.:0.014710      3rd Qu.:0.023480   3rd Qu.:0.0045580
##  Max.   :0.052790      Max.   :0.078950   Max.   :0.0298400
##  radius..nucC.   texture..nucC.  perimeter..nucC.  area..nucC.
##  Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
##  1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
##  Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
##  Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
##  3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness..nucC. compactness..nucC. concavity..nucC. concave.points..nucC.
##  Min.   :0.07117   Min.   :0.02729    Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14720    1st Qu.:0.1145   1st Qu.:0.06493
##  Median :0.13130   Median :0.21190    Median :0.2267   Median :0.09993
##  Mean   :0.13237   Mean   :0.25427    Mean   :0.2722   Mean   :0.11461
##  3rd Qu.:0.14600   3rd Qu.:0.33910    3rd Qu.:0.3829   3rd Qu.:0.16140
##  Max.   :0.22260   Max.   :1.05800    Max.   :1.2520   Max.   :0.29100
##  symmetry..nucC.  fractal.dimension..nucC.
##  Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2822   Median :0.08004
##  Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.6638   Max.   :0.20750
```

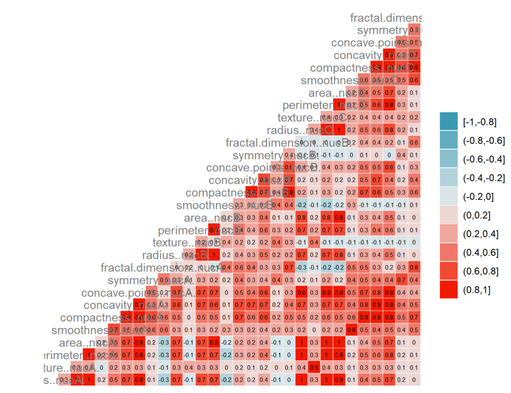
```
describe(data) # statitical estimations of the data
```

```
##                                    vars   n   mean     sd median trimmed    mad
## diagnosis..M.malignant..B.benign.*    1 569   1.37   0.48   1.00    1.34   0.00
## radius..nucA.                         2 569  14.13   3.52  13.37   13.82   2.82
## texture..nucA.                        3 569  19.29   4.30  18.84   19.04   4.17
## perimeter..nucA.                      4 569  91.97  24.30  86.24   89.74  18.84
## area..nucA.                           5 569 654.89 351.91 551.10  606.13 227.28
## smoothness..nucA.                     6 569   0.10   0.01   0.10    0.10   0.01
## compactness..nucA.                    7 569   0.10   0.05   0.09    0.10   0.05
## concavity..nucA.                      8 569   0.09   0.08   0.06    0.08   0.06
## concave.points..nucA.                 9 569   0.05   0.04   0.03    0.04   0.03
## symmetry..nucA.                      10 569   0.18   0.03   0.18    0.18   0.03
## fractal.dimension..nucA.             11 569   0.06   0.01   0.06    0.06   0.01
## radius..nucB.                        12 569   0.41   0.28   0.32    0.36   0.16
## texture..nucB.                       13 569   1.22   0.55   1.11    1.16   0.47
## perimeter..nucB.                     14 569   2.87   2.02   2.29    2.51   1.14
## area..nucB.                          15 569  40.34  45.49  24.53   31.69  13.63
## smoothness..nucB.                    16 569   0.01   0.00   0.01    0.01   0.00
## compactness..nucB.                   17 569   0.03   0.02   0.02    0.02   0.01
## concavity..nucB.                     18 569   0.03   0.03   0.03    0.03   0.02
## concave.points..nucB.                19 569   0.01   0.01   0.01    0.01   0.01
## symmetry..nucB.                      20 569   0.02   0.01   0.02    0.02   0.01
## fractal.dimension..nucB.             21 569   0.00   0.00   0.00    0.00   0.00
## radius..nucC.                        22 569  16.27   4.83  14.97   15.73   3.65
## texture..nucC.                       23 569  25.68   6.15  25.41   25.39   6.42
## perimeter..nucC.                     24 569 107.26  33.60  97.66  103.42  25.01
## area..nucC.                          25 569 880.58 569.36 686.50  788.02 319.65
## smoothness..nucC.                    26 569   0.13   0.02   0.13    0.13   0.02
## compactness..nucC.                   27 569   0.25   0.16   0.21    0.23   0.13
## concavity..nucC.                     28 569   0.27   0.21   0.23    0.25   0.20
## concave.points..nucC.                29 569   0.11   0.07   0.10    0.11   0.07
## symmetry..nucC.                      30 569   0.29   0.06   0.28    0.28   0.05
## fractal.dimension..nucC.             31 569   0.08   0.02   0.08    0.08   0.01
##                                       min     max   range skew kurtosis    se
## diagnosis..M.malignant..B.benign.*   1.00    2.00    1.00 0.53    -1.73  0.02
## radius..nucA.                        6.98   28.11   21.13 0.94     0.81  0.15
## texture..nucA.                       9.71   39.28   29.57 0.65     0.73  0.18
## perimeter..nucA.                    43.79  188.50  144.71 0.99     0.94  1.02
## area..nucA.                        143.50 2501.00 2357.50 1.64     3.59 14.75
## smoothness..nucA.                    0.05    0.16    0.11 0.45     0.82  0.00
## compactness..nucA.                   0.02    0.35    0.33 1.18     1.61  0.00
## concavity..nucA.                     0.00    0.43    0.43 1.39     1.95  0.00
## concave.points..nucA.                0.00    0.20    0.20 1.17     1.03  0.00
## symmetry..nucA.                      0.11    0.30    0.20 0.72     1.25  0.00
## fractal.dimension..nucA.             0.05    0.10    0.05 1.30     2.95  0.00
## radius..nucB.                        0.11    2.87    2.76 3.07    17.45  0.01
## texture..nucB.                       0.36    4.88    4.52 1.64     5.26  0.02
## perimeter..nucB.                     0.76   21.98   21.22 3.43    21.12  0.08
## area..nucB.                          6.80  542.20  535.40 5.42    48.59  1.91
## smoothness..nucB.                    0.00    0.03    0.03 2.30    10.32  0.00
## compactness..nucB.                   0.00    0.14    0.13 1.89     5.02  0.00
## concavity..nucB.                     0.00    0.40    0.40 5.08    48.24  0.00
## concave.points..nucB.                0.00    0.05    0.05 1.44     5.04  0.00
## symmetry..nucB.                      0.01    0.08    0.07 2.18     7.78  0.00
## fractal.dimension..nucB.             0.00    0.03    0.03 3.90    25.94  0.00
## radius..nucC.                        7.93   36.04   28.11 1.10     0.91  0.20
## texture..nucC.                      12.02   49.54   37.52 0.50     0.20  0.26
## perimeter..nucC.                    50.41  251.20  200.79 1.12     1.04  1.41
## area..nucC.                        185.20 4254.00 4068.80 1.85     4.32 23.87
## smoothness..nucC.                    0.07    0.22    0.15 0.41     0.49  0.00
## compactness..nucC.                   0.03    1.06    1.03 1.47     2.98  0.01
## concavity..nucC.                     0.00    1.25    1.25 1.14     1.57  0.01
## concave.points..nucC.                0.00    0.29    0.29 0.49    -0.55  0.00
## symmetry..nucC.                      0.16    0.66    0.51 1.43     4.37  0.00
## fractal.dimension..nucC.             0.06    0.21    0.15 1.65     5.16  0.00
```

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
```



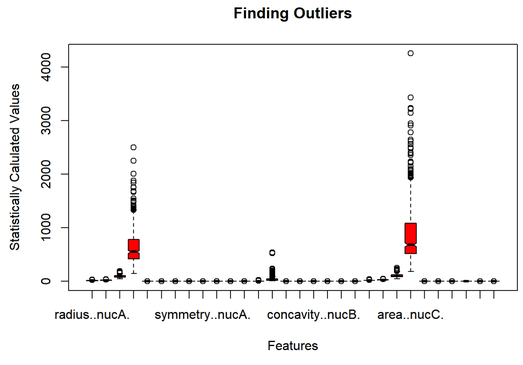

```
#cor.plot(data[c(-1)])
#cor.plot(createDummyFeatures(data)[c(-1)])
```

A strong correlation i.e. [0.8, 1] is showen 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
```



Removing Outliers

```
#install.packages("ggstatsplot")
#update.packages("ggstatsplot")
require("ggstatsplot", lib.loc="~/R/win-library/3.6")
```

```
## Loading required package: ggstatsplot
```

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

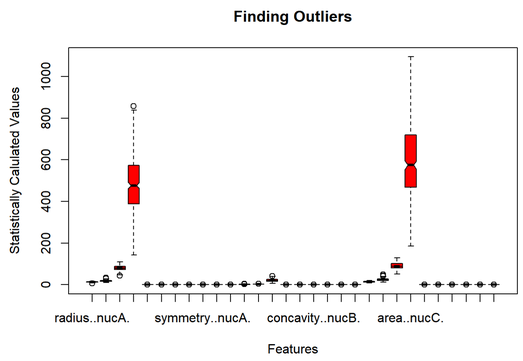
```
## Error: package or namespace load failed for 'ggstatsplot' in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), version
Check = vI[[j]]):
##  namespace 'PMCMRplus' 1.7.0 is being loaded, but >= 1.7.1 is required
```

```
for(i in c(1:3)){
  outliers <- boxplot(data$area..nucA., plot=FALSE)$out
  x <- data
  x <- x[-which(x$area..nucA. %in% outliers), ]
  #boxplot(x[c(-1)], col = "red")
  data <- x

  outliers <- boxplot(data$area..nucB., plot=FALSE)$out
  x <- data
  x <- x[-which(x$area..nucB. %in% outliers), ]
  #boxplot(x[c(-1)], col = "red")
  data <- x

  outliers <- boxplot(data$area..nucC., plot=FALSE)$out
  x <- data
  x <- x[-which(x$area..nucC. %in% outliers), ]
  #boxplot(x[c(-1)], col = "red")
  data <- x
}

boxplot(data[c(-1)], col = "red", main = "Finding Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Calulate
d Values")
```



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)]) #original data
```

```
##     radius..nucA. texture..nucA. perimeter..nucA. area..nucA. smoothness..nucA.
## 559         14.59          22.68            96.39       657.1           0.08473
## 560         11.51          23.93            74.52       403.5           0.09261
## 561         14.05          27.15            91.38       600.4           0.09929
## 562         11.20          29.37            70.67       386.0           0.07449
## 563         15.22          30.62           103.40       716.9           0.10480
## 569          7.76          24.54            47.92       181.0           0.05263
##     compactness..nucA. concavity..nucA. concave.points..nucA. symmetry..nucA.
## 559            0.13300          0.10290               0.03736          0.1454
## 560            0.10210          0.11120               0.04105          0.1388
## 561            0.11260          0.04462               0.04304          0.1537
## 562            0.03558          0.00000               0.00000          0.1060
## 563            0.20870          0.25500               0.09429          0.2128
## 569            0.04362          0.00000               0.00000          0.1587
##     fractal.dimension..nucA. radius..nucB. texture..nucB. perimeter..nucB.
## 559                  0.06147        0.2254          1.108            2.224
## 560                  0.06570        0.2388          2.904            1.936
## 561                  0.06171        0.3645          1.492            2.888
## 562                  0.05502        0.3141          3.896            2.041
## 563                  0.07152        0.2602          1.205            2.362
## 569                  0.05884        0.3857          1.428            2.548
##     area..nucB. smoothness..nucB. compactness..nucB. concavity..nucB.
## 559       19.54          0.004242           0.046390          0.06578
## 560       16.97          0.008200           0.029820          0.05738
## 561       29.84          0.007256           0.026780          0.02071
## 562       22.81          0.007594           0.008878          0.00000
## 563       22.65          0.004625           0.048440          0.07359
## 569       19.15          0.007189           0.004660          0.00000
##     concave.points..nucB. symmetry..nucB. fractal.dimension..nucB.
## 559               0.01606          0.01638                0.004406
## 560               0.01267          0.01488                0.004738
## 561               0.01626          0.02080                0.005304
## 562               0.00000          0.01989                0.001773
## 563               0.01608          0.02137                0.006142
## 569               0.00000          0.02676                0.002783
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 559        15.480          27.27           105.90       733.5           0.10260
## 560        12.480          37.16            82.28       474.2           0.12980
## 561        15.300          33.17           100.20       706.7           0.12410
## 562        11.920          38.30            75.19       439.6           0.09267
## 563        17.520          42.79           128.70       915.0           0.14170
## 569         9.456          30.37            59.16       268.6           0.08996
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 559            0.31710           0.3662                0.11050          0.2258
## 560            0.25170           0.3630                0.09653          0.2112
## 561            0.22640           0.1326                0.10480          0.2250
## 562            0.05494           0.0000                0.00000          0.1566
## 563            0.79170           1.1700                0.23560          0.4089
## 569            0.06444           0.0000                0.00000          0.2871
##     fractal.dimension..nucC.
## 559                  0.08004
## 560                  0.08732
## 561                  0.08321
## 562                  0.05905
## 563                  0.14090
## 569                  0.07039
```

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

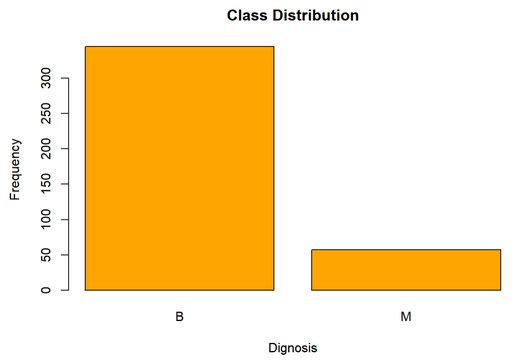
```
##     radius..nucA. texture..nucA. perimeter..nucA. area..nucA. smoothness..nucA.
## 559     1.2074246       1.044616        1.3320394   1.2613611       -0.66118207
## 560    -0.4776464       1.350437       -0.4341414  -0.5578477       -0.09888665
## 561     0.9119901       2.138231        0.9274411   0.8546216        0.37778004
## 562    -0.6472477       2.681368       -0.7450602  -0.6833846       -1.39188070
## 563     1.5520982       2.987189        1.8981540   1.6903386        0.77095869
## 569    -2.5292750       1.499677       -2.5823081  -2.1539596       -2.95175101
##     compactness..nucA. concavity..nucA. concave.points..nucA. symmetry..nucA.
## 559          1.0532034        0.8868547              0.3143427      -1.2279767
## 560          0.3240996        1.0467332              0.4868179      -1.4872965
## 561          0.5718533       -0.2357619              0.5798330      -0.9018624
## 562         -1.2454793       -1.0952533             -1.4319109      -2.7760373
## 563          2.8393898        3.8166762              2.9753227       1.4202283
## 569         -1.0557707       -1.0952533             -1.4319109      -0.7054081
##     fractal.dimension..nucA. radius..nucB. texture..nucB. perimeter..nucB.
## 559              -0.2747282    -0.5792937    -0.14577404       0.34864287
## 560               0.3390652    -0.4325470     3.08738365      -0.07790180
## 561              -0.2399030     0.9440246     0.54550244       1.33206532
## 562              -1.2106542     0.3920818     4.87318122       0.07760928
## 563               1.1835752    -0.1981903     0.02884528       0.55302886
## 569              -0.6563538     1.1761910     0.43028969       0.82850563
##     area..nucB. smoothness..nucB. compactness..nucB. concavity..nucB.
## 559  -0.2228502       -0.95795433          1.4397753         1.501510
## 560  -0.5682078        0.39301635          0.4399588         1.179023
## 561   1.1612679        0.07080404          0.2565284        -0.228788
## 562   0.2165737        0.18617243         -0.8236596        -1.023873
## 563   0.1950729       -0.82722624          1.5634701         1.801347
## 569  -0.2752585        0.04793516         -1.0781693        -1.023873
##     concave.points..nucB. symmetry..nucB. fractal.dimension..nucB.
## 559             1.1883397     -0.53089675                0.3399485
## 560             0.5218143     -0.73236658                0.4798061
## 561             1.2276628      0.06276771                0.7182380
## 562            -1.9693001     -0.05945732               -0.7692234
## 563             1.1922720      0.13932625                1.0712521
## 569            -1.9693001      0.86327453               -0.3437530
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 559     0.7741599      0.4561861        1.0136417   0.7468115       -1.16107491
## 560    -0.5979053      2.1132404       -0.5104709  -0.6513054        0.02343475
## 561     0.6918360      1.4447220        0.6458415   0.6023089       -0.22478970
## 562    -0.8540242      2.3042456       -0.9679629  -0.8378647       -1.59350803
## 563     1.7071643      3.0565382        2.4848428   1.7254394        0.54165773
## 569    -1.9809471      0.9755863       -2.0023204  -1.7598778       -1.71152352
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 559         0.67109766        0.8449898              0.4597465     -0.98501486
## 560         0.22169563        0.8277076              0.1802202     -1.23834065
## 561         0.04784439       -0.4166083              0.3456950     -0.99889572
## 562        -1.13035854       -1.1327380             -1.7512527     -2.18570970
## 563         3.93235458        5.1860536              2.9628777      2.19196817
## 569        -1.06507843       -1.1327380             -1.7512527      0.07860644
##     fractal.dimension..nucC.
## 559            -0.1716873034
## 560             0.2212809939
## 561            -0.0005733607
## 562            -1.3047099079
## 563             3.1134844789
## 569            -0.6925862140
```

Resampling

```
prop.table(table(data$diagnosis..M.malignant..B.benign.)) #frequency table for diagnosis into Malignant and Benign
```

```
##
##        B        M
## 0.858209 0.141791
```

```
barplot(table(data$diagnosis..M.malignant..B.benign.), col = "orange", xlab = "Dignosis", ylab = "Frequency", main = "Class
 Distribution") #frequency plot for diagnosis into Malignant and Benign
```




```
#install.packages("ROSE")
require(ROSE) #using rose library for over-sampling the data
```

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

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

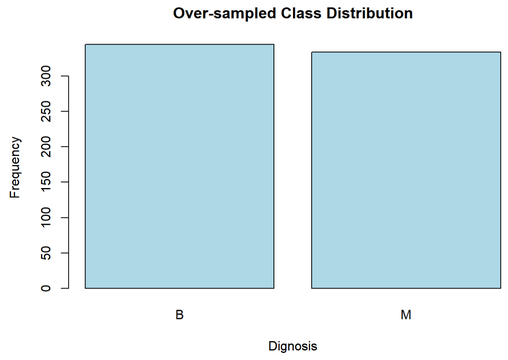
```
## Loaded ROSE 0.0-3
```

```
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.)) #frequency table for diagnosis into Malignant and Benign
```

```
##
##         B         M
## 0.5081001 0.4918999
```

```
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 = "Over-sampled Class Distribution") #f
requency plot for diagnosis into Malignant and Benign
```



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= 543
##
##           CP nsplit  rel error     xerror       xstd
## 1 0.83146067      0 1.00000000 1.03745318 0.04362842
## 2 0.03745318      1 0.16853933 0.16853933 0.02406078
## 3 0.03183521      2 0.13108614 0.15730337 0.02331483
## 4 0.02247191      4 0.06741573 0.10486891 0.01930063
## 5 0.01000000      5 0.04494382 0.07490637 0.01643822
##
## Variable importance
##    concave.points..nucC.    concave.points..nucA.         concavity..nucC.
##                       18                       15                       14
##         concavity..nucA.       compactness..nucC.       compactness..nucA.
##                       14                       12                       12
##           radius..nucC.         perimeter..nucC.              area..nucC.
##                        2                        2                        2
##             area..nucA.         perimeter..nucA.           radius..nucA.
##                        2                        2                        2
##          texture..nucC. fractal.dimension..nucC.        smoothness..nucA.
##                        1                        1                        1
##          texture..nucA.
##                        1
##
## Node number 1: 543 observations,    complexity param=0.8314607
##   predicted class=B  expected loss=0.4917127  P(node) =1
##     class counts:   276   267
##    probabilities: 0.508 0.492
##   left son=2 (251 obs) right son=3 (292 obs)
##   Primary splits:
##       concave.points..nucC. < -0.1665203   to the left,  improve=190.6126, (0 missing)
##       concave.points..nucA. < 0.1343971    to the left,  improve=170.9440, (0 missing)
##       concavity..nucC.      < -0.4698279   to the left,  improve=161.7647, (0 missing)
##       perimeter..nucC.      < 0.1631721    to the left,  improve=161.7017, (0 missing)
##       concavity..nucA.      < -0.2312098   to the left,  improve=153.0918, (0 missing)
##   Surrogate splits:
##       concave.points..nucA. < -0.2748458   to the left,  agree=0.917, adj=0.821, (0 split)
##       concavity..nucC.      < -0.2262298   to the left,  agree=0.908, adj=0.801, (0 split)
##       concavity..nucA.      < -0.3020997   to the left,  agree=0.895, adj=0.773, (0 split)
##       compactness..nucC.    < -0.197674    to the left,  agree=0.877, adj=0.733, (0 split)
##       compactness..nucA.    < -0.2867455   to the left,  agree=0.869, adj=0.717, (0 split)
##
## Node number 2: 251 observations,    complexity param=0.02247191
##   predicted class=B  expected loss=0.03984064  P(node) =0.4622468
##     class counts:   241    10
##    probabilities: 0.960 0.040
##   left son=4 (243 obs) right son=5 (8 obs)
##   Primary splits:
##       radius..nucA.     < 1.303419     to the left,  improve=11.52726, (0 missing)
##       perimeter..nucA.  < 1.129559     to the left,  improve=11.52726, (0 missing)
##       area..nucA.       < 1.43081      to the left,  improve=11.52726, (0 missing)
##       smoothness..nucB. < -1.483797    to the right, improve=11.52726, (0 missing)
##       radius..nucC.     < 1.210413     to the left,  improve=11.52726, (0 missing)
##   Surrogate splits:
##       perimeter..nucA. < 1.129559     to the left,  agree=1, adj=1, (0 split)
##       area..nucA.      < 1.43081      to the left,  agree=1, adj=1, (0 split)
##       radius..nucC.    < 1.210413     to the left,  agree=1, adj=1, (0 split)
##       perimeter..nucC. < 0.9525673    to the left,  agree=1, adj=1, (0 split)
##       area..nucC.      < 1.333888     to the left,  agree=1, adj=1, (0 split)
##
## Node number 3: 292 observations,    complexity param=0.03745318
##   predicted class=M  expected loss=0.119863  P(node) =0.5377532
##     class counts:    35   257
##    probabilities: 0.120 0.880
##   left son=6 (10 obs) right son=7 (282 obs)
##   Primary splits:
##       texture..nucC.        < -1.096192    to the left,  improve=16.04221, (0 missing)
##       concave.points..nucA. < 0.1343971    to the left,  improve=14.39048, (0 missing)
##       texture..nucA.        < -1.068879    to the left,  improve=14.38697, (0 missing)
##       radius..nucC.         < -0.004517485 to the left,  improve=14.23038, (0 missing)
##       perimeter..nucC.      < -0.6800321   to the left,  improve=12.74339, (0 missing)
##   Surrogate splits:
##       texture..nucA.   < -1.068879    to the left,  agree=0.983, adj=0.5, (0 split)
##       perimeter..nucB. < -1.347062    to the left,  agree=0.973, adj=0.2, (0 split)
##
## Node number 4: 243 observations
##   predicted class=B  expected loss=0.01234568  P(node) =0.4475138
##     class counts:   240     3
##    probabilities: 0.988 0.012
##
## Node number 5: 8 observations
##   predicted class=M  expected loss=0.125  P(node) =0.01473297
##     class counts:     1     7
##    probabilities: 0.125 0.875
##
## Node number 6: 10 observations
##   predicted class=B  expected loss=0  P(node) =0.01841621
##     class counts:    10     0
##    probabilities: 1.000 0.000
##
## Node number 7: 282 observations,    complexity param=0.03183521
##   predicted class=M  expected loss=0.08865248  P(node) =0.519337
##     class counts:    25   257
##    probabilities: 0.089 0.911
##   left son=14 (34 obs) right son=15 (248 obs)
##   Primary splits:
##       radius..nucC.         < -0.004517485 to the left,  improve=15.021360, (0 missing)
##       perimeter..nucC.      < -0.6800321   to the left,  improve=13.676860, (0 missing)
##       area..nucC.           < 0.01442689   to the left,  improve=12.172330, (0 missing)
##       concave.points..nucA. < 0.1343971    to the left,  improve=10.035930, (0 missing)
##       concave.points..nucC. < 0.233253     to the left,  improve= 7.572433, (0 missing)
##   Surrogate splits:
##       perimeter..nucC. < -0.07098562  to the left,  agree=0.989, adj=0.912, (0 split)
##       area..nucC.      < 0.01442689   to the left,  agree=0.979, adj=0.824, (0 split)
##       area..nucA.      < -0.6701464   to the left,  agree=0.947, adj=0.559, (0 split)
##       radius..nucA.    < -0.6659743   to the left,  agree=0.943, adj=0.529, (0 split)
##       perimeter..nucA. < -0.5931788   to the left,  agree=0.943, adj=0.529, (0 split)
##
## Node number 14: 34 observations,    complexity param=0.03183521
##   predicted class=B  expected loss=0.4705882  P(node) =0.0626151
##     class counts:    18    16
##    probabilities: 0.529 0.471
##   left son=28 (17 obs) right son=29 (17 obs)
##   Primary splits:
##       concave.points..nucA. < 0.4028951    to the left,  improve=15.058820, (0 missing)
##       concave.points..nucC. < 0.3325954    to the left,  improve=10.541180, (0 missing)
##       smoothness..nucA.     < 0.6708758    to the left,  improve= 9.322129, (0 missing)
##       concavity..nucA.      < 0.2170118    to the left,  improve= 9.322129, (0 missing)
##       concavity..nucC.      < 0.3597701    to the left,  improve= 8.213904, (0 missing)
##   Surrogate splits:
##       concave.points..nucC.    < 0.3325954    to the left,  agree=0.912, adj=0.824, (0 split)
##       concavity..nucA.         < 0.2170118    to the left,  agree=0.882, adj=0.765, (0 split)
##       concavity..nucC.         < 0.3597701    to the left,  agree=0.853, adj=0.706, (0 split)
##       smoothness..nucA.        < 0.6708758    to the left,  agree=0.824, adj=0.647, (0 split)
##       fractal.dimension..nucC. < 0.1733008    to the left,  agree=0.824, adj=0.647, (0 split)
##
## Node number 15: 248 observations
##   predicted class=M  expected loss=0.02822581  P(node) =0.4567219
##     class counts:     7   241
##    probabilities: 0.028 0.972
##
## Node number 28: 17 observations
##   predicted class=B  expected loss=0  P(node) =0.03130755
##     class counts:    17     0
##    probabilities: 1.000 0.000
##
## Node number 29: 17 observations
##   predicted class=M  expected loss=0.05882353  P(node) =0.03130755
##     class counts:     1    16
##    probabilities: 0.059 0.941
```

Predictions

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

```
## 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
```

```
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 94.12
```

```
## Confusion Matrix and Statistics
##
##    dtresult
##      B  M
##   B 63  6
##   M  2 65
##
##                Accuracy : 0.9412
##                  95% CI : (0.8874, 0.9743)
##     No Information Rate : 0.5221
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8824
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9692
##             Specificity : 0.9155
##          Pos Pred Value : 0.9130
##          Neg Pred Value : 0.9701
##              Prevalence : 0.4779
##          Detection Rate : 0.4632
##    Detection Prevalence : 0.5074
##       Balanced Accuracy : 0.9424
##
##        '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
```

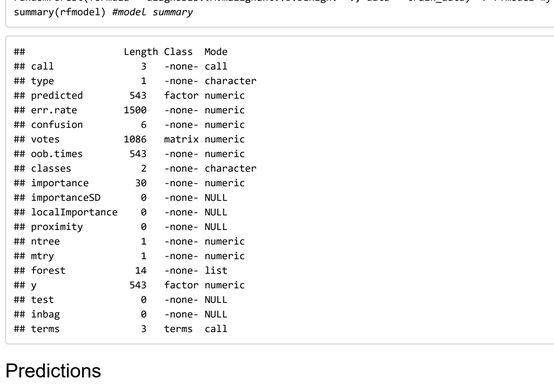
```
## Loading required package: sandwich
```

```
## 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        543   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes           1086   matrix numeric
## oob.times        543   -none- numeric
## classes            2   -none- character
## importance        30   -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                543   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)
```

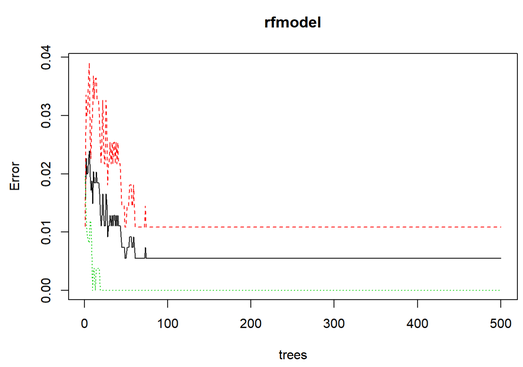
Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., rfresult)) #9the maximum accuracy of the model is 98.53
```

```
## Confusion Matrix and Statistics
##
##    rfresult
##      B  M
##   B 66  3
##   M  0 67
##
##                Accuracy : 0.9779
##                  95% CI : (0.9369, 0.9954)
##     No Information Rate : 0.5147
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9559
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9571
##          Pos Pred Value : 0.9565
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4853
##          Detection Rate : 0.4853
##    Detection Prevalence : 0.5074
##       Balanced Accuracy : 0.9786
##
##        '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:  49
##
##  ( 24 25 )
##
##
## 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 98.75
```

```
## Confusion Matrix and Statistics
##
##    svmresult
##      B  M
##   B 67  2
##   M  4 63
##
##               Accuracy : 0.9559
##                 95% CI : (0.9064, 0.9836)
##    No Information Rate : 0.5221
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9117
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9437
##             Specificity : 0.9692
##          Pos Pred Value : 0.9710
##          Neg Pred Value : 0.9403
##              Prevalence : 0.5221
##          Detection Rate : 0.4926
##    Detection Prevalence : 0.5074
##       Balanced Accuracy : 0.9564
##
##        '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    30     -none- list
## levels     2     -none- character
## isnumeric 30     -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 96.25
```

```
## Confusion Matrix and Statistics
##
##    nbresult
##      B  M
##   B 62  7
##   M  6 61
##
##               Accuracy : 0.9044
##                 95% CI : (0.8421, 0.9481)
##    No Information Rate : 0.5
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8088
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9118
##             Specificity : 0.8971
##          Pos Pred Value : 0.8986
##          Neg Pred Value : 0.9104
##              Prevalence : 0.5000
##          Detection Rate : 0.4559
##    Detection Prevalence : 0.5074
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : B
##
```

KNN

```
# require(class) #using library class to build a KNN model
#
# knn(train, test, cl = train$diagnosis..M.malignant..B.benign., k=5) -> knnmodel #fitting the model
# confusionMatrix(table(test$diagnosis..M.malignant..B.benign., knnmodel)) #the maximum accuracy of the model is 98.75
```

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.
## 671                                 1     1.3959040    -0.04648936
## 672                                 1     0.4112075    -0.92879190
## 673                                 1     0.3731253     0.74733715
## 675                                 1     0.9933209    -0.71251719
## 677                                 1     0.4112075    -0.92879190
## 679                                 1     0.7648278     0.15995468
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 671        1.1794760   1.5445701       -1.32980965         -1.0302595
## 672        0.4271786   0.3451039        0.14507581          0.3158605
## 673        0.6763722   0.3052362        0.96519150          2.2139948
## 675        1.0725665   1.0214791        1.17022042          1.0072836
## 677        0.4271786   0.3451039        0.14507581          0.3158605
## 679        0.7463351   0.7479046        0.01412186          0.1772012
##     concavity..nucA. concave.points..nucA. symmetry..nucA.
## 671       -0.6969296            -0.6036115      -1.1041854
## 672       -0.1615164             0.1982736      -0.6838445
## 673        1.8561539             1.2650479       0.8037966
## 675        1.2010953             1.5891221       1.1254487
## 677       -0.1615164             0.1982736      -0.6838445
## 679        0.4383558             0.5270393       0.3688352
##     fractal.dimension..nucA. radius..nucB. texture..nucB. perimeter..nucB.
## 671              -1.3468887    -0.7272243    -0.31122886       -0.9195451
## 672               0.1078856    -0.1842480    -1.06312302       -0.3044437
## 673               1.5535917    -0.9237617     0.02705395       -0.1532131
## 675               0.3695636     1.4580054    -0.41040813        1.1962292
## 677               0.1078856    -0.1842480    -1.06312302       -0.3044437
## 679              -0.0216580    -0.9559628    -1.00270118       -1.0475094
##     area..nucB. smoothness..nucB. compactness..nucB. concavity..nucB.
## 671 -0.21670387       -1.54989432        -1.00130216      -0.77450395
## 672 -0.07033233       -0.82108637        -0.32568122      -0.58572999
## 673 -0.58826240       -0.16126660         1.93965325       0.99103827
## 675  2.16026985        0.05302066        -0.03424837      -0.05817417
## 677 -0.07033233       -0.82108637        -0.32568122      -0.58572999
## 679 -0.56199058       -1.08046646        -0.59955788      -0.48417970
##     concave.points..nucB. symmetry..nucB. fractal.dimension..nucB.
## 671            -1.2287634     -1.15076704              -1.11988147
## 672            -0.4316557     -0.91639217              -0.31496359
## 673             0.9668412     -0.07739808               1.92302377
## 675             0.2610688     -0.48387430               0.02504094
## 677            -0.4316557     -0.91639217              -0.31496359
## 679            -0.8078700     -0.57445396              -0.83534187
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 671    1.32580935      0.8055690         1.017611  1.47188263        -1.1643857
## 672    0.74229447     -0.9227234         0.686479  0.60195140         0.0720850
## 673    0.07168782      0.8599279         0.597783  0.03565677         0.9795864
## 675    1.71772233      0.1180892         1.679875  1.99719866         1.8379316
## 677    0.74229447     -0.9227234         0.686479  0.60195140         0.0720850
## 679    1.50870208      0.5577565         1.135873  1.74540244         0.6052421
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 671         -0.6384312       -0.3914892            -0.5698186      -0.5295484
## 672          0.2649517       -0.1630747             0.2733104      -0.6499918
## 673          2.4429637        1.6695227             1.5952059       0.6402904
## 675          0.7533513        0.6841275             1.3051901       0.6261960
## 677          0.2649517       -0.1630747             0.2733104      -0.6499918
## 679          0.5325822        0.7998751             1.0023558       2.1227696
##     fractal.dimension..nucC.
## 671               -0.9736145
## 672                0.4653673
## 673                2.1278038
## 675                0.6104677
## 677                0.4653673
## 679                0.3948899
```

```
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.
## 662                                 1     0.7811487     0.46470542
## 663                                 1    -0.3830781     1.79184573
## 664                                 1     0.6560215     0.52860477
## 674                                 1     0.7485068     0.04935966
## 676                                 1     0.8355518     0.48190909
## 678                                 1     0.8137906     1.95896710
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 662        0.7274687   0.7857101       -0.26895033        -0.21872751
## 663       -0.3023219  -0.3883234        0.32364939         0.45264612
## 664        0.6787305   0.6812294       -0.01828594        -0.02516654
## 674        0.7455490   0.7465298       -0.06524418         0.05540579
## 676        0.9758762   0.7623394        0.45592611         1.41764032
## 678        0.9224214   0.8585717        1.01810219         0.90609971
##     concavity..nucA. concave.points..nucA. symmetry..nucA.
## 662       0.47424941            0.15099776       0.8257274
## 663       0.29627686            0.01530522       0.3432492
## 664       0.64623968            0.38918147       0.1312512
## 674       0.07762488            0.47615461       0.1093204
## 676       0.80476985            1.53859830       1.4726868
## 678       1.12482132            1.02650333       1.6590988
##     fractal.dimension..nucA. radius..nucB. texture..nucB. perimeter..nucB.
## 662              -1.0968696    1.38916178      2.1159801       1.64992101
## 663               0.5884922    1.22149426      1.2990927       0.69164250
## 664              -0.4634016   -0.10319020     -0.2710801       0.71345460
## 674              -0.1201111   -0.06432687      1.3825702       0.30484115
## 676               0.5716515   -0.45296020     -0.3422348      -0.08196019
## 678               0.7698532    0.82952979     -0.2432543       1.03627377
##     area..nucB. smoothness..nucB. compactness..nucB. concavity..nucB.
## 662   1.8712799         0.9718351          0.2115384       0.93626012
## 663   1.3170697         0.8673893          0.7159864       0.25869642
## 664   0.1248297        -0.3736268          0.2185608       0.70197814
## 674   0.3675312         0.4461376          0.1600402       0.07245067
## 676  -0.3580713        -0.9232197          0.2536732      -0.19722642
## 678   1.0806234        -0.4780726          0.9471430       0.67079673
##     concave.points..nucB. symmetry..nucB. fractal.dimension..nucB.
## 662            1.22977604      0.44003822              -0.26130240
## 663            0.35991648     -0.17137448               0.83131162
## 664            0.04360391     -0.00153762               0.08726988
## 674            0.56354269     -0.64238871               0.28117167
## 676            0.42120204     -0.65144667              -0.05297071
## 678           -0.09675979     -0.19515164               0.73841914
##     radius..nucC. texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 662     0.5855293      0.4186617       0.5741307   0.5846713       -0.3249469
## 663     0.3416723      2.0318412       0.1779548   0.3871840        1.4862748
## 664     0.4461824      0.5961275       1.0353504   0.5387555        0.2044290
## 674     0.6377843      0.6760670       0.6391745   0.6710719        0.1477101
## 676     1.1995263      1.0517827       1.4019609   1.0191432        0.5031482
## 678     1.1298529      1.6785085       1.5024831   1.2477346        1.0816804
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 662       -0.550435961       0.08778516            0.02014739     -0.09774588
## 663        0.871025341       0.56133818            0.39508503      0.15211016
## 664        0.627346261       0.89625811            0.37585746      0.63388385
## 674       -0.003720076       0.01956888            0.56492858     -0.48470241
## 676        1.879585979       1.05161518            2.38513865      1.49620750
## 678        1.845220981       1.70605137            0.80046632      1.43726710
##     fractal.dimension..nucC.
## 662              -0.96656675
## 663               1.19086949
## 664               0.39903567
## 674               0.07276697
## 676               1.48107037
## 678               1.75468834
```

Fitting Model

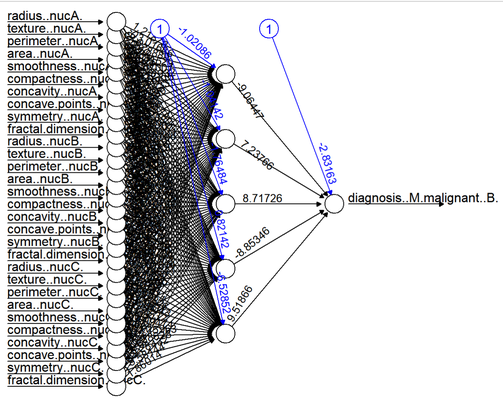
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, hidden = 5, err.fct = "ce", linear.output = FALSE, lifesign = 'full', rep = 1, algorithm = "rprop+", stepmax = 100000) -> nnmodel #fitting the model
```

```
## hidden: 5    thresh: 0.01    rep: 1/1    steps:      91  error: 0.03966  time: 0.09 secs
```

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

```
##                   Length Class      Mode
## call                  10 -none-     call
## response             543 -none-     numeric
## covariate          16290 -none-     numeric
## model.list             2 -none-     list
## err.fct                1 -none-     function
## act.fct                1 -none-     function
## linear.output          1 -none-     logical
## data                  31 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        164 -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
## 5       0 1.161572e-09
## 12      0 9.844518e-10
## 16      0 9.740602e-10
## 17      0 3.608597e-08
## 23      0 1.017906e-09
## 26      0 9.938265e-01
```

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 67  2
##      1  2 65
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 97.5
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 67  2
##      1  2 65
##
##               Accuracy : 0.9706
##                 95% CI : (0.9264, 0.9919)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9412
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9710
##            Specificity : 0.9701
##         Pos Pred Value : 0.9710
##         Neg Pred Value : 0.9701
##             Prevalence : 0.5074
##         Detection Rate : 0.4926
##   Detection Prevalence : 0.5074
##      Balanced Accuracy : 0.9706
##
##       'Positive' Class : 0
##
```

Neural Network: Model 2

Fitting Model

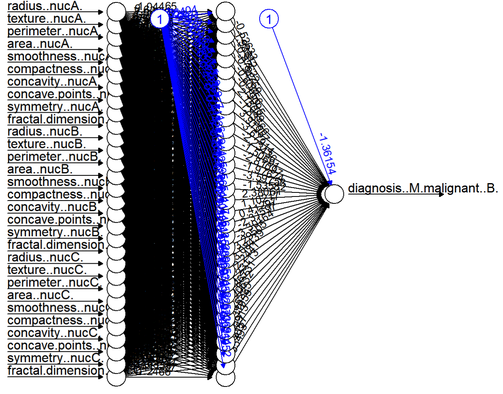
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, threshold = 0.03, hidden = 32, err.fct = "ce", linear.output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> nnmodel
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.217965433376419
##                                           2000 min thresh: 0.0962128267962865
##                                           3000 min thresh: 0.0604581594587903
##                                           4000 min thresh: 0.0437216015083219
##                                           5000 min thresh: 0.0340884896648613
##                                           5611 error: 0.18605  time: 14.46 secs
```

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

```
##                   Length Class      Mode
## call                  13 -none-     call
## response             543 -none-     numeric
## covariate          16290 -none-     numeric
## model.list             2 -none-     list
## err.fct                1 -none-     function
## act.fct                1 -none-     function
## linear.output          1 -none-     logical
## data                  31 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       1028 -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
## 5       0 4.462708e-10
## 12      0 1.937442e-10
## 16      0 4.749167e-11
## 17      0 6.620901e-08
## 23      0 1.671939e-07
## 26      0 2.386366e-01
```

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 68  1
##      1  0 67
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 98.75
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 68  1
##      1  0 67
##
##               Accuracy : 0.9926
##                 95% CI : (0.9597, 0.9998)
##    No Information Rate : 0.5
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9853
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 1.0000
##            Specificity : 0.9853
##         Pos Pred Value : 0.9855
##         Neg Pred Value : 1.0000
##             Prevalence : 0.5000
##         Detection Rate : 0.5000
##   Detection Prevalence : 0.5074
##      Balanced Accuracy : 0.9926
##
##       '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$dia
gnosis..M.malignant..B.benign.)) #averaged ensemble model with maximum accuracy 97.5
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 66  2
##   1  3 65
##
##               Accuracy : 0.9632
##                 95% CI : (0.9163, 0.988)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9265
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9565
##             Specificity : 0.9701
##          Pos Pred Value : 0.9706
##          Neg Pred Value : 0.9559
##              Prevalence : 0.5074
##          Detection Rate : 0.4853
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9633
##
##       '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.)) #average
d ensemble model with maximum accuracy 97.5
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 68  4
##   1  1 63
##
##               Accuracy : 0.9632
##                 95% CI : (0.9163, 0.988)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9264
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9855
##             Specificity : 0.9403
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9844
##              Prevalence : 0.5074
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5294
##       Balanced Accuracy : 0.9629
##
##       '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.)) #average
d ensemble model with maximum accuracy 98.75
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 68  4
##   1  1 63
##
##               Accuracy : 0.9632
##                 95% CI : (0.9163, 0.988)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9264
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9855
##             Specificity : 0.9403
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9844
##              Prevalence : 0.5074
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5294
##       Balanced Accuracy : 0.9629
##
##       '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.)) #averaged
ensemble model with maximum accuracy 98.75
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 67  6
##   1  2 61
##
##               Accuracy : 0.9412
##                 95% CI : (0.8874, 0.9743)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8822
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9710
##             Specificity : 0.9104
##          Pos Pred Value : 0.9178
##          Neg Pred Value : 0.9683
##              Prevalence : 0.5074
##          Detection Rate : 0.4926
##    Detection Prevalence : 0.5368
##       Balanced Accuracy : 0.9407
##
##       '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.)) #av
eraged ensemble model with maximum accuracy 98.75
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 66  0
##   1  3 67
##
##               Accuracy : 0.9779
##                 95% CI : (0.9369, 0.9954)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9559
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.9565
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9571
##              Prevalence : 0.5074
##          Detection Rate : 0.4853
##    Detection Prevalence : 0.4853
##       Balanced Accuracy : 0.9783
##
##       '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.)) #averaged
ensemble model with maximum accuracy 97.5
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 65  6
##   1  4 61
##
##               Accuracy : 0.9265
##                 95% CI : (0.8689, 0.9642)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8528
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9420
##             Specificity : 0.9104
##          Pos Pred Value : 0.9155
##          Neg Pred Value : 0.9385
##              Prevalence : 0.5074
##          Detection Rate : 0.4779
##    Detection Prevalence : 0.5221
##       Balanced Accuracy : 0.9262
##
##       '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.)) #averaged
ensemble model with maximum accuracy 98.75
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 67  4
##   1  2 63
##
##               Accuracy : 0.9559
##                 95% CI : (0.9064, 0.9836)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9117
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9710
##             Specificity : 0.9403
##          Pos Pred Value : 0.9437
##          Neg Pred Value : 0.9692
##              Prevalence : 0.5074
##          Detection Rate : 0.4926
##    Detection Prevalence : 0.5221
##       Balanced Accuracy : 0.9557
##
##       '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.)) #averaged
ensemble model with maximum accuracy 97.5
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 62  6
##   1  7 61
##
##               Accuracy : 0.9044
##                 95% CI : (0.8421, 0.9481)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.8088
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8986
##             Specificity : 0.9104
##          Pos Pred Value : 0.9118
##          Neg Pred Value : 0.8971
##              Prevalence : 0.5074
##          Detection Rate : 0.4559
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9045
##
##       '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.)) #av
eraged ensemble model with maximum accuracy 98.75
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 65  0
##   1  4 67
##
##               Accuracy : 0.9706
##                 95% CI : (0.9264, 0.9919)
##    No Information Rate : 0.5074
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9412
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.9420
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9437
##              Prevalence : 0.5074
##          Detection Rate : 0.4779
##    Detection Prevalence : 0.4779
##       Balanced Accuracy : 0.9710
##
##       '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

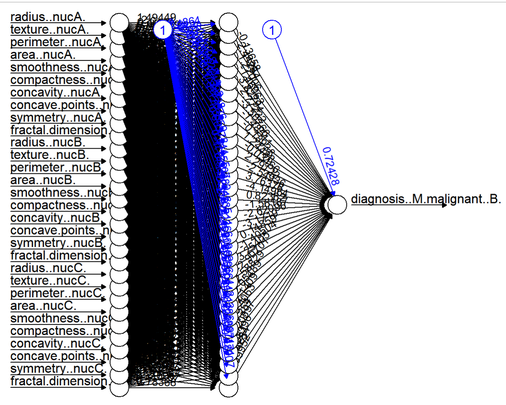
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = ensembletrain, threshold = 0.03, hidden = 32, err.fct = "ce", linear.
output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> ensemblemodel #fitting th
e model
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.192942924087557
##                                           2000 min thresh: 0.0902704276126855
##                                           3000 min thresh: 0.0575713627933114
##                                           4000 min thresh: 0.0418529489415114
##                                           5000 min thresh: 0.0327056433197237
##                                           5408 error: 0.18613  time: 11.54 secs
```

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

```
##                     Length Class      Mode
## call                   13  -none-     call
## response              543  -none-     numeric
## covariate           16290  -none-     numeric
## model.list              2  -none-     list
## err.fct                 1  -none-     function
## act.fct                 1  -none-     function
## linear.output           1  -none-     logical
## data                   31  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        1028  -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
## 5       0 4.270690e-11
## 12      0 4.067729e-09
## 16      0 1.470873e-09
## 17      0 5.565320e-10
## 23      0 2.397216e-07
## 26      0 9.531038e-01
```

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 98.53
```

```
## Confusion Matrix and Statistics
##
##       prediction
## actual  0  1
##      0 69  3
##      1  0 64
##
##               Accuracy : 0.9779
##                 95% CI : (0.9369, 0.9954)
##     No Information Rate : 0.5074
##     P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9558
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9552
##          Pos Pred Value : 0.9583
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5074
##          Detection Rate : 0.5074
##    Detection Prevalence : 0.5294
##       Balanced Accuracy : 0.9776
##
##        'Positive' Class : 0
##
```