R Final Project : Breast Cancer Classification :: Notebook 3

Utpal Mishra - 20207425

24 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

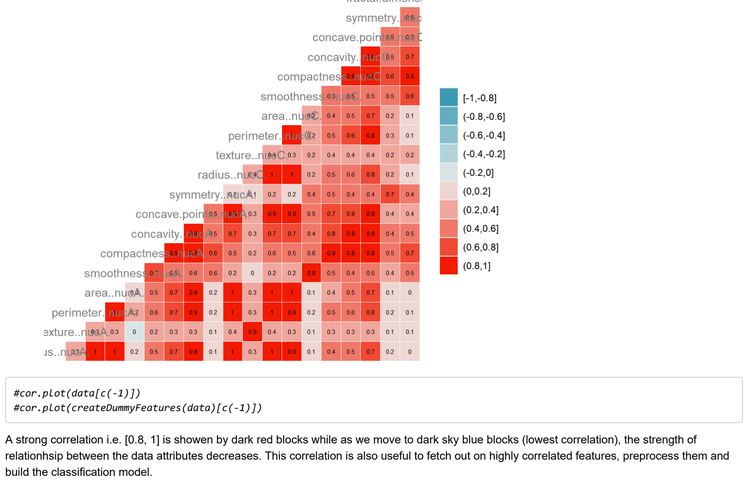
```
data = data[, c("diagnosis..M.malignant..B.benign.", "radius..nucA.", "texture..nucA.", "perimeter..nucA.", "area..nucA.",
"smoothness..nucA.", "compactness..nucA.", "concavity..nucA.", "concave.points..nucA.", "symmetry..nucA.", "radius..nucC.",
"texture..nucC.", "perimeter..nucC.", "area..nucC.",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



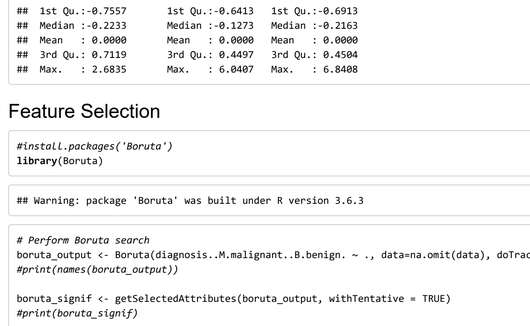

```
#cor.plot(data[c(-1)])
#cor.plot(createDummyFeatures(data)[c(-1)])
```

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

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

```
##  radius..nucA.     texture..nucA.    perimeter..nucA.   area..nucA.
##  Min.   :-2.0279   Min.   :-2.2273   Min.   :-1.9828   Min.   :-1.4532
##  1st Qu.:-0.6888   1st Qu.:-0.7253   1st Qu.:-0.6913   1st Qu.:-0.6666
##  Median :-0.2149   Median :-0.1045   Median :-0.2358   Median :-0.2949
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4690   3rd Qu.: 0.5837   3rd Qu.: 0.4992   3rd Qu.: 0.3632
##  Max.   : 3.9678   Max.   : 4.6478   Max.   : 3.9726   Max.   : 5.2459
##  smoothness..nucA.  compactness..nucA. concavity..nucA.  concave.points..nucA.
##  Min.   :-3.10935   Min.   :-1.6087    Min.   :-1.1139   Min.   :-1.2607
##  1st Qu.:-0.71034   1st Qu.:-0.7464    1st Qu.:-0.7431   1st Qu.:-0.7373
##  Median :-0.03486   Median :-0.2217    Median :-0.3419   Median :-0.3974
##  Mean   : 0.00000   Mean   : 0.0000    Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.63564   3rd Qu.: 0.4934    3rd Qu.: 0.5256   3rd Qu.: 0.6464
##  Max.   : 4.76672   Max.   : 4.5644    Max.   : 4.2399   Max.   : 3.9245
##  symmetry..nucA.    radius..nucC.      texture..nucC.     perimeter..nucC.
##  Min.   :-2.74171   Min.   :-1.7254   Min.   :-2.22204   Min.   :-1.6919
##  1st Qu.:-0.70262   1st Qu.:-0.6743   1st Qu.:-0.74797   1st Qu.:-0.6890
##  Median :-0.07156   Median :-0.2688   Median :-0.04348   Median :-0.2857
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.53031   3rd Qu.: 0.5216   3rd Qu.: 0.65776   3rd Qu.: 0.5398
##  Max.   : 4.48081   Max.   : 4.0906   Max.   : 3.88249   Max.   : 4.2836
##   area..nucC.      smoothness..nucC. compactness..nucC. concavity..nucC.
##  Min.   :-1.2213   Min.   :-2.6803   Min.   :-1.4426    Min.   :-1.3047
##  1st Qu.:-0.6416   1st Qu.:-0.6906   1st Qu.:-0.6805    1st Qu.:-0.7558
##  Median :-0.3409   Median :-0.0468   Median :-0.2693    Median :-0.2180
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.3573   3rd Qu.: 0.5970   3rd Qu.: 0.5392    3rd Qu.: 0.5307
##  Max.   : 5.9250   Max.   : 3.9519   Max.   : 5.1084    Max.   : 4.6965
##  concave.points..nucC. symmetry..nucC.   fractal.dimension..nucC.
##  Min.   :-1.7435       Min.   :-2.1591   Min.   :-1.6004
##  1st Qu.:-0.7557       1st Qu.:-0.6413   1st Qu.:-0.6913
##  Median :-0.2233       Median :-0.1273   Median :-0.2163
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.7119       3rd Qu.: 0.4497   3rd Qu.: 0.4504
##  Max.   : 2.6835       Max.   : 6.0407   Max.   : 6.8408
```

Feature Selection

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

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

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

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

roughFixMod <- TentativeRoughFix(boruta_output)
```

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

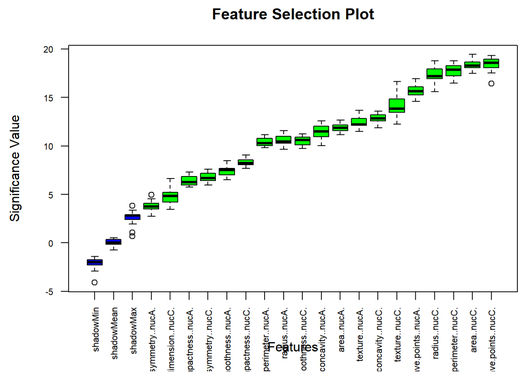
```
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)
```

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

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

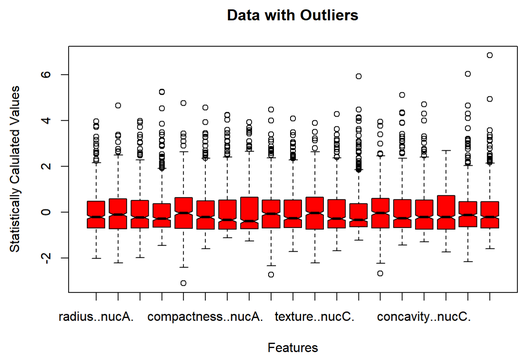
```
##                       meanImp  decision
## concave.points..nucC. 18.40253 Confirmed
## area..nucC.           18.37066 Confirmed
## perimeter..nucC.      17.73933 Confirmed
## radius..nucC.         17.34329 Confirmed
## concave.points..nucA. 15.69123 Confirmed
## texture..nucC.        14.11237 Confirmed
```

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



Removing Outliers

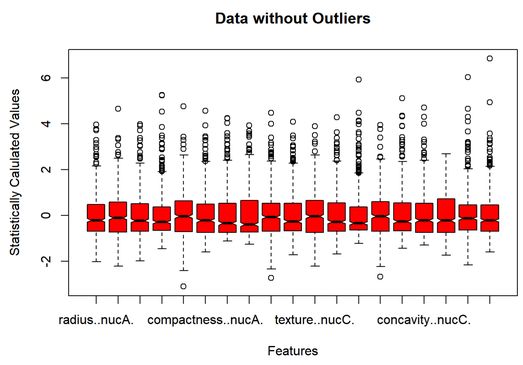
```
boxplot(data[c(-1)], col = "red", main = "Data with Outliers", notch = TRUE, xlab = "Features", ylab = "Statistically Calula
ted Values")  #using boxplot to represent data with outliers
```




```
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)])))
```

```
##    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
```

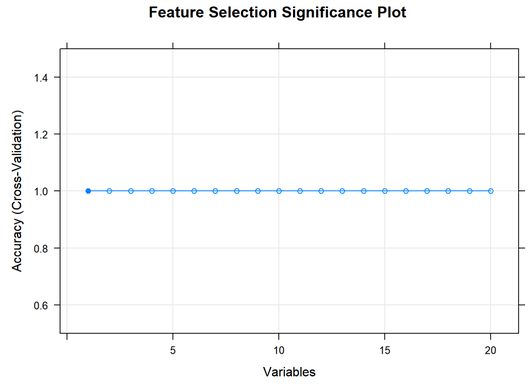
```
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(data[, c(1:dim(data)[2])], data[, c(1)], sizes=c(1:dim(data)[2]), rfeControl=control)
# summarize the results
print(results)
```

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



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= 456
##
##           CP nsplit rel error    xerror       xstd
## 1 0.82352941      0 1.0000000 1.0000000 0.06074019
## 2 0.07058824      1 0.1764706 0.2294118 0.03512926
## 3 0.01000000      2 0.1058824 0.1764706 0.03114111
##
## Variable importance
##        radius..nucC.           area..nucC.      perimeter..nucC.
##                   17                    16                    16
##          area..nucA.         radius..nucA.      perimeter..nucA.
##                   15                    15                    15
## concave.points..nucC.    compactness..nucC. concave.points..nucA.
##                    2                     1                     1
##      symmetry..nucC.      concavity..nucA.      concavity..nucC.
##                    1                     1                     1
##
## Node number 1: 456 observations,    complexity param=0.8235294
##   predicted class=B  expected loss=0.372807  P(node) =1
##     class counts:   286   170
##    probabilities: 0.627 0.373
##   left son=2 (304 obs) right son=3 (152 obs)
##   Primary splits:
##       radius..nucC.         < 0.1077559   to the left,  improve=157.5088, (0 missing)
##       perimeter..nucC.      < 0.08894527  to the left,  improve=157.2307, (0 missing)
##       area..nucC.           < -0.02174932 to the left,  improve=155.9851, (0 missing)
##       concave.points..nucC. < 0.4121834   to the left,  improve=145.2986, (0 missing)
##       concave.points..nucA. < 0.1804212   to the left,  improve=143.6826, (0 missing)
##   Surrogate splits:
##       area..nucC.      < -0.02174932 to the left,  agree=0.993, adj=0.980, (0 split)
##       perimeter..nucC. < 0.1320968   to the left,  agree=0.982, adj=0.947, (0 split)
##       radius..nucA.    < 0.260413    to the left,  agree=0.961, adj=0.882, (0 split)
##       area..nucA.      < 0.1219357   to the left,  agree=0.961, adj=0.882, (0 split)
##       perimeter..nucA. < 0.1825577   to the left,  agree=0.958, adj=0.875, (0 split)
##
## Node number 2: 304 observations,    complexity param=0.07058824
##   predicted class=B  expected loss=0.07894737  P(node) =0.6666667
##     class counts:   280    24
##    probabilities: 0.921 0.079
##   left son=4 (288 obs) right son=5 (16 obs)
##   Primary splits:
##       concave.points..nucC. < 0.6951491   to the left,  improve=21.40497, (0 missing)
##       concave.points..nucA. < 0.008011117 to the left,  improve=15.88002, (0 missing)
##       compactness..nucC.    < 0.8302903   to the left,  improve=11.76739, (0 missing)
##       concavity..nucC.      < 0.4664439   to the left,  improve=11.32006, (0 missing)
##       perimeter..nucC.      < -0.1550839  to the left,  improve=10.88852, (0 missing)
##   Surrogate splits:
##       compactness..nucC.    < 0.9815584   to the left,  agree=0.974, adj=0.500, (0 split)
##       concave.points..nucA. < 0.7857891   to the left,  agree=0.970, adj=0.438, (0 split)
##       symmetry..nucC.       < 1.157707    to the left,  agree=0.970, adj=0.438, (0 split)
##       concavity..nucA.      < 0.8467241   to the left,  agree=0.964, adj=0.313, (0 split)
##       concavity..nucC.      < 0.911023    to the left,  agree=0.964, adj=0.313, (0 split)
##
## Node number 3: 152 observations
##   predicted class=M  expected loss=0.03947368  P(node) =0.3333333
##     class counts:     6   146
##    probabilities: 0.039 0.961
##
## Node number 4: 288 observations
##   predicted class=B  expected loss=0.03472222  P(node) =0.6315789
##     class counts:   278    10
##    probabilities: 0.965 0.035
##
## Node number 5: 16 observations
##   predicted class=M  expected loss=0.125  P(node) =0.03508772
##     class counts:     2    14
##    probabilities: 0.125 0.875
```

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 92.04
```

```
## Confusion Matrix and Statistics
##
##    dtresult
##      B  M
##   B 66  5
##   M  5 37
##
##               Accuracy : 0.9115
##                 95% CI : (0.8433, 0.9567)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : 6.062e-12
##
##                  Kappa : 0.8105
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9296
##             Specificity : 0.8810
##          Pos Pred Value : 0.9296
##          Neg Pred Value : 0.8810
##              Prevalence : 0.6283
##          Detection Rate : 0.5841
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9053
##
##        '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
```

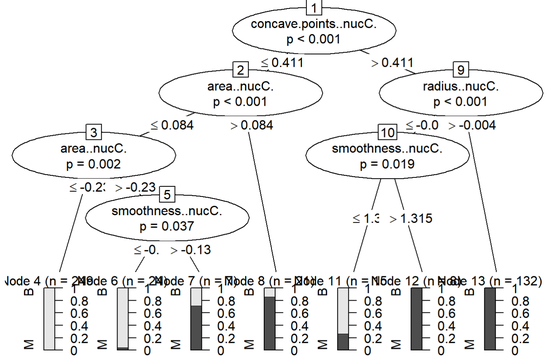
```
## Loading required package: sandwich
```

```
## 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        456   factor numeric
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes            912   matrix numeric
## oob.times        456   -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                456   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)
```

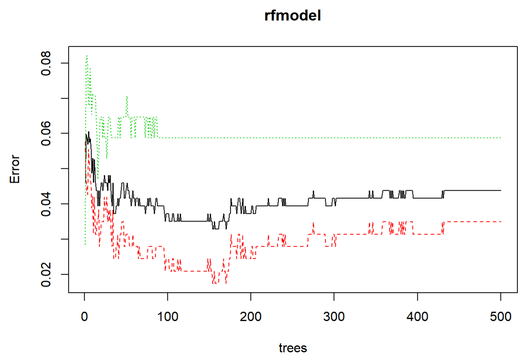
Confusion Matrix

```
confusionMatrix(table(test_data$diagnosis..M.malignant..B.benign., rfresult)) #the maximum accuracy of the model is 97.35
```

```
## Confusion Matrix and Statistics
##
##    rfresult
##      B  M
##   B 69  2
##   M  2 40
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9524
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9524
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9621
##
##        '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:  37
##
##  ( 19 18 )
##
##
## 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 98.23
```

```
## Confusion Matrix and Statistics
##
##    svmresult
##      B  M
##   B 70  1
##   M  1 41
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9621
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9859
##             Specificity : 0.9762
##          Pos Pred Value : 0.9859
##          Neg Pred Value : 0.9762
##              Prevalence : 0.6283
##          Detection Rate : 0.6195
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9811
##
##        '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 95.58
```

```
## Confusion Matrix and Statistics
##
##    nbresult
##      B  M
##   B 66  5
##   M  2 40
##
##               Accuracy : 0.9381
##                 95% CI : (0.8765, 0.9747)
##    No Information Rate : 0.6018
##    P-Value [Acc > NIR] : 2.83e-16
##
##                  Kappa : 0.8693
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9706
##             Specificity : 0.8889
##          Pos Pred Value : 0.9296
##          Neg Pred Value : 0.9524
##              Prevalence : 0.6018
##          Detection Rate : 0.5841
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9297
##
##        '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 97.5
```

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.
## 564                                 1     1.9275296      1.3485941
## 565                                 1     2.1091388      0.7208383
## 566                                 1     1.7033556      2.0833009
## 567                                 1     0.7016669      2.0437755
## 568                                 1     1.8367249      2.3344032
## 569                                 0    -1.8068114      1.2207179
##     perimeter..nucA. area..nucA. smoothness..nucA. compactness..nucA.
## 564        2.1001278   1.9667039         0.9627130         2.25814785
## 565        2.0589739   2.3417954         1.0409262         0.21886787
## 566        1.6145108   1.7223261         0.1023682        -0.01781736
## 567        0.6720844   0.5774446        -0.8397450        -0.03864567
## 568        1.9807813   1.7336925         1.5244257         3.26926717
## 569       -1.8127934  -1.3466044        -3.1093489        -1.14974083
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 564       2.86755184             2.5379803       1.2306774     1.6595095
## 565       1.94557271             2.3189242      -0.3123140     1.8995140
## 566       0.69243373             1.2625583      -0.2174729     1.5353692
## 567       0.04654658             0.1056844      -0.8084058     0.5608679
## 568       3.29404559             2.6565283       2.1353154     1.9595152
## 569      -1.11389274            -1.2607103      -0.8193490    -1.4096522
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 564      0.6073251        2.1378974   1.6482047         0.3648935
## 565      0.1175962        1.7510219   2.0135291         0.3780327
## 566      2.0455987        1.4206897   1.4936444        -0.6906227
## 567      1.3736451        0.5784916   0.4275294        -0.8088756
## 568      2.2359585        2.3015755   1.6517174         1.4291692
## 569      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
```

```
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.
## 542                                 0    0.09724844      1.3253439
## 543                                 0    0.17386486      1.4253198
## 547                                 0   -1.08037429     -0.6834746
## 551                                 0   -0.92714145      0.5092614
## 556                                 0   -1.08888722      1.9344995
## 561                                 0   -0.02193265      1.8275485
##     perimeter..nucA.  area..nucA. smoothness..nucA. compactness..nucA.
## 542      0.15807110  0.004293366        -0.5681320         0.3533051
## 543      0.11239017  0.038960915        -0.9677302        -0.6097198
## 547     -1.09712557 -0.937697797        -0.1436478        -1.0300728
## 551     -0.96543280 -0.836536755        -1.5678385        -1.1753028
## 556     -1.08231013 -0.947643405        -0.4309034        -0.5256492
## 561     -0.02424107 -0.154836363         0.2083114         0.1563830
##     concavity..nucA. concave.points..nucA. symmetry..nucA. radius..nucC.
## 542        0.1517902            -0.2582065       0.2202552   -0.01017739
## 543       -0.5989643            -0.4806129       0.1035277    0.04982374
## 547       -0.9869481            -1.1190970       0.2676757   -1.03847278
## 551       -1.1138927            -1.2607103      -0.5494167   -0.95364358
## 556       -0.3613821            -0.5550919      -0.7974626   -1.12330197
## 561       -0.5541824            -0.1515133      -1.0017356   -0.20052584
##     texture..nucC. perimeter..nucC. area..nucC. smoothness..nucC.
## 542      0.9847906      0.185664147 -0.12590190        0.07145127
## 543      1.0759030      0.004130263 -0.09516547       -1.15487465
## 547     -0.6357077     -1.075549950 -0.87060163       -0.16943418
## 551     -0.1476058     -0.987461377 -0.82247717       -1.41327905
## 556      1.5021786     -1.121677412 -0.91855046        0.26415962
## 561      1.2190795     -0.210139239 -0.30540264       -0.36214254
##     compactness..nucC. concavity..nucC. concave.points..nucC. symmetry..nucC.
## 542         1.0546502        0.6318129            0.08966327       0.4626734
## 543        -0.7415002       -0.5324811           -0.07768205      -0.2889333
## 547        -1.0540787       -1.0945441           -1.38130213      -0.3552040
## 551        -1.1490344       -1.3046827           -1.74352870      -0.7156519
## 556        -0.5292164       -0.3460215           -0.35501890      -1.0906471
## 561        -0.1771048       -0.6690903           -0.14918415      -1.0518545
##     fractal.dimension..nucC.
## 542              1.01621788
## 543             -0.79650098
## 547             -0.55122473
## 551             -0.99803723
## 556             -0.06177956
## 561             -0.04074007
```

Fitting Model

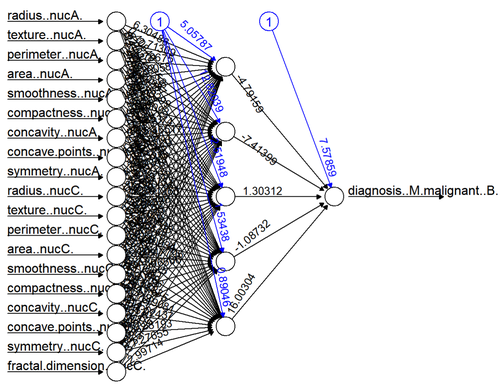
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, hidden = 5, err.fct = "ce", linear.output = FALSE, lifesign =
'full', rep = 1, algorithm = "rprop+", stepmax = 100000) -> nnmodel #fitting the model
```

```
## hidden: 5    thresh: 0.01    rep: 1/1    steps:     415  error: 6.75451  time: 0.24 secs
```

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

```
##                   Length Class      Mode
## call                10   -none-     call
## response           456   -none-     numeric
## covariate         8664   -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
## 3       1  1.0000000
## 7       1  1.0000000
## 14      1  0.8398321
## 17      1  1.0000000
## 30      1  1.0000000
## 37      1  1.0000000
```

Prediction

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

Confusion Matrix

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

```
##      prediction
## actual  0  1
##      0 68  3
##      1  1 41
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 96.46
```

```
## Confusion Matrix and Statistics
##
##      prediction
## actual  0  1
##      0 68  3
##      1  1 41
##
##               Accuracy : 0.9646
##                 95% CI : (0.9118, 0.9903)
##    No Information Rate : 0.6106
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9249
##
##  Mcnemar's Test P-Value : 0.6171
##
##            Sensitivity : 0.9855
##            Specificity : 0.9318
##         Pos Pred Value : 0.9577
##         Neg Pred Value : 0.9762
##             Prevalence : 0.6106
##         Detection Rate : 0.6018
##   Detection Prevalence : 0.6283
##      Balanced Accuracy : 0.9587
##
##       'Positive' Class : 0
##
```

Neural Network: Model 2

Fitting Model

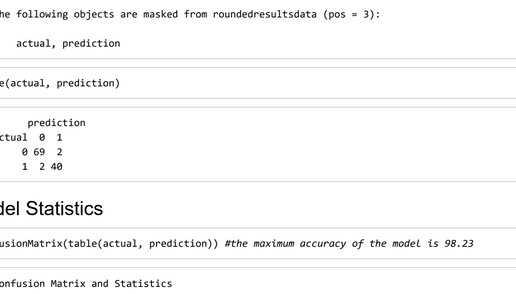
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = train, threshold = 0.03, hidden = 32, err.fct = "ce", linear.output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> nnmodel
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.384738595526611
##                                           2000 min thresh: 0.190132001385477
##                                           3000 min thresh: 0.118899045386974
##                                           4000 min thresh: 0.0841821921466113
##                                           5000 min thresh: 0.0642650126564778
##                                           6000 min thresh: 0.051557149741531
##                                           7000 min thresh: 0.0428265131971157
##                                           8000 min thresh: 0.0364963002454605
##                                           9000 min thresh: 0.0317155288403582
##                                           9432 error: 0.34377  time: 23.22 secs
```

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

```
##                   Length Class      Mode
## call                13   -none-     call
## response           456   -none-     numeric
## covariate         8664   -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
## 3       1  0.9999999
## 7       1  1.0000000
## 14      1  0.7622966
## 17      1  1.0000000
## 30      1  0.9999813
## 37      1  0.9992561
```

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 69  2
##      1  2 40
```

Model Statistics

```
confusionMatrix(table(actual, prediction)) #the maximum accuracy of the model is 98.23
```

```
## Confusion Matrix and Statistics
##
##      prediction
## actual  0  1
##      0 69  2
##      1  2 40
##
##               Accuracy : 0.9646
##                 95% CI : (0.9118, 0.9903)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##            Sensitivity : 0.9718
##            Specificity : 0.9524
##         Pos Pred Value : 0.9718
##         Neg Pred Value : 0.9524
##             Prevalence : 0.6283
##         Detection Rate : 0.6106
##   Detection Prevalence : 0.6283
##      Balanced Accuracy : 0.9621
##
##       '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 69  5
##   1  2 37
##
##               Accuracy : 0.9381
##                 95% CI : (0.8765, 0.9747)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : 1.718e-14
##
##                  Kappa : 0.8654
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9718
##             Specificity : 0.8810
##          Pos Pred Value : 0.9324
##          Neg Pred Value : 0.9487
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6549
##       Balanced Accuracy : 0.9264
##
##        'Positive' Class : 0
##
```

Decision Tree and SVM

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

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 71  5
##   1  0 37
##
##               Accuracy : 0.9558
##                 95% CI : (0.8998, 0.9855)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : < 2e-16
##
##                  Kappa : 0.9029
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.8810
##          Pos Pred Value : 0.9342
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6283
##          Detection Rate : 0.6283
##    Detection Prevalence : 0.6726
##       Balanced Accuracy : 0.9405
##
##        'Positive' Class : 0
##
```

Random Forest and SVM

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1) +
                             ifelse(svmresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the max
imum accuracy of the model is 97.35
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 71  2
##   1  0 40
##
##               Accuracy : 0.9823
##                 95% CI : (0.9375, 0.9978)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9617
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9524
##          Pos Pred Value : 0.9726
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6283
##          Detection Rate : 0.6283
##    Detection Prevalence : 0.6460
##       Balanced Accuracy : 0.9762
##
##        'Positive' Class : 0
##
```

Random Forest and Naive Bayes

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1) +
                             ifelse(nbresult %in% c("B", "B"), 0, 1))/2), test$diagnosis..M.malignant..B.benign.)) #the maxi
mum accuracy of the model is 94.69
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  3
##   1  2 39
##
##               Accuracy : 0.9558
##                 95% CI : (0.8998, 0.9855)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9048
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9286
##          Pos Pred Value : 0.9583
##          Neg Pred Value : 0.9512
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6372
##       Balanced Accuracy : 0.9502
##
##        'Positive' Class : 0
##
```

Random Forest and Neural Network

```
confusionMatrix(table(round((ifelse(rfresult %in% c("B", "B"), 0, 1)*0.90 +
                             (ifelse(nnresult %in% c("B", "B"), 0, 1)*0.90))/2), test$diagnosis..M.malignant..B.benign.)) #th
e maximum accuracy of the model is 97.35
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  2
##   1  2 40
##
##               Accuracy : 0.9646
##                 95% CI : (0.9118, 0.9903)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9524
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9524
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9621
##
##        '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 93.81
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 69  6
##   1  2 36
##
##               Accuracy : 0.9292
##                 95% CI : (0.8653, 0.9689)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : 1.372e-13
##
##                  Kappa : 0.8454
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9718
##             Specificity : 0.8571
##          Pos Pred Value : 0.9200
##          Neg Pred Value : 0.9474
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6637
##       Balanced Accuracy : 0.9145
##
##        '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 70  1
##   1  1 41
##
##               Accuracy : 0.9823
##                 95% CI : (0.9375, 0.9978)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9621
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9859
##             Specificity : 0.9762
##          Pos Pred Value : 0.9859
##          Neg Pred Value : 0.9762
##              Prevalence : 0.6283
##          Detection Rate : 0.6195
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9811
##
##        'Positive' Class : 0
##
```

Naive Bayes and Neural Network

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

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 66  2
##   1  5 40
##
##               Accuracy : 0.9381
##                 95% CI : (0.8765, 0.9747)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : 1.718e-14
##
##                  Kappa : 0.8693
##
##  Mcnemar's Test P-Value : 0.4497
##
##            Sensitivity : 0.9296
##            Specificity : 0.9524
##         Pos Pred Value : 0.9706
##         Neg Pred Value : 0.8889
##             Prevalence : 0.6283
##         Detection Rate : 0.5841
##   Detection Prevalence : 0.6018
##      Balanced Accuracy : 0.9410
##
##       '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 99.12
```

```
## Confusion Matrix and Statistics
##
##
##      0  1
##   0 68  1
##   1  3 41
##
##               Accuracy : 0.9646
##                 95% CI : (0.9118, 0.9903)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9249
##
##  Mcnemar's Test P-Value : 0.6171
##
##            Sensitivity : 0.9577
##            Specificity : 0.9762
##         Pos Pred Value : 0.9855
##         Neg Pred Value : 0.9318
##             Prevalence : 0.6283
##         Detection Rate : 0.6018
##   Detection Prevalence : 0.6106
##      Balanced Accuracy : 0.9670
##
##       '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

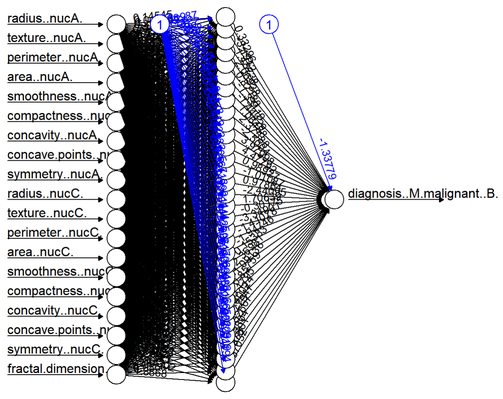
```
neuralnet(diagnosis..M.malignant..B.benign. ~., data = ensembletrain, threshold = 0.03, hidden = 32, err.fct = "ce", linear.
output = FALSE, lifesign = 'full',
  act.fct = "logistic",rep = 1, algorithm = "backprop", learningrate = 0.003, stepmax = 100000) -> ensemblemodel #fitting th
e model
```

```
## hidden: 32    thresh: 0.03    rep: 1/1    steps:    1000 min thresh: 0.20132347352197
##                                                     2000 min thresh: 0.094199903221874
##                                                     3000 min thresh: 0.0598013595579179
##                                                     4000 min thresh: 0.0433536042060217
##                                                     5000 min thresh: 0.0338131874543704
##                                                     5569 error: 0.15364  time: 17.11 secs
```

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

```
##                     Length Class      Mode
## call                    13 -none-     call
## response               456 -none-     numeric
## covariate             8664 -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
## 3       1  1.0000000
## 7       1  1.0000000
## 14      1  0.9816670
## 17      1  0.9999998
## 30      1  0.9989465
## 37      1  0.9999751
```

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 71  2
##      1  0 40
##
##               Accuracy : 0.9823
##                 95% CI : (0.9375, 0.9978)
##    No Information Rate : 0.6283
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.9617
##
##  Mcnemar's Test P-Value : 0.4795
##
##            Sensitivity : 1.0000
##            Specificity : 0.9524
##         Pos Pred Value : 0.9726
##         Neg Pred Value : 1.0000
##             Prevalence : 0.6283
##         Detection Rate : 0.6283
##   Detection Prevalence : 0.6460
##      Balanced Accuracy : 0.9762
##
##       'Positive' Class : 0
##
```