Libray

# Load libraries  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

library(funModeling)

## Loading required package: Hmisc

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

## funModeling v.1.9.4 :)  
## Examples and tutorials at livebook.datascienceheroes.com  
## / Now in Spanish: librovivodecienciadedatos.ai

library(tidyverse)

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

## √ tibble 3.0.2 √ dplyr 1.0.0  
## √ tidyr 1.1.0 √ stringr 1.4.0  
## √ readr 1.3.1 √ forcats 0.5.0  
## √ purrr 0.3.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::combine() masks randomForest::combine()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::lift() masks caret::lift()  
## x randomForest::margin() masks ggplot2::margin()  
## x dplyr::src() masks Hmisc::src()  
## x dplyr::summarize() masks Hmisc::summarize()

library(readr)  
library(GA)

## Loading required package: foreach

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loading required package: iterators

## Package 'GA' version 3.2  
## Type 'citation("GA")' for citing this R package in publications.

##   
## Attaching package: 'GA'

## The following object is masked from 'package:utils':  
##   
## de

Data

# Load Data  
data=read\_delim("wdbc\_data.csv", delim = ",")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## diagnosis = col\_character()  
## )

## See spec(...) for full column specifications.

summary(data)

## id diagnosis radius\_mean texture\_mean   
## Min. : 8670 Length:569 Min. : 6.981 Min. : 9.71   
## 1st Qu.: 869218 Class :character 1st Qu.:11.700 1st Qu.:16.17   
## Median : 906024 Mode :character Median :13.370 Median :18.84   
## Mean : 30371831 Mean :14.127 Mean :19.29   
## 3rd Qu.: 8813129 3rd Qu.:15.780 3rd Qu.:21.80   
## Max. :911320502 Max. :28.110 Max. :39.28   
## perimeter\_mean area\_mean smoothness\_mean compactness\_mean   
## 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\_mean concave points\_mean symmetry\_mean fractal\_dimension\_mean  
## Min. :0.00000 Min. :0.00000 Min. :0.1060 Min. :0.04996   
## 1st Qu.:0.02956 1st Qu.:0.02031 1st Qu.:0.1619 1st Qu.:0.05770   
## Median :0.06154 Median :0.03350 Median :0.1792 Median :0.06154   
## Mean :0.08880 Mean :0.04892 Mean :0.1812 Mean :0.06280   
## 3rd Qu.:0.13070 3rd Qu.:0.07400 3rd Qu.:0.1957 3rd Qu.:0.06612   
## Max. :0.42680 Max. :0.20120 Max. :0.3040 Max. :0.09744   
## radius\_se texture\_se perimeter\_se area\_se   
## Min. :0.1115 Min. :0.3602 Min. : 0.757 Min. : 6.802   
## 1st Qu.:0.2324 1st Qu.:0.8339 1st Qu.: 1.606 1st Qu.: 17.850   
## Median :0.3242 Median :1.1080 Median : 2.287 Median : 24.530   
## Mean :0.4052 Mean :1.2169 Mean : 2.866 Mean : 40.337   
## 3rd Qu.:0.4789 3rd Qu.:1.4740 3rd Qu.: 3.357 3rd Qu.: 45.190   
## Max. :2.8730 Max. :4.8850 Max. :21.980 Max. :542.200   
## smoothness\_se compactness\_se concavity\_se concave points\_se   
## Min. :0.001713 Min. :0.002252 Min. :0.00000 Min. :0.000000   
## 1st Qu.:0.005169 1st Qu.:0.013080 1st Qu.:0.01509 1st Qu.:0.007638   
## Median :0.006380 Median :0.020450 Median :0.02589 Median :0.010930   
## Mean :0.007041 Mean :0.025478 Mean :0.03189 Mean :0.011796   
## 3rd Qu.:0.008146 3rd Qu.:0.032450 3rd Qu.:0.04205 3rd Qu.:0.014710   
## Max. :0.031130 Max. :0.135400 Max. :0.39600 Max. :0.052790   
## symmetry\_se fractal\_dimension\_se radius\_worst texture\_worst   
## Min. :0.007882 Min. :0.0008948 Min. : 7.93 Min. :12.02   
## 1st Qu.:0.015160 1st Qu.:0.0022480 1st Qu.:13.01 1st Qu.:21.08   
## Median :0.018730 Median :0.0031870 Median :14.97 Median :25.41   
## Mean :0.020542 Mean :0.0037949 Mean :16.27 Mean :25.68   
## 3rd Qu.:0.023480 3rd Qu.:0.0045580 3rd Qu.:18.79 3rd Qu.:29.72   
## Max. :0.078950 Max. :0.0298400 Max. :36.04 Max. :49.54   
## perimeter\_worst area\_worst smoothness\_worst compactness\_worst  
## Min. : 50.41 Min. : 185.2 Min. :0.07117 Min. :0.02729   
## 1st Qu.: 84.11 1st Qu.: 515.3 1st Qu.:0.11660 1st Qu.:0.14720   
## Median : 97.66 Median : 686.5 Median :0.13130 Median :0.21190   
## Mean :107.26 Mean : 880.6 Mean :0.13237 Mean :0.25427   
## 3rd Qu.:125.40 3rd Qu.:1084.0 3rd Qu.:0.14600 3rd Qu.:0.33910   
## Max. :251.20 Max. :4254.0 Max. :0.22260 Max. :1.05800   
## concavity\_worst concave points\_worst symmetry\_worst fractal\_dimension\_worst  
## Min. :0.0000 Min. :0.00000 Min. :0.1565 Min. :0.05504   
## 1st Qu.:0.1145 1st Qu.:0.06493 1st Qu.:0.2504 1st Qu.:0.07146   
## Median :0.2267 Median :0.09993 Median :0.2822 Median :0.08004   
## Mean :0.2722 Mean :0.11461 Mean :0.2901 Mean :0.08395   
## 3rd Qu.:0.3829 3rd Qu.:0.16140 3rd Qu.:0.3179 3rd Qu.:0.09208   
## Max. :1.2520 Max. :0.29100 Max. :0.6638 Max. :0.20750

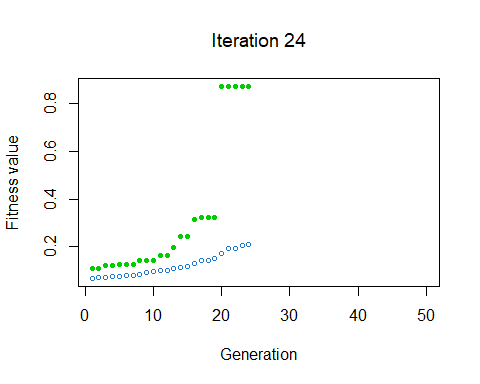
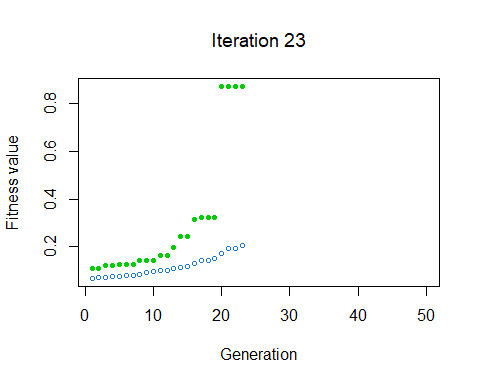
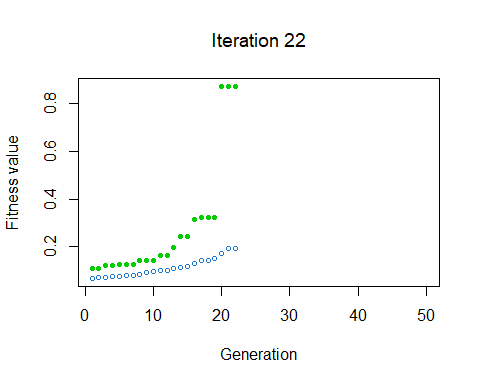
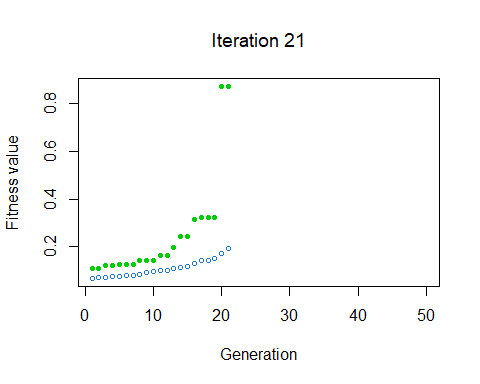
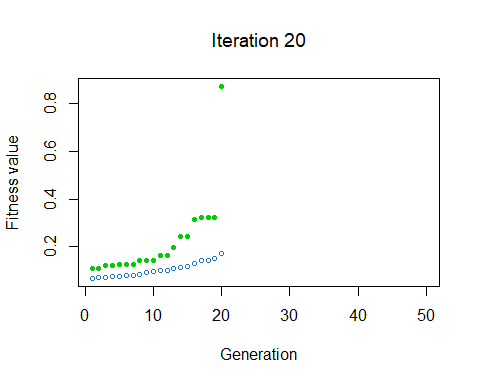
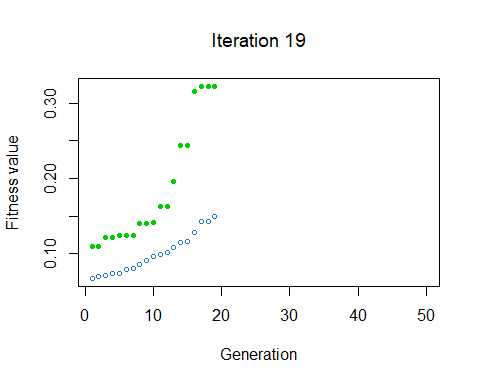
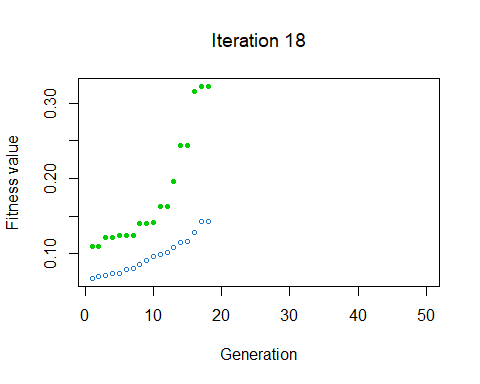
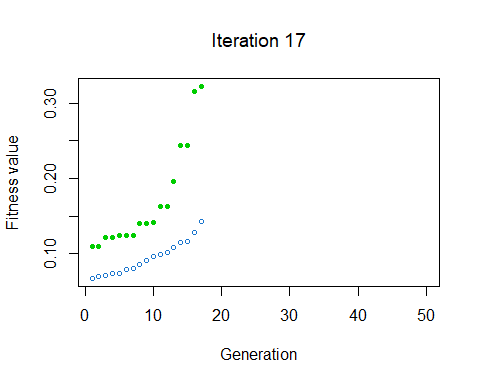
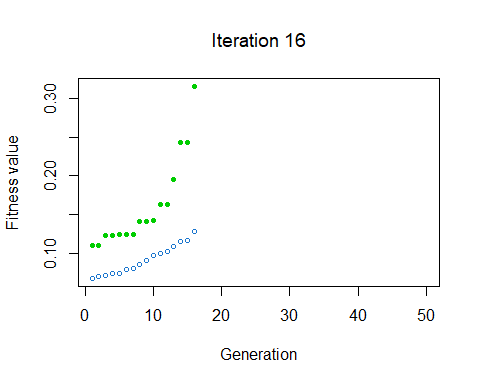
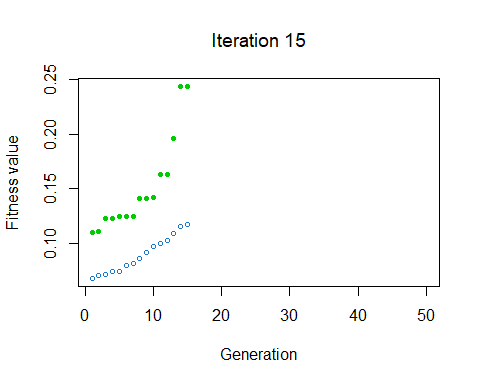
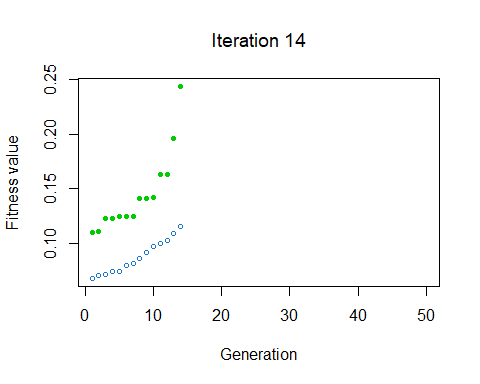
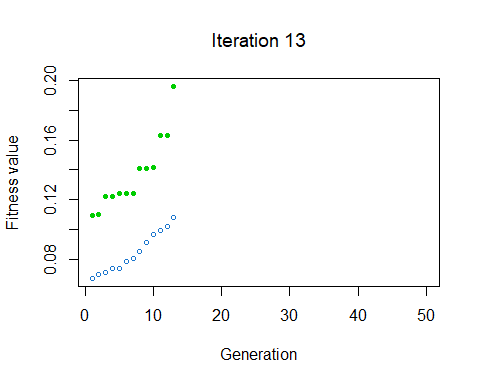
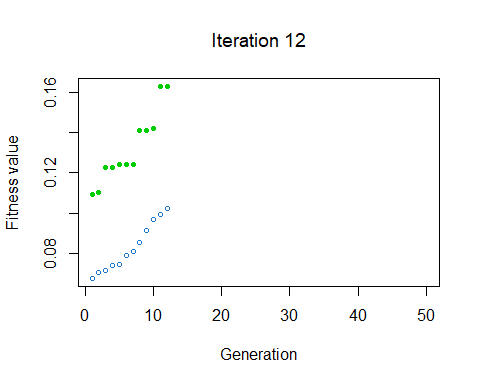
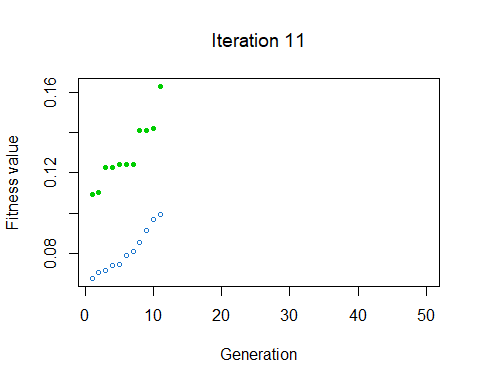
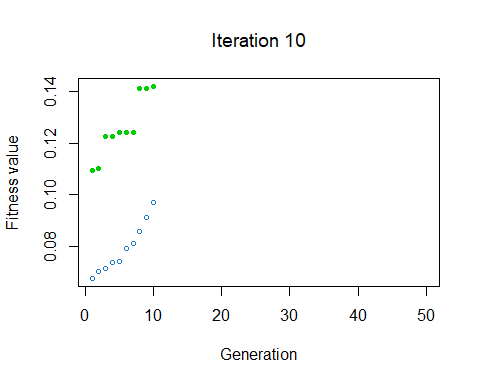
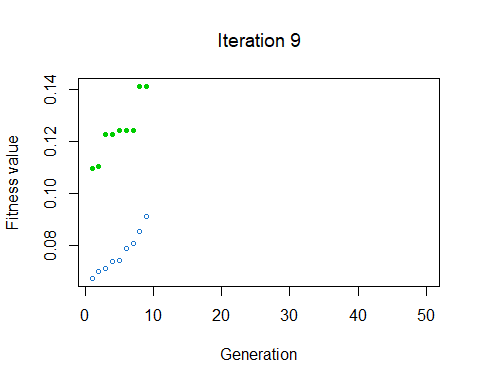
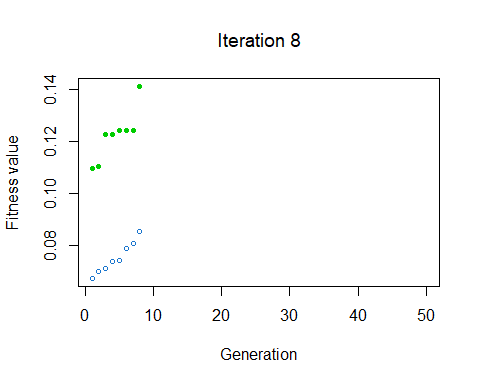
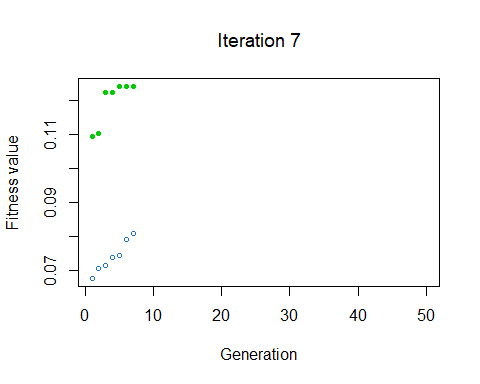
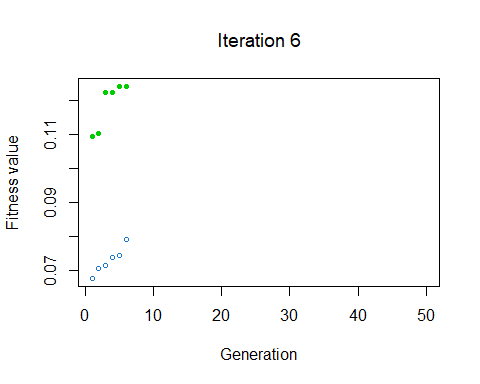
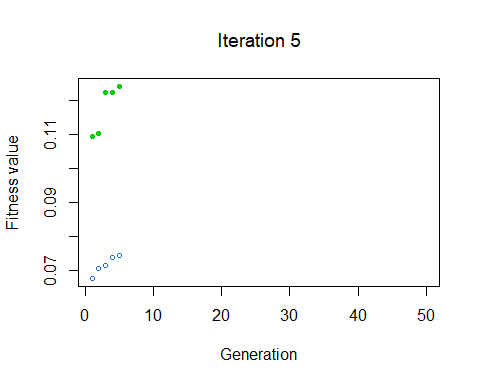
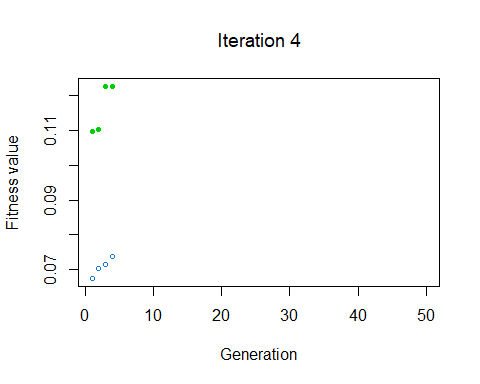
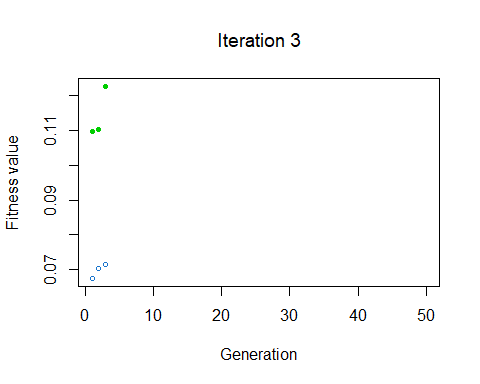
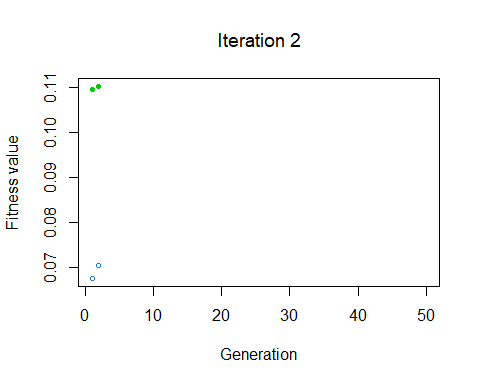
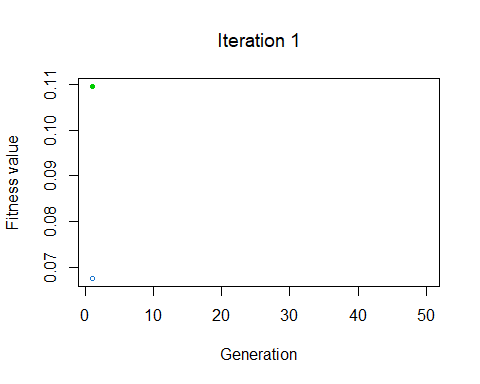
# Data preparation  
data2=na.omit(data) # remove NA rows  
  
data\_y=as.factor(data2$diagnosis) # Y as factor B/ M  
data\_x=select(data2, -diagnosis, -id) # X all Numeric

Fitness Function

# Accuracy Metric (Fitness function)  
custom\_fitness <- function(vars, data\_x, data\_y, p\_sampling)  
{  
 # speeding up things with sampling  
 ix=get\_sample(data\_x, percentage\_tr\_rows = p\_sampling)  
 data\_2=data\_x[ix,]  
 data\_y\_smp=data\_y[ix]  
   
 # keep only vars from current solution  
 names=colnames(data\_2)  
 names\_2=names[vars==1]  
 # get the columns of the current solution  
 data\_sol=data\_2[, names\_2]  
   
 # get the roc value from the created model  
 roc\_value=get\_roc\_metric(data\_sol, data\_y\_smp, names\_2)  
   
 # get the total number of vars for the current selection  
 q\_vars=sum(vars)  
   
 # time for your magic  
 fitness\_value=roc\_value/q\_vars  
   
 return(fitness\_value)  
}  
  
get\_roc\_metric <- function(data\_tr\_sample, target, best\_vars)   
{  
 # data\_tr\_sample=data\_sol  
 # target = target\_var\_s  
 # best\_vars=names\_2  
   
 fitControl <- trainControl(method = "cv",   
 number = 3,   
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
   
 data\_model=select(data\_tr\_sample, one\_of(best\_vars))  
   
 mtry = sqrt(ncol(data\_model))  
 tunegrid = expand.grid(.mtry=round(mtry))  
   
 fit\_model\_1 = train(x=data\_model,   
 y= target,   
 method = "rf",   
 trControl = fitControl,  
 metric = "ROC",  
 tuneGrid=tunegrid  
 )  
   
 metric=fit\_model\_1$results["ROC"][1,1]  
   
 return(metric)  
}  
  
get\_accuracy\_metric <- function(data\_tr\_sample, target, best\_vars)   
{  
 data\_model=select(data\_tr\_sample, one\_of(best\_vars))  
   
 fitControl <- trainControl(method = "cv",   
 number = 3,   
 summaryFunction = twoClassSummary)  
   
 data\_model=select(data\_tr\_sample, one\_of(best\_vars))  
   
 mtry = sqrt(ncol(data\_model))  
 tunegrid = expand.grid(mtry=round(mtry))  
   
 fit\_model\_1 = train(x=data\_model,   
 y= target,   
 method = "rf",  
 tuneGrid = tunegrid)  
   
   
   
 metric=fit\_model\_1$results["Accuracy"][1,1]  
 return(metric)  
}

GA setting

# GA parameters  
param\_nBits=ncol(data\_x)  
col\_names=colnames(data\_x)  
  
# Executing the GA   
ga\_GA\_1 = ga(fitness = function(vars) custom\_fitness(vars = vars,   
 data\_x = data\_x,   
 data\_y = data\_y,   
 p\_sampling = 0.7), # custom fitness function (fitness\_value=roc\_value/q\_vars)  
 type = "binary", # optimization data type  
 crossover=gabin\_uCrossover, # cross-over method  
 elitism = 3, # number of best ind. to pass to next iteration  
 pmutation = 0.03, # mutation rate prob  
 popSize = 50, # the number of indivduals/solutions  
 nBits = param\_nBits, # total number of variables (ncol(data\_x))  
 names=col\_names, # variable name  
 run=5, # max iter without improvement (stopping criteria)  
 maxiter = 50, # total runs or generations  
 monitor=plot, # plot the result at each iteration  
 keepBest = TRUE, # keep the best solution at the end  
 parallel = T, # allow parallel procesing  
 seed=84211 # for reproducibility purposes  
)



Result

# Checking the results  
summary(ga\_GA\_1)

## -- Genetic Algorithm -------------------   
##   
## GA settings:   
## Type = binary   
## Population size = 50   
## Number of generations = 50   
## Elitism = 3   
## Crossover probability = 0.8   
## Mutation probability = 0.03   
##   
## GA results:   
## Iterations = 24   
## Fitness function value = 0.8726041   
## Solution =   
## radius\_mean texture\_mean perimeter\_mean area\_mean smoothness\_mean  
## [1,] 0 0 0 0 0  
## compactness\_mean concavity\_mean concave points\_mean symmetry\_mean  
## [1,] 0 0 0 0  
## fractal\_dimension\_mean ... symmetry\_worst fractal\_dimension\_worst  
## [1,] 0 0 0

# Following line will return the variable names of the final and BEST solution  
best\_vars\_ga=col\_names[ga\_GA\_1@solution[1,]==1]  
  
# Checking the variables of the best solution...  
best\_vars\_ga

## [1] "area\_se"

# Checking the accuracy  
get\_accuracy\_metric(data\_tr\_sample = data\_x, target = data\_y, best\_vars\_ga)

## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.

## [1] 0.793973