The most accurate traditional method for diagnostic is a rather invasive technique, called breast biopsy, where a small piece of breast tissue is surgically removed, and then the tissue sample has to be examined by specialist. However, a much less invasive technique can be used, where the samples can be obtained by a minimally invasive fine needle aspirate method. The sample obtained by this method can be easily digitized and used for computationally based diagnostic. Using machine learning methods for diagnostic can significantly increase processing speed and on a big scale can make the diagnostic significantly cheaper.

```
> View(cancer)
> # Loading the libraries
> library(caTools)
> library(ggplot2)
> library(dplyr)
> library(caret)
> library(e1071)
> library(corrplot)
> library (ROCR)
> # Getting the structure of the dataset
> str(cancer)
'data.frame':	569 obs. of  31 variables:
 $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
 $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
 $ area_mean              : num  1001 1326 1203 386 1297 ...
 $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
 $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
 $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
 $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
 $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
 $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
 $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
 $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
 $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
 $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
```

```
$ smoothness_se         : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
$ compactness_se        : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
$ concavity_se          : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
$ concave.points_se     : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
$ symmetry_se           : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
$ fractal_dimension_se  : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
$ radius_worst          : num  25.4 25 23.6 14.9 22.5 ...
$ texture_worst         : num  17.3 23.4 25.5 26.5 16.7 ...
$ perimeter_worst       : num  184.6 158.8 152.5 98.9 152.2 ...
$ area_worst            : num  2019 1956 1709 568 1575 ...
$ smoothness_worst      : num  0.162 0.124 0.144 0.21 0.137 ...
$ compactness_worst     : num  0.666 0.187 0.424 0.866 0.205 ...
$ concavity_worst       : num  0.712 0.242 0.45 0.687 0.4 ...
$ concave.points_worst  : num  0.265 0.186 0.243 0.258 0.163 ...
$ symmetry_worst        : num  0.46 0.275 0.361 0.664 0.236 ...
$ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
> ncol(cancer)
[1] 31
> # Checking the NA values
> sum(is.na(heart))
[1] 0
> # Removing the X column as no valid data,
> cancer[,c(1,33)]=NULL
> # Converting the character variable in to Factor type
> cancer$diagnosis=as.factor(cancer$diagnosis)
> summary(cancer)
diagnosis  radius_mean     texture_mean   perimeter_mean     area_mean
B:357     Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5
M:212     1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3
          Median :13.370   Median :18.84   Median : 86.24   Median : 551.1
          Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9
          3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7
          Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0
```

```
smoothness_mean   compactness_mean  concavity_mean    concave.points_mean symmetry_mean
Min.   :0.05263   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000     Min.   :0.1060
1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619
Median :0.09587   Median :0.09263   Median :0.06154   Median :0.03350     Median :0.1792
Mean   :0.09636   Mean   :0.10434   Mean   :0.08880   Mean   :0.04892     Mean   :0.1812
3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957
Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.   :0.20120     Max.   :0.3040
fractal_dimension_mean   radius_se       texture_se      perimeter_se      area_se
Min.   :0.04996       Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
1st Qu.:0.05770       1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
Median :0.06154       Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
Mean   :0.06280       Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
3rd Qu.:0.06612       3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
Max.   :0.09744       Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
smoothness_se     compactness_se     concavity_se     concave.points_se   symmetry_se
Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000   Min.   :0.007882
1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160
Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930   Median :0.018730
Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796   Mean   :0.020542
3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480
Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790   Max.   :0.078950
fractal_dimension_se  radius_worst   texture_worst   perimeter_worst    area_worst
Min.   :0.0008948    Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
Median :0.0031870    Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
Mean   :0.0037949    Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
Max.   :0.0298400    Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
smoothness_worst  compactness_worst concavity_worst  concave.points_worst symmetry_worst
Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000     Min.   :0.1565
1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493     1st Qu.:0.2504
Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993     Median :0.2822
Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461     Mean   :0.2901
```

```
3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140     3rd Qu.:0.3179

Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100     Max.   :0.6638

fractal_dimension_worst

Min.   :0.05504

1st Qu.:0.07146

Median :0.08004

Mean   :0.08395

3rd Qu.:0.09208

Max.   :0.20750
```

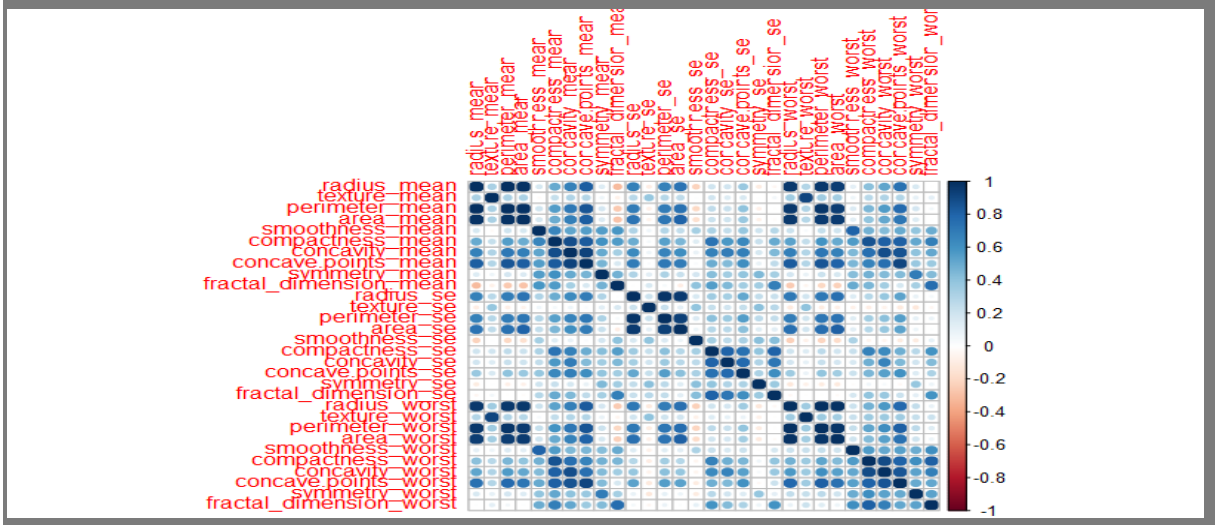
```
> # There is imbalance between data so there is high amount of imbalance between data

> table(cancer$diagnosis)
```

```
B   M

357 212
```

```
> pie(table(cancer$diagnosis))

> data_corelation=cancer[-1]

> cor(data_corelation)
```

#The plot shows high amount of correlation between variables

```
corrplot(cor(data_corelation))
```




```
> #Modeling

>
```

```
> #We are going to get a training and a testing set to use when building some models:

> set.seed(101)
```

```
> attach(cancer)
> split=sample.split(cancer$diagnosis,SplitRatio = 0.70)
> train=subset(cancer,split==T)
> test=subset(cancer,split==F)
> # Feature scaling
> train[-1]=scale(train[-1])
> test[-1]=scale(test[-1])
> # Now creating the logistic regression model 1
> model_reg=glm(diagnosis~.,train, family = 'binomial')
> summary(model_reg)
Call:
glm(formula = diagnosis ~ ., family = "binomial", data = train)


Deviance Residuals:
Min        1Q     Median        3Q       Max
-2.352e-04  -2.100e-08  -2.100e-08   2.100e-08   2.360e-04


Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept)            1.396e+02  2.353e+04   0.006    0.995
radius_mean           -3.263e+03  9.749e+05  -0.003    0.997
texture_mean           9.433e+01  1.429e+04   0.007    0.995
perimeter_mean         1.288e+03  1.068e+06   0.001    0.999
area_mean              1.915e+03  3.793e+05   0.005    0.996
smoothness_mean        1.528e+02  3.445e+04   0.004    0.996
compactness_mean      -4.991e+02  1.453e+05  -0.003    0.997
concavity_mean         3.245e+02  7.055e+04   0.005    0.996
concave.points_mean    3.977e+01  5.896e+04   0.001    0.999
symmetry_mean         -1.664e+01  1.756e+04  -0.001    0.999
fractal_dimension_mean  7.827e+01  6.333e+04   0.001    0.999
radius_se              4.954e+02  1.359e+05   0.004    0.997
texture_se             3.206e+00  1.728e+04   0.000    1.000
perimeter_se          -3.393e+02  8.800e+04  -0.004    0.997
```

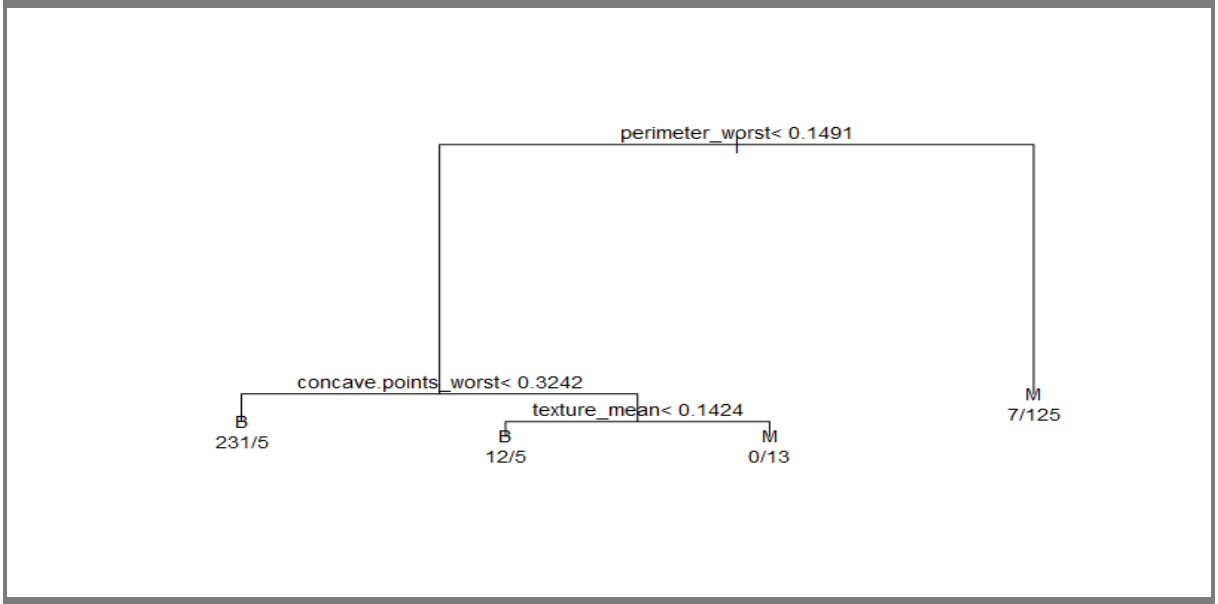
```
area_se                -3.094e+02  1.224e+05  -0.003    0.998
smoothness_se           2.770e+00  2.125e+04   0.000    1.000
compactness_se          1.425e+02  3.703e+04   0.004    0.997
concavity_se           -2.467e+02  7.549e+04  -0.003    0.997
concave.points_se       1.471e+02  4.353e+04   0.003    0.997
symmetry_se            -1.073e+01  3.375e+04   0.000    1.000
fractal_dimension_se   -1.912e+02  4.504e+04  -0.004    0.997
radius_worst            6.149e+02  2.767e+05   0.002    0.998
texture_worst          -1.653e+01  1.718e+04  -0.001    0.999
perimeter_worst         5.215e+02  1.735e+05   0.003    0.998
area_worst             -3.952e+02  3.124e+05  -0.001    0.999
smoothness_worst       -8.393e+01  3.654e+04  -0.002    0.998
compactness_worst       1.155e+02  9.173e+04   0.001    0.999
concavity_worst         6.515e+01  8.160e+04   0.001    0.999
concave.points_worst   -2.776e+01  3.845e+04  -0.001    0.999
symmetry_worst          5.669e+01  2.696e+04   0.002    0.998
fractal_dimension_worst 5.121e+01  5.084e+04   0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 5.2531e+02  on 397  degrees of freedom
Residual deviance: 6.5257e-07  on 367  degrees of freedom
AIC: 62

Number of Fisher Scoring iterations: 25

> # Getting the summary of the model
> prediction=predict(model_reg,test,family='response')
> thresshold=ifelse(prediction>0.5,'M','B')
> # setting a threshold amount of 0.5
> # Testing of test dataset the accuracy of the model
> confusionMatrix(data=as.factor(thresshold),reference = test$diagnosis)
Confusion Matrix and Statistics
```

```
Reference
Prediction  B  M
B 96  2
M 11 62

Accuracy : 0.924
95% CI : (0.8735, 0.9589)
No Information Rate : 0.6257
P-Value [Acc > NIR] : <2e-16

Kappa : 0.8422

Mcnemar's Test P-Value : 0.0265

Sensitivity : 0.8972
Specificity : 0.9688
Pos Pred Value : 0.9796
Neg Pred Value : 0.8493
Prevalence : 0.6257
Detection Rate : 0.5614
Detection Prevalence : 0.5731
Balanced Accuracy : 0.9330

'Positive' Class : B

> #In this model the accuracy we find to be 92%
>
> # Second model : SVM
> model_svm=svm(diagnosis~.,data=train,type='C-classification',kernel='linear')
> prediction=predict(model_svm,newdata = test)
> confusionMatrix(table(prediction,test$diagnosis))
Confusion Matrix and Statistics
```

```
prediction   B   M
B 106   4
M   1  60

Accuracy : 0.9708
95% CI : (0.9331, 0.9904)
No Information Rate : 0.6257
P-Value [Acc > NIR] : <2e-16

Kappa : 0.937

Mcnemar's Test P-Value : 0.3711

Sensitivity : 0.9907
Specificity : 0.9375
Pos Pred Value : 0.9636
Neg Pred Value : 0.9836
Prevalence : 0.6257
Detection Rate : 0.6199
Detection Prevalence : 0.6433
Balanced Accuracy : 0.9641

'Positive' Class : B

>
> # It has a accuracy of 97% which is quite good
>
> # Third model: Naive Bayes
>
> model_naive=naiveBayes(diagnosis~.,data=train)
> # Checking the accuracy
```

```
> prediction=predict(model_naive,newdata = test)
>
> confusionMatrix(table(prediction,test$diagnosis))
Confusion Matrix and Statistics


prediction   B   M
B 101   5
M   6  59

Accuracy : 0.9357
95% CI : (0.8878, 0.9675)
No Information Rate : 0.6257
P-Value [Acc > NIR] : <2e-16

Kappa : 0.8631

Mcnemar's Test P-Value : 1

Sensitivity : 0.9439
Specificity : 0.9219
Pos Pred Value : 0.9528
Neg Pred Value : 0.9077
Prevalence : 0.6257
Detection Rate : 0.5906
Detection Prevalence : 0.6199
Balanced Accuracy : 0.9329

'Positive' Class : B

>
> # Accuracy of the model is 92%
>
```

```
> # Forth Model: Decision tree
>
> library(rpart)
> model_tree=rpart(diagnosis~.,data=train)
> plot(model_tree,margin=0.1)
> text(model_tree,use.n = T,pretty = T,cex=0.9)
```

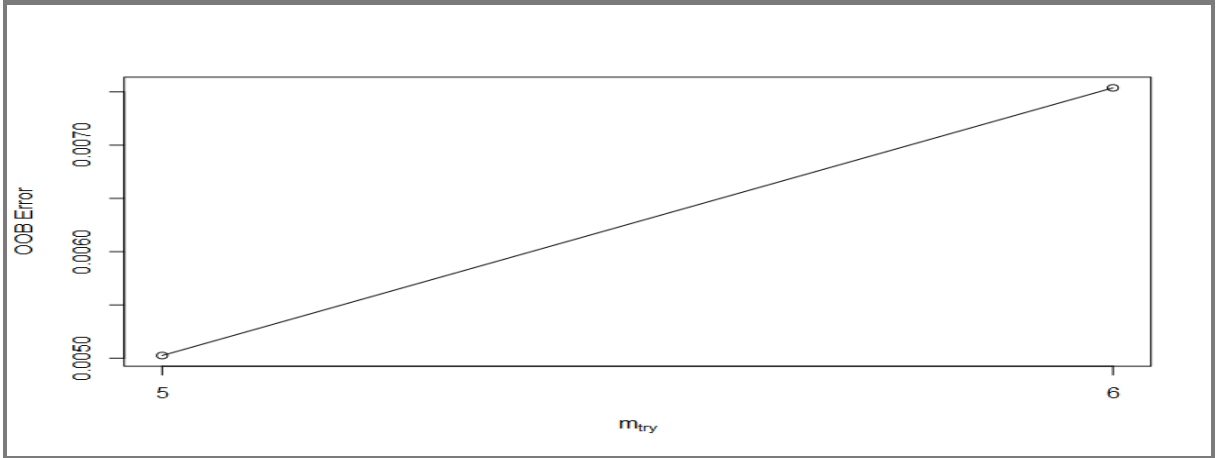



```
> # Checking the accuracy
> prediction=predict(model_tree,test, type='class')
> confusionMatrix(prediction,test$diagnosis)
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 101   5
         M   6  59

               Accuracy : 0.9357
                 95% CI : (0.8878, 0.9675)
    No Information Rate : 0.6257
    P-Value [Acc > NIR] : <2e-16
```

```
Kappa : 0.8631


Mcnemar's Test P-Value : 1


Sensitivity : 0.9439

Specificity : 0.9219

Pos Pred Value : 0.9528

Neg Pred Value : 0.9077

Prevalence : 0.6257

Detection Rate : 0.5906

Detection Prevalence : 0.6199

Balanced Accuracy : 0.9329


'Positive' Class : B


>
> # Fifth model: Random forest
> # Checking the best N value
> best_n_value=tuneRF(train,train$diagnosis,stepFactor = 1.2,improve = 0.1,trace = T,plot=T)
mtry = 5  OOB error = 0.5%
Searching left ...
Searching right ...
mtry = 6             OOB error = 0.25%
0.5 0.1
mtry = 7             OOB error = 0%
1 0.1
mtry = 8             OOB error = 0%
NaN 0.1
Error in if (Improve > improve) { : missing value where TRUE/FALSE needed
> best_n_value
mtry    OOBError
5.OOB    5 0.002512563
6.OOB    6 0.000000000
```

7.OOB 7 0.005025126

>



> # building the model

>

> library(randomForest)

> model_forest=randomForest(diagnosis~.,data=train)

>

> # Checking the accuracy

> prediction=predict(model_forest,newdata = test)

> confusionMatrix(table(prediction,test$diagnosis))

Confusion Matrix and Statistics

prediction B M

B 104 3

M 3 61

Accuracy : 0.9649

95% CI : (0.9252, 0.987)

No Information Rate : 0.6257

P-Value [Acc > NIR] : <2e-16

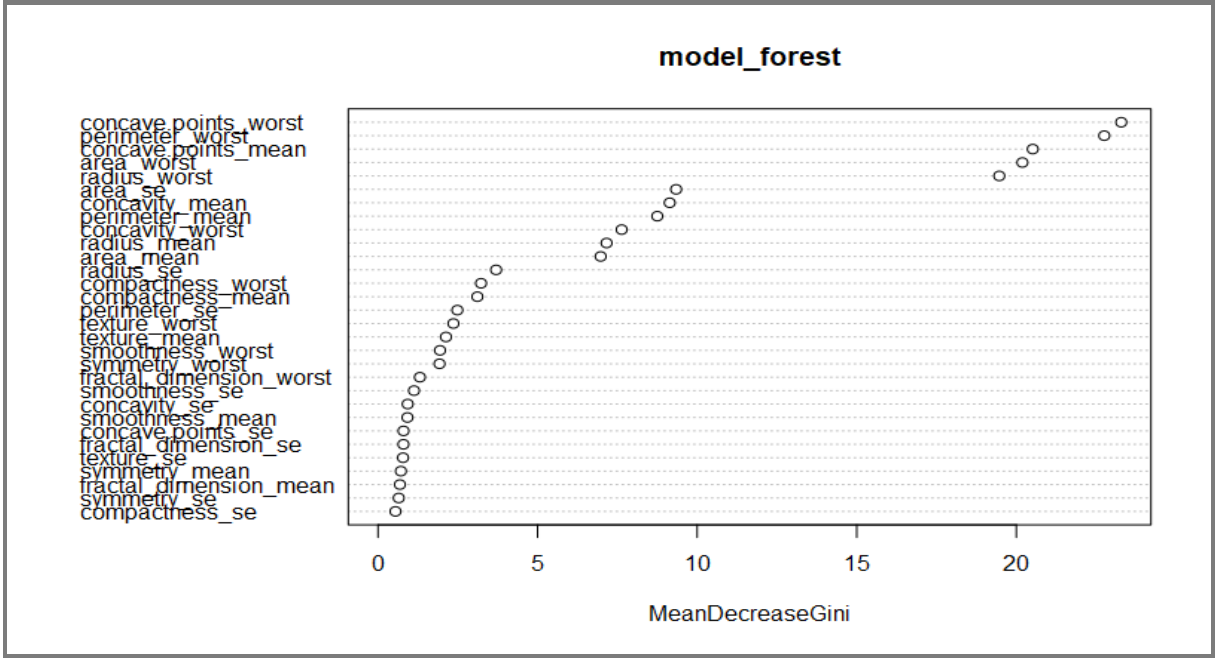
Kappa : 0.9251

Mcnemar's Test P-Value : 1

```
Sensitivity : 0.9720
Specificity : 0.9531
Pos Pred Value : 0.9720
Neg Pred Value : 0.9531
Prevalence : 0.6257
Detection Rate : 0.6082
Detection Prevalence : 0.6257
Balanced Accuracy : 0.9625

'Positive' Class : B

>
> # No checking the priority of variable
>
> importance(model_forest)
                    MeanDecreaseGini
radius_mean                7.1624807
texture_mean               2.1299918
perimeter_mean             8.7511024
area_mean                  6.9776281
smoothness_mean            0.9227874
compactness_mean           3.1137205
concavity_mean             9.1398374
concave.points_mean       20.5152859
symmetry_mean              0.7173783
fractal_dimension_mean     0.6872670
radius_se                  3.7005375
texture_se                 0.7837229
perimeter_se               2.4872109
area_se                    9.3415616
smoothness_se              1.1281511
compactness_se             0.5470751
```

```
concavity_se              0.9304822

concave.points_se         0.7996469

symmetry_se               0.6504623

fractal_dimension_se      0.7969123

radius_worst             19.4666463

texture_worst             2.3627693

perimeter_worst          22.7555534

area_worst               20.1908056

smoothness_worst          1.9436956

compactness_worst         3.2294886

concavity_worst           7.6347205

concave.points_worst     23.2880487

symmetry_worst            1.9283409

fractal_dimension_worst   1.3122365
```

```
> varImpPlot(model_forest)

>
```




```
> # Accuracy measure

> model_name=c('log regression','svm','naivebyes','decisontree', 'random forest')

> model_name=as.data.frame(model_name)

> # Model vs Accuracy percentage

> accuracy_percentage=c(92,97,93,94,95)
```

```
> cbind(model_name,accuracy_percentage)
     model_name accuracy_percentage
1 log regression                 92
2           svm                 97
3     naivebyes                 93
4    decisontree                 94
5  random forest                 95
> # From this table we can see that svm and random forest have the highest accuracy percentage.
```