Bayes' Theorem

Example : Telecom Customers

- A telecom firm has many customers. Each customer either talks for the duration of more than 100 minutes or less than 100 minutes. The firm has launched a plan for the customers who talk more specially to optimize the amount spent by them on bills.
- Call Centre staff had been instructed to call some customers. In that operation, some customers bought the new plan and others didn't.
- In this case each customer is a record, and the response of interest, Y = {Bought ,Not Bought}, has two classes: C1 = Bought and C2 = Not Bought.

Conditional Probabilities

- A conditional probability of event A given event B [denoted by P(A|B)] represents the chances of event A occurring only under the scenario that event B occurs.
- In the response example, we may be interested in P(bought| Talk Time >=100, gender=Male), also P(bought| Talk Time >=100, gender=Female), as we have gender as additional feature of the customers

BAYES FORMULA

- The Bayes theorem gives us the following formula to compute the probability that the record belongs to class Ci:

$$P(C_i|X_1,\ldots,X_p) = \frac{P(X_1,\ldots,X_p|C_i)P(C_i)}{P(X_1,\ldots,X_p|C_1)P(C_1)+\cdots+P(X_1,\ldots,X_p|C_m)P(C_m)}.$$

Where

Ci : classes of interest

$X_1, X_2, \ldots X_p$: Variables which co-exist with Classes of interest

Example

Talks for more than 100 min? (TT >= 100)	Gender	Response
y	male	not bought
n	male	not bought
n	female	not bought
n	female	not bought
n	male	not bought
n	male	not bought
y	male	bought
y	female	bought
n	female	bought
y	female	bought

Bayes' Formula Calculations

$P(Buy|Male, TT \geq 100)$

$$= \frac{P(Male, TT \geq 100 \mid Buy)\ P(Buy)}{P(Male, TT \geq 100 \mid Buy)\ P(Buy) + P(Male, TT \geq 100 \mid Not\ Buy)\ P(\ Not\ Buy)}$$

$$= \frac{P(Male|Buy)P(TT \geq 100|Buy)\ P(Buy)}{P(Male|Buy)P(TT \geq 100|Buy)P(Buy) + P(Male|\ Not\ Buy)P(TT \geq 100|Not\ Buy)P(Not\ Buy)}$$

$$= \frac{\frac{1}{4} \times \frac{3}{4} \times \frac{4}{10}}{\frac{1}{4} \times \frac{3}{4} \times \frac{4}{10} + \frac{4}{6} \times \frac{1}{6} \times \frac{6}{10}}$$

$= 0.529$

(TT >= 100)	Gender	Response
y	male	not bought
n	male	not bought
n	female	not bought
n	female	not bought
n	male	not bought
n	male	not bought
y	male	bought
y	female	bought
n	female	bought
y	female	bought

Bayes Probabilities

- For the conditional probability of bought behaviors given (TT >= 100) = y, gender = male , the numerator is a multiplication of the proportion of (TT >= 100) = y instances among the bought customers, times the proportion of gender = male instances among the bought customers, times the proportion of bought customers: (3/4)(1/4)(4/10) = 0.075.
- To get the actual probabilities, we must also compute the numerator for the conditional probability of not bought given (TT >= 100) = y, gender = male : (1/6)(4/6)(6/10) = 0.067.
- The denominator is then the sum of these two conditional probabilities (0.075 + 0.067 = 0.14).

Bayes Probabilities

- The conditional probability of bought behaviors given (TT >= 100) = y, gender = male is therefore 0.075/0.14 = 0.53.
- Similarly,
  - P(bought | (TT >= 100) = y, gender = female) = 0.87,
  - P(bought | (TT >= 100) = n, gender = male) = 0.07,
  - P(bought | (TT >= 100) = n, gender = female) = 0.31.

Naïve Bayes Algorithm

Naïve Bayes

- Naïve Bayes is a classification algorithm
- There are two types of Naïve Bayes Algorithms:
  - Discrete Naïve Bayes: For categorical predictors
  - Kernel Naïve Bayes: For numerical predictors

Discrete Naive Bayes

- In this algorithm, the probability of a record belonging to a certain class is evaluated on the basis of conditional probability calculated using Bayes theorem
- Discrete Naive Bayes works only with predictors that are categorical.
- Numerical predictors must be binned and converted to categorical variables before the application Naive Bayes algorithm

Kernel Naïve Bayes

- Kernel Naïve Bayes works with numeric predictors assuming some distribution of the predictors
- It can assume Normal Distribution (Gaussian Naïve Bayes) or any other distribution
- On assuming the distribution, the prior probabilities are calculated
- In R, package e1071 supports only Gaussian distribution(assumes Normality of predictors) and package klar supports "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" and "optcosine" kernels

Application of Naive Bayes in Web Search

- Web search companies such as Google use naive Bayes classifiers to correct misspellings that users type in.
- When we type into Google a phrase that includes a misspelled word, a spelling correction is suggested.
- Suggestion(s) are based on information not only on the frequencies of similarly spelled words typed by millions of other users, but also on the other words in your phrase.

Example 1 : Telecom Customers

- A telecom firm has many customers. Each customer either talks for the duration of more than 100 minutes or less than 100 minutes. The firm has launched a plan for the customers who talk more specially to optimize the amount spent by them on bills. In this case each customer is a record, and the response of interest, Y = {Bought ,Not Bought}, has two classes that a company can be classified into: C1 = Bought and C2 = Not Bought.
- Apart from talk time we also have information about the gender of the customer

Data

- 150 Observations , 3 variables
- First 16 observations:

	Gender	TT_gt_100	Response
1	F	Y	N
2	M	N	N
3	M	N	N
4	F	Y	Y
5	F	N	N
6	F	N	N
7	F	Y	Y
8	M	Y	Y
9	M	Y	N
10	M	N	N
11	F	Y	Y
12	F	Y	Y
13	M	Y	N
14	F	Y	Y
15	F	N	N
16	M	N	Y

Program and Output

```
library(caret)

set.seed(333)
intrain <- createDataPartition(y=telecom$Response,p=0.7,list = FALSE)

training <- telecom[intrain,   ]
testing <- telecom[-intrain,]

library(e1071)
classifier <- naiveBayes(training[,1:2], training[,3])

PredY <- predict(classifier, newdata=testing[,-3], type="class")

# Adjusting the factor levels for relevant output generation
PredY <- factor(PredY,levels = c("Y","N"))
testing[,3] <- factor(testing[,3],levels = c("Y","N"))

tbl <- table(PredY, testing[,3],dnn=list('predicted','actual'))
confusionMatrix(tbl)
```

```
Confusion Matrix and Statistics

         actual
predicted  Y  N
        Y 22  8
        N  1 13

               Accuracy : 0.7955
                 95% CI : (0.647, 0.902)
    No Information Rate : 0.5227
    P-Value [Acc > NIR] : 0.0001702
```

```
'Positive' Class : Y
```

Example 2: Naïve Bayes

- **Qualitative Bankruptcy**: Whether a bank is bankrupt or not can be known from the following qualitative parameters:
  - Industrial Risk
  - Management Risk
  - Financial Flexibility
  - Credibility
  - Competitiveness
  - Operating Risk
- The opinion was taken from experts. They have responded as:
  - P : Positive
  - N : Negative
  - A: Average
- The response variable is Class which contains values as:
  - B: Bankrupt
  - NB: Not Bankrupt

Data

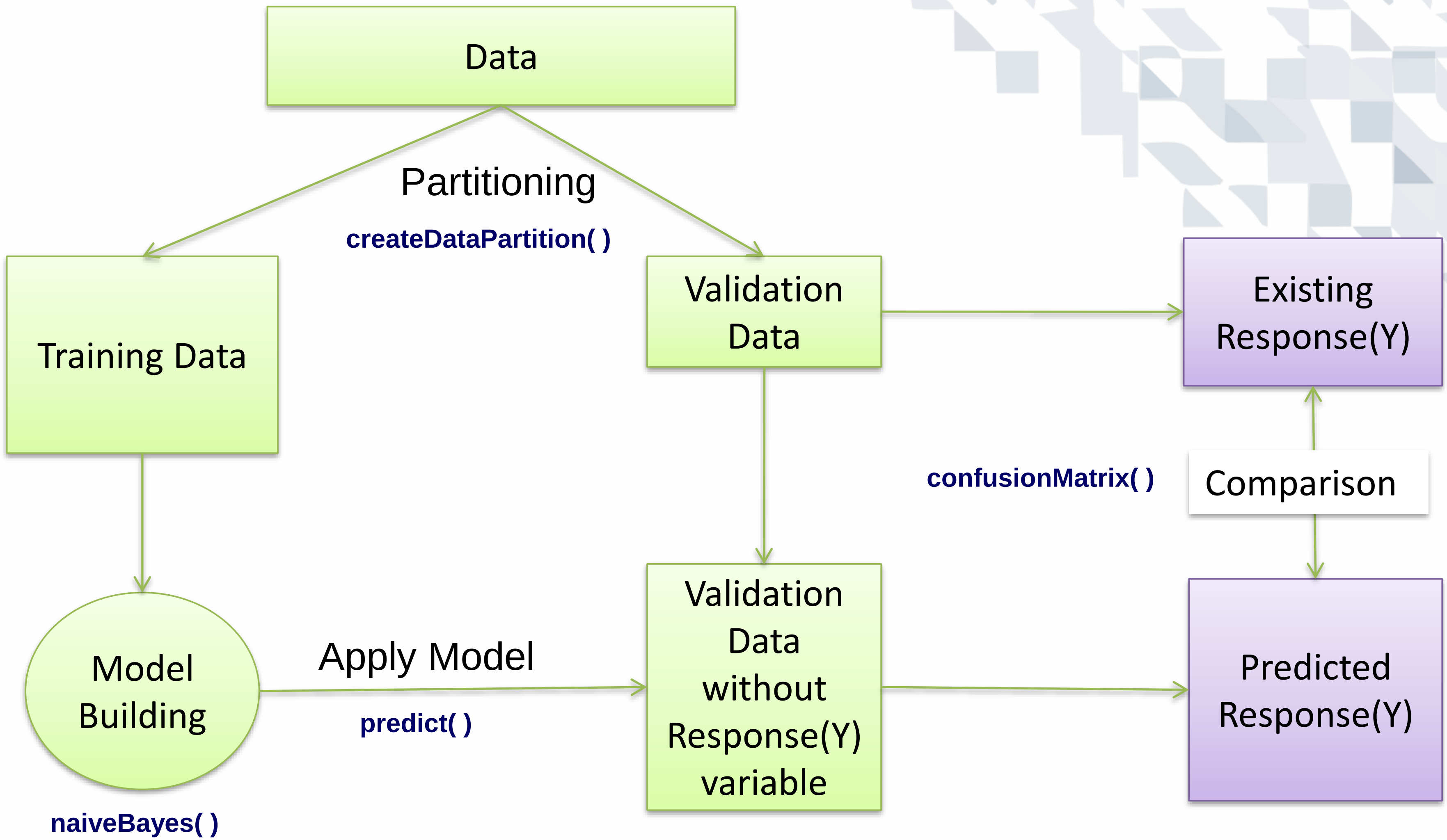
- 250 observations and 7 variables
- First 20 observations:

	Industrial.Risk	Management.Risk	Financial.Flexibility	Credibility	Competitiveness	Operating.Risk	Class
1	P	P	A	A	A	P	NB
2	N	N	A	A	A	N	NB
3	A	A	A	A	A	A	NB
4	P	P	P	P	P	P	NB
5	N	N	P	P	P	N	NB
6	A	A	P	P	P	A	NB
7	P	P	A	P	P	P	NB
8	P	P	P	A	A	P	NB
9	P	P	A	P	A	P	NB
10	P	P	A	A	P	P	NB
11	P	P	P	P	A	P	NB
12	P	P	P	A	P	P	NB
13	N	N	A	P	P	N	NB
14	N	N	P	A	A	N	NB
15	N	N	A	P	A	N	NB
16	N	N	A	P	A	N	NB
17	N	N	A	A	P	N	NB
18	N	N	P	P	A	N	NB
19	N	N	P	A	P	N	NB
20	A	A	A	P	P	A	NB

R Program

- R Program for implementing this classifier can be written using different alternatives
- The following two alternatives we are covering
  - Using package e1071
  - Using package caret (indirectly package klaR)

Naïve Bayes Classifier with e1071



naiveBayes( )

- One of the ways for using Naïve Bayes Algorithm in R is with function naiveBayes( ) in package e1071.
- Package e1071 implements discrete and Gaussian Naïve Bayes Algorithms

Syntax : naiveBayes(formula, data)

naiveBayes(x, y)

Where

formula: A formula of the form class ~ x1 + x2 + ...

Data: data frame of the training data

x : data frame of categorical predictors

y : Class vector or Response in the form of categorical values

predict()

- The function predict( ) is a virtual function which can be executed for different models

Syntax : predict (classifier , newdata , type)

Where

classifier : Object from the built model

newdata : Data frame object containing predictors

type : default is class.

type=class gives the values of class with maximum probabilities

type=raw gives the values of probabilities of different classess

Program and Output

```
library(caret)

set.seed(333)
intrain <- createDataPartition(y=brupt$Class,p=0.7,list = FALSE)

training <- brupt[intrain,   ]
validation <- brupt[-intrain,]
```

Global Environment

Data	
brupt	250 obs. of 7 variables
intrain	int [1:176, 1] 2 3 4 5 6 7 8 9 11 12 ...
training	176 obs. of 7 variables
validation	74 obs. of 7 variables

Program and Output

```
library(e1071)
classifier <- naiveBayes(training[,1:6], training[,7])

PredY <- predict(classifier, newdata=validation[,-7], type="class")

PredYProb <- predict(classifier, newdata=validation[,-7],type="raw")

tbl <- table(PredY, validation[,7],dnn=list('predicted','actual'))

confusionMatrix(tbl)
```

```
Confusion Matrix and Statistics

         actual
predicted  B NB
       B  31  0
       NB  1 42

               Accuracy : 0.9865
                 95% CI : (0.927, 0.9997)
    No Information Rate : 0.5676
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9724
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9688
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9767
             Prevalence : 0.4324
         Detection Rate : 0.4189
   Detection Prevalence : 0.4189
      Balanced Accuracy : 0.9844

       'Positive' Class : B
```

Recap about confusionMatrix()

	Reference	
Predicted	Event	No Event
Event	A	B
No Event	C	D

The formulas used here are:

$$Sensitivity = \frac{A}{A+C}$$

$$Specificity = \frac{D}{B+D}$$

$$Prevalence = \frac{A+C}{A+B+C+D}$$

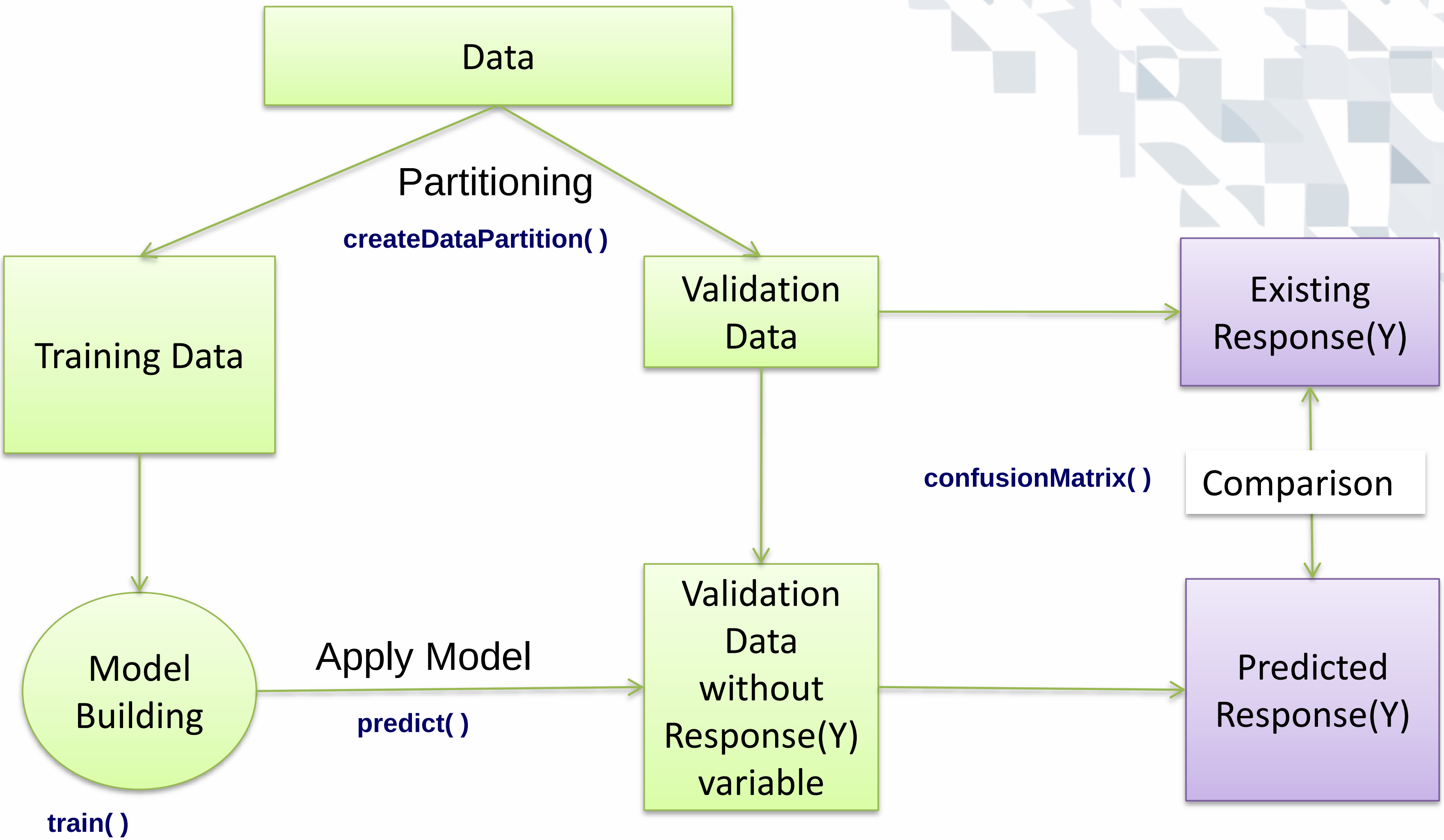
$$PPV = \frac{sensitivity \times prevalence}{((sensitivity \times prevalence) + ((1 - specificity) \times (1 - prevalence))}$$

$$NPV = \frac{specificity \times (1 - prevalence)}{((1 - sensitivity) \times prevalence) + ((specificity) \times (1 - prevalence))}$$

$$Detection\ Rate = \frac{A}{A+B+C+D}$$

$$Detection\ Prevalence = \frac{A+B}{A+B+C+D}$$

Naïve Bayes Classifier with package caret



Package caret Functions

- Package caret provides two functions namely,
  - Function train() for building model
  - Function predict() for predicting on the built model

Syntax:

train(x , y , method , …)

where x : Data frame of predictors

y : Vector of Response variable

Package caret Functions

Syntax:

predict (object , newdata , type , …)

Where

object : Object created of the trained model

newdata : Data frame of predictors

type : either "raw" or "prob“. “raw” for class predictions and “prob” for class probabilites

Executing train( )

- On running the train() function first time with a method we get a prompt as shown below.
- The concerned package must be installed first before the execution of the function train( )

```
> classifier <- train(training[,1:6], training[,7],method = "nb")
1 package is needed for this model and is not installed. (klaR). Would you like to try to install it now?
1: yes
2: no

Selection: 1
Installing package into ‘C:/Users/ssane/Documents/R/win-library/3.2’
(as ‘lib’ is unspecified)
also installing the dependency ‘combinat’

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/combinat_0.0-8.zip'
Content type 'application/zip' length 29294 bytes (28 KB)
downloaded 28 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/klaR_0.6-12.zip'
Content type 'application/zip' length 319596 bytes (312 KB)
downloaded 312 KB

package ‘combinat’ successfully unpacked and MD5 sums checked
package ‘klaR’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ssane\AppData\Local\Temp\Rtmp6xyUFa\downloaded_packages
Loading required package: klaR
Loading required package: MASS
```

Program and Output

```
classifier <- train(training[,1:6], training[,7],method = "nb")

PredY <- predict.train(classifier, newdata=validation[,-7], type="raw")

tbl <- table(PredY, validation[,7],dnn=list('predicted','actual'))
confusionMatrix(tbl)
```

```
Confusion Matrix and Statistics

         actual
predicted  B NB
       B  31  0
       NB  1 42

               Accuracy : 0.9865
                 95% CI : (0.927, 0.9997)
    No Information Rate : 0.5676
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9724
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9688
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9767
             Prevalence : 0.4324
         Detection Rate : 0.4189
   Detection Prevalence : 0.4189
      Balanced Accuracy : 0.9844

       'Positive' Class : B
```