Survival Analysis Versus Classification

NIANG Mohamed
DAVIDAS ROCH Anthnony
KAINA Mohamed Abdellah

8 Novembre 2019

Contents

1 Introduction

The wpbc dataset (available at https://archive.ics.uci.edu/ml/machine learningdatabases/breast-cancer-wisconsin/wpbc.data). It is available at https://archive.ics.uci.edu/ml/machinelearning databases/breast-cancer-wisconsin/wpbc.names. We want to predict the probability of relapse ("recurrent") at 24 months. To do this, you will compare the methods of survival analysis (Cox models, survival random forests,. . .) with the classification methods. Performance measurements (including AUC) will be made on a test sub-sample consisting of 20 to 30% of the data (be careful to stratify well!).

2 Preliminaries

2.1 Set Working Directory

```
WORKING_DIR <- "C:/Users/HP/Desktop/Lab 4"
(WORKING_DIR)
```

```
## [1] "C:/Users/HP/Desktop/Lab 4"
```

```
getwd()
```

```
## [1] "C:/Users/HP/Desktop/Lab 4"
```

2.2 Load libraries

```
# Load Libraries
library(ggfortify)
```

```
## Loading required package: ggplot2
```

```
library(MASS)
library(knitr) # Markdown
library(kableExtra)
library(KMsurv)
library(caret)
```

```
## Loading required package: lattice
```

```
library(e1071)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------ tidyverse
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_confli
## x purrr::accumulate() masks foreach::accumulate()
## x tidyr::expand()     masks Matrix::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::group_rows() masks kableExtra::group_rows()
## x dplyr::lag()        masks stats::lag()
## x purrr::lift()       masks caret::lift()
## x tidyr::pack()       masks Matrix::pack()
## x dplyr::select()     masks MASS::select()
## x tidyr::unpack()     masks Matrix::unpack()
## x purrr::when()       masks foreach::when()
```

```r
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:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(randomForestSRC)
```

```
##
##  randomForestSRC 2.9.1
##
##  Type rfsrc.news() to see new features, changes, and bug fixes.
##
##
## Attaching package: 'randomForestSRC'
## The following object is masked from 'package:purrr':
##
##     partial
## The following objects are masked from 'package:e1071':
##
##     impute, tune
```

```r
library(survival)
```

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

```r
library(foreign) # For reading and writing data stored
library(RWeka) # Weka
```

```
##
## Attaching package: 'RWeka'
## The following objects are masked from 'package:foreign':
##
##     read.arff, write.arff
```

```r
library(ROCR)
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
```

```
##     lowess
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(cvAUC) # AUC for classification
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
##
```

```
## cvAUC version: 1.1.0
```

```
## Notice to cvAUC users: Major speed improvements in version 1.1.0
```

```
##
```

```
library(survAUC) # AUC for Survival
library(risksetROC) # AUC for Survival Random Forest
library(CoxBoost) # For Cox Boost Model
```

```
## Loading required package: prodlim
```

```
library(coxrobust) # For Cox Robust Model
```

2.3 Load data

```
### Load data
wpbc = read_delim("wpbc.data",delim=",",col_names=F,na = '?')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X2 = col_character()
## )
```

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

```
names_cov = paste0(rep(c('radius','texture','perimeter','area','smoothness','compactness',
                         'concavity','concave_points','symmetry','fractal_dimension'),3),
                   c(rep('_mean',10),rep('_SD',10),rep('_worst',10)))
```

```r
names(wpbc) = c('id','recurrent','time',names_cov,c('Tumor_size','Lymph_node_status'))

wpbc = wpbc %>% mutate(id = factor(id)) %>%
                mutate( recurrent = recode_factor(recurrent , "N" = FALSE, 'R' = TRUE ))
glimpse(wpbc)
```

```
## Observations: 198
## Variables: 35
## $ id                      <fct> 119513, 8423, 842517, 843483, 843584, ...
## $ recurrent               <fct> FALSE, FALSE, FALSE, FALSE, TRUE, TRUE...
## $ time                    <dbl> 31, 61, 116, 123, 27, 77, 60, 77, 119,...
## $ radius_mean             <dbl> 18.02, 17.99, 21.37, 11.42, 20.29, 12....
## $ texture_mean            <dbl> 27.60, 10.38, 17.44, 20.38, 14.34, 15....
## $ perimeter_mean          <dbl> 117.50, 122.80, 137.50, 77.58, 135.10,...
## $ area_mean               <dbl> 1013.0, 1001.0, 1373.0, 386.1, 1297.0,...
## $ smoothness_mean         <dbl> 0.09489, 0.11840, 0.08836, 0.14250, 0....
## $ compactness_mean        <dbl> 0.10360, 0.27760, 0.11890, 0.28390, 0....
## $ concavity_mean          <dbl> 0.10860, 0.30010, 0.12550, 0.24140, 0....
## $ concave_points_mean     <dbl> 0.07055, 0.14710, 0.08180, 0.10520, 0....
## $ symmetry_mean           <dbl> 0.1865, 0.2419, 0.2333, 0.2597, 0.1809...
## $ fractal_dimension_mean  <dbl> 0.06333, 0.07871, 0.06010, 0.09744, 0....
## $ radius_SD               <dbl> 0.6249, 1.0950, 0.5854, 0.4956, 0.7572...
## $ texture_SD              <dbl> 1.8900, 0.9053, 0.6105, 1.1560, 0.7813...
## $ perimeter_SD            <dbl> 3.972, 8.589, 3.928, 3.445, 5.438, 2.9...
## $ area_SD                 <dbl> 71.55, 153.40, 82.15, 27.23, 94.44, 30...
## $ smoothness_SD           <dbl> 0.004433, 0.006399, 0.006167, 0.009110...
## $ compactness_SD          <dbl> 0.014210, 0.049040, 0.034490, 0.074580...
## $ concavity_SD            <dbl> 0.03233, 0.05373, 0.03300, 0.05661, 0....
## $ concave_points_SD       <dbl> 0.009854, 0.015870, 0.018050, 0.018670...
## $ symmetry_SD             <dbl> 0.01694, 0.03003, 0.03094, 0.05963, 0....
## $ fractal_dimension_SD    <dbl> 0.003495, 0.006193, 0.005039, 0.009208...
## $ radius_worst            <dbl> 21.63, 25.38, 24.90, 14.91, 22.54, 15....
## $ texture_worst           <dbl> 37.08, 17.33, 20.98, 26.50, 16.67, 20....
## $ perimeter_worst         <dbl> 139.70, 184.60, 159.10, 98.87, 152.20,...
## $ area_worst              <dbl> 1436.0, 2019.0, 1949.0, 567.7, 1575.0,...
## $ smoothness_worst        <dbl> 0.1195, 0.1622, 0.1188, 0.2098, 0.1374...
## $ compactness_worst       <dbl> 0.1926, 0.6656, 0.3449, 0.8663, 0.2050...
## $ concavity_worst         <dbl> 0.3140, 0.7119, 0.3414, 0.6869, 0.4000...
## $ concave_points_worst    <dbl> 0.11700, 0.26540, 0.20320, 0.25750, 0....
## $ symmetry_worst          <dbl> 0.2677, 0.4601, 0.4334, 0.6638, 0.2364...
## $ fractal_dimension_worst <dbl> 0.08113, 0.11890, 0.09067, 0.17300, 0....
## $ Tumor_size              <dbl> 5.0, 3.0, 2.5, 2.0, 3.5, 2.5, 1.5, 4.0...
## $ Lymph_node_status       <dbl> 5, 2, 0, 0, 0, 0, NA, 10, 1, 20, 0, 0,...
```

---

3 Exploratory data analysis

```r
DATASET <- as.data.frame(wpbc)
```

3.1 Look at the data set

```r
head(DATASET)
```

```
##       id recurrent time radius_mean texture_mean perimeter_mean area_mean
## 1 119513     FALSE   31       18.02        27.60         117.50    1013.0
## 2   8423     FALSE   61       17.99        10.38         122.80    1001.0
## 3 842517     FALSE  116       21.37        17.44         137.50    1373.0
## 4 843483     FALSE  123       11.42        20.38          77.58     386.1
## 5 843584      TRUE   27       20.29        14.34         135.10    1297.0
## 6 843786      TRUE   77       12.75        15.29          84.60     502.7
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1         0.09489           0.1036         0.1086             0.07055
## 2         0.11840           0.2776         0.3001             0.14710
## 3         0.08836           0.1189         0.1255             0.08180
## 4         0.14250           0.2839         0.2414             0.10520
## 5         0.10030           0.1328         0.1980             0.10430
## 6         0.11890           0.1569         0.1664             0.07666
##   symmetry_mean fractal_dimension_mean radius_SD texture_SD perimeter_SD
## 1        0.1865                0.06333    0.6249     1.8900        3.972
## 2        0.2419                0.07871    1.0950     0.9053        8.589
## 3        0.2333                0.06010    0.5854     0.6105        3.928
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.1995                0.07164    0.3877     0.7402        2.999
##   area_SD smoothness_SD compactness_SD concavity_SD concave_points_SD
## 1   71.55      0.004433        0.01421      0.03233          0.009854
## 2  153.40      0.006399        0.04904      0.05373          0.015870
## 3   82.15      0.006167        0.03449      0.03300          0.018050
## 4   27.23      0.009110        0.07458      0.05661          0.018670
## 5   94.44      0.011490        0.02461      0.05688          0.018850
## 6   30.85      0.007775        0.02987      0.04561          0.013570
##   symmetry_SD fractal_dimension_SD radius_worst texture_worst
## 1     0.01694             0.003495        21.63         37.08
## 2     0.03003             0.006193        25.38         17.33
## 3     0.03094             0.005039        24.90         20.98
## 4     0.05963             0.009208        14.91         26.50
## 5     0.01756             0.005115        22.54         16.67
## 6     0.01774             0.005114        15.51         20.37
##   perimeter_worst area_worst smoothness_worst compactness_worst
## 1          139.70     1436.0           0.1195            0.1926
## 2          184.60     2019.0           0.1622            0.6656
## 3          159.10     1949.0           0.1188            0.3449
## 4           98.87      567.7           0.2098            0.8663
## 5          152.20     1575.0           0.1374            0.2050
## 6          107.30      733.2           0.1706            0.4196
##   concavity_worst concave_points_worst symmetry_worst
## 1          0.3140               0.1170         0.2677
## 2          0.7119               0.2654         0.4601
## 3          0.3414               0.2032         0.4334
## 4          0.6869               0.2575         0.6638
## 5          0.4000               0.1625         0.2364
## 6          0.5999               0.1709         0.3485
##   fractal_dimension_worst Tumor_size Lymph_node_status
## 1                 0.08113        5.0                 5
## 2                 0.11890        3.0                 2
## 3                 0.09067        2.5                 0
## 4                 0.17300        2.0                 0
```

```
## 5                 0.07678       3.5                0
## 6                 0.11790       2.5                0
```

```r
dim(DATASET)
```

```
## [1] 198  35
```

```r
colnames(DATASET)
```

```
##  [1] "id"                      "recurrent"
##  [3] "time"                    "radius_mean"
##  [5] "texture_mean"            "perimeter_mean"
##  [7] "area_mean"               "smoothness_mean"
##  [9] "compactness_mean"        "concavity_mean"
## [11] "concave_points_mean"     "symmetry_mean"
## [13] "fractal_dimension_mean"  "radius_SD"
## [15] "texture_SD"              "perimeter_SD"
## [17] "area_SD"                 "smoothness_SD"
## [19] "compactness_SD"          "concavity_SD"
## [21] "concave_points_SD"       "symmetry_SD"
## [23] "fractal_dimension_SD"    "radius_worst"
## [25] "texture_worst"           "perimeter_worst"
## [27] "area_worst"              "smoothness_worst"
## [29] "compactness_worst"       "concavity_worst"
## [31] "concave_points_worst"    "symmetry_worst"
## [33] "fractal_dimension_worst" "Tumor_size"
## [35] "Lymph_node_status"
```

```r
summary(DATASET)
```

```
##        id       recurrent        time          radius_mean
##  8423    :  1   FALSE:151   Min.   :  1.00   Min.   :10.95
##  85715   :  1   TRUE : 47   1st Qu.: 14.00   1st Qu.:15.05
##  86208   :  1               Median : 39.50   Median :17.29
##  86517   :  1               Mean   : 46.73   Mean   :17.41
##  87112   :  1               3rd Qu.: 72.75   3rd Qu.:19.58
##  87163   :  1               Max.   :125.00   Max.   :27.22
##  (Other):192
##   texture_mean   perimeter_mean     area_mean      smoothness_mean
##  Min.   :10.38   Min.   : 71.90   Min.   : 361.6   Min.   :0.07497
##  1st Qu.:19.41   1st Qu.: 98.16   1st Qu.: 702.5   1st Qu.:0.09390
##  Median :21.75   Median :113.70   Median : 929.1   Median :0.10190
##  Mean   :22.28   Mean   :114.86   Mean   : 970.0   Mean   :0.10268
##  3rd Qu.:24.66   3rd Qu.:129.65   3rd Qu.:1193.5   3rd Qu.:0.11098
##  Max.   :39.28   Max.   :182.10   Max.   :2250.0   Max.   :0.14470
##
##  compactness_mean  concavity_mean    concave_points_mean symmetry_mean
##  Min.   :0.04605   Min.   :0.02398   Min.   :0.02031     Min.   :0.1308
##  1st Qu.:0.11020   1st Qu.:0.10685   1st Qu.:0.06367     1st Qu.:0.1741
##  Median :0.13175   Median :0.15135   Median :0.08607     Median :0.1893
##  Mean   :0.14265   Mean   :0.15624   Mean   :0.08678     Mean   :0.1928
##  3rd Qu.:0.17220   3rd Qu.:0.20050   3rd Qu.:0.10393     3rd Qu.:0.2093
##  Max.   :0.31140   Max.   :0.42680   Max.   :0.20120     Max.   :0.3040
##
##  fractal_dimension_mean   radius_SD        texture_SD      perimeter_SD
##  Min.   :0.05025        Min.   :0.1938   Min.   :0.3621   Min.   : 1.153
##  1st Qu.:0.05672        1st Qu.:0.3882   1st Qu.:0.9213   1st Qu.: 2.743
```

```
##  Median :0.06171        Median :0.5333   Median :1.1685   Median : 3.767
##  Mean   :0.06271        Mean   :0.6033   Mean   :1.2645   Mean   : 4.255
##  3rd Qu.:0.06671        3rd Qu.:0.7509   3rd Qu.:1.4632   3rd Qu.: 5.213
##  Max.   :0.09744        Max.   :1.8190   Max.   :3.5030   Max.   :13.280
##
##     area_SD       smoothness_SD      compactness_SD      concavity_SD
##  Min.   : 13.99   Min.   :0.002667   Min.   :0.007347   Min.   :0.01094
##  1st Qu.: 35.37   1st Qu.:0.005001   1st Qu.:0.019803   1st Qu.:0.02681
##  Median : 58.45   Median :0.006193   Median :0.027880   Median :0.03691
##  Mean   : 70.23   Mean   :0.006762   Mean   :0.031199   Mean   :0.04075
##  3rd Qu.: 92.48   3rd Qu.:0.007973   3rd Qu.:0.038335   3rd Qu.:0.04897
##  Max.   :316.00   Max.   :0.031130   Max.   :0.135400   Max.   :0.14380
##
##  concave_points_SD   symmetry_SD       fractal_dimension_SD
##  Min.   :0.005174   Min.   :0.007882   Min.   :0.001087
##  1st Qu.:0.011423   1st Qu.:0.014795   1st Qu.:0.002748
##  Median :0.014175   Median :0.017905   Median :0.003719
##  Mean   :0.015099   Mean   :0.020555   Mean   :0.003987
##  3rd Qu.:0.017665   3rd Qu.:0.022880   3rd Qu.:0.004630
##  Max.   :0.039270   Max.   :0.060410   Max.   :0.012560
##
##   radius_worst   texture_worst   perimeter_worst   area_worst
##  Min.   :12.84   Min.   :16.67   Min.   : 85.1   Min.   : 508.1
##  1st Qu.:17.63   1st Qu.:26.21   1st Qu.:118.1   1st Qu.: 947.3
##  Median :20.52   Median :30.14   Median :136.5   Median :1295.0
##  Mean   :21.02   Mean   :30.14   Mean   :140.3   Mean   :1405.0
##  3rd Qu.:23.73   3rd Qu.:33.55   3rd Qu.:159.9   3rd Qu.:1694.2
##  Max.   :35.13   Max.   :49.54   Max.   :232.2   Max.   :3903.0
##
##  smoothness_worst  compactness_worst concavity_worst
##  Min.   :0.08191   Min.   :0.05131   Min.   :0.02398
##  1st Qu.:0.12932   1st Qu.:0.24870   1st Qu.:0.32215
##  Median :0.14185   Median :0.35130   Median :0.40235
##  Mean   :0.14392   Mean   :0.36510   Mean   :0.43669
##  3rd Qu.:0.15488   3rd Qu.:0.42368   3rd Qu.:0.54105
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.17000
##
##  concave_points_worst symmetry_worst   fractal_dimension_worst
##  Min.   :0.02899      Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.15265      1st Qu.:0.2759   1st Qu.:0.07658
##  Median :0.17925      Median :0.3103   Median :0.08689
##  Mean   :0.17878      Mean   :0.3234   Mean   :0.09083
##  3rd Qu.:0.20713      3rd Qu.:0.3588   3rd Qu.:0.10138
##  Max.   :0.29030      Max.   :0.6638   Max.   :0.20750
##
##    Tumor_size     Lymph_node_status
##  Min.   : 0.400   Min.   : 0.000
##  1st Qu.: 1.500   1st Qu.: 0.000
##  Median : 2.500   Median : 1.000
##  Mean   : 2.847   Mean   : 3.211
##  3rd Qu.: 3.500   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :27.000
##                   NA's   :4
```

3.2 Preprocessing of NA value

```
summary(is.na(DATASET))
```

```
##      id           recurrent           time          radius_mean   
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198      
##                                                                 
##  texture_mean    perimeter_mean  area_mean       smoothness_mean
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198      
##                                                                 
##  compactness_mean concavity_mean  concave_points_mean symmetry_mean  
##  Mode :logical    Mode :logical   Mode :logical       Mode :logical  
##  FALSE:198        FALSE:198       FALSE:198           FALSE:198      
##                                                                      
##  fractal_dimension_mean radius_SD       texture_SD      perimeter_SD   
##  Mode :logical          Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198              FALSE:198       FALSE:198       FALSE:198      
##                                                                        
##   area_SD        smoothness_SD   compactness_SD  concavity_SD   
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198      
##                                                                 
##  concave_points_SD symmetry_SD     fractal_dimension_SD radius_worst   
##  Mode :logical     Mode :logical   Mode :logical        Mode :logical  
##  FALSE:198         FALSE:198       FALSE:198            FALSE:198      
##                                                                        
##  texture_worst   perimeter_worst area_worst      smoothness_worst
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical   
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198       
##                                                                  
##  compactness_worst concavity_worst concave_points_worst symmetry_worst 
##  Mode :logical     Mode :logical   Mode :logical        Mode :logical  
##  FALSE:198         FALSE:198       FALSE:198            FALSE:198      
##                                                                        
##  fractal_dimension_worst Tumor_size      Lymph_node_status
##  Mode :logical           Mode :logical   Mode :logical    
##  FALSE:198               FALSE:198       FALSE:194        
##                                          TRUE :4          
```

3.3 Replace the NA value with the median value of variable(numeric)

```
DATASET = DATASET %>% replace_na(list(`Lymph_node_status` = median(DATASET$`Lymph_node_status`, na.rm =
# Verification
summary(is.na(DATASET))
```

```
##      id           recurrent           time          radius_mean   
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198      
##  texture_mean    perimeter_mean  area_mean       smoothness_mean
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:198       FALSE:198       FALSE:198       FALSE:198      
##  compactness_mean concavity_mean  concave_points_mean symmetry_mean  
##  Mode :logical    Mode :logical   Mode :logical       Mode :logical  
##  FALSE:198        FALSE:198       FALSE:198           FALSE:198      
```

```
##   fractal_dimension_mean radius_SD       texture_SD      perimeter_SD
##   Mode :logical          Mode :logical   Mode :logical   Mode :logical
##   FALSE:198              FALSE:198       FALSE:198       FALSE:198
##    area_SD        smoothness_SD   compactness_SD  concavity_SD
##   Mode :logical   Mode :logical   Mode :logical   Mode :logical
##   FALSE:198       FALSE:198       FALSE:198       FALSE:198
##   concave_points_SD symmetry_SD     fractal_dimension_SD radius_worst
##   Mode :logical     Mode :logical   Mode :logical        Mode :logical
##   FALSE:198         FALSE:198       FALSE:198            FALSE:198
##   texture_worst   perimeter_worst area_worst      smoothness_worst
##   Mode :logical   Mode :logical   Mode :logical   Mode :logical
##   FALSE:198       FALSE:198       FALSE:198       FALSE:198
##   compactness_worst concavity_worst concave_points_worst symmetry_worst
##   Mode :logical     Mode :logical   Mode :logical        Mode :logical
##   FALSE:198         FALSE:198       FALSE:198            FALSE:198
##   fractal_dimension_worst Tumor_size      Lymph_node_status
##   Mode :logical           Mode :logical   Mode :logical
##   FALSE:198               FALSE:198       FALSE:198
```

3.4 Normalize the data

```
# Normalized dataset
normalize <- function(x){
  return ((x-mean(x, na.rm = T))/(sd(x, na.rm = T)))
}
DATASET_NORMALIZE = DATASET %>% mutate_at(-c(1,2,3), normalize)
```

3.5 Recode the target variable

```
DATASET_NORMALIZE = DATASET_NORMALIZE %>% arrange(time)
DATASET_FINAL = DATASET_NORMALIZE %>%
                mutate(outcome_classif =
                ifelse((time <= 24)&(recurrent==TRUE),1,
                ifelse((time > 24)&(recurrent==TRUE),0,
                ifelse((time > 24)&(recurrent==FALSE),0,NA))))
```

3.6 Delete the NA value in the outcome variable

```
DATASET_FINAL = DATASET_FINAL[!is.na(DATASET_FINAL[,"outcome_classif"]),]
```

3.7 Delete the row that contain NA

```
DATASET_FINAL = na.omit(DATASET_FINAL)
```

3.8 Train test split With stratification

```
set.seed(1234)
DATASET_FINAL = DATASET_FINAL %>% mutate(id_1n = c(1:nrow(DATASET_FINAL)))
train_index = createDataPartition(DATASET_FINAL$recurrent, p = 0.8, list = FALSE, times = 1)

DATASET_TRAIN = DATASET_FINAL[train_index,]
DATASET_TEST = DATASET_FINAL[-train_index,]

print(nrow(DATASET_TRAIN))
```

```
## [1] 127
```

```
print(nrow(DATASET_TEST))
```

```
## [1] 31
```

3.9 Transformation. Transform Label as Factor (Categorical) and Change Column Names (TRAINING data set)

```
DATASET_TRAIN = dplyr::select(DATASET_TRAIN,-c("id","recurrent","time","id_1n"))
DATASET_TEST = dplyr::select(DATASET_TEST,-c("id","recurrent","time","id_1n"))

DATASET_TRAIN$outcome_classif <- as.factor(DATASET_TRAIN$outcome_classif) # As Category
class(DATASET_TRAIN$outcome_classif)
```

```
## [1] "factor"
```

```
levels(DATASET_TRAIN$outcome_classif)
```

```
## [1] "0" "1"
```

4 Machine Learning Classifiers

4.1 Classification. Predictive Model. Random Forest Algorithm

```
pc <- proc.time()
model.forest <- randomForest(DATASET_TRAIN$outcome_classif ~ ., method="class", data = DATASET_TRAIN)
proc.time() - pc
```

```
##    user  system elapsed
##    0.09    0.00    0.09
```

4.1.1 Confusion Matrix (Random Forest)

```
prediction.forest <- predict(model.forest, newdata=DATASET_TEST, type='class')
table("Actual Class" = DATASET_TEST$outcome_classif, "Predicted Class"=prediction.forest)
```

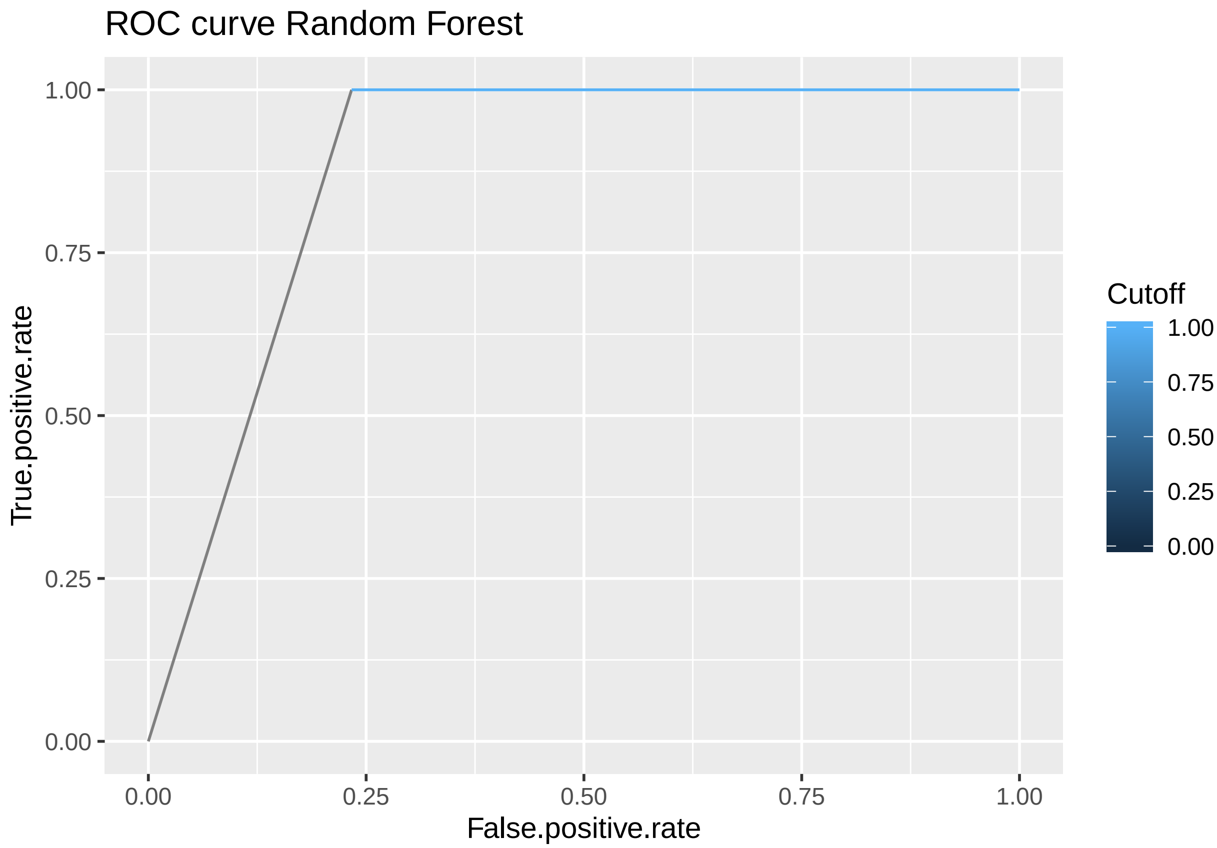
```
##               Predicted Class
## Actual Class  0  1
##             0 23  0
##             1  7  1
```

```
error.rate.forest <- sum(DATASET_TEST$outcome_classif != prediction.forest) / nrow(DATASET_TEST)
accuracy.forest <- 1 - error.rate.forest
print (paste0("Accuary Random Forest (Precision): ", accuracy.forest))
```

```
## [1] "Accuary Random Forest (Precision): 0.774193548387097"
```

4.1.2 ROC curve Random Forest (x-axis: fpr, y-axis: tpr)

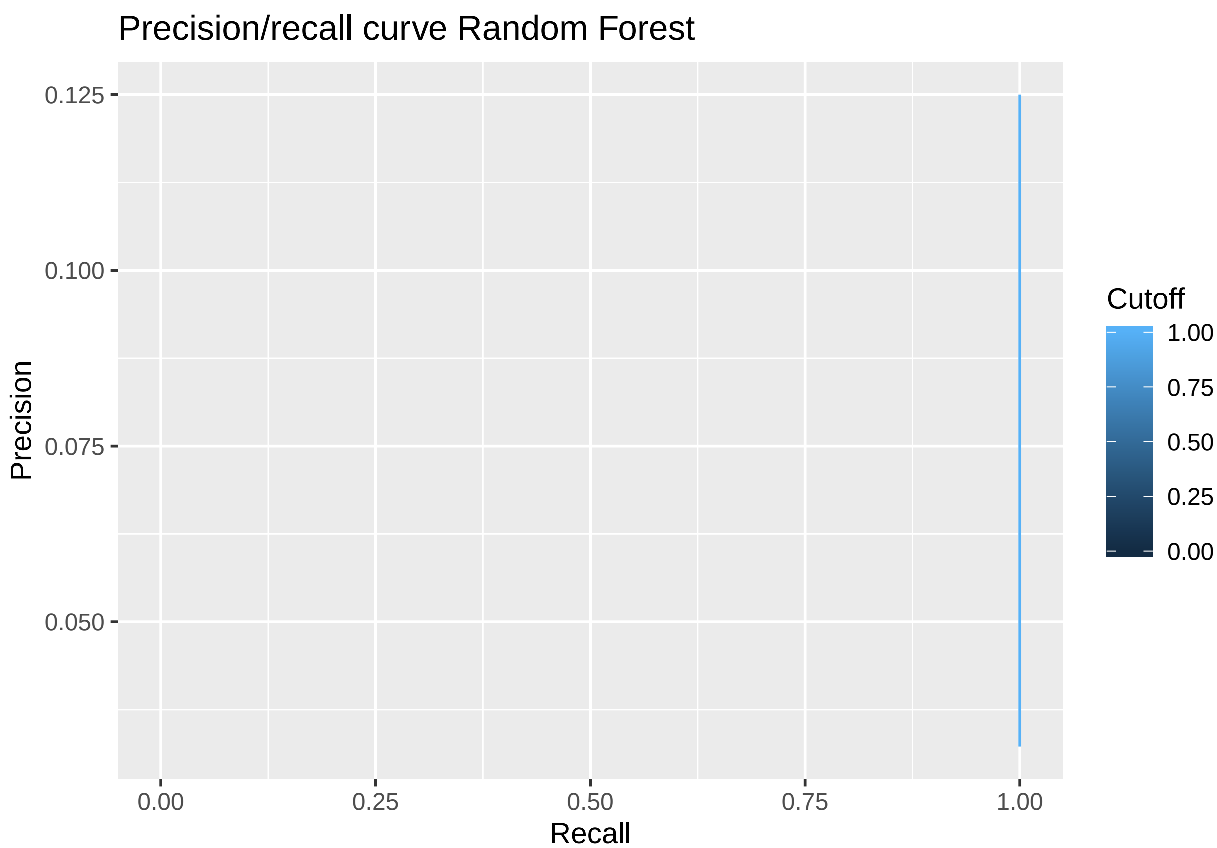
```
pred.forest <- prediction(DATASET_TEST$outcome_classif, prediction.forest)
perf.forest <- performance(pred.forest, "tpr", "fpr")
autoplot(perf.forest, main = 'ROC curve Random Forest')
```



4.1.3 Precision/recall curve Random Forest (x-axis: recall, y-axis: precision)

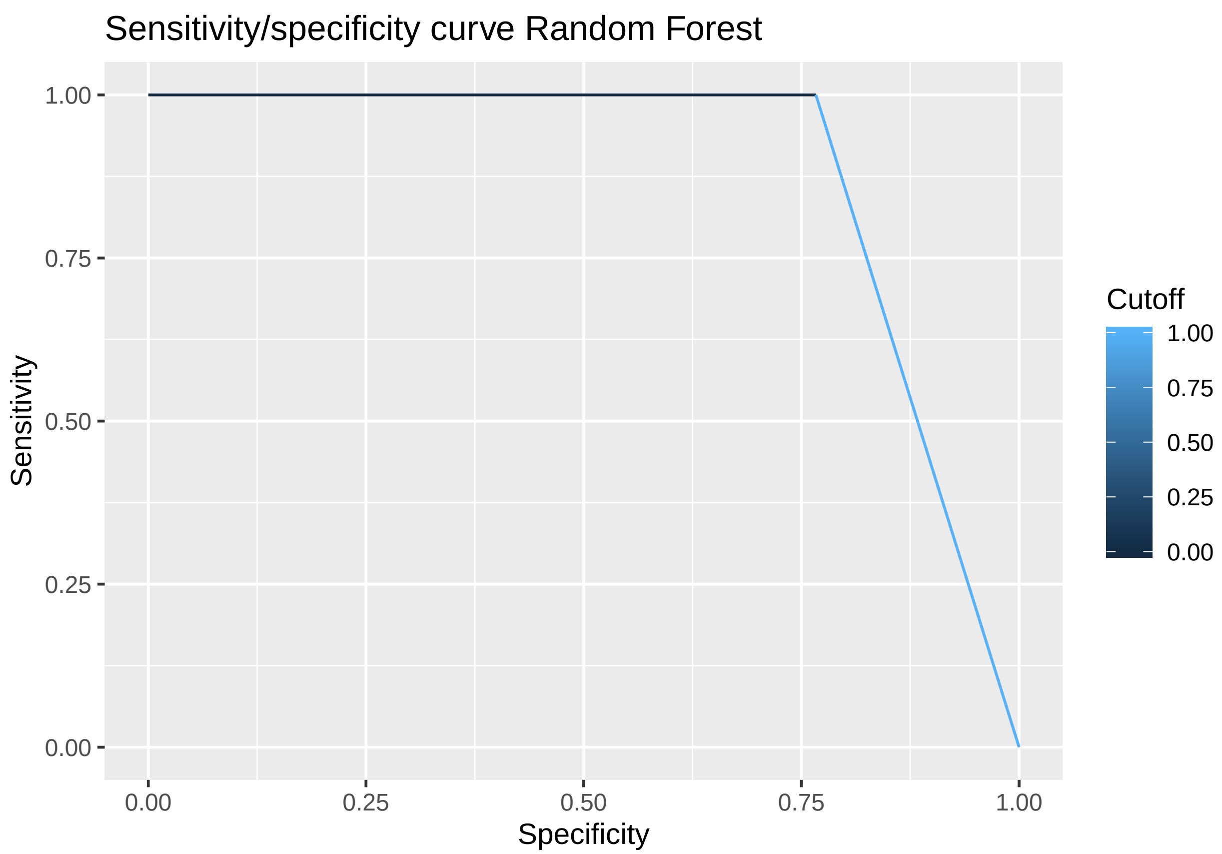
```
perf2.forest <- performance(pred.forest, "prec", "rec")
autoplot(perf2.forest, main = 'Precision/recall curve Random Forest')
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



4.1.4 Sensitivity/specificity curve Random Forest (x-axis: specificity, y-axis: sensitivity)

```
perf2.forest <- performance(pred.forest, "sens", "spec")
autoplot(perf2.forest, main = 'Sensitivity/specificity curve Random Forest')
```



4.1.5 AUC Random Forest

```
auc.forest <- AUC(DATASET_TEST$outcome_classif, prediction.forest)
print (paste0("AUC Random Forest : ", auc.forest))
```

```
## [1] "AUC Random Forest : 0.883333333333333"
```

4.2 Classification. k-Nearest Neighbors (kNN) Algorithm

```
pc <- proc.time()
model.knn <- IBk(DATASET_TRAIN$outcome_classif ~ . , data=DATASET_TRAIN)
proc.time() - pc
```

```
##    user  system elapsed
##    0.25    0.02    0.14
```

```
summary(model.knn)
```

```
##
## === Summary ===
##
## Correctly Classified Instances         127               100      %
## Incorrectly Classified Instances         0                 0      %
## Kappa statistic                          1
## Mean absolute error                      0.0078
## Root mean squared error                  0.0078
## Relative absolute error                  2.7735 %
```

```
## Root relative squared error              2.0865 %
## Total Number of Instances              127
##
## === Confusion Matrix ===
##
##    a   b   <-- classified as
##  106   0 |   a = 0
##    0  21 |   b = 1
```

4.2.1 Confusion Matrix (kNN)

```
prediction.knn <- predict(model.knn, newdata=DATASET_TEST, type='class')
table("Actual Class"=DATASET_TEST$outcome_classif, "Predicted Class"=prediction.knn)
```

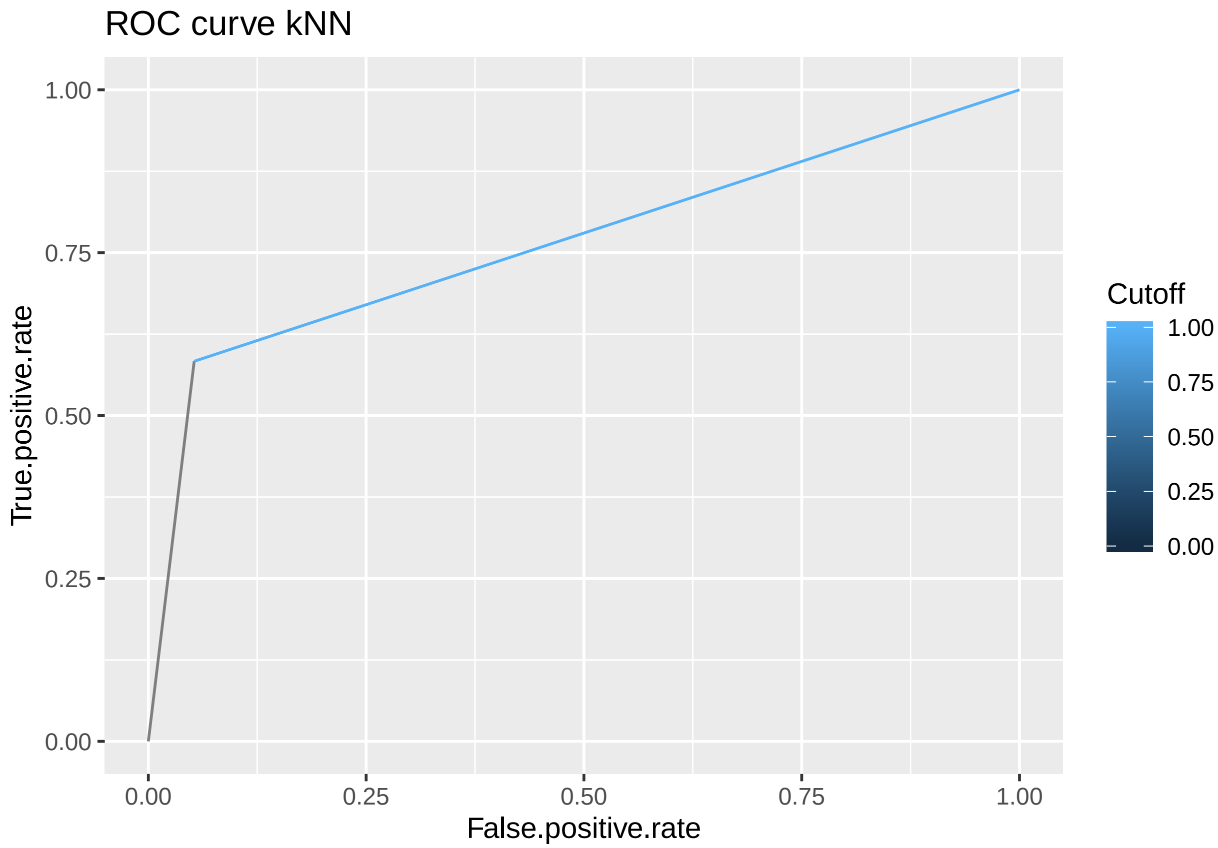
```
##             Predicted Class
## Actual Class  0  1
##            0 18  5
##            1  1  7
```

```
error.rate.knn <- sum(DATASET_TEST$outcome_classif != prediction.knn) / nrow(DATASET_TEST)
print (paste0("Accuary kNN (Precision): ", 1 - error.rate.knn))
```

```
## [1] "Accuary kNN (Precision): 0.806451612903226"
```

4.2.2 ROC curve kNN (x-axis: fpr, y-axis: tpr)

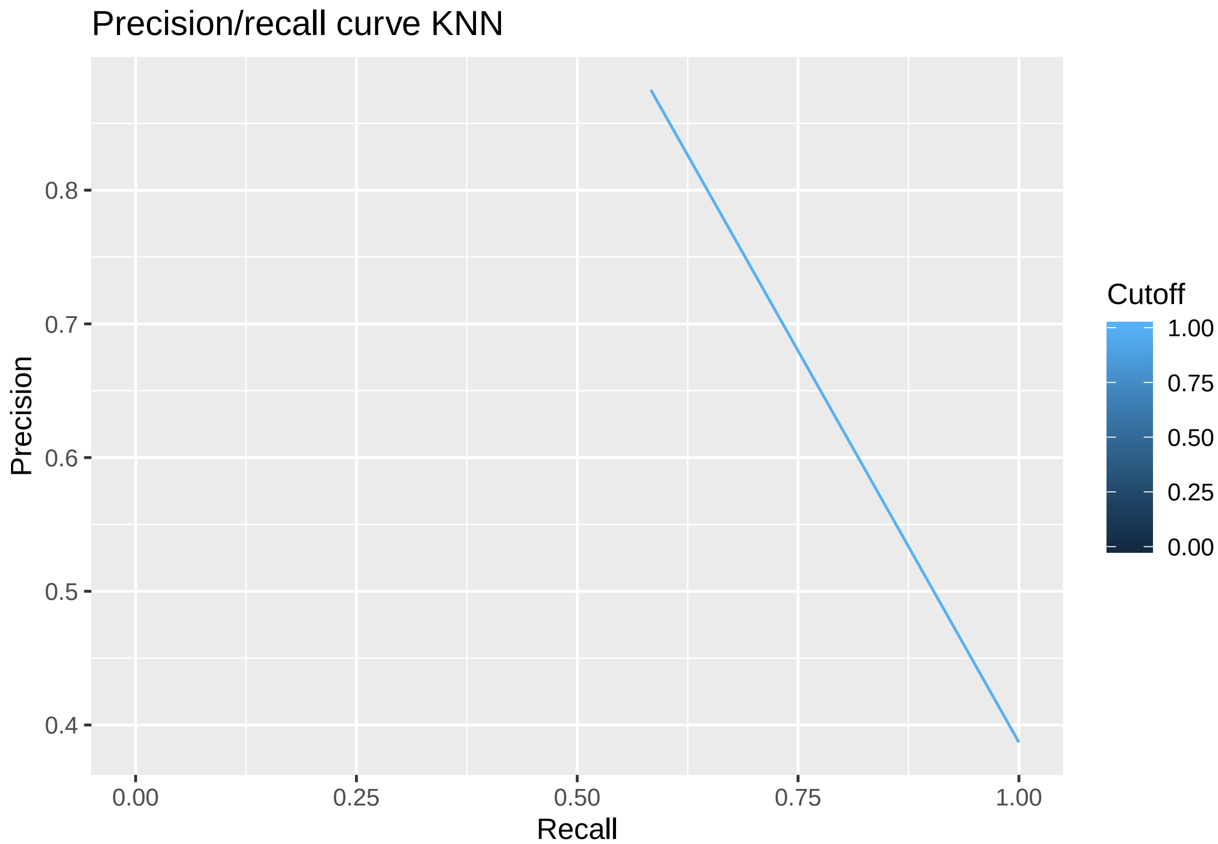
```
pred.knn <- prediction(DATASET_TEST$outcome_classif, prediction.knn)
perf.knn <- performance(pred.knn, "tpr", "fpr")
autoplot(perf.knn, main = 'ROC curve kNN')
```



4.2.3 Precision/recall curve KNN (x-axis: recall, y-axis: precision)

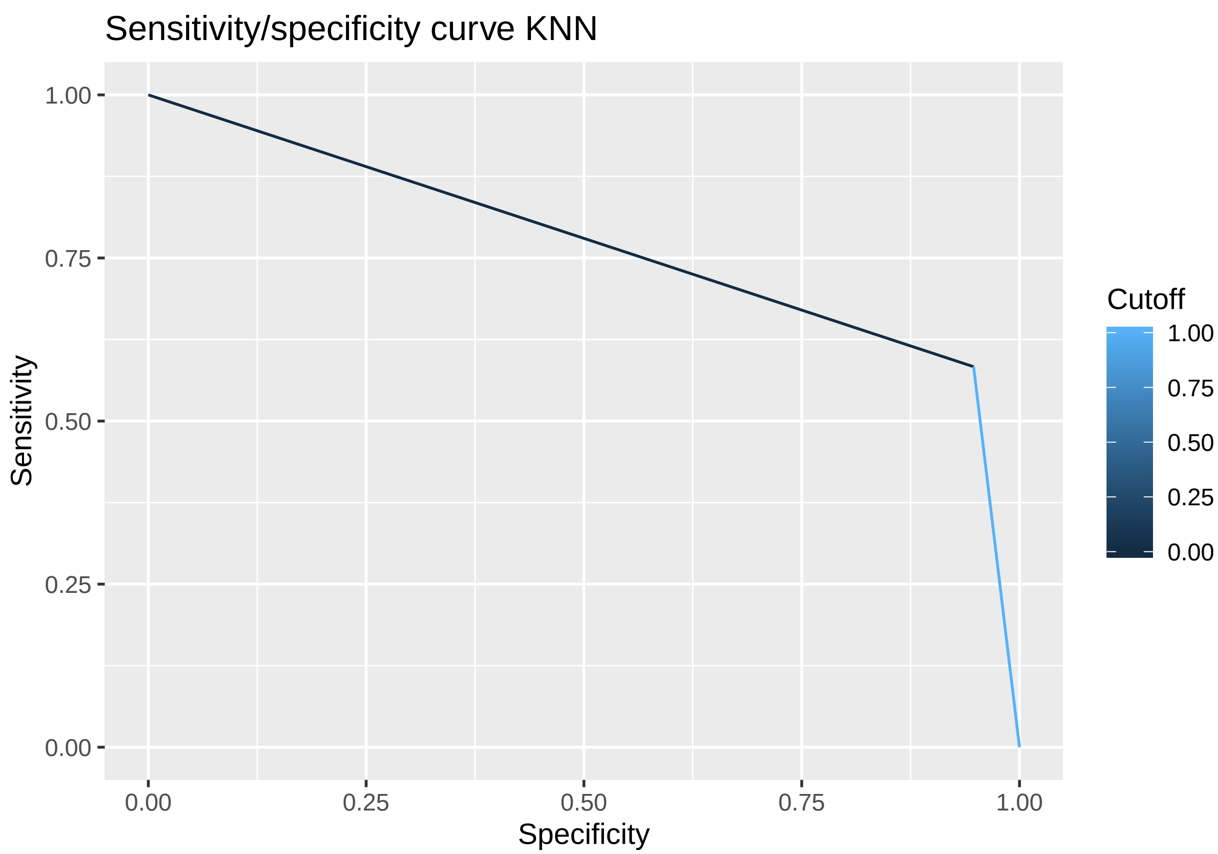
```
perf2.knn <- performance(pred.knn, "prec", "rec")
autoplot(perf2.knn, main = 'Precision/recall curve KNN')
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



4.2.4 Sensitivity/specificity curve KNN (x-axis: specificity, y-axis: sensitivity)

```
perf2.knn <- performance(pred.knn, "sens", "spec")
autoplot(perf2.knn, main = 'Sensitivity/specificity curve KNN')
```



4.2.5 AUC KNN

```
auc.knn <- AUC(DATASET_TEST$outcome_classif, prediction.knn)
print (paste0("AUC KNN : ", auc.knn))
```

```
## [1] "AUC KNN : 0.765350877192983"
```

4.3 Classification. Predictive Model. Naive Bayes Algorithm

```
pc <- proc.time()
model.naiveBayes <- naiveBayes(DATASET_TRAIN$outcome_classif ~ . , data=DATASET_TRAIN)
proc.time() - pc
```

```
##    user  system elapsed
##    0.01    0.00    0.01
```

```
summary(model.naiveBayes)
```

```
##           Length Class  Mode
## apriori    2     table  numeric
## tables    32     -none- list
## levels     2     -none- character
## isnumeric 32     -none- logical
## call       4     -none- call
```

4.3.1 Confusion Matrix (naiveBayes)

```
prediction.naiveBayes <- predict(model.naiveBayes, newdata=DATASET_TEST, type='class')
table("Actual Class"=DATASET_TEST$outcome_classif, "Predicted Class"=prediction.naiveBayes)
```

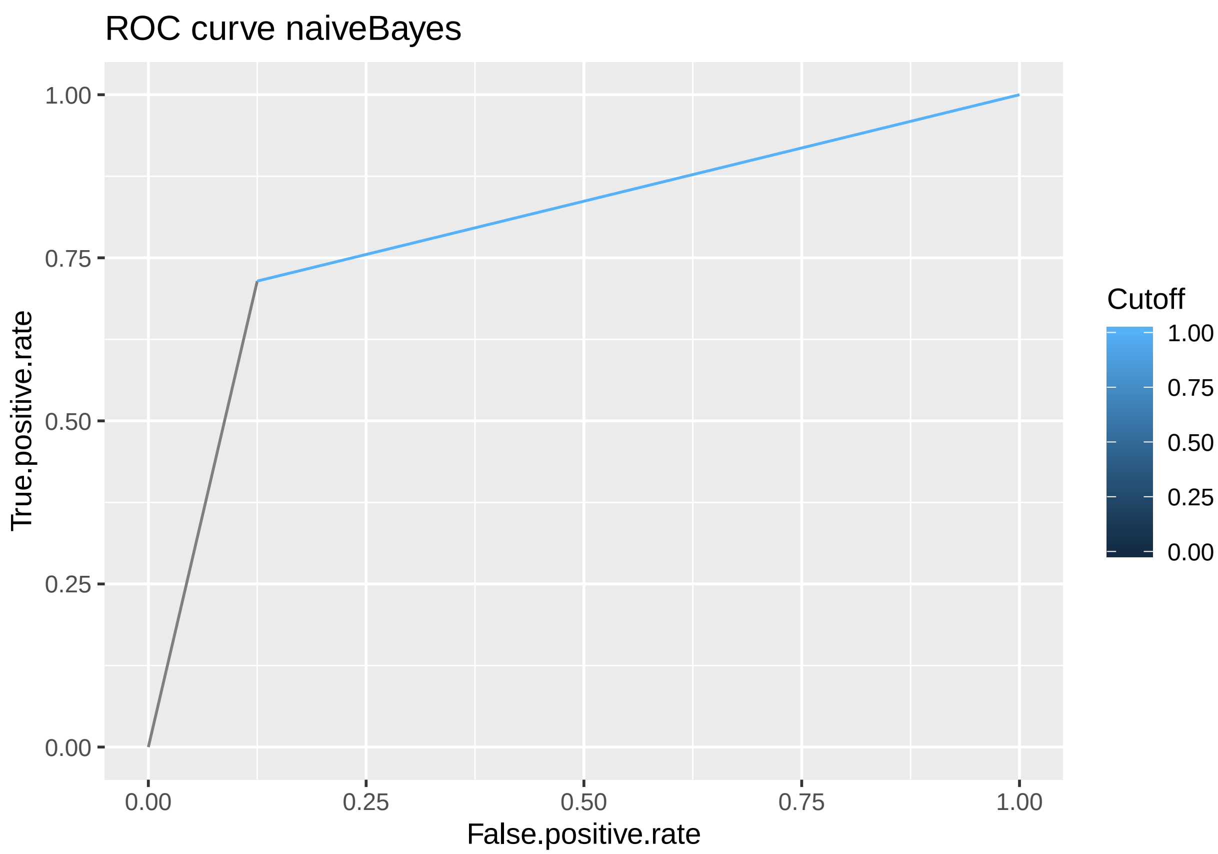
```
##               Predicted Class
## Actual Class   0  1
##             0 21  2
##             1  3  5
```

```
error.rate.naiveBayes <- sum(DATASET_TEST$outcome_classif != prediction.naiveBayes) / nrow(DATASET_TEST
print (paste0("Accuary naiveBayes (Precision): ", 1 - error.rate.naiveBayes))
```

```
## [1] "Accuary naiveBayes (Precision): 0.838709677419355"
```

4.3.2 ROC curve naiveBayes (x-axis: fpr, y-axis: tpr)

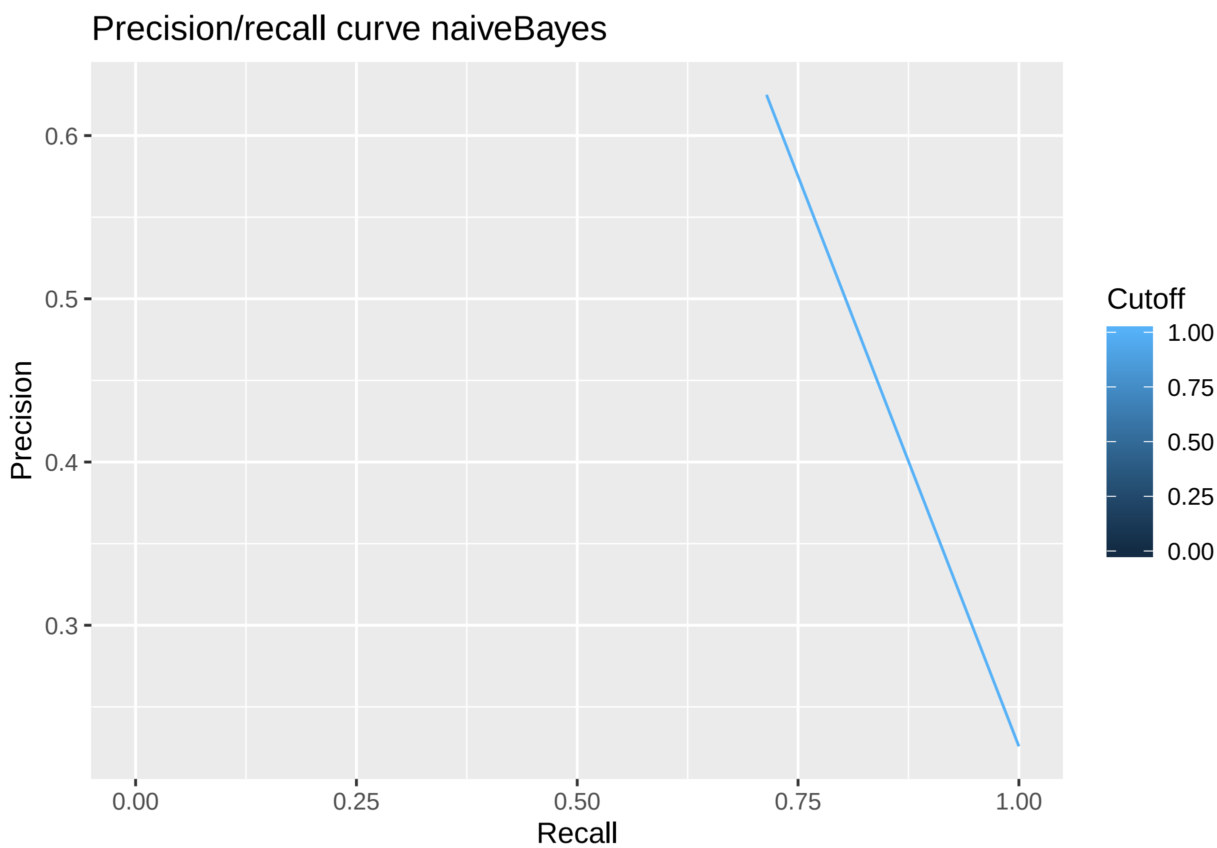
```
pred.naiveBayes <- prediction(DATASET_TEST$outcome_classif, prediction.naiveBayes)
perf.naiveBayes <- performance(pred.naiveBayes, "tpr", "fpr")
autoplot(perf.naiveBayes, main = 'ROC curve naiveBayes')
```



4.3.3 Precision/recall curve naiveBayes (x-axis: recall, y-axis: precision)

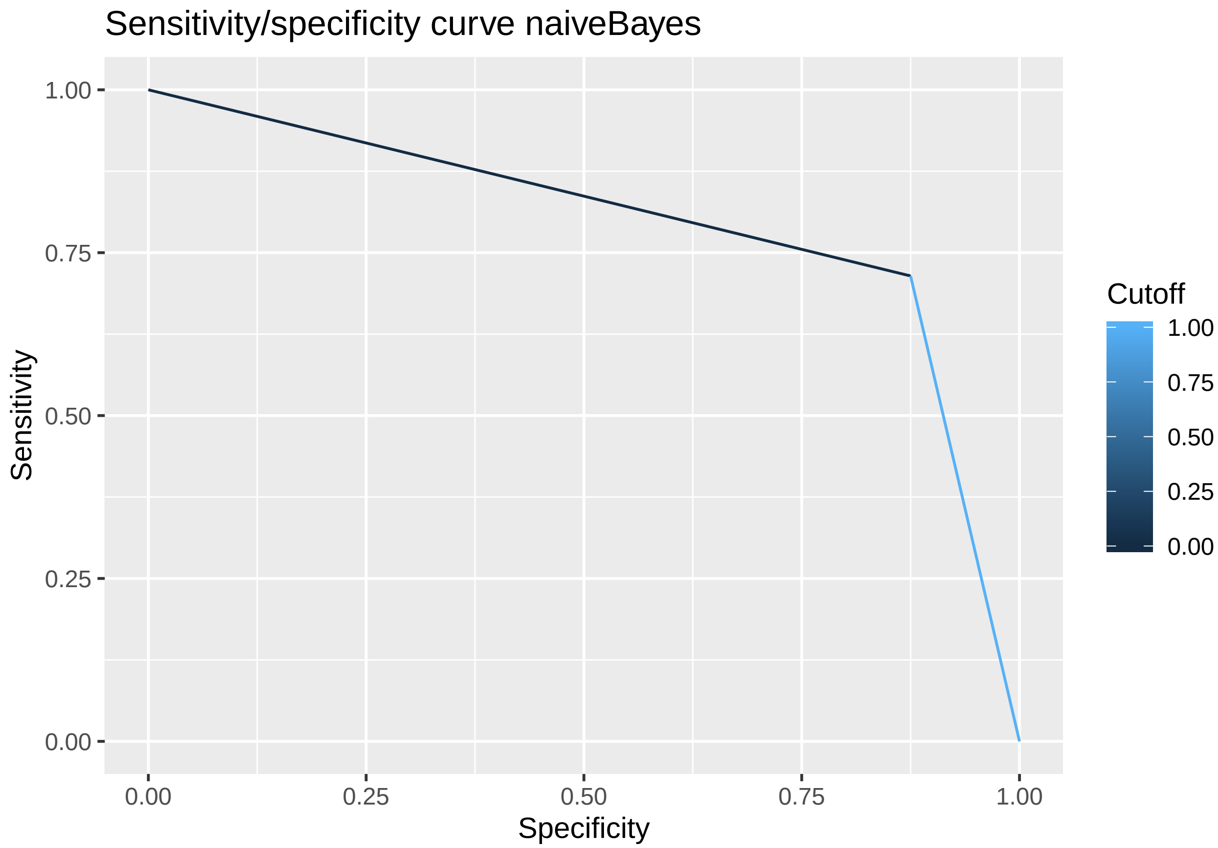
```
perf2.naiveBayes <- performance(pred.naiveBayes, "prec", "rec")
autoplot(perf2.naiveBayes, main = 'Precision/recall curve naiveBayes')
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



4.3.4 Sensitivity/specificity curve naiveBayes (x-axis: specificity, y-axis: sensitivity)

```
perf2.naiveBayes <- performance(pred.naiveBayes, "sens", "spec")
autoplot(perf2.naiveBayes, main = 'Sensitivity/specificity curve naiveBayes')
```



4.3.5 AUC naiveBayes

```
auc.naiveBayes <- AUC(DATASET_TEST$outcome_classif, prediction.naiveBayes)
print (paste0("AUC naiveBayes : ", auc.naiveBayes))
```

```
## [1] "AUC naiveBayes : 0.794642857142857"
```

4.4 Classification. Predictive Model. Logistic Regression Algorithm

```
pc <- proc.time()
model.logistic <- glm(DATASET_TRAIN$outcome_classif ~ . , data=DATASET_TRAIN, family = binomial(logit))
proc.time() - pc
```

```
##    user  system elapsed
##       0       0       0
```

4.4.1 Confusion Matrix (Logistic Regression)

```
prediction.logistic <- predict.glm(model.logistic, newdata=DATASET_TEST, type="response")
sprediction.logistic <- prediction.logistic > 0.5
confusion.matrix <- table("Actual Class" = DATASET_TEST$outcome_classif, "Predicted Class" = spredictio
confusion.matrix
```

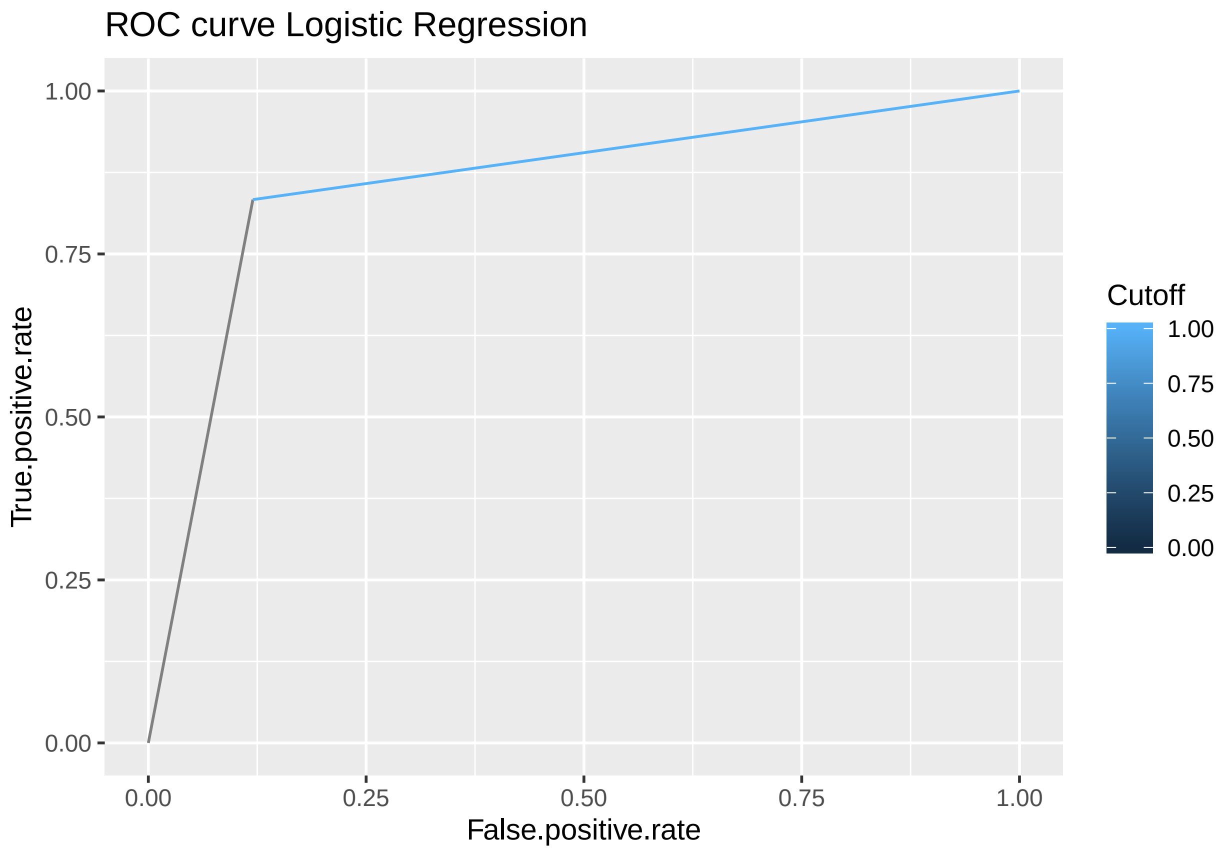
```
##             Predicted Class
## Actual Class FALSE TRUE
##            0    22    1
##            1     3    5
```

```
error.rate.logistic <- (confusion.matrix[2,1]+confusion.matrix[1,2])/sum(confusion.matrix)
print (paste0("Accuary Logistic Regression (Precision): ", 1 - error.rate.logistic))
```

```
## [1] "Accuary Logistic Regression (Precision): 0.870967741935484"
```

4.4.2 ROC curve Logistic Regression (x-axis: fpr, y-axis: tpr)

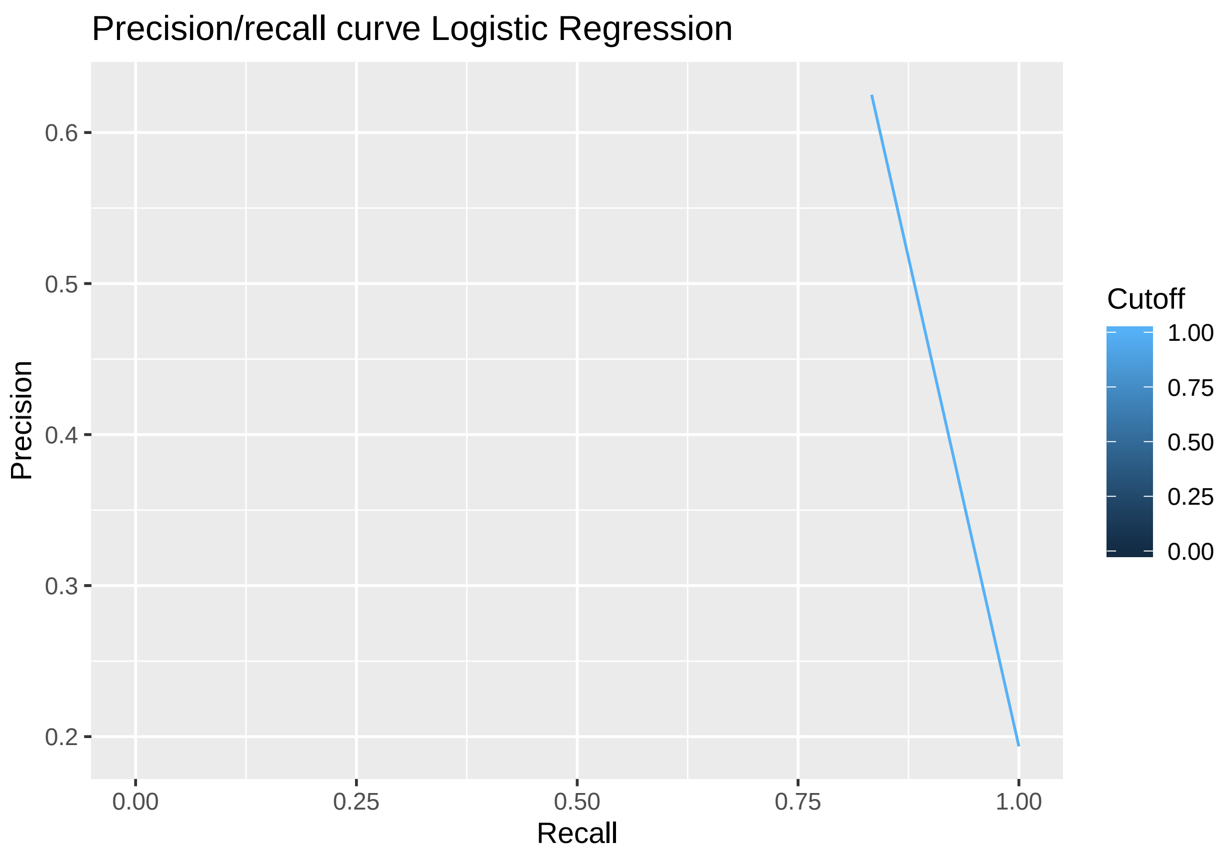
```
pred.logistic <- prediction(DATASET_TEST$outcome_classif, sprediction.logistic)
perf.logistic <- performance(pred.logistic, "tpr", "fpr")
autoplot(perf.logistic, main = 'ROC curve Logistic Regression')
```



4.4.3 Precision/recall curve Logistic Regression (x-axis: recall, y-axis: precision)

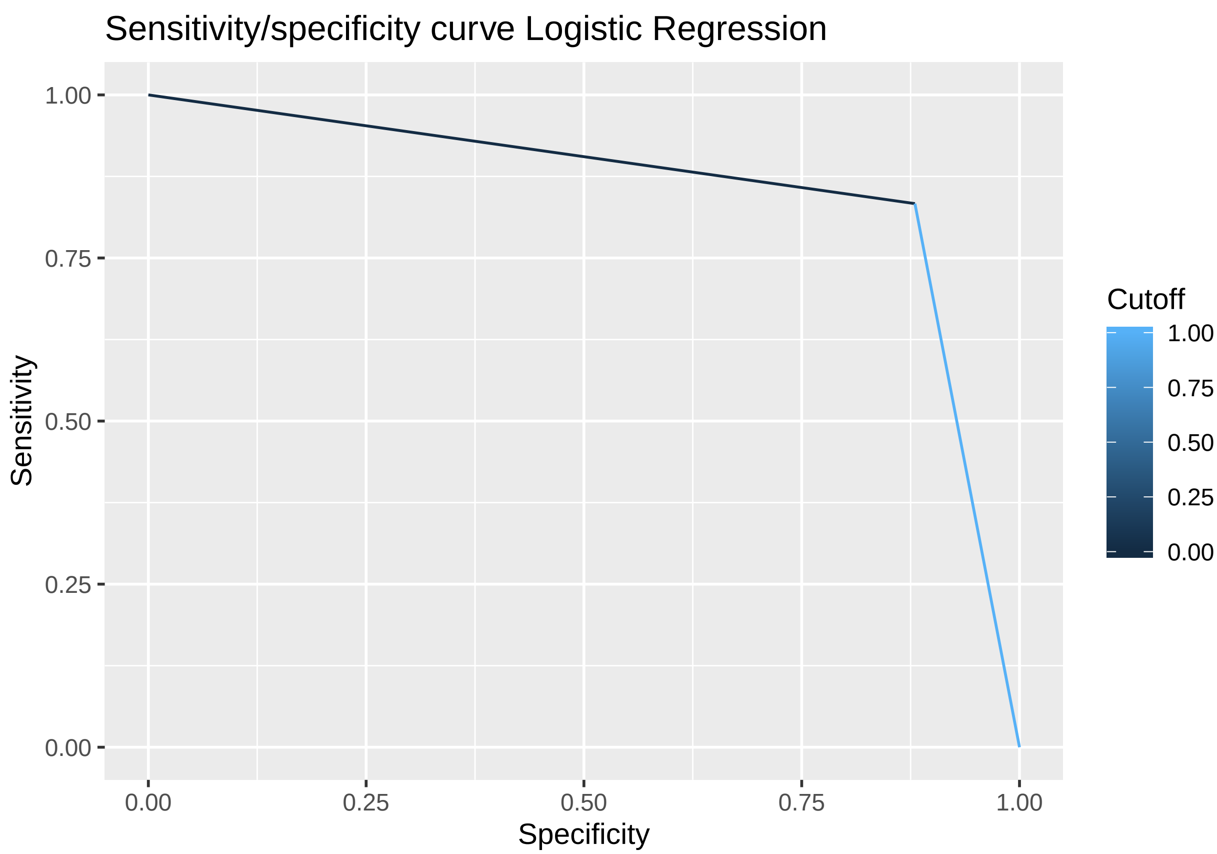
```
perf2.logistic <- performance(pred.logistic, "prec", "rec")
autoplot(perf2.logistic, main = 'Precision/recall curve Logistic Regression')
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



4.4.4 Sensitivity/specificity curve Logistic Regression (x-axis: specificity, y-axis: sensitivity)

```
perf2.logistic <- performance(pred.logistic, "sens", "spec")
autoplot(perf2.logistic, main = 'Sensitivity/specificity curve Logistic Regression')
```



4.4.5 AUC Logistic Regression

```
auc.logistic <- AUC(DATASET_TEST$outcome_classif, sprediction.logistic)
print (paste0("AUC : ", auc.logistic))
```

```
## [1] "AUC : 0.856666666666667"
```

---

5 Survival Analysis

5.1 Preprocessing

```
DATASET_TRAIN2 = DATASET_FINAL[train_index,]
DATASET_TEST2 = DATASET_FINAL[-train_index,]
```

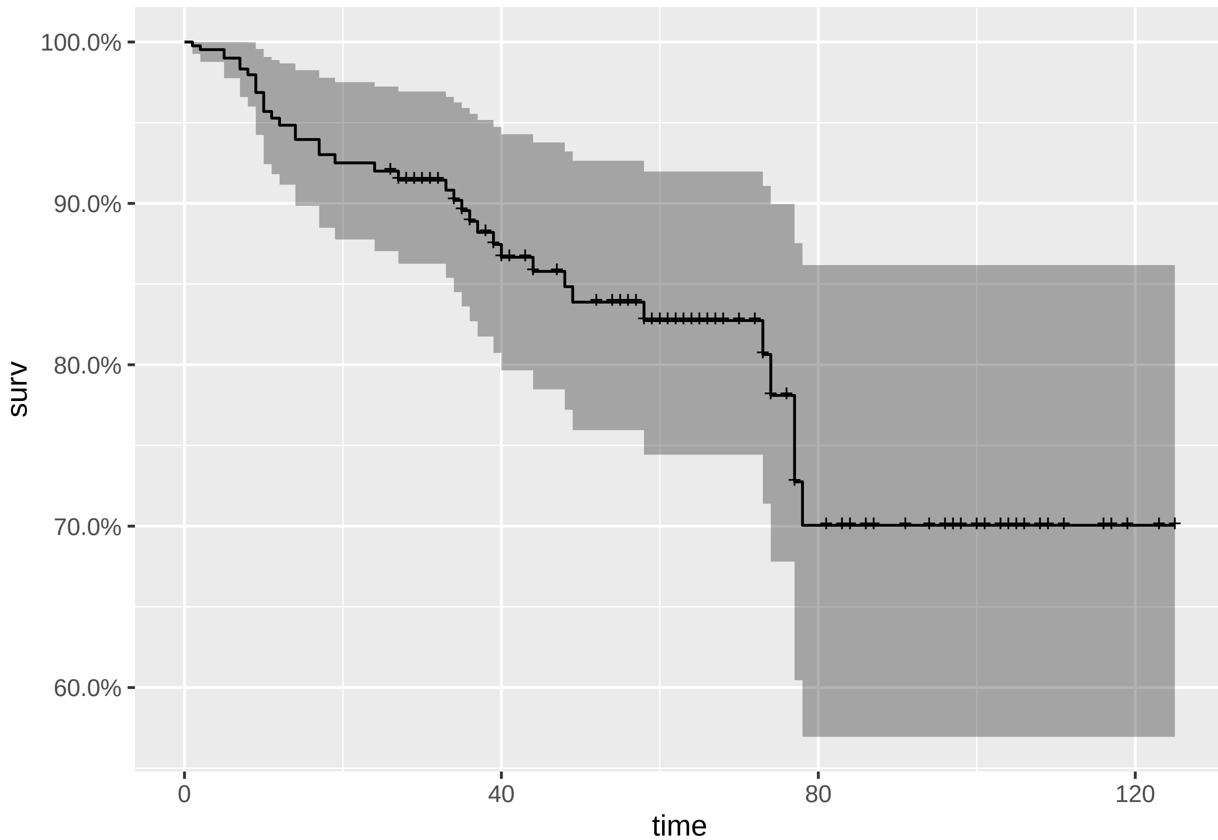
```
DATASET_TRAIN3 = dplyr::select(DATASET_TRAIN2,-c("id","id_1n","outcome_classif"))
DATASET_TRAIN3$recurrent = as.logical(DATASET_TRAIN3$recurrent)

DATASET_TEST3 = dplyr::select(DATASET_TEST2,-c("id","id_1n","outcome_classif"))
DATASET_TEST3$recurrent = as.logical(DATASET_TEST3$recurrent)
```

5.2 Cox Model

```
pc <- proc.time()
cox_all = coxph(Surv(time,recurrent)~., data=DATASET_TRAIN3,x=T,y=T)
proc.time() - pc
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
cox_fit <- survfit(cox_all)
autoplot(cox_fit)
```




```
cox_AIC = stepAIC(cox_all,trace=F)
summary(cox_AIC)
```

```
## Call:
## coxph(formula = Surv(time, recurrent) ~ radius_mean + perimeter_mean +
##     area_mean + smoothness_mean + concavity_mean + fractal_dimension_mean +
##     area_SD + compactness_SD + concave_points_SD + symmetry_SD +
##     radius_worst + texture_worst + area_worst + compactness_worst +
##     concavity_worst + Tumor_size, data = DATASET_TRAIN3, x = T,
##     y = T)
##
##   n= 127, number of events= 38
##
##                              coef  exp(coef)   se(coef)      z Pr(>|z|)
## radius_mean            -2.497e+01  1.428e-11  6.387e+00 -3.910 9.24e-05
## perimeter_mean          1.992e+01  4.464e+08  6.178e+00  3.224 0.001264
## area_mean               6.009e+00  4.071e+02  2.239e+00  2.683 0.007290
## smoothness_mean         1.131e+00  3.098e+00  4.549e-01  2.486 0.012921
## concavity_mean         -2.065e+00  1.268e-01  9.191e-01 -2.247 0.024632
## fractal_dimension_mean -1.618e+00  1.982e-01  4.729e-01 -3.422 0.000621
## area_SD                 1.680e+00  5.364e+00  5.883e-01  2.855 0.004303
```

```
## compactness_SD          6.385e-01  1.894e+00  4.212e-01  1.516 0.129522
## concave_points_SD      -1.165e+00  3.119e-01  4.211e-01 -2.767 0.005663
## symmetry_SD             4.498e-01  1.568e+00  2.572e-01  1.749 0.080361
## radius_worst            5.147e+00  1.719e+02  2.098e+00  2.453 0.014158
## texture_worst           3.895e-01  1.476e+00  2.390e-01  1.630 0.103191
## area_worst             -6.348e+00  1.750e-03  2.166e+00 -2.931 0.003381
## compactness_worst      -1.257e+00  2.844e-01  5.967e-01 -2.107 0.035130
## concavity_worst         1.309e+00  3.701e+00  6.139e-01  2.132 0.033043
## Tumor_size              3.464e-01  1.414e+00  1.532e-01  2.262 0.023724
##
## radius_mean            ***
## perimeter_mean         **
## area_mean              **
## smoothness_mean        *
## concavity_mean         *
## fractal_dimension_mean ***
## area_SD                **
## compactness_SD
## concave_points_SD      **
## symmetry_SD            .
## radius_worst           *
## texture_worst
## area_worst             **
## compactness_worst      *
## concavity_worst        *
## Tumor_size             *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                        exp(coef) exp(-coef) lower .95 upper .95
## radius_mean            1.428e-11  7.001e+10 5.227e-17 3.903e-06
## perimeter_mean         4.464e+08  2.240e-09 2.462e+03 8.091e+13
## area_mean              4.071e+02  2.456e-03 5.053e+00 3.280e+04
## smoothness_mean        3.098e+00  3.228e-01 1.270e+00 7.555e+00
## concavity_mean         1.268e-01  7.888e+00 2.093e-02 7.680e-01
## fractal_dimension_mean 1.982e-01  5.044e+00 7.847e-02 5.008e-01
## area_SD                5.364e+00  1.864e-01 1.693e+00 1.699e+01
## compactness_SD         1.894e+00  5.281e-01 8.295e-01 4.323e+00
## concave_points_SD      3.119e-01  3.206e+00 1.366e-01 7.120e-01
## symmetry_SD            1.568e+00  6.378e-01 9.471e-01 2.596e+00
## radius_worst           1.719e+02  5.819e-03 2.815e+00 1.049e+04
## texture_worst          1.476e+00  6.774e-01 9.241e-01 2.358e+00
## area_worst             1.750e-03  5.713e+02 2.509e-05 1.221e-01
## compactness_worst      2.844e-01  3.516e+00 8.831e-02 9.161e-01
## concavity_worst        3.701e+00  2.702e-01 1.111e+00 1.233e+01
## Tumor_size             1.414e+00  7.072e-01 1.047e+00 1.909e+00
##
## Concordance= 0.757  (se = 0.041 )
## Likelihood ratio test= 43.36  on 16 df,   p=2e-04
## Wald test            = 41.49  on 16 df,   p=5e-04
## Score (logrank) test = 46.72  on 16 df,   p=8e-05
```

```
cox_AIC$score
```

```
## [1] 46.72174
```

```r
summary(survfit(cox_all), time=24)
```

```
## Call: survfit(formula = cox_all)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    24    107      21     0.92   0.026         0.87        0.972
```

```r
summary(survfit(cox_AIC), time=24)
```

```
## Call: survfit(formula = cox_AIC)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    24    107      21    0.904  0.0271        0.853        0.959
```

The survival probability at time 24 is approximately 92%.

5.2.1 AUC Cox Model

```r
lp <- predict(cox_AIC)
lpnew <- predict(cox_AIC, newdata=DATASET_TEST3)
Surv.rsp <- Surv(DATASET_TRAIN3$time, DATASET_TRAIN3$recurrent)
Surv.rsp.new <- Surv(DATASET_TEST3$time, DATASET_TEST3$recurrent)
times <- seq(10, 1000, 10)

AUC_CD.cox <- AUC.cd(Surv.rsp, Surv.rsp.new, lp, lpnew, times)
auc.cox <- AUC_CD.cox[3]
print (paste0("AUC Cox Model : ", auc.cox))
```

```
## [1] "AUC Cox Model : 0.856123737622364"
```

```r
auc.cox <- 0.856123737622364
```

5.3 Survival Random Forests

```r
pc <- proc.time()
rf_surv = rfsrc(Surv(time,recurrent)~radius_mean + perimeter_mean +
    area_mean + smoothness_mean + concavity_mean + fractal_dimension_mean +
    area_SD + compactness_SD + concave_points_SD + symmetry_SD +
    radius_worst + texture_worst + area_worst + compactness_worst +
    concavity_worst + Tumor_size,DATASET_TRAIN3)
proc.time() -pc
```

```
##    user  system elapsed
##    3.65    0.53    0.72
```

```r
rf_surv
```

```
##                          Sample size: 127
##                     Number of deaths: 38
##                      Number of trees: 1000
##            Forest terminal node size: 15
##        Average no. of terminal nodes: 6.139
## No. of variables tried at each split: 4
##               Total no. of variables: 16
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 80
##                             Analysis: RSF
```

```
##                                Family: surv
##                        Splitting rule: logrank *random*
##         Number of random split points: 10
##                            Error rate: 42.21%
```

```
pred_rf=predict(rf_surv,DATASET_TEST3,outcome="test")
pred_rf
```

```
##   Sample size of test (predict) data: 31
##        Number of deaths in test data: 9
##                 Number of grow trees: 1000
##   Average no. of grow terminal nodes: 6.139
##          Total no. of grow variables: 16
##        Resampling used to grow trees: swor
##     Resample size used to grow trees: 20
##                             Analysis: RSF
##                               Family: surv
##                  Test set error rate: 38.03%
```

5.3.1 AUC Survival Random Forests

```
w.ROC1 = risksetAUC(Stime = DATASET_TEST3$time,
                   status = DATASET_TEST3$recurrent,
                   marker = pred_rf$predicted.oob, tmax = 250, plot = F)
w.ROC1
```

```
## $utimes
## [1]  1  3  4  8 11 12 16 19 26
##
## $St
## [1] 0.9677419 0.9354839 0.9032258 0.8709677 0.8387097 0.8064516 0.7741935
## [8] 0.7419355 0.7096774
##
## $AUC
## [1] 0.5758698 0.5829044 0.5909609 0.5755421 0.5776602 0.5735938 0.5700098
## [8] 0.5816028 0.5574550
##
## $Cindex
## [1] 0.5766965
```

```
print (paste0("Survival probability at time ", 24," is between ",w.ROC1$St[9]," and ",w.ROC1$St[8]))
```

```
## [1] "Survival probability at time 24 is between 0.709677419354839 and 0.741935483870968"
```

```
print (paste0("AUC Survival Random Forestsl : ", w.ROC1$Cindex))
```

```
## [1] "AUC Survival Random Forestsl : 0.576696507918134"
```

```
auc.srf <- 0.576696507918134
```

5.4 Cox Boost Model

```
pc <- proc.time()
coxboost_surv = iCoxBoost(Surv(time,recurrent) ~.,data=DATASET_TRAIN3)
proc.time() -pc
```

```
##    user  system elapsed
```

```
##    3.34    0.00    3.36
```

```
summary(coxboost_surv)
```

```
## 8 boosting steps resulting in 3 non-zero coefficients
## partial log-likelihood: -166.8915
##
## Optional covariates with non-zero coefficients at boosting step 8:
## parameter estimate > 0:
##  area_mean, perimeter_worst, Tumor_size
## parameter estimate < 0:
##
```

```
pc <- proc.time()
coxboost_surv = iCoxBoost(Surv(time,recurrent) ~ radius_mean + perimeter_mean +
    area_mean + smoothness_mean + concavity_mean + fractal_dimension_mean +
    area_SD + compactness_SD + concave_points_SD + symmetry_SD +
    radius_worst + texture_worst + area_worst + compactness_worst +
    concavity_worst + Tumor_size,data=DATASET_TRAIN3)
proc.time() - pc
```

```
##    user  system elapsed
##    2.89    0.00    2.89
```

```
summary(coxboost_surv)
```

```
## 10 boosting steps resulting in 3 non-zero coefficients
## partial log-likelihood: -166.4092
##
## Optional covariates with non-zero coefficients at boosting step 10:
## parameter estimate > 0:
##  area_mean, Tumor_size
## parameter estimate < 0:
##  fractal_dimension_mean
```

5.4.1 AUC Cox Boost Model

```
lp2 <- predict(coxboost_surv)
lpnew2 <- predict(coxboost_surv, newdata=DATASET_TEST3)
Surv.rsp2 <- Surv(DATASET_TRAIN3$time, DATASET_TRAIN3$recurrent)
Surv.rsp.new2 <- Surv(DATASET_TEST3$time, DATASET_TEST3$recurrent)
times2 <- seq(10, 1000, 10)

AUC_CD.coxboost <- AUC.cd(Surv.rsp2, Surv.rsp.new2, lp2, lpnew2, times2)
auc.coxboost <- AUC_CD.coxboost[3]
print (paste0("AUC Cox Boost Model : ", auc.coxboost))
```

```
## [1] "AUC Cox Boost Model : 0.567571015744152"
```

```
auc.coxboost <- 0.567571015744152
```

5.5 Cox Robust Model

```
pc <- proc.time()
coxrobust_surv = coxr(Surv(time,recurrent) ~.,data=DATASET_TRAIN3)
summary(coxrobust_surv)
```

```
##                 Length Class  Mode
## coefficients      32   -none- numeric
```

```
## ple.coefficients   32   -none- numeric
## lambda            127   -none- numeric
## lambda.ple        127   -none- numeric
## var              1024   -none- numeric
## var.ple          1024   -none- numeric
## wald.test           1   -none- numeric
## ewald.test          1   -none- numeric
## skip                0   -none- numeric
## call                3   -none- call
## terms               3   terms  call
## x                4064   -none- numeric
## y                 127   Surv   numeric
```

```
pc <- proc.time()
coxrobust_surv = coxr(Surv(time,recurrent) ~  radius_mean + perimeter_mean +
    area_mean + smoothness_mean + concavity_mean + fractal_dimension_mean +
    area_SD + compactness_SD + concave_points_SD + symmetry_SD +
    radius_worst + texture_worst + area_worst + compactness_worst +
    concavity_worst + Tumor_size,data=DATASET_TRAIN3)
proc.time() - pc
```

```
##    user  system elapsed
##    0.06    0.00    0.06
```

```
summary(coxrobust_surv)
```

```
##                   Length Class  Mode
## coefficients       16   -none- numeric
## ple.coefficients   16   -none- numeric
## lambda            127   -none- numeric
## lambda.ple        127   -none- numeric
## var               256   -none- numeric
## var.ple           256   -none- numeric
## wald.test           1   -none- numeric
## ewald.test          1   -none- numeric
## skip                0   -none- numeric
## call                3   -none- call
## terms               3   terms  call
## x                2032   -none- numeric
## y                 127   Surv   numeric
```

5.5.1 AUC Cox Robust Model

```
lp3 <- predict(coxrobust_surv)
Surv.rsp3 <- Surv(DATASET_TRAIN3$time, DATASET_TRAIN3$recurrent)
Surv.rsp.new3 <- Surv(DATASET_TEST3$time, DATASET_TEST3$recurrent)
times3 <- seq(10, 1000, 10)

AUC_CD.coxrobust <- AUC.cd(Surv.rsp3, Surv.rsp.new3, lp3, lp3, times3)
auc.coxrobust <- AUC_CD.coxrobust[3]
print (paste0("AUC Cox Robust Model : ", auc.coxrobust))
```

```
## [1] "AUC Cox Robust Model : 0.813890605104405"
```

```
auc.coxrobust <- 0.813890605104405
```

6 Model comparison and Conclusion

6.1 Model Comparison

```
modelsclass <- c('randomforest', 'knn', 'naiveBayes', 'logreg')
modelssurv <- c('cox', 'srf', 'boostcox', 'robustcox')
aucmodelsclass <- c(auc.forest, auc.knn, auc.naiveBayes, auc.logistic)
aucmodelssurv <- c(auc.cox, auc.srf, auc.coxboost, auc.coxrobust)
resultsclass <- data.frame("Models Classifiers" = modelsclass, "AUC Classifiers" = aucmodelsclass)
resultssurv <- data.frame("Models Survival" = modelssurv, "AUC Survival" = aucmodelssurv)

resultfinal <- cbind(resultsclass,resultssurv)
# Table comparison
kable(arrange(resultfinal,desc(aucmodelsclass),desc(aucmodelssurv)), digits = 2) %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = F,
                font_size = 12,
                position = "left")
```

Models.Classifiers	AUC.Classifiers	Models.Survival	AUC.Survival
randomforest	0.88	cox	0.86
logreg	0.86	robustcox	0.81
naiveBayes	0.79	boostcox	0.57
knn	0.77	srf	0.58

6.2 Conclusion

From the results of the different models we have had, it seems that **the random forest for classification** model gives better results with an AUC of **0.88** and could be used for prediction for new observations. However, the **cox** model makes a good prediction with an AUC of **0.86**.