Laboratório 5

Pedro Silvestre

2023-10-31

Introdução

Vamos trabalhar com o conjunto de dados "data.csv" que diz respeito ao conjunto de dados de câncer de mama Wisconsin. Neste conjunto eles armazenam algumas variaveis de uma amostra de células de câncer de mama. A variável resposta é se o tumor é maligno ou benigno.

```
dados <- read.csv("data.csv")
str(dados)
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                      : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis               : chr  "M" "M" "M" "M" ...
##  $ 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 ...
##  $ X                       : logi  NA NA NA NA NA NA ...
```

Temos 32 variaveis, sendo que a primeira é o id, a segunda é a variavel resposta e as outras 30 são variaveis preditoras. Vamos remover a primeira variavel e a ultima variavel que é a variavel resposta.

```
dados <- dados[,2:32]
head(dados)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         M       17.99        10.38         122.80    1001.0         0.11840
## 2         M       20.57        17.77         132.90    1326.0         0.08474
## 3         M       19.69        21.25         130.00    1203.0         0.10960
## 4         M       11.42        20.38          77.58     386.1         0.14250
## 5         M       20.29        14.34         135.10    1297.0         0.10030
## 6         M       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                0.11890
## 2         0.2750                0.08902
## 3         0.3613                0.08758
## 4         0.6638                0.17300
## 5         0.2364                0.07678
## 6         0.3985                0.12440
```

```r
str(dados)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ 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 ...
```

```r
dados$diagnosis <- as.factor(dados$diagnosis)

n <- 0.8*nrow(dados)

dados <- dados[sample(nrow(dados)),]
treino <- dados[1:n,]
teste <- dados[-(1:n),]
```

Nesta sessão nós dividimos os dados em treino e teste, e também transformamos a variavel resposta em fator.

Modelagem e resultados

K Nearest Neighbors

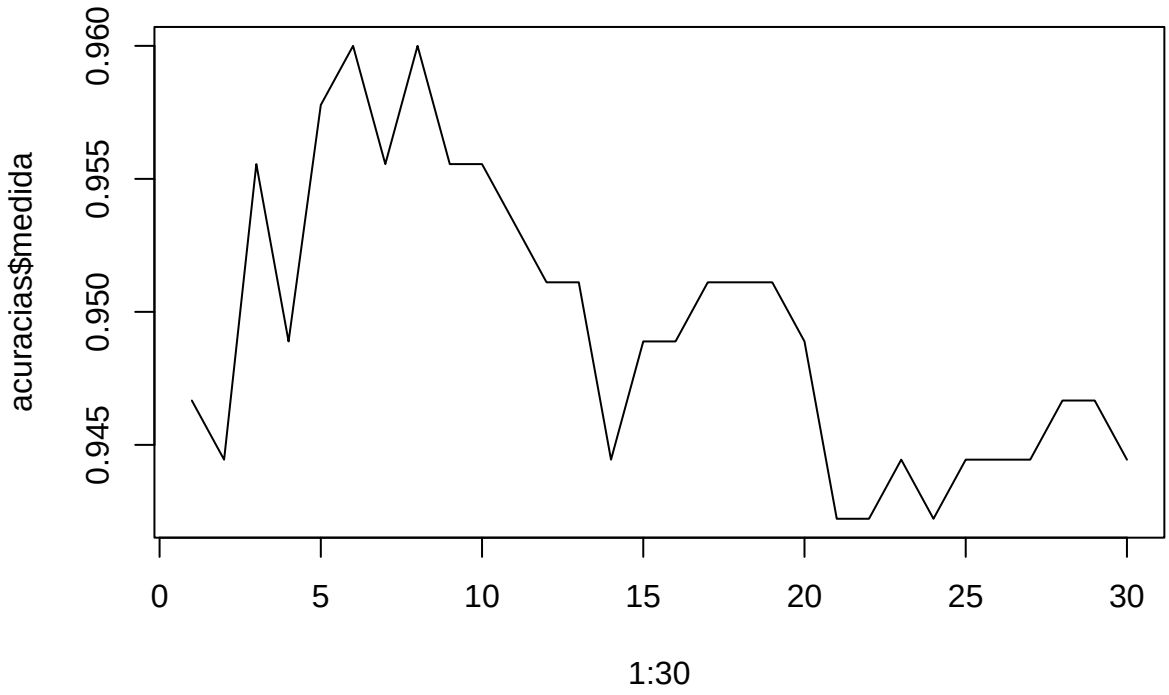
Primeiro vamos ler a biblioteca e calcular o melhor k para o modelo. Primeiro vamos fazer o cross validation com 10 pastas

```r
library(class)

indices <- seq(0, n, by = n/10)
```

```
medida_mod <- c()
medidas <- c()

for (k in 1:30) {
  for (j in 1:10) {
    a <- (indices[j]+1)
    b <- (indices[j+1])
    teste_cross <- treino[(a:b),]
    treino_cross <- treino[-(a:b),]
    mod_cv <- knn(train = scale(treino_cross[,-1]), test = scale(teste_cross[,-1]), k, cl = treino_cross
    medidas[j] <- mean(mod_cv == teste_cross$diagnosis)
  }
  medida_mod[k] <- mean(medidas)
}
```

Agora vamos plotar um grafico para ver qual o melhor k. Além do recurso visual vamos usar o which.max para determinar qual de fato é o melhor k.

```
acuracias <- data.frame(k = 1:30, medida_mod)
plot(x = 1:30, y = acuracias$medida, type = "l")
```




```
melhor_k <- which.max(acuracias$medida_mod)
```

Com o melhor k definido vamos criar ver a sua acurácia no conjunto de teste e a matriz de confusão, já que nesse problema especifico vamos tambem avaliar a sensibilidade e a especificidade do modelo.

```
modelo_knn <- knn(train = treino[,-1], test = teste[,-1], cl = treino$diagnosis, k = melhor_k)

acuraciaknn <- mean(modelo_knn == teste$diagnosis)
acuraciaknn
```

```
## [1] 0.9473684
```

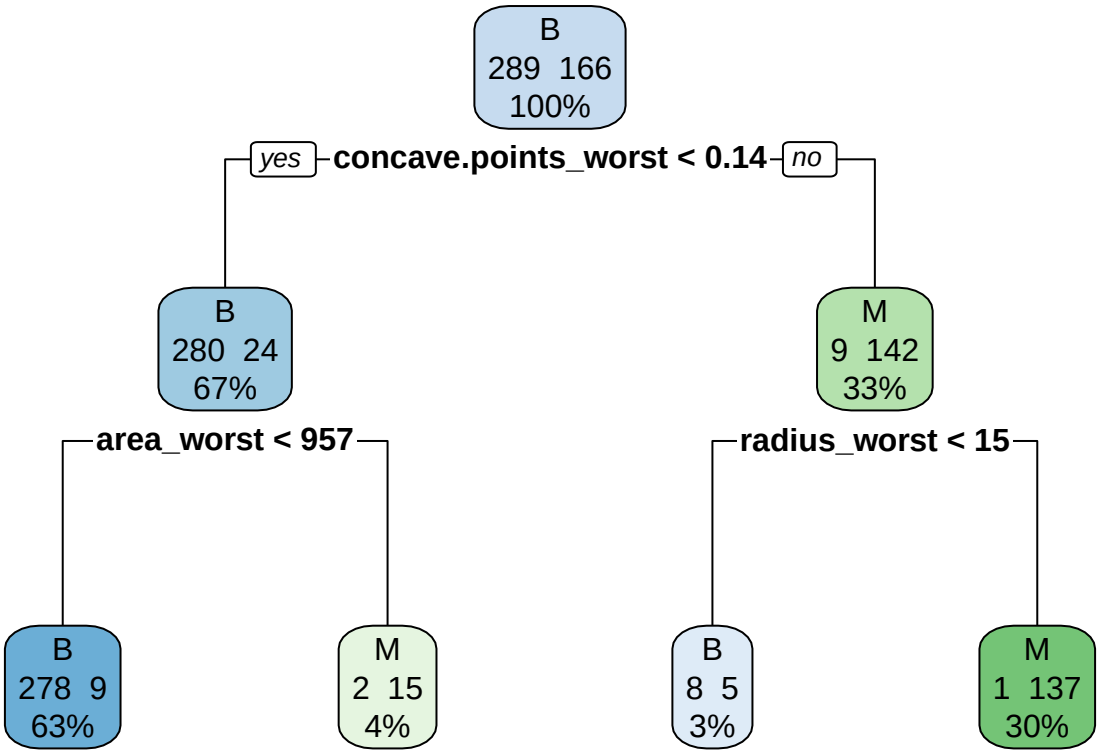
```
matrizknn <- table(teste$diagnosis, modelo_knn)
matrizknn
```

```
##    modelo_knn
##      B  M
##   B 66  2
##   M  4 42
```

Arvore de decisão

Agora vamos fazer o modelo de arvore de decisão. Primeiro vamos ler a biblioteca e fazer o modelo com a biblioteca rpart, já que nela temos recursos visuais melhores. Tambem vamos fazer a matriz de confusão e a acurácia do modelo.

```
library(rpart)
library(rpart.plot)

modeloarvore <- rpart(formula = diagnosis~., data = treino, method = "class")
rpart.plot(modeloarvore, extra = 101)
```

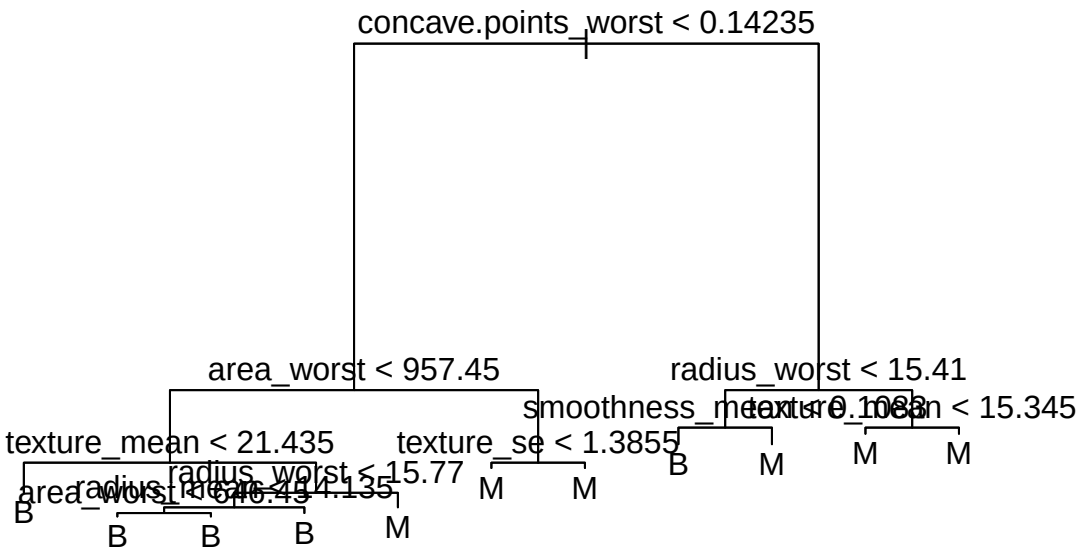


```
acuraciarplot <- mean(predict(modeloarvore, newdata = teste, type = "class")==teste$diagnosis)
matrizarvore <- table(teste$diagnosis, predict(modeloarvore, newdata = teste, type = "class"))
matrizarvore
```

```
##
##      B  M
##   B 64  4
##   M  6 40
```

Agora seguiremos o mesmo caminho, porem com a biblioteca tree já que com ela podemos fazer a poda da arvore e ver qual o melhor tamanho de arvore.

```
library(tree)

tree <- tree(data = treino, formula = diagnosis~., split = "gini")
plot(tree)
text(tree, pretty = 0)
```




```
previsao.cancer <- predict(tree, newdata = teste, type = "class")
mean(previsao.cancer==teste$diagnosis)
```

```
## [1] 0.9298246
```

```
cv.cancer <- cv.tree(tree, FUN = prune.misclass)
cv.cancer
```

```
## $size
## [1] 11  7  5  3  2  1
```

```
##
## $dev
## [1]  36  36  29  28  46 166
##
## $k
## [1]  -Inf   0.0   1.5   4.0  13.0 133.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
arvore.podada <- prune.misclass(tree, best = 10)
previsao.podagem <- predict(arvore.podada, newdata = teste, type = "class")
acuraciatree <- mean(previsao.podagem == teste$diagnosis)
matrizarvore2 <- table(teste$diagnosis, previsao.podagem)
matrizarvore2
```

```
##    previsao.podagem
##      B  M
##   B 63  5
##   M  3 43
```

Podemos comcluir que a arvore podada traz melhores resultados com menos galhos, ou seja um processamento menor.

Floresta Aleatoria

Vamos seguir por um caminho parecido com o da arvore de decisão, porem com a biblioteca randomForest. Primeiro vamos fazer o modelo e depois vamos fazer a matriz de confusão e a acurácia do modelo.

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
modelofloresta <- randomForest(diagnosis~., treino, importance = TRUE)
modelofloresta
```

```
##
## Call:
##  randomForest(formula = diagnosis ~ ., data = treino, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 3.52%
## Confusion matrix:
##     B   M class.error
## B 282   7  0.02422145
```

```
## M   9 157  0.05421687
```

```
importance(modelofloresta)
```

```
##                                B           M MeanDecreaseAccuracy
## radius_mean             8.54220985  5.74442364            9.4718530
## texture_mean            8.83088122  8.58709554           11.9093731
## perimeter_mean          8.13178430  6.62017955            9.6772459
## area_mean               9.74866765  5.04655676           10.6285288
## smoothness_mean         2.76873380  7.53658333            8.2638961
## compactness_mean        4.36844037  3.38173847            5.4359732
## concavity_mean          8.06907315  8.83859905           12.1019924
## concave.points_mean     9.99853730 11.44903828           14.8158576
## symmetry_mean          -0.09745996  4.13688107            3.0335631
## fractal_dimension_mean  3.98241174  2.12253701            4.3684103
## radius_se               4.77472972  5.08613080            7.0739623
## texture_se              3.71559225  1.32844671            4.0148347
## perimeter_se            5.33921545  4.26503832            6.9442706
## area_se                 8.13584304  5.81349092            9.9053129
## smoothness_se           3.04699560  1.33181713            3.2387425
## compactness_se          5.39349957 -1.53414346            4.7583163
## concavity_se            3.42473486  3.92411350            5.0726564
## concave.points_se       3.54550889  1.37306986            3.9733070
## symmetry_se             0.02069488  0.59566252            0.2931561
## fractal_dimension_se    3.41320182  0.02425908            3.0433678
## radius_worst           13.38206558 10.32499261           15.8179727
## texture_worst           7.92474681 11.55386154           12.4481517
## perimeter_worst        12.58381867 12.24585087           16.1167765
## area_worst             14.79405664 12.67450156           17.9664853
## smoothness_worst        6.71946194  9.34815177           11.1958659
## compactness_worst       6.49644145  6.11460290            8.3124983
## concavity_worst         7.04748227 10.34312038           12.2854133
## concave.points_worst   14.19150883 13.61514803           18.4348364
## symmetry_worst          4.56679320  5.54883671            6.7972691
## fractal_dimension_worst 2.51730907  1.82777585            3.3991645
##                        MeanDecreaseGini
## radius_mean                   7.6327346
## texture_mean                  2.8984645
## perimeter_mean               11.4734299
## area_mean                    11.3428363
## smoothness_mean               1.6664739
## compactness_mean              2.5260978
## concavity_mean               12.2636675
## concave.points_mean          24.4945948
## symmetry_mean                 0.6290895
## fractal_dimension_mean        0.7540111
## radius_se                     1.7023263
## texture_se                    0.8901740
## perimeter_se                  1.9918835
## area_se                       4.8089140
## smoothness_se                 0.9481304
## compactness_se                0.9289785
## concavity_se                  1.4629817
## concave.points_se             0.9993908
## symmetry_se                   0.8616271
```

```
## fractal_dimension_se           1.2558781
## radius_worst                  18.2489896
## texture_worst                  3.9084661
## perimeter_worst               25.6863409
## area_worst                    23.7547387
## smoothness_worst               2.9677625
## compactness_worst              3.7789964
## concavity_worst                9.2537202
## concave.points_worst          28.0640737
## symmetry_worst                 2.0896156
## fractal_dimension_worst        1.0176778
```

```
acuraciafloresta <- mean(predict(modelofloresta, newdata = teste, type = "class")==teste$diagnosis)
acuraciafloresta
```

```
## [1] 0.9473684
```

```
matrizfloresta <- table(teste$diagnosis, predict(modelofloresta, newdata = teste, type = "class"))
matrizfloresta
```

```
##
##      B  M
##   B 65  3
##   M  3 43
```

Conclusão

Com os modelos feitos podemos ver que o melhor modelo foi o de floresta aleatoria junto do knn com a melhor acuracia, porem o knn possui a melhor sensibilidade e tambem a maior espeficidade. Na sequencia vem o arvore podada, arvore sem poda. Porem usando esses modelos, percebi que em varias seeds diferentes obtive acuracias diferentes e sempre alternando qual era o melhor modelo.

Table 1: Tabela 1 - Acuracias dos modelos

Modelo	Acuracia
KNN	0.9473684
Arvore (rpart)	0.9122807
Arvore (tree)	0.9298246
Floresta aleatoria	0.9473684

```
matrizknn
```

```
##    modelo_knn
##      B  M
##   B 66  2
##   M  4 42
```

```
matrizarvore
```

```
##
##      B  M
##   B 64  4
##   M  6 40
```

```
matrizarvore2
```

```
##    previsao.podagem
##       B  M
##   B 63  5
##   M  3 43
```

```
matrizfloresta
```

```
##
##       B  M
##   B 65  3
##   M  3 43
```