Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas
Campus - Poços de Caldas / MG

2° Trabalho Prático - Pré-Processamento e Análise de Dados

Maria Clara Sanchez de Mira - 201911020026

1 Identificação do atributo alvo (saída);

O atributo de saída da minha base de dados é o **Class**, pois ela retorna se o paciente tem ou não recorrência de eventos(no-recurrence-events, recurrence-events), o que leva a a conclusão de que essa é a classe de atributo alvo.

```
1 library(readr)
2
3 base <- read.csv("breast-cancer.csv", stringsAsFactors = FALSE)
4
5 atributo_alvo <- base$class
6 table(atributo_alvo)
```

Listing 1: Código fonte em R

Quando se utiliza o comando da linha 6 do código acima é possível visualizar a quantidade de instâncias do atributo alvo, e da um total de 70.3%: no-recurrence-events e 29.7%: recurrence-events. O que representa uma base de dados desbalanceada. E uma base de dados desbalanceada significa que o número de valores varia mais em determinados atributos.

Porém dados desbalanceados, podem ocasionar vários problemas se não forem tratos de forma correta, pode acontecer de ter **Overfitting**, ou seja quando se tem vários dados de um valor apenas, com isso a hipótese vai apenas memorizar os dados e se especializar nos dados de treinamento, ou também ter **Underfiting**, ou seja essa situação pode ocorrer, quando os exemplos de treinamento disponíveis são pouco representativos ou o modelo usado é muito simples e não captura os padrões existentes nos dados.

2 Identificação dos tipos de dados dos atributos de entrada (quantitativo, qualitativo);

Todos os dados dessa base de dados são classificados como qualitativos/simbólicos pois as instâncias de cada atributo, assumem valores que são associados a categoria. Pois mesmo tendo valores que são números na base, por exemplo o atributo *age* , que segue um padrão ou seja

uma ordem de idades que varia de *20-29, 30-39, 40-49, 50-59, 60-69, 70-79*. É possível ver os atributos e suas respectivas classificações na tabela abaixo:

Atributos	Classificação
class	Qualitativo
age	Qualitativo
menopause	Qualitativo
tumor.size	Qualitativo
inv.nodes	Qualitativo
node.caps	Qualitativo
deg.malig	Qualitativo
breast	Qualitativo
breast.quad	Qualitativo
irradiat	Qualitativo

```r
#Ver um por um

unique(base$class)
unique(base$age)
unique(base$menopause)
unique(base$tumor.size)
unique(base$inv.nodes)
unique(base$node.caps)
unique(base$deg.malig)
unique(base$breast)
unique(base$breast.quad)
unique(base$irradiat)
```

Listing 2: Código fonte em R

O código acima retorna os valores que se tem dentro de cada atributo, por exemplo no atributo ***irradiat***, ele retorna ***yes*** ou ***no*** e com isso é possível montar a tabela acima também com qual tipo de cada atributo.

3 Identificação da escala de dados dos atributos de entrada (nominal, ordinal, intervalar, racional);

Na escala, como todos os dados são do tipo qualitativos, a escala vai poder assumir apenas dois valores ou **Nominal** ou **Ordinal**, pois essas duas escalas estão relacionadas ao tipo Qualitativo.

Na escala nominal, os valores assumem apenas nomes que são diferentes, ou seja trazem quase nada de informações e não existe ordem entre seus valores. Por exemplo o atributo **class**

que assume apenas dois valores ou é *no-recurrence-events* ou *recurrence-events*, ou seja pu o valor é uma coisa ou ele é outra, não tem-se relação de ordem, que um deva vim primeiro que o outro.

Na escala ordinal, os valores seguem uma certa ordem, pois mesmo os valores sendo simbólicos é possível definir uma certa ordem entre eles, por exemplo o atributo **deg.malig** que é o grau de malignidade do tumor que pode assumir 1, 2 ou 3 que é a tendência do câncer se tornar progressivamente piores e potencialmente causar a morte de uma pessoa.

Na tabela abaixo é possível ver melhor cada atributo com sua determinada classificação.

Atributos	Classificação
class	Nominal
age	Ordinal
menopause	Ordinal
tumor.size	Ordinal
inv.nodes	Ordinal
node.caps	Nominal
deg.malig	Ordinal
breast	Nominal
breast.quad	Nominal
irradiat	Nominal

Com o código abaixo, também é possível a visualização das categorias que que se tem em cada atributo e com isso foi possível a determinação do tipo Nominal ou Ordinal.

```
1  library('dplyr')
2
3  #Ver como um todo o tipo da variavel
4  glimpse(base)
5
6  #Utilizando a funcao table, para encontrar as cateogiras presente
7  table(base$class)
8  table(base$age)
9  table(base$menopause)
10 table(base$tumor.size)
11 table(base$inv.nodes)
12 table(base$node.caps)
13 table(base$deg.malig)
14 table(base$breast)
15 table(base$breast.quad)
16 table(base$irradiat)
```

Listing 3: Código fonte em R

O retorno da linha 4 do código acima quando usando a função glimpse(base) é:

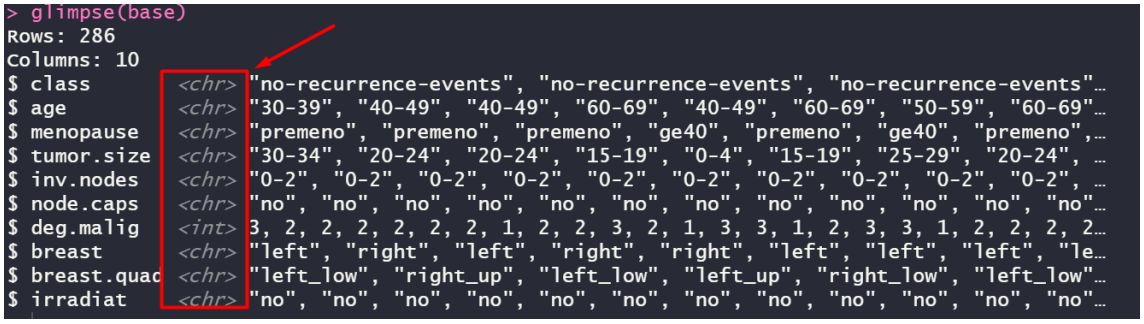

Figura 1: Tipo de cada atributo

É possível reparar que o atributo deg.malig é classificado como tipo inteiro, mas mesmo ele sendo do tipo inteiro, como o câncer segue um padrão de grau 1,2 e 3, como já explicado, esse tipo acaba se tornando um categórico ordinal, e por isso também vai ocorrer a conversão desse dado.

4 Exploração dos dados através de medidas de localidade;

As medidas apontam os pontos de direção nos dados e sofrem uma variação de acordo com o tipo de dado. Por conta da base ter todos os atributos do tipo qualitativo, a única medida de localidade que se aplica é a moda.

E na tabela abaixo é possível ver cada atributo com sua determinada moda, ou seja o valor que mais se repete em cada atributo.

Atributos	Moda
class	no-recurrence-events
age	50-59
menopause	premeno
tumor.size	30-34
inv.nodes	0-2
node.caps	no
deg.malig	2
breast	left
breast.quad	left-low
irradiat	no

```r
# Funcao que encontra a moda
find_mode <- function(x) {
  u <- unique(x)
  tab <- tabulate(match(x, u))
  u[tab == max(tab)]
}
```

```r
#Encontrando a moda de cada atributo
find_mode(base$class)
find_mode(base$age)
find_mode(base$menopause)
find_mode(base$tumor.size)
find_mode(base$inv.nodes)
find_mode(base$node.caps)
find_mode(base$deg.malig)
find_mode(base$breast)
find_mode(base$breast.quad)
find_mode(base$irradiat)
```

Listing 4: Código fonte em R

Porém depois de realizar o passo 13 letra A *(Conversão de tipos (simbólico para numérico, ordinal para numérico, nominal para binário, numérico para ordinal))* é possível a criação de gráficos para melhor visualização da distribuição dos dados e nesse caso é possível ver essa distribuição através do boxplot.

Vale salientar que para a visualização dos dados do boxplot foi utilizada toda a base de dados, e a sua divisão em conjunto de teste e treinamento vem subsequentemente.

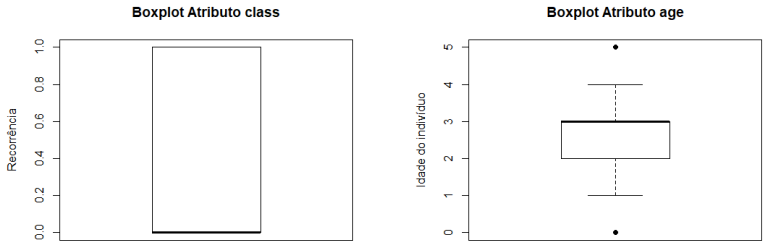

Figura 2: Boxplot atributo class e age

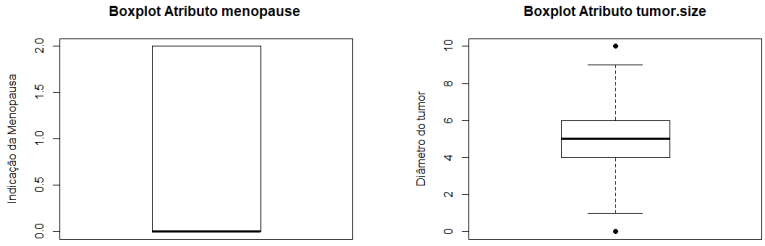

Figura 3: Boxplot atributo menopause e tumor.size

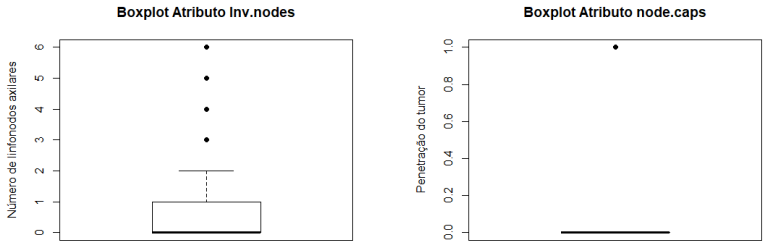

Figura 4: Boxplot atributo inv.nodes e node.caps

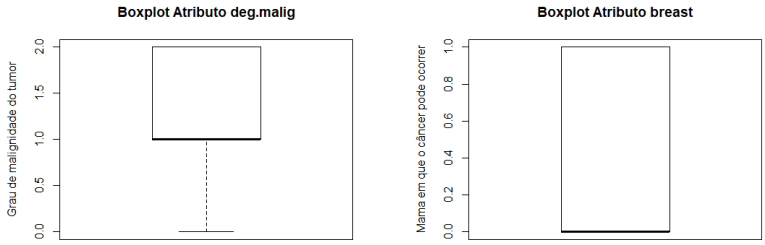

Figura 5: Boxplot atributo deg.malig e breast

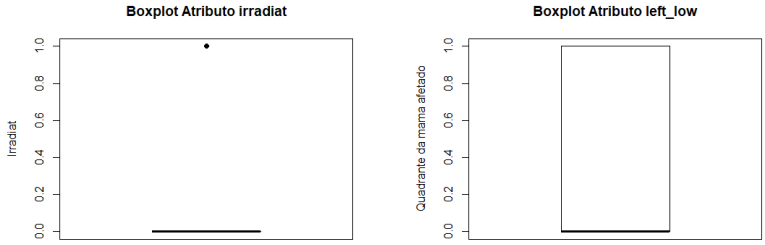

Figura 6: Boxplot atributo irradiat e left-low

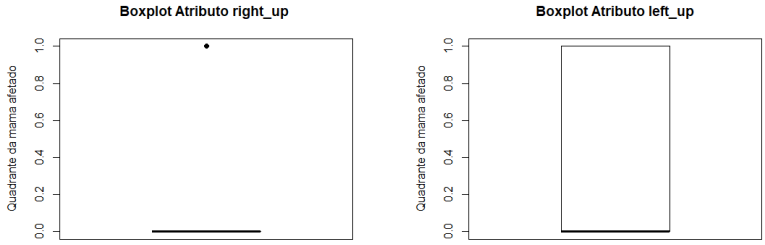

Figura 7: Boxplot atributo right-up e left-up

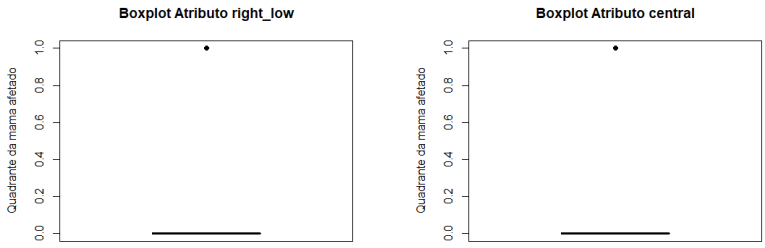

Figura 8: Boxplot atributo right-low e central

Após a visualização dos boxplots é possível notar a presença de alguns outliers em certos atributos, tais como ***age,tumor.size e inv.nodes***, isso ocorre por conta de alguns valores saírem fora da normalidade dos dados e por conta disso acontece as anomalias no gráfico.

Os gráficos ***node.caps,irradiat,right-up,right-low e center*** apesar de apresentarem bolinhas não são considerados outliers, pois os atributos assumem apenas 2 valores, 0 ou 1, se por acaso um sair a mais que o outro, vai parecer que os dados tem uma anomalia.

Com o código abaixo, foi possível visualizar a criação de cada boxplot que está representado bem como nas figuras acima.

```
#Criacao dos boxplots
boxplot(base$class,
        main = "Boxplot Atributo class",
        ylab = "Recorrencia",
        pch = 16)

boxplot(base$age,
        main = "Boxplot Atributo age",
        ylab = "Idade do individuo",
        pch = 16)

boxplot(base$menopause,
        main = "Boxplot Atributo menopause ",
        ylab = "Indicacao da Menopausa",
        pch = 16)

boxplot(base$tumor.size,
        main = "Boxplot Atributo tumor.size",
        ylab = "Diametro do tumor",
        pch = 16)

boxplot(base$inv.nodes,
        main = "Boxplot Atributo Inv.nodes",
        ylab = "Numero de linfonodos axilares",
        pch = 16)
```

```
boxplot(base$node.caps,
        main = "Boxplot Atributo node.caps",
        ylab = "Penetracao do tumor",
        pch = 16)

boxplot(base$deg.malig ,
        main = "Boxplot Atributo deg.malig",
        ylab = "Grau de malignidade do tumor",
        pch = 16)

boxplot(base$breast,
        main = "Boxplot Atributo breast",
        ylab = "Mama em que o cancer pode ocorrer",
        pch = 16)

boxplot(base$irradiat,
        main = "Boxplot Atributo irradiat",
        ylab = "Irradiat",
        pch = 16)

boxplot(base$left_low,
        main = "Boxplot Atributo left-low",
        ylab = "Quadrante da mama afetado",
        pch = 16)

boxplot(base$right_up,
        main = "Boxplot Atributo right-up ",
        ylab = "Quadrante da mama afetado",
        pch = 16)

boxplot(base$left_up,
        main = "Boxplot Atributo left-up",
        ylab = "Quadrante da mama afetado",
        pch = 16)

boxplot(base$right_low,
        main = "Boxplot Atributo right-low",
        ylab = "Quadrante da mama afetado",
        pch = 16)

boxplot(base$central,
        main = "Boxplot Atributo central",
        ylab = "Quadrante da mama afetado",
        pch = 16)
```

Listing 5: Código fonte em R

5 Exploração dos dados através de medidas de espalhamento;

As medidas de espalhamento são responsáveis pela medição da dispersão de um conjunto de valores. E com isso possibilita a a visualização dos dados, ou seja se eles estão muito espalhados ou relativamente concentrados em torno da média por exemplo.

Com isso eu construí uma tabela simples, com a média de cada atributo para poder visualizar melhor como os gráficos se comportam em relação a média.

Atributos	Média
class	0.29720280
age	2.66136486
menopause	0.92657343
tumor.size	4.88111888
inv.nodes	0.51748252
node.caps	0.19580420
deg.malig	1.04895105
breast	0.46853147
left_low	0.38811189
right_up	0.11538462
left_up	0.33916084
right_low	0.08391608
central	0.07342657
irradiat	0.23776224

Figura 9: Tabela com as média dos atributos

Abaixo o histograma de cada atributo com sua respectiva frequência, ou seja, número de objetos que um certo atributo tem.

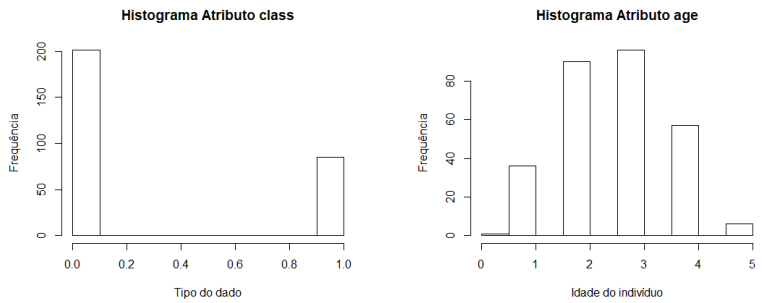

Figura 10: Histograma atributo class e age

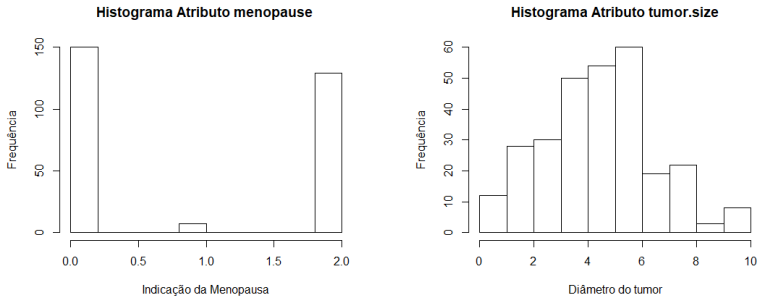

Figura 11: Histograma atributo menopause e tumor.size

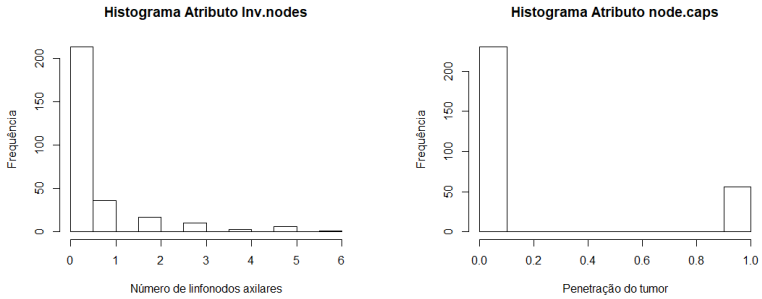

Figura 12: Histograma atributo inv.nodes e node.caps

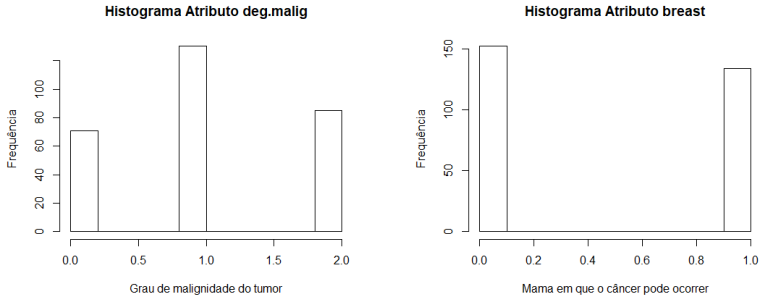

Figura 13: Histograma atributo deg.malig e breast

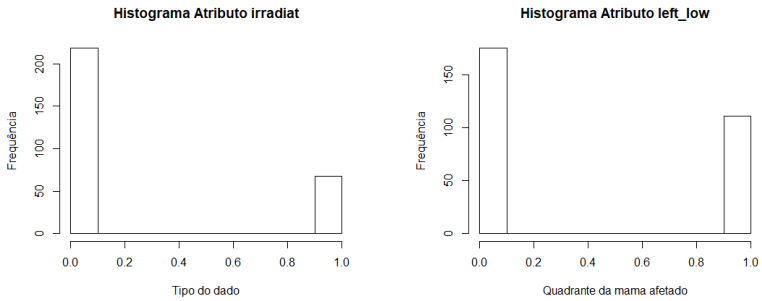

Figura 14: Histograma atributo irradiat e left-low

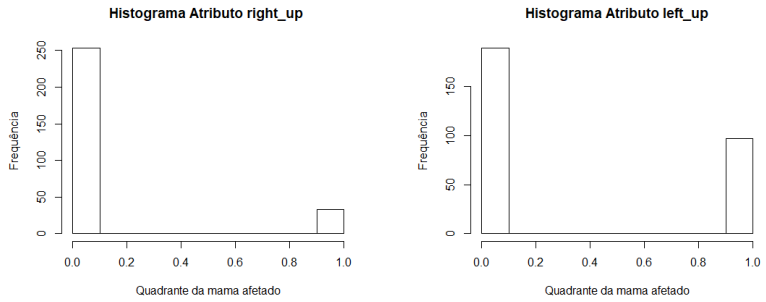

Figura 15: Histograma atributo right-up e left-up

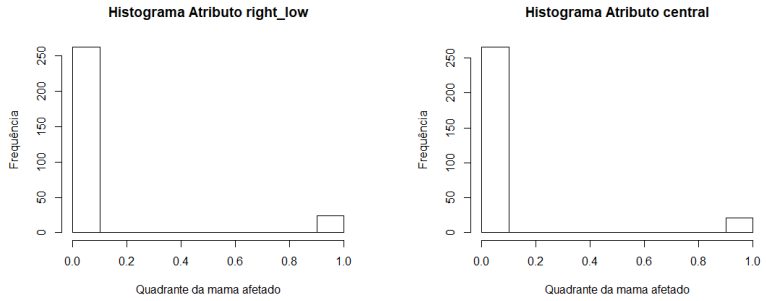

Figura 16: Histograma atributo right-low e central

Após a análise de todos os gráficos e comparando suas respectivas médias, é muito notório ver que os gráficos estão sempre concentrados em cima da média. E que os dados não seguem uma distribuição normal, então posteriormente esses dados terão que passar pela re-escala.

A média e a criação da tabela da figura 9, foi construída a partir do código fonte abaixo. Nele eu crio um novo data frame e atribuo os valores para eles, depois é importado uma biblioteca e depois eu exporto a imagem obtida.

```
#Tabela com a media de cada atributoo
media <- data.frame(
  "Atributos" = c("class","age","menopause","tumor.size",
                  "inv.nodes","node.caps","deg.malig",
                  "breast","left_low","right_up",
                  "left_up","right_low","central","irradiat"),
  "Media" = c(mean(base$class),
              mean(base$age),
              mean(base$menopause),
              mean(base$tumor.size),
              mean(base$inv.nodes),
              mean(base$node.caps),
              mean(base$deg.malig),
              mean(base$breast),
              mean(base$left_low),
              mean(base$right_up),
              mean(base$left_up),
```

```
                mean(base$right_low),
                mean(base$central),
                mean(base$irradiat)))

#biblioteca para salvar a tabela criada
library(gridExtra)

png("media.png", height = 50*nrow(media), width = 200*ncol(media))
grid.table(media)
dev.off()
```

Listing 6: Código fonte em R

Com o código abaixo, foi possível visualizar a criação de cada histograma que está representado as gráficos acima.

```
#Plotagem dos histogramas
hist(base$class,
        main = "Histograma Atributo class",
        ylab = "Frequencia",
        xlab = "Tipo do dado")

hist(base$age,
        main = "Histograma Atributo age",
        ylab = "Frequencia",
        xlab = "Idade do individuo")

hist(base$menopause,
        main = "Histograma Atributo menopause ",
        ylab = "Frequencia",
        xlab = "Indicacao da Menopausa")

hist(base$tumor.size,
        main = "Histograma Atributo tumor.size",
        ylab = "Frequencia",
        xlab = "Diametro do tumor")

hist(base$inv.nodes,
        main = "Histograma Atributo Inv.nodes",
        ylab = "Frequencia",
        xlab = "Numero de linfonodos axilares")

hist(base$node.caps,
        main = "Histograma Atributo node.caps",
        ylab = "Frequencia",
        xlab = "Penetracao do tumor")

hist(base$deg.malig ,
```

```
33          main = "Histograma Atributo deg.malig",
34          ylab = "Frequencia",
35          xlab = "Grau de malignidade do tumor")
36
37  hist(base$breast,
38          main = "Histograma Atributo breast",
39          ylab = "Frequencia",
40          xlab = "Mama em que o cancer pode ocorrer")
41
42  hist(base$irradiat,
43          main = "Histograma Atributo irradiat",
44          ylab = "Frequencia",
45          xlab = "Tipo do dado")
46
47  hist(base$left_low,
48          main = "Histograma Atributo left-low",
49          ylab = "Frequencia",
50          xlab = "Quadrante da mama afetado")
51
52  hist(base$right_up,
53          main = "Histograma Atributo right-up ",
54          ylab = "Frequencia",
55          xlab = "Quadrante da mama afetado")
56
57  hist(base$left_up,
58          main = "Histograma Atributo left-up",
59          ylab = "Frequencia",
60          xlab = "Quadrante da mama afetado")
61
62  hist(base$right_low,
63          main = "Histograma Atributo right-low",
64          ylab = "Frequencia",
65          xlab = "Quadrante da mama afetado")
66
67  hist(base$central,
68          main = "Histograma Atributo central",
69          ylab = "Frequencia",
70          xlab = "Quadrante da mama afetado")
```

Listing 7: Código fonte em R

6 Exploração dos dados através de medidas de distribuição;

A medida de distribuição também estão relacionadas com a média de cada valor, nesse caso não gerei gráficos, apenas uma tabela para poder fazer a análise de acordo com os gráficos gerados no Item anterior o 5.

Atributos	Momento	Curtose	Obliquidade
class	0.29720280	-1.22488733	0.88281304
age	2.66136486	-0.97284195	-0.10329948
menopause	0.92657343	-1.96027095	0.14644434
tumor.size	4.88111888	-0.03145662	0.05513091
inv.nodes	0.51748252	6.76553585	2.59157235
node.caps	0.19580420	0.32723116	1.52513946
deg.malig	1.04895105	-1.16928931	-0.07684805
breast	0.46853147	-1.99118452	0.12546326
left_low	0.38811189	-1.79759047	0.45679117
right_up	0.11538462	3.74965237	2.39510022
left_up	0.33916084	-1.54852255	0.67591066
right_low	0.08391608	6.93840418	2.98565025
central	0.07342657	8.61662968	3.25368483
irradiat	0.23776224	0.23776224	1.22553804

Figura 17: Tabela com as Medidas de distribuição

O momento, curtose e obliquidade e a criação da tabela da figura 17, foi construída a partir do código fonte abaixo. Nele eu crio um novo data frame(assim como foi feita na média do item 6) e atribuo os valores para eles, depois exporto a imagem obtida. Nesse código criei uma variável para cada atributo para receber o momento, a curtose e a obliquidade e assim depois inserir essas variáveis na tabela.

```
#Momento
mC  <- moment(base$class)
mA  <- moment(base$age)
mM  <- moment(base$menopause)
mT  <- moment(base$tumor.size)
mI  <- moment(base$inv.nodes)
mN  <- moment(base$node.caps)
mD  <- moment(base$deg.malig)
mB  <- moment(base$breast)
mLl <- moment(base$left_low)
mRr <- moment(base$right_up)
```

```
mLu <- moment(base$left_up)
mRl <- moment(base$right_low)
mCc <- moment(base$central)
mIr <- moment(base$irradiat)

#Curtose
cC  <- kurtosis(base$class)
cA  <- kurtosis(base$age)
cM  <- kurtosis(base$menopause)
cT  <- kurtosis(base$tumor.size)
cI  <- kurtosis(base$inv.nodes)
cN  <- kurtosis(base$node.caps)
cD  <- kurtosis(base$deg.malig)
cB  <- kurtosis(base$breast)
cLl <- kurtosis(base$left_low)
cRr <- kurtosis(base$right_up)
cLu <- kurtosis(base$left_up)
cRl <- kurtosis(base$right_low)
cCc <- kurtosis(base$central)
cIr <- kurtosis(base$irradiat)

# obliquidade
oC  <- skewness(base$class)
oA  <- skewness(base$age)
oM  <- skewness(base$menopause)
oT  <- skewness(base$tumor.size)
oI  <- skewness(base$inv.nodes)
oN  <- skewness(base$node.caps)
oD  <- skewness(base$deg.malig)
oB  <- skewness(base$breast)
oLl <- skewness(base$left_low)
oRr <- skewness(base$right_up)
oLu <- skewness(base$left_up)
oRl <- skewness(base$right_low)
oCc <- skewness(base$central)
oIr <- skewness(base$irradiat)

#Criacao da tabela de distribuicao
data <- data.frame(
  "Atributos" = c("class","age","menopause","tumor.size",
                  "inv.nodes","node.caps","deg.malig",
                  "breast","left_low","right_up",
                  "left_up","right_low","central","irradiat"),
  "Momento" = c(mC,mA,mM,mT,mI,mN,mD,mB,mLl,mRr,mLu,mRl,mCc,mIr),
  "Curtose" = c(cC,cA,cM,cT,cI,cN,cD,cB,cLl,cRr,cLu,cRl,cCc,mIr),
  "Obliquidade" = c(oC,oA,oM,oT,oI,oN,oD,oB,oLl,oRr,oLu,oRl,oCc,oIr))

```

```
59 png("distribuicao.png", height = 50*nrow(data), width = 200*ncol(data))
60 grid.table(data)
61 dev.off()
```

Listing 8: Código fonte em R

7 Identificação e separação do conjunto de teste, que será utilizado para testar o desempenho dos modelos – o conjunto de testes deve ser representativo e ter as características da população completa. Caso sua base de dados já tenha o conjunto de teste definido, analisar se este segue as características do conjunto de treinamento;

A divisão da base foi utilizando a estratificação dos dados,por conta da base de dados ser desbalanceada foi preciso utilizar essa abordagem afim de manter a mesma proporção do atributo de saída definido no item 1, ou seja, manter 70.3% e 29,7%, com isso destino 70.3% para a base de treinamento e 29.7% destinada para base de teste.

Com o código abaixo, foi possível realizar a divisão da base de dados, mantendo a mesma proporção.

```
1  #Biblioteca para divisao da base
2  library(caTools)
3
4  #Porcao da base de dados
5  set.seed(1)
6
7  #Quanto maior for o SplitRatio maior e a base de treinamento
8  divisao <- sample.split(base$class, SplitRatio = 0.703)
9
10 #Criando a base de teste e de treinamento
11 # 201 dados - 70.3% da base
12 base_treinamento <- subset(base, divisao == TRUE)
13
14 #Criando a variavel de teste
15 #85 dados - 29.7 % da base
16 base_teste <- subset(base, divisao == FALSE)
```

Listing 9: Código fonte em R

8 Identificação e eliminação de atributos não necessários;

No caso desta base apenas o atributo ***breast.quad*** será eliminada, mas isso só vai ocorrer depois de fazer o passo 13 onde ocorre a conversão dos dados simbólicos para numéricos, pois com esse atributo foi necessário fazer a divisão dos valores transformar eles usando a binarização, então após afazer isso o atributo breast.quad não é mais necessário para minha base.

Abaixo o código de como é feita a eliminação desse atributo:

```
1 base$breast.quad = NULL
```

Listing 10: Código fonte em R

E o restante dos atributos não são eliminados, pois como se trata de uma base relacionada ao câncer de mama e não se tem muito conhecimento do que é relevante ou não para se tirar da base, o método abordado foi a não eliminação de atributos, pois se por acaso tirar um atributo que é muito importante para detecção do câncer isso vai interferir no resultado do algoritmo.

9 Identificação e eliminação de exemplos não necessários;

O caso da eliminação de exemplos segue o mesmo padrão da eliminação de atributos, é difícil tirar uma conclusão de quais dados deve-se remover da base, sendo que todos devem ter sua certa importância, quando se trata mais de fatores biosóficas do que computacionais.

10 Análise e aplicação de técnicas de amostragem de dados (caso não seja necessário, analisar o porquê);

A amostragem de dados é muito importante pois ajuda os na criação de uma amostra precisa. Porém por conta dessa base de dados ser uma base muito pequena, pois é uma técnica aplicada mais para bases de dados grandes com mais de 1000 instâncias por exemplo e não é o caso dessa base de dados. Por isso essa base não precisou realizar a amostragem dos dados.

11 Identificação e aplicação de técnicas para minimizar problemas de desbalanceamento (caso não seja necessário, analisar o porquê);

Com o código abaixo, foi possível corrigir o problema de balanceamento, nele foi feito o seguinte, primeiro a criação de uma variável auxiliar para receber TRUE - para todos os dados que são iguais a 1, ou seja da classe minoritária e FALSE para o 0 - classe majoritária, depois

foi criada uma base auxilia, que recebe apenas os valores que são TRUE, ou seja, da classe minoritária e por fim é feita uma junção dessa base auxiliar com a base de treinamento, e agora tem uma base de treinamento balanceada.

```
#variavel auxiliar, para armazenar os TRUE e FALSE
aux <- base_treinamento[,1]==1

#Base auxiliar para receber apenas os valores TRUE
base_aux <- subset(base_treinamento,aux == TRUE)

#Base de treinamentos recebe a combinacao dela mesmo e da base auxiliar
base_treinamento <- rbind(base_aux,base_treinamento)
```

Listing 11: Código fonte em R

12 Limpeza de dados:

(a) Identificação e eliminação de ruídos ou outliers;

(b) Identificação e eliminação de dados inconsistentes;

(c) Identificação e eliminação de dados redundantes;

(d) Identificação e resolução de dados incompletos (ausentes) utilização de alguma técnica de preenchimento e justificar;

A) Os atributos que tem a presença de outliers são: ***age, tumor.size, inv.nodes*** neste caso como já se tem a divisão da base de dados, a identificação e a correção dos outliers estão a partir de agora estão relacionados a Base de ***TREINAMENTO***. Antes de fazer a correção dos ruídos, utilizando o código abaixo foi possível criar a figura 18 com o boxplot de todos os atributos da base de treinamento sem a correção dos ruídos.

```
boxplot(base_treinamento,
        main = "Boxplot com os outliers",
        ylab = "Valores",
        xlab = "Atributos",
        pch = 16)
```

Listing 12: Código fonte em R

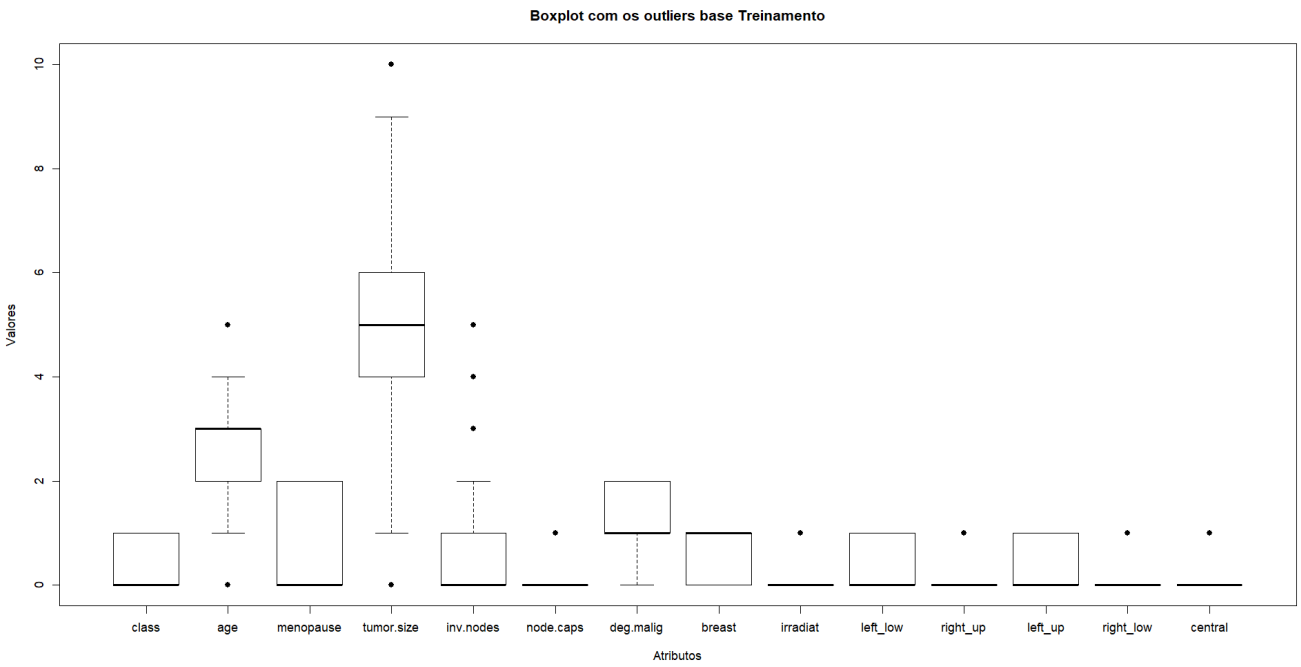

Figura 18: Boxplot com todos atributos e com todos os outliers

Agora para a correção dos ruídos foi foi utilizado o seguinte código:

```
#Atributo age
summary(base_treinamento$class)

#Q3 - Q1
IQR_age = 1
x_age = mean(base_treinamento$age)
Ls_age = x_age + 1.5*IQR_age
Li_age = x_age - 1.5*IQR_age

table(base_treinamento$age)

base_treinamento$age[base_treinamento$age == 0] <- Li_age
base_treinamento$age[base_treinamento$age == 5] <- Ls_age

#Atributo tumor.size
summary(base_treinamento$tumor.size)

IQR_tumor.size = 6 - 4
x_tumor.size = mean(base_treinamento$tumor.size)
Ls_tumor.size = x_tumor.size + 1.5*IQR_tumor.size
Li_tumor.size = x_tumor.size - 1.5*IQR_tumor.size

table(base_treinamento$tumor.size)

base_treinamento$tumor.size[base_treinamento$tumor.size == 0] <- Li_tumor.size
```

```r
base_treinamento$tumor.size[base_treinamento$tumor.size == 10] <- Ls_tumor.size

# Atributo inv.nodes
summary(base_treinamento$inv.nodes)

IQR_inv.nodes = 1
x_inv.nodes = mean(base_treinamento$inv.nodes)
Ls_inv.nodes = x_inv.nodes + 1.5*IQR_inv.nodes

table(base_treinamento$inv.nodes)

base_treinamento$inv.nodes[base_treinamento$inv.nodes >= 3] <- Ls_inv.nodes

boxplot(base_treinamento,
        main = "Boxplot sem os outliers",
        ylab = "Valores",
        xlab = "Atributos",
        pch = 16)
```

Listing 13: Código fonte em R

Esse código acima faz o seguinte : Para cada atributo com outliers ele faz uma análise da amplitude interquartil utilizando sigla IQR acrescido do nome do atributo, para analisar a amplitude é feita $IQR = Q3 - Q1$, após isso analiso o limite superior chamado de Ls acrescido do nome do atributo e $Ls = media + 1.5 * IQR$, essa média é dado de acordo com cada atributo, após encontrar o limite superior, faço o mesmo para o limite inferior $Ls = media - 1.5 * IQR$, depois dou um comando $table(atributo)$ e vejo no manualmente olhando para o gráfico e para os valores que tenho retornados do comando table, após isso substituo esse valor a onde eu encontro os outliers, se for no limite inferior troca o valor para o valor calculado no Li e o mesmo se for o limite superior.

Após o código acima, foi possível gerar uma imagem, que pode ser vista na figura 19, sem a presença dos ruídos. O que mostra agora que a base não tme mais dados que são completamente longe da média dos dados.

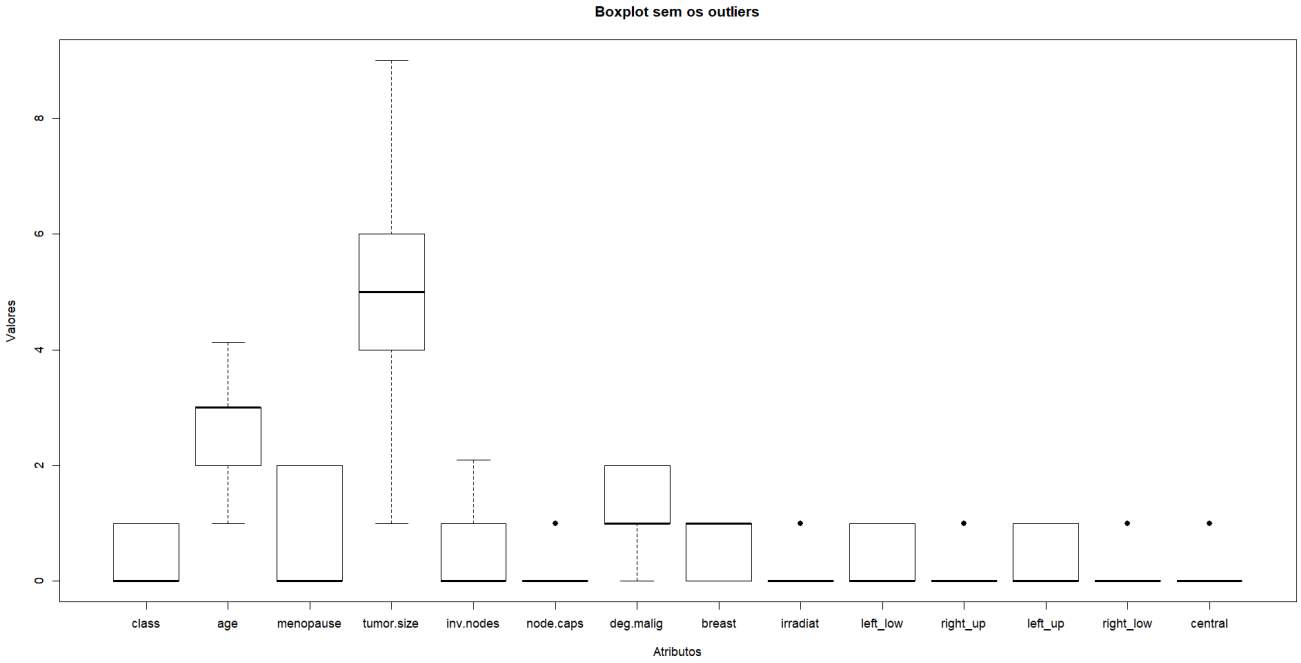

Figura 19: Boxplot com todos atributos sem os outliers

B) Os dados inconsistentes encontrados foram os dos atributos ***breast.quad*** e ***node.caps***, como a base de dados é muito pequena optei por não eliminar esses dados e sim realizar a substituição desses dados pela moda, esse processo é realizado na letra D desse mesmo tópico.

C)A redundância de dados está relacionada a ocorrência do mesmo conjunto de dados ser armazenado em dois ou mais locais. O que não acontece nesse base de dados. Não se tem redundância de dados. Em nenhum momento os atributos se repetem e é possível ver isso no código abaixo:

```
1 #Verifica colunas duplicadas e cria uma lista
2 colunas_duplicadas <- duplicated(as.list(base))
3
4 print(colunas_duplicadas)
5
6 #Mostra o nome das colunas duplicadas
7 colnames(base[colunas_duplicadas])
```

Listing 14: Código fonte em R

O código acima faz a criação de uma lista com todas as colunas duplicadas e depois na linha 4 tem um print para ver o retorno dessa lista, ela retorna tudo ***false***, o que já mostra que não se tem atributos repetidos e para confirmar na linha 7 utilizando a função colnames para mostrar o nome das colunas duplicadas, e ela da um retorno de 0, ou seja confirma novamente a não existência de dados redundantes.

D) Por conta da base de dados sem composta de dados simbólicos, o método mais eficiente para resolução dos dados ausentes foi utilizando a moda, isso antes dos dados serem convertidos para numéricos e o motivo de não utilizar a média por exemplo é por conta de que se eu tenho valores que vão de $0-1-2-4-5$ e a media cai em 2.5 fica difícil encaixar ela em algum lugar, sendo que os valores são definidos como inteiros, ai a média é a melhor solução nesse caso.

```
base$breast.quad[base$breast.quad == "?"] <- "left_low"
base$node.caps[base$node.caps =="?"] <- "no"
```

Listing 15: Código fonte em R

13 Identificação e conversão dos tipos de dados (caso não seja necessário, analisar o porquê). Os tipos de conversão que podem ser utilizados são:

(a) Conversão de tipos (simbólico para numérico, ordinal para numérico, nominal para binário, numérico para ordinal);

(b) Normalização dos dados (re-escala ou padronização);

A) Como todos os atributos da minha base são, simbólicos, tenho que converter eles para numérico, para seja possível realizar algumas operações. Então fiz uma tabela para explicar melhor quais foram as conversões realizadas em cada atributo.

Os atributos **menopause** e **breast.quad** eu destaquei para explicar melhor sobre esses dois, pois os outros seguem um padrão de ordem que varia de 0 a quantidade de valores que tem que dentro de cada atributo.

No atributo **menopause**, a conversão dele se deu da seguinte maneira: *premeno*: está relacionado a pré-menopausa, por isso decidi atribuir o valor de **0** para ele, o *It40*: está relacionado a idade antes dos 40 anos, então para ela foi definido o valor de **1** e por fim o valor *ge40*: está relacionado a idade após 40 anos, então foi definido o valor de **2** para ela.

No atributo **premeno**, foi utilziado a técnica de *One-Hot*, que é o processo de converter uma variável categórica com várias categorias em várias variáveis, cada uma com um valor de 1 ou 0. Foi criada várias outras instâncias com os valores que tinha no atributo, ou seja dentro do atributo ***breast.quad*** tinha os valores do quadrante da mama afetado ***left-low***, ***right-up***, ***left-up***, ***right-low*** e ***central***, agora cada valor do quadrante é um novo atributo da base.

Atributos	Conversão
class	0 - 1
age	0 - 1 - 2 - 3 - 4 - 5
menopause	premeno:0 - lt40:1 - ge40:2
tumor.size	0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10
inv.nodes	0 - 1 - 2 - 3 - 4 - 5 -6
node.caps	0 - 1
deg.malig	0 - 1 - 2
breast	0 - 1
breast.quad	Divide em instâncias
irradiat	0 - 1

Com o código abaixo é possível ver como foi realizada a conversão dos dados, vou destacar alguns pontos do código, da linha 28 até a 36 faço novamente uma conversão para numérico, acontece que quando eu uso o factor para conversão ele acaba convertendo os valores em números, porém o estado dele ainda continua sendo de uma variável categórica, para isso utilizo o método as.numeric(as.character(atributo)), para que os dados agora sejam de fato convertidos.

```
1  #A) Conversao dos atributos qualitativos para quantitativos:
2
3  #Com apenas 2 tipos
4  base$class = factor(base$class,
5      levels = c('no-recurrence-events','recurrence-events'), labels = c(0,1))
6  base$irradiat = factor(base$irradiat, levels = c('no','yes'), labels = c(0,1))
7  base$breast = factor(base$breast, levels = c('left','right'), labels = c(0,1))
8  base$node.caps = factor(base$node.caps, levels = c('no','yes'), labels = c(0,1))
9
10 #Com 3 tipos
11 base$deg.malig = factor(base$deg.malig, levels = c('1','2','3'),labels = c
       (0,1,2))
12
13 #Com mais de 3
14 base$age = factor(base$age,
15     levels = c('20-29','30-39','40-49','50-59','60-69','70-79'),
16     labels = c(0,1,2,3,4,5))
17
18 base$menopause = factor(base$menopause,
19     levels = c('premeno','lt40','ge40'),labels = c(0,1,2))
20 base$tumor.size = factor(base$tumor.size,
21     levels = c('0-4','5-9','10-14','15-19','20-24','25-29','30-34','35-39','
           40-44','45-49','50-54'),
22     labels = c(0,1,2,3,4,5,6,7,8,9,10))
23 base$inv.nodes = factor(base$inv.nodes,
```

```
    levels = c('0-2','3-5','6-8','9-11','12-14','15-17','24-26'),
    labels = c(0,1,2,3,4,5,6))

#Converter para numerico
base$class <- as.numeric(as.character(base$class))
base$age <- as.numeric(as.character(base$age))
base$menopause <- as.numeric(as.character(base$menopause))
base$tumor.size <- as.numeric(as.character(base$tumor.size))
base$inv.nodes <- as.numeric(as.character(base$inv.nodes))
base$node.caps <- as.numeric(as.character(base$node.caps))
base$deg.malig <- as.numeric(as.character(base$deg.malig))
base$breast <- as.numeric(as.character(base$breast))
base$irradiat <- as.numeric(as.character(base$irradiat))

#Criando novas colunas
base$left_low <- 0
base$left_low[base$breast.quad == "left_low"] <- 1

base$right_up <- 0
base$right_up[base$breast.quad == "right_up"] <- 1

base$left_up <- 0
base$left_up[base$breast.quad == "left_up"] <- 1

base$right_low <- 0
base$right_low[base$breast.quad == "right_low"] <- 1

base$central <- 0
base$central[base$breast.quad == "central"] <- 1

#verificar se deu certo
base[is.na(base$inv.nodes)]
```

Listing 16: Código fonte em R

B) A técnica utilizada foi a reescala, ou seja deixei todos os dados da base de teste e treinamento na mesma escala, optei por deixar a base original com os dados normais(sem reescala), optei por isso para não ter que refazer os gráficos executados nos itens anteriores.

Com o código abaixo, foi possível fazer a reescala de todos atributos exceto o do atributo de saída. O que esse código faz é basicamente pegar a posição da base a partir do 2, pois a contagem aqui começa em 1 e ir até o último atributo da classe que termina em 14.

```
base_treinamento[,2:14] = scale(base_treinamento[,2:14])
base_teste[,2:14] = scale(base_teste[,2:14])
```

Listing 17: Código fonte em R

14 Análise e aplicação de alguma técnica para redução de dimensionalidade – pesquisar alguma técnica utilizada na literatura e aplicar;

A técnica de dimensionalidade está relacionada como o próprio nome já diz a uma redução dos dados,com isso é possível aumentar a qualidade, mas ao mesmo tempo minimizando a perda de informação. E essa técnica faz isso criando novas variáveis, que não são correlacionadas e que possam maximizar sucessivamente a medida de dispersão.

Para encontrar essas novas variáveis, os componentes principais, são definidas pelo conjunto de dados em mãos, o que torna o PCA uma técnica de análise de dados adaptativa, ou seja, o mais bacana é que sempre o usuário pode variar o número de variáveis que desejo ter.

O código abaixo representa toda essa técnica de PCA:

```
# 0 - no-recurrence-events
# 1 - recurrence-events

#Aplicar reducao
library('caret')

pca = preProcess(x = base_treinamento[-1], method = 'pca', pcaComp = 4)
base_treinamento_pca = predict(pca, base_treinamento)
base_teste_pca = predict(pca, base_teste)

#Testes - Extra

#Teste Com toda a base
classificador = naiveBayes(x = base_treinamento[-1], y = base_treinamento$class)

previsoes = predict(classificador, newdata = base_teste[-1])
# 64,7% de acertos
matriz_confusao = table(base_teste[,1], previsoes)
print(matriz_confusao)
#analisa a matriz de confusao e tras varias metricas
confusionMatrix(matriz_confusao)
#Accuracy : 0.6471


#Teste com apenas 4 atributos
classificador_pca = naiveBayes(x = base_treinamento_pca[-1], y = base_
    treinamento_pca$class)

previsoes_pca = predict(classificador_pca, newdata = base_teste_pca[-1])
# 70,6% de acertos
matriz_confusao_pca = table(base_teste_pca[,1], previsoes_pca)
```

```
31 print(matriz_confusao_pca)
32 #analisa a matriz de confusao e tras varias metricas
33
34 confusionMatrix(matriz_confusao_pca)
35 #Accuracy : 0.7059
```

Listing 18: Código fonte em R

Esse código acima basicamente faz o seguinte:

- Cria o pca usando a função preProcess

  Essa parte é feita em cima da base de treinamentos[-1] pois é necessário retirar o atributo de saída e atribui 4 atributos aleatórios.

- Atribuição para as bases

  Após criar o pca é preciso fazer as bases receberem seus novos valores e atributos. No caso foi criada uma nova variável mas com o intuito de fazer comparações ao mesmo tempo, sem ter que ficar trocando os valores, comparações no sentindo de classificação e previsão e matriz de confusão, é possível até ver que esses nomes vão se repetir, o motivo foi mais para comparação dos acertos e da acurácia.

- Testes

  Após isso, foi realizado teste de previsões e de matriz de confusão para toda a base de dados de teste para testar se o algoritmo está bom.

15 Conclusão

Neste trabalho foram abordadas várias técnicas de pré-processamento de dados, e com tudo que foi realizado, foi possível perceber o quão importante é a pré analise dos dados, pois se fizemos alguma análise errada isso pode repercutir em erros no trabalho.

No passo 14 foi possível ver como realmente está a porcentagem de acertos do algoritmo, com a técnica de dimensionalidade e sem ela, ainda não se pode concluir de fato nada, mas aparentemente o algoritmo está sendo acertando bem, porém como a base de dados é pequena pode ter ocorrido dele ter aprendendo mais de um determinado dado, ou seja pode ter ocorrido um Overfitting, pois ele pode ter tido um bom desempenho nos dados de treinamento porém ele pode ter generalizado mal os outros dados. A expectava é que no próximo trabalho consiga tirar uma conclusão bem melhor sobre essa base de dados.