Relatório do 2º Trabalho

Aprendizado de Máquina
Inteligência Artificial
2016/2

Igor Guilherme Bianchi
558400
Felipe Churuyuki Chinen
496235

- ## Árvore de decisão para classificação

O conjunto de dados escolhido para a aplicação da árvore de decisão foi o “Car Evaluation” do repositório http://archive.ics.uci.edu/ml/. A separação dos dados de aprendizado e de teste se deu de forma randômica respeitando a proporção de 25% para teste e 75% para aprendizado. O conjunto possui 1849 instâncias, o arquivo de teste ficou com 463 instâncias (representando 25,04%) e o de aprendizado com 1386 instâncias (representando 74,96%). O objetivo é avaliar carros a partir de suas características estruturais e econômicas.

Os tipos de atributos e seus possíveis valores são:

- **Buying**: vhigh, high, med, low.
- **Maint**: vhigh, high, med, low.
- **Doors**: 2, 3, 4, 5more.
- **Persons**: 2, 4, more.
- **Lug_boot**: small, med, big.
- **Safety**: low, med, high.
- **Class**: unacc, acc, good, vgood.

Primeiro abrimos o arquivo de aprendizado no nosso script com o seguinte comando:

```
dados <- read.table("D:/Desktop/trabIA/trab2/data_car.csv", header = TRUE, sep=",")
```

Aplicamos então a árvore de decisão no conjunto de dados:

```
modelo_ad <- rpart(Class ~ buying + maint + doors + persons + lug_boot + safety,
                   data = dados, method = "class", parms = list(split = "Information"))
```

A classe do conjunto é justamente a que tem esse nome, “Class”.
O gráfico é gerado:

```
plot_ad <- rpart.plot(modelo_ad, type = 3, cex = 0.5)
```

A visualização da árvore pode ser feita na página 4 deste relatório.
Leitura dos dados de teste:

```
dados_teste <- read.table("D:/Desktop/trabIA/trab2/data_carTest.csv",
                          header = TRUE, sep =",")
```

Previsão dos dados de teste:

```
y_estimado <- predict(modelo_ad, dados_teste, "class")
```

Apresentando o seguinte resultado:

```
> y_estimado
    1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20    21
vgood   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc unacc   acc   acc
   22    23    24    25    26    27    28    29    30    31    32    33    34    35    36    37    38    39    40    41    42
unacc   acc   acc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc unacc unacc unacc unacc
   43    44    45    46    47    48    49    50    51    52    53    54    55    56    57    58    59    60    61    62    63
unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc
   64    65    66    67    68    69    70    71    72    73    74    75    76    77    78    79    80    81    82    83    84
unacc unacc unacc unacc unacc unacc unacc unacc   acc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc unacc
   85    86    87    88    89    90    91    92    93    94    95    96    97    98    99   100   101   102   103   104   105
  acc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
  106   107   108   109   110   111   112   113   114   115   116   117   118   119   120   121   122   123   124   125   126
  acc unacc   acc   acc unacc   acc   acc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc
  127   128   129   130   131   132   133   134   135   136   137   138   139   140   141   142   143   144   145   146   147
unacc unacc unacc unacc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc   acc unacc   acc   acc unacc   acc
  148   149   150   151   152   153   154   155   156   157   158   159   160   161   162   163   164   165   166   167   168
  acc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc   acc
  169   170   171   172   173   174   175   176   177   178   179   180   181   182   183   184   185   186   187   188   189
vgood unacc   acc vgood unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc unacc unacc unacc unacc unacc unacc unacc
  190   191   192   193   194   195   196   197   198   199   200   201   202   203   204   205   206   207   208   209   210
unacc unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc unacc
  211   212   213   214   215   216   217   218   219   220   221   222   223   224   225   226   227   228   229   230   231
unacc unacc unacc unacc unacc unacc unacc unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc   acc   acc unacc   acc
  232   233   234   235   236   237   238   239   240   241   242   243   244   245   246   247   248   249   250   251   252
vgood unacc   acc vgood unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc  good unacc  good vgood unacc  good
  253   254   255   256   257   258   259   260   261   262   263   264   265   266   267   268   269   270   271   272   273
vgood unacc   acc  good unacc  good vgood unacc  good vgood unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc
  274   275   276   277   278   279   280   281   282   283   284   285   286   287   288   289   290   291   292   293   294
 good unacc  good vgood unacc  good vgood unacc   acc  good unacc  good vgood unacc  good vgood unacc unacc unacc unacc unacc
  295   296   297   298   299   300   301   302   303   304   305   306   307   308   309   310   311   312   313   314   315
unacc unacc unacc unacc unacc   acc  good unacc  good vgood unacc  good vgood unacc   acc  good unacc  good vgood unacc  good
  316   317   318   319   320   321   322   323   324   325   326   327   328   329   330   331   332   333   334   335   336
vgood unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc   acc  good unacc  good vgood unacc  good vgood unacc   acc
  337   338   339   340   341   342   343   344   345   346   347   348   349   350   351   352   353   354   355   356   357
 good unacc  good vgood unacc  good vgood unacc unacc unacc unacc unacc   acc  good unacc  good vgood unacc  good vgood unacc
  358   359   360   361   362   363   364   365   366   367   368   369   370   371   372   373   374   375   376   377   378
  acc  good unacc  good vgood unacc  good vgood unacc   acc   acc unacc   acc vgood unacc   acc vgood unacc unacc unacc unacc
  379   380   381   382   383   384   385   386   387   388   389   390   391   392   393   394   395   396   397   398   399
unacc unacc unacc unacc unacc unacc   acc unacc   acc   acc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
  400   401   402   403   404   405   406   407   408   409   410   411   412   413   414   415   416   417   418   419   420
  acc unacc   acc   acc unacc   acc   acc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc unacc unacc
  421   422   423   424   425   426   427   428   429   430   431   432   433   434   435   436   437   438   439   440   441
unacc unacc unacc   acc   acc unacc   acc unacc unacc   acc unacc   acc   acc unacc   acc   acc unacc unacc unacc unacc unacc
  442   443   444   445   446   447   448   449   450   451   452   453   454   455   456   457   458   459   460   461   462
unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
  463
unacc
Levels: acc good unacc vgood
```

Para calcular a quantidade de acertos, é necessária a criação de um loop para verificar a classe de todas as instâncias da previsão e comparar com a classe dos dados de teste originais:

```
c <- 0

for(i in 1:length(y_estimado)){
  if(as.vector(y_estimado)[i] == as.vector(dados_teste[['Class']])[i]){
    c <- c + 1
  }
}
```

Dessa forma temos a quantidade de acertos e conseguimos calcular a precisão:

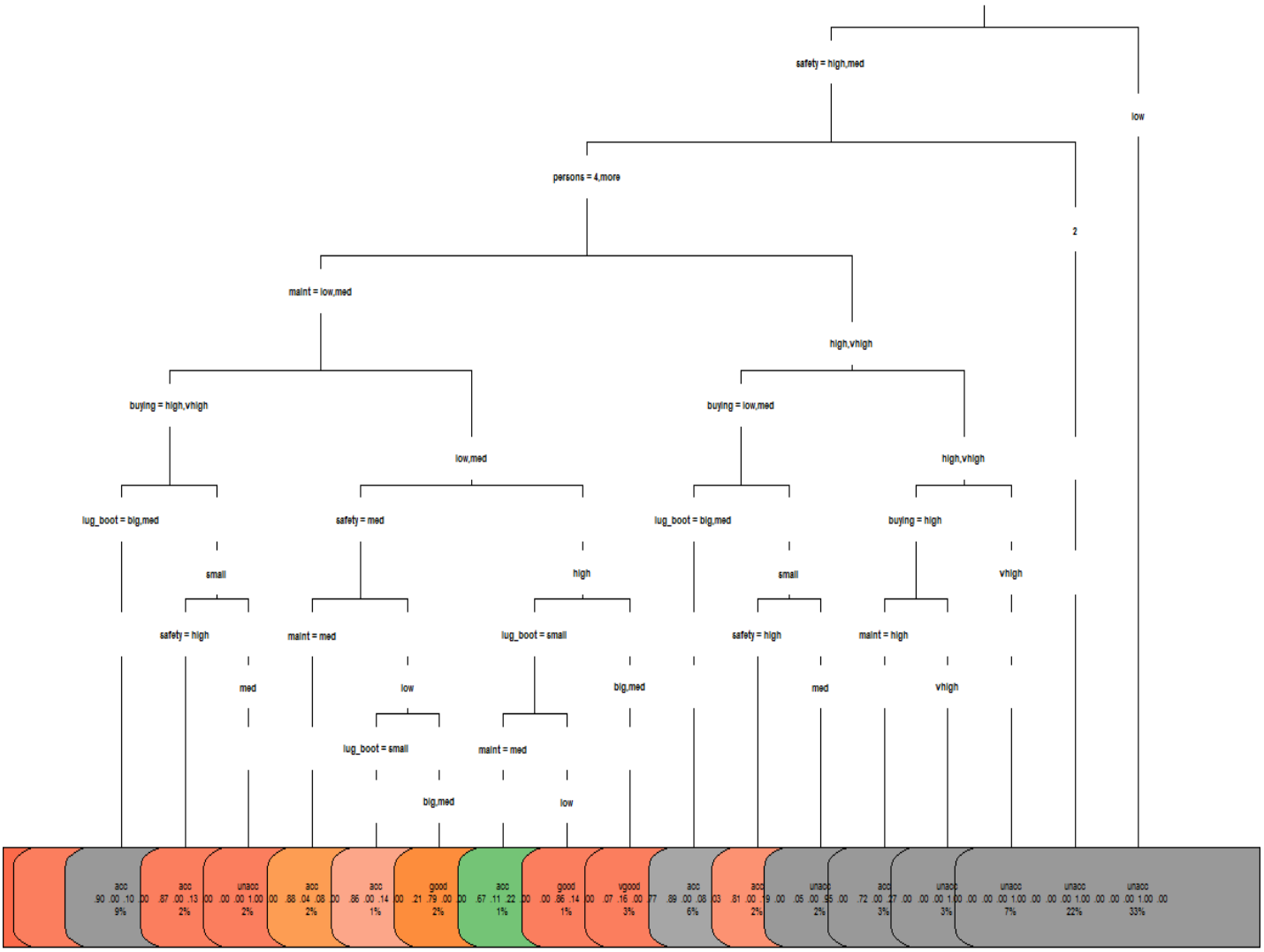
```
precisao <- c/length(y_estimado)
```

E a taxa de erros:

```
erro <- 1 – precisao
```

Que nos dão os seguintes resultados:

- **Taxa de erros**: 10,15%
- **Precisão**: 89,85%



Visualização da árvore de decisão.

- **Árvore de regressão**

O conjunto de dados escolhido para a aplicação da árvore de decisão foi o “Abalone Data Set” do repositório http://archive.ics.uci.edu/ml/. A separação dos dados de aprendizado e de teste se deu de forma randômica respeitando a proporção de 25% para teste e 75% para aprendizado. O conjunto possui 4177 instâncias, o arquivo de teste ficou com 1044 instâncias (representando 24,99%) e o de aprendizado com 3133 instâncias (representando 75,01%). O objetivo é saber a idade do Abalone a partir de suas características físicas, já que o trabalho para verificar a partir de suas argolas (“rings”) é extremamente trabalhoso.

Seus atributos e seus escopos são:

- **Sex** / nominal / -- / M, F, and I (infant)
- **Length** / contínuo / mm / Maior medição da casca
- **Diameter** / contínuo / mm / perpendicular ao comprimento
- **Height** / contínuo / mm / com carne na casca
- **Whole_weight** / contínuo / gramas / peso do abalone inteiro
- **Shucked_weight** / contínuo / gramas / peso da carne
- **Viscera_weight** / contínuo / gramas / peso do intestino (após sangramento)
- **Shell_weight** / contínuo / grama s / peso após secar
- **Rings** / inteiro / -- / +1.5 dá a idade em anos

Primeiro abrimos o arquivo de aprendizado no nosso script com o seguinte comando:

dados <- read.table("D:/Desktop/trabIA/trab2/data_abalone.csv", header = TRUE, sep=",")

Aplicamos então a árvore de regressão no conjunto de dados:

modelo_ad <- rpart(rings ~ sex + length + diameter + heigth + whole_weight + shucked_weight + viscera_weight + shell_weight,
data = dados, method = "anova", parms = list(split = "Information"))

A classe é dada

O gráfico é gerado:

rpart.plot(modelo_ad, type = 3,cex = 0.55)

A visualização da árvore pode ser feita na página 7 deste relatório.

Leitura dos dados de teste:

dados_teste <- read.table("D:/Desktop/trabIA/trab2/data_abaloneTest.csv", header = TRUE, sep =",")

Previsão dos dados de teste:

y_estimado <- predict(modelo_ad, dados_teste, "vector")

A seguir, a imagem parcial da previsão (a quantidade de dados era muito grande, não caberia em uma imagem somente):

```
> y_estimado
        1         2         3         4         5         6         7         8         9
 7.880081  9.283379  9.283379  9.283379  9.283379  7.880081 11.807377 10.866920 10.282051
       10        11        12        13        14        15        16        17        18
11.807377 11.807377 11.807377 11.807377  9.281690 11.807377 11.558201 10.282051 11.558201
       19        20        21        22        23        24        25        26        27
11.558201  9.283379 10.282051  9.283379 11.807377  9.283379  9.283379 14.477528  7.880081
       28        29        30        31        32        33        34        35        36
 9.281690  9.281690 10.866920 14.477528 11.807377 11.807377  9.283379 13.335714 10.866920
       37        38        39        40        41        42        43        44        45
13.335714  9.283379  9.283379  7.880081  7.880081 13.335714  7.880081 10.866920  5.920821
       46        47        48        49        50        51        52        53        54
 9.283379 10.866920 10.866920 13.335714  9.283379  7.880081  9.283379  9.283379  9.283379
       55        56        57        58        59        60        61        62        63
10.866920 10.866920  9.283379  7.880081  9.283379  9.283379  7.880081 11.807377 13.335714
       64        65        66        67        68        69        70        71        72
11.807377 13.335714 14.477528  5.920821  7.880081  5.920821  5.920821  5.920821  5.920821
       73        74        75        76        77        78        79        80        81
 5.920821 11.807377  5.920821  5.920821  5.920821  5.920821  5.920821  5.920821  7.880081
       82        83        84        85        86        87        88        89        90
 5.920821  7.880081  5.920821 14.477528 10.282051 10.282051 10.282051 14.477528 14.477528
       91        92        93        94        95        96        97        98        99
11.807377 11.558201 10.282051 11.807377 11.807377 13.335714  5.920821  5.920821  9.283379
      100       101       102       103       104       105       106       107       108
 9.281690  9.283379 13.335714  7.880081  9.283379  5.920821  9.283379  5.920821  5.920821
      109       110       111       112       113       114       115       116       117
 7.880081  7.880081  9.283379  9.283379  5.920821  5.920821  5.920821  5.920821 10.866920
      118       119       120       121       122       123       124       125       126
10.866920  9.283379  7.880081 10.866920  9.283379  9.283379 11.807377  5.920821  7.880081
      127       128       129       130       131       132       133       134       135
 9.283379  9.283379  7.880081  9.283379  5.920821  9.283379  5.920821  7.880081 10.866920
      136       137       138       139       140       141       142       143       144
 5.920821 14.477528 11.807377  5.920821  9.283379  7.880081  5.920821  9.283379  7.880081
      145       146       147       148       149       150       151       152       153
 9.283379  5.920821  5.920821  7.880081  5.920821  9.283379  5.920821 11.807377 14.477528
      154       155       156       157       158       159       160       161       162
14.477528 11.807377 11.558201 13.335714  9.283379 10.282051 14.477528 11.558201 10.282051
      163       164       165       166       167       168       169       170       171
14.477528 11.558201 14.477528 11.558201 10.282051 10.282051 10.282051 11.558201 11.558201
```

Para calcular o erro quadrático, é necessária a criação de um loop para subtrair a classe de todas as instâncias da previsão e somar com o anterior:

```
c <- 0
for(i in 1:length(y_estimado)){
  c <- c + (as.vector(y_estimado)[i] - as.vector(dados_teste[['rings']])[i])^2
}
erro <- c/length(y_estimado)
```

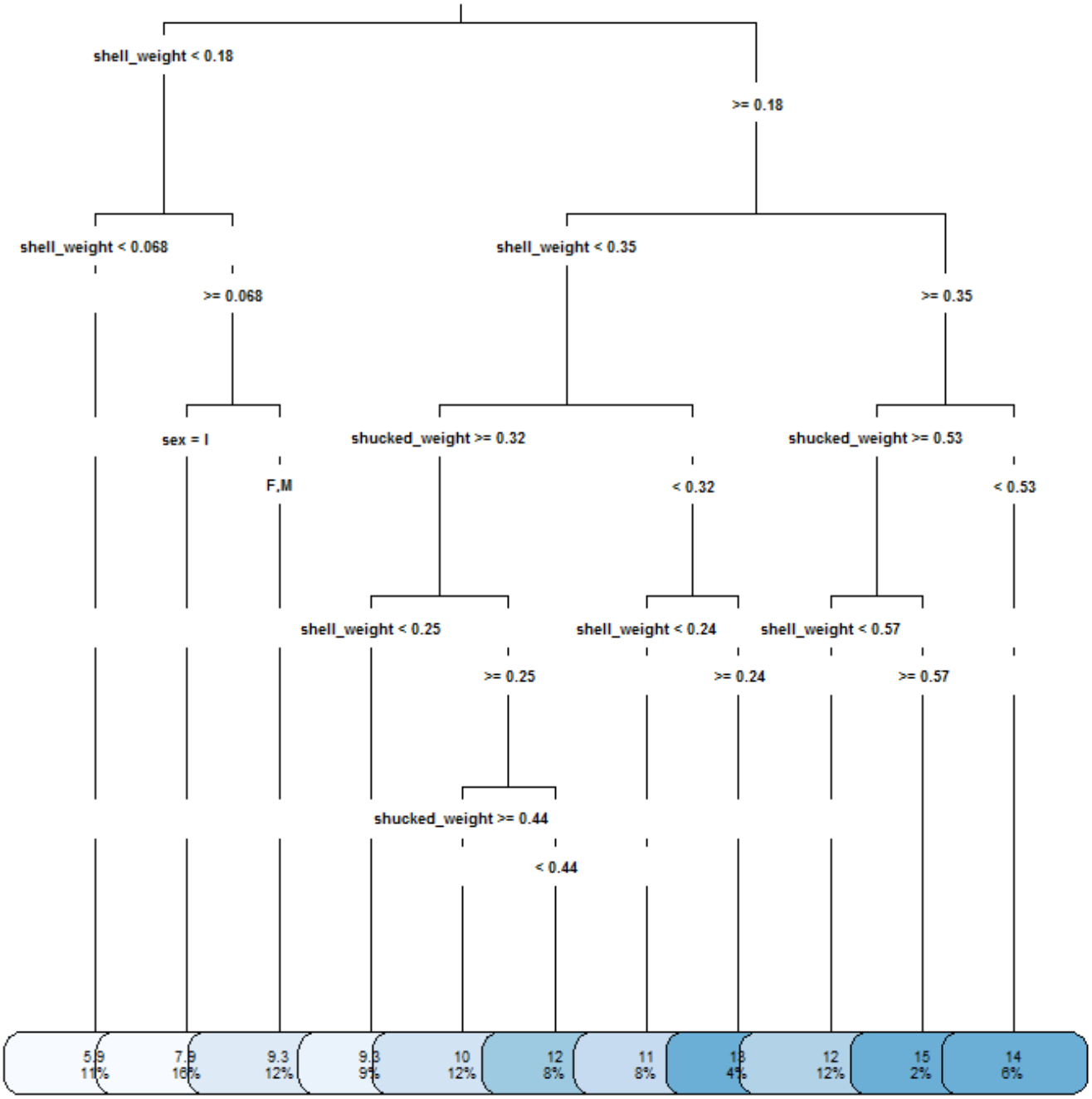
Para o cálculo da distância absoluta média o procedimento é semelhante, a diferença está na ausência da potência 2:

```
a <- 0
for(i in 1:length(y_estimado)){
  a <- a + abs(as.vector(y_estimado)[i] - as.vector(dados_teste[['rings']])[i])

}
distancia <- a/length(y_estimado)
```

Que nos dão os seguintes resultados:

- **Erro quadrático**: 5,47
- **Distância absoluta média**: 1,69



Visualização da árvore de regressão.