In R, use the[**CPU Performance**](http://www.biz.uiowa.edu/faculty/jledolter/datamining/cpu.csv) dataset to build the classification trees and the Naive Bayes Analysis we built in class (you may reference the R code you were given this week).  Then compare the two different kinds of analysis.  If you haven't done so yet, research each of the trees and Naive Bayes and explain which one produces better-quality results FOR YOUR DATASET and why?

Remember to

* First determine what the class attribute (i.e. the answer to the "big question") is
* Transform data into factors as needed
* CITE YOUR SOURCES (including URLs) and INTERPRET YOUR RESULTS.

Taking CPU performance as dataset, which is been loaded into R using cpu.csv and been executed where system are been provided by 24 companies, and now identifying the variance of attributes of best cpu performance given by which system. By seeing the reference of <https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/machine.names>, I identified that PRP is the linear regression in the dataset where PRP is continuously valued, so taking it in consideration as class attribute and also there are no none values in the dataset, so it was been categorized in this way. Dataset is been classified and predicted in the following format.

Summary Statistics:

Min Max Mean SD PRP Correlation

MCYT: 17 1500 203.8 260.3 -0.3071

MMIN: 64 32000 2868.0 3878.7 0.7949

MMAX: 64 64000 11796.1 11726.6 0.8630

CACH: 0 256 25.2 40.6 0.6626

CHMIN: 0 52 4.7 6.8 0.6089

CHMAX: 0 176 18.2 26.0 0.6052

PRP: 6 1150 105.6 160.8 1.0000

ERP: 15 1238 99.3 154.8 0.9665

Class Distribution: the class value (PRP) is continuously valued.

PRP Value Range: Number of Instances in Range:

0-20 31

21-100 121

101-200 27

201-300 13

301-400 7

401-500 4

501-600 2

above 600 4

Now by executing the tree, and ctree and identifying the variance of the dataset, is how best it is solving the big question regarding the identification of cu performance taking class attribute of PRP.

R Code:

library(readr)

cpu <- read\_csv("C:/DM SEM 2/week 7/cpu.csv")

View(cpu)

summary(cpu)

range(cpu$PRP)

install.packages("tree")

install.packages("party")

install.packages("rpart")

install.packages("car")

install.packages("mlbench")

install.packages("mboost")

install.packages("textir")

install.packages("class")

install.packages("e1071")

install.packages("randomForest")

library(tree)

data(cpu.csv)

cpu

cpu$PRP <- ordered(cut(cpu[["PRP"]],c(0,20,100,200,300,400,500,600,1550)))

ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

trainDataTree <- cpu[ind==1,]

testDataTree <- cpu[ind==2,]

myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

PRP\_tree <- tree(myFormula, data=trainDataTree)

summary(PRP\_tree)

print(PRP\_tree)  
  
Output:  
  
> library(readr)

> cpu <- read\_csv("C:/DM SEM 2/week 7/cpu.csv")

Parsed with column specification:

cols(

Vendor = col\_character(),

MYCT = col\_integer(),

MMIN = col\_integer(),

MMAX = col\_integer(),

CACH = col\_integer(),

CHMIN = col\_integer(),

CHMAX = col\_integer(),

PRP = col\_integer(),

ERP = col\_integer()

)

> View(cpu)

> summary(cpu)

Vendor MYCT MMIN MMAX CACH CHMIN CHMAX

Length:209 Min. : 17.0 Min. : 64 Min. : 64 Min. : 0.00 Min. : 0.000 Min. : 0.00

Class :character 1st Qu.: 50.0 1st Qu.: 768 1st Qu.: 4000 1st Qu.: 0.00 1st Qu.: 1.000 1st Qu.: 5.00

Mode :character Median : 110.0 Median : 2000 Median : 8000 Median : 8.00 Median : 2.000 Median : 8.00

Mean : 203.8 Mean : 2868 Mean :11796 Mean : 25.21 Mean : 4.699 Mean : 18.27

3rd Qu.: 225.0 3rd Qu.: 4000 3rd Qu.:16000 3rd Qu.: 32.00 3rd Qu.: 6.000 3rd Qu.: 24.00

Max. :1500.0 Max. :32000 Max. :64000 Max. :256.00 Max. :52.000 Max. :176.00

PRP ERP

Min. : 6.0 Min. : 15.00

1st Qu.: 27.0 1st Qu.: 28.00

Median : 50.0 Median : 45.00

Mean : 105.6 Mean : 99.33

3rd Qu.: 113.0 3rd Qu.: 101.00

Max. :1150.0 Max. :1238.00

> range(cpu$PRP)

[1] 6 1150

> library(tree)

> cpu

# A tibble: 209 × 9

Vendor MYCT MMIN MMAX CACH CHMIN CHMAX PRP ERP

<chr> <int> <int> <int> <int> <int> <int> <int> <int>

1 adviser 125 256 6000 256 16 128 198 199

2 amdahl 29 8000 32000 32 8 32 269 253

3 amdahl 29 8000 32000 32 8 32 220 253

4 amdahl 29 8000 32000 32 8 32 172 253

5 amdahl 29 8000 16000 32 8 16 132 132

6 amdahl 26 8000 32000 64 8 32 318 290

7 amdahl 23 16000 32000 64 16 32 367 381

8 amdahl 23 16000 32000 64 16 32 489 381

9 amdahl 23 16000 64000 64 16 32 636 749

10 amdahl 23 32000 64000 128 32 64 1144 1238

# ... with 199 more rows

> cpu$PRP <- ordered(cut(cpu[["PRP"]],c(0,20,100,200,300,400,500,600,1550)))

> ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

> trainDataTree <- cpu[ind==1,]

> testDataTree <- cpu[ind==2,]

> myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

> PRP\_tree <- tree(myFormula, data=trainDataTree)

> summary(PRP\_tree)

Classification tree:

tree(formula = myFormula, data = trainDataTree)

Variables actually used in tree construction:

[1] "ERP" "CHMAX" "MMAX" "CHMIN" "MYCT" "CACH"

Number of terminal nodes: 15

Residual mean deviance: 0.8289 = 111.1 / 134

Misclassification error rate: 0.1946 = 29 / 149

> print(PRP\_tree)

node), split, n, deviance, yval, (yprob)

\* denotes terminal node

1) root 149 385.400 (20,100] ( 0.134228 0.604027 0.134228 0.040268 0.040268 0.020134 0.006711 0.020134 )

2) ERP < 98.5 114 150.500 (20,100] ( 0.175439 0.771930 0.052632 0.000000 0.000000 0.000000 0.000000 0.000000 )

4) ERP < 37.5 62 77.970 (20,100] ( 0.322581 0.677419 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )

8) CHMAX < 4.5 29 39.340 (0,20] ( 0.586207 0.413793 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )

16) MMAX < 1750 6 0.000 (0,20] ( 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

17) MMAX > 1750 23 31.840 (20,100] ( 0.478261 0.521739 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

9) CHMAX > 4.5 33 20.110 (20,100] ( 0.090909 0.909091 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )

18) CHMIN < 2.5 16 15.440 (20,100] ( 0.187500 0.812500 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

19) CHMIN > 2.5 17 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

5) ERP > 37.5 52 37.190 (20,100] ( 0.000000 0.884615 0.115385 0.000000 0.000000 0.000000 0.000000 0.000000 )

10) ERP < 47.5 22 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

11) ERP > 47.5 30 30.020 (20,100] ( 0.000000 0.800000 0.200000 0.000000 0.000000 0.000000 0.000000 0.000000 )

22) MYCT < 56.5 10 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

23) MYCT > 56.5 20 24.430 (20,100] ( 0.000000 0.700000 0.300000 0.000000 0.000000 0.000000 0.000000 0.000000 )

46) MYCT < 170 13 17.940 (20,100] ( 0.000000 0.538462 0.461538 0.000000 0.000000 0.000000 0.000000 0.000000 )

92) ERP < 69.5 8 8.997 (20,100] ( 0.000000 0.750000 0.250000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

93) ERP > 69.5 5 5.004 (100,200] ( 0.000000 0.200000 0.800000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

47) MYCT > 170 7 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

3) ERP > 98.5 35 116.000 (100,200] ( 0.000000 0.057143 0.400000 0.171429 0.171429 0.085714 0.028571 0.085714 )

6) ERP < 285.5 24 51.610 (100,200] ( 0.000000 0.083333 0.583333 0.250000 0.083333 0.000000 0.000000 0.000000 )

12) CHMAX < 14 5 6.730 (100,200] ( 0.000000 0.400000 0.600000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*

13) CHMAX > 14 19 34.860 (100,200] ( 0.000000 0.000000 0.578947 0.315789 0.105263 0.000000 0.000000 0.000000 )

26) ERP < 209.5 13 17.860 (100,200] ( 0.000000 0.000000 0.769231 0.153846 0.076923 0.000000 0.000000 0.000000 )

52) MYCT < 60 6 5.407 (100,200] ( 0.000000 0.000000 0.833333 0.000000 0.166667 0.000000 0.000000 0.000000 ) \*

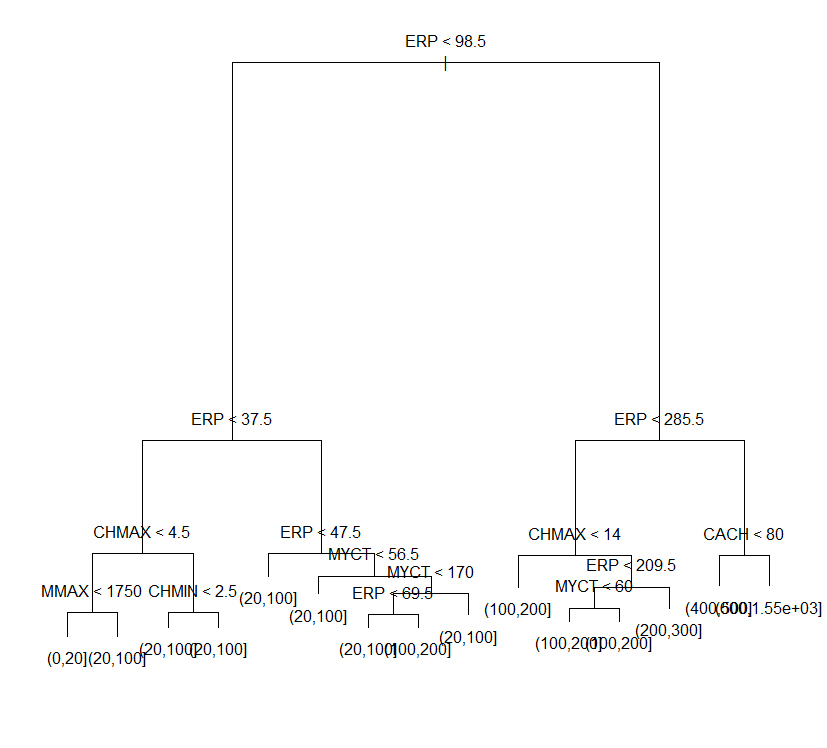
53) MYCT > 60 7 8.376 (100,200] ( 0.000000 0.000000 0.714286 0.285714 0.000000 0.000000 0.000000 0.000000 ) \*

27) ERP > 209.5 6 10.410 (200,300] ( 0.000000 0.000000 0.166667 0.666667 0.166667 0.000000 0.000000 0.000000 ) \*

7) ERP > 285.5 11 28.480 (300,400] ( 0.000000 0.000000 0.000000 0.000000 0.363636 0.272727 0.090909 0.272727 )

14) CACH < 80 5 6.730 (400,500] ( 0.000000 0.000000 0.000000 0.000000 0.400000 0.600000 0.000000 0.000000 ) \*

15) CACH > 80 6 12.140 (600,1.55e+03] ( 0.000000 0.000000 0.000000 0.000000 0.333333 0.000000 0.166667 0.500000 ) \*

Now by plotting the execution of what the tree values what are been taken in the above the dataset taking PRP as class attribute, now identifying the plot and text to identifying what best lies in what region.   
  
  
  
  
Now in this to make the information more cleared , I had taken the reference of the link which I had provided above to make the value order and been categorized in the specified order of value and make them ideolized with ordered placed of value to have the data most reliable to the the class attribute, and make it nearest to it, value is if the value lie in ERP if it is <37.5 it has taken least preferable to It have a chmax and erp to value to make It keeping value in this, the plot is been identified, and PRP value range is been identified between 6 to 150

range(cpu$PRP)

|  |
| --- |
| > range(cpu$PRP)  [1] (0,20] (600,1.55e+03]  Levels: (0,20] < (20,100] < (100,200] < (200,300] < (300,400] < (400,500] < (500,600] < (600,1.55e+03] |
|  |
| |  | | --- | | > | |

now let see the prediction values which is been given in the below code.

R Code:   
  
testPred <- predict(PRP\_tree, newdata = testDataTree)

show(testPred)

library(MASS)

library(tree)

cpu

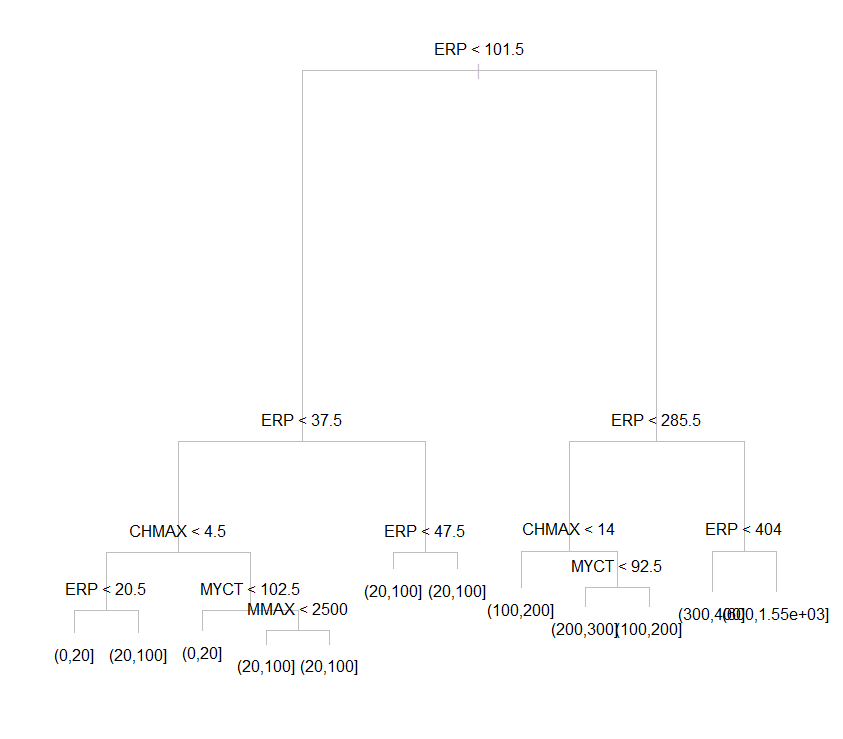
PRPtree <- tree(PRP~.,data=cpu)

PRPtree  
  
  
Output:

|  |
| --- |
| > testPred <- predict(PRP\_tree, newdata = testDataTree)  > show(testPred)  (0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]  1 0.0000000 0.0000000 0.1666667 0.6666667 0.1666667 0.0 0.0000000 0.0  2 0.0000000 0.0000000 0.1666667 0.6666667 0.1666667 0.0 0.0000000 0.0  3 0.0000000 0.0000000 0.0000000 0.0000000 0.4000000 0.6 0.0000000 0.0  4 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  5 0.0000000 0.2000000 0.8000000 0.0000000 0.0000000 0.0 0.0000000 0.0  6 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  7 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  8 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  9 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  10 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  11 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  12 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  13 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  14 0.0000000 0.0000000 0.7142857 0.2857143 0.0000000 0.0 0.0000000 0.0  15 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  16 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  17 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  18 0.0000000 0.7500000 0.2500000 0.0000000 0.0000000 0.0 0.0000000 0.0  19 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  20 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  21 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  22 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  23 0.0000000 0.0000000 0.7142857 0.2857143 0.0000000 0.0 0.0000000 0.0  24 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  25 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  26 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  27 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  28 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  29 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  30 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  31 0.0000000 0.4000000 0.6000000 0.0000000 0.0000000 0.0 0.0000000 0.0  32 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  33 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  34 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  35 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  36 0.0000000 0.4000000 0.6000000 0.0000000 0.0000000 0.0 0.0000000 0.0  37 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  38 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  39 0.0000000 0.7500000 0.2500000 0.0000000 0.0000000 0.0 0.0000000 0.0  40 0.0000000 0.2000000 0.8000000 0.0000000 0.0000000 0.0 0.0000000 0.0  41 0.0000000 0.4000000 0.6000000 0.0000000 0.0000000 0.0 0.0000000 0.0  42 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  43 0.0000000 0.0000000 0.1666667 0.6666667 0.1666667 0.0 0.0000000 0.0  44 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333 0.0 0.1666667 0.5  45 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  46 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  47 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  48 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  49 0.0000000 0.4000000 0.6000000 0.0000000 0.0000000 0.0 0.0000000 0.0  50 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  51 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  52 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  53 0.1875000 0.8125000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  54 0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  55 0.0000000 0.4000000 0.6000000 0.0000000 0.0000000 0.0 0.0000000 0.0  56 0.0000000 0.0000000 0.8333333 0.0000000 0.1666667 0.0 0.0000000 0.0  57 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333 0.0 0.1666667 0.5  58 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  59 0.4782609 0.5217391 0.0000000 0.0000000 0.0000000 0.0 0.0000000 0.0  60 0.0000000 0.7500000 0.2500000 0.0000000 0.0000000 0.0 0.0000000 0.0  > library(MASS)  > library(tree)  > cpu  # A tibble: 209 × 9  Vendor MYCT MMIN MMAX CACH CHMIN CHMAX PRP ERP  <chr> <int> <int> <int> <int> <int> <int> <ord> <int>  1 adviser 125 256 6000 256 16 128 (100,200] 199  2 amdahl 29 8000 32000 32 8 32 (200,300] 253  3 amdahl 29 8000 32000 32 8 32 (200,300] 253  4 amdahl 29 8000 32000 32 8 32 (100,200] 253  5 amdahl 29 8000 16000 32 8 16 (100,200] 132  6 amdahl 26 8000 32000 64 8 32 (300,400] 290  7 amdahl 23 16000 32000 64 16 32 (300,400] 381  8 amdahl 23 16000 32000 64 16 32 (400,500] 381  9 amdahl 23 16000 64000 64 16 32 (600,1.55e+03] 749  10 amdahl 23 32000 64000 128 32 64 (600,1.55e+03] 1238  # ... with 199 more rows  > PRPtree <- tree(PRP~.,data=cpu)  Warning message:  In tree(PRP ~ ., data = cpu) : NAs introduced by coercion  > PRPtree  node), split, n, deviance, yval, (yprob)  \* denotes terminal node  1) root 209 562.700 (20,100] ( 0.148325 0.578947 0.129187 0.062201 0.033493 0.019139 0.009569 0.019139 )  2) ERP < 101.5 157 215.600 (20,100] ( 0.197452 0.751592 0.050955 0.000000 0.000000 0.000000 0.000000 0.000000 )  4) ERP < 37.5 88 114.200 (20,100] ( 0.352273 0.647727 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  8) CHMAX < 4.5 39 50.920 (0,20] ( 0.641026 0.358974 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  16) ERP < 20.5 15 7.348 (0,20] ( 0.933333 0.066667 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  17) ERP > 20.5 24 33.100 (20,100] ( 0.458333 0.541667 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  9) CHMAX > 4.5 49 36.430 (20,100] ( 0.122449 0.877551 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  18) MYCT < 102.5 8 11.090 (0,20] ( 0.500000 0.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  19) MYCT > 102.5 41 15.980 (20,100] ( 0.048780 0.951220 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  38) MMAX < 2500 9 9.535 (20,100] ( 0.222222 0.777778 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  39) MMAX > 2500 32 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  5) ERP > 37.5 69 49.510 (20,100] ( 0.000000 0.884058 0.115942 0.000000 0.000000 0.000000 0.000000 0.000000 )  10) ERP < 47.5 24 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  11) ERP > 47.5 45 42.120 (20,100] ( 0.000000 0.822222 0.177778 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  3) ERP > 101.5 52 173.600 (100,200] ( 0.000000 0.057692 0.365385 0.250000 0.134615 0.076923 0.038462 0.076923 )  6) ERP < 285.5 38 84.700 (100,200] ( 0.000000 0.078947 0.500000 0.342105 0.078947 0.000000 0.000000 0.000000 )  12) CHMAX < 14 8 10.590 (100,200] ( 0.000000 0.375000 0.625000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  13) CHMAX > 14 30 56.900 (100,200] ( 0.000000 0.000000 0.466667 0.433333 0.100000 0.000000 0.000000 0.000000 )  26) MYCT < 92.5 25 48.110 (200,300] ( 0.000000 0.000000 0.360000 0.520000 0.120000 0.000000 0.000000 0.000000 ) \*  27) MYCT > 92.5 5 0.000 (100,200] ( 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  7) ERP > 285.5 14 37.850 (300,400] ( 0.000000 0.000000 0.000000 0.000000 0.285714 0.285714 0.142857 0.285714 )  14) ERP < 404 8 11.090 (300,400] ( 0.000000 0.000000 0.000000 0.000000 0.500000 0.500000 0.000000 0.000000 ) \*  15) ERP > 404 6 7.638 (600,1.55e+03] ( 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.333333 0.666667 ) \* |
|  |
| |  | | --- | | > | |

Now I had plot and text the plot with the value identified with the output.   
  
  
  
R Code:   
  
plot(PRPtree)

plot(PRPtree,col=8)

text(PRPtree,digits=2)  
  
  
Output:  
  
  
  
  
Now identifying the snipping of the dataset, to make it with identified with   
  
  
R Code:   
  
summary(PRPtree)

PRPsnip=snip.tree(PRPtree,nodes=c(7,12))

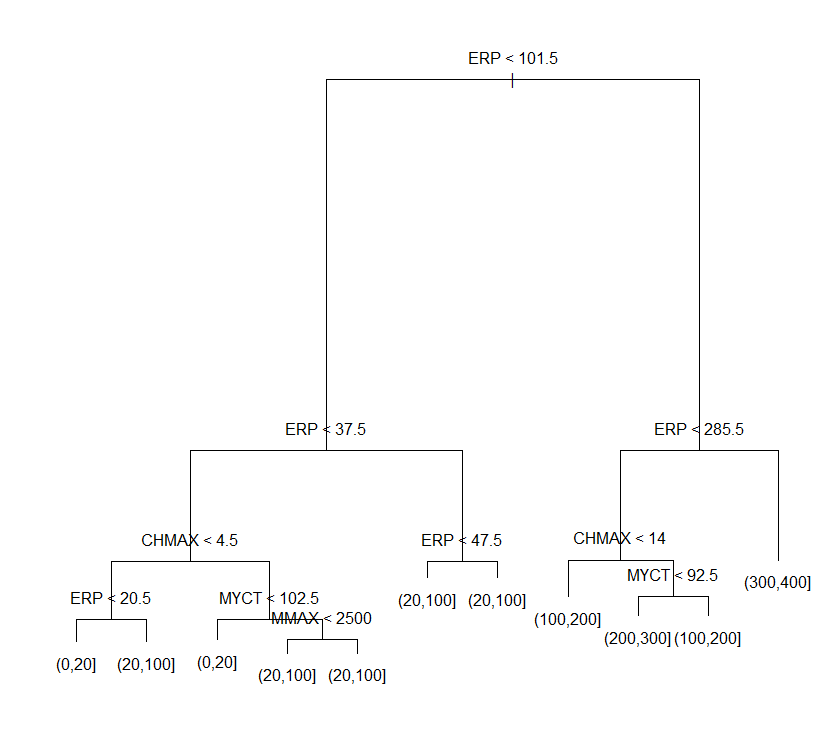
PRPsnip

plot(PRPsnip)

text(PRPsnip)

summary(PRPsnip)

|  |
| --- |
| > summary(PRPtree)  Classification tree:  tree(formula = PRP ~ ., data = cpu)  Variables actually used in tree construction:  [1] "ERP" "CHMAX" "MYCT" "MMAX"  Number of terminal nodes: 12  Residual mean deviance: 0.9169 = 180.6 / 197  Misclassification error rate: 0.2249 = 47 / 209  > PRPsnip=snip.tree(PRPtree,nodes=c(7,12))  Warning message:  In node.match(nodes, node, tree$frame$var == "<leaf>") :  supplied nodes 12 are leaves  > PRPsnip  node), split, n, deviance, yval, (yprob)  \* denotes terminal node  1) root 209 562.700 (20,100] ( 0.148325 0.578947 0.129187 0.062201 0.033493 0.019139 0.009569 0.019139 )  2) ERP < 101.5 157 215.600 (20,100] ( 0.197452 0.751592 0.050955 0.000000 0.000000 0.000000 0.000000 0.000000 )  4) ERP < 37.5 88 114.200 (20,100] ( 0.352273 0.647727 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  8) CHMAX < 4.5 39 50.920 (0,20] ( 0.641026 0.358974 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  16) ERP < 20.5 15 7.348 (0,20] ( 0.933333 0.066667 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  17) ERP > 20.5 24 33.100 (20,100] ( 0.458333 0.541667 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  9) CHMAX > 4.5 49 36.430 (20,100] ( 0.122449 0.877551 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  18) MYCT < 102.5 8 11.090 (0,20] ( 0.500000 0.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  19) MYCT > 102.5 41 15.980 (20,100] ( 0.048780 0.951220 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 )  38) MMAX < 2500 9 9.535 (20,100] ( 0.222222 0.777778 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  39) MMAX > 2500 32 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  5) ERP > 37.5 69 49.510 (20,100] ( 0.000000 0.884058 0.115942 0.000000 0.000000 0.000000 0.000000 0.000000 )  10) ERP < 47.5 24 0.000 (20,100] ( 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  11) ERP > 47.5 45 42.120 (20,100] ( 0.000000 0.822222 0.177778 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  3) ERP > 101.5 52 173.600 (100,200] ( 0.000000 0.057692 0.365385 0.250000 0.134615 0.076923 0.038462 0.076923 )  6) ERP < 285.5 38 84.700 (100,200] ( 0.000000 0.078947 0.500000 0.342105 0.078947 0.000000 0.000000 0.000000 )  12) CHMAX < 14 8 10.590 (100,200] ( 0.000000 0.375000 0.625000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  13) CHMAX > 14 30 56.900 (100,200] ( 0.000000 0.000000 0.466667 0.433333 0.100000 0.000000 0.000000 0.000000 )  26) MYCT < 92.5 25 48.110 (200,300] ( 0.000000 0.000000 0.360000 0.520000 0.120000 0.000000 0.000000 0.000000 ) \*  27) MYCT > 92.5 5 0.000 (100,200] ( 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ) \*  7) ERP > 285.5 14 37.850 (300,400] ( 0.000000 0.000000 0.000000 0.000000 0.285714 0.285714 0.142857 0.285714 ) \*  > plot(PRPsnip)  > text(PRPsnip)  > summary(PRPsnip)  Classification tree:  snip.tree(tree = PRPtree, nodes = 7L)  Variables actually used in tree construction:  [1] "ERP" "CHMAX" "MYCT" "MMAX"  Number of terminal nodes: 11  Residual mean deviance: 1.009 = 199.7 / 198  Misclassification error rate: 0.244 = 51 / 209 |
|  |
| |  | | --- | | > | |

  
  
Now working with Ctree, to identifying the confidence and best nearest satisfying the big question.   
  
  
 R Code:   
  
  
set.seed(1234)

ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

trainData <- cpu[ind==1,]

testData <- cpu[ind==2,]

library(party)

myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

PRP\_ctree <- ctree(myFormula, data=trainData)

table(predict(PRP\_ctree), trainData$PRP)

print(PRP\_ctree)  
  
  
Output:   
  
  
> set.seed(1234)

> ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

> trainData <- cpu[ind==1,]

> testData <- cpu[ind==2,]

> library(party)

> myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

> PRP\_ctree <- ctree(myFormula, data=trainData)

> table(predict(PRP\_ctree), trainData$PRP)

(0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]

(0,20] 14 2 0 0 0 0 0 0

(20,100] 9 88 6 0 0 0 0 0

(100,200] 0 2 10 9 1 0 0 0

(200,300] 0 0 0 0 0 0 0 0

(300,400] 0 0 0 0 0 0 0 0

(400,500] 0 0 0 0 3 4 1 3

(500,600] 0 0 0 0 0 0 0 0

(600,1.55e+03] 0 0 0 0 0 0 0 0

> print(PRP\_ctree)

Conditional inference tree with 7 terminal nodes

Response: PRP

Inputs: MYCT, MMIN, MMAX, CACH, CHMIN, CHMAX, ERP

Number of observations: 152

1) ERP <= 266; criterion = 1, statistic = 117.286

2) ERP <= 95; criterion = 1, statistic = 81.091

3) ERP <= 24; criterion = 1, statistic = 31.817

4) CHMAX <= 4; criterion = 0.969, statistic = 8.091

5)\* weights = 16

4) CHMAX > 4

6)\* weights = 14

3) ERP > 24

7) CACH <= 30; criterion = 0.999, statistic = 14.061

8) MYCT <= 400; criterion = 0.998, statistic = 13.022

9)\* weights = 73

8) MYCT > 400

10)\* weights = 7

7) CACH > 30

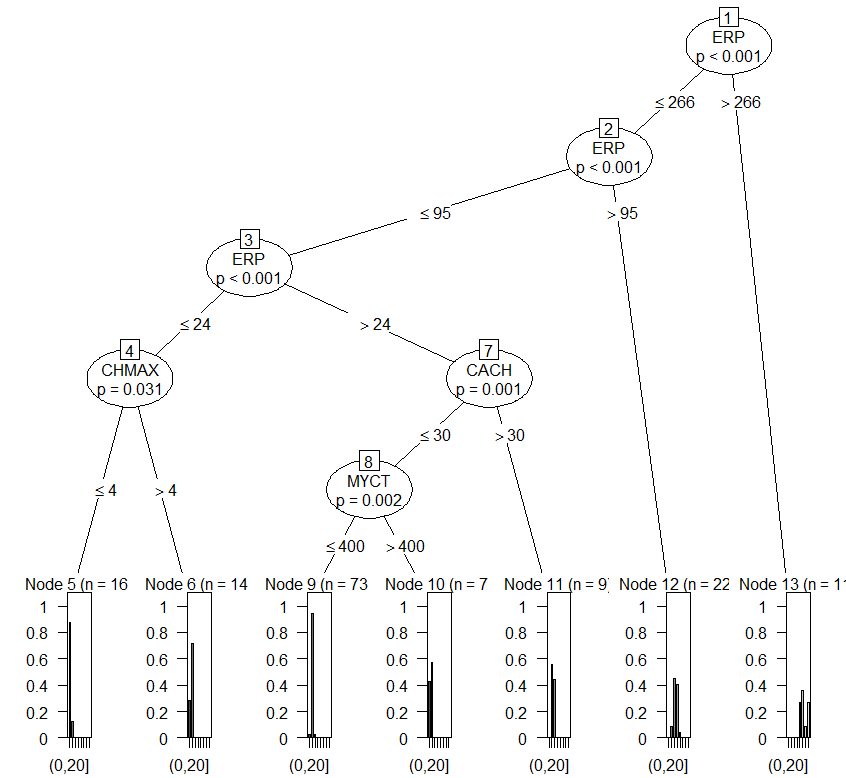
11)\* weights = 9

2) ERP > 95

12)\* weights = 22

1) ERP > 266

13)\* weights = 11

  
  
  
  
R Code:   
  
  
testPred <- predict(PRP\_ctree, newdata = testData)

table(testPred, testData$PRP)

data("cpu")

cpu

attributes(cpu)

set.seed(1234)

ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

cpu.train <- cpu[ind==1,]

cpu.test <- cpu[ind==2,]

library(rpart)

myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

cpu\_rpart <- rpart(myFormula, data = cpu.train, control = rpart.control(minsplit = 10))

attributes(cpu\_rpart)

print(cpu\_rpart$PRP)

print(cpu\_rpart)

plot(cpu\_rpart)

text(cpu\_rpart, use.n=T)  
  
  
Output:  
  
  
> testPred <- predict(PRP\_ctree, newdata = testData)

> table(testPred, testData$PRP)

testPred (0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]

(0,20] 5 4 0 0 0 0 0 0

(20,100] 3 23 1 0 0 0 0 0

(100,200] 0 2 9 3 0 0 0 0

(200,300] 0 0 0 0 0 0 0 0

(300,400] 0 0 0 0 0 0 0 0

(400,500] 0 0 1 1 3 0 1 1

(500,600] 0 0 0 0 0 0 0 0

(600,1.55e+03] 0 0 0 0 0 0 0 0

> data("cpu")

Warning message:

In data("cpu") : data set ‘cpu’ not found

> cpu

# A tibble: 209 × 9

Vendor MYCT MMIN MMAX CACH CHMIN CHMAX PRP ERP

<chr> <int> <int> <int> <int> <int> <int> <ord> <int>

1 adviser 125 256 6000 256 16 128 (100,200] 199

2 amdahl 29 8000 32000 32 8 32 (200,300] 253

3 amdahl 29 8000 32000 32 8 32 (200,300] 253

4 amdahl 29 8000 32000 32 8 32 (100,200] 253

5 amdahl 29 8000 16000 32 8 16 (100,200] 132

6 amdahl 26 8000 32000 64 8 32 (300,400] 290

7 amdahl 23 16000 32000 64 16 32 (300,400] 381

8 amdahl 23 16000 32000 64 16 32 (400,500] 381

9 amdahl 23 16000 64000 64 16 32 (600,1.55e+03] 749

10 amdahl 23 32000 64000 128 32 64 (600,1.55e+03] 1238

# ... with 199 more rows

> attributes(cpu)

$row.names

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

[32] 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62

[63] 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93

[94] 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124

[125] 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155

[156] 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186

[187] 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209

$names

[1] "Vendor" "MYCT" "MMIN" "MMAX" "CACH" "CHMIN" "CHMAX" "PRP" "ERP"

$spec

cols(

Vendor = col\_character(),

MYCT = col\_integer(),

MMIN = col\_integer(),

MMAX = col\_integer(),

CACH = col\_integer(),

CHMIN = col\_integer(),

CHMAX = col\_integer(),

PRP = col\_integer(),

ERP = col\_integer()

)

$class

[1] "tbl\_df" "tbl" "data.frame"

> testPred <- predict(PRP\_ctree, newdata = testData)

> table(testPred, testData$PRP)

testPred (0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]

(0,20] 5 4 0 0 0 0 0 0

(20,100] 3 23 1 0 0 0 0 0

(100,200] 0 2 9 3 0 0 0 0

(200,300] 0 0 0 0 0 0 0 0

(300,400] 0 0 0 0 0 0 0 0

(400,500] 0 0 1 1 3 0 1 1

(500,600] 0 0 0 0 0 0 0 0

(600,1.55e+03] 0 0 0 0 0 0 0 0

> data("cpu")

Warning message:

In data("cpu") : data set ‘cpu’ not found

> cpu

# A tibble: 209 × 9

Vendor MYCT MMIN MMAX CACH CHMIN CHMAX PRP ERP

<chr> <int> <int> <int> <int> <int> <int> <ord> <int>

1 adviser 125 256 6000 256 16 128 (100,200] 199

2 amdahl 29 8000 32000 32 8 32 (200,300] 253

3 amdahl 29 8000 32000 32 8 32 (200,300] 253

4 amdahl 29 8000 32000 32 8 32 (100,200] 253

5 amdahl 29 8000 16000 32 8 16 (100,200] 132

6 amdahl 26 8000 32000 64 8 32 (300,400] 290

7 amdahl 23 16000 32000 64 16 32 (300,400] 381

8 amdahl 23 16000 32000 64 16 32 (400,500] 381

9 amdahl 23 16000 64000 64 16 32 (600,1.55e+03] 749

10 amdahl 23 32000 64000 128 32 64 (600,1.55e+03] 1238

# ... with 199 more rows

> attributes(cpu)

$row.names

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

[32] 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62

[63] 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93

[94] 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124

[125] 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155

[156] 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186

[187] 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209

$names

[1] "Vendor" "MYCT" "MMIN" "MMAX" "CACH" "CHMIN" "CHMAX" "PRP" "ERP"

$spec

cols(

Vendor = col\_character(),

MYCT = col\_integer(),

MMIN = col\_integer(),

MMAX = col\_integer(),

CACH = col\_integer(),

CHMIN = col\_integer(),

CHMAX = col\_integer(),

PRP = col\_integer(),

ERP = col\_integer()

)

$class

[1] "tbl\_df" "tbl" "data.frame"

> set.seed(1234)

> ind <- sample(2, nrow(cpu), replace=TRUE, prob=c(0.7, 0.3))

> cpu.train <- cpu[ind==1,]

> cpu.test <- cpu[ind==2,]

> library(rpart)

> myFormula <- PRP ~ MYCT + MMIN + MMAX + CACH + CHMIN + CHMAX + ERP

> cpu\_rpart <- rpart(myFormula, data = cpu.train, control = rpart.control(minsplit = 10))

> attributes(cpu\_rpart)

$names

[1] "frame" "where" "call" "terms" "cptable"

[6] "method" "parms" "control" "functions" "numresp"

[11] "splits" "variable.importance" "y" "ordered"

$xlevels

named list()

$ylevels

[1] "(0,20]" "(20,100]" "(100,200]" "(200,300]" "(300,400]" "(400,500]" "(500,600]"

[8] "(600,1.55e+03]"

$class

[1] "rpart"

> print(cpu\_rpart$PRP)

NULL

> print(cpu\_rpart)

n= 152

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 152 60 (20,100] (0.15 0.61 0.11 0.059 0.026 0.026 0.0066 0.02)

2) ERP< 98.5 119 29 (20,100] (0.19 0.76 0.05 0 0 0 0 0)

4) ERP< 24.5 30 12 (0,20] (0.6 0.4 0 0 0 0 0 0)

8) CHMAX< 4.5 16 2 (0,20] (0.88 0.12 0 0 0 0 0 0) \*

9) CHMAX>=4.5 14 4 (20,100] (0.29 0.71 0 0 0 0 0 0)

18) MYCT< 115 3 1 (0,20] (0.67 0.33 0 0 0 0 0 0) \*

19) MYCT>=115 11 2 (20,100] (0.18 0.82 0 0 0 0 0 0)

38) CACH>=5 3 1 (0,20] (0.67 0.33 0 0 0 0 0 0) \*

39) CACH< 5 8 0 (20,100] (0 1 0 0 0 0 0 0) \*

5) ERP>=24.5 89 11 (20,100] (0.056 0.88 0.067 0 0 0 0 0)

10) MYCT>=750 5 2 (0,20] (0.6 0.4 0 0 0 0 0 0) \*

11) MYCT< 750 84 8 (20,100] (0.024 0.9 0.071 0 0 0 0 0) \*

3) ERP>=98.5 33 23 (100,200] (0 0.061 0.3 0.27 0.12 0.12 0.03 0.091)

6) ERP< 268 22 12 (100,200] (0 0.091 0.45 0.41 0.045 0 0 0)

12) MYCT>=67.5 5 0 (100,200] (0 0 1 0 0 0 0 0) \*

13) MYCT< 67.5 17 8 (200,300] (0 0.12 0.29 0.53 0.059 0 0 0)

26) CHMIN< 6.5 3 1 (20,100] (0 0.67 0.33 0 0 0 0 0) \*

27) CHMIN>=6.5 14 5 (200,300] (0 0 0.29 0.64 0.071 0 0 0) \*

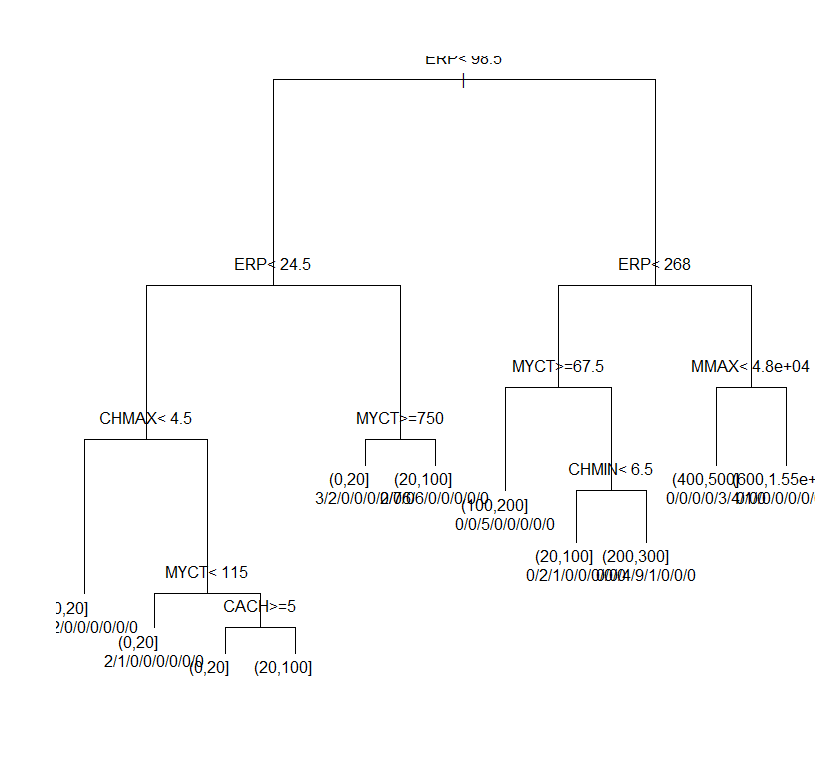
7) ERP>=268 11 7 (400,500] (0 0 0 0 0.27 0.36 0.091 0.27)

14) MMAX< 48000 8 4 (400,500] (0 0 0 0 0.38 0.5 0.12 0) \*

15) MMAX>=48000 3 0 (600,1.55e+03] (0 0 0 0 0 0 0 1) \*

> plot(cpu\_rpart)

> text(cpu\_rpart, use.n=T)

  
  
Prune content of the dataset Is :   
  
  
R Code:   
  
  
opt <- which.min(cpu\_rpart$PRP[,"xerror"])

cp <- cpu\_rpart$PRP[opt, "cp"]

cpu\_prune <- prune(cpu\_rpart, cp = cp)

print(cpu\_prune)

plot(cpu\_prune)

text(cpu\_prune, use.n=T)  
  
  
Output:   
  
  
> opt <- which.min(cpu\_rpart$PRP[,"xerror"])

> cp <- cpu\_rpart$PRP[opt, "cp"]

> cpu\_prune <- prune(cpu\_rpart, cp = cp)

> print(cpu\_prune)

n= 152

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 152 60 (20,100] (0.15 0.61 0.11 0.059 0.026 0.026 0.0066 0.02)

2) ERP< 98.5 119 29 (20,100] (0.19 0.76 0.05 0 0 0 0 0)

4) ERP< 24.5 30 12 (0,20] (0.6 0.4 0 0 0 0 0 0)

8) CHMAX< 4.5 16 2 (0,20] (0.88 0.12 0 0 0 0 0 0) \*

9) CHMAX>=4.5 14 4 (20,100] (0.29 0.71 0 0 0 0 0 0)

18) MYCT< 115 3 1 (0,20] (0.67 0.33 0 0 0 0 0 0) \*

19) MYCT>=115 11 2 (20,100] (0.18 0.82 0 0 0 0 0 0)

38) CACH>=5 3 1 (0,20] (0.67 0.33 0 0 0 0 0 0) \*

39) CACH< 5 8 0 (20,100] (0 1 0 0 0 0 0 0) \*

5) ERP>=24.5 89 11 (20,100] (0.056 0.88 0.067 0 0 0 0 0)

10) MYCT>=750 5 2 (0,20] (0.6 0.4 0 0 0 0 0 0) \*

11) MYCT< 750 84 8 (20,100] (0.024 0.9 0.071 0 0 0 0 0) \*

3) ERP>=98.5 33 23 (100,200] (0 0.061 0.3 0.27 0.12 0.12 0.03 0.091)

6) ERP< 268 22 12 (100,200] (0 0.091 0.45 0.41 0.045 0 0 0)

12) MYCT>=67.5 5 0 (100,200] (0 0 1 0 0 0 0 0) \*

13) MYCT< 67.5 17 8 (200,300] (0 0.12 0.29 0.53 0.059 0 0 0)

26) CHMIN< 6.5 3 1 (20,100] (0 0.67 0.33 0 0 0 0 0) \*

27) CHMIN>=6.5 14 5 (200,300] (0 0 0.29 0.64 0.071 0 0 0) \*

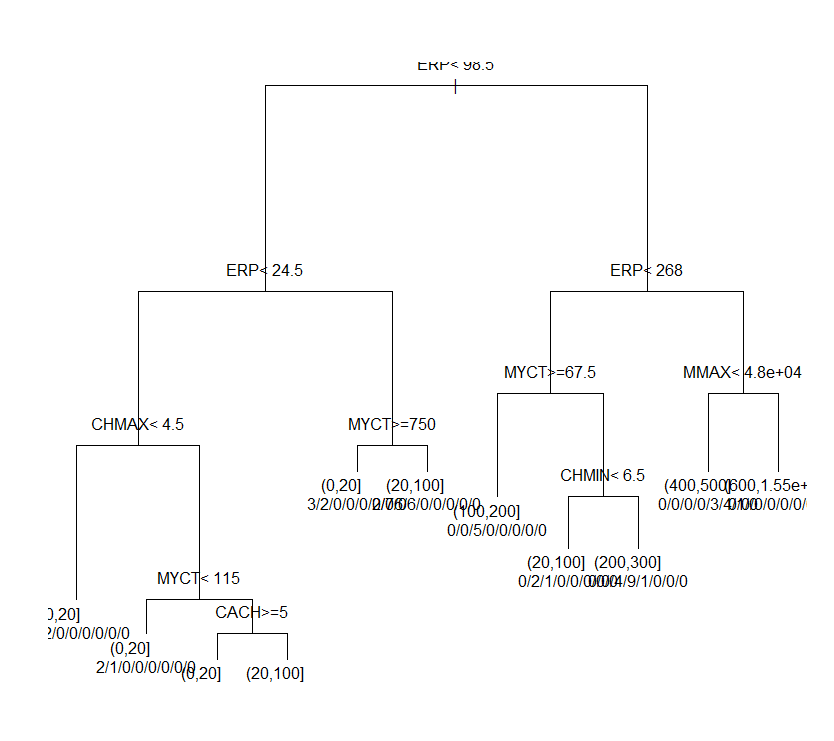
7) ERP>=268 11 7 (400,500] (0 0 0 0 0.27 0.36 0.091 0.27)

14) MMAX< 48000 8 4 (400,500] (0 0 0 0 0.38 0.5 0.12 0) \*

15) MMAX>=48000 3 0 (600,1.55e+03] (0 0 0 0 0 0 0 1) \*

> plot(cpu\_prune)

> text(cpu\_prune, use.n=T)

  
  
  
  
Prediction using abline:  
  
R Code:  
  
  
  
DEXfat\_pred <- predict(cpu\_prune, newdata=cpu.test)

xlim <- range(PRP)

plot(DEXfat\_pred ~ PRP, data=cpu.test, xlab="Observed", ylab="Predicted")

abline(a=0, b=1)

Output:  
  
> text(cpu\_prune, use.n=T)

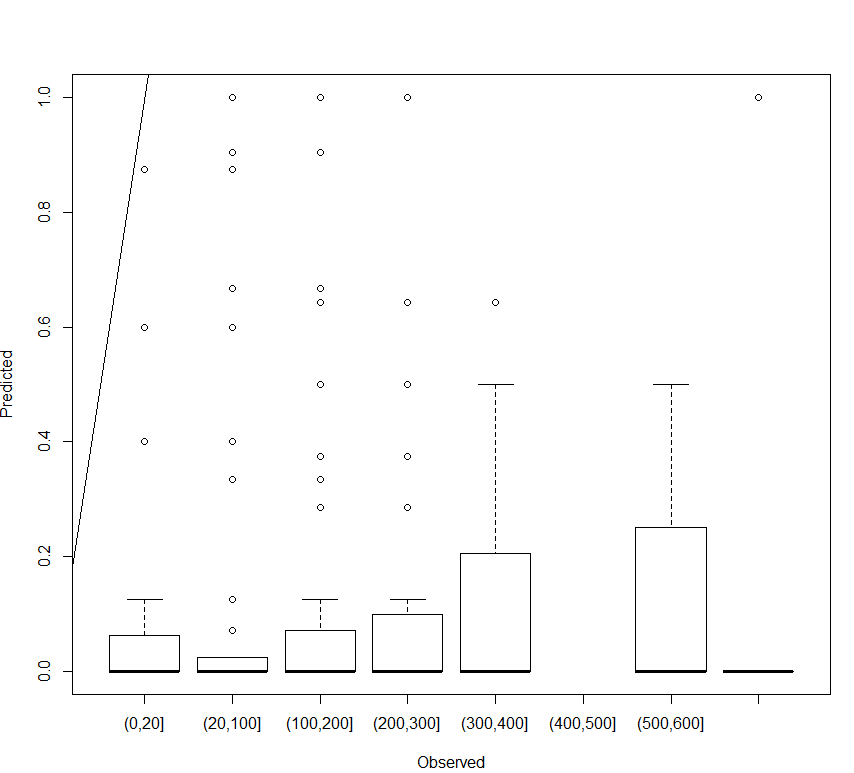
> DEXfat\_pred <- predict(cpu\_prune, newdata=cpu.test)

> xlim <- range(PRP)

Error: object 'PRP' not found

> plot(DEXfat\_pred ~ PRP, data=cpu.test, xlab="Observed", ylab="Predicted")

> abline(a=0, b=1)

  
  
  
  
  
  
Now identifying the Naïve bayes theorem , making the idenfying using Bayes theorem.   
  
R Code:   
  
  
  
library(mlbench)

cpu[,"train"] <- ifelse(runif(nrow(cpu))<0.70,1,0)

trainColNum <- grep('train', names(cpu))

traincpu <- cpu[cpu$train==1,-trainColNum]

testcpu <- cpu[cpu$train==0,-trainColNum]

library(e1071)

nb\_model <- naiveBayes(PRP~.,data = traincpu)

nb\_model

summary(nb\_model)

str(nb\_model)

Output:  
  
> library(mlbench)

> cpu[,"train"] <- ifelse(runif(nrow(cpu))<0.70,1,0)

> trainColNum <- grep('train', names(cpu))

> traincpu <- cpu[cpu$train==1,-trainColNum]

> testcpu <- cpu[cpu$train==0,-trainColNum]

> library(e1071)

> nb\_model <- naiveBayes(PRP~.,data = traincpu)

> nb\_model

Naive Bayes Classifier for Discrete Predictors

Call:

naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:

Y

(0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]

0.169117647 0.573529412 0.110294118 0.058823529 0.044117647 0.022058824 0.014705882 0.007352941

Conditional probabilities:

Vendor

Y adviser amdahl apollo basf bti burroughs c.r.d cambex cdc dec

(0,20] 0.00000000 0.00000000 0.00000000 0.00000000 0.04347826 0.04347826 0.00000000 0.00000000 0.04347826 0.08695652

(20,100] 0.00000000 0.00000000 0.01282051 0.01282051 0.00000000 0.07692308 0.05128205 0.03846154 0.05128205 0.02564103

(100,200] 0.06666667 0.00000000 0.00000000 0.06666667 0.00000000 0.06666667 0.00000000 0.00000000 0.00000000 0.00000000

(200,300] 0.00000000 0.25000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.12500000 0.00000000

(300,400] 0.00000000 0.33333333 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(400,500] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(500,600] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

Vendor

Y dg formation four-phase gould harris honeywell hp ibm ipl magnuson

(0,20] 0.00000000 0.13043478 0.00000000 0.00000000 0.00000000 0.00000000 0.04347826 0.43478261 0.00000000 0.00000000

(20,100] 0.02564103 0.01282051 0.01282051 0.00000000 0.05128205 0.06410256 0.05128205 0.14102564 0.02564103 0.06410256

(100,200] 0.06666667 0.00000000 0.00000000 0.06666667 0.00000000 0.06666667 0.00000000 0.13333333 0.00000000 0.00000000

(200,300] 0.00000000 0.00000000 0.00000000 0.12500000 0.00000000 0.00000000 0.00000000 0.25000000 0.00000000 0.00000000

(300,400] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(400,500] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.66666667 0.00000000 0.00000000

(500,600] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

Vendor

Y microdata nas ncr nixdorf perkin-elmer prime siemens sperry sratus wang

(0,20] 0.00000000 0.00000000 0.08695652 0.00000000 0.00000000 0.00000000 0.04347826 0.04347826 0.00000000 0.00000000

(20,100] 0.01282051 0.05128205 0.06410256 0.02564103 0.02564103 0.03846154 0.01282051 0.01282051 0.01282051 0.02564103

(100,200] 0.00000000 0.13333333 0.13333333 0.00000000 0.00000000 0.06666667 0.13333333 0.00000000 0.00000000 0.00000000

(200,300] 0.00000000 0.00000000 0.12500000 0.00000000 0.00000000 0.00000000 0.00000000 0.12500000 0.00000000 0.00000000

(300,400] 0.00000000 0.33333333 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.33333333 0.00000000 0.00000000

(400,500] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.33333333 0.00000000 0.00000000 0.00000000

(500,600] 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 1.00000000 0.00000000 0.00000000

MYCT

Y [,1] [,2]

(0,20] 655.34783 437.198271

(20,100] 186.10256 178.062924

(100,200] 75.06667 47.673692

(200,300] 46.00000 18.299102

(300,400] 37.00000 11.489125

(400,500] 26.00000 0.000000

(500,600] 34.00000 5.656854

(600,1.55e+03] 30.00000 NA

MMIN

Y [,1] [,2]

(0,20] 591.6522 302.1641

(20,100] 1628.3077 1224.1456

(100,200] 4150.4000 2721.3210

(200,300] 5155.0000 2587.4808

(300,400] 7666.6667 4802.7770

(400,500] 13333.3333 4618.8022

(500,600] 16000.0000 0.0000

(600,1.55e+03] 8000.0000 NA

MMAX

Y [,1] [,2]

(0,20] 2634.261 2423.125

(20,100] 7700.359 4592.780

(100,200] 15133.333 7491.583

(200,300] 25621.250 7291.417

(300,400] 32000.000 0.000

(400,500] 32000.000 0.000

(500,600] 32000.000 0.000

(600,1.55e+03] 64000.000 NA

CACH

Y [,1] [,2]

(0,20] 1.565217 3.130874

(20,100] 9.756410 13.036598

(100,200] 67.133333 68.578283

(200,300] 57.750000 46.958188

(300,400] 80.000000 24.787093

(400,500] 85.333333 36.950417

(500,600] 192.000000 90.509668

(600,1.55e+03] 128.000000 NA

CHMIN

Y [,1] [,2]

(0,20] 1.086957 0.8481554

(20,100] 2.974359 2.9584761

(100,200] 6.266667 5.1888158

(200,300] 11.375000 7.3472541

(300,400] 24.000000 21.9089023

(400,500] 16.000000 8.0000000

(500,600] 16.000000 0.0000000

(600,1.55e+03] 12.000000 NA

CHMAX

Y [,1] [,2]

(0,20] 3.26087 3.466953

(20,100] 15.47436 21.640945

(100,200] 26.66667 32.052784

(200,300] 31.25000 10.793517

(300,400] 53.33333 39.408967

(400,500] 26.66667 4.618802

(500,600] 28.00000 5.656854

(600,1.55e+03] 176.00000 NA

ERP

Y [,1] [,2]

(0,20] 21.95652 5.716629

(20,100] 45.62821 23.717553

(100,200] 129.46667 61.161223

(200,300] 200.87500 44.121383

(300,400] 318.16667 48.889331

(400,500] 364.33333 16.258331

(500,600] 514.50000 125.157900

(600,1.55e+03] 978.00000 NA

> library(mlbench)

> cpu[,"train"] <- ifelse(runif(nrow(cpu))<0.70,1,0)

> trainColNum <- grep('train', names(cpu))

> traincpu <- cpu[cpu$train==1,-trainColNum]

> testcpu <- cpu[cpu$train==0,-trainColNum]

> library(e1071)

> nb\_model <- naiveBayes(PRP~.,data = traincpu)

> nb\_model

Naive Bayes Classifier for Discrete Predictors

Call:

naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:

Y

(0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]

0.12837838 0.58108108 0.15540541 0.06081081 0.02702703 0.02027027 0.01351351 0.01351351

Conditional probabilities:

Vendor

Y adviser amdahl apollo basf bti burroughs c.r.d cambex cdc dec

(0,20] 0.00000000 0.00000000 0.00000000 0.00000000 0.05263158 0.05263158 0.00000000 0.00000000 0.00000000 0.10526316

(20,100] 0.00000000 0.00000000 0.01162791 0.01162791 0.01162791 0.02325581 0.06976744 0.04651163 0.04651163 0.02325581

(100,200] 0.04347826 0.08695652 0.00000000 0.04347826 0.00000000 0.04347826 0.00000000 0.00000000 0.04347826 0.00000000

(200,300] 0.00000000 0.11111111 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.22222222 0.00000000

(300,400] 0.00000000 0.50000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.25000000 0.00000000

(400,500] 0.00000000 0.33333333 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(500,600] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.50000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

Vendor

Y dg formation four-phase gould harris honeywell hp ibm ipl magnuson

(0,20] 0.00000000 0.10526316 0.00000000 0.00000000 0.00000000 0.10526316 0.05263158 0.31578947 0.00000000 0.05263158

(20,100] 0.03488372 0.01162791 0.01162791 0.00000000 0.04651163 0.04651163 0.05813953 0.11627907 0.04651163 0.04651163

(100,200] 0.04347826 0.00000000 0.00000000 0.04347826 0.00000000 0.08695652 0.00000000 0.13043478 0.04347826 0.00000000

(200,300] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.22222222 0.00000000 0.00000000

(300,400] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(400,500] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.33333333 0.00000000 0.00000000

(500,600] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

Vendor

Y microdata nas ncr nixdorf perkin-elmer prime siemens sperry sratus wang

(0,20] 0.00000000 0.00000000 0.10526316 0.00000000 0.00000000 0.00000000 0.05263158 0.00000000 0.00000000 0.00000000

(20,100] 0.01162791 0.06976744 0.04651163 0.03488372 0.03488372 0.04651163 0.01162791 0.04651163 0.01162791 0.02325581

(100,200] 0.00000000 0.08695652 0.04347826 0.00000000 0.00000000 0.04347826 0.17391304 0.04347826 0.00000000 0.00000000

(200,300] 0.00000000 0.22222222 0.00000000 0.00000000 0.00000000 0.00000000 0.11111111 0.11111111 0.00000000 0.00000000

(300,400] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.25000000 0.00000000 0.00000000

(400,500] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.33333333 0.00000000 0.00000000 0.00000000

(500,600] 0.00000000 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000

(600,1.55e+03] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.50000000 0.00000000 0.00000000

MYCT

Y [,1] [,2]

(0,20] 627.52632 468.074824

(20,100] 191.69767 178.827032

(100,200] 70.00000 43.197222

(200,300] 40.55556 14.850178

(300,400] 31.00000 12.727922

(400,500] 25.00000 1.732051

(500,600] 34.00000 5.656854

(600,1.55e+03] 26.50000 4.949747

MMIN

Y [,1] [,2]

(0,20] 656.6316 306.5051

(20,100] 1528.5581 1036.4786

(100,200] 4325.9130 2550.5088

(200,300] 5394.4444 2721.9805

(300,400] 7327.5000 6401.7465

(400,500] 13333.3333 4618.8022

(500,600] 16000.0000 0.0000

(600,1.55e+03] 12000.0000 5656.8542

MMAX

Y [,1] [,2]

(0,20] 2740.842 2558.039

(20,100] 7454.977 4559.459

(100,200] 16868.261 8496.956

(200,300] 25732.222 9742.176

(300,400] 24655.000 14690.000

(400,500] 32000.000 0.000

(500,600] 32000.000 0.000

(600,1.55e+03] 64000.000 0.000

CACH

Y [,1] [,2]

(0,20] 1.263158 2.997075

(20,100] 10.337209 15.036122

(100,200] 48.260870 53.587836

(200,300] 51.666667 36.530809

(300,400] 92.750000 34.091788

(400,500] 85.333333 36.950417

(500,600] 192.000000 90.509668

(600,1.55e+03] 96.000000 45.254834

CHMIN

Y [,1] [,2]

(0,20] 1.210526 0.976328

(20,100] 2.627907 2.361687

(100,200] 6.173913 4.628491

(200,300] 11.777778 5.696002

(300,400] 22.000000 20.264912

(400,500] 18.666667 4.618802

(500,600] 16.000000 0.000000

(600,1.55e+03] 14.000000 2.828427

CHMAX

Y [,1] [,2]

(0,20] 3.684211 3.772011

(20,100] 13.267442 18.082442

(100,200] 25.652174 26.825376

(200,300] 26.222222 10.790943

(300,400] 48.000000 37.523326

(400,500] 29.333333 4.618802

(500,600] 28.000000 5.656854

(600,1.55e+03] 104.000000 101.823376

ERP

Y [,1] [,2]

(0,20] 22.36842 6.29118

(20,100] 44.12791 22.52099

(100,200] 126.04348 51.39904

(200,300] 200.11111 62.10967

(300,400] 283.25000 126.94192

(400,500] 374.66667 11.84624

(500,600] 514.50000 125.15790

(600,1.55e+03] 863.50000 161.92745

> summary(nb\_model)

Length Class Mode

apriori 8 table numeric

tables 8 -none- list

levels 8 -none- character

call 4 -none- call

> str(nb\_model)

List of 4

$ apriori: 'table' int [1:8(1d)] 19 86 23 9 4 3 2 2

..- attr(\*, "dimnames")=List of 1

.. ..$ Y: chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

$ tables :List of 8

..$ Vendor: table [1:8, 1:30] 0 0 0.0435 0 0 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ Vendor: chr [1:30] "adviser" "amdahl" "apollo" "basf" ...

..$ MYCT : num [1:8, 1:2] 627.5 191.7 70 40.6 31 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ MYCT: NULL

..$ MMIN : num [1:8, 1:2] 657 1529 4326 5394 7328 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ MMIN: NULL

..$ MMAX : num [1:8, 1:2] 2741 7455 16868 25732 24655 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ MMAX: NULL

..$ CACH : num [1:8, 1:2] 1.26 10.34 48.26 51.67 92.75 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ CACH: NULL

..$ CHMIN : num [1:8, 1:2] 1.21 2.63 6.17 11.78 22 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ CHMIN: NULL

..$ CHMAX : num [1:8, 1:2] 3.68 13.27 25.65 26.22 48 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ CHMAX: NULL

..$ ERP : num [1:8, 1:2] 22.4 44.1 126 200.1 283.2 ...

.. ..- attr(\*, "dimnames")=List of 2

.. .. ..$ Y : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

.. .. ..$ ERP: NULL

$ levels : chr [1:8] "(0,20]" "(20,100]" "(100,200]" "(200,300]" ...

$ call : language naiveBayes.default(x = X, y = Y, laplace = laplace)

- attr(\*, "class")= chr "naiveBayes"

Let identify the prediction of the values taken,   
  
  
R Code:   
  
  
nb\_test\_predict <- predict(nb\_model,testcpu[,-1])

table(pred=nb\_test\_predict,true=testcpu$PRP)

mean(nb\_test\_predict==testcpu$PRP)

is.factor(testcpu$PRP)

is.factor(nb\_test\_predict)

nb\_test\_predict==as.factor(nb\_test\_predict)

testcpu$PRP==as.factor(testcpu$PRP)

nb\_test\_predict == testcpu$PRP

nb\_test\_predict == is.ordered(nb\_test\_predict)

nb\_test\_predict == as.ordered(nb\_test\_predict)

testcpu$PRP==as.ordered(testcpu$PRP)

testcpu$PRP==is.ordered(testcpu$PRP)

nb\_multiple\_runs <- function(train\_fraction,n){

fraction\_correct <- rep(NA,n)

for (i in 1:n){

cpu[,"PRP"] <- ifelse(runif(nrow(cpu))<train\_fraction,1,0)

trainColNum <- grep('PRP',names(cpu))

traincpu <- cpu[cpu$PRP==1,-trainColNum]

testcpu <- cpu[cpu$PRP==0,-trainColNum]

nb\_model <- naiveBayes(PRP~.,data = cpu)

nb\_test\_predict <- predict(nb\_model,testcpu[,-1])

fraction\_correct[i] <- mean(nb\_test\_predict==testcpu$PRP)

}

return(fraction\_correct)

}

fraction\_correct\_predictions <- nb\_multiple\_runs(0.8,20)

na.rm=TRUE

fraction\_correct\_predictions

summary(fraction\_correct\_predictions)

sd(fraction\_correct\_predictions)  
  
  
Output:

|  |
| --- |
| > nb\_test\_predict <- predict(nb\_model,testcpu[,-1])  > table(pred=nb\_test\_predict,true=testcpu$PRP)  true  pred (0,20] (20,100] (100,200] (200,300] (300,400] (400,500] (500,600] (600,1.55e+03]  (0,20] 12 4 0 0 0 0 0 0  (20,100] 0 28 0 0 0 0 0 0  (100,200] 0 2 3 2 0 0 0 0  (200,300] 0 0 1 0 0 0 0 0  (300,400] 0 0 0 0 1 0 0 0  (400,500] 0 0 0 0 0 0 0 0  (500,600] 0 1 0 2 2 1 0 0  (600,1.55e+03] 0 0 0 0 0 0 0 2  > mean(nb\_test\_predict==testcpu$PRP)  Error in nb\_test\_predict == testcpu$PRP :  comparison of these types is not implemented  In addition: Warning message:  In mean(nb\_test\_predict == testcpu$PRP) :  Incompatible methods ("Ops.factor", "Ops.ordered") for "=="  > is.factor(testcpu$PRP)  [1] TRUE  > is.factor(nb\_test\_predict)  [1] TRUE  > nb\_test\_predict==as.factor(nb\_test\_predict)  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [26] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [51] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  > testcpu$PRP==as.factor(testcpu$PRP)  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [26] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [51] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  > nb\_test\_predict == testcpu$PRP  Error in nb\_test\_predict == testcpu$PRP :  comparison of these types is not implemented  In addition: Warning message:  Incompatible methods ("Ops.factor", "Ops.ordered") for "=="  > nb\_test\_predict == is.ordered(nb\_test\_predict)  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [21] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [41] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [61] FALSE  > nb\_test\_predict == as.ordered(nb\_test\_predict)  Error in nb\_test\_predict == as.ordered(nb\_test\_predict) :  comparison of these types is not implemented  In addition: Warning message:  Incompatible methods ("Ops.factor", "Ops.ordered") for "=="  > testcpu$PRP==as.ordered(testcpu$PRP)  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [26] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  [51] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  > testcpu$PRP==is.ordered(testcpu$PRP)  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [21] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [41] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  [61] FALSE |
|  |
| |  | | --- | | > | |

These are the end results where the required values and there is no reason for us to make the dataset to be converted to factor as all values are filled and there are no null values, and also make the identified values no need to filling the information of the dataset, identify what percentage of the all the attributes are nearest to the class attribute.   
  
  
  
  
  
Referrences:  
  
  
[1] <https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/machine.names>  
  
[2] <https://stat.ethz.ch/pipermail/r-devel/2003-February/026002.html>  
  
[3] <http://r.789695.n4.nabble.com/Segfault-with-match-td4705296.html>  
  
[4] <https://github.com/dgrtwo/gganimate/issues/16>