**Is there a Millionaire Algorithm?**



Anil Poonai

Table of Contents

CART(Classification and Regression Trees) ISLR Example 3

CART on AOS Stock Range Data 9

CART on Cardiac Data 14

Comparison of Methodologies on Stock Data 21

Appendix 22

Bibliography 37

Baruch College, STA-CIS 3920, Final Anil Poonai 37

6 August 2020

I.

Code is in the appendix.

A common problem was that I had to make variables into factors for the example to work.

Another problem is that I could not put my name into the graphs that involved tree graphs.

I picked the CART methodology. I also have some comments in the code that could be helpful to understand the differences, which only exists when I am calculating the percentage from the prediction table as they are higher in my code than they are in the example. Meaning they updated the functions to make them more feasible. The examples are from the ISLR text on page 324 section 8.3. A screenshot of a video game

Description automatically generatedClassification tree for high sales relevant to every other variable in Carseats dataset.

A close up of a map

Description automatically generatedCross validation tree size vs cross validation error rate

A close up of a map

Description automatically generated Cost complexity parameter size vs cross validation error rate

A picture containing boat, flying, large, filled

Description automatically generatedPruned categorical tree Pruned categorical tree with a 9 type treeA picture containing small, boat, filled, table

Description automatically generatedPruned categorical tree with a 15 best or a 15 type treeA picture containing screenshot, boat, person, water

Description automatically generatedRegression tree for Boston dataset.A picture containing boat

Description automatically generatedCross validation tree size vs error rateA picture containing boat, table, airplane, small

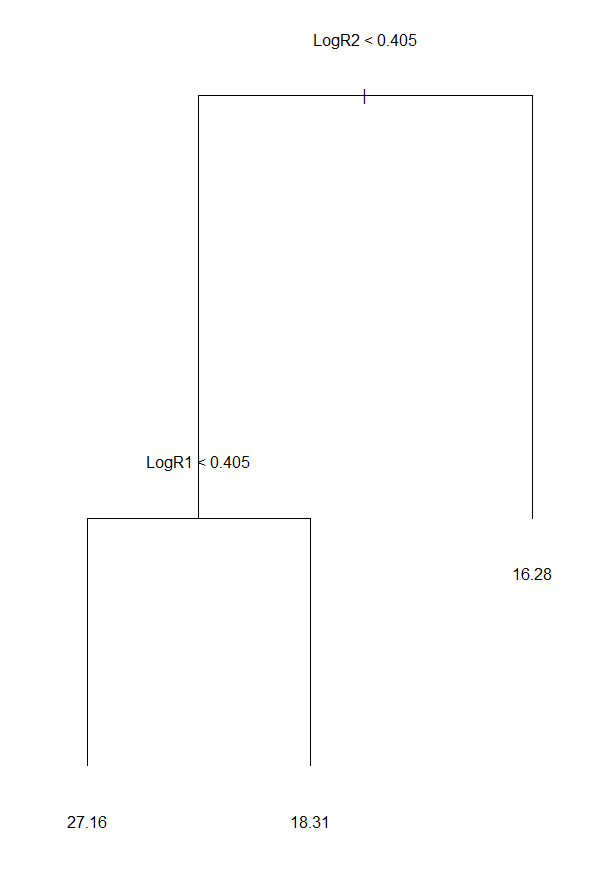
Description automatically generatedPruned tree with 5 type treeA picture containing map, text, boat, photo

Description automatically generatedUnpruned tree vs actual results on test data for mean squared error.

**II.**

**A.**

I used the CART methodology on the stock market data, ticker: AOS, with the HIGH and LOW risk category as the y variable for the categorical tree and the adjusted closing price as the y variable for the regression tree. The categorical tree did not work out so well as it kept having one node only so graphs could not be shown and it kept being 52% accurate, which is fine compared to all of the other methodologies used on the data. I also could not cross validate it as it only had one node. The regression tree worked a little better graph wise as it made 3 nodes and was able to be shown and it had a mean squared error of 355.3712. Although I realize it does not make entire sense to use both the classification and regression tree in the way described in the ISLR text as they are two different methods instead of a combined one. I tried to find two variables that have similar significance like I mentioned previously it was the HIGH/LOW Risk variable for the categorical tree and the adjusted closing price variable for the regression variable. Although categorical method only had one node, I would still pick it as its around the same percentage as the other methods used on the stock data and the regression tree had a really high mean squared error.

Regression tree for AOS

Cross validation tree size vs error rate A screenshot of a video game

Description automatically generated

A picture containing table, boat

Description automatically generatedUnpruned tree vs actual results on test data for mean squared error.

This is the classification space between logged lagged ranged 1 and 3. There seems to be a correlation between the two in regards to the risk variable.

A picture containing clock

Description automatically generated

**B.**

I then used the CART methodology on the cardiac dataset from the NIH. When I used the classification tree I used the death column as the y variable and it ended up having 5 nodes and a .4% error, Which is really good objectively. I decided to then cross validate and use a pruning technique to get rid of excess data with the hope that it would be even more accurate. Nothing changed. I thenhappened to do the regression tree with systolic blood pressure as the y variablke as a study in Oxford Academin from 2005 highly correlated sdp with deaths, cited in the bibliography right after the appendix. This didn’t seem to work as well as it only had one node so graphs and pruning couldn’t be made. Although it did have a low mean squared error - .01631906. I would stick with the classification tree for this as it is over 99% accurate.

**A screenshot of a cell phone

Description automatically generated**

Classification tree for Cardiac dataset

**A screenshot of a cell phone

Description automatically generated**

Cross validated tree size vs error rate

**A screenshot of a cell phone

Description automatically generated**

Cost complexity parameter size vs cross validation error rate

**A screenshot of a cell phone

Description automatically generated**

Pruned classification tree

**A screenshot of a cell phone

Description automatically generated**

Unpruned tree vs actual results on test data for mean squared error.

This is the classification space between sbp and bhr regarding if a person has died. It seems to be splattered out and I can’t clearly see the definition between them.

A picture containing clock

Description automatically generated

**III.**

The CART performance on my stock range data compares about average to the other methods such as knn, naïve Bayes and logistic regression where I got an accuracy rate of 55%, 52% and 45% respectively. CART got me a 52% accuracy rate, which is on par with naïve Bayes. Admittingly I had to go back and redo the knn methodology as I used repetitive data in the dataset when calculating it the first time, but it is fixed now. I tried to reverse the logistic regression methodology in order to get it to be 55% as it seemed reasonable to flip the classifications if it was less than 50%. That did not work as it became even lower. So, for AOS, regarding its risk it is not going to be too predictable no matter what methodology used. The problem with this is that the stock market is not predictable from a day to day basis and is really only predictable over a long period of time due to factors such as earnings and inflation. Another problem with this that is much more apart of decision trees is the splitting of the data and the nodes used as these drastically change how the tree looks and performs. It also stops us from being able to prune it when there are not enough nodes. Also depending on the whether it is a classification or regression tree we need either ideal labels or a continuous variable which is not always the easiest to find but is derivative of some kind or preexisting variables. I tackle problems with the other assignments which will be on my GitHub at the end of this semester along with this document and the R code. <https://github.com/DevonARP>

**Appendix**

**I.**

R version 4.0.2 (2020-06-22) -- "Taking Off Again"

Copyright (C) 2020 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

[Previously saved workspace restored]

> #Fitting Classification Trees

> #The tree graphs that take titles well, so my name couldn't be placed on those but all of the other graphs have my name.

> install.packages("tree")

Installing package into ‘C:/Users/poona/Documents/R/win-library/4.0’

(as ‘lib’ is unspecified)

--- Please select a CRAN mirror for use in this session ---

trying URL 'https://cloud.r-project.org/bin/windows/contrib/4.0/tree\_1.0-40.zip'

Content type 'application/zip' length 178403 bytes (174 KB)

downloaded 174 KB

package ‘tree’ successfully unpacked and MD5 sums checked

Warning: cannot remove prior installation of package ‘tree’

Warning: restored ‘tree’

The downloaded binary packages are in

C:\Users\poona\AppData\Local\Temp\Rtmp2llIH0\downloaded\_packages

Warning message:

In file.copy(savedcopy, lib, recursive = TRUE) :

problem copying C:\Users\poona\Documents\R\win-library\4.0\00LOCK\tree\libs\x64\tree.dll to C:\Users\poona\Documents\R\win-library\4.0\tree\libs\x64\tree.dll: Permission denied

> install.packages("ISLR")

Installing package into ‘C:/Users/poona/Documents/R/win-library/4.0’

(as ‘lib’ is unspecified)

trying URL 'https://cloud.r-project.org/bin/windows/contrib/4.0/ISLR\_1.2.zip'

Content type 'application/zip' length 2924406 bytes (2.8 MB)

downloaded 2.8 MB

package ‘ISLR’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\poona\AppData\Local\Temp\Rtmp2llIH0\downloaded\_packages

> library(tree)

> library(ISLR)

> attach(Carseats)

> High=ifelse(Sales<=8,"No","Yes")

> Carseats=data.frame(Carseats,High)

> tree.carseats=tree(as.factor(High)~.-Sales,Carseats)

> #Had to change High to a factor data type, they didn't have to do that at the time of the ISLR example using the tree package.

> summary(tree.carseats)

Classification tree:

tree(formula = as.factor(High) ~ . - Sales, data = Carseats)

Variables actually used in tree construction:

[1] "ShelveLoc" "Price" "Income" "CompPrice" "Population" "Advertising" "Age" "US"

Number of terminal nodes: 27

Residual mean deviance: 0.4575 = 170.7 / 373

Misclassification error rate: 0.09 = 36 / 400

> plot(tree.carseats)

> text(tree.carseats,pretty=0)

> tree.carseats

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

\* denotes terminal node

1) root 400 541.500 No ( 0.59000 0.41000 )

2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )

4) Price < 92.5 46 56.530 Yes ( 0.30435 0.69565 )

8) Income < 57 10 12.220 No ( 0.70000 0.30000 )

16) CompPrice < 110.5 5 0.000 No ( 1.00000 0.00000 ) \*

17) CompPrice > 110.5 5 6.730 Yes ( 0.40000 0.60000 ) \*

9) Income > 57 36 35.470 Yes ( 0.19444 0.80556 )

18) Population < 207.5 16 21.170 Yes ( 0.37500 0.62500 ) \*

19) Population > 207.5 20 7.941 Yes ( 0.05000 0.95000 ) \*

5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )

10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )

20) CompPrice < 124.5 96 44.890 No ( 0.93750 0.06250 )

40) Price < 106.5 38 33.150 No ( 0.84211 0.15789 )

80) Population < 177 12 16.300 No ( 0.58333 0.41667 )

160) Income < 60.5 6 0.000 No ( 1.00000 0.00000 ) \*

161) Income > 60.5 6 5.407 Yes ( 0.16667 0.83333 ) \*

81) Population > 177 26 8.477 No ( 0.96154 0.03846 ) \*

41) Price > 106.5 58 0.000 No ( 1.00000 0.00000 ) \*

21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )

42) Price < 122.5 51 70.680 Yes ( 0.49020 0.50980 )

84) ShelveLoc: Bad 11 6.702 No ( 0.90909 0.09091 ) \*

85) ShelveLoc: Medium 40 52.930 Yes ( 0.37500 0.62500 )

170) Price < 109.5 16 7.481 Yes ( 0.06250 0.93750 ) \*

171) Price > 109.5 24 32.600 No ( 0.58333 0.41667 )

342) Age < 49.5 13 16.050 Yes ( 0.30769 0.69231 ) \*

343) Age > 49.5 11 6.702 No ( 0.90909 0.09091 ) \*

43) Price > 122.5 77 55.540 No ( 0.88312 0.11688 )

86) CompPrice < 147.5 58 17.400 No ( 0.96552 0.03448 ) \*

87) CompPrice > 147.5 19 25.010 No ( 0.63158 0.36842 )

174) Price < 147 12 16.300 Yes ( 0.41667 0.58333 )

348) CompPrice < 152.5 7 5.742 Yes ( 0.14286 0.85714 ) \*

349) CompPrice > 152.5 5 5.004 No ( 0.80000 0.20000 ) \*

175) Price > 147 7 0.000 No ( 1.00000 0.00000 ) \*

11) Advertising > 13.5 45 61.830 Yes ( 0.44444 0.55556 )

22) Age < 54.5 25 25.020 Yes ( 0.20000 0.80000 )

44) CompPrice < 130.5 14 18.250 Yes ( 0.35714 0.64286 )

88) Income < 100 9 12.370 No ( 0.55556 0.44444 ) \*

89) Income > 100 5 0.000 Yes ( 0.00000 1.00000 ) \*

45) CompPrice > 130.5 11 0.000 Yes ( 0.00000 1.00000 ) \*

23) Age > 54.5 20 22.490 No ( 0.75000 0.25000 )

46) CompPrice < 122.5 10 0.000 No ( 1.00000 0.00000 ) \*

47) CompPrice > 122.5 10 13.860 No ( 0.50000 0.50000 )

94) Price < 125 5 0.000 Yes ( 0.00000 1.00000 ) \*

95) Price > 125 5 0.000 No ( 1.00000 0.00000 ) \*

3) ShelveLoc: Good 85 90.330 Yes ( 0.22353 0.77647 )

6) Price < 135 68 49.260 Yes ( 0.11765 0.88235 )

12) US: No 17 22.070 Yes ( 0.35294 0.64706 )

24) Price < 109 8 0.000 Yes ( 0.00000 1.00000 ) \*

25) Price > 109 9 11.460 No ( 0.66667 0.33333 ) \*

13) US: Yes 51 16.880 Yes ( 0.03922 0.96078 ) \*

7) Price > 135 17 22.070 No ( 0.64706 0.35294 )

14) Income < 46 6 0.000 No ( 1.00000 0.00000 ) \*

15) Income > 46 11 15.160 Yes ( 0.45455 0.54545 ) \*

> set.seed (2)

> train=sample (1:nrow(Carseats),200)

> Carseats.test=Carseats[-train,]

> High.test=High[-train]

> tree.carseats=tree(as.factor(High)~.-Sales,Carseats,subset=train)

> tree.pred=predict(tree.carseats,Carseats.test,type="class")

> table(tree.pred,High.test)

High.test

tree.pred No Yes

No 104 33

Yes 13 50

> (104+50)/200

[1] 0.77

> #A little better than the examples.

> set.seed (3)

> cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)

> names(cv.carseats)

[1] "size" "dev" "k" "method"

> cv.carseats

$size

[1] 21 19 14 9 8 5 3 2 1

$dev

[1] 74 76 81 81 75 77 78 85 81

$k

[1] -Inf 0.0 1.0 1.4 2.0 3.0 4.0 9.0 18.0

$method

[1] "misclass"

attr(,"class")

[1] "prune" "tree.sequence"

> par(mfrow =c(1,2))

> plot(cv.carseats$size,cv.carseats$dev,type="b",main = "Anil Poonai")

> plot(cv.carseats$k,cv.carseats$dev,type="b",main = "Anil Poonai")

> prune.carseats =prune.misclass(tree.carseats,best =9)

> plot(prune.carseats)

> text(prune.carseats,pretty=0)

> tree.pred=predict(prune.carseats,Carseats.test,type="class")

> table(tree.pred,High.test)

High.test

tree.pred No Yes

No 97 25

Yes 20 58

> (97+58)/200

[1] 0.775

> #A little better again.

> prune.carseats=prune.misclass(tree.carseats,best =15)

> plot(prune.carseats)

> text(prune.carseats,pretty=0)

> tree.pred=predict(prune.carseats,Carseats.test,type="class")

> table(tree.pred,High.test)

High.test

tree.pred No Yes

No 102 30

Yes 15 53

> (102+53)/200

[1] 0.775

> #Upgraded their function a bit didn't they.

> #Fitting Restression Trees

> library(MASS)

> set.seed(1)

> train=sample(1:nrow(Boston), nrow(Boston)/2)

> tree.boston=tree(medv~.,Boston,subset=train)

> summary(tree.boston)

Regression tree:

tree(formula = medv ~ ., data = Boston, subset = train)

Variables actually used in tree construction:

[1] "rm" "lstat" "crim" "age"

Number of terminal nodes: 7

Residual mean deviance: 10.38 = 2555 / 246

Distribution of residuals:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-10.1800 -1.7770 -0.1775 0.0000 1.9230 16.5800

> plot(tree.boston)

> text(tree.boston,pretty=0)

> cv.boston=cv.tree(tree.boston)

> plot(cv.boston$size,cv.boston$dev,type='b',main = "Anil Poonai")

> prune.boston=prune.tree(tree.boston,best=5)

> plot(prune.boston)

> text(prune.boston,pretty =0)

> yhat=predict(tree.boston,newdata=Boston[-train,])

> boston.test=Boston[-train,"medv"]

> plot(yhat,boston.test,main = "Anil Poonai")

> abline(0,1)

> mean((yhat-boston.test)^2)

[1] 35.28688

> #Higher than what they got.

**II.**

**A.**

> #AOS Classification

> library(tidyverse)

Registered S3 method overwritten by 'cli':

method from

print.tree tree

-- Attaching packages -------------------------------------------------------------------------------------------------------------------------------------------- tidyverse 1.3.0 --

v ggplot2 3.3.0 v purrr 0.3.4

v tibble 3.0.1 v dplyr 0.8.5

v tidyr 1.0.2 v stringr 1.4.0

v readr 1.3.1 v forcats 0.5.0

-- Conflicts ----------------------------------------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --

x dplyr::filter() masks stats::filter()

x dplyr::lag() masks stats::lag()

x dplyr::select() masks MASS::select()

> AOS <- read\_csv("C:/Users/poona/Desktop/School/AOS.csv")

Parsed with column specification:

cols(

.default = col\_double(),

Date = col\_character(),

Direction = col\_character(),

HiLoRisk = col\_character()

)

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

> tree.AOS=tree(as.factor(HiLoRisk)~LogR1+LogR2+LogR3,AOS)

> summary(tree.AOS)

Classification tree:

tree(formula = as.factor(HiLoRisk) ~ LogR1 + LogR2 + LogR3, data = AOS)

Variables actually used in tree construction:

character(0)

Number of terminal nodes: 1

Residual mean deviance: 1.387 = 5057 / 3647

Misclassification error rate: 0.4975 = 1815 / 3648

> #Single Node and an error rate of 49.75%. Not too promising so far.

> #Can't plot single Nodes.

> set.seed(4)

> train=sample(3648,2000)

> AOS.test=AOS[-train,]

> tree.AOS=tree(as.factor(HiLoRisk)~LogR1+LogR2+LogR3,AOS,subset = train)

> tree.pred=predict(tree.AOS,AOS.test,type="class")

> table(tree.pred,AOS$HiLoRisk[-train])

tree.pred HiRisk LoRisk

HiRisk 619 588

LoRisk 195 246

> 856/1648

[1] 0.5194175

> #52.49 success rate

> cv.AOS=cv.tree(tree.AOS,FUN = prune.misclass)

Error in prune.tree(tree = list(frame = list(var = 1L, n = 1800, dev = 2493.46063759019, :

can not prune singlenode tree

> #can't cross validate as it is single noded.

> #AOS Regression

> tree.AOS=tree(`Adj Close`~LogR1+LogR2+LogR3,data=AOS,subset=train)

> summary(tree.AOS)

Regression tree:

tree(formula = `Adj Close` ~ LogR1 + LogR2 + LogR3, data = AOS,

subset = train)

Variables actually used in tree construction:

[1] "LogR2" "LogR1"

Number of terminal nodes: 3

Residual mean deviance: 356.5 = 711800 / 1997

Distribution of residuals:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-24.490 -14.480 -6.577 0.000 17.890 46.420

> plot(tree.AOS)

> text(tree.AOS,pretty=0)

> cv.AOS=cv.tree(tree.AOS)

> plot(cv.AOS$size,cv.AOS$dev,type='b')

> prune.AOS=prune.tree(tree.AOS,best=3)

> plot(prune.AOS)

> text(prune.AOS,pretty=0)

> yhat=predict(tree.AOS,newdata=AOS[-train,])

> AOS.test=AOS$`Adj Close`[-train]

> plot(yhat,AOS.test)

> abline(0,1)

> mean((yhat-AOS.test)^2)

[1] 355.3712

> #355.3712

#Space

> AOS$HiLoRisk=ifelse(AOS$HiLoRisk=='Present',1,0)

> glm.fit=glm(HiLoRisk~LogR1+LogR3,data=AOS,family=binomial)

> newdata=ProbeX

> X=AOS[,c(17,19)]

> StdX=apply(X,2,scale)

> dfX=as.data.frame(StdX)

> glm.probs=predict(glm.fit,newdata=dfX,type="response")

> StCard2=as.data.frame(cbind(dfX,AOS$HiLoRisk))

> glm.fit2=glm(AOS$HiLoRisk~LogR1+LogR3,data=StCard2,family=binomial)

> names(StProbeX)[1]="LogR1"

> names(StProbeX)[2]="LogR3"

> glm.probe=predict(glm.fit2,newdata=StProbeX,type="response")

> glm.y=glm.probe

> glm.y[glm.probe>.5]=1

> glm.y[glm.probe<.5]=0

> ProbeGlm(ProbeX=StProbeX,ProbeYhat = c(glm.y),InX = dfX,InY = AOS$HiLoRisk,xr=c(-3,3),yr=c(-3,3))

**B.**

> #Cardiac Classification

> Cardiac <- read\_csv("C:/Users/poona/Downloads/Cardiac.csv")

Parsed with column specification:

cols(

.default = col\_double()

)

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

> View(Cardiac)

> Dead=ifelse(Cardiac$death==0,"Yes","No")

> Cardiac=data.frame(Cardiac,Dead)

> tree.Cardiac=tree(as.factor(Dead)~.-death,data=Cardiac)

> summary(tree.Cardiac)

Classification tree:

tree(formula = as.factor(Dead) ~ . - death, data = Cardiac)

Variables actually used in tree construction:

[1] "hardness" "newMI" "deltaEF" "basedp"

Number of terminal nodes: 5

Residual mean deviance: 0.01627 = 8.997 / 553

Misclassification error rate: 0.003584 = 2 / 558

> #5 Nodes and .4% error rate

> plot(tree.Cardiac)

> text(tree.Cardiac,pretty=0)

> set.seed(5)

> train=sample(558,558/2)

> Cardiac.test=Cardiac[-train,]

> tree.Cardiac=tree(as.factor(Dead)~.-death,data=Cardiac,subset=train)

> tree.pred=predict(tree.Cardiac,Cardiac.test,type="class")

> table(tree.pred,Cardiac$Dead[-train])

tree.pred No Yes

No 261 2

Yes 6 10

> 271/279

[1] 0.9713262

> #97.13% success rate

> cv.Cardiac=cv.tree(tree.Cardiac,FUN=prune.misclass)

> names(cv.Cardiac)

[1] "size" "dev" "k" "method"

> par(mfrow=c(1,2))

> plot(cv.Cardiac$size,cv.Cardiac$dev,type="b")

> plot(cv.Cardiac$k,cv.Cardiac$dev,type="b")

> prune.Cardiac=prune.misclass(tree.Cardiac,best=4)

> plot(prune.Cardiac)

> text(prune.Cardiac,pretty=0)

> tree.pred=predict(prune.Cardiac,Cardiac.test,type="class")

> table(tree.pred,Cardiac$Dead[-train])

tree.pred No Yes

No 261 2

Yes 6 10

> #No difference

> prune.Cardiac=prune.misclass(tree.Cardiac,best=5)

Warning message:

In prune.tree(tree = tree.Cardiac, best = 5, method = "misclass") :

best is bigger than tree size

> #Best can't be any bigger.

> #Cardiac Regression

> tree.Cardiac=tree(sbp~-sbp,data=Cardiac,subset=train)

> #I'm using sdp due to a study that heavily correlates systolic blood pressure with death.

> summary(tree.Cardiac)

Regression tree:

tree(formula = sbp ~ -sbp, data = Cardiac, subset = train)

Number of terminal nodes: 1

Residual mean deviance: 0.02057 = 5.719 / 278

Distribution of residuals:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-0.32390 -0.10080 -0.02649 0.00000 0.09433 0.60180

> #Single Noded so graphs won't work, neither will prune

> yhat=predict(tree.Cardiac,newdata=Cardiac[-train,])

> Cardiac.test=Cardiac$sbp[-train]

> plot(yhat,Cardiac.test)

> abline(0,1)

> mean((yhat-Cardiac.test)^2)

[1] 0.01631906

> #.01631906

> Cardiac <- read\_csv("C:/Users/poona/Downloads/Cardiac.csv")

> Dead=ifelse(Cardiac$death==0,"Yes","No")

> Cardiac$Dead=ifelse(Cardiac$death==0,1,0)

> glm.fit=glm(Dead~sbp+bhr,data=Cardiac,family=binomial)

> newdata=ProbeX

> View(Cardiac)

> X=Cardiac[,c(1,5)]

> StdX=apply(X,2,scale)

> dfX=as.data.frame(StdX)

> glm.probs=predict(glm.fit,newdata=dfX,type="response")

> StCard2=as.data.frame(cbind(dfX,Cardiac$Dead))

> glm.fit2=glm(Cardiac$Dead~sbp+bhr,data=StCard2,family=binomial)

> names(StProbeX)[1]="bhr"

> names(StProbeX)[2]="sbp"

> glm.probe=predict(glm.fit2,newdata=StProbeX,type="response")

> glm.y=glm.probe

> glm.y[glm.probe>.5]=1

> glm.y[glm.probe<.5]=0

> ProbeGlm(ProbeX=StProbeX,ProbeYhat = c(glm.y),InX = dfX,InY = Cardiac$Dead,xr=c(-3,3),yr=c(-3,3))

**Bibliography**

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. New York: Springer.

Says, T. (2017, November 26), Machine Learning cartoon: Marketoonist: Tom Fishburne. Retrieved August 6, 2020, from <https://marketoonist.com/2017/11/machine-learning.html>

Yalcin, Bektas Murat, et al. “Which Anthropometric Measurements Is Most Closely Related to Elevated Blood Pressure?” *OUP Academic*, Oxford University Press, 17 June 2005, academic.oup.com/fampra/article/22/5/541/609092.