Bankruptcy.R

Sharada Sarangan

2023-03-16

```
# Final Project

# Set working directory to current required folder

library(class)
library(MASS)
library(e1071)
library(tree)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(smotefamily)
library(boot)
library(reshape2)
library(ggplot2)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-6
```

```
options(scipen=999)

# Part 1
# EXPLORATORY EDA AND SETTING UP TRAIN AND TEST SET
dat <- read.csv("bankrupcy_data.csv")
table(dat$bankrupt)
```

```
##
##    0    1
## 6599  220
```

```r
# we drop the repeated and redundant columns to prevent multi-colinearity
# based on domain knowledge, since several measures correspond to the same thing
# but have been recoded in different ways for accounting purposes.
dat <- dat[,c(1,7:16,19:22, 25:38,40,41,45:77,83, 86, 87:92, 94)]
# str(dat)

# setting up training and testing set
# separating the dependent variable
yes <- subset(dat, dat$bankrupt == 1)
no <- subset(dat, dat$bankrupt == 0)

# setting seed
set.seed(112233)

# Randomly selecting 1550 rows from both groups of
# of numbers (1-->50 and 1-->1500)
test.yes <- sample(1:nrow(yes),50)
test.no <- sample(1:nrow(no),1500)

# separating our testing set
dat.test <- rbind(yes[test.yes,],no[test.no,])
# checking our train set
table(dat.test$bankrupt)
```

```
##
##    0    1
## 1500   50
```

```r
# creating the training set
newyes <- yes[-test.yes,]
newno <- no[-test.no,]

# our base train set
train_og <- rbind(newyes,newno)

#  Check results
table(train_og$bankrupt)
```

```
##
##    0    1
## 5099  170
```

```r
# bootstrapped train set (with replacement)
train.newyes <- sample(1:nrow(newyes),500, replace=TRUE)
train.newno <- sample(1:nrow(newno),3000)

# our base train set
train_boot <- rbind(newyes[train.newyes,],newno[train.newno,])

#  Check results
table(train_boot$bankrupt)
```

```
##
##    0    1
## 3000  500
```

```
# ADAS bootstrapping - Generate synthetic positive instances using ADASYN algorithm
adas_train <- ADAS(train_og,train_og$bankrupt,K=5)
train_adas <- adas_train$data

# dropping the last class column since its the same as bankrupt
train_adas <- train_adas[,c(1:73)]

# lets look at our synthetic values
# adas_train$syn_data

# checking our distribution
table(train_adas$bankrupt)
```

```
##
##    0    1
## 5099 5152
```

```
# removing unnecessary variables
rm(yes, no, test.yes, test.no, newyes, newno, train.newyes,
   train.newno)


# we run an initial log_reg to identify significant variables
glm_var <- glm(bankrupt ~ ., data=train_og, family='binomial')
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_var)
```

```
##
## Call:
## glm(formula = bankrupt ~ ., family = "binomial", data = train_og)
##
## Deviance Residuals:
##    Min     1Q  Median     3Q    Max
##  -8.49   0.00    0.00   0.00   8.49
##
## Coefficients:
##                                       Estimate                  Std. Error
## (Intercept)   1883979708516677451776.0000000    1699064189984867.7500000
## X6           -465183775008882163712.0000000      15916642620196.4003906
## X7            388119709635854991360.0000000      13294282568795.6269531
## X8               283983102233256640.0000000          5713580607.6899691
## X9           -222377426493851041792.0000000       7609152820787.1191406
## X10              137351940171298880.0000000          2991145588.2508001
```

```
## X11                    -2781.8383180              0.0003187
## X12                     6406.0329957              0.0003657
## X13     -11674391317988228.0000000      199034816.6428339
## X14                 -1128938.4341195              0.0086069
## X15       -903101588679473.1250000        7154954.4825070
## X18      -1590748969963942.5000000       48770603.0520286
## X19      -6469232962841985.0000000       66792839.3558250
## X20       1176030986828447.0000000      107148022.4485826
## X21                  2272229.4327178              0.0581123
## X24       -407791986138473.4375000       69223260.9213606
## X25      -1992878399596448.5000000      108281472.1661287
## X26      12869912493968414.0000000     1253497879.9802492
## X27     -11458159198611292.0000000     1250207134.1004519
## X28        162820692948075.0937500       81826413.3755742
## X29                   -2761.4022618              0.0003311
## X30                  388483.4184615              0.0082377
## X31       1484647239702127.2500000       99098898.8917462
## X32       2226928032426705.2500000      111528249.2206729
## X33                 -1246098.0609360              0.0245209
## X34                   51066.5822066              0.0035889
## X35       5273774632004118.0000000      121049302.7006431
## X36                  237993.2494538              0.0067400
## X37       3390971075940789.5000000       87957268.2152067
## X39      -1059782924336249.1250000       40519475.6397451
## X40       9559741624139022.0000000      293710759.3670629
## X44      -3042218055542296.0000000      336243559.8469479
## X45       -838239631906503.0000000       21336378.3386176
## X46                 -889060.6917348              0.0038217
## X47                 -160828.6135705              0.0046055
## X48                   -2248.4327626              0.0003031
## X49                    7151.8508011              0.0004539
## X50        -46762232216170.1875000       57622024.0116544
## X51                  1067902.9366004              0.0205171
## X52        646895347379493.1250000       34289241.5966871
## X53                 -115111.2878664              0.0032202
## X54  -22096259711322245038.0000000   2259369896535990.0000000
## X55        -37517750018965.8671875        9721379.4010338
## X56    6051998111462512394.0000000    618823786425694.3750000
## X57      -1377838456967522.5000000       10559349.2127029
## X58                 -416671.9369221              0.0067287
## X59                 -133962.9149332              0.0018808
## X60  -19709838769708733562.0000000   2015352757388043.0000000
## X61       3456621710105407.0000000       86105188.2379008
## X62        -20467877072612.1250000       84167887.4097392
## X63                  -15126.8207547              0.0015570
## X64        164031833095332.2187500       13773798.5227675
## X65      -5596858581754939.0000000      430285063.0177115
## X66       2869397278865996.0000000      515611972.4494765
## X67                 -132198.7820026              0.0015964
## X68       1625186356087748.5000000       67526247.5542146
## X69      -2149618684838555.0000000       81775877.8324365
## X70      -3523319003549844.5000000       58273454.8268475
## X71                    8341.2562027              0.0004010
## X72                    1430.8472679              0.0003307
```

```
## X73          -7410886968343630 4.0000000          1718597696.3550916
## X74                      -21443.3707844                   0.0003355
## X75            -8808095994031961.0000000           709063857.2721953
## X76             -153246393465485.2812500             6979420.6307562
## X82             -868977826373918.1250000            44627211.4787531
## X85             1383878271926223.7500000            35772501.6389378
## X86            -1201412510424466.0000000            59351605.5431496
## X87                      -57991.6915018                   0.0024935
## X88             1964706613133726.7500000            86332886.1440868
## X89             5958137359182219.0000000            89686543.6969836
## X90            -2298060225279485.5000000           125109021.3483346
## X91           -14294886339432660.0000000           492426317.5584345
## X93             1652645407011857.0000000            76495068.7825037
##                  z value            Pr(>|z|)
## (Intercept)      1108834 <0.0000000000000002 ***
## X6             -29226250 <0.0000000000000002 ***
## X7              29194483 <0.0000000000000002 ***
## X8              49703176 <0.0000000000000002 ***
## X9             -29224991 <0.0000000000000002 ***
## X10             45919510 <0.0000000000000002 ***
## X11             -8728171 <0.0000000000000002 ***
## X12             17518963 <0.0000000000000002 ***
## X13            -58655021 <0.0000000000000002 ***
## X14           -131167341 <0.0000000000000002 ***
## X15           -126220452 <0.0000000000000002 ***
## X18            -32616963 <0.0000000000000002 ***
## X19            -96855187 <0.0000000000000002 ***
## X20             10975760 <0.0000000000000002 ***
## X21             39100662 <0.0000000000000002 ***
## X24             -5890968 <0.0000000000000002 ***
## X25            -18404611 <0.0000000000000002 ***
## X26             10267199 <0.0000000000000002 ***
## X27             -9165009 <0.0000000000000002 ***
## X28              1989830 <0.0000000000000002 ***
## X29             -8339408 <0.0000000000000002 ***
## X30             47158948 <0.0000000000000002 ***
## X31             14981471 <0.0000000000000002 ***
## X32             19967390 <0.0000000000000002 ***
## X33            -50817859 <0.0000000000000002 ***
## X34             14228939 <0.0000000000000002 ***
## X35             43567162 <0.0000000000000002 ***
## X36             35310670 <0.0000000000000002 ***
## X37             38552483 <0.0000000000000002 ***
## X39            -26154902 <0.0000000000000002 ***
## X40             32548149 <0.0000000000000002 ***
## X44             -9047662 <0.0000000000000002 ***
## X45            -39286875 <0.0000000000000002 ***
## X46           -232636868 <0.0000000000000002 ***
## X47            -34920973 <0.0000000000000002 ***
## X48             -7418175 <0.0000000000000002 ***
## X49             15755695 <0.0000000000000002 ***
## X50              -811534 <0.0000000000000002 ***
## X51             52049365 <0.0000000000000002 ***
## X52             18865840 <0.0000000000000002 ***
```

```
## X53              -35746949  <0.0000000000000002 ***
## X54                -977983  <0.0000000000000002 ***
## X55               -3859303  <0.0000000000000002 ***
## X56                 977984  <0.0000000000000002 ***
## X57             -130485168  <0.0000000000000002 ***
## X58              -61924315  <0.0000000000000002 ***
## X59              -71227328  <0.0000000000000002 ***
## X60                -977985  <0.0000000000000002 ***
## X61               40144175  <0.0000000000000002 ***
## X62                -243179  <0.0000000000000002 ***
## X63               -9715390  <0.0000000000000002 ***
## X64               11908976  <0.0000000000000002 ***
## X65              -13007327  <0.0000000000000002 ***
## X66                5565032  <0.0000000000000002 ***
## X67              -82812341  <0.0000000000000002 ***
## X68               24067476  <0.0000000000000002 ***
## X69              -26286709  <0.0000000000000002 ***
## X70              -60461818  <0.0000000000000002 ***
## X71               20798640  <0.0000000000000002 ***
## X72                4326372  <0.0000000000000002 ***
## X73              -43121709  <0.0000000000000002 ***
## X74              -63907466  <0.0000000000000002 ***
## X75              -12422148  <0.0000000000000002 ***
## X76              -21956893  <0.0000000000000002 ***
## X82              -19471927  <0.0000000000000002 ***
## X85               38685532  <0.0000000000000002 ***
## X86              -20242292  <0.0000000000000002 ***
## X87              -23257188  <0.0000000000000002 ***
## X88               22757337  <0.0000000000000002 ***
## X89               66432902  <0.0000000000000002 ***
## X90              -18368461  <0.0000000000000002 ***
## X91              -29029493  <0.0000000000000002 ***
## X93               21604601  <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  1501.9  on 5268  degrees of freedom
## Residual deviance: 11894.4  on 5196  degrees of freedom
## AIC: 12040
##
## Number of Fisher Scoring iterations: 25
```

```
# we can see that all our variables are highly significant, this is not that
# surprising since most of our data is financial/account related. Lets look at
# some correlations

# variable correlations with deposit for our original data
cor(train_og)[1,]
```

```
##        bankrupt               X6               X7               X8               X9
##   1.00000000000  -0.00007453889  -0.00859526287  -0.00905603766  -0.01953743085
##             X10              X11              X12              X13              X14
```

```
## -0.00808300696 -0.00894448007 -0.02103260822 -0.07601010012 -0.02214680734
##            X15            X18            X19            X20            X21
## -0.11590293570 -0.17066675112 -0.22970143937 -0.06865124674 -0.00410877305
##            X24            X25            X26            X27            X28
##  0.00040277150 -0.01542160582 -0.03961266194 -0.03959341737 -0.01032684566
##            X29            X30            X31            X32            X33
## -0.04631284864  0.07432027893 -0.01644192193 -0.08173993269 -0.00251570132
##            X34            X35            X36            X37            X39
##  0.02851076149 -0.00830286100  0.01677766684  0.24621563363 -0.00831038112
##            X40            X44            X45            X46            X47
##  0.17969279455  0.07261501572 -0.06791619403 -0.00376523494 -0.00704420745
##            X48            X49            X50            X51            X52
## -0.01165696982  0.06444932724  0.01707013802  0.07545624030 -0.10047018078
##            X53            X54            X55            X56            X57
## -0.00686167084 -0.19806952513 -0.07828541138 -0.05397970860 -0.09246088276
##            X58            X59            X60            X61            X62
## -0.00355547355  0.04576263927  0.18871935813 -0.07669897197 -0.00287457809
##            X63            X64            X65            X66            X67
## -0.00175106921 -0.02398596102 -0.15599397455  0.16177220370 -0.01747418867
##            X68            X69            X70            X71            X72
## -0.22144477813 -0.00698721071  0.15430324129  0.01553398983  0.01477656233
##            X73            X74            X75            X76            X82
## -0.00313044000 -0.02269523483  0.00096476077  0.07128956421 -0.11485546262
##            X85            X86            X87            X88            X89
##  0.14077739128 -0.32748227827  0.02048078633 -0.00184035707 -0.10895366954
##            X90            X91            X93
## -0.19318297847  0.17344579137 -0.00873311758
```

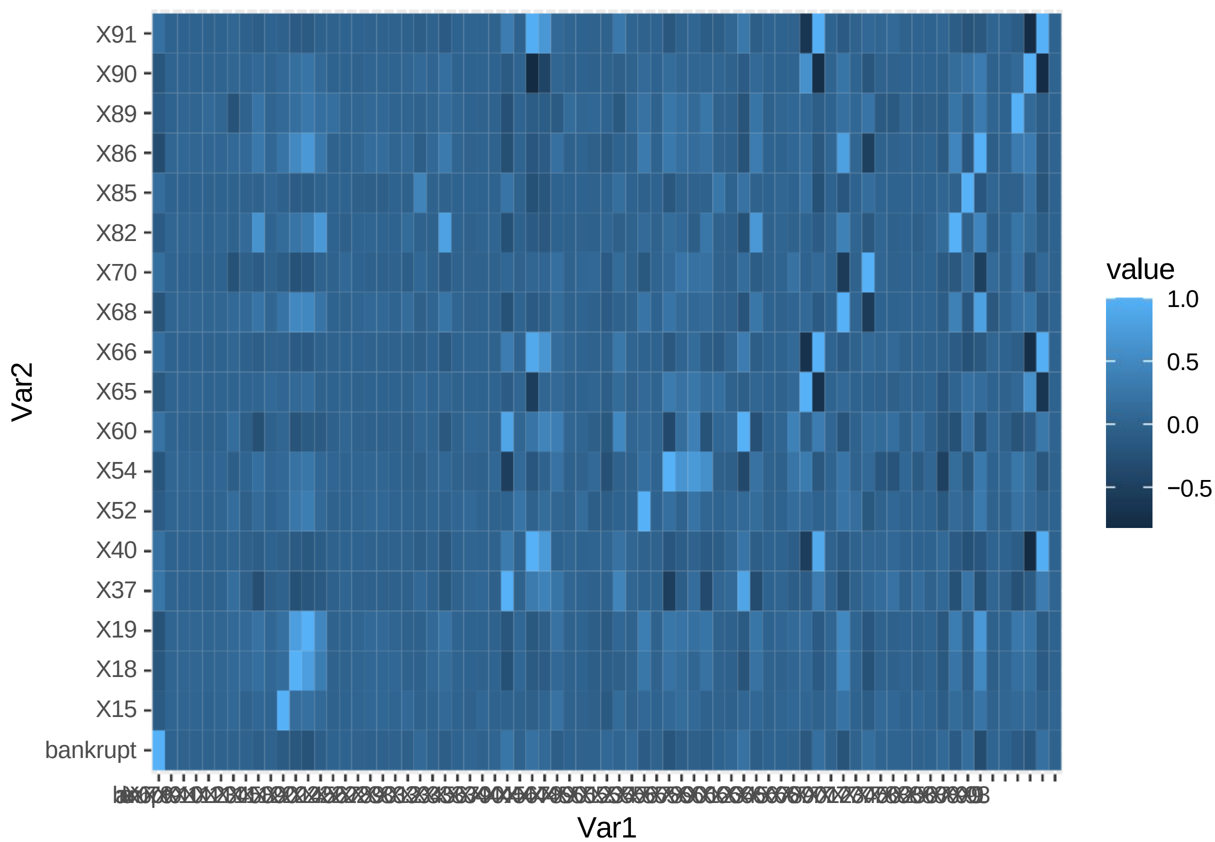
```
# looking at the the variables correlated at 10%
cor(train_og)[1,abs(cor(train_og)[1,])>0.1]
```

```
##   bankrupt        X15        X18        X19        X37        X40        X52
##  1.0000000 -0.1159029 -0.1706668 -0.2297014  0.2462156  0.1796928 -0.1004702
##        X54        X60        X65        X66        X68        X70        X82
## -0.1980695  0.1887194 -0.1559940  0.1617722 -0.2214448  0.1543032 -0.1148555
##        X85        X86        X89        X90        X91
##  0.1407774 -0.3274823 -0.1089537 -0.1931830  0.1734458
```

```
# heatmap
cormat <- round(cor(train_og),2)[,abs(cor(train_og)[1,])>0.1]
melted_cormat <- melt(cormat)
head(melted_cormat)
```

```
##       Var1     Var2 value
## 1 bankrupt bankrupt  1.00
## 2       X6 bankrupt  0.00
## 3       X7 bankrupt -0.01
## 4       X8 bankrupt -0.01
## 5       X9 bankrupt -0.02
## 6      X10 bankrupt -0.01
```

```r
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```




```r
# variable correlations with deposit for our bootstrap data
cor(train_boot)[1,]
```

```
##      bankrupt           X6           X7           X8           X9          X10
##   1.000000000  0.001269780 -0.014748590 -0.018572125 -0.032999701 -0.014278401
##           X11          X12          X13          X14          X15          X18
##  -0.030802183 -0.025822878 -0.168364959 -0.048692056 -0.225424492 -0.320522574
##           X19          X20          X21          X24          X25          X26
##  -0.437360527 -0.139624128 -0.006901642  0.002476036 -0.020831916 -0.043735480
##           X27          X28          X29          X30          X31          X32
##  -0.043137051 -0.012868486 -0.096389280  0.082855226 -0.030870279 -0.167379526
##           X33          X34          X35          X36          X37          X39
##  -0.006901642  0.022963415 -0.032722234  0.048076638  0.445076438 -0.011738453
##           X40          X44          X45          X46          X47          X48
##   0.211220586  0.133906319 -0.126361692 -0.009622971 -0.011547053 -0.019794676
##           X49          X50          X51          X52          X53          X54
##   0.130575635  0.027971782  0.058570741 -0.204513416 -0.017522941 -0.370352196
##           X55          X56          X57          X58          X59          X60
##  -0.157169733 -0.097343532 -0.189795499 -0.169431354  0.080571901  0.369217121
##           X61          X62          X63          X64          X65          X66
##  -0.159277479  0.012108297  0.007678266 -0.023290180 -0.167138928  0.186662988
##           X67          X68          X69          X70          X71          X72
```

```
## -0.033516710 -0.422099219 -0.260502138  0.283345302  0.020842594  0.009377460
##          X73          X74          X75          X76          X82          X85
## -0.014860074 -0.057185299 -0.008900479  0.111714515 -0.247827095  0.143673943
##          X86          X87          X88          X89          X90          X91
## -0.501051859  0.037906053 -0.008434189 -0.196022271 -0.196609362  0.191929935
##          X93
## -0.040979441
```

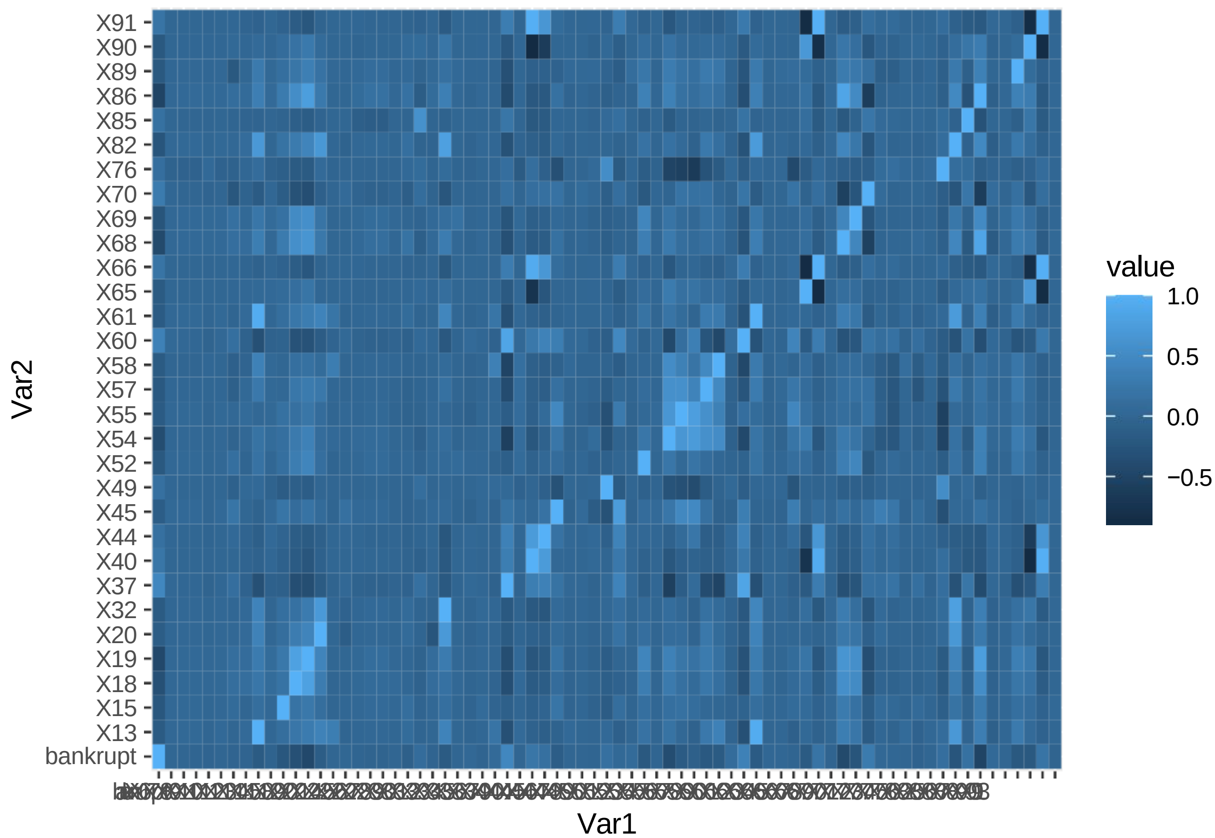
```
# looking at the the variables correlated at 10%
cor(train_boot)[1,abs(cor(train_boot)[1,])>0.1]
```

```
##   bankrupt        X13        X15        X18        X19        X20        X32
##  1.0000000 -0.1683650 -0.2254245 -0.3205226 -0.4373605 -0.1396241 -0.1673795
##        X37        X40        X44        X45        X49        X52        X54
##  0.4450764  0.2112206  0.1339063 -0.1263617  0.1305756 -0.2045134 -0.3703522
##        X55        X57        X58        X60        X61        X65        X66
## -0.1571697 -0.1897955 -0.1694314  0.3692171 -0.1592775 -0.1671389  0.1866630
##        X68        X69        X70        X76        X82        X85        X86
## -0.4220992 -0.2605021  0.2833453  0.1117145 -0.2478271  0.1436739 -0.5010519
##        X89        X90        X91
## -0.1960223 -0.1966094  0.1919299
```

```
# heatmap
cormat1 <- round(cor(train_boot),2)[,abs(cor(train_boot)[1,])>0.1]
melted_cormat1 <- melt(cormat1)
head(melted_cormat1)
```

```
##       Var1     Var2 value
## 1 bankrupt bankrupt  1.00
## 2       X6 bankrupt  0.00
## 3       X7 bankrupt -0.01
## 4       X8 bankrupt -0.02
## 5       X9 bankrupt -0.03
## 6      X10 bankrupt -0.01
```

```
ggplot(data = melted_cormat1, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```




```
# variable correlations with deposit for our adas data
cor(train_adas)[1,]
```

```
##      bankrupt            X6            X7            X8            X9
##  1.0000000000  0.0001337597 -0.0266761346 -0.0260248922 -0.0591843779
##           X10           X11           X12           X13           X14
## -0.0244059690 -0.0277445105 -0.0892203619 -0.2565259865 -0.0858258286
##           X15           X18           X19           X20           X21
## -0.3722347901 -0.5240936782 -0.6059632367 -0.2371465250 -0.0162144854
##           X24           X25           X26           X27           X28
##  0.0004678217 -0.0606238577 -0.1168736839 -0.1171019097 -0.0387250396
##           X29           X30           X31           X32           X33
## -0.1529343306  0.0443583900 -0.0652131961 -0.1657137965 -0.0099284967
##           X34           X35           X36           X37           X39
##  0.0310383860 -0.0345936967  0.0326812695  0.5871421185 -0.0253004691
##           X40           X44           X45           X46           X47
##  0.1867017742  0.1105480089 -0.2006302899 -0.0218274455 -0.0277919855
##           X48           X49           X50           X51           X52
## -0.0673366054  0.1327373234  0.0661134419  0.0504894102 -0.3397749652
##           X53           X54           X55           X56           X57
## -0.0270723120 -0.5066096060 -0.2152955532 -0.1456141015 -0.3130036771
##           X58           X59           X60           X61           X62
## -0.0140314286  0.0559302513  0.4929489981 -0.2455596375 -0.0041206130
##           X63           X64           X65           X66           X67
## -0.0292786635 -0.0557531127 -0.1486698405  0.1588117980 -0.0688140422
##           X68           X69           X70           X71           X72
```

```
## -0.5243143065 -0.0275030368  0.3105872814  0.0312979905  0.0258742879
##           X73           X74           X75           X76           X82
## -0.0096233022 -0.0826320429  0.0037244488  0.1712961917 -0.3085143185
##           X85           X86           X87           X88           X89
##  0.1494133189 -0.5700673798  0.0306836340 -0.0057688397 -0.3350124767
##           X90           X91           X93
## -0.1719110436  0.1693506083 -0.0195169333
```

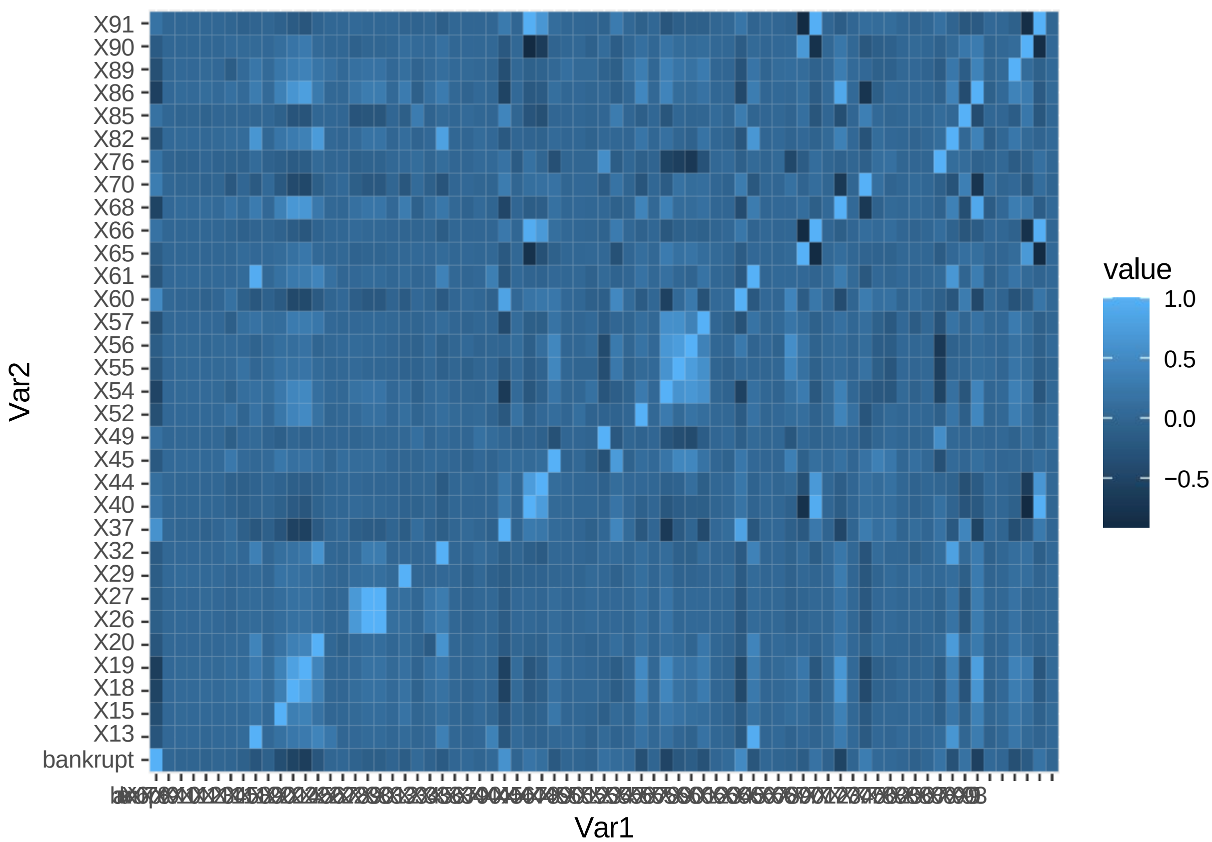
```
# looking at the the variables correlated at 10%
cor(train_adas)[1,abs(cor(train_adas)[1,])>0.1]
```

```
##   bankrupt        X13        X15        X18        X19        X20        X26
##  1.0000000 -0.2565260 -0.3722348 -0.5240937 -0.6059632 -0.2371465 -0.1168737
##        X27        X29        X32        X37        X40        X44        X45
## -0.1171019 -0.1529343 -0.1657138  0.5871421  0.1867018  0.1105480 -0.2006303
##        X49        X52        X54        X55        X56        X57        X60
##  0.1327373 -0.3397750 -0.5066096 -0.2152956 -0.1456141 -0.3130037  0.4929490
##        X61        X65        X66        X68        X70        X76        X82
## -0.2455596 -0.1486698  0.1588118 -0.5243143  0.3105873  0.1712962 -0.3085143
##        X85        X86        X89        X90        X91
##  0.1494133 -0.5700674 -0.3350125 -0.1719110  0.1693506
```

```
# heatmap
cormat2 <- round(cor(train_adas),2)[,abs(cor(train_adas)[1,])>0.1]
melted_cormat2 <- melt(cormat2)
head(melted_cormat2)
```

```
##       Var1     Var2 value
## 1 bankrupt bankrupt  1.00
## 2       X6 bankrupt  0.00
## 3       X7 bankrupt -0.03
## 4       X8 bankrupt -0.03
## 5       X9 bankrupt -0.06
## 6      X10 bankrupt -0.02
```

```
ggplot(data = melted_cormat2, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```




```
# we are mainly concerned with the variables which are correlated to bankrupt
# at 0.1 and more, thus we only plot heatmaps for those variables, to make sure
# our model doesnt not have any multi-colinearity. We can see that there is some
# correlation, bit its not significant enough for us to worry about for now.
# thus based on the above correlations we will use the common variables that are
# correlated to bankrupt for all the 3 datasets, so we can have comparable
# results. We have 18 variables that are common in the 3 training datasets.
#  'X15',  'X18','X19','X37','X40','X52',  'X54','X60','X65','X66','X68','X70','X82','X85','X86','X89','X9

train_og <- train_og[,c(1,11,12,13,29,31,40,42,48,53,54,56,58,65,66,67,70,71,72)]
train_boot <- train_boot[,c(1,11,12,13,29,31,40,42,48,53,54,56,58,65,66,67,70,71,72)]
train_adas  <-  train_adas[,c(1,11,12,13,29,31,40,42,48,53,54,56,58,65,66,67,70,71,72)]

# since we have removed these columns for our training data we will match out test set as well.
dat.test <- dat.test[,c(1,11,12,13,29,31,40,42,48,53,54,56,58,65,66,67,70,71,72)]



# Part 2
# LOGISTIC REGRESSION

# part 2a
# on our original train data
glm_base <- glm(bankrupt ~ ., data=train_og, family='binomial')
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_base)
```

```
##
## Call:
## glm(formula = bankrupt ~ ., family = "binomial", data = train_og)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
##   -8.49     0.00     0.00     0.00     8.49
##
## Coefficients:
##                       Estimate   Std. Error    z value
## (Intercept)  22293475726326296    156320854  142613574
## X15          -3294058247162912      6937270 -474834949
## X18          -5137897431815598     47490858 -108187083
## X19          -5955934136660202     59829356  -99548691
## X37           2934067722166466     40654152   72171415
## X40           4898560537117784    248513369   19711457
## X52           1232179730472399     30628200   40230236
## X54           2313306120747400     22972669  100698186
## X60           1567632463229763     42684610   36725941
## X65         -21480204190973108    142568659 -150665681
## X66         -23834031343701252    315985035  -75427722
## X68           3230897396364499     64665002   49963617
## X70          -5133109221427403     48852581 -105073450
## X82           -523395175375647     18461454  -28350701
## X85           1148891185114548     29888395   38439374
## X86          -5545165800535981     54217060 -102277138
## X89           1681189375362367     71993150   23352074
## X90          -3136378855956487    118763290  -26408656
## X91           6409121791671571    402603737   15919181
##                          Pr(>|z|)
## (Intercept) <0.0000000000000002 ***
## X15         <0.0000000000000002 ***
## X18         <0.0000000000000002 ***
## X19         <0.0000000000000002 ***
## X37         <0.0000000000000002 ***
## X40         <0.0000000000000002 ***
## X52         <0.0000000000000002 ***
## X54         <0.0000000000000002 ***
## X60         <0.0000000000000002 ***
## X65         <0.0000000000000002 ***
## X66         <0.0000000000000002 ***
## X68         <0.0000000000000002 ***
## X70         <0.0000000000000002 ***
## X82         <0.0000000000000002 ***
## X85         <0.0000000000000002 ***
## X86         <0.0000000000000002 ***
## X89         <0.0000000000000002 ***
## X90         <0.0000000000000002 ***
## X91         <0.0000000000000002 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  1501.9  on 5268  degrees of freedom
## Residual deviance: 13192.0  on 5250  degrees of freedom
## AIC: 13230
##
## Number of Fisher Scoring iterations: 25
```

```
# prediction using our train set for error
yhat.train.lr <- predict(glm_base, train_og,
                         type = "response")

# classifying the deposit variable as 0/1 or no/yes with a threshold of 0.5
lr_yhat.train.class <- ifelse(yhat.train.lr > 0.5, 1, 0)

# confusion matrix of our train set
tab.lr_train <- table(train_og$bankrupt,
                      lr_yhat.train.class,
                      dnn = c("Actual","Predicted"))
tab.lr_train
```

```
##       Predicted
## Actual0    1
##      0 5020   79
##      1  104   66
```

```
# overall train error
lr_train <- mean(lr_yhat.train.class != train_og$bankrupt)
lr_train
```

```
## [1] 0.03473145
```

```
# class 1 test error
lr_train_class1 <- tab.lr_train[2,1]/5269
lr_train_class1
```

```
## [1] 0.01973809
```

```
# class 0 test error
lr_train_class0 <- tab.lr_train[1,2]/5269
lr_train_class0
```

```
## [1] 0.01499336
```

```
# prediction using our test set for error
yhat.test.lr <- predict(glm_base, dat.test,
                        type = "response")
lr_yhat.test.class <- ifelse(yhat.test.lr > 0.5, 1, 0)
```

```
tab.lr_test <- table(dat.test$bankrupt,
                     lr_yhat.test.class,
                     dnn = c("Actual","Predicted"))
tab.lr_test
```

```
##          Predicted
## Actual      0    1
##        0 1476   24
##        1   36   14
```

```
# overall test error
lr_test <- mean(lr_yhat.test.class != dat.test$bankrupt)
lr_test
```

```
## [1] 0.03870968
```

```
# class 1 test error
lr_test_class1 <- tab.lr_test[2,1]/1550
lr_test_class1
```

```
## [1] 0.02322581
```

```
# class 0 test error
lr_test_class0 <- tab.lr_test[1,2]/1550
lr_test_class0
```

```
## [1] 0.01548387
```

```
# part 2c
# model on our train_boot data
glm_boot <- glm(bankrupt ~ ., data=train_boot, family='binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_boot)
```

```
##
## Call:
## glm(formula = bankrupt ~ ., family = "binomial", data = train_boot)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5761  -0.3503  -0.1807  -0.0726   3.0443
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 110.9584    32.2186   3.444 0.000573 ***
## X15          -0.2585     0.5855  -0.442 0.658845
## X18          -7.5216     5.5964  -1.344 0.178948
```

```
## X19          -52.7236    7.1637  -7.360 0.000000000000184 ***
## X37           18.7193    3.7202   5.032 0.000000486048885 ***
## X40           88.4478   20.1373   4.392 0.000011218647184 ***
## X52            1.7424    3.2816   0.531          0.595450
## X54            2.3515    2.1816   1.078          0.281098
## X60            7.9126    4.4397   1.782          0.074709 .
## X65          -73.9842   15.0535  -4.915 0.000000888918430 ***
## X66         -107.0638   53.3659  -2.006          0.044833 *
## X68           -2.6829    4.0238  -0.667          0.504927
## X70           -4.2330    2.4200  -1.749          0.080256 .
## X82           -1.9469    1.3200  -1.475          0.140228
## X85           17.3475  387.1364   0.045          0.964259
## X86           -2.1190    4.1769  -0.507          0.611937
## X89           17.2668    5.8173   2.968          0.002996 **
## X90          -40.1819   21.1498  -1.900          0.057449 .
## X91          -74.9727   37.5379  -1.997          0.045798 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2870.8  on 3499  degrees of freedom
## Residual deviance: 1542.8  on 3481  degrees of freedom
## AIC: 1580.8
##
## Number of Fisher Scoring iterations: 15
```

```
# prediction using our train set for error
yhat.train.lr_boot <- predict(glm_boot, train_boot,
                              type = "response")


# classifying the deposit variable as 0/1 or no/yes with a threshold of 0.5
lr_yhat.train.boot.class <- ifelse(yhat.train.lr_boot > 0.5, 1, 0)

# confusion matrix of our train set
tab.lr_train_boot <- table(train_boot$bankrupt,
                           lr_yhat.train.boot.class,
                           dnn = c("Actual","Predicted"))
tab.lr_train_boot
```

```
##        Predicted
## Actual    0    1
##       0 2910   90
##       1  229  271
```

```
# overall train error
lr_train_boot <- mean(lr_yhat.train.boot.class != train_boot$bankrupt)
lr_train_boot
```

```
## [1] 0.09114286
```

```r
# class 1 test error
lr_train_boot_class1 <- tab.lr_train_boot[2,1]/3500
lr_train_boot_class1
```

```
## [1] 0.06542857
```

```r
# class 0 test error
lr_train_boot_class0 <- tab.lr_train_boot[1,2]/3500
lr_train_boot_class0
```

```
## [1] 0.02571429
```

```r
# prediction using our test set for error
yhat.test.lr_boot <- predict(glm_boot, dat.test,
                             type = "response")
lr_yhat.test.boot.class <- ifelse(yhat.test.lr_boot > 0.5, 1, 0)

tab.lr_test.boot <- table(dat.test$bankrupt,
                          lr_yhat.test.boot.class,
                          dnn = c("Actual","Predicted"))
tab.lr_test.boot
```

```
##        Predicted
## Actual    0    1
##      0 1457   43
##      1   30   20
```

```r
# overall test error
lr_test_boot <- mean(lr_yhat.test.boot.class != dat.test$bankrupt)
lr_test_boot
```

```
## [1] 0.04709677
```

```r
# class 1 test error
lr_test_boot_class1 <- tab.lr_test.boot[2,1]/1550
lr_test_boot_class1
```

```
## [1] 0.01935484
```

```r
# class 0 test error
lr_test_boot_class0 <- tab.lr_test.boot[1,2]/1550
lr_test_boot_class0
```

```
## [1] 0.02774194
```

```r
# part 2d
# on our train_adas data
glm_adas <- glm(bankrupt ~ ., data=train_adas, family='binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(glm_adas)
```

```
##
## Call:
## glm(formula = bankrupt ~ ., family = "binomial", data = train_adas)
##
## Deviance Residuals:
## Min          1Q Median        3Q      Max
## -8.4904  -0.3372   0.0000   0.3738   2.5836
##
## Coefficients:
##               Estimate Std. Error z value             Pr(>|z|)
## (Intercept)   167.0427    19.5686   8.536 < 0.0000000000000002 ***
## X15             0.5194     0.2743   1.893              0.05829 .
## X18            -4.1505     2.9049  -1.429              0.15306
## X19           -61.3436     4.4711 -13.720 < 0.0000000000000002 ***
## X37            16.9427     2.2456   7.545   0.0000000000000453 ***
## X40           184.8304    12.5566  14.720 < 0.0000000000000002 ***
## X52            -2.2390     2.0327  -1.101              0.27069
## X54             9.7153     1.5556   6.245   0.0000000004231045 ***
## X60            16.5366     2.7843   5.939   0.0000000028618956 ***
## X65          -144.4739    15.0711  -9.586 < 0.0000000000000002 ***
## X66          -189.8534    38.7263  -4.902   0.0000009465525807 ***
## X68            -0.3500     2.8739  -0.122              0.90306
## X70            -8.0730     1.3768  -5.864   0.0000000045275152 ***
## X82            -1.8411     0.7157  -2.572              0.01010 *
## X85            10.6461     2.6403   4.032   0.0000552727583670 ***
## X86           -14.1992     2.9749  -4.773   0.0000018146094624 ***
## X89             6.7979     2.8602   2.377              0.01747 *
## X90           -35.2725    11.9687  -2.947              0.00321 **
## X91           -95.7023    34.5691  -2.768              0.00563 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 14210.6  on 10250  degrees of freedom
## Residual deviance:  5744.3  on 10232  degrees of freedom
## AIC: 5782.3
##
## Number of Fisher Scoring iterations: 10
```

```r
# prediction using our train set for error
yhat.train.lr_adas <- predict(glm_adas, train_adas,
                              type = "response")


# classifying the deposit variable as 0/1 or no/yes with a threshold of 0.5
lr_yhat.train.adas.class <- ifelse(yhat.train.lr_adas > 0.5, 1, 0)

# confusion matrix of our train set
tab.lr_train_adas <- table(train_adas$bankrupt,
                           lr_yhat.train.adas.class,
```

```r
                                  dnn  =  c("Actual","Predicted"))
tab.lr_train_adas
```

```
##         Predicted
## Actual     0    1
##        0 4524  575
##        1  533 4619
```

```r
# overall train error
lr_train_adas <- mean(lr_yhat.train.adas.class != train_adas$bankrupt)
lr_train_adas
```

```
## [1] 0.108087
```

```r
# class 1 test error
lr_train_adas_class1 <- tab.lr_train_adas[2,1]/10251
lr_train_adas_class1
```

```
## [1] 0.05199493
```

```r
# class 0 test error
lr_train_adas_class0 <- tab.lr_train_adas[1,2]/10251
lr_train_adas_class0
```

```
## [1] 0.05609209
```

```r
# prediction using our test set for error
yhat.test.lr_adas <- predict(glm_adas, dat.test,
                                    type = "response")
lr_yhat.test.adas.class <- ifelse(yhat.test.lr_adas > 0.5, 1, 0)

tab.lr_test.adas <- table(dat.test$bankrupt,
                                 lr_yhat.test.adas.class,
                                 dnn  =  c("Actual","Predicted"))
tab.lr_test.adas
```

```
##         Predicted
## Actual     0    1
##        0 1335  165
##        1   10   40
```

```r
# overall test error
lr_test_adas <- mean(lr_yhat.test.adas.class != dat.test$bankrupt)
lr_test_adas
```

```
## [1] 0.1129032
```

```r
# class 1 test error
lr_test_adas_class1 <- tab.lr_test.adas[2,1]/1550
lr_test_adas_class1
```

```
## [1] 0.006451613
```

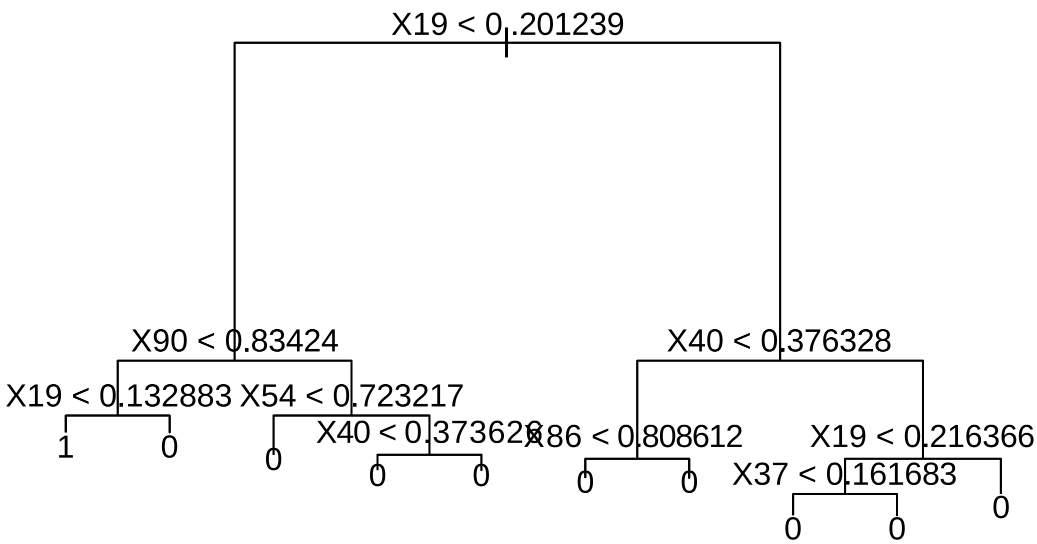
```r
# class 0 test error
lr_test_adas_class0 <- tab.lr_test.adas[1,2]/1550
lr_test_adas_class0
```

```
## [1] 0.1064516
```

```r
# After performing the predictions on the test dataset using the models built using the 3  train sets,
# the logistic regression performs the best[considering only performance]




# Part 3
# DECISION TREES

# part 3a
# on our original train data
# converting the Y-variable to a "factor" since classification tree models assume
# that the Y variable is qualitative
new.train_og <- train_og
new.train_og[,1] <- as.factor(new.train_og[,1])

# preparing our test data
new.dat.test <- dat.test
new.dat.test[,1] <- as.factor(new.dat.test[,1])

# building our first tree
tree_og <- tree(bankrupt ~., data = new.train_og)
summary(tree_og)
```

```
##
## Classification tree:
## tree(formula = bankrupt ~ ., data = new.train_og)
## Variables actually used in tree construction:
## [1] "X19" "X90" "X54" "X40" "X86" "X37"
## Number of terminal nodes:  10
## Residual mean deviance:  0.1634 = 859.4 / 5259
## Misclassification error rate: 0.03056 = 161 / 5269
```

```r
# plotting our tree
plot(tree_og)
text(tree_og, pretty = 0)
```




```r
# metadata
tree_og
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 5269 1502.00 0 ( 0.967736 0.032264 )
##    2) X19 < 0.201239 555   563.70 0 ( 0.794595 0.205405 )
##      4) X90 < 0.83424 186   249.20 0 ( 0.607527 0.392473 )
##        8) X19 < 0.132883 9      0.00 1 ( 0.000000 1.000000 ) *
##        9) X19 > 0.132883 177   231.60 0 ( 0.638418 0.361582 ) *
##      5) X90 > 0.83424 369   257.40 0 ( 0.888889 0.111111 )
##       10) X54 < 0.723217 49    66.92 0 ( 0.571429 0.428571 ) *
##       11) X54 > 0.723217 320   149.60 0 ( 0.937500 0.062500 )
##         22) X40 < 0.373626 138    11.85 0 ( 0.992754 0.007246 ) *
##         23) X40 > 0.373626 182   121.80 0 ( 0.895604 0.104396 ) *
##    3) X19 > 0.201239 4714   607.80 0 ( 0.988120 0.011880 )
##      6) X40 < 0.376328 3712   126.40 0 ( 0.997575 0.002425 )
##       12) X86 < 0.808612 1233   106.50 0 ( 0.992701 0.007299 ) *
##       13) X86 > 0.808612 2479     0.00 0 ( 1.000000 0.000000 ) *
##      7) X40 > 0.376328 1002   379.40 0 ( 0.953094 0.046906 )
##       14) X19 < 0.216366 306   213.50 0 ( 0.888889 0.111111 )
##         28) X37 < 0.161683 145    36.61 0 ( 0.972414 0.027586 ) *
##         29) X37 > 0.161683 161   154.80 0 ( 0.813665 0.186335 ) *
##       15) X19 > 0.216366 696   129.20 0 ( 0.981322 0.018678 ) *
```

```r
# making predictions on our train data using tree1
tree.pred.train_og <- predict(tree_og, new.train_og, type = "class")

# confusion matrix
tab_tree_og <- table(new.train_og$bankrupt, tree.pred.train_og,
                     dnn = c("Actual", "Predicted"))
tab_tree_og
```

```
##       Predicted
## Actual    0    1
##      0 5099    0
##      1  161    9
```

```r
# overall error
tree_og_err <- mean(new.train_og$bankrupt != tree.pred.train_og)
tree_og_err
```

```
## [1] 0.03055608
```

```r
# class 1 train error
tree_og_class1 <- tab_tree_og[2,1]/5269
tree_og_class1
```

```
## [1] 0.03055608
```

```r
# class 0 train error
tree_og_class0 <- tab_tree_og[1,2]/5269
tree_og_class0
```

```
## [1] 0
```

```r
# making predictions using our test data
tree.pred.test <- predict(tree_og, new.dat.test, type = "class")

# confusion matrix
tab_tree_og.test <- table(new.dat.test$bankrupt, tree.pred.test,
                          dnn = c("Actual", "Predicted"))
tab_tree_og.test
```

```
##       Predicted
## Actual    0    1
##      0 1495    5
##      1   50    0
```

```r
# overall error
tree_og_err.test <- mean(new.dat.test$bankrupt != tree.pred.test)
tree_og_err.test
```

```
## [1] 0.03548387
```

```
# class 1 test error
tree_og_class1.test <- tab_tree_og.test[2,1]/1550
tree_og_class1.test
```

```
## [1] 0.03225806
```

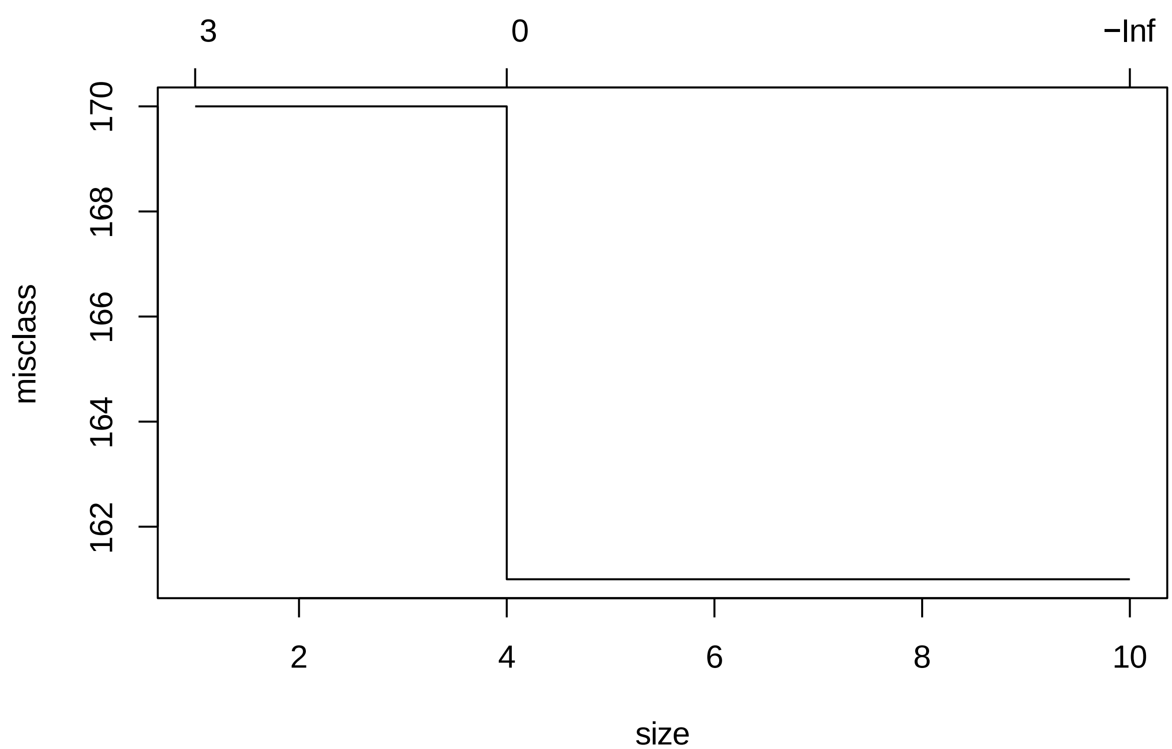
```
# class 0 test error
tree_og_class0.test <- tab_tree_og.test[1,2]/1550
tree_og_class0.test
```

```
## [1] 0.003225806
```

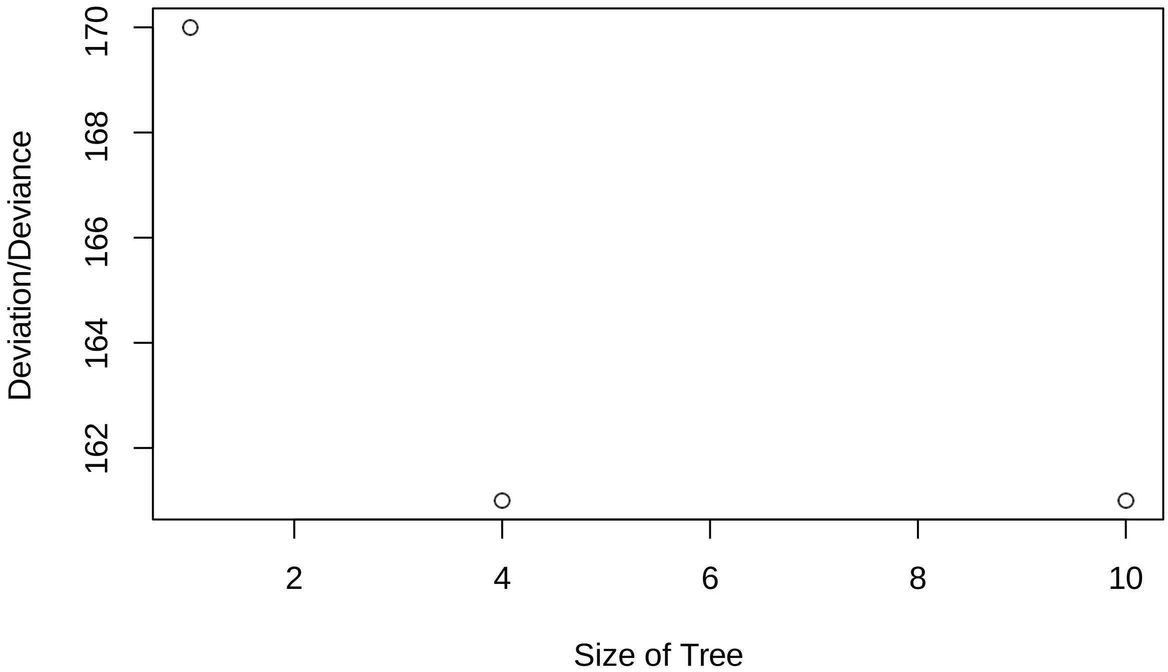
```
# pruning our decision tree
prune_og <- prune.misclass(tree_og)
names(prune_og)
```

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

```
#  Plotting the results of the prune to identify the right tree size
plot(prune_og)
```



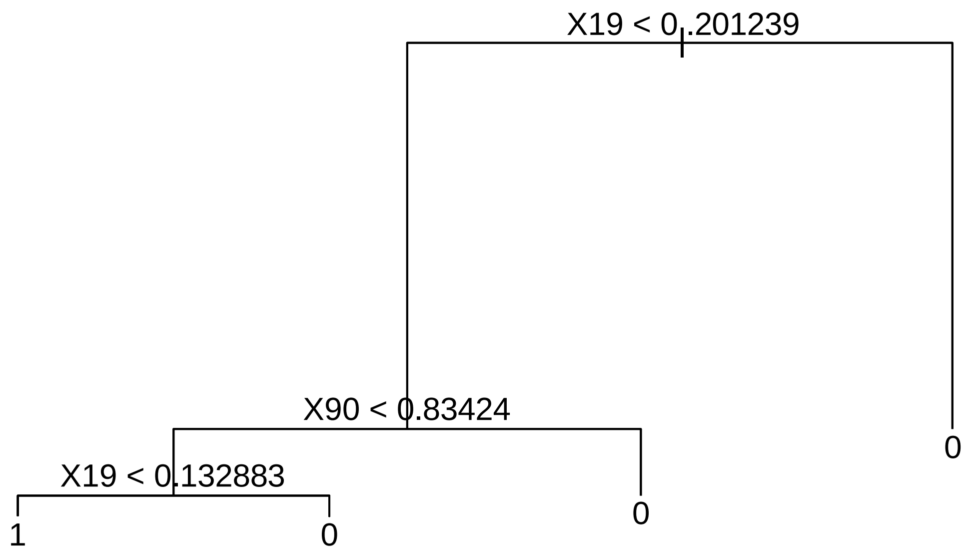
```r
plot(prune_og$size, prune_og$dev, xlab = "Size of Tree",
     ylab = "Deviation/Deviance")
```




```r
# pruning our tree1
prune.tree_og <- prune.misclass(tree_og, best = 4)
summary(prune.tree_og)
```

```
##
## Classification tree:
## snip.tree(tree = tree_og, nodes = c(3L, 5L))
## Variables actually used in tree construction:
## [1] "X19" "X90"
## Number of terminal nodes:  4
## Residual mean deviance:  0.2083 = 1097 / 5265
## Misclassification error rate: 0.03056 = 161 / 5269
```

```r
# plotting our pruned tree
plot(prune.tree_og)
text(prune.tree_og, pretty = 0)
```




```r
# making predictions on our test data using prune.tree1
pt1.pred <- predict(prune.tree_og, new.train_og, type = "class")

# confusion matrix
tab_pt1 <-table(new.train_og$bankrupt, pt1.pred,
                dnn = c("Actual", "Predicted"))

tab_pt1
```

```
##         Predicted
## Actual     0    1
##        0 5099    0
##        1  161    9
```

```r
# overall error
pt1_err <- mean(new.train_og$bankrupt != pt1.pred)
pt1_err
```

```
## [1] 0.03055608
```

```r
# class 1 test error
pt1_class1 <- tab_pt1[2,1]/5269
pt1_class1
```

```
## [1] 0.03055608
```

```r
# class 0 test error
pt1_class0 <- tab_pt1[1,2]/5269
pt1_class0
```

```
## [1] 0
```

```r
# prediction using our test set for error
pt1.test.pred <- predict(prune.tree_og, new.dat.test, type = "class")

# confusion matrix
tab_pt1.test <-table(new.dat.test$bankrupt, pt1.test.pred,
                     dnn = c("Actual", "Predicted"))

tab_pt1.test
```

```
##         Predicted
## Actual      0    1
##        0 1495    5
##        1   50    0
```

```r
# overall error
pt1_err.test <- mean(new.dat.test$bankrupt != pt1.test.pred)
pt1_err.test
```

```
## [1] 0.03548387
```

```r
# class 1 test error
pt1_class1.test <- tab_pt1.test[2,1]/1550
pt1_class1.test
```

```
## [1] 0.03225806
```

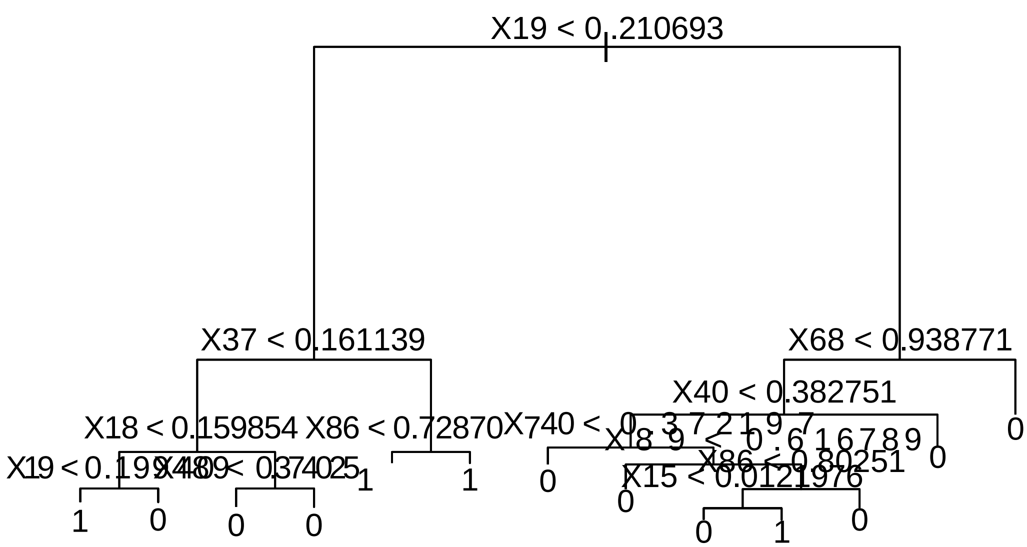
```r
# class 0 test error
pt1_class0.test <- tab_pt1.test[1,2]/1550
pt1_class0.test
```

```
## [1] 0.003225806
```

```r
# part 3b
# model on our train_boot data
# converting the Y-variable to a "factor" since classification tree models assume
# that the Y variable is qualitative
new.train_boot <- train_boot
new.train_boot[,1] <- as.factor(new.train_boot[,1])

# building our first tree
tree_boot <- tree(bankrupt ~., data = new.train_boot)
summary(tree_boot)
```

```
##
## Classification tree:
## tree(formula = bankrupt ~ ., data = new.train_boot)
## Variables actually used in tree construction:
## [1] "X19" "X37" "X18" "X40" "X86" "X68" "X89" "X15"
## Number of terminal nodes:  13
## Residual mean deviance:  0.3419 = 1192 / 3487
## Misclassification error rate: 0.08057 = 282 / 3500
```

```
# plotting our tree
plot(tree_boot)
text(tree_boot, pretty = 0)
```




```
# metadata
tree_boot
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 3500 2871.00 0 ( 0.85714 0.14286 )
##     2) X19 < 0.210693 959 1312.00 0 ( 0.56726 0.43274 )
##       4) X37 < 0.161139 502  504.10 0 ( 0.79880 0.20120 )
##         8) X18 < 0.159854 155  214.40 0 ( 0.52903 0.47097 )
##          16) X19 < 0.199489 117  157.60 1 ( 0.40171 0.59829 ) *
##          17) X19 > 0.199489 38   20.99 0 ( 0.92105 0.07895 ) *
```

```
##          9) X18 > 0.159854 347   194.60 0 ( 0.91931 0.08069 )
##           18) X40 < 0.374025 192     0.00 0 ( 1.00000 0.00000 ) *
##           19) X40 > 0.374025 155   146.40 0 ( 0.81935 0.18065 ) *
##        5) X37 > 0.161139 457   568.00 1 ( 0.31291 0.68709 )
##         10) X86 < 0.728707 135   109.70 1 ( 0.14074 0.85926 ) *
##         11) X86 > 0.728707 322   429.20 1 ( 0.38509 0.61491 ) *
##     3) X19 > 0.210693 2541   744.70 0 ( 0.96655 0.03345 )
##       6) X68 < 0.938771 1121   601.90 0 ( 0.92417 0.07583 )
##         12) X40 < 0.382751 1006   367.60 0 ( 0.95527 0.04473 )
##           24) X40 < 0.372197 380     0.00 0 ( 1.00000 0.00000 ) *
##           25) X40 > 0.372197 626   323.60 0 ( 0.92812 0.07188 )
##             50) X89 < 0.616789 567   179.60 0 ( 0.96296 0.03704 ) *
##             51) X89 > 0.616789 59    79.73 0 ( 0.59322 0.40678 )
##              102) X86 < 0.80251 30    30.02 1 ( 0.20000 0.80000 )
##                204) X15 < 0.0121976 6     0.00 0 ( 1.00000 0.00000 ) *
##                205) X15 > 0.0121976 24     0.00 1 ( 0.00000 1.00000 ) *
##              103) X86 > 0.80251 29     0.00 0 ( 1.00000 0.00000 ) *
##         13) X40 > 0.382751 115   148.60 0 ( 0.65217 0.34783 ) *
##       7) X68 > 0.938771 1420     0.00 0 ( 1.00000 0.00000 ) *
```

```
# making predictions on our test data using tree1
tree.pred.train_boot <- predict(tree_boot, new.train_boot, type = "class")

# confusion matrix
tab_tree_boot <- table(new.train_boot$bankrupt, tree.pred.train_boot,
                       dnn = c("Actual", "Predicted"))
tab_tree_boot
```

```
##         Predicted
## Actual     0    1
##      0  2810  190
##      1    92  408
```

```
# overall error
tree_boot_err <- mean(new.train_boot$bankrupt != tree.pred.train_boot)
tree_boot_err
```

```
## [1] 0.08057143
```

```
# class 1 test error
tree_boot_class1 <- tab_tree_boot[2,1]/3500
tree_boot_class1
```

```
## [1] 0.02628571
```

```
# class 0 test error
tree_boot_class0 <- tab_tree_boot[1,2]/3500
tree_boot_class0
```

```
## [1] 0.05428571
```

```r
# predictions on our test data
tree.pred.test_boot <- predict(tree_boot, new.dat.test, type = "class")

# confusion matrix
tab_tree_boot.test <- table(new.dat.test$bankrupt, tree.pred.test_boot,
                            dnn = c("Actual", "Predicted"))
tab_tree_boot.test
```

```
##        Predicted
## Actual    0    1
##       0 1419   81
##       1   19   31
```

```r
# overall error
tree_boot_err.test <- mean(new.dat.test$bankrupt != tree.pred.test_boot)
tree_boot_err.test
```

```
## [1] 0.06451613
```

```r
# class 1 test error
tree_boot_class1.test <- tab_tree_boot.test[2,1]/1550
tree_boot_class1.test
```

```
## [1] 0.01225806
```

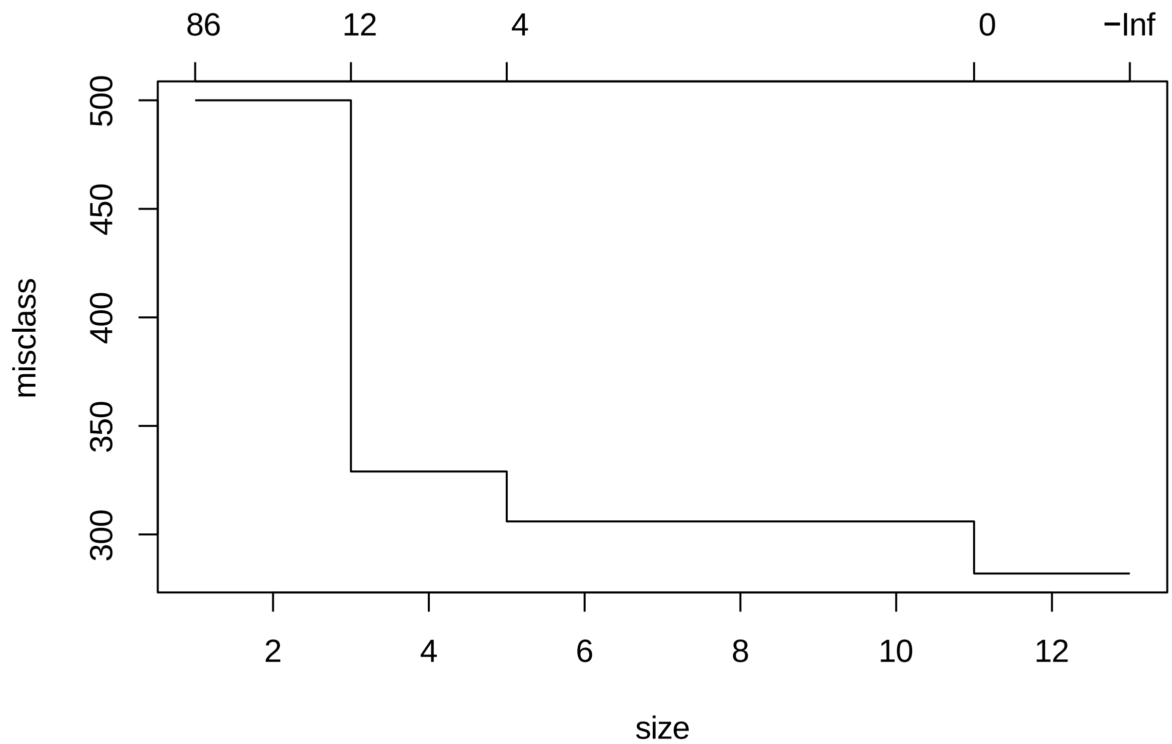
```r
# class 0 test error
tree_boot_class0.test <- tab_tree_boot.test[1,2]/1550
tree_boot_class0.test
```

```
## [1] 0.05225806
```

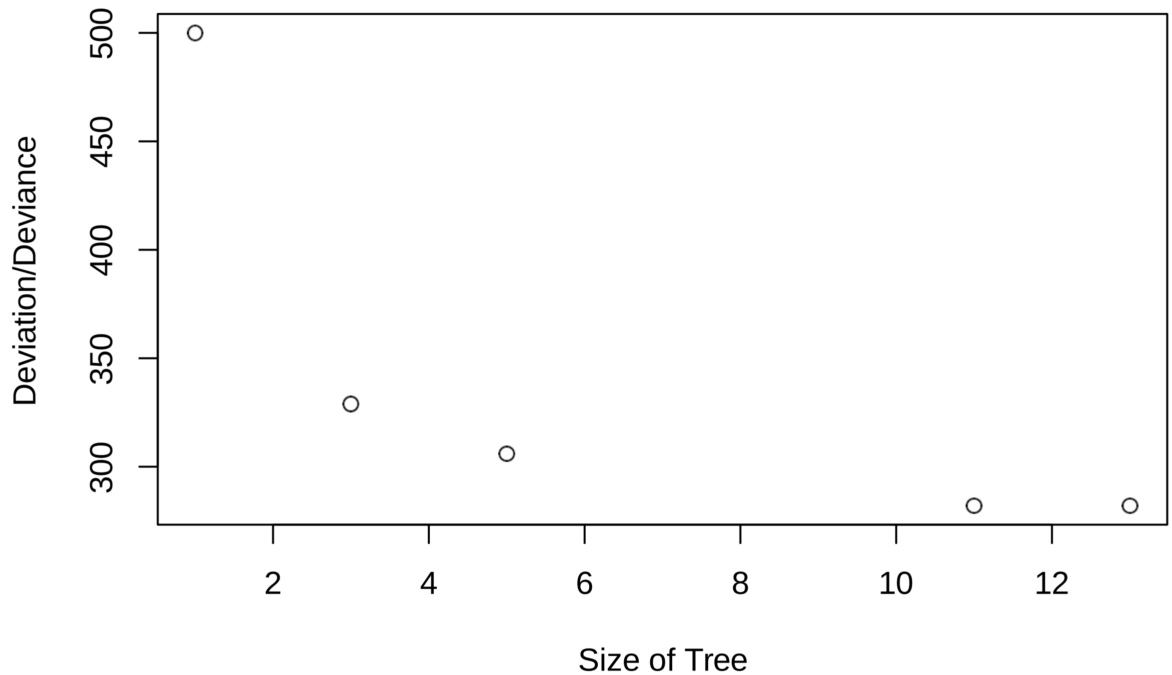
```r
# pruning our decision tree
prune_boot <- prune.misclass(tree_boot)
names(prune_boot)
```

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

```r
#  Plotting the results of the prune to identify the right tree size
plot(prune_boot)
```



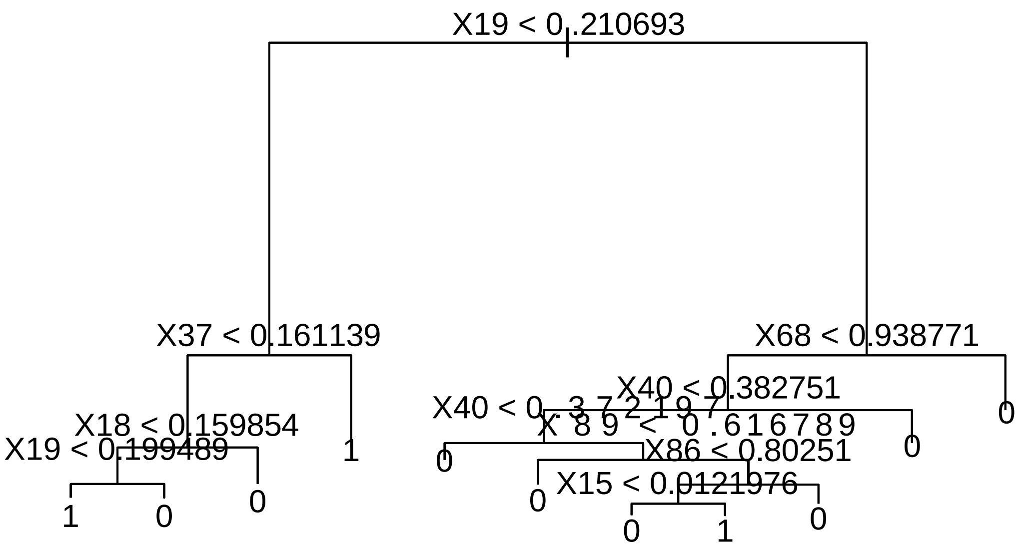

```
plot(prune_boot$size, prune_boot$dev, xlab = "Size of Tree",
     ylab = "Deviation/Deviance")
```




```
# pruning our tree1
prune.tree_boot <- prune.misclass(tree_boot, best = 11)
summary(prune.tree_boot)
```

```
##
## Classification tree:
## snip.tree(tree = tree_boot, nodes = c(5L, 9L))
## Variables actually used in tree construction:
## [1] "X19" "X37" "X18" "X68" "X40" "X89" "X86" "X15"
## Number of terminal nodes:  11
## Residual mean deviance:  0.3639 = 1269 / 3489
## Misclassification error rate: 0.08057 = 282 / 3500
```

```
# plotting our pruned tree
plot(prune.tree_boot)
text(prune.tree_boot, pretty = 0)
```




```
# making predictions on our test data using prune.tree1
pt2.pred <- predict(prune.tree_boot, new.train_boot, type = "class")

# confusion matrix
tab_pt2 <-table(new.train_boot$bankrupt, pt2.pred,
                dnn = c("Actual", "Predicted"))

tab_pt2
```

```
##         Predicted
## Actual     0    1
##      0  2810  190
##      1    92  408
```

```
# overall error
pt2_err <- mean(new.train_boot$bankrupt != pt2.pred)
pt2_err
```

```
## [1] 0.08057143
```

```
# class 1 test error
pt2_class1 <- tab_pt2[2,1]/3500
pt2_class1
```

```
## [1] 0.02628571
```

```r
# class 0 test error
pt2_class0 <- tab_pt2[1,2]/3500
pt2_class0
```

```
## [1] 0.05428571
```

```r
# predictions using our test data
pt2.test.pred <- predict(prune.tree_boot, new.dat.test, type = "class")

# confusion matrix
tab_pt2.test <-table(new.dat.test$bankrupt, pt2.test.pred,
                     dnn = c("Actual", "Predicted"))

tab_pt2.test
```

```
##         Predicted
## Actual      0    1
##       0  1419   81
##       1    19   31
```

```r
# overall error
pt2_err.test <- mean(new.dat.test$bankrupt != pt2.test.pred)
pt2_err.test
```

```
## [1] 0.06451613
```

```r
# class 1 test error
pt2_class1.test <- tab_pt2.test[2,1]/1550
pt2_class1.test
```

```
## [1] 0.01225806
```

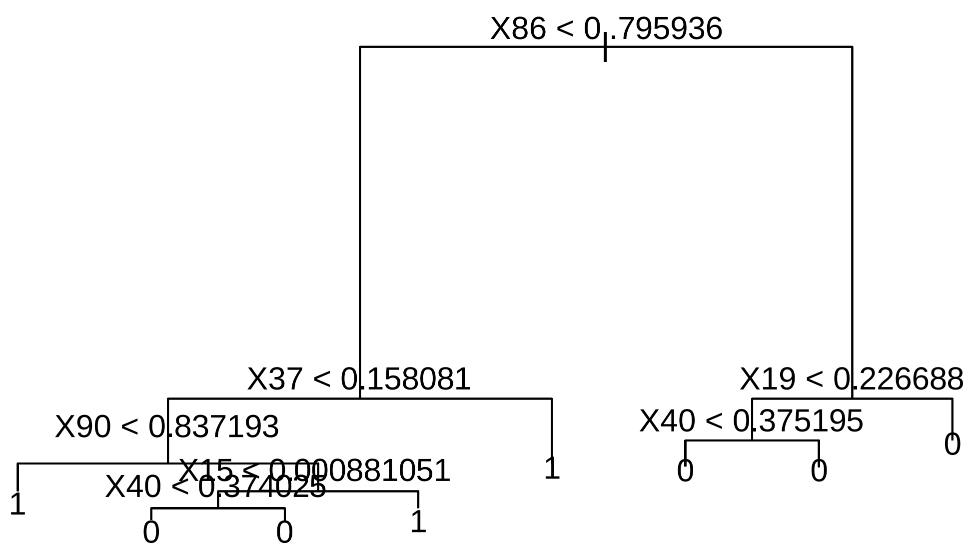
```r
# class 0 test error
pt2_class0.test <- tab_pt2.test[1,2]/1550
pt2_class0.test
```

```
## [1] 0.05225806
```

```r
# part 3c
# model on our train_adas data
# converting the Y-variable to a "factor" since classification tree models assume
# that the Y variable is qualitative
new.train_adas <- train_adas
new.train_adas[,1] <- as.factor(new.train_adas[,1])

# building our first tree
tree_adas <- tree(bankrupt ~., data = new.train_adas)
summary(tree_adas)
```

```
##
## Classification tree:
## tree(formula = bankrupt ~ ., data = new.train_adas)
## Variables actually used in tree construction:
## [1] "X86" "X37" "X90" "X15" "X40" "X19"
## Number of terminal nodes:  8
## Residual mean deviance:  0.5243 = 5370 / 10240
## Misclassification error rate: 0.1057 = 1084 / 10251
```

```
# plotting our tree
plot(tree_adas)
text(tree_adas, pretty = 0)
```




```
# metadata
tree_adas
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 10251 14210.0 1 ( 0.497415 0.502585 )
##    2) X86 < 0.795936 5759  5509.0 1 ( 0.184581 0.815419 )
##      4) X37 < 0.158081 1808  2479.0 1 ( 0.438606 0.561394 )
##        8) X90 < 0.837193 904   894.1 1 ( 0.195796 0.804204 ) *
##        9) X90 > 0.837193 904  1131.0 0 ( 0.681416 0.318584 )
##         18) X15 < 0.000881051 797   854.0 0 ( 0.772898 0.227102 )
```

```
##                36) X40 < 0.374025 419    148.5 0 ( 0.957041 0.042959 ) *
##                37) X40 > 0.374025 378    516.8 0 ( 0.568783 0.431217 ) *
##              19) X15 > 0.000881051 107      0.0 1 ( 0.000000 1.000000 ) *
##          5) X37 > 0.158081 3951   1970.0 1 ( 0.068337 0.931663 ) *
##        3) X86 > 0.795936 4492   2950.0 0 ( 0.898486 0.101514 )
##          6) X19 < 0.226688 2123   2196.0 0 ( 0.787565 0.212435 )
##           12) X40 < 0.375195 1180    491.8 0 ( 0.946610 0.053390 ) *
##           13) X40 > 0.375195 943   1278.0 0 ( 0.588547 0.411453 ) *
##          7) X19 > 0.226688 2369     71.6 0 ( 0.997889 0.002111 ) *
```

```
# making predictions on our test data using tree1
tree.pred.train_adas <- predict(tree_adas, new.train_adas, type = "class")

# confusion matrix
tab_tree_adas <- table(new.train_adas$bankrupt, tree.pred.train_adas,
                       dnn = c("Actual", "Predicted"))
tab_tree_adas
```

```
##         Predicted
## Actual      0    1
##        0 4652  447
##        1  637 4515
```

```
# overall error
tree_adas_err <- mean(new.train_adas$bankrupt != tree.pred.train_adas)
tree_adas_err
```

```
## [1] 0.1057458
```

```
# class 1 test error
tree_adas_class1 <- tab_tree_adas[2,1]/10251
tree_adas_class1
```

```
## [1] 0.06214028
```

```
# class 0 test error
tree_adas_class0 <- tab_tree_adas[1,2]/10251
tree_adas_class0
```

```
## [1] 0.0436055
```

```
# predictions using our test data
tree.pred.test_adas <- predict(tree_adas, new.dat.test, type = "class")

# confusion matrix
tab_tree_adas.test <- table(new.dat.test$bankrupt, tree.pred.test_adas,
                            dnn = c("Actual", "Predicted"))
tab_tree_adas.test
```

```
##         Predicted
## Actual      0    1
##        0 1381  119
##        1   17   33
```

```r
# overall error
tree_adas_err.test <- mean(new.dat.test$bankrupt != tree.pred.test_adas)
tree_adas_err.test
```

```
## [1] 0.08774194
```

```r
# class 1 test error
tree_adas_class1.test <- tab_tree_adas.test[2,1]/1550
tree_adas_class1.test
```

```
## [1] 0.01096774
```

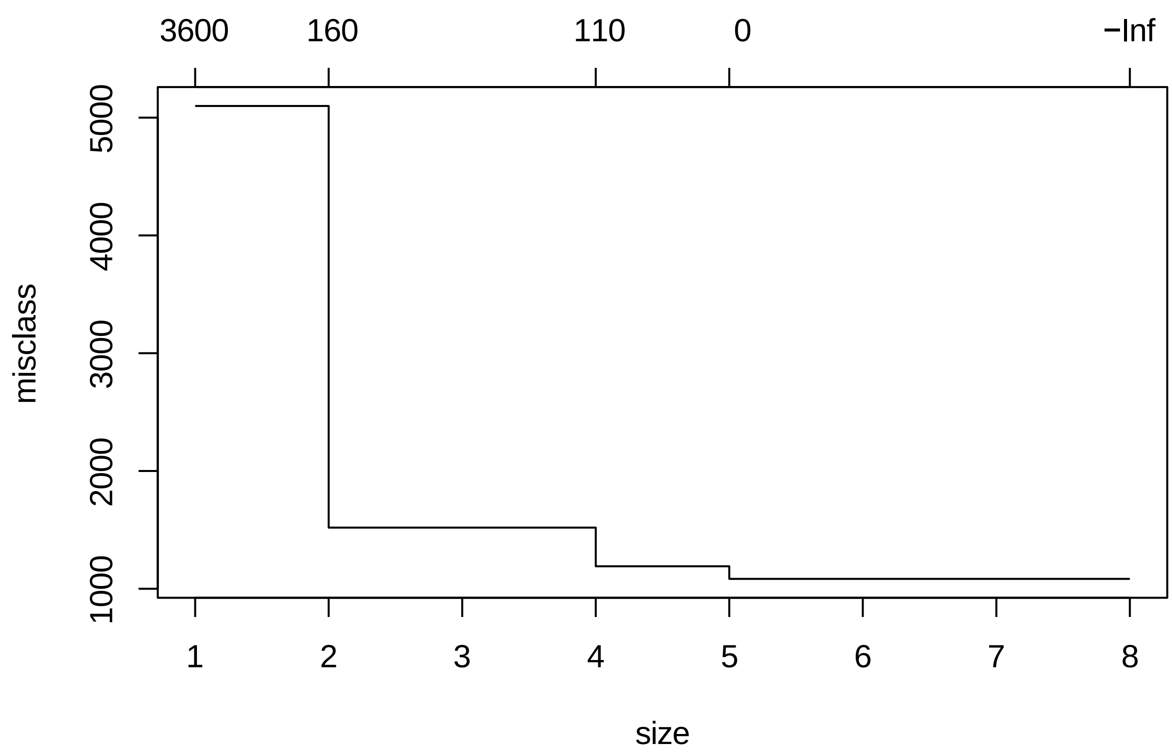
```r
# class 0 test error
tree_adas_class0.test <- tab_tree_adas.test[1,2]/1550
tree_adas_class0.test
```

```
## [1] 0.07677419
```

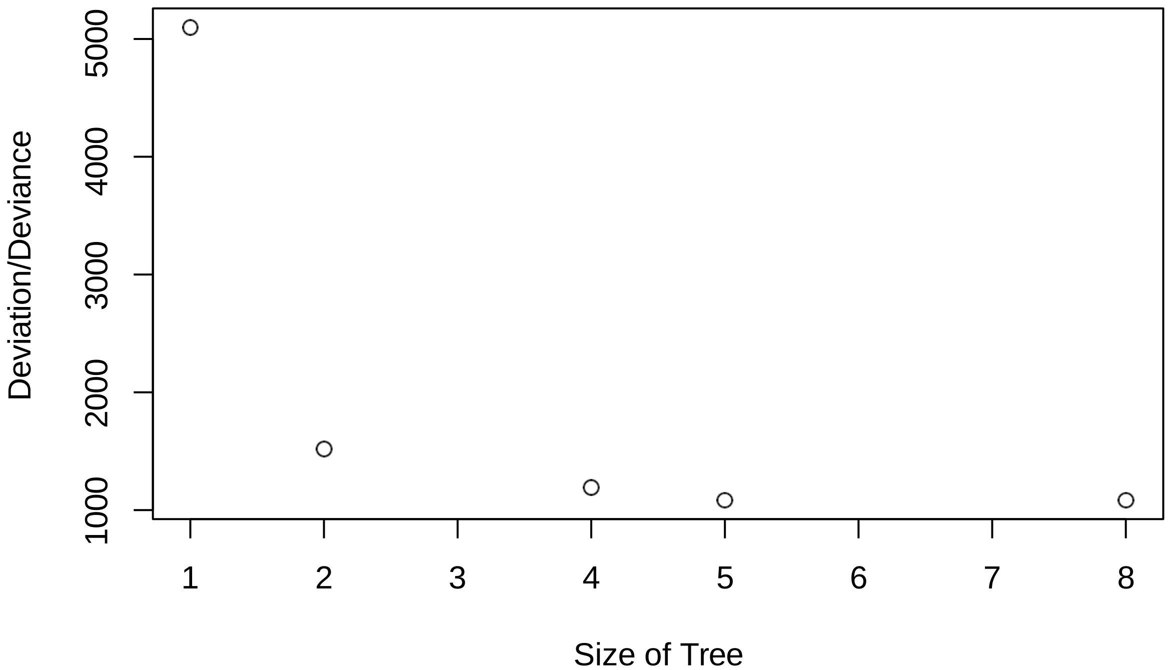
```r
# pruning our decision tree
prune_adas <- prune.misclass(tree_adas)
names(prune_boot)
```

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

```r
#  Plotting the results of the prune to identify the right tree size
plot(prune_adas)
```



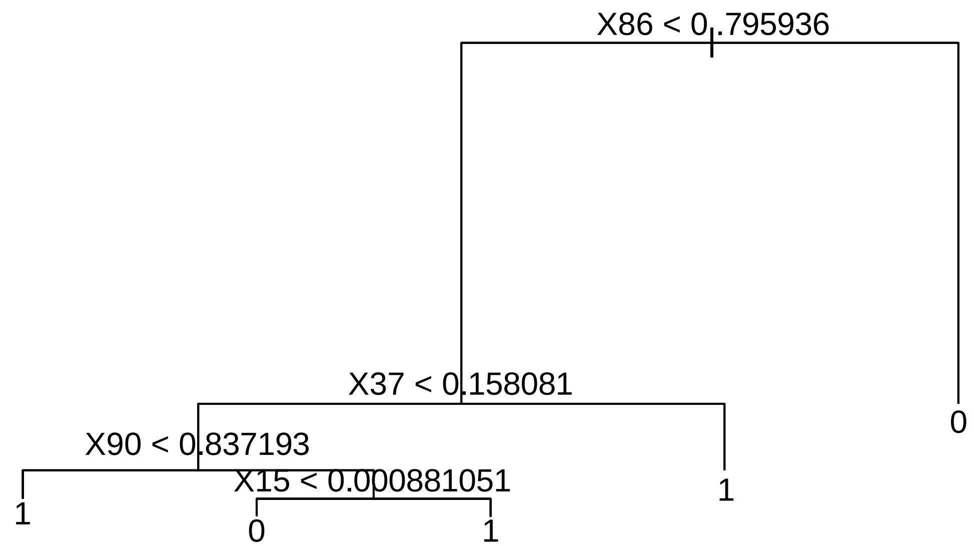
```r
plot(prune_adas$size, prune_adas$dev, xlab = "Size of Tree",
     ylab = "Deviation/Deviance")
```




```r
# pruning our tree1
prune.tree_adas <- prune.misclass(tree_adas, best = 5)
summary(prune.tree_adas)
```

```
##
## Classification tree:
## snip.tree(tree = tree_adas, nodes = c(3L, 18L))
## Variables actually used in tree construction:
## [1] "X86" "X37" "X90" "X15"
## Number of terminal nodes:  5
## Residual mean deviance:  0.6508 = 6668 / 10250
## Misclassification error rate: 0.1057 = 1084 / 10251
```

```r
# plotting our pruned tree
plot(prune.tree_adas)
text(prune.tree_adas, pretty = 0)
```




```r
# making predictions on our test data using prune.tree1
pt3.pred <- predict(prune.tree_adas, new.train_adas, type = "class")

# confusion matrix
tab_pt3 <-table(new.train_adas$bankrupt, pt3.pred,
                dnn = c("Actual", "Predicted"))

tab_pt3
```

```
##          Predicted
## Actual     0    1
##        0 4652  447
##        1  637 4515
```

```r
# overall error
pt3_err <- mean(new.train_adas$bankrupt != pt3.pred)
pt3_err
```

```
## [1] 0.1057458
```

```r
# class 1 test error
pt3_class1 <- tab_pt3[2,1]/10251
pt3_class1
```

```
## [1] 0.06214028
```

```r
# class 0 test error
pt3_class0 <- tab_pt3[1,2]/10251
pt3_class0
```

```
## [1] 0.0436055
```

```r
# prediction using our test set for error
pt3.pred.test <- predict(prune.tree_adas, new.dat.test, type = "class")

# confusion matrix
tab_pt3.test <-table(new.dat.test$bankrupt, pt3.pred.test,
                     dnn = c("Actual", "Predicted"))

tab_pt3.test
```

```
##          Predicted
## Actual      0    1
##        0 1381  119
##        1   17   33
```

```r
# overall error
pt3_err.test <- mean(new.dat.test$bankrupt != pt3.pred.test)
pt3_err.test
```

```
## [1] 0.08774194
```

```r
# class 1 test error
pt3_class1.test <- tab_pt3.test[2,1]/1550
pt3_class1.test
```

```
## [1] 0.01096774
```

```r
# class 0 test error
pt3_class0.test <- tab_pt3.test[1,2]/1550
pt3_class0.test
```

```
## [1] 0.07677419
```

```r
# Part 4
# table to compare our train values
train.error <- data.frame(matrix(0,3,10))
names(train.error) <- c("Error" , "Logistic Reg Train","Logistic Reg Boot", "Logistic Reg ADAS",
                        "Decision Tree", "Pruned Tree", "Decision Tree Boot","Pruned Tree Boot",
                        "Decision Tree Adas", "Pruned Tree ADAS")

train.error[1,] <- c('Overall Error', lr_train, lr_train_boot, lr_train_adas, tree_og_err, pt1_err,
                     tree_boot_err, pt2_err, tree_adas_err, pt3_err)
train.error[2,] <- c('Class 1 Err', lr_train_class1,lr_train_boot_class1, lr_train_adas_class1,
                     tree_og_class1, pt1_class1, tree_boot_class1, pt2_class1, tree_adas_class1,
                     pt3_class1)
```

```
train.error[3,] <- c('Class 0 Err', lr_train_class0, lr_train_boot_class0, lr_train_adas_class0,
                      tree_og_class0, pt1_class0, tree_boot_class0, pt2_class0, tree_adas_class0,
                      pt3_class0)

# lets look at our table
train.error
```

```
##             Error Logistic Reg Train  Logistic Reg Boot  Logistic Reg ADAS
## 1 Overall Error 0.0347314480926172  0.0911428571428571   0.108087015900888
## 2     Class 1 Err 0.0197380907193016  0.0654285714285714  0.0519949273241635
## 3     Class 0 Err 0.0149933573733156  0.0257142857142857  0.0560920885767242
##          Decision Tree         Pruned Tree Decision Tree Boot    Pruned Tree Boot
## 1 0.0305560827481496  0.0305560827481496 0.0805714285714286   0.0805714285714286
## 2 0.0305560827481496  0.0305560827481496 0.0262857142857143   0.0262857142857143
## 3                  0                   0 0.0542857142857143   0.0542857142857143
##    Decision Tree Adas   Pruned Tree ADAS
## 1   0.105745780899424   0.105745780899424
## 2   0.062140278997171   0.062140278997171
## 3 0.0436055019022534  0.0436055019022534
```

```
# table to compare our test values
test.error <- data.frame(matrix(0,3,10))
names(test.error) <- c("Error" , "Logistic Reg","Logistic Reg Boot", "Logistic Reg ADAS",
                        "Decision Tree", "Pruned Tree", "Decision Tree Boot","Pruned Tree Boot",
                        "Decision Tree Adas", "Pruned Tree ADAS")

test.error[1,] <- c('Overall Error', lr_test, lr_test_boot, lr_test_adas, tree_og_err.test, pt1_err.tes
                     tree_boot_err.test, pt2_err.test, tree_adas_err.test, pt3_err.test)
test.error[2,] <- c('Class 1 Err', lr_test_class1, lr_test_boot_class1, lr_test_adas_class1,
                     tree_og_class1.test, pt1_class1.test, tree_boot_class1.test, pt2_class1.test
                     , tree_adas_class1.test,
                     pt3_class1.test)
test.error[3,] <- c('Class 0 Err', lr_test_class0, lr_test_boot_class0, lr_test_adas_class0,
                     tree_og_class0.test, pt1_class0.test, tree_boot_class0.test, pt2_class0.test
                     , tree_adas_class0.test,
                     pt3_class0.test)
test.error
```

```
##             Error        Logistic Reg  Logistic Reg Boot   Logistic Reg ADAS
## 1 Overall Error 0.0387096774193548  0.0470967741935484    0.112903225806452
## 2     Class 1 Err 0.0232258064516129  0.0193548387096774  0.00645161290322581
## 3     Class 0 Err 0.0154838709677419   0.027741935483871    0.106451612903226
##          Decision Tree         Pruned Tree Decision Tree Boot    Pruned Tree Boot
## 1 0.0354838709677419  0.0354838709677419 0.0645161290322581  0.0645161290322581
## 2   0.032258064516129    0.032258064516129   0.012258064516129   0.012258064516129
## 3 0.0032258064516129  0.0032258064516129   0.052258064516129   0.052258064516129
##    Decision Tree Adas   Pruned Tree ADAS
## 1   0.087741935483871   0.087741935483871
## 2 0.0109677419354839  0.0109677419354839
## 3 0.0767741935483871  0.0767741935483871
```

```r
# Part 5
# Prediction of Bankrupty for MediaTek INC. using our best model

max(dat.test$X40)
```

```
## [1] 0.4427657
```

```r
# setting up our data
mediatek <- data.frame(X15 = 0.1262, X18 = 2.73, X19 = 0.7626, X37 = 0.2718,
                       X40 = 0.4427, X52 = 8.211, X54 = 0.2565, X60 = 0.2327,
                       X65 = 0.3523, X65 = 0.3523, X66 = 0.3195, X68 = 0.4985,
                       X70 = 0.7078, X82 = 0.2369, X85 = 0, X86 = 0.1942, X89 = 0.4666,
                       X90 = 0.2666, X91 = 0.3732)

# lets check
mediatek
```

```
##      X15  X18    X19    X37    X40   X52    X54    X60    X65  X65.1    X66
## 1 0.1262 2.73 0.7626 0.2718 0.4427 8.211 0.2565 0.2327 0.3523 0.3523 0.3195
##      X68    X70    X82 X85    X86    X89    X90    X91
## 1 0.4985 0.7078 0.2369   0 0.1942 0.4666 0.2666 0.3732
```

```r
# prediction using our train set for error
yhat.mediatek <- predict(glm_base, mediatek,
                         type = "response")

# classifying the deposit variable as 0/1 or no/yes with a threshold of 0.5
lr_yhat.mediatek.class <- ifelse(yhat.mediatek > 0.5, 1, 0)
lr_yhat.mediatek.class
```

```
## 1
## 1
```

```r
# we tried to predict the bankruptcy of Mediatek Inc. since their share prices
# have been dropping for the past year and eps is very low, However, our model did
# not do very well since it predicted that mediatek will go bankrupt, which is not
# likely since the company is doing financially well as of today. It must be mentioned
# that our model is not 100% accurate. Ths not included in the main presentation.
```