# ISYE6501x - WEEK 4 HW

#### Vivian Peng

Question 9.1 Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components.

Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function prcomp for PCA. (Note that to first scale the data, you can include scale. = TRUE to scale as part of the PCA function. Don’t forget that, to make a prediction for the new city, you’ll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)

Hide

Data\_crime = read.csv("uscrime.txt",sep = "")  
str(Data\_crime)

'data.frame': 47 obs. of 16 variables:  
 $ M : num 15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...  
 $ So : int 1 0 1 0 0 0 1 1 1 0 ...  
 $ Ed : num 9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...  
 $ Po1 : num 5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...  
 $ Po2 : num 5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...  
 $ LF : num 0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...  
 $ M.F : num 95 101.2 96.9 99.4 98.5 ...  
 $ Pop : int 33 13 18 157 18 25 4 50 39 7 ...  
 $ NW : num 30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...  
 $ U1 : num 0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...  
 $ U2 : num 4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...  
 $ Wealth: int 3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...  
 $ Ineq : num 26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...  
 $ Prob : num 0.0846 0.0296 0.0834 0.0158 0.0414 ...  
 $ Time : num 26.2 25.3 24.3 29.9 21.3 ...  
 $ Crime : int 791 1635 578 1969 1234 682 963 1555 856 705 ...

Hide

print(summary(Data\_crime))

M So Ed Po1 Po2   
 Min. :11.9 Min. :0.00 Min. : 8.70 Min. : 4.50 Min. : 4.10   
 1st Qu.:13.0 1st Qu.:0.00 1st Qu.: 9.75 1st Qu.: 6.25 1st Qu.: 5.85   
 Median :13.6 Median :0.00 Median :10.80 Median : 7.80 Median : 7.30   
 Mean :13.9 Mean :0.34 Mean :10.56 Mean : 8.50 Mean : 8.02   
 3rd Qu.:14.6 3rd Qu.:1.00 3rd Qu.:11.45 3rd Qu.:10.45 3rd Qu.: 9.70   
 Max. :17.7 Max. :1.00 Max. :12.20 Max. :16.60 Max. :15.70   
 LF M.F Pop NW U1   
 Min. :0.480 Min. : 93.4 Min. : 3.0 Min. : 0.2 Min. :0.0700   
 1st Qu.:0.530 1st Qu.: 96.4 1st Qu.: 10.0 1st Qu.: 2.4 1st Qu.:0.0805   
 Median :0.560 Median : 97.7 Median : 25.0 Median : 7.6 Median :0.0920   
 Mean :0.561 Mean : 98.3 Mean : 36.6 Mean :10.1 Mean :0.0955   
 3rd Qu.:0.593 3rd Qu.: 99.2 3rd Qu.: 41.5 3rd Qu.:13.2 3rd Qu.:0.1040   
 Max. :0.641 Max. :107.1 Max. :168.0 Max. :42.3 Max. :0.1420   
 U2 Wealth Ineq Prob Time   
 Min. :2.00 Min. :2880 Min. :12.6 Min. :0.0069 Min. :12.2   
 1st Qu.:2.75 1st Qu.:4595 1st Qu.:16.6 1st Qu.:0.0327 1st Qu.:21.6   
 Median :3.40 Median :5370 Median :17.6 Median :0.0421 Median :25.8   
 Mean :3.40 Mean :5254 Mean :19.4 Mean :0.0471 Mean :26.6   
 3rd Qu.:3.85 3rd Qu.:5915 3rd Qu.:22.8 3rd Qu.:0.0544 3rd Qu.:30.5   
 Max. :5.80 Max. :6890 Max. :27.6 Max. :0.1198 Max. :44.0   
 Crime   
 Min. : 342   
 1st Qu.: 658   
 Median : 831   
 Mean : 905   
 3rd Qu.:1058   
 Max. :1993

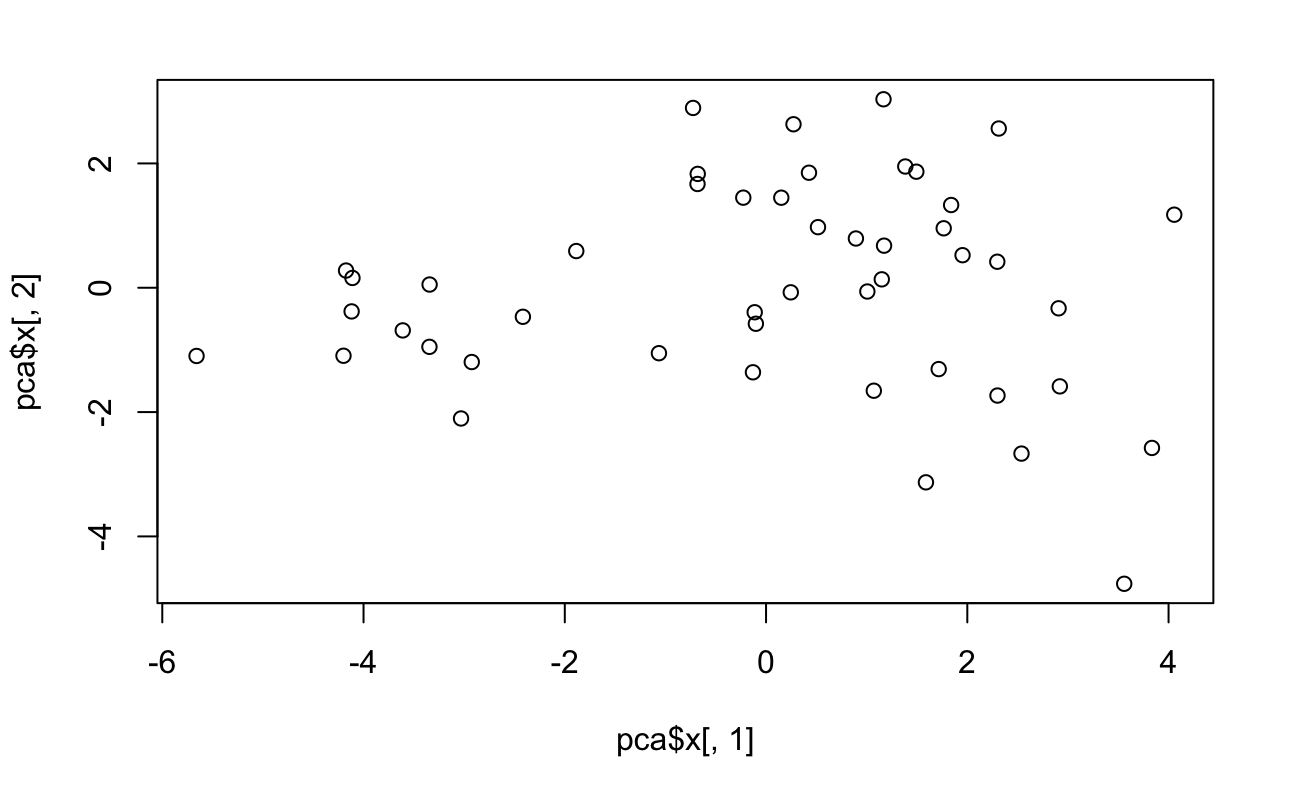
Hide

data <- Data\_crime[,1:15]  
pca <- prcomp(data,scale=TRUE)  
pca

Standard deviations (1, .., p=15):  
 [1] 2.4534 1.6739 1.4160 1.0781 0.9789 0.7438 0.5673 0.5544 0.4849 0.4471 0.4191  
[12] 0.3580 0.2633 0.2418 0.0679  
  
Rotation (n x k) = (15 x 15):  
 PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9  
M -0.3037 0.06280 0.172420 -0.0204 -0.3583 -0.44913 -0.1571 -0.5537 0.1547  
So -0.3309 -0.15837 0.015543 0.2925 -0.1206 -0.10050 0.1965 0.2273 -0.6560  
Ed 0.3396 0.21461 0.067740 0.0797 -0.0244 -0.00857 -0.2394 -0.1464 -0.4433  
Po1 0.3086 -0.26982 0.050646 0.3333 -0.2353 -0.09578 0.0801 0.0461 0.1943  
Po2 0.3110 -0.26396 0.053065 0.3519 -0.2047 -0.11952 0.0952 0.0317 0.1951  
LF 0.1762 0.31943 0.271530 -0.1433 -0.3941 0.50423 -0.1593 0.2551 0.1439  
M.F 0.1164 0.39434 -0.203162 0.0105 -0.5788 -0.07450 0.1555 -0.0551 -0.2438  
Pop 0.1131 -0.46723 0.077021 -0.0321 -0.0832 0.54710 0.0905 -0.5908 -0.2024  
NW -0.2936 -0.22801 0.078816 0.2393 -0.3608 0.05122 -0.3115 0.2043 0.1898  
U1 0.0405 0.00807 -0.659029 -0.1828 -0.1314 0.01739 -0.1735 -0.2021 0.0207  
U2 0.0181 -0.27971 -0.578501 -0.0689 -0.1350 0.04816 -0.0753 0.2437 0.0558  
Wealth 0.3797 -0.07719 0.010065 0.1178 0.0117 -0.15468 -0.1486 0.0863 -0.2320  
Ineq -0.3658 -0.02752 -0.000294 -0.0807 -0.2167 0.27203 0.3748 0.0718 -0.0249  
Prob -0.2589 0.15832 -0.117673 0.4930 0.1656 0.28354 -0.5616 -0.0860 -0.0531  
Time -0.0206 -0.38015 0.223566 -0.5406 -0.1476 -0.14820 -0.4420 0.1951 -0.2355  
 PC10 PC11 PC12 PC13 PC14 PC15  
M -0.0144 0.3945 0.1658 -0.0514 0.0490 0.00514  
So 0.0614 0.2340 -0.0575 -0.2937 -0.2936 0.00844  
Ed 0.5189 -0.1182 0.4779 0.1944 0.0396 -0.02801  
Po1 -0.1432 -0.1304 0.2261 -0.1859 -0.0949 -0.68942  
Po2 -0.0593 -0.1389 0.1909 -0.1345 -0.0826 0.72003  
LF 0.0308 0.3853 0.0271 -0.2774 -0.1539 0.03368  
M.F -0.3532 -0.2803 -0.2393 0.3162 -0.0413 0.00979  
Pop -0.0397 0.0585 -0.1835 0.1265 -0.0533 0.00015  
NW 0.4920 -0.2070 -0.3667 0.2290 0.1323 -0.03708  
U1 0.2277 -0.1786 -0.0931 -0.5904 -0.0234 0.01114  
U2 -0.0475 0.4702 0.2844 0.4329 -0.0399 0.00736  
Wealth -0.1122 0.3196 -0.3217 -0.1408 0.7003 -0.00257  
Ineq -0.0139 -0.1828 0.4376 -0.1218 0.5928 0.01776  
Prob -0.4253 -0.0898 0.1557 -0.0355 0.0476 0.02934  
Time -0.2926 -0.2636 0.1354 -0.0574 -0.0449 0.03768

Hide

#Plot the 1st and 2nd principal components  
plot(pca$x[,1], pca$x[,2])



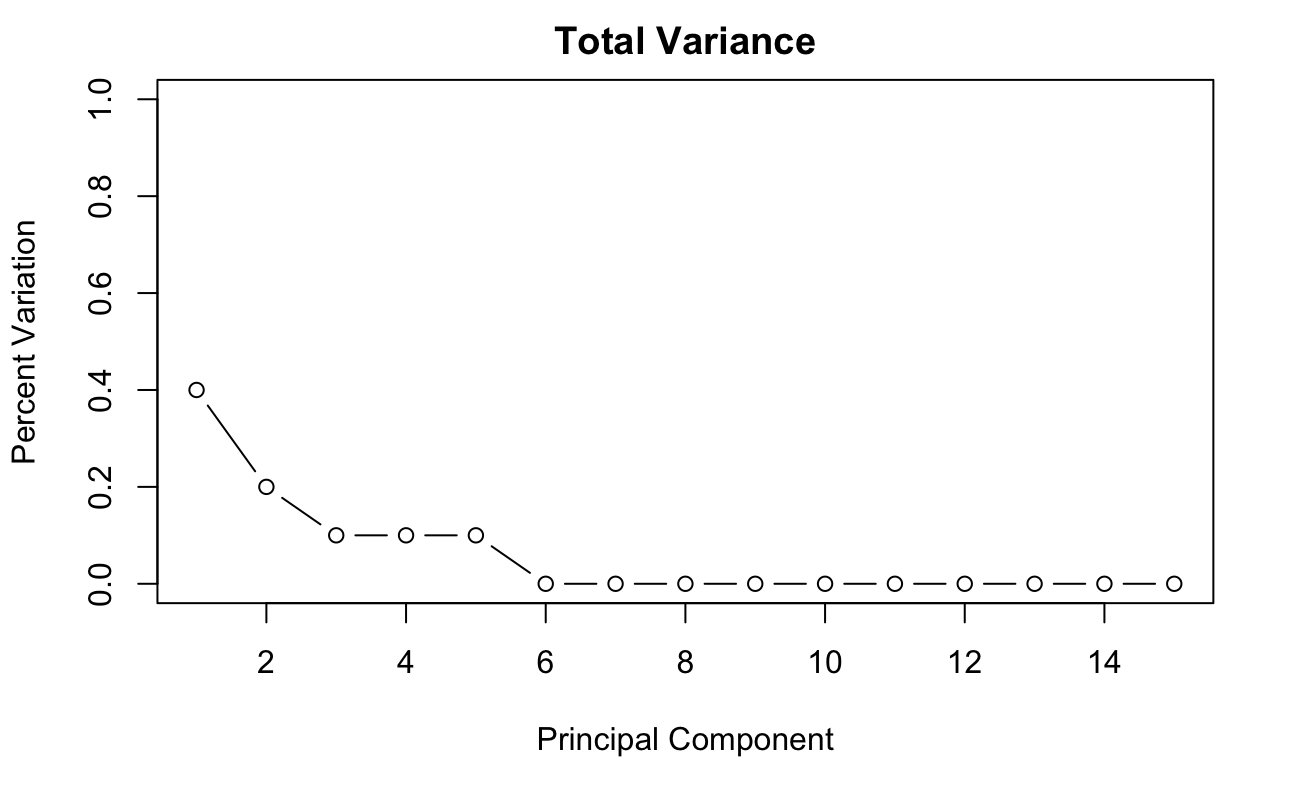
Hide

pca.var <- pca$sdev^2  
pca.var.per <- round(pca.var/sum(pca.var), 1)  
pca.var.per

[1] 0.4 0.2 0.1 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Hide

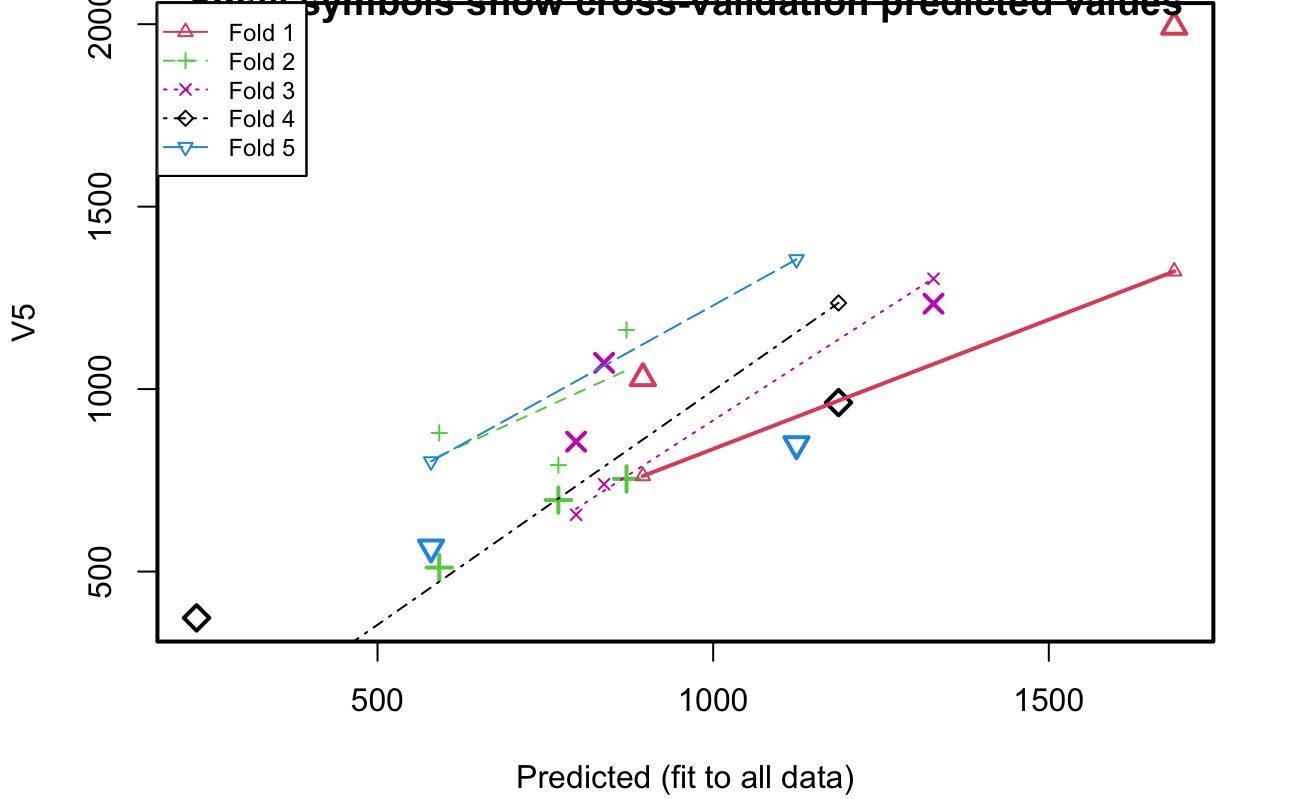
plot(pca.var.per, main = "Total Variance", xlab="Principal Component",  
 ylab="Percent Variation", ylim = c(0,1), type = "b")



Hide

# From the Total Variance Plot, the first 4 principal components  
# are explaining most of the variance in the data  
# so I will make the judgement to use the first four components  
# to make a linear regression model  
pca\_data <- pca$x[,1:4]   
crime\_4\_pc <- cbind(pca\_data, Data\_crime[,16])  
  
  
smp\_siz = floor(0.75\*nrow(crime\_4\_pc))   
set.seed(123)  
train\_ind = sample(seq\_len(nrow(crime\_4\_pc)),size=smp\_siz)  
train = crime\_4\_pc[train\_ind,]  
test = crime\_4\_pc[-train\_ind,]  
  
# build a lr model with first 4 principal components  
model <- lm(V5 ~ . , data = as.data.frame(train))  
test\_cv <- cv.lm(as.data.frame(test), model,m=5)

Analysis of Variance Table  
  
Response: V5  
 Df Sum Sq Mean Sq F value Pr(>F)   
PC1 1 539158 539158 10.75 0.0135 \*   
PC2 1 88136 88136 1.76 0.2267   
PC3 1 139723 139723 2.78 0.1391   
PC4 1 845689 845689 16.85 0.0045 \*\*  
Residuals 7 351243 50178   
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
 As there is >1 explanatory variable, cross-validation  
 predicted values for a fold are not a linear function  
 of corresponding overall predicted values. Lines that  
 are shown for the different folds are approximate  
fold 1   
Observations in test set: 2   
 5 11  
Predicted 1687 895  
cvpred 1323 762  
V5 1993 1030  
CV residual 670 268  
  
Sum of squares = 521082 Mean square = 260541 n = 2   
  
fold 2   
Observations in test set: 3   
 4 6 8  
Predicted 592 769.4 871  
cvpred 880 791.8 1162  
V5 511 696.0 754  
CV residual -369 -95.8 -408  
  
Sum of squares = 311832 Mean square = 103944 n = 3   
  
fold 3   
Observations in test set: 3   
 1 3 9  
Predicted 1328.2 796 838  
cvpred 1302.3 656 739  
V5 1234.0 856 1072  
CV residual -68.3 200 333  
  
Sum of squares = 155472 Mean square = 51824 n = 3   
  
fold 4   
Observations in test set: 2   
 2 7  
Predicted 1187 230.41  
cvpred 1236 8.05  
V5 963 373.00  
CV residual -273 364.95  
  
Sum of squares = 207736 Mean square = 103868 n = 2   
  
fold 5   
Observations in test set: 2   
 10 12  
Predicted 580 1124  
cvpred 802 1356  
V5 566 849  
CV residual -236 -507  
  
Sum of squares = 312141 Mean square = 156071 n = 2   
  
Overall (Sum over all 2 folds)   
 ms   
125689



Hide

test\_cv

|  |
| --- |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC1**  <dbl> | **PC2**  <dbl> | **PC3**  <dbl> | **PC4**  <dbl> | **V5**  <dbl> | **Predicted**  <dbl> | **cvpred**  <dbl> |
| 1.839 | 1.3310 | 1.279 | 0.718 | 1234 | 1328 | 1302.29 |
| 0.246 | -0.0736 | -0.907 | 1.137 | 963 | 1187 | 1236.03 |
| -3.610 | -0.6862 | 1.284 | 0.552 | 856 | 796 | 655.67 |
| 0.516 | 0.9749 | 1.834 | -1.591 | 511 | 592 | 879.90 |
| 4.057 | 1.1753 | -0.817 | 1.670 | 1993 | 1687 | 1322.84 |
| -4.118 | -0.3807 | 1.435 | 0.633 | 696 | 769 | 791.76 |
| -0.681 | 1.6693 | -2.886 | -1.310 | 373 | 230 | 8.05 |
| 1.716 | -1.3084 | -0.560 | -0.706 | 754 | 871 | 1162.14 |
| -1.886 | 0.5906 | 1.436 | 0.182 | 1072 | 838 | 739.32 |
| -0.725 | 2.8926 | -0.363 | -0.506 | 566 | 580 | 801.77 |

Next

**1**2

Previous

1-10 of 12 rows

Hide

# Using cross validation it gives the overall mean square of prediction error  
# is 125689   
  
# My Solution to Question 8.2  
train\_original = Data\_crime[train\_ind,]  
test\_original = Data\_crime[-train\_ind,]  
train\_original

|  |
| --- |
|  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **M**  <dbl> | **So**  <int> | **Ed**  <dbl> | **Po1**  <dbl> | **Po2**  <dbl> | **LF**  <dbl> | **M.F**  <dbl> | **Pop**  <int> | **NW**  <dbl> |  |
| 14 | 13.5 | 0 | 11.7 | 6.2 | 6.1 | 0.595 | 98.6 | 22 | 4.6 |  |
| 37 | 17.7 | 1 | 8.7 | 5.8 | 5.6 | 0.638 | 97.4 | 24 | 34.9 |  |
| 19 | 13.0 | 0 | 11.6 | 12.8 | 12.8 | 0.536 | 93.4 | 51 | 2.4 |  |
| 39 | 14.9 | 1 | 8.8 | 6.1 | 5.4 | 0.515 | 95.3 | 36 | 16.5 |  |
| 41 | 14.8 | 0 | 12.2 | 7.2 | 6.6 | 0.601 | 99.8 | 9 | 1.9 |  |
| 2 | 14.3 | 0 | 11.3 | 10.3 | 9.5 | 0.583 | 101.2 | 13 | 10.2 |  |
| 22 | 15.7 | 1 | 8.9 | 4.7 | 4.4 | 0.512 | 96.2 | 22 | 42.3 |  |
| 36 | 15.0 | 0 | 10.0 | 10.9 | 9.8 | 0.531 | 96.4 | 9 | 2.4 |  |
| 43 | 16.2 | 1 | 9.9 | 7.5 | 7.0 | 0.522 | 99.6 | 40 | 20.8 |  |
| 18 | 13.5 | 1 | 10.4 | 12.3 | 11.5 | 0.537 | 97.8 | 31 | 17.0 |  |

Next

**1**234

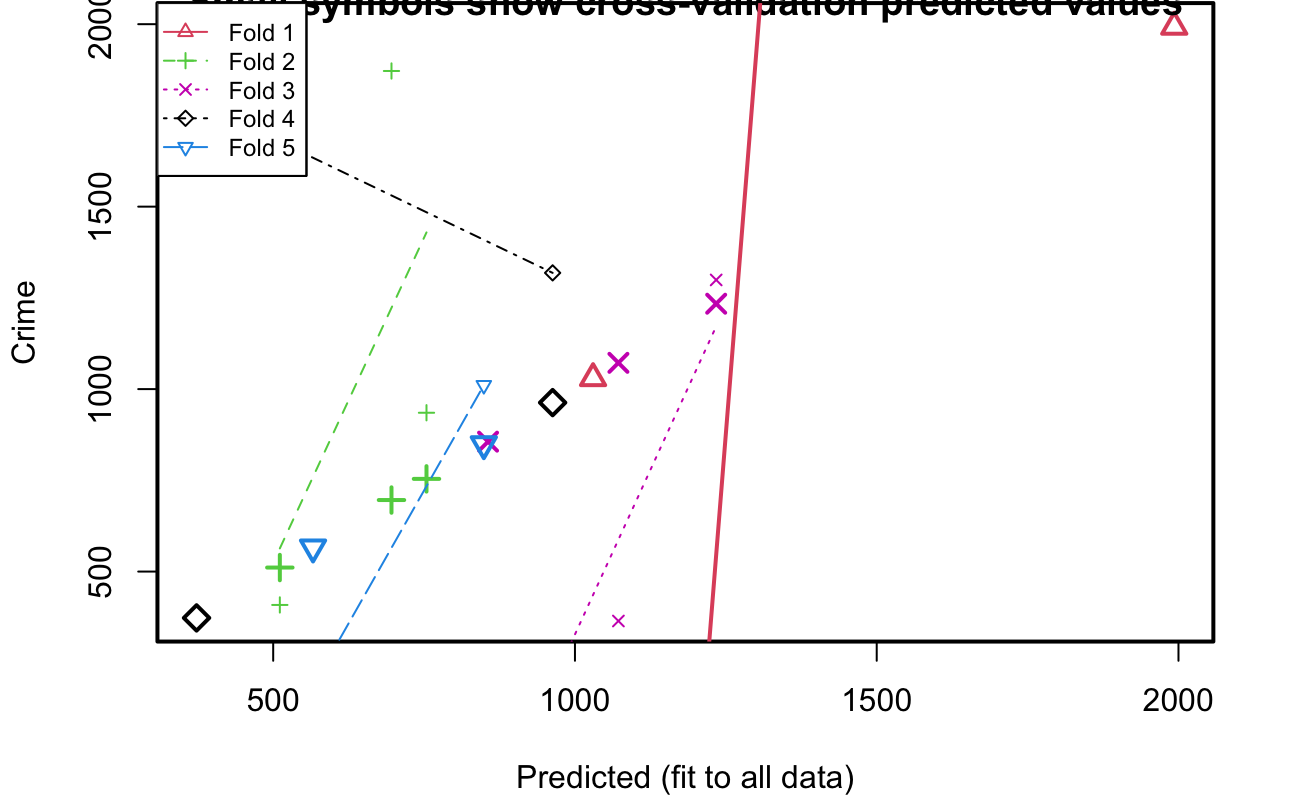
Previous

1-10 of 35 rows | 1-10 of 16 columns

Hide

model\_orig <- lm(Crime ~ ., train\_original)  
test\_orig\_cv <- cv.lm(as.data.frame(test\_original), model\_orig,m=5)

prediction from a rank-deficient fit may be misleadingprediction from a rank-deficient fit may be misleadingprediction from a rank-deficient fit may be misleadingprediction from a rank-deficient fit may be misleadingprediction from a rank-deficient fit may be misleadingANOVA F-tests on an essentially perfect fit are unreliableAnalysis of Variance Table  
  
Response: Crime  
 Df Sum Sq Mean Sq F value Pr(>F)  
M 1 19217 19217   
So 1 5922 5922   
Ed 1 955183 955183   
Po1 1 833986 833986   
Po2 1 425 425   
LF 1 16590 16590   
M.F 1 20652 20652   
Pop 1 2273 2273   
NW 1 22702 22702   
U1 1 30 30   
U2 1 86968 86968   
Residuals 0 0   
  
 As there is >1 explanatory variable, cross-validation  
 predicted values for a fold are not a linear function  
 of corresponding overall predicted values. Lines that  
 are shown for the different folds are approximate  
fold 1   
Observations in test set: 2   
 26 44  
Predicted 1993 1030  
cvpred 16265 -3679  
Crime 1993 1030  
CV residual -14272 4709  
  
Sum of squares = 2.26e+08 Mean square = 1.13e+08 n = 2   
  
fold 2   
Observations in test set: 3   
 13 30 32  
Predicted 511 696 754  
cvpred 409 1872 935  
Crime 511 696 754  
CV residual 102 -1176 -181  
  
Sum of squares = 1425699 Mean square = 475233 n = 3   
  
fold 3   
Observations in test set: 3   
 5 9 33  
Predicted 1234.0 856.0 1072  
cvpred 1299.2 -96.2 365  
Crime 1234.0 856.0 1072  
CV residual -65.2 952.2 707  
  
Sum of squares = 1411375 Mean square = 470458 n = 3   
  
fold 4   
Observations in test set: 2   
 7 31  
Predicted 963 373  
cvpred 1319 1787  
Crime 963 373  
CV residual -356 -1414  
  
Sum of squares = 2126770 Mean square = 1063385 n = 2   
  
fold 5   
Observations in test set: 2   
 38 47  
Predicted 566 849  
cvpred 186 1011  
Crime 566 849  
CV residual 380 -162  
  
Sum of squares = 170374 Mean square = 85187 n = 2   
  
Overall (Sum over all 2 folds)   
 ms   
19249220



Hide

# Using original solution, the overall mean square of prediction error is 19249220 which is much higher than the lm model with only 4 PCs   
  
#Next, convert the PCs back to predict   
# if pca$scale is TRUE, need to re-scale, o/w  
# use t(t(pca$x %\*% t(pca$rotation)) + pca$center)  
# convert 4 pca into original variables  
lm\_data <- t(t(pca$x[,1:4] %\*% t(pca$rotation[,1:4])) \* pca$scale + pca$center)  
print("The original variables ")

[1] "The original variables "

Hide

print(lm\_data)

M So Ed Po1 Po2 LF M.F Pop NW U1 U2 Wealth  
 [1,] 15.1 1.1747 8.68 6.02 5.67 0.501 96.3 33.894 26.1005 0.1033 4.10 3862  
 [2,] 13.5 0.0911 11.16 8.94 8.45 0.578 99.5 29.571 4.7373 0.0973 3.29 5623  
 [3,] 15.4 1.0311 9.05 4.77 4.51 0.529 97.4 12.176 22.6931 0.0956 3.43 3743  
 [4,] 12.2 -0.0166 11.45 14.50 13.67 0.556 96.5 99.166 5.7033 0.0939 3.93 6897  
 [5,] 13.5 0.0581 11.74 10.02 9.54 0.601 99.7 23.732 4.2427 0.0794 2.45 5923  
 [6,] 12.8 0.0798 11.74 12.72 12.08 0.576 98.6 55.082 5.5479 0.0872 3.19 6488  
 [7,] 13.5 0.4596 10.67 9.77 9.28 0.545 98.9 34.934 11.6050 0.1027 3.80 5470  
 [8,] 13.9 0.6704 10.36 10.99 10.42 0.541 96.4 60.238 17.7222 0.0833 3.34 5477  
 [9,] 15.4 1.0515 9.17 6.48 6.12 0.538 95.5 36.371 25.0187 0.0757 2.85 4057  
[10,] 13.7 -0.1122 11.74 6.91 6.57 0.614 102.0 -10.826 -0.9229 0.0932 2.53 5427  
[11,] 13.0 0.0292 10.93 12.33 11.57 0.567 95.1 100.599 7.7986 0.0815 3.36 6298  
[12,] 13.8 0.0105 10.90 8.35 7.83 0.588 97.8 47.115 4.9496 0.0864 2.92 5489  
[13,] 14.2 -0.0246 10.99 6.89 6.46 0.607 98.5 28.819 3.8403 0.0794 2.37 5207  
[14,] 14.1 0.1292 11.24 7.49 7.10 0.600 100.0 8.650 5.1287 0.0841 2.47 5273  
[15,] 14.9 0.8712 9.24 5.33 5.01 0.526 97.8 18.221 19.4811 0.1048 3.82 4025  
[16,] 14.4 1.0400 8.82 7.65 7.18 0.489 95.9 54.970 23.9987 0.1130 4.71 4342  
[17,] 13.8 0.2967 10.75 7.21 6.84 0.562 100.7 5.473 6.9953 0.1109 3.71 5083  
[18,] 13.7 0.9428 10.75 12.60 12.07 0.528 98.2 37.138 20.8539 0.0868 3.45 5691  
[19,] 12.8 0.1155 11.44 13.38 12.65 0.568 96.9 79.307 7.7441 0.0832 3.29 6541  
[20,] 12.2 0.0538 10.79 11.34 10.65 0.526 98.8 69.365 4.3735 0.1312 5.24 6172  
[21,] 13.5 -0.1579 10.86 7.31 6.80 0.587 98.6 42.684 1.1473 0.1002 3.37 5418  
[22,] 16.0 1.3140 8.15 4.15 3.88 0.508 95.0 32.130 29.7234 0.0902 3.52 3257  
[23,] 14.0 0.3435 10.40 8.54 8.03 0.563 97.2 49.111 11.2602 0.0883 3.21 5216  
[24,] 12.8 -0.0994 11.30 7.40 6.99 0.566 102.8 -0.436 -1.9409 0.1334 4.44 5546  
[25,] 14.4 0.1186 11.44 6.97 6.64 0.617 100.4 -3.686 4.7354 0.0741 1.88 5182  
[26,] 12.2 -0.1638 12.47 12.81 12.21 0.587 101.6 28.737 -1.4430 0.1028 3.48 6834  
[27,] 13.9 0.0676 11.14 8.30 7.84 0.595 98.8 30.789 5.1705 0.0831 2.64 5469  
[28,] 14.0 0.1358 10.93 6.99 6.60 0.585 100.0 12.441 5.0911 0.0962 3.03 5144  
[29,] 12.3 0.2338 10.89 16.34 15.38 0.530 93.6 138.058 12.7329 0.0863 4.17 6993  
[30,] 15.7 1.1214 9.07 5.87 5.56 0.539 95.6 29.092 26.1595 0.0733 2.69 3859  
[31,] 13.7 0.1168 10.37 4.81 4.48 0.554 101.7 -2.872 2.6937 0.1338 4.48 4703  
[32,] 13.0 0.0647 10.80 10.34 9.70 0.554 97.7 66.498 5.8112 0.1055 4.05 5894  
[33,] 14.9 0.6308 10.11 6.69 6.34 0.570 97.5 21.979 16.0342 0.0765 2.52 4554  
[34,] 13.0 0.0476 11.41 10.20 9.66 0.571 100.0 32.944 3.4661 0.1045 3.65 5973  
[35,] 12.7 0.0770 10.13 10.54 9.80 0.523 96.2 96.119 7.1036 0.1223 5.11 5861  
[36,] 13.5 0.0428 10.47 9.10 8.47 0.567 96.2 75.282 6.8511 0.0927 3.53 5567  
[37,] 16.0 0.9178 9.17 4.40 4.12 0.563 95.6 23.742 22.6673 0.0663 2.18 3695  
[38,] 14.3 0.1624 10.91 4.96 4.71 0.592 101.6 -18.408 3.9810 0.1013 2.91 4712  
[39,] 15.1 0.9053 9.05 5.95 5.57 0.529 95.9 40.035 21.9164 0.0915 3.49 4069  
[40,] 14.4 0.5783 9.96 8.37 7.88 0.550 96.2 53.337 16.1698 0.0851 3.23 4937  
[41,] 13.9 -0.1730 11.62 7.60 7.17 0.622 99.9 16.463 0.0898 0.0787 2.15 5559  
[42,] 13.9 0.4300 10.71 7.10 6.77 0.556 101.2 -4.900 8.8676 0.1118 3.71 4961  
[43,] 15.0 0.8271 9.68 7.28 6.88 0.551 96.1 38.206 20.6725 0.0753 2.75 4464  
[44,] 12.9 -0.1492 11.43 9.55 8.99 0.580 99.9 37.949 0.1106 0.1070 3.69 5986  
[45,] 14.2 0.6609 8.78 4.80 4.40 0.497 97.9 37.275 15.3621 0.1379 5.36 4069  
[46,] 13.4 0.1435 11.63 10.55 10.04 0.591 99.5 28.801 5.9393 0.0817 2.66 5959  
[47,] 12.8 -0.1483 11.99 8.92 8.50 0.590 103.0 -4.365 -2.6840 0.1144 3.57 5963  
 Ineq Prob Time  
 [1,] 25.4 0.07840 25.8  
 [2,] 17.6 0.04183 24.8  
 [3,] 25.3 0.07789 24.4  
 [4,] 14.0 0.01892 31.9  
 [5,] 16.3 0.04568 22.0  
 [6,] 14.8 0.04106 23.2  
 [7,] 18.7 0.06055 21.0  
 [8,] 19.3 0.05751 25.7  
 [9,] 24.6 0.06862 28.9  
[10,] 17.5 0.04688 20.0  
[11,] 16.3 0.00859 39.2  
[12,] 18.4 0.02305 33.5  
[13,] 19.0 0.02482 32.9  
[14,] 18.6 0.04762 23.5  
[15,] 24.2 0.07066 25.0  
[16,] 23.9 0.06813 27.8  
[17,] 19.5 0.06057 19.4  
[18,] 18.3 0.09199 11.8  
[19,] 15.1 0.03040 29.0  
[20,] 16.3 0.03226 27.4  
[21,] 18.4 0.01541 34.9  
[22,] 27.8 0.07560 30.7  
[23,] 19.8 0.03872 31.0  
[24,] 17.3 0.04992 17.8  
[25,] 18.7 0.05064 22.2  
[26,] 12.8 0.04835 15.1  
[27,] 18.2 0.03484 28.5  
[28,] 19.2 0.04540 24.8  
[29,] 14.5 0.01424 37.6  
[30,] 25.2 0.07322 28.1  
[31,] 20.6 0.05015 22.6  
[32,] 17.3 0.02587 31.7  
[33,] 22.0 0.05852 26.9  
[34,] 16.4 0.04447 22.0  
[35,] 18.0 0.01220 38.5  
[36,] 18.6 0.01194 39.3  
[37,] 25.6 0.05916 32.7  
[38,] 20.3 0.05707 20.3  
[39,] 24.5 0.05973 31.0  
[40,] 21.1 0.04600 31.4  
[41,] 17.4 0.02826 28.3  
[42,] 19.9 0.07277 15.5  
[43,] 22.8 0.06043 28.9  
[44,] 16.2 0.02968 26.5  
[45,] 24.3 0.05227 30.6  
[46,] 16.4 0.04935 21.3  
[47,] 15.6 0.05364 14.6

Hide

# Use cross validation to find the model with best R^2  
library("DAAG")  
lm\_original <- cbind(lm\_data, Data\_crime[,16])  
colnames(lm\_original)[16] <- "Crime"  
model\_lm\_original <- lm(Crime~., data = as.data.frame(lm\_original))  
model\_lm\_original$coef

(Intercept) M So Ed Po1 Po2 LF   
 -380.9 69.7 -132.5 -41.4 94.4 NA NA   
 M.F Pop NW U1 U2 Wealth Ineq   
 NA NA NA NA NA NA NA   
 Prob Time   
 NA NA

Hide

# Coefficients of original variables   
# Many dont have coefficients.  
#(Intercept) M So Ed Po1 Po2 LF   
# -380.9 69.7 -132.5 -41.4 94.4 NA NA   
# M.F Pop NW U1 U2 Wealth Ineq   
# NA NA NA NA NA NA NA   
# Prob Time   
# NA NA   
  
  
test<-data.frame(M = 14.0,So = 0,Ed = 10.0, Po1 = 12.0,Po2 = 15.5,  
 LF = 0.640, M.F = 94.0,Pop = 150,NW = 1.1,U1 = 0.120,  
 U2 = 3.6, Wealth = 3200,Ineq = 20.1,Prob = 0.04, Time = 39.0)  
  
print(predict(model\_orig,test))

1   
410

Hide

#Original prediction result from Question 8.2 is 410   
  
print(predict(model\_lm\_original ,test))

prediction from a rank-deficient fit may be misleading 1   
1314

Hide

#The output is 1314.   
#This prediction is quite reasonable than the original prediction.

Question 10.1 Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using (a) a regression tree model, and (b) a random forest model.

In R,you can use the tree package or the rpart package,and the random Forest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don’t just stop when you have a good model, but interpret it too).

Hide

# Regression Tree   
library(rpart) # performing regression trees  
library(rpart.plot)  
Data\_crime = read.csv("uscrime.txt",sep = "")  
library(ISLR)  
smp\_siz = floor(0.75\*nrow(Data\_crime))   
smp\_siz

[1] 35

Hide

set.seed(123)  
train\_ind = sample(seq\_len(nrow(Data\_crime)),size=smp\_siz)  
train = Data\_crime[train\_ind,]  
test = Data\_crime[-train\_ind,]  
dim(test)

[1] 12 16

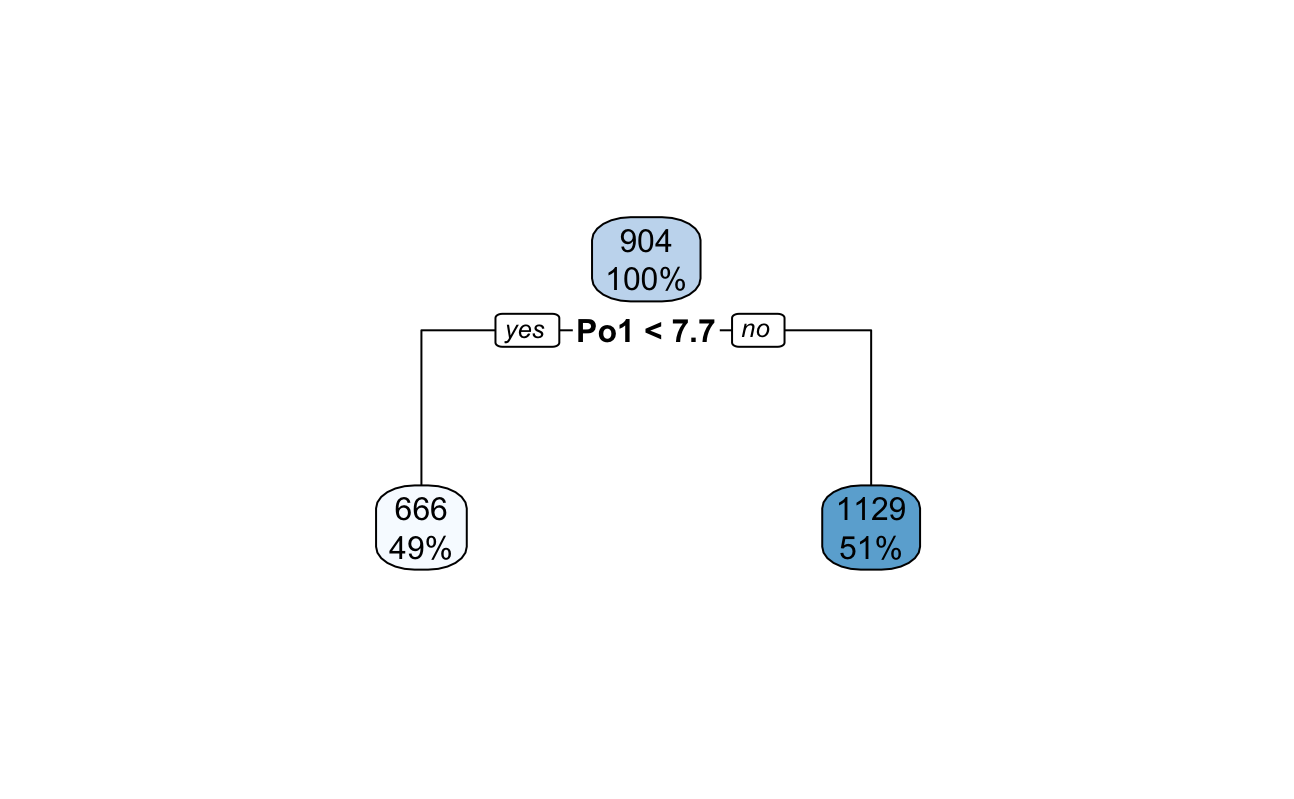
Hide

crime\_tree\_1<- rpart(Crime~ .,data=train, method = "anova")  
summary(crime\_tree\_1)

Call:  
rpart(formula = Crime ~ ., data = train, method = "anova")  
 n= 35   
  
 CP nsplit rel error xerror xstd  
1 0.38 0 1.00 1.08 0.294  
2 0.01 1 0.62 0.89 0.243  
  
Variable importance  
 Po1 Po2 Prob Wealth M Pop   
 25 25 15 13 12 10   
  
Node number 1: 35 observations, complexity param=0.38  
 mean=904, MSE=1.4e+05   
 left son=2 (17 obs) right son=3 (18 obs)  
 Primary splits:  
 Po1 < 7.65 to the left, improve=0.380, (0 missing)  
 Po2 < 7.2 to the left, improve=0.380, (0 missing)  
 Prob < 0.043 to the right, improve=0.298, (0 missing)  
 NW < 7.75 to the left, improve=0.281, (0 missing)  
 Pop < 41.5 to the left, improve=0.264, (0 missing)  
 Surrogate splits:  
 Po2 < 7.2 to the left, agree=1.000, adj=1.000, (0 split)  
 Prob < 0.043 to the right, agree=0.800, adj=0.588, (0 split)  
 Wealth < 5330 to the left, agree=0.771, adj=0.529, (0 split)  
 M < 13.3 to the right, agree=0.743, adj=0.471, (0 split)  
 Pop < 38 to the left, agree=0.714, adj=0.412, (0 split)  
  
Node number 2: 17 observations  
 mean=666, MSE=2.69e+04   
  
Node number 3: 18 observations  
 mean=1.13e+03, MSE=1.44e+05

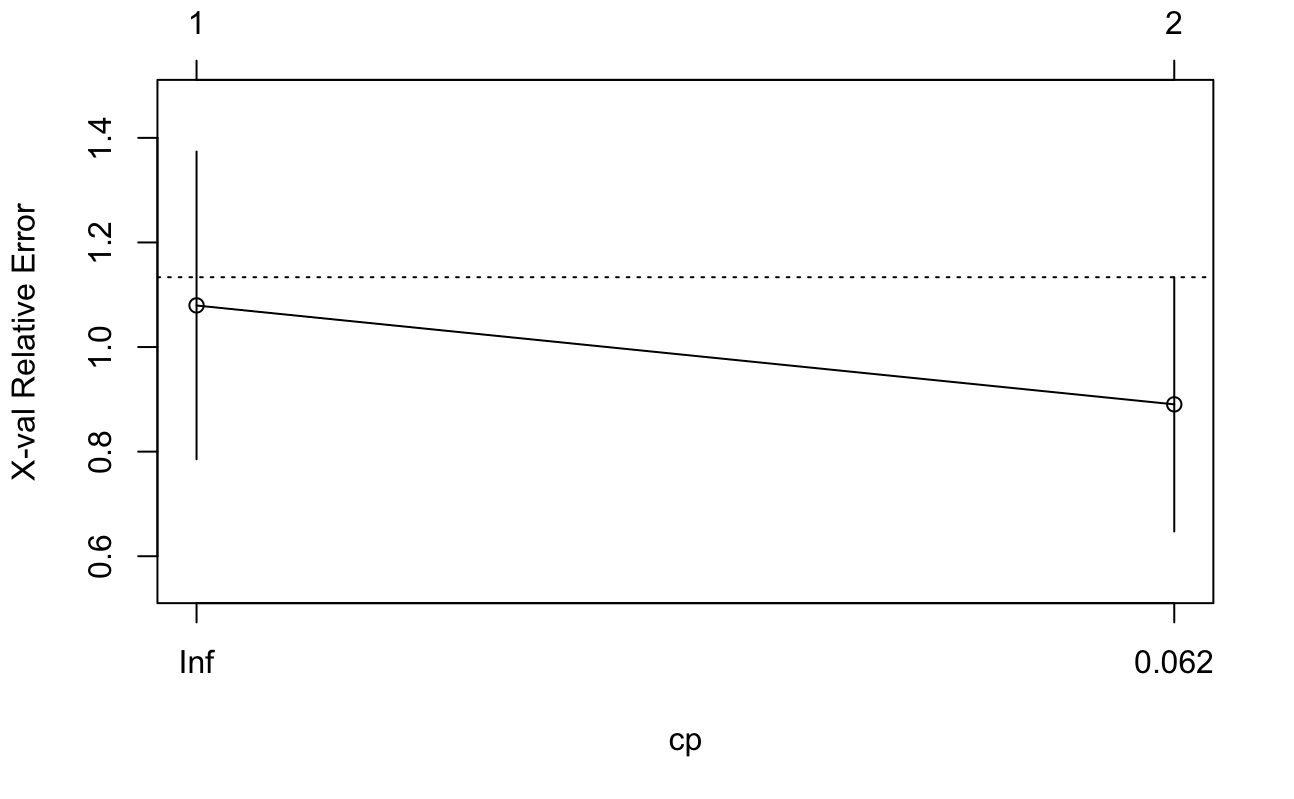
Hide

rpart.plot(crime\_tree\_1)



Hide

plotcp(crime\_tree\_1)



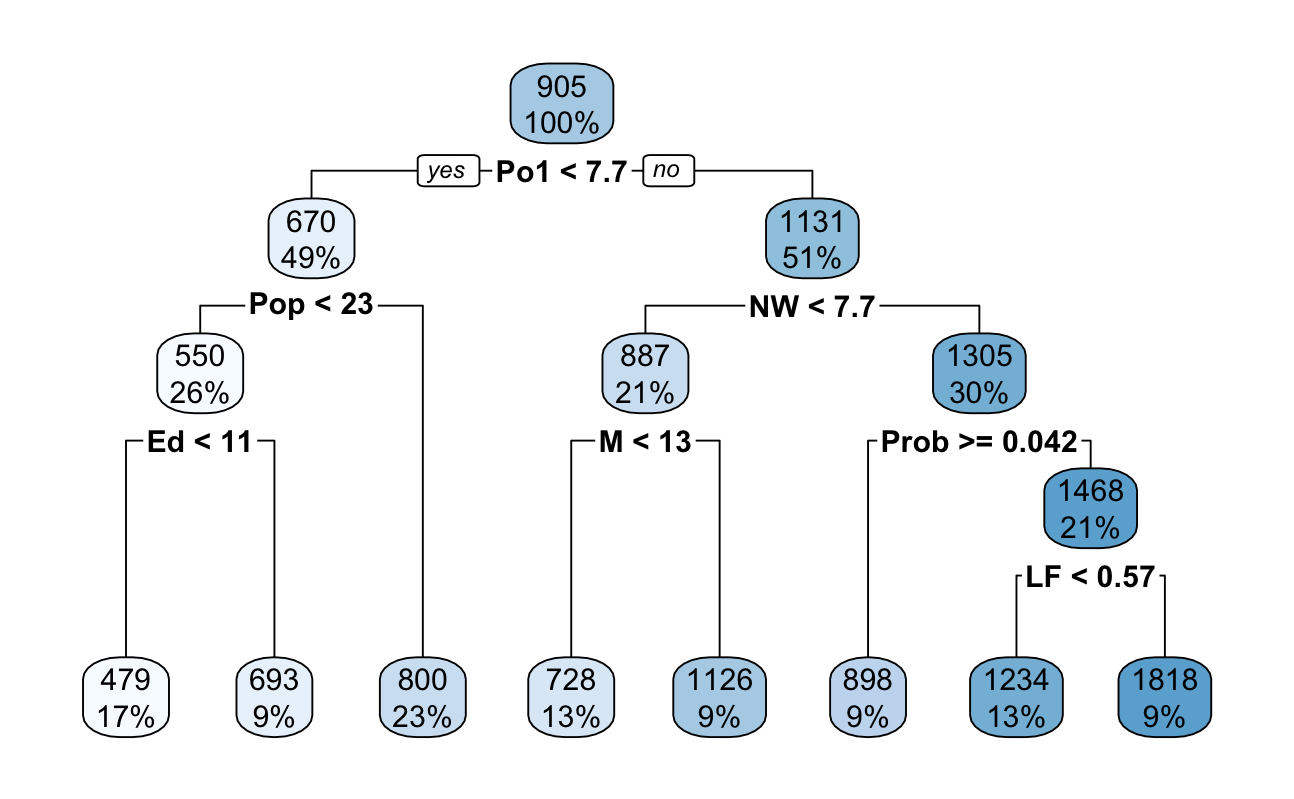
Hide

#The tree only has only 3 nodes based on variables "Po1" and "NW".  
# The reason might be the independent variables do not provide enough  
# information to grow the tree. So to see whether nodes might be allowed  
# I will loosen the control parameters  
  
#A regression tree with loosened control parameters  
crime\_tree\_2 <- rpart(Crime~ .,data=Data\_crime, method = "anova",control=rpart.control(minsplit=5, minbucket=4))  
  
summary(crime\_tree\_2)

Call:  
rpart(formula = Crime ~ ., data = Data\_crime, method = "anova",   
 control = rpart.control(minsplit = 5, minbucket = 4))  
 n= 47   
  
 CP nsplit rel error xerror xstd  
1 0.3630 0 1.000 1.037 0.258  
2 0.1481 1 0.637 0.896 0.211  
3 0.1348 2 0.489 1.027 0.268  
4 0.1187 3 0.354 1.009 0.266  
5 0.0554 4 0.235 0.902 0.248  
6 0.0517 5 0.180 0.872 0.240  
7 0.0177 6 0.128 0.861 0.248  
8 0.0100 7 0.111 0.795 0.240  
  
Variable importance  
 Po2 Po1 Prob Wealth Ineq M NW LF Ed Time Pop M.F   
 14 14 11 11 8 8 8 8 6 5 3 2   
 So   
 2   
  
Node number 1: 47 observations, complexity param=0.363  
 mean=905, MSE=1.46e+05   
 left son=2 (23 obs) right son=3 (24 obs)  
 Primary splits:  
 Po1 < 7.65 to the left, improve=0.363, (0 missing)  
 Po2 < 7.2 to the left, improve=0.363, (0 missing)  
 Prob < 0.0418 to the right, improve=0.322, (0 missing)  
 Wealth < 6470 to the left, improve=0.289, (0 missing)  
 NW < 7.65 to the left, improve=0.236, (0 missing)  
 Surrogate splits:  
 Po2 < 7.2 to the left, agree=1.000, adj=1.000, (0 split)  
 Wealth < 5330 to the left, agree=0.830, adj=0.652, (0 split)  
 Prob < 0.0436 to the right, agree=0.809, adj=0.609, (0 split)  
 M < 13.2 to the right, agree=0.745, adj=0.478, (0 split)  
 Ineq < 17.2 to the right, agree=0.745, adj=0.478, (0 split)  
  
Node number 2: 23 observations, complexity param=0.0517  
 mean=670, MSE=3.39e+04   
 left son=4 (12 obs) right son=5 (11 obs)  
 Primary splits:  
 Pop < 22.5 to the left, improve=0.457, (0 missing)  
 M < 14.5 to the left, improve=0.393, (0 missing)  
 Po1 < 5.65 to the left, improve=0.328, (0 missing)  
 NW < 5.4 to the left, improve=0.318, (0 missing)  
 U1 < 0.093 to the right, improve=0.212, (0 missing)  
 Surrogate splits:  
 NW < 5.4 to the left, agree=0.826, adj=0.636, (0 split)  
 M < 14.5 to the left, agree=0.783, adj=0.545, (0 split)  
 Time < 22.3 to the left, agree=0.783, adj=0.545, (0 split)  
 So < 0.5 to the left, agree=0.739, adj=0.455, (0 split)  
 Ed < 10.9 to the right, agree=0.739, adj=0.455, (0 split)  
  
Node number 3: 24 observations, complexity param=0.148  
 mean=1.13e+03, MSE=1.5e+05   
 left son=6 (10 obs) right son=7 (14 obs)  
 Primary splits:  
 NW < 7.65 to the left, improve=0.283, (0 missing)  
 M < 13 to the left, improve=0.271, (0 missing)  
 Wealth < 6470 to the left, improve=0.268, (0 missing)  
 Po2 < 11.6 to the left, improve=0.221, (0 missing)  
 Time < 21.1 to the left, improve=0.219, (0 missing)  
 Surrogate splits:  
 Ed < 11.4 to the right, agree=0.750, adj=0.4, (0 split)  
 Ineq < 16.2 to the left, agree=0.750, adj=0.4, (0 split)  
 Time < 21.9 to the left, agree=0.750, adj=0.4, (0 split)  
 Pop < 30 to the left, agree=0.708, adj=0.3, (0 split)  
 LF < 0.588 to the right, agree=0.667, adj=0.2, (0 split)  
  
Node number 4: 12 observations, complexity param=0.0177  
 mean=550, MSE=2.03e+04   
 left son=8 (8 obs) right son=9 (4 obs)  
 Primary splits:  
 Ed < 11.3 to the left, improve=0.500, (0 missing)  
 LF < 0.568 to the left, improve=0.482, (0 missing)  
 M.F < 96.8 to the left, improve=0.277, (0 missing)  
 Wealth < 5080 to the left, improve=0.233, (0 missing)  
 U2 < 3.35 to the right, improve=0.230, (0 missing)  
 Surrogate splits:  
 LF < 0.568 to the left, agree=0.917, adj=0.75, (0 split)  
 Po1 < 5.9 to the left, agree=0.833, adj=0.50, (0 split)  
 Po2 < 5.55 to the left, agree=0.833, adj=0.50, (0 split)  
 M.F < 98.1 to the left, agree=0.833, adj=0.50, (0 split)  
 U1 < 0.0785 to the right, agree=0.833, adj=0.50, (0 split)  
  
Node number 5: 11 observations  
 mean=800, MSE=1.63e+04   
  
Node number 6: 10 observations, complexity param=0.0554  
 mean=887, MSE=5.58e+04   
 left son=12 (6 obs) right son=13 (4 obs)  
 Primary splits:  
 M < 13 to the left, improve=0.684, (0 missing)  
 Pop < 21.5 to the right, improve=0.472, (0 missing)  
 Wealth < 5830 to the right, improve=0.251, (0 missing)  
 Time < 21.1 to the left, improve=0.218, (0 missing)  
 Ed < 11.7 to the left, improve=0.178, (0 missing)  
 Surrogate splits:  
 Wealth < 5660 to the right, agree=0.8, adj=0.50, (0 split)  
 Ineq < 17.2 to the left, agree=0.8, adj=0.50, (0 split)  
 Ed < 12 to the left, agree=0.7, adj=0.25, (0 split)  
 Po1 < 9.6 to the right, agree=0.7, adj=0.25, (0 split)  
 LF < 0.56 to the left, agree=0.7, adj=0.25, (0 split)  
  
Node number 7: 14 observations, complexity param=0.135  
 mean=1.3e+03, MSE=1.45e+05   
 left son=14 (4 obs) right son=15 (10 obs)  
 Primary splits:  
 Prob < 0.0419 to the right, improve=0.457, (0 missing)  
 Ed < 11.1 to the left, improve=0.438, (0 missing)  
 M.F < 99 to the left, improve=0.438, (0 missing)  
 Po2 < 11.6 to the left, improve=0.368, (0 missing)  
 Wealth < 6340 to the left, improve=0.368, (0 missing)  
 Surrogate splits:  
 LF < 0.52 to the left, agree=0.857, adj=0.50, (0 split)  
 NW < 13.2 to the right, agree=0.857, adj=0.50, (0 split)  
 Time < 21.4 to the left, agree=0.857, adj=0.50, (0 split)  
 So < 0.5 to the right, agree=0.786, adj=0.25, (0 split)  
 Po2 < 8.2 to the left, agree=0.786, adj=0.25, (0 split)  
  
Node number 8: 8 observations  
 mean=479, MSE=7.14e+03   
  
Node number 9: 4 observations  
 mean=693, MSE=1.62e+04   
  
Node number 12: 6 observations  
 mean=728, MSE=1.82e+04   
  
Node number 13: 4 observations  
 mean=1.13e+03, MSE=1.68e+04   
  
Node number 14: 4 observations  
 mean=898, MSE=7.06e+03   
  
Node number 15: 10 observations, complexity param=0.119  
 mean=1.47e+03, MSE=1.07e+05   
 left son=30 (6 obs) right son=31 (4 obs)  
 Primary splits:  
 LF < 0.574 to the left, improve=0.762, (0 missing)  
 Ed < 10.9 to the left, improve=0.396, (0 missing)  
 M.F < 99 to the left, improve=0.345, (0 missing)  
 Po1 < 11.4 to the left, improve=0.299, (0 missing)  
 Po2 < 10.7 to the left, improve=0.299, (0 missing)  
 Surrogate splits:  
 Ed < 11.2 to the left, agree=0.9, adj=0.75, (0 split)  
 Wealth < 6470 to the left, agree=0.9, adj=0.75, (0 split)  
 Po1 < 11.8 to the left, agree=0.8, adj=0.50, (0 split)  
 Po2 < 11.2 to the left, agree=0.8, adj=0.50, (0 split)  
 M.F < 99 to the left, agree=0.8, adj=0.50, (0 split)  
  
Node number 30: 6 observations  
 mean=1.23e+03, MSE=2.45e+04   
  
Node number 31: 4 observations  
 mean=1.82e+03, MSE=2.69e+04

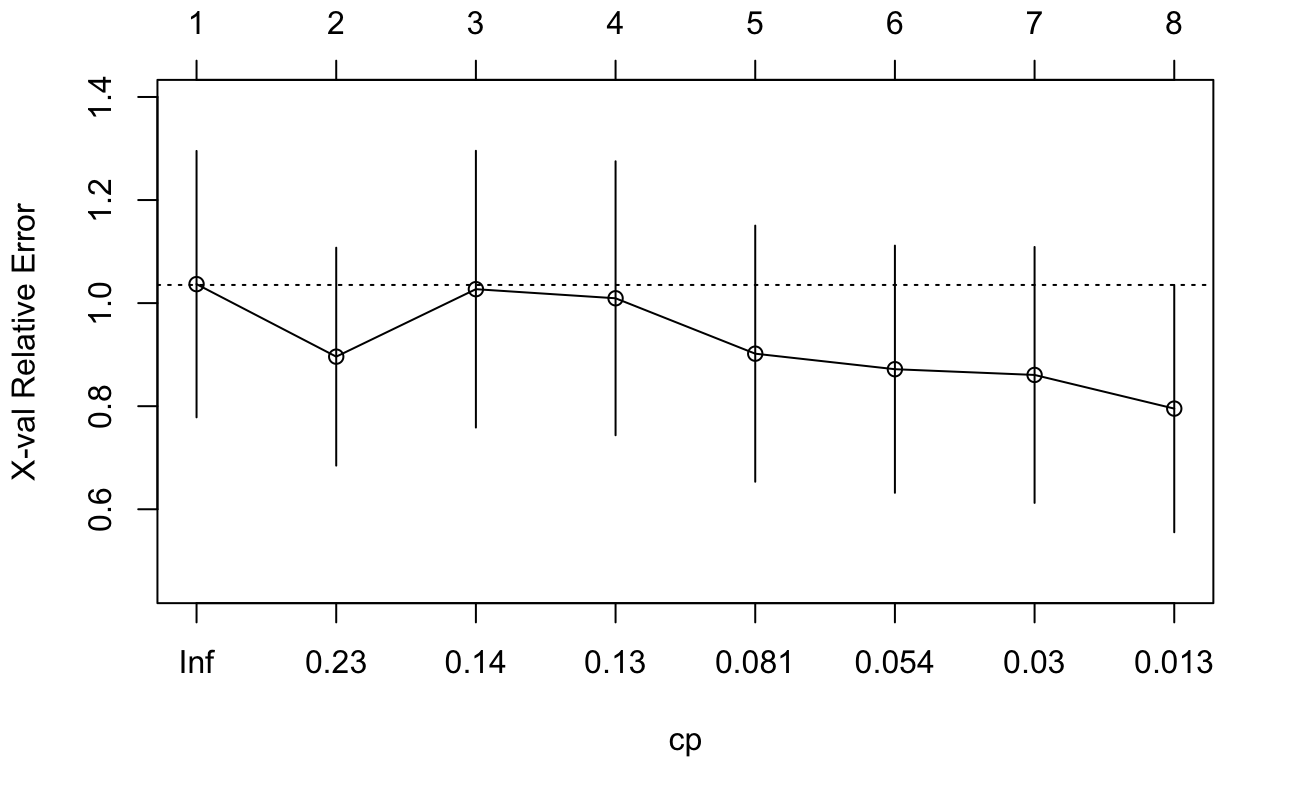
Hide

rpart.plot(crime\_tree\_2)



Hide

plotcp(crime\_tree\_2)



Hide

#Some qualitative takeaways:  
# 1. crime\_tree\_2 has far more terminal nodes than crime\_tree\_1  
# However, when I compare the end leaf nodes MSE   
# 2. crime\_tree\_2 is actually doing better in terms of minimizing mse  
# this is reasonable that because we have a small sample (47 observations)  
# so a more pruned tree might work better than a more fully grown tree  
# 3. I also see for both trees, the same variables rank highest in terms of   
# Variable importance which are : Po1, Po2, and Weath

Hide

library(randomForest)  
library(Metrics)  
  
#Check if there are any missing values， answer is no  
summary(Data\_crime)

M So Ed Po1 Po2   
 Min. :11.9 Min. :0.00 Min. : 8.70 Min. : 4.50 Min. : 4.10   
 1st Qu.:13.0 1st Qu.:0.00 1st Qu.: 9.75 1st Qu.: 6.25 1st Qu.: 5.85   
 Median :13.6 Median :0.00 Median :10.80 Median : 7.80 Median : 7.30   
 Mean :13.9 Mean :0.34 Mean :10.56 Mean : 8.50 Mean : 8.02   
 3rd Qu.:14.6 3rd Qu.:1.00 3rd Qu.:11.45 3rd Qu.:10.45 3rd Qu.: 9.70   
 Max. :17.7 Max. :1.00 Max. :12.20 Max. :16.60 Max. :15.70   
 LF M.F Pop NW U1   
 Min. :0.480 Min. : 93.4 Min. : 3.0 Min. : 0.2 Min. :0.0700   
 1st Qu.:0.530 1st Qu.: 96.4 1st Qu.: 10.0 1st Qu.: 2.4 1st Qu.:0.0805   
 Median :0.560 Median : 97.7 Median : 25.0 Median : 7.6 Median :0.0920   
 Mean :0.561 Mean : 98.3 Mean : 36.6 Mean :10.1 Mean :0.0955   
 3rd Qu.:0.593 3rd Qu.: 99.2 3rd Qu.: 41.5 3rd Qu.:13.2 3rd Qu.:0.1040   
 Max. :0.641 Max. :107.1 Max. :168.0 Max. :42.3 Max. :0.1420   
 U2 Wealth Ineq Prob Time   
 Min. :2.00 Min. :2880 Min. :12.6 Min. :0.0069 Min. :12.2   
 1st Qu.:2.75 1st Qu.:4595 1st Qu.:16.6 1st Qu.:0.0327 1st Qu.:21.6   
 Median :3.40 Median :5370 Median :17.6 Median :0.0421 Median :25.8   
 Mean :3.40 Mean :5254 Mean :19.4 Mean :0.0471 Mean :26.6   
 3rd Qu.:3.85 3rd Qu.:5915 3rd Qu.:22.8 3rd Qu.:0.0544 3rd Qu.:30.5   
 Max. :5.80 Max. :6890 Max. :27.6 Max. :0.1198 Max. :44.0   
 Crime   
 Min. : 342   
 1st Qu.: 658   
 Median : 831   
 Mean : 905   
 3rd Qu.:1058   
 Max. :1993

Hide

#build a Random Forest model  
rf <- randomForest(Crime ~ ., data=train)  
summary(rf)

Length Class Mode   
call 3 -none- call   
type 1 -none- character  
predicted 35 -none- numeric   
mse 500 -none- numeric   
rsq 500 -none- numeric   
oob.times 35 -none- numeric   
importance 15 -none- numeric   
importanceSD 0 -none- NULL   
localImportance 0 -none- NULL   
proximity 0 -none- NULL   
ntree 1 -none- numeric   
mtry 1 -none- numeric   
forest 11 -none- list   
coefs 0 -none- NULL   
y 35 -none- numeric   
test 0 -none- NULL   
inbag 0 -none- NULL   
terms 3 terms call

Hide

#Compare the predictions given by regression tree and random forest tree  
regression\_tree\_result <- predict(crime\_tree\_1, test)  
# 5 7 9 13 26 30 31 32 33 38 44 47   
# 887 1305 800 800 1305 800 550 1305 800 550 887 887   
mse(test$Crime,regression\_tree\_result)

[1] 111301

Hide

#MSE of regression tree is 111301  
  
rf\_result <- predict(rf, test)  
# 5 7 9 13 26 30 31 32 33 38 44 47   
# 992 1006 783 667 1240 774 726 1099 762 666 1077 962   
  
mse(test$Crime,rf\_result)

[1] 82609

Hide

#MSE of rf tree is 85615 which is lower than the regression tree  
#random forest is performing better however, it lacks   
#the interpretability of a simple regression tree

Question 10.2 Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

For credit card marketing campaign, logistic regression is used to assign a probability to a potential prospect about how likely for he/she to respond to an offer. Some predictors are: 1. income 2. industry 3. no. of times they visit the company’s website 4. if they made purchases with this company before, how much they spent 5. what is their credit score

Question 10.3 1. Using the GermanCredit data set germancredit.txt from <http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german> / (description at <http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29> ), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not.

Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link=”logit”) in your glm function call.

Hide

Data\_credit = read.csv("germancredit.txt",sep = "", header=FALSE)  
dim(Data\_credit)

[1] 1000 21

Hide

summary(Data\_credit)

V1 V2 V3 V4   
 Length:1000 Min. : 4.0 Length:1000 Length:1000   
 Class :character 1st Qu.:12.0 Class :character Class :character   
 Mode :character Median :18.0 Mode :character Mode :character   
 Mean :20.9   
 3rd Qu.:24.0   
 Max. :72.0   
 V5 V6 V7 V8   
 Min. : 250 Length:1000 Length:1000 Min. :1.00   
 1st Qu.: 1366 Class :character Class :character 1st Qu.:2.00   
 Median : 2320 Mode :character Mode :character Median :3.00   
 Mean : 3271 Mean :2.97   
 3rd Qu.: 3972 3rd Qu.:4.00   
 Max. :18424 Max. :4.00   
 V9 V10 V11 V12   
 Length:1000 Length:1000 Min. :1.00 Length:1000   
 Class :character Class :character 1st Qu.:2.00 Class :character   
 Mode :character Mode :character Median :3.00 Mode :character   
 Mean :2.85   
 3rd Qu.:4.00   
 Max. :4.00   
 V13 V14 V15 V16   
 Min. :19.0 Length:1000 Length:1000 Min. :1.00   
 1st Qu.:27.0 Class :character Class :character 1st Qu.:1.00   
 Median :33.0 Mode :character Mode :character Median :1.00   
 Mean :35.5 Mean :1.41   
 3rd Qu.:42.0 3rd Qu.:2.00   
 Max. :75.0 Max. :4.00   
 V17 V18 V19 V20   
 Length:1000 Min. :1.00 Length:1000 Length:1000   
 Class :character 1st Qu.:1.00 Class :character Class :character   
 Mode :character Median :1.00 Mode :character Mode :character   
 Mean :1.16   
 3rd Qu.:1.00   
 Max. :2.00   
 V21   
 Min. :1.0   
 1st Qu.:1.0   
 Median :1.0   
 Mean :1.3   
 3rd Qu.:2.0   
 Max. :2.0

Hide

table(Data\_credit$V21) #70% are good and 30% are bad

1 2   
700 300

Hide

# 1 = Good, 2 = Bad  
Data\_credit$V21

[1] 1 2 1 1 2 1 1 1 1 2 2 2 1 2 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1  
 [41] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1  
 [81] 2 1 1 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1 1 2 1 2 1  
 [121] 2 1 1 1 2 1 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1  
 [161] 1 1 1 1 1 1 2 1 1 2 2 1 2 1 2 2 1 1 1 1 2 2 2 1 2 1 2 1 2 1 2 2 2 1 2 2 1 2 1 2  
 [201] 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 2 1 1 1 1 2 2 2 1 1  
 [241] 2 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 2 1 1 1 2 1  
 [281] 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 2 1 2 1 1 2 2 1 1 1 1 2 1 2 1 1 1 1  
 [321] 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 2 1 2 1 2  
 [361] 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1  
 [401] 1 1 2 1 1 2 1 1 1 2 1 1 2 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 2 1 1 1 2  
 [441] 1 1 1 2 2 1 2 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 1 2 2 1 1 1 1  
 [481] 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 2 2 1 1 1 2 1 1 2 2 2 1 2 1 1 2 1 1 1 1 1 1 2 1 1  
 [521] 1 2 2 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 2 1 2 1 2 2 1 2 1 1 2 1 1 1 2 1 1 2 2 2 2 2  
 [561] 1 2 1 2 1 1 2 1 1 2 2 1 1 1 1 1 1 1 2 1 2 1 1 2 1 2 1 1 2 2 1 1 1 2 2 2 2 2 2 1  
 [601] 1 2 2 2 1 1 1 2 1 1 2 2 1 1 2 1 1 1 2 1 1 2 2 1 2 1 1 2 1 1 1 2 1 2 2 1 1 1 1 2  
 [641] 2 1 2 1 1 2 1 2 2 2 1 2 2 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1  
 [681] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2 1 1 1 2 1 1 2 1 1 1 1 1  
 [721] 2 2 2 1 2 1 1 2 2 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 2 1 2  
 [761] 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 2 2 2 1 1 1 1 1 2 1 1 1  
 [801] 1 1 1 1 1 2 1 1 1 2 1 1 2 2 2 1 1 1 1 2 1 1 2 1 1 1 2 2 2 1 1 2 2 1 2 2 1 1 1 1  
 [841] 2 1 2 1 1 1 2 1 1 2 2 1 1 2 1 1 1 1 2 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1  
 [881] 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 2 2  
 [921] 1 1 2 1 2 2 1 2 1 1 1 2 1 1 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2 2 1 2 2 2 1 1 1 1 2 1  
 [961] 1 1 1 2 1 1 2 1 1 1 1 1 2 2 1 1 1 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1

Hide

Data\_credit$V21[Data\_credit$V21==1] <- 0  
Data\_credit$V21[Data\_credit$V21==2] <- 1  
  
# Split data into training and testing  
smp\_siz = floor(0.75\*nrow(Data\_credit))   
smp\_siz

[1] 750

Hide

set.seed(123)  
train\_ind = sample(seq\_len(nrow(Data\_credit)),size=smp\_siz)  
train = Data\_credit[train\_ind,]  
test = Data\_credit[-train\_ind,]  
  
# Train a logistic Model  
creditLogitModel <- glm(V21 ~ ., data=train,family=binomial(link="logit"))  
summary(creditLogitModel)

Call:  
glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-2.199 -0.711 -0.372 0.743 2.832   
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) 4.41e-01 1.40e+00 0.31 0.75281   
V1A12 -3.13e-01 2.51e-01 -1.24 0.21331   
V1A13 -1.00e+00 4.12e-01 -2.43 0.01530 \*   
V1A14 -1.81e+00 2.76e-01 -6.56 5.3e-11 \*\*\*  
V2 2.65e-02 1.06e-02 2.50 0.01247 \*   
V3A31 -9.82e-02 6.55e-01 -0.15 0.88088   
V3A32 -7.09e-01 4.84e-01 -1.47 0.14276   
V3A33 -7.76e-01 5.24e-01 -1.48 0.13901   
V3A34 -1.42e+00 4.94e-01 -2.86 0.00418 \*\*   
V4A41 -1.51e+00 4.42e-01 -3.42 0.00062 \*\*\*  
V4A410 -1.46e+00 8.60e-01 -1.70 0.08848 .   
V4A42 -6.19e-01 3.02e-01 -2.05 0.04039 \*   
V4A43 -7.62e-01 2.90e-01 -2.63 0.00858 \*\*   
V4A44 -5.40e-01 9.70e-01 -0.56 0.57778   
V4A45 -2.94e-01 6.23e-01 -0.47 0.63728   
V4A46 1.89e-01 4.50e-01 0.42 0.67533   
V4A48 -1.88e+00 1.31e+00 -1.44 0.14935   
V4A49 -6.74e-01 3.86e-01 -1.75 0.08096 .   
V5 1.10e-04 4.97e-05 2.21 0.02694 \*   
V6A62 -4.57e-01 3.37e-01 -1.35 0.17550   
V6A63 -1.56e-01 4.66e-01 -0.33 0.73763   
V6A64 -1.50e+00 6.49e-01 -2.31 0.02115 \*   
V6A65 -9.78e-01 3.12e-01 -3.14 0.00172 \*\*   
V7A72 -1.01e-01 5.18e-01 -0.19 0.84593   
V7A73 -1.76e-01 4.89e-01 -0.36 0.71867   
V7A74 -8.08e-01 5.26e-01 -1.54 0.12426   
V7A75 -9.55e-02 4.88e-01 -0.20 0.84470   
V8 3.27e-01 1.02e-01 3.21 0.00134 \*\*   
V9A92 -1.74e-01 4.41e-01 -0.40 0.69269   
V9A93 -9.09e-01 4.32e-01 -2.10 0.03552 \*   
V9A94 -3.44e-01 5.27e-01 -0.65 0.51356   
V10A102 3.01e-01 4.42e-01 0.68 0.49646   
V10A103 -1.11e+00 5.22e-01 -2.13 0.03332 \*   
V11 -6.20e-02 1.01e-01 -0.61 0.54076   
V12A122 7.51e-02 2.99e-01 0.25 0.80170   
V12A123 1.31e-01 2.71e-01 0.48 0.63001   
V12A124 2.63e-01 5.23e-01 0.50 0.61521   
V13 -1.83e-02 1.09e-02 -1.68 0.09333 .   
V14A142 -2.25e-02 4.50e-01 -0.05 0.96011   
V14A143 -7.14e-01 2.81e-01 -2.54 0.01108 \*   
V15A152 -3.61e-01 2.86e-01 -1.26 0.20727   
V15A153 -2.33e-01 5.70e-01 -0.41 0.68294   
V16 2.18e-01 2.09e-01 1.04 0.29703   
V17A172 1.05e+00 1.00e+00 1.05 0.29546   
V17A173 9.36e-01 9.82e-01 0.95 0.34038   
V17A174 9.46e-01 9.90e-01 0.96 0.33937   
V18 2.92e-01 2.91e-01 1.00 0.31566   
V19A192 -1.67e-01 2.33e-01 -0.72 0.47373   
V20A202 -1.37e+00 7.27e-01 -1.89 0.05936 .   
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 916.30 on 749 degrees of freedom  
Residual deviance: 674.06 on 701 degrees of freedom  
AIC: 772.1  
  
Number of Fisher Scoring iterations: 5

Hide

#This outputs the probability  
creditPredict <- predict(creditLogitModel,newdata = test[,-21],type="response")  
  
#Given the default threshold of 0.5, below is confusion matrix  
cm\_1 <- table(round(creditPredict)<0.5,test$V21)  
cm\_1

0 1  
 FALSE 22 33  
 TRUE 153 42

Hide

# 0 1  
# FALSE 22 33  
# TRUE 153 42  
accuracy\_1 <- (cm\_1[1,1]+cm\_1[2,2])/sum(cm\_1)  
accuracy\_1 #0.256

[1] 0.256

Hide

#To improve the models by using a different threshold  
threshold <- 0.7  
cm\_2 <- table(round(creditPredict) > threshold,test$V21)  
names(dimnames(cm\_2)) <- c("Predicted", "Observed")  
cm\_2

Observed  
Predicted 0 1  
 FALSE 153 42  
 TRUE 22 33

Hide

accuracy\_2 <- (cm\_2[1,1]+cm\_2[2,2])/sum(cm\_2)  
accuracy\_2 #0.744, the accuracy improves significantly

[1] 0.744

1. Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between “good” and “bad” answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

Hide

# Identify 1(bad) as 0(good) is 5 times worse (cost $5) than identify 0(good) as 1(bad) (cost $1)  
  
threshold <- 0.7  
cm\_2 <- table(round(creditPredict > threshold),test$V21)  
total\_cost <- 5\*cm\_2[1,2] + 1\*cm\_2[2,1]  
total\_cost

[1] 300

Hide

cost\_threshold= function(X){  
 threshold <- X  
 creditPredict <- predict(creditLogitModel,newdata = test[,-21],type="response")  
 cm <- table(round(creditPredict > X),test$V21)  
 total\_cost <- 5\*cm[1,2] + 1\*cm[2,1]  
 total\_cost  
 return(total\_cost)  
}  
  
# Try out 13 different threshold ranges from 0.1 to 0.9  
threshold\_li <- seq(from = 0.1, to = 0.9, by = 0.05)  
a <- length(threshold\_li)  
  
cost = rep(0,a)  
  
for (x in 1:17){  
 a <- threshold\_li[x]  
 cost[x] <- cost\_threshold(a)  
}  
  
cm <- table(round(creditPredict > threshold\_li[which.min(cost)]  
),test$V21)  
cm

0 1  
 0 92 4  
 1 83 71

Hide

# Threshold 0.15 gives the least cost out of all the thresholds  
# This makes sense as wrongly identify a true bad account cost so much higher  
# than wrongly identify a good account  
# the threshold is set especially low for an account to be identified bad ("0")  
# 0 1  
# 0 92 4  
# 1 83 71