library("GGally")

## Loading required package: ggplot2

library("DAAG")

## Loading required package: lattice

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(pROC)

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

##   
## Attaching package: 'pROC'

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

set.seed(1234)

#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.

crime<- read.table("http://www.statsci.org/data/general/uscrime.txt",header=TRUE)  
head(crime)

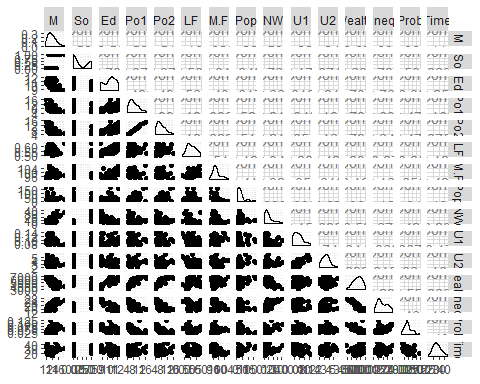
## M So Ed Po1 Po2 LF M.F Pop NW U1 U2 Wealth Ineq  
## 1 15.1 1 9.1 5.8 5.6 0.510 95.0 33 30.1 0.108 4.1 3940 26.1  
## 2 14.3 0 11.3 10.3 9.5 0.583 101.2 13 10.2 0.096 3.6 5570 19.4  
## 3 14.2 1 8.9 4.5 4.4 0.533 96.9 18 21.9 0.094 3.3 3180 25.0  
## 4 13.6 0 12.1 14.9 14.1 0.577 99.4 157 8.0 0.102 3.9 6730 16.7  
## 5 14.1 0 12.1 10.9 10.1 0.591 98.5 18 3.0 0.091 2.0 5780 17.4  
## 6 12.1 0 11.0 11.8 11.5 0.547 96.4 25 4.4 0.084 2.9 6890 12.6  
## Prob Time Crime  
## 1 0.084602 26.2011 791  
## 2 0.029599 25.2999 1635  
## 3 0.083401 24.3006 578  
## 4 0.015801 29.9012 1969  
## 5 0.041399 21.2998 1234  
## 6 0.034201 20.9995 682

#Check out if there are correlations between the predictors

names(crime)

## [1] "M" "So" "Ed" "Po1" "Po2" "LF" "M.F"   
## [8] "Pop" "NW" "U1" "U2" "Wealth" "Ineq" "Prob"   
## [15] "Time" "Crime"

ggpairs(crime,columns=c("M","So","Ed","Po1","Po2","LF","M.F","Pop","NW","U1","U2","Wealth","Ineq","Prob","Time"))



#There are correlations between Po1 vs Po2, Wealth vs Ed/Po1/Po2/Ineq , so PCA is a good choose.

# remove the response variable (it’s in the 16th column)

vars<-crime[-16]  
pca<-prcomp(vars, scale = TRUE)  
summary(pca)

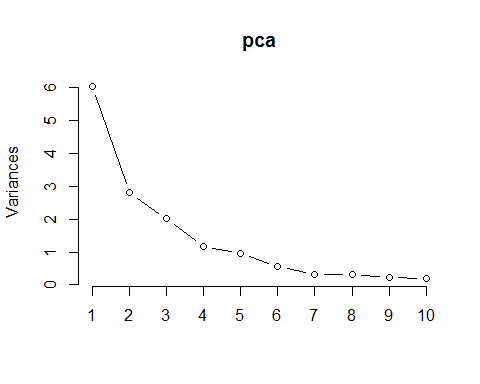
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 2.4534 1.6739 1.4160 1.07806 0.97893 0.74377  
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688  
## Cumulative Proportion 0.4013 0.5880 0.7217 0.79920 0.86308 0.89996  
## PC7 PC8 PC9 PC10 PC11 PC12  
## Standard deviation 0.56729 0.55444 0.48493 0.44708 0.41915 0.35804  
## Proportion of Variance 0.02145 0.02049 0.01568 0.01333 0.01171 0.00855  
## Cumulative Proportion 0.92142 0.94191 0.95759 0.97091 0.98263 0.99117  
## PC13 PC14 PC15  
## Standard deviation 0.26333 0.2418 0.06793  
## Proportion of Variance 0.00462 0.0039 0.00031  
## Cumulative Proportion 0.99579 0.9997 1.00000

#get the eigenvector of the matrix

eigen<-pca$rotation

#Use the screenplot to plot the variance of each princpal component

screeplot(pca,type="line",col="black")



#get the first 4 pc

pc<-pca$x[,1:4]

#fit a linear regression model with the these 4 pc

crimepc<-as.data.frame(cbind(pc,crime$Crime))  
modelpca<-lm(V5~.,crimepc)  
summary(modelpca)

##   
## Call:  
## lm(formula = V5 ~ ., data = crimepc)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -557.76 -210.91 -29.08 197.26 810.35   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 905.09 49.07 18.443 < 2e-16 \*\*\*  
## PC1 65.22 20.22 3.225 0.00244 \*\*   
## PC2 -70.08 29.63 -2.365 0.02273 \*   
## PC3 25.19 35.03 0.719 0.47602   
## PC4 69.45 46.01 1.509 0.13872   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 336.4 on 42 degrees of freedom  
## Multiple R-squared: 0.3091, Adjusted R-squared: 0.2433   
## F-statistic: 4.698 on 4 and 42 DF, p-value: 0.003178

#Get the parameters for the original model,scaled

beta0<-modelpca$coefficients[1]  
betas<-modelpca$coefficients[2:5]

#coefficents equals beta times eigenvector matrix

alpha<-eigen[,1:4] %\*% betas  
alpha

## [,1]  
## M -21.277963  
## So 10.223091  
## Ed 14.352610  
## Po1 63.456426  
## Po2 64.557974  
## LF -14.005349  
## M.F -24.437572  
## Pop 39.830667  
## NW 15.434545  
## U1 -27.222281  
## U2 1.425902  
## Wealth 38.607855  
## Ineq -27.536348  
## Prob 3.295707  
## Time -6.612616

mean <- sapply(vars, mean)  
sd <- sapply(vars, sd)

#get the un-scaled coefficents for each input

alpha\_org<- alpha/sd

#get the un-scaled intercept

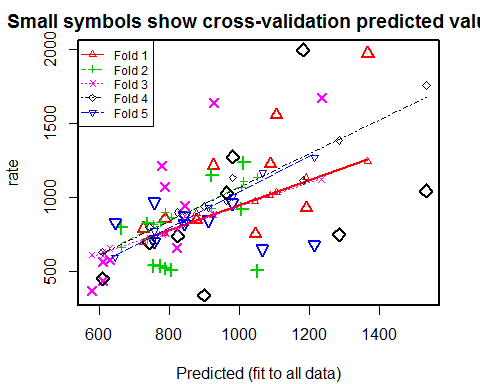
beta\_org <-beta0-sum(alpha\* mean/sd)  
  
point<-data.frame(  
 M = 14.0,  
 So = 0,  
 Ed = 10.0,  
 Po1 = 12.0,  
 Po2 = 15.5,  
 LF = 0.640,  
 M.F = 94.0,  
 Pop = 150,  
 NW = 1.1,  
 U1 = 0.120,  
 U2 = 3.6,  
 Wealth = 3200,  
 Ineq = 20.1,  
 Prob = 0.04,  
 Time = 39.0  
)  
  
predict<-beta\_org+sum(alpha\_org\*point)

#cross validate the model

rate<-crime[,16]  
PClist <- as.data.frame(pca$x[, 1:4])  
PC<-cbind(rate, PClist)  
model2 <- lm(rate ~ ., PC)  
cv <-cv.lm(PC, model2, m = 5)

## Analysis of Variance Table  
##   
## Response: rate  
## Df Sum Sq Mean Sq F value Pr(>F)   
## PC1 1 1177568 1177568 10.40 0.0024 \*\*  
## PC2 1 633037 633037 5.59 0.0227 \*   
## PC3 1 58541 58541 0.52 0.4760   
## PC4 1 257832 257832 2.28 0.1387   
## Residuals 42 4753950 113189   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Warning in cv.lm(PC, model2, m = 5):   
##   
## 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: 9   
## 1 4 8 9 18 20 23 32 47  
## Predicted 726.3 1368 1107 788 1192 1089 926 1046 878.1  
## cvpred 699.9 1245 1034 757 1134 1014 880 976 894.5  
## rate 791.0 1969 1555 856 929 1225 1216 754 849.0  
## CV residual 91.1 724 521 99 -205 211 336 -222 -45.5  
##   
## Sum of squares = 1064640 Mean square = 118293 n = 9   
##   
## fold 2   
## Observations in test set: 10   
## 5 13 15 17 25 34 39 40 42 46  
## Predicted 1014 806 664 774 788 1007 739 922 757 1051  
## cvpred 1110 866 662 830 901 1065 720 925 822 1138  
## rate 1234 511 798 539 523 923 826 1151 542 508  
## CV residual 124 -355 136 -291 -378 -142 106 226 -280 -630  
##   
## Sum of squares = 946539 Mean square = 94654 n = 10   
##   
## fold 3   
## Observations in test set: 10   
## 2 3 11 14 16 22 28 31 33 38  
## Predicted 927 630.4 1236 824 845.0 612 781 580 790 611  
## cvpred 887 657.4 1123 779 873.6 635 751 614 756 609  
## rate 1635 578.0 1674 664 946.0 439 1216 373 1072 566  
## CV residual 748 -79.4 551 -115 72.4 -196 465 -241 316 -43  
##   
## Sum of squares = 1301458 Mean square = 130146 n = 10   
##   
## fold 4   
## Observations in test set: 9   
## 19 21 26 27 29 30 36 44 45  
## Predicted 1286 825 1183 900 1535 743 982 965.3 610  
## cvpred 1385 902 1117 940 1755 802 1133 978.1 634  
## rate 750 742 1993 342 1043 696 1272 1030.0 455  
## CV residual -635 -160 876 -598 -712 -106 139 51.9 -179  
##   
## Sum of squares = 2126795 Mean square = 236311 n = 9   
##   
## fold 5   
## Observations in test set: 9   
## 6 7 10 12 24 35 37 41 43  
## Predicted 1216 982.4 758.2 913.8 758 1067 646 843.8 844.8  
## cvpred 1271 1009.9 734.3 935.7 776 1166 598 834.7 833.3  
## rate 682 963.0 705.0 849.0 968 653 831 880.0 823.0  
## CV residual -589 -46.9 -29.3 -86.7 192 -513 233 45.3 -10.3  
##   
## Sum of squares = 714380 Mean square = 79376 n = 9   
##   
## Overall (Sum over all 9 folds)   
## ms   
## 130932

mn<- mean(rate)  
R2 <- 1 - attr(cv, "ms") \* nrow(crime) / sum((rate - mn) ^ 2)  
R2

## [1] 0.106

In conclusion, the model generated by the PCA method is: Crime=1666.485-16.9307630*M+21.3436771*So+12.8297238*Ed +21.3521593*Po1+23.0883154*Po2 -346.5657125*LF-8.2930969*M.F+1.0462155*Pop+1.5009941*NW-1509.9345216*U1+1.6883674*U2 +0.0400119*Wealth-6.9020218*Ineq+144.9492678*Prob-0.9330765\*Time

The adjusted R-square of this model is 0.2433, cross-validated R-square is 0.0392,which is pretty low. The crime rate for the city with given data is 1112.678 Compared to my model for question 8.2, with predictors:M, Ed, Po1, M.F, U1,U2,Ineq,Prob, and R-square: 0.7444, and crime rate: 1038.296, My conclustion is, adding more principle compenents to the model may be helpful (currenlty we used first 4 compnents). Although PCA method addressed for the correlations between the predcitors and ranked the coordinates by importance. It didn’t address for the overfitting, which is a big issue in our data.

#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.

#(b) a random forest model

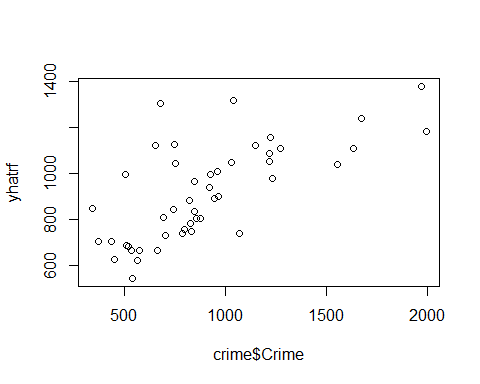
#Set the number of predictors at each split of the tree to be 4 (mtry=4),which is calculated based 1+log(n)=1+log(16)=4

rf <- randomForest(Crime ~ ., data=crime, mtry=4,importance=TRUE, na.action=na.omit)  
rf

##   
## Call:  
## randomForest(formula = Crime ~ ., data = crime, mtry = 4, importance = TRUE, na.action = na.omit)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## Mean of squared residuals: 80064  
## % Var explained: 45.3

# Plot of actual vs. predicted crime values

yhatrf <- predict(rf)  
plot(crime$Crime, yhatrf)



#Calculate sum of square error-resdiduals

SSres <- sum((yhatrf-crime$Crime)^2)

#Calculate sum of square error-total and R-squared

SStot <- sum((crime$Crime - mean(crime$Crime))^2)  
rs <- 1 - SSres/SStot  
rs

## [1] 0.453

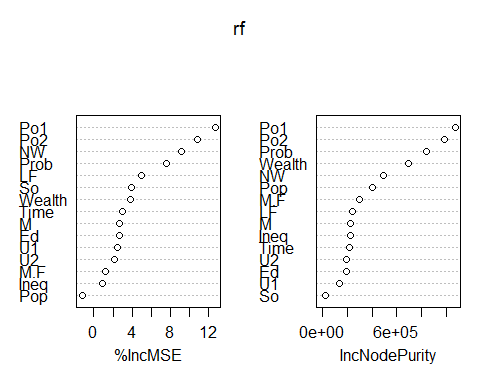
#This model is slightly better than the previous model.But it is not a real model. Iit is the average of all the different trees, which is better than just #one tree.

#variable importance

round(importance(rf), 2)

## %IncMSE IncNodePurity  
## M 2.70 228111  
## So 3.97 25893  
## Ed 2.63 197340  
## Po1 12.72 1069106  
## Po2 10.81 977901  
## LF 4.96 244263  
## M.F 1.17 296405  
## Pop -1.23 403519  
## NW 9.16 489672  
## U1 2.44 142153  
## U2 2.14 199194  
## Wealth 3.81 691927  
## Ineq 0.86 227026  
## Prob 7.64 837897  
## Time 2.97 218910

varImpPlot(rf)

 #We can see that Po1 is the most important variable among all the predictors.It also suggest the overfitting if we use all the predictors in the model.

#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. #The likelyhood of the applicant be admiited to the graduate school #Predictors: #GRE score, GPA from the undergraduate,have related research exprience or not, whether or not the undergraduate major is related to the program applying for

#Question 10.3 #1. Using the GermanCredit data set germancredit.txt, 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.

credit<- read.table("germancredit.txt",header=FALSE)  
head(credit)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17  
## 1 A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173  
## 2 A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173  
## 3 A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172  
## 4 A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173  
## 5 A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173  
## 6 A14 36 A32 A46 9055 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172  
## V18 V19 V20 V21  
## 1 1 A192 A201 1  
## 2 1 A191 A201 2  
## 3 2 A191 A201 1  
## 4 2 A191 A201 1  
## 5 2 A191 A201 2  
## 6 2 A192 A201 1

str(credit)

## 'data.frame': 1000 obs. of 21 variables:  
## $ V1 : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...  
## $ V2 : int 6 48 12 42 24 36 24 36 12 30 ...  
## $ V3 : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...  
## $ V4 : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...  
## $ V5 : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...  
## $ V6 : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...  
## $ V7 : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...  
## $ V8 : int 4 2 2 2 3 2 3 2 2 4 ...  
## $ V9 : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...  
## $ V10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...  
## $ V11: int 4 2 3 4 4 4 4 2 4 2 ...  
## $ V12: Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...  
## $ V13: int 67 22 49 45 53 35 53 35 61 28 ...  
## $ V14: Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ V15: Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...  
## $ V16: int 2 1 1 1 2 1 1 1 1 2 ...  
## $ V17: Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...  
## $ V18: int 1 1 2 2 2 2 1 1 1 1 ...  
## $ V19: Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...  
## $ V20: Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...  
## $ V21: int 1 2 1 1 2 1 1 1 1 2 ...

#accordingly to the description, we found that V21 is the response. 1 means good, 2 means bad. #Recode the V21 to be a 0/1 variable, instead of 1/2

credit$V21[credit$V21==1]<-0  
credit$V21[credit$V21==2]<-1

# Divide the data into training and test datasets.

trainno <- sample(1:nrow(credit), size = round(nrow(credit)\*0.7), replace = FALSE)  
train <- credit[trainno,]  
test<- credit[-trainno,]

#Fit the logistic model

log<-glm(V21~.,data=train,family=binomial(link="logit"))  
summary(log)

##   
## Call:  
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.255 -0.670 -0.304 0.629 2.782   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.48e-01 1.36e+00 0.48 0.63371   
## V1A12 -4.03e-01 2.69e-01 -1.50 0.13376   
## V1A13 -1.42e+00 4.96e-01 -2.86 0.00430 \*\*   
## V1A14 -1.67e+00 2.90e-01 -5.77 8.1e-09 \*\*\*  
## V2 3.15e-02 1.14e-02 2.75 0.00597 \*\*   
## V3A31 6.10e-01 6.98e-01 0.87 0.38197   
## V3A32 -8.16e-01 5.41e-01 -1.51 0.13122   
## V3A33 -1.53e+00 6.06e-01 -2.52 0.01166 \*   
## V3A34 -1.87e+00 5.65e-01 -3.31 0.00093 \*\*\*  
## V4A41 -1.85e+00 4.65e-01 -3.98 6.8e-05 \*\*\*  
## V4A410 -8.77e-01 8.86e-01 -0.99 0.32214   
## V4A42 -1.04e+00 3.29e-01 -3.17 0.00155 \*\*   
## V4A43 -8.39e-01 3.04e-01 -2.76 0.00574 \*\*   
## V4A44 -1.62e+01 4.77e+02 -0.03 0.97299   
## V4A45 -4.44e-01 7.28e-01 -0.61 0.54143   
## V4A46 3.14e-01 4.95e-01 0.63 0.52588   
## V4A48 -1.69e+00 1.41e+00 -1.20 0.22892   
## V4A49 -8.35e-01 4.14e-01 -2.02 0.04363 \*   
## V5 1.44e-04 5.47e-05 2.63 0.00859 \*\*   
## V6A62 -6.01e-01 3.54e-01 -1.70 0.08972 .   
## V6A63 -9.20e-01 5.99e-01 -1.54 0.12460   
## V6A64 -1.29e+00 7.08e-01 -1.82 0.06831 .   
## V6A65 -8.51e-01 3.19e-01 -2.66 0.00773 \*\*   
## V7A72 8.95e-02 5.22e-01 0.17 0.86386   
## V7A73 -1.94e-01 4.93e-01 -0.39 0.69377   
## V7A74 -6.44e-01 5.37e-01 -1.20 0.23078   
## V7A75 -1.51e-01 4.99e-01 -0.30 0.76209   
## V8 3.70e-01 1.11e-01 3.34 0.00084 \*\*\*  
## V9A92 -3.79e-01 4.56e-01 -0.83 0.40525   
## V9A93 -1.18e+00 4.48e-01 -2.64 0.00839 \*\*   
## V9A94 -7.41e-01 5.39e-01 -1.38 0.16881   
## V10A102 9.87e-01 4.90e-01 2.02 0.04380 \*   
## V10A103 -1.06e+00 5.43e-01 -1.95 0.05165 .   
## V11 -8.35e-03 1.07e-01 -0.08 0.93784   
## V12A122 4.22e-01 3.21e-01 1.31 0.18874   
## V12A123 1.89e-01 2.92e-01 0.65 0.51744   
## V12A124 4.52e-01 5.26e-01 0.86 0.39015   
## V13 -2.60e-02 1.17e-02 -2.22 0.02645 \*   
## V14A142 3.59e-01 5.02e-01 0.72 0.47425   
## V14A143 -4.56e-01 3.02e-01 -1.51 0.13110   
## V15A152 -3.39e-01 2.92e-01 -1.16 0.24573   
## V15A153 -2.85e-01 5.87e-01 -0.49 0.62743   
## V16 4.84e-01 2.44e-01 1.98 0.04732 \*   
## V17A172 2.17e-01 8.67e-01 0.25 0.80225   
## V17A173 5.36e-01 8.33e-01 0.64 0.52020   
## V17A174 4.89e-01 8.21e-01 0.60 0.55121   
## V18 2.34e-01 3.12e-01 0.75 0.45276   
## V19A192 -3.12e-01 2.55e-01 -1.22 0.22073   
## V20A202 -3.15e+00 1.32e+00 -2.38 0.01714 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 873.04 on 699 degrees of freedom  
## Residual deviance: 592.50 on 651 degrees of freedom  
## AIC: 690.5  
##   
## Number of Fisher Scoring iterations: 14

#keep the significant preditors under p-value=0.1,for the categorical predictors, keep them if any of the categories are significant. Then re-fit the model

log2<-glm(V21~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V12+V14+V16+V19+V20,data=train,family=binomial(link="logit"))  
summary(log2)

##   
## Call:  
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +   
## V10 + V12 + V14 + V16 + V19 + V20, family = binomial(link = "logit"),   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.301 -0.704 -0.316 0.641 2.789   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.10e-01 9.89e-01 -0.21 0.83215   
## V1A12 -4.11e-01 2.66e-01 -1.55 0.12195   
## V1A13 -1.48e+00 4.83e-01 -3.06 0.00219 \*\*   
## V1A14 -1.71e+00 2.84e-01 -6.02 1.7e-09 \*\*\*  
## V2 3.50e-02 1.12e-02 3.12 0.00183 \*\*   
## V3A31 5.65e-01 6.84e-01 0.83 0.40867   
## V3A32 -8.07e-01 5.28e-01 -1.53 0.12651   
## V3A33 -1.52e+00 5.96e-01 -2.55 0.01083 \*   
## V3A34 -1.90e+00 5.50e-01 -3.45 0.00057 \*\*\*  
## V4A41 -1.74e+00 4.52e-01 -3.84 0.00012 \*\*\*  
## V4A410 -7.94e-01 8.78e-01 -0.90 0.36586   
## V4A42 -9.05e-01 3.21e-01 -2.82 0.00477 \*\*   
## V4A43 -8.03e-01 2.98e-01 -2.69 0.00707 \*\*   
## V4A44 -1.64e+01 4.70e+02 -0.03 0.97221   
## V4A45 -6.02e-01 7.31e-01 -0.82 0.41021   
## V4A46 3.22e-01 4.82e-01 0.67 0.50415   
## V4A48 -1.63e+00 1.49e+00 -1.10 0.27246   
## V4A49 -7.96e-01 4.09e-01 -1.95 0.05160 .   
## V5 1.35e-04 5.27e-05 2.56 0.01051 \*   
## V6A62 -5.22e-01 3.49e-01 -1.49 0.13535   
## V6A63 -9.58e-01 5.85e-01 -1.64 0.10119   
## V6A64 -1.28e+00 6.84e-01 -1.88 0.06051 .   
## V6A65 -8.65e-01 3.14e-01 -2.75 0.00590 \*\*   
## V7A72 3.56e-01 4.47e-01 0.80 0.42571   
## V7A73 5.99e-02 4.10e-01 0.15 0.88400   
## V7A74 -3.85e-01 4.64e-01 -0.83 0.40583   
## V7A75 -1.09e-01 4.27e-01 -0.26 0.79770   
## V8 3.38e-01 1.07e-01 3.15 0.00163 \*\*   
## V9A92 -1.60e-01 4.42e-01 -0.36 0.71763   
## V9A93 -9.98e-01 4.34e-01 -2.30 0.02142 \*   
## V9A94 -4.93e-01 5.24e-01 -0.94 0.34679   
## V10A102 1.00e+00 4.87e-01 2.06 0.03985 \*   
## V10A103 -1.08e+00 5.33e-01 -2.03 0.04201 \*   
## V12A122 3.93e-01 3.14e-01 1.25 0.21003   
## V12A123 2.31e-01 2.85e-01 0.81 0.41726   
## V12A124 3.53e-01 3.80e-01 0.93 0.35202   
## V14A142 3.10e-01 4.94e-01 0.63 0.53103   
## V14A143 -3.81e-01 2.94e-01 -1.30 0.19516   
## V16 4.79e-01 2.34e-01 2.05 0.04084 \*   
## V19A192 -3.19e-01 2.34e-01 -1.36 0.17233   
## V20A202 -3.21e+00 1.33e+00 -2.41 0.01574 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 873.04 on 699 degrees of freedom  
## Residual deviance: 602.35 on 659 degrees of freedom  
## AIC: 684.4  
##   
## Number of Fisher Scoring iterations: 14

#For the categorical variables, not all the levels are significant. So create a binary (0/1) variable for each of them: 0 for not significant, 1 for significant

train$V1A12[train$V1=="A12"]<-1  
train$V1A12[train$V1!="A12"]<-0  
  
train$V1A13[train$V1=="A13"]<-1  
train$V1A13[train$V1!="A13"]<-0  
  
train$V1A14[train$V1=="A14"]<-1  
train$V1A14[train$V1!="A14"]<-0  
  
train$V3A34[train$V1=="A34"]<-1  
train$V3A34[train$V1!="A34"]<-0  
  
train$V4A41[train$V1=="A41"]<-1  
train$V4A41[train$V1!="A41"]<-0  
  
train$V4A42[train$V1=="A42"]<-1  
train$V4A42[train$V1!="A42"]<-0  
  
train$V4A43[train$V1=="A43"]<-1  
train$V4A43[train$V1!="A43"]<-0  
  
train$V6A65[train$V1=="A65"]<-1  
train$V6A65[train$V1!="A65"]<-0  
  
train$V7A74[train$V1=="A74"]<-1  
train$V7A74[train$V1!="A74"]<-0  
  
train$V9A92[train$V1=="A92"]<-1  
train$V9A92[train$V1!="A92"]<-0  
  
train$V9A93[train$V1=="A93"]<-1  
train$V9A93[train$V1!="A93"]<-0  
  
train$V10A103[train$V1=="A103"]<-1  
train$V10A103[train$V1!="A103"]<-0  
  
train$V12A124[train$V1=="A124"]<-1  
train$V12A124[train$V1!="A124"]<-0  
  
train$V14A143[train$V1=="A143"]<-1  
train$V14A143[train$V1!="A143"]<-0  
  
train$V19A192[train$V1=="A192"]<-1  
train$V19A192[train$V1!="A192"]<-0  
  
train$V20A202[train$V1=="A202"]<-1  
train$V20A202[train$V1!="A202"]<-0

#re-fit the model with these significant variables

log3<-glm(V21~V1A12+V1A13+V1A14+V2+V3A34+V4A41+V4A42+V4A43+V5+V6A65+V7A74+V8+V9A92+V9A93+V10A103+V12A124+V14A143+V16+V19A192+V20A202,data=train,family=binomial(link="logit"))  
summary(log3)

##   
## Call:  
## glm(formula = V21 ~ V1A12 + V1A13 + V1A14 + V2 + V3A34 + V4A41 +   
## V4A42 + V4A43 + V5 + V6A65 + V7A74 + V8 + V9A92 + V9A93 +   
## V10A103 + V12A124 + V14A143 + V16 + V19A192 + V20A202, family = binomial(link = "logit"),   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.802 -0.870 -0.482 0.988 2.387   
##   
## Coefficients: (13 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.59e+00 4.15e-01 -3.82 0.00013 \*\*\*  
## V1A12 -5.36e-01 2.14e-01 -2.51 0.01224 \*   
## V1A13 -1.33e+00 4.27e-01 -3.12 0.00180 \*\*   
## V1A14 -2.05e+00 2.43e-01 -8.43 < 2e-16 \*\*\*  
## V2 2.70e-02 9.45e-03 2.86 0.00422 \*\*   
## V3A34 NA NA NA NA   
## V4A41 NA NA NA NA   
## V4A42 NA NA NA NA   
## V4A43 NA NA NA NA   
## V5 9.31e-05 4.28e-05 2.17 0.02982 \*   
## V6A65 NA NA NA NA   
## V7A74 NA NA NA NA   
## V8 2.62e-01 9.23e-02 2.84 0.00448 \*\*   
## V9A92 NA NA NA NA   
## V9A93 NA NA NA NA   
## V10A103 NA NA NA NA   
## V12A124 NA NA NA NA   
## V14A143 NA NA NA NA   
## V16 -5.17e-02 1.59e-01 -0.33 0.74500   
## V19A192 NA NA NA NA   
## V20A202 NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 873.04 on 699 degrees of freedom  
## Residual deviance: 733.00 on 692 degrees of freedom  
## AIC: 749  
##   
## Number of Fisher Scoring iterations: 4

#only keep the significant terms

log4<-glm(V21~V1A12+V1A13+V1A14+V2+V5+V8,data=train,family=binomial(link="logit"))  
summary(log4)

##   
## Call:  
## glm(formula = V21 ~ V1A12 + V1A13 + V1A14 + V2 + V5 + V8, family = binomial(link = "logit"),   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.794 -0.869 -0.482 0.996 2.376   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.66e+00 3.55e-01 -4.67 3.1e-06 \*\*\*  
## V1A12 -5.34e-01 2.14e-01 -2.50 0.0126 \*   
## V1A13 -1.33e+00 4.26e-01 -3.11 0.0018 \*\*   
## V1A14 -2.06e+00 2.43e-01 -8.45 < 2e-16 \*\*\*  
## V2 2.73e-02 9.43e-03 2.89 0.0039 \*\*   
## V5 9.21e-05 4.27e-05 2.16 0.0311 \*   
## V8 2.61e-01 9.23e-02 2.83 0.0046 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 873.04 on 699 degrees of freedom  
## Residual deviance: 733.11 on 693 degrees of freedom  
## AIC: 747.1  
##   
## Number of Fisher Scoring iterations: 4

#Now every term are significant, this is the final model

#Add the remained binary variables to the test dataset

test$V1A12[test$V1=="A12"]<-1  
test$V1A12[test$V1!="A12"]<-0  
  
test$V1A13[test$V1=="A13"]<-1  
test$V1A13[test$V1!="A13"]<-0  
  
test$V1A14[test$V1=="A14"]<-1  
test$V1A14[test$V1!="A14"]<-0

#validate the model using the test dataset

yhatlog<-predict(log4,test,type = "response")  
head(yhatlog)

## 1 6 8 9 11 14   
## 0.4158 0.2019 0.4883 0.0704 0.2768 0.5382

#round the yhatlog to be 0/1 variabls

y<- as.integer(yhatlog > 0.5)  
head(y)

## [1] 0 0 0 0 0 1

t <- table(y,test$V21)  
t

##   
## y 0 1  
## 0 204 57  
## 1 17 22

correct<-(182+33)/300  
correct

## [1] 0.717

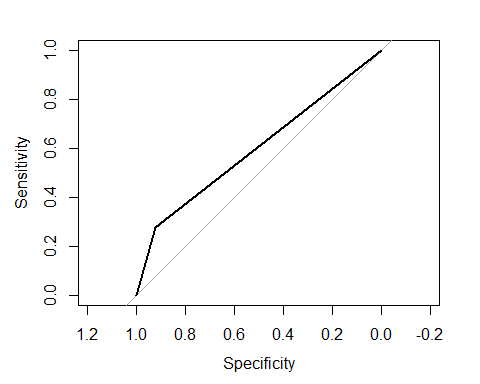
roc<-roc(test$V21,y)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

# Plot the ROC curve

plot(roc)



roc

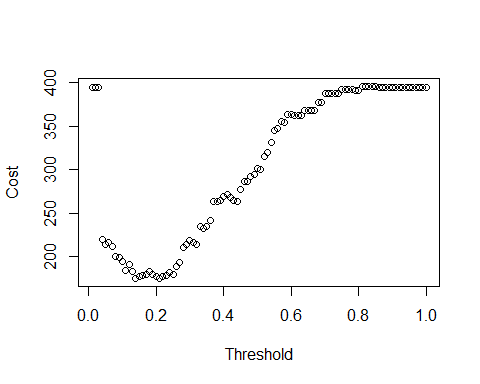
##   
## Call:  
## roc.default(response = test$V21, predictor = y)  
##   
## Data: y in 221 controls (test$V21 0) < 79 cases (test$V21 1).  
## Area under the curve: 0.601

#The model I developed is: log(p/(1-p))=-1.285e+00-5.099e-01*V1A12-1.155e+00*V1A13-2.316e+00*V1A14+2.545e-02*V2+9.637e-05*V5+1.633e-01*V8 The accuracy rate is 71.67%, AIC is 717.06,and AUC is 61.67%,which means the model will correctly classify the samples 61.67% of the times.

#2. 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.

Calulating loss for tthe cost for thresholds ranging from 0.01 to 1.

cost <- c()  
for(i in 1:100){  
 y.hat<- as.integer(yhatlog > (i/100)) #0.01-100  
   
 table<-as.matrix(table(y.hat,test$V21))  
   
 if(nrow(table)>1) { cst1 <- table[2,1] } else { cst1 <- 0 }  
 if(ncol(table)>1) { cst2 <- table[1,2] } else { cst2 <- 0 }  
 cost <- c(cost, cst1+cst2\*5)  
}  
  
plot(c(1:100)/100,cost,xlab = "Threshold",ylab = "Cost")



which.min(cost)

## [1] 14

cost

## [1] 395 395 395 220 214 216 212 200 199 195 184 191 183 175 177 178 179  
## [18] 183 180 177 175 177 178 182 179 189 193 210 214 219 216 214 235 232  
## [35] 235 242 264 264 265 269 272 268 265 264 277 287 287 292 295 302 300  
## [52] 315 320 332 345 348 356 355 364 364 363 363 363 368 368 368 368 378  
## [69] 378 388 388 388 388 388 393 393 393 393 391 391 396 396 396 396 396  
## [86] 395 395 395 395 395 395 395 395 395 395 395 395 395 395 395

#when threshold=0.08, we have minimum cost 187.