Homework 4

library("GGally")

library("DAAG")

library(tree)  
library(randomForest)

library(pROC)

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)

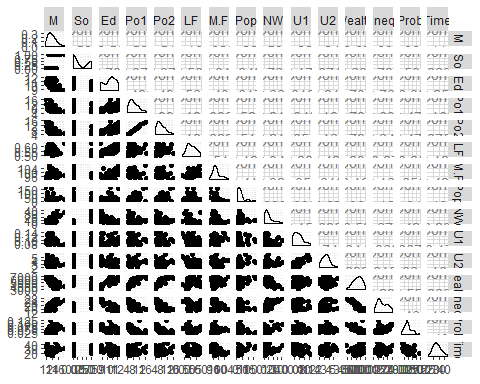
## M So Ed Po1 Po2 LF M.F Pop NW U1 U2 Wealth Ineq  
## 1 15.1 1 9.1 5.8 5.6 0.510 95.0 33 30.1 0.108 4.1 3940 26.1  
## 2 14.3 0 11.3 10.3 9.5 0.583 101.2 13 10.2 0.096 3.6 5570 19.4  
## 3 14.2 1 8.9 4.5 4.4 0.533 96.9 18 21.9 0.094 3.3 3180 25.0  
## 4 13.6 0 12.1 14.9 14.1 0.577 99.4 157 8.0 0.102 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)

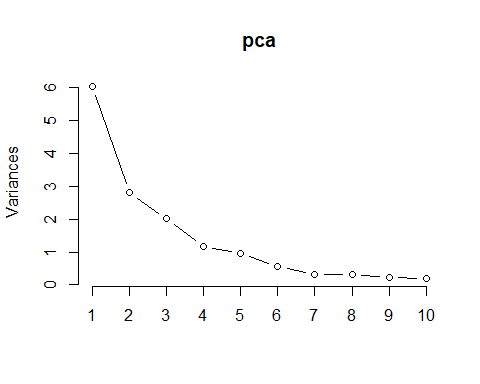
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 2.4534 1.6739 1.4160 1.07806 0.97893 0.74377  
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688  
## Cumulative Proportion 0.4013 0.5880 0.7217 0.79920 0.86308 0.89996  
## PC7 PC8 PC9 PC10 PC11 PC12  
## 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 screeplot 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(pc1,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

fold 1

Observations in test set: 9

1 3 17 18 19 22 36 38 40

Predicted 726.3 630 774 1192 1286 612 982 610.7 922

cvpred 806.4 687 828 1483 1355 638 879 606.3 935

rate 791.0 578 539 929 750 439 1272 566.0 1151

CV residual -15.4 -109 -289 -554 -605 -199 393 -40.3 216

Sum of squares = 1010591 Mean square = 112288 n = 9

fold 2

Observations in test set: 10

4 6 12 25 28 32 34 41 44 46

Predicted 1368 1216 913.8 788 781 1046 1007 843.8 965.3 1051

cvpred 1381 1282 929.4 881 817 1033 1046 906.3 982.7 1134

rate 1969 682 849.0 523 1216 754 923 880.0 1030.0 508

CV residual 588 -600 -80.4 -358 399 -279 -123 -26.3 47.3 -626

Sum of squares = 1487411 Mean square = 148741 n = 10

fold 3

Observations in test set: 10

5 8 9 11 15 23 37 39 43 47

Predicted 1014 1107 788 1236 664 926 646 739 845 878.1

cvpred 950 992 642 1090 615 831 481 629 707 942.3

rate 1234 1555 856 1674 798 1216 831 826 823 849.0

CV residual 284 563 214 584 183 385 350 197 116 -93.3

Sum of squares = 1149649 Mean square = 114965 n = 10

fold 4

Observations in test set: 9

7 13 14 20 24 27 30 35 45

Predicted 982.362 806 824 1089 758 900 743.3 1067 610

cvpred 963.673 923 865 1110 757 971 774.4 1167 665

rate 963.000 511 664 1225 968 342 696.0 653 455

CV residual -0.673 -412 -201 115 211 -629 -78.4 -514 -210

Sum of squares = 977599 Mean square = 108622 n = 9

fold 5

Observations in test set: 9

2 10 16 21 26 29 31 33 42

Predicted 927 758.2 845.0 825 1183 1535 580 790 757

cvpred 873 634.6 889.8 852 1036 1620 535 758 643

rate 1635 705.0 946.0 742 1993 1043 373 1072 542

CV residual 762 70.4 56.2 -110 957 -577 -162 314 -101

Sum of squares = 1986093 Mean square = 220677 n = 9

Overall (Sum over all 9 folds)

ms

140667

Warning message:

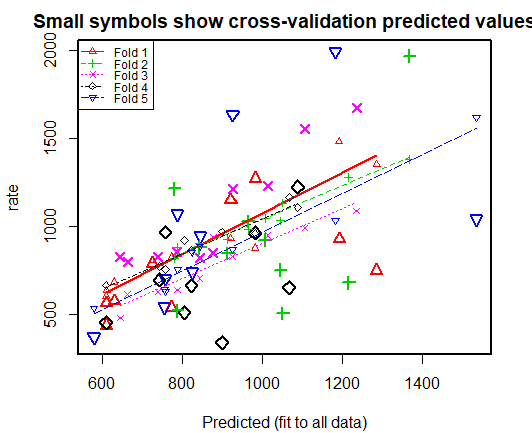
In cv.lm(PC, model2, m = 5) :

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



mn<- mean(rate)

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

R2

[1] 0.0392

**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, Ed, Po1, M.F, U1,U2,Ineq,Prob, and R-square: 0.7444, and crime rate: 1038.296,**

**My conclusion is, adding more principle components to the model may be helpful (currently we used first 4 components). Although PCA method addressed for the correlations between the predictors and ranked the coordinates by importance, it didn't address for the over fitting, 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**

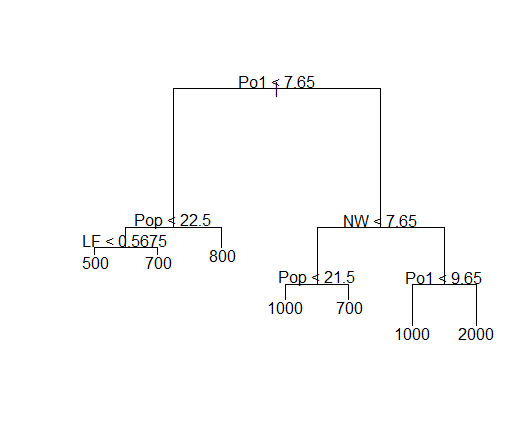
**(b) a random forest model.**

**(a) a regression tree model**

fita<-tree(Crime~.,data=crime)

plot(fita)

text(fita)



summary(fita)

Regression tree:

tree(formula = Crime ~ ., data = crime)

Variables actually used in tree construction:

[1] "Po1" "Pop" "LF" "NW"

Number of terminal nodes: 7

Residual mean deviance: 47400 = 1900000 / 40

Distribution of residuals:

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

-574 -98 -2 0 111 490

**Since we only have 47 data points for the crime data. I used the whole dataset to fit the tree model, instead of half of them.**

**From the summary, I found that only "Po1" "Pop" "LF" "NW" are used in the construction of the tree. There are 7 terminal nodes.**

**Check out how the tree was split**

fita$frame

var n dev yval splits.cutleft splits.cutright

1 Po1 47 6880928 905 <7.65 >7.65

2 Pop 23 779243 670 <22.5 >22.5

4 LF 12 243811 550 <0.5675 >0.5675

8 <leaf> 7 48519 467

9 <leaf> 5 77757 668

5 <leaf> 11 179471 800

3 NW 24 3604163 1131 <7.65 >7.65

6 Pop 10 557575 887 <21.5 >21.5

12 <leaf> 5 146391 1049

13 <leaf> 5 147771 725

7 Po1 14 2027225 1305 <9.65 >9.65

14 <leaf> 6 170828 1041

15 <leaf> 8 1124985 1503

fita$where

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

6 13 4 13 9 10 12 13 6 5 13 6 6 5 6 12 4 13 10 13 6 4 12 9 5 13 4

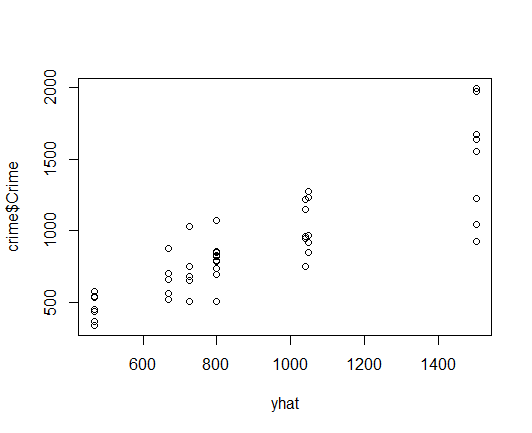
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

12 13 6 4 12 6 9 10 9 6 5 6 12 5 4 6 10 4 10 9

**Manually calculate R square to see how it fits**

yhat<-predict(fita)

plot(yhat,crime$Crime)



sse<-sum((yhat - crime$Crime) ^ 2)

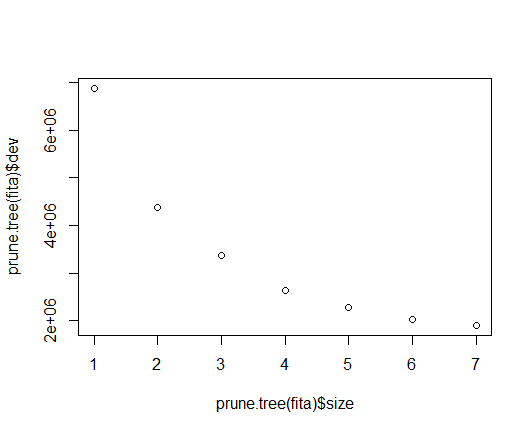
sst<-sum((crime$Crime - mean(crime$Crime)) ^ 2) #total sum of squares

1 - sse / sst

[1] 0.724

**Prune tree**

plot(prune.tree(fita)$size,prune.tree(fita)$dev)

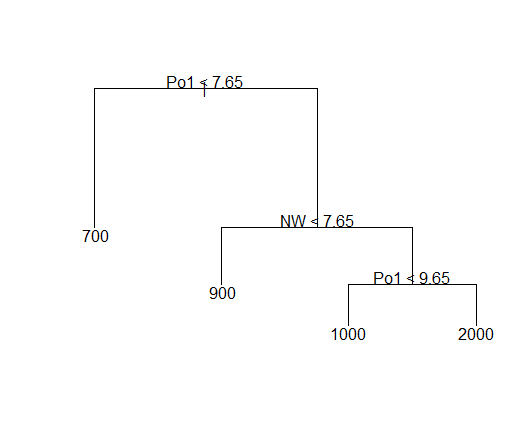
****

**Prune tree to 4 leaves is desired**

fit4<-prune.tree(fita,best=4)

plot(fit4)

text(fit4)



summary(fit4)

Regression tree:

snip.tree(tree = fita, nodes = c(6L, 2L))

Variables actually used in tree construction:

[1] "Po1" "NW"

Number of terminal nodes: 4

Residual mean deviance: 61200 = 2630000 / 43

Distribution of residuals:

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

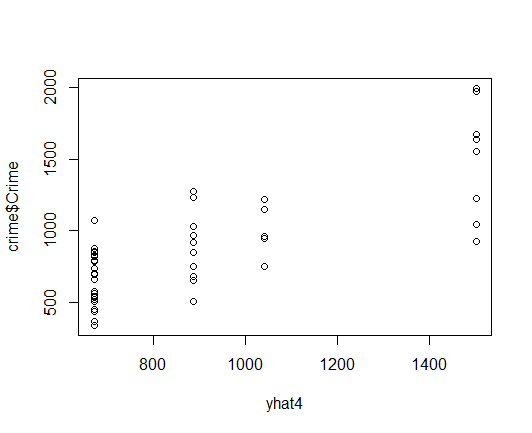
-574 -153 35 0 159 490

**Now Only po1 and NW was included, the residual mean deviance is 61220.**

**Calculate R square**

yhat4<-predict(fit4)

plot(yhat4,crime$Crime)



sse4<-sum((yhat4 - crime$Crime) ^ 2)

sst4<-sum((crime$Crime - mean(crime$Crime)) ^ 2) #total sum of squares

1 - sse4 / sst4

[1] 0.617

**The R square dropped from 0.7244962 to 0.6174017, which was expected, because we have fewer predictors left in the model.**

**Now do a cross validate on the pruned tree**

cv<-cv.tree(fit4)

cv$dev

[1] 6247077 7117073 6105515 8367604

cv$size

[1] 4 3 2 1

**The deviance becomes 7608563, even larger, indicating our model is not a good fit.**

**(b) 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

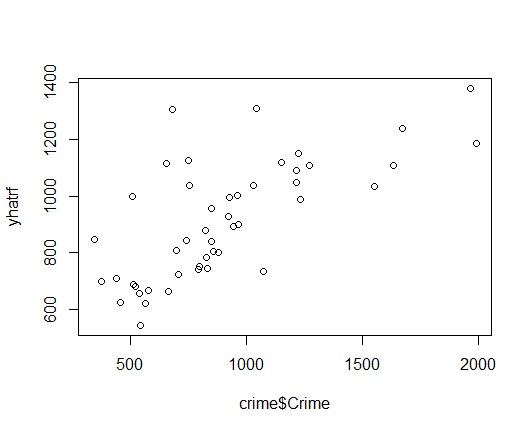
Mean of squared residuals: 79589

% Var explained: 45.6

**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] 0.456

**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 3.47 223237

So 4.01 24806

Ed 2.66 203437

Po1 12.80 1066501

Po2 10.99 964128

LF 5.14 245356

M.F 1.10 295984

Pop -1.30 403125

NW 9.21 491851

U1 2.64 143057

U2 2.23 197509

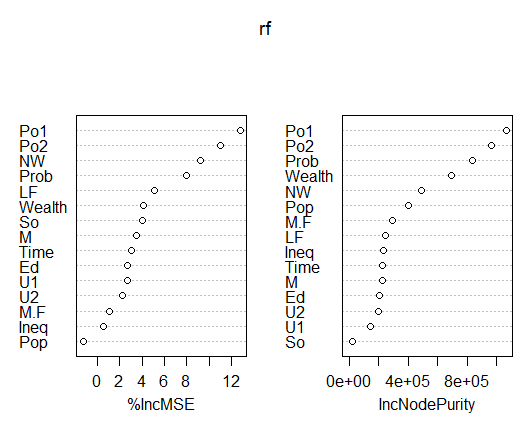
Wealth 4.11 690687

Ineq 0.52 234308

Prob 7.97 837428

Time 3.01 226674

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 likelihood of the applicant be admitted to the graduate school Predictors:**

1. **GRE score,**
2. **GPA from the undergraduate,**
3. **have related research experience or not,**
4. **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("german.data")  
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   
## -1.9894 -0.6316 -0.2844 0.5607 2.7712   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 4.145e-01 1.412e+00 0.293 0.769176   
## V1A12 -5.691e-01 2.826e-01 -2.014 0.043993 \*   
## V1A13 -9.997e-01 4.345e-01 -2.301 0.021396 \*   
## V1A14 -1.981e+00 3.044e-01 -6.509 7.57e-11 \*\*\*  
## V2 2.384e-02 1.146e-02 2.081 0.037438 \*   
## V3A31 1.183e+00 7.156e-01 1.653 0.098312 .   
## V3A32 -8.881e-02 5.584e-01 -0.159 0.873638   
## V3A33 -5.834e-01 6.099e-01 -0.957 0.338774   
## V3A34 -1.323e+00 5.695e-01 -2.322 0.020207 \*   
## V4A41 -1.779e+00 4.929e-01 -3.610 0.000307 \*\*\*  
## V4A410 6.396e-02 9.810e-01 0.065 0.948015   
## V4A42 -6.925e-01 3.418e-01 -2.026 0.042774 \*   
## V4A43 -9.077e-01 3.209e-01 -2.829 0.004669 \*\*   
## V4A44 -6.715e-01 9.159e-01 -0.733 0.463487   
## V4A45 -1.905e-01 6.888e-01 -0.277 0.782109   
## V4A46 -2.280e-01 5.014e-01 -0.455 0.649233   
## V4A48 -1.157e+00 1.351e+00 -0.857 0.391482   
## V4A49 -8.954e-02 3.941e-01 -0.227 0.820265   
## V5 1.789e-04 5.568e-05 3.213 0.001313 \*\*   
## V6A62 -1.459e-01 3.608e-01 -0.405 0.685816   
## V6A63 -5.666e-02 4.604e-01 -0.123 0.902048   
## V6A64 -7.304e-01 5.976e-01 -1.222 0.221658   
## V6A65 -1.294e+00 3.428e-01 -3.773 0.000161 \*\*\*  
## V7A72 -5.221e-01 5.568e-01 -0.938 0.348351   
## V7A73 -6.229e-01 5.344e-01 -1.166 0.243777   
## V7A74 -1.373e+00 5.814e-01 -2.361 0.018235 \*   
## V7A75 -7.513e-01 5.330e-01 -1.409 0.158691   
## V8 3.515e-01 1.146e-01 3.068 0.002153 \*\*   
## V9A92 -1.056e+00 4.911e-01 -2.150 0.031521 \*   
## V9A93 -1.287e+00 4.808e-01 -2.677 0.007431 \*\*   
## V9A94 -9.030e-01 5.949e-01 -1.518 0.129011   
## V10A102 5.670e-01 4.892e-01 1.159 0.246489   
## V10A103 -1.771e+00 6.244e-01 -2.837 0.004557 \*\*   
## V11 3.038e-02 1.111e-01 0.273 0.784591   
## V12A122 5.665e-01 3.361e-01 1.686 0.091891 .   
## V12A123 3.049e-01 3.107e-01 0.981 0.326462   
## V12A124 1.126e+00 5.160e-01 2.182 0.029089 \*   
## V13 -1.561e-02 1.175e-02 -1.329 0.183844   
## V14A142 -6.678e-01 5.323e-01 -1.255 0.209627   
## V14A143 -6.982e-01 2.960e-01 -2.359 0.018342 \*   
## V15A152 -7.042e-01 2.928e-01 -2.405 0.016165 \*   
## V15A153 -8.727e-01 5.865e-01 -1.488 0.136744   
## V16 4.489e-01 2.321e-01 1.934 0.053112 .   
## V17A172 1.107e+00 8.815e-01 1.256 0.209218   
## V17A173 1.198e+00 8.462e-01 1.416 0.156874   
## V17A174 1.201e+00 8.478e-01 1.417 0.156550   
## V18 6.828e-02 3.218e-01 0.212 0.831962   
## V19A192 -6.701e-01 2.641e-01 -2.538 0.011160 \*   
## V20A202 -1.520e+00 8.447e-01 -1.799 0.071946 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 853.51 on 699 degrees of freedom  
## Residual deviance: 571.94 on 651 degrees of freedom  
## AIC: 669.94  
##   
## Number of Fisher Scoring iterations: 6

**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   
## -2.1214 -0.6483 -0.2913 0.5920 2.7799   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.851e-01 1.040e+00 0.274 0.784106   
## V1A12 -6.247e-01 2.768e-01 -2.257 0.024021 \*   
## V1A13 -1.148e+00 4.261e-01 -2.694 0.007055 \*\*   
## V1A14 -2.013e+00 2.981e-01 -6.753 1.45e-11 \*\*\*  
## V2 2.335e-02 1.112e-02 2.100 0.035753 \*   
## V3A31 9.932e-01 6.935e-01 1.432 0.152091   
## V3A32 -1.948e-01 5.429e-01 -0.359 0.719776   
## V3A33 -6.234e-01 5.984e-01 -1.042 0.297499   
## V3A34 -1.456e+00 5.545e-01 -2.626 0.008634 \*\*   
## V4A41 -1.647e+00 4.714e-01 -3.493 0.000479 \*\*\*  
## V4A410 -1.807e-01 9.518e-01 -0.190 0.849462   
## V4A42 -5.854e-01 3.360e-01 -1.742 0.081439 .   
## V4A43 -8.615e-01 3.126e-01 -2.756 0.005845 \*\*   
## V4A44 -8.063e-01 9.435e-01 -0.855 0.392761   
## V4A45 -4.838e-01 6.711e-01 -0.721 0.470968   
## V4A46 -1.709e-01 4.955e-01 -0.345 0.730189   
## V4A48 -1.028e+00 1.286e+00 -0.799 0.424009   
## V4A49 -1.522e-01 3.865e-01 -0.394 0.693758   
## V5 1.751e-04 5.289e-05 3.310 0.000932 \*\*\*  
## V6A62 2.042e-02 3.463e-01 0.059 0.952985   
## V6A63 -7.969e-02 4.498e-01 -0.177 0.859365   
## V6A64 -7.470e-01 5.898e-01 -1.266 0.205378   
## V6A65 -1.246e+00 3.349e-01 -3.720 0.000199 \*\*\*  
## V7A72 -5.074e-02 4.710e-01 -0.108 0.914201   
## V7A73 -1.768e-01 4.423e-01 -0.400 0.689397   
## V7A74 -8.837e-01 4.950e-01 -1.785 0.074243 .   
## V7A75 -4.078e-01 4.537e-01 -0.899 0.368781   
## V8 3.368e-01 1.103e-01 3.054 0.002257 \*\*   
## V9A92 -8.548e-01 4.785e-01 -1.786 0.074028 .   
## V9A93 -1.233e+00 4.701e-01 -2.624 0.008700 \*\*   
## V9A94 -6.741e-01 5.798e-01 -1.163 0.244924   
## V10A102 6.601e-01 4.876e-01 1.354 0.175818   
## V10A103 -1.715e+00 5.998e-01 -2.859 0.004253 \*\*   
## V12A122 5.097e-01 3.257e-01 1.565 0.117623   
## V12A123 3.091e-01 2.987e-01 1.035 0.300678   
## V12A124 8.195e-01 3.813e-01 2.149 0.031622 \*   
## V14A142 -7.085e-01 5.220e-01 -1.357 0.174661   
## V14A143 -6.281e-01 2.891e-01 -2.173 0.029792 \*   
## V16 4.308e-01 2.229e-01 1.933 0.053242 .   
## V19A192 -6.423e-01 2.416e-01 -2.658 0.007859 \*\*   
## V20A202 -1.539e+00 8.367e-01 -1.839 0.065913 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 853.51 on 699 degrees of freedom  
## Residual deviance: 584.77 on 659 degrees of freedom  
## AIC: 666.77  
##   
## Number of Fisher Scoring iterations: 5

**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   
## -1.9255 -0.8605 -0.4281 0.9381 2.4280   
##   
## Coefficients: (13 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.173e+00 4.258e-01 -2.755 0.00586 \*\*   
## V1A12 -5.140e-01 2.220e-01 -2.316 0.02058 \*   
## V1A13 -1.166e+00 3.830e-01 -3.045 0.00233 \*\*   
## V1A14 -2.313e+00 2.540e-01 -9.107 < 2e-16 \*\*\*  
## V2 2.538e-02 9.340e-03 2.718 0.00657 \*\*   
## V3A34 NA NA NA NA   
## V4A41 NA NA NA NA   
## V4A42 NA NA NA NA   
## V4A43 NA NA NA NA   
## V5 9.641e-05 4.164e-05 2.316 0.02058 \*   
## V6A65 NA NA NA NA   
## V7A74 NA NA NA NA   
## V8 1.633e-01 9.208e-02 1.773 0.07620 .   
## 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 -7.674e-02 1.574e-01 -0.488 0.62581   
## 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: 853.51 on 699 degrees of freedom  
## Residual deviance: 702.82 on 692 degrees of freedom  
## AIC: 718.82  
##   
## Number of Fisher Scoring iterations: 5

**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   
## -1.9446 -0.8595 -0.4272 0.9275 2.4128   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.285e+00 3.596e-01 -3.574 0.000352 \*\*\*  
## V1A12 -5.099e-01 2.218e-01 -2.299 0.021477 \*   
## V1A13 -1.155e+00 3.820e-01 -3.023 0.002503 \*\*   
## V1A14 -2.316e+00 2.539e-01 -9.123 < 2e-16 \*\*\*  
## V2 2.545e-02 9.339e-03 2.725 0.006436 \*\*   
## V5 9.637e-05 4.162e-05 2.316 0.020582 \*   
## V8 1.633e-01 9.207e-02 1.774 0.076030 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 853.51 on 699 degrees of freedom  
## Residual deviance: 703.06 on 693 degrees of freedom  
## AIC: 717.06  
##   
## Number of Fisher Scoring iterations: 5

**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)

## 5 8 9 16 18 20   
## 0.5707649 0.5292916 0.0644921 0.5256126 0.6417318 0.1024695

**Round the yhatlog to be 0/1 variabls**

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

## [1] 1 1 0 1 1 0

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

##   
## y 0 1  
## 0 182 58  
## 1 27 33

correct<-(182+33)/300  
correct

## [1] 0.7166667

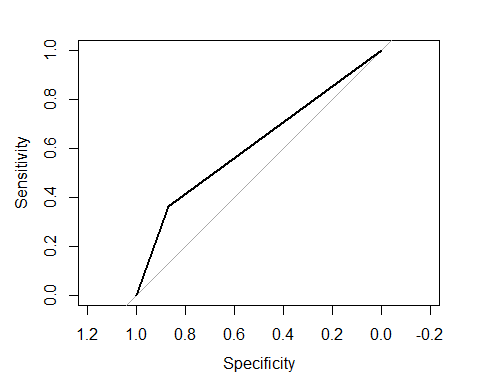
roc<-roc(test$V21,y)

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

## Setting direction: controls < cases

# Plot the ROC curve

plot(roc)



roc

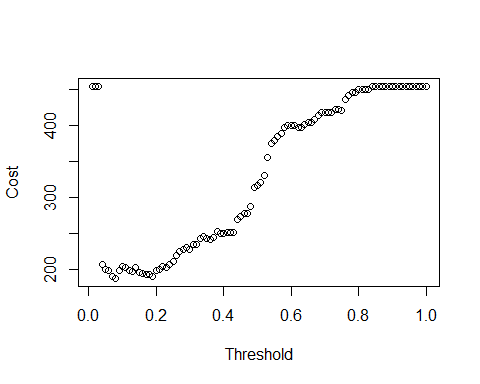
##   
## Call:  
## roc.default(response = test$V21, predictor = y)  
##   
## Data: y in 209 controls (test$V21 0) < 91 cases (test$V21 1).  
## Area under the curve: 0.6167

**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.**

**Calculating loss for the 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] 8

cost

## [1] 455 455 455 207 200 198 190 187 198 204 202 198 197 202 195 194 192  
## [18] 192 189 198 200 204 202 207 211 219 224 227 230 227 234 235 242 245  
## [35] 242 241 244 252 249 249 251 251 251 269 274 277 278 287 314 317 321  
## [52] 330 356 375 379 385 389 397 401 401 400 398 397 402 405 405 409 414  
## [69] 419 419 419 418 423 423 422 437 442 446 446 451 451 450 450 455 455  
## [86] 455 455 455 455 455 455 455 455 455 455 455 455 455 455 455

**When threshold=0.08, we have minimum cost 187.**