predictive model for credit card applicants

R Markdown

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.

```
# Clear environment
rm(list=ls())
# Read the data
data<-read.table("germancredit.txt",sep = " ")
head(data)
```

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

Since binomial family of glm recognises 0 and 1 as the classfication values, convert 1s and 2s to 0s and 1s for the response variable

```
data$V21[data$V21==1]<-0
data$V21[data$V21==2]<-1
```

set seed

```
set.seed(1)
```

Split data

```
m <- nrow(data)
trn <- sample(1:m, size = round(m*0.7), replace = FALSE)
d.learn <- data[trn,]
d.valid <- data[-trn,]
```

1st iteration: Use all the available variables

```r
reg = glm(V21 ~.,family=binomial(link = "logit"),data=d.learn)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4438  -0.6861  -0.3608   0.6750   2.4540
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.823e-01  1.332e+00   0.287 0.774162
## V1A12       -5.201e-01  2.681e-01  -1.940 0.052408 .
## V1A13       -1.150e+00  4.473e-01  -2.570 0.010173 *
## V1A14       -1.675e+00  2.750e-01  -6.091 1.12e-09 ***
## V2           2.570e-02  1.159e-02   2.217 0.026647 *
## V3A31        8.440e-02  6.580e-01   0.128 0.897943
## V3A32       -8.078e-01  4.996e-01  -1.617 0.105907
## V3A33       -7.683e-01  5.372e-01  -1.430 0.152634
## V3A34       -1.446e+00  5.127e-01  -2.821 0.004784 **
## V4A41       -1.513e+00  4.479e-01  -3.379 0.000728 ***
## V4A410      -2.412e+00  1.160e+00  -2.080 0.037543 *
## V4A42       -5.496e-01  3.195e-01  -1.720 0.085354 .
## V4A43       -9.142e-01  3.024e-01  -3.023 0.002503 **
## V4A44       -4.163e-01  9.455e-01  -0.440 0.659751
## V4A45       -1.562e-01  6.742e-01  -0.232 0.816732
## V4A46       -2.569e-01  5.085e-01  -0.505 0.613382
## V4A48       -1.531e+01  4.556e+02  -0.034 0.973202
## V4A49       -5.397e-01  4.017e-01  -1.344 0.179086
## V5           1.076e-04  5.600e-05   1.922 0.054633 .
## V6A62       -3.474e-01  3.579e-01  -0.971 0.331777
## V6A63       -2.440e-01  4.761e-01  -0.513 0.608232
## V6A64       -1.379e+00  6.535e-01  -2.110 0.034823 *
## V6A65       -8.106e-01  3.223e-01  -2.515 0.011910 *
## V7A72       -1.814e-01  5.243e-01  -0.346 0.729300
## V7A73       -5.253e-01  5.001e-01  -1.050 0.293529
## V7A74       -1.129e+00  5.455e-01  -2.070 0.038431 *
## V7A75       -5.927e-01  5.052e-01  -1.173 0.240705
## V8           3.523e-01  1.094e-01   3.219 0.001284 **
## V9A92        4.849e-02  4.760e-01   0.102 0.918863
## V9A93       -4.446e-01  4.691e-01  -0.948 0.343279
## V9A94       -4.288e-01  5.837e-01  -0.735 0.462524
## V10A102      3.052e-01  5.338e-01   0.572 0.567472
## V10A103     -3.086e-01  5.237e-01  -0.589 0.555669
## V11         -1.080e-01  1.073e-01  -1.007 0.314147
## V12A122      2.219e-01  3.161e-01   0.702 0.482767
## V12A123      3.274e-01  2.922e-01   1.120 0.262504
## V12A124      1.156e+00  5.656e-01   2.044 0.040944 *
## V13         -2.257e-02  1.140e-02  -1.980 0.047667 *
## V14A142     -5.214e-01  4.925e-01  -1.059 0.289757
```

```
## V14A143     -7.780e-01  2.848e-01  -2.732 0.006299 **
## V15A152     -6.323e-01  2.870e-01  -2.203 0.027579 *
## V15A153     -6.674e-01  6.202e-01  -1.076 0.281931
## V16          2.866e-01  2.236e-01   1.282 0.199939
## V17A172      1.565e+00  8.891e-01   1.760 0.078442 .
## V17A173      1.564e+00  8.582e-01   1.823 0.068370 .
## V17A174      1.400e+00  8.772e-01   1.596 0.110563
## V18          1.645e-01  3.004e-01   0.548 0.583871
## V19A192     -3.319e-01  2.413e-01  -1.376 0.168942
## V20A202     -2.137e+00  8.573e-01  -2.493 0.012665 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 613.21  on 651  degrees of freedom
## AIC: 711.21
##
## Number of Fisher Scoring iterations: 14
```

2nd iteration: Use all the variables found significant in the 1st iteration.

```
reg = glm(V21 ~ V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V12+V14+V16+V20,family=binomial(link = "logit"),data=d.le
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
##     V10 + V12 + V14 + V16 + V20, family = binomial(link = "logit"),
##     data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2462  -0.6956  -0.3953   0.6760   2.4277
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.303e-01  9.669e-01   0.135  0.89278
## V1A12       -5.800e-01  2.627e-01  -2.208  0.02724 *
## V1A13       -1.332e+00  4.334e-01  -3.074  0.00211 **
## V1A14       -1.710e+00  2.692e-01  -6.353 2.12e-10 ***
## V2           2.722e-02  1.137e-02   2.394  0.01665 *
## V3A31        9.884e-02  6.436e-01   0.154  0.87794
## V3A32       -7.908e-01  4.866e-01  -1.625  0.10415
## V3A33       -7.357e-01  5.249e-01  -1.402  0.16105
## V3A34       -1.483e+00  5.000e-01  -2.966  0.00302 **
## V4A41       -1.427e+00  4.381e-01  -3.257  0.00112 **
## V4A410      -2.681e+00  1.112e+00  -2.411  0.01592 *
## V4A42       -4.295e-01  3.053e-01  -1.407  0.15938
## V4A43       -8.399e-01  2.944e-01  -2.853  0.00433 **
## V4A44       -3.986e-01  9.629e-01  -0.414  0.67893
## V4A45       -3.058e-01  6.584e-01  -0.465  0.64228
```

```
## V4A46       -1.262e-01  4.882e-01  -0.258  0.79606
## V4A48       -1.528e+01  4.594e+02  -0.033  0.97348
## V4A49       -5.376e-01  3.911e-01  -1.375  0.16928
## V5           8.507e-05  5.311e-05   1.602  0.10919
## V6A62       -1.985e-01  3.434e-01  -0.578  0.56309
## V6A63       -4.018e-01  4.662e-01  -0.862  0.38883
## V6A64       -1.325e+00  6.314e-01  -2.098  0.03589 *
## V6A65       -8.503e-01  3.144e-01  -2.705  0.00683 **
## V7A72        4.714e-01  4.537e-01   1.039  0.29882
## V7A73        9.016e-02  4.253e-01   0.212  0.83212
## V7A74       -5.633e-01  4.757e-01  -1.184  0.23640
## V7A75       -2.568e-01  4.444e-01  -0.578  0.56339
## V8           3.071e-01  1.056e-01   2.908  0.00364 **
## V9A92        9.178e-02  4.517e-01   0.203  0.83898
## V9A93       -4.491e-01  4.507e-01  -0.997  0.31897
## V9A94       -3.538e-01  5.594e-01  -0.632  0.52709
## V10A102      2.701e-01  5.233e-01   0.516  0.60577
## V10A103     -3.480e-01  5.203e-01  -0.669  0.50356
## V12A122      1.943e-01  3.063e-01   0.634  0.52597
## V12A123      3.065e-01  2.814e-01   1.089  0.27607
## V12A124      7.751e-01  3.716e-01   2.086  0.03699 *
## V14A142     -5.525e-01  4.866e-01  -1.135  0.25621
## V14A143     -7.336e-01  2.805e-01  -2.615  0.00892 **
## V16          2.078e-01  2.124e-01   0.979  0.32782
## V20A202     -1.753e+00  8.231e-01  -2.130  0.03320 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 630.28  on 660  degrees of freedom
## AIC: 710.28
##
## Number of Fisher Scoring iterations: 14
```

3rd iteration: Use only the significant variables obtained in the 2nd iteration.

```r
reg = glm(V21 ~ V1+V2+V3+V4+V5+V6+V8+V9+V10+V14+V20,family=binomial(link = "logit"),data=d.learn)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 +
##     V14 + V20, family = binomial(link = "logit"), data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0705  -0.7088  -0.4083   0.7427   2.7196
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  8.340e-01  7.399e-01   1.127  0.25968
```

```
## V1A12       -4.914e-01  2.540e-01  -1.935  0.05305 .
## V1A13       -1.244e+00  4.237e-01  -2.935  0.00334 **
## V1A14       -1.697e+00  2.644e-01  -6.417 1.39e-10 ***
## V2           2.752e-02  1.104e-02   2.493  0.01265 *
## V3A31       -1.019e-01  6.117e-01  -0.167  0.86774
## V3A32       -1.000e+00  4.599e-01  -2.175  0.02963 *
## V3A33       -8.503e-01  5.180e-01  -1.641  0.10073
## V3A34       -1.599e+00  4.880e-01  -3.277  0.00105 **
## V4A41       -1.400e+00  4.273e-01  -3.277  0.00105 **
## V4A410      -2.744e+00  1.122e+00  -2.445  0.01448 *
## V4A42       -4.520e-01  2.956e-01  -1.529  0.12617
## V4A43       -8.796e-01  2.862e-01  -3.074  0.00211 **
## V4A44       -3.383e-01  9.079e-01  -0.373  0.70940
## V4A45       -1.842e-01  6.395e-01  -0.288  0.77334
## V4A46        1.022e-01  4.683e-01   0.218  0.82726
## V4A48       -1.554e+01  4.572e+02  -0.034  0.97289
## V4A49       -6.235e-01  3.831e-01  -1.627  0.10363
## V5           9.679e-05  5.174e-05   1.871  0.06139 .
## V6A62       -1.834e-01  3.313e-01  -0.554  0.57989
## V6A63       -4.272e-01  4.539e-01  -0.941  0.34656
## V6A64       -1.385e+00  6.168e-01  -2.246  0.02471 *
## V6A65       -9.440e-01  3.076e-01  -3.069  0.00215 **
## V8           3.191e-01  1.031e-01   3.095  0.00197 **
## V9A92        1.032e-01  4.404e-01   0.234  0.81479
## V9A93       -5.570e-01  4.370e-01  -1.275  0.20240
## V9A94       -3.969e-01  5.454e-01  -0.728  0.46684
## V10A102      2.805e-01  5.057e-01   0.555  0.57906
## V10A103     -6.405e-01  5.099e-01  -1.256  0.20907
## V14A142     -4.740e-01  4.749e-01  -0.998  0.31816
## V14A143     -7.766e-01  2.746e-01  -2.828  0.00468 **
## V20A202     -1.688e+00  8.147e-01  -2.071  0.03832 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 644.88  on 668  degrees of freedom
## AIC: 708.88
##
## Number of Fisher Scoring iterations: 14
```

can create a binary variable for each significant factor:

```
d.learn$V1A13[d.learn$V1 == "A13"] <- 1
d.learn$V1A13[d.learn$V1 != "A13"] <- 0

d.learn$V1A14[d.learn$V1 == "A14"] <- 1
d.learn$V1A14[d.learn$V1 != "A14"] <- 0

d.learn$V3A32[d.learn$V3 == "A32"] <- 1
d.learn$V3A32[d.learn$V3 != "A32"] <- 0

d.learn$V3A33[d.learn$V3 == "A33"] <- 1
```

```
d.learn$V3A33[d.learn$V3 != "A33"] <- 0

d.learn$V3A34[d.learn$V3 == "A34"] <- 1
d.learn$V3A34[d.learn$V3 != "A34"] <- 0

d.learn$V4A41[d.learn$V4 == "A41"] <- 1
d.learn$V4A41[d.learn$V4 != "A41"] <- 0

d.learn$V4A410[d.learn$V4 == "A410"] <- 1
d.learn$V4A410[d.learn$V4 != "A410"] <- 0

d.learn$V4A42[d.learn$V4 == "A42"] <- 1
d.learn$V4A42[d.learn$V4 != "A42"] <- 0

d.learn$V4A43[d.learn$V4 == "A43"] <- 1
d.learn$V4A43[d.learn$V4 != "A43"] <- 0

d.learn$V4A48[d.learn$V4 == "A48"] <- 1
d.learn$V4A48[d.learn$V4 != "A48"] <- 0

d.learn$V4A49[d.learn$V4 == "A49"] <- 1
d.learn$V4A49[d.learn$V4 != "A49"] <- 0

d.learn$V6A63[d.learn$V6 == "A63"] <- 1
d.learn$V6A63[d.learn$V6 != "A63"] <- 0

d.learn$V6A65[d.learn$V6 == "A65"] <- 1
d.learn$V6A65[d.learn$V6 != "A65"] <- 0

d.learn$V9A93[d.learn$V9 == "A93"] <- 1
d.learn$V9A93[d.learn$V9 != "A93"] <- 0

d.learn$V10A103[d.learn$V10 == "A103"] <- 1
d.learn$V10A103[d.learn$V10 != "A103"] <- 0

d.learn$V14A143[d.learn$V14 == "A143"] <- 1
d.learn$V14A143[d.learn$V14 != "A143"] <- 0

d.learn$V20A202[d.learn$V20 == "A202"] <- 1
d.learn$V20A202[d.learn$V20 != "A202"] <- 0
```

Next round model:

```
reg = glm(V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 + V4A41 + V4A410 + V4A42 + V4A43 + V4A48 + V4
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 +
##     V4A41 + V4A410 + V4A42 + V4A43 + V4A48 + V4A49 + V5 + V6A63 +
##     V6A65 + V8 + V9A93 + V10A103 + V14A143 + V20A202, family = binomial(link = "logit"),
##     data = d.learn)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1416  -0.7391  -0.4092   0.8315   2.6916
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.501e-01  5.069e-01   0.493 0.621773
## V1A13       -9.234e-01  3.973e-01  -2.324 0.020133 *
## V1A14       -1.461e+00  2.286e-01  -6.390 1.65e-10 ***
## V2           3.049e-02  1.077e-02   2.832 0.004619 **
## V3A32       -9.343e-01  3.323e-01  -2.812 0.004927 **
## V3A33       -9.099e-01  4.162e-01  -2.186 0.028806 *
## V3A34       -1.521e+00  3.689e-01  -4.124 3.73e-05 ***
## V4A41       -1.404e+00  4.102e-01  -3.423 0.000618 ***
## V4A410      -2.728e+00  1.127e+00  -2.420 0.015507 *
## V4A42       -3.584e-01  2.689e-01  -1.333 0.182619
## V4A43       -9.171e-01  2.604e-01  -3.522 0.000428 ***
## V4A48       -1.546e+01  4.582e+02  -0.034 0.973083
## V4A49       -7.429e-01  3.575e-01  -2.078 0.037707 *
## V5           1.023e-04  5.052e-05   2.025 0.042853 *
## V6A63       -3.822e-01  4.490e-01  -0.851 0.394650
## V6A65       -8.648e-01  2.937e-01  -2.944 0.003237 **
## V8           2.993e-01  1.001e-01   2.991 0.002779 **
## V9A93       -4.985e-01  2.018e-01  -2.470 0.013496 *
## V10A103     -6.012e-01  4.911e-01  -1.224 0.220880
## V14A143     -6.059e-01  2.383e-01  -2.542 0.011009 *
## V20A202     -1.548e+00  8.046e-01  -1.924 0.054323 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 659.89  on 679  degrees of freedom
## AIC: 701.89
##
## Number of Fisher Scoring iterations: 14
```

Remove V4A48 and V6A63 (p-value above 0.05) and V20A202 (p-value above 0.1)

```r
reg = glm(V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 + V4A41 + V4A410 + V4A42 + V4A43 + V4A49 + V5
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 +
##     V4A41 + V4A410 + V4A42 + V4A43 + V4A49 + V5 + V6A65 + V8 +
##     V9A93 + V10A103 + V14A143, family = binomial(link = "logit"),
##     data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0847  -0.7664  -0.4327   0.8550   2.7041
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.1351555  0.4858104  -0.278 0.780854
## V1A13       -0.9029019  0.3932036  -2.296 0.021660 *
## V1A14       -1.5015986  0.2253411  -6.664 2.67e-11 ***
## V2           0.0334993  0.0107055   3.129 0.001753 **
## V3A32       -0.7390892  0.3185290  -2.320 0.020324 *
## V3A33       -0.7000787  0.4063906  -1.723 0.084947 .
## V3A34       -1.3376698  0.3556626  -3.761 0.000169 ***
## V4A41       -1.3342616  0.4076872  -3.273 0.001065 **
## V4A410      -3.0301355  1.1707945  -2.588 0.009651 **
## V4A42       -0.2896628  0.2660379  -1.089 0.276241
## V4A43       -0.8480179  0.2569246  -3.301 0.000965 ***
## V4A49       -0.6363316  0.3527955  -1.804 0.071281 .
## V5           0.0001033  0.0000503   2.054 0.040004 *
## V6A65       -0.8717463  0.2882645  -3.024 0.002494 **
## V8           0.3043349  0.0981857   3.100 0.001938 **
## V9A93       -0.4928265  0.1992747  -2.473 0.013395 *
## V10A103     -0.6750464  0.4855160  -1.390 0.164417
## V14A143     -0.5883351  0.2360064  -2.493 0.012671 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 673.15  on 682  degrees of freedom
## AIC: 709.15
##
## Number of Fisher Scoring iterations: 5
```

Validation

add the binary variables to the validation set

```
d.valid$V1A13[d.valid$V1 == "A13"] <- 1
d.valid$V1A13[d.valid$V1 != "A13"] <- 0

d.valid$V1A14[d.valid$V1 == "A14"] <- 1
d.valid$V1A14[d.valid$V1 != "A14"] <- 0

d.valid$V3A32[d.valid$V3 == "A32"] <- 1
d.valid$V3A32[d.valid$V3 != "A32"] <- 0

d.valid$V3A33[d.valid$V3 == "A33"] <- 1
d.valid$V3A33[d.valid$V3 != "A33"] <- 0

d.valid$V3A34[d.valid$V3 == "A34"] <- 1
d.valid$V3A34[d.valid$V3 != "A34"] <- 0

d.valid$V4A41[d.valid$V4 == "A41"] <- 1
```

```r
d.valid$V4A41[d.valid$V4 != "A41"] <- 0

d.valid$V4A410[d.valid$V4 == "A410"] <- 1
d.valid$V4A410[d.valid$V4 != "A410"] <- 0

d.valid$V4A42[d.valid$V4 == "A42"] <- 1
d.valid$V4A42[d.valid$V4 != "A42"] <- 0

d.valid$V4A43[d.valid$V4 == "A43"] <- 1
d.valid$V4A43[d.valid$V4 != "A43"] <- 0

d.valid$V4A49[d.valid$V4 == "A49"] <- 1
d.valid$V4A49[d.valid$V4 != "A49"] <- 0

d.valid$V6A65[d.valid$V6 == "A65"] <- 1
d.valid$V6A65[d.valid$V6 != "A65"] <- 0

d.valid$V9A93[d.valid$V9 == "A93"] <- 1
d.valid$V9A93[d.valid$V9 != "A93"] <- 0

d.valid$V10A103[d.valid$V10 == "A103"] <- 1
d.valid$V10A103[d.valid$V10 != "A103"] <- 0

d.valid$V14A143[d.valid$V14 == "A143"] <- 1
d.valid$V14A143[d.valid$V14 != "A143"] <- 0
```

test the model

```r
y_hat<-predict(reg,d.valid,type = "response")
y_hat # y_hat is a vector of fractions.
```

```
##           2          10          14          18          21          23
## 0.627317196 0.668583287 0.399195562 0.577216145 0.089779614 0.156674392
##          24          26          32          34          38          46
## 0.084175959 0.267081411 0.542683307 0.021588444 0.253363660 0.137876222
##          47          50          53          57          59          63
## 0.161888689 0.121452967 0.054183457 0.119579175 0.343680447 0.684038387
##          64          68          70          74          75          76
## 0.864543423 0.564824497 0.117808468 0.627493638 0.450257208 0.108003848
##          78          88          90          94          95          96
## 0.215100702 0.702161608 0.626553803 0.097671327 0.450181525 0.925276429
##          98         106         107         113         114         120
## 0.377404059 0.050288652 0.437609720 0.551612740 0.564696051 0.377123133
##         123         125         131         136         142         144
## 0.095431223 0.353216696 0.611734251 0.028329559 0.647484024 0.314511972
##         147         149         151         152         154         155
## 0.187399134 0.300495484 0.059078532 0.030267006 0.234794172 0.546917897
##         156         157         160         163         165         170
## 0.352250420 0.132217142 0.006218601 0.167030259 0.280940966 0.383126014
##         171         172         174         178         186         188
## 0.646031097 0.174264705 0.136215770 0.124688953 0.045507687 0.216198096
##         189         191         195         196         199         200
## 0.349406686 0.099170671 0.391601135 0.269737100 0.219827398 0.634353022
```

```
##         202         204         210         212         221         223
## 0.568907592 0.562156828 0.034545566 0.077760158 0.328843411 0.116672042
##         224         227         228         230         231         237
## 0.069322172 0.696225409 0.546526942 0.665934201 0.299819900 0.306099629
##         238         239         244         246         249         253
## 0.667312629 0.065491718 0.059429586 0.125420243 0.177047211 0.709942249
##         254         255         262         263         269         272
## 0.133977110 0.119521348 0.469114607 0.311164747 0.559243925 0.083737861
##         274         275         289         292         297         298
## 0.583693172 0.747631986 0.222091594 0.421699517 0.018325375 0.066391830
##         301         314         317         318         319         321
## 0.083981810 0.551608349 0.137486630 0.256116769 0.085971392 0.645666043
##         322         323         331         332         337         344
## 0.477740599 0.171300622 0.142788029 0.175101108 0.151296485 0.345795892
##         347         351         352         364         366         370
## 0.088417034 0.097028886 0.259732820 0.071352413 0.051008862 0.442602048
##         372         373         374         376         379         385
## 0.059488910 0.047814498 0.425681013 0.726226458 0.875100805 0.119264126
##         387         388         391         394         398         400
## 0.060238769 0.580951831 0.172997413 0.099597334 0.502670973 0.096837804
##         401         405         409         415         416         417
## 0.104887955 0.338861044 0.146657475 0.457513166 0.048717387 0.482585292
##         427         429         430         432         438         443
## 0.078351726 0.047190710 0.325944192 0.140098691 0.196470671 0.056894049
##         445         447         452         456         458         459
## 0.596808588 0.759692972 0.066316986 0.119160930 0.164526434 0.497796134
##         460         461         463         475         486         491
## 0.089928296 0.261943487 0.468637077 0.276827384 0.407772509 0.024288195
##         496         497         502         503         505         517
## 0.288665263 0.752385989 0.287482077 0.100921249 0.664500568 0.167459266
##         522         523         527         528         531         534
## 0.316599951 0.851993339 0.156475719 0.017885989 0.598292149 0.160372551
##         535         540         541         542         550         559
## 0.055476187 0.300576932 0.190645335 0.233834615 0.041886466 0.762463534
##         561         566         569         575         578         580
## 0.223462542 0.402709510 0.392010559 0.236343868 0.074573098 0.300257539
##         583         585         588         589         595         596
## 0.169153331 0.119594756 0.237281473 0.618562824 0.085624360 0.614516961
##         599         603         612         617         623         624
## 0.122500625 0.888539936 0.168251160 0.482084064 0.342013679 0.514746652
##         629         632         633         636         637         640
## 0.213587464 0.743281260 0.141099367 0.248719386 0.161278666 0.644575546
##         641         655         656         659         660         666
## 0.607118412 0.042039246 0.316110374 0.665926621 0.273893574 0.021375924
##         667         669         671         676         678         681
## 0.841012361 0.323277282 0.072600884 0.151800452 0.783797646 0.091124139
##         683         685         688         691         692         695
## 0.212092664 0.493931285 0.690517113 0.242512271 0.555933324 0.114276603
##         696         698         700         706         707         709
## 0.020756255 0.059480441 0.280555195 0.166945533 0.776112754 0.339125178
##         714         717         723         726         728         730
## 0.146087103 0.028207923 0.570426880 0.105369689 0.572597917 0.113443019
##         738         739         741         745         746         747
## 0.503250307 0.054583044 0.735911008 0.567825019 0.255807684 0.402179726
```

```
##         748         750         753         754         759         760 
## 0.496002466 0.033335195 0.268669768 0.169051469 0.070731038 0.296654185 
##         763         769         774         776         778         779 
## 0.277798893 0.146325192 0.112612485 0.457256814 0.309412324 0.084971713 
##         782         784         785         786         793         794 
## 0.137776957 0.696658896 0.116481579 0.361023946 0.016419677 0.345373511 
##         795         797         800         804         809         815 
## 0.196381227 0.077395547 0.240795856 0.017036626 0.669940169 0.781757447 
##         826         830         835         837         840         844 
## 0.506283905 0.522714574 0.134472252 0.048656242 0.107190962 0.358114194 
##         850         861         863         864         865         866 
## 0.372262133 0.059021303 0.490664191 0.237289711 0.120638654 0.128613698 
##         875         877         881         886         888         890 
## 0.296589391 0.559806595 0.093879450 0.559434644 0.776203680 0.086946681 
##         892         893         896         897         899         903 
## 0.047658861 0.469518993 0.119988292 0.469887528 0.075534004 0.033414929 
##         908         914         915         926         927         933 
## 0.465960794 0.062594825 0.621325376 0.743328902 0.396003872 0.062113898 
##         934         944         945         949         953         963 
## 0.038039281 0.040596501 0.373580299 0.218373728 0.341859944 0.072698811 
##         967         969         970         971         979         986 
## 0.265048770 0.126959126 0.186290586 0.319897963 0.243328175 0.381549196 
##         989         991         996         997         999        1000 
## 0.356912067 0.027965023 0.146849048 0.456081348 0.527959836 0.269903006
```

use a threshold to make yes/no decisions,and view the confusion matrix.

```
y_hat_round <- as.integer(y_hat > 0.5)

t <- table(y_hat_round,d.valid$V21)
t
```

```
## 
## y_hat_round   0   1
##           0 183  47
##           1  25  45
```

Model's accuracy is (183 + 43) / (183 + 43 + 22 + 52) = 75%.

```
acc <- (t[1,1] + t[2,2]) / sum(t)
acc
```

```
## [1] 0.76
```

ROC curve Develop ROC curve to determine the quality of fit

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.2
```

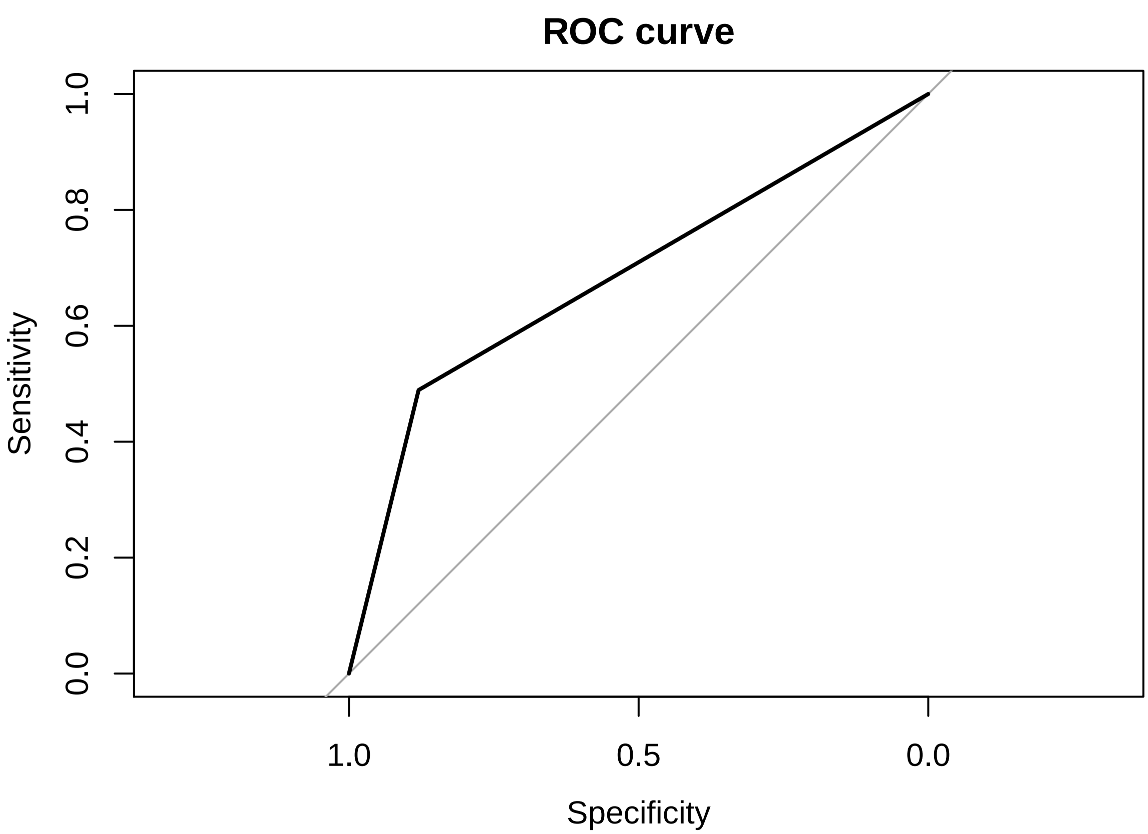
```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

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

```
r<-roc(d.valid$V21,y_hat_round)
```

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

## Setting direction: controls < cases
```

```
plot(r,main="ROC curve")
```




```
r
```

```
##
## Call:
## roc.default(response = d.valid$V21, predictor = y_hat_round)
##
## Data: y_hat_round in 208 controls (d.valid$V21 0) < 92 cases (d.valid$V21 1).
## Area under the curve: 0.6845
```

The area under the curve is 67%. This means that whenever a sample is chosen from the response group and another sample is chosen from the non-response group, then the model will correctly classify both the samples 67% of the times.

```
# try more
acc <- c()
auc <- c()
for (i in 1:9) {
  y_hat_round <- as.integer(y_hat > i/10)
  t <- table(y_hat_round,d.valid$V21)
  acc <- cbind(acc,(t[1,1] + t[2,2]) / sum(t))
  r<-roc(d.valid$V21,y_hat_round)
  auc <- cbind(auc,r$auc)
}
```

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

```r
acc
```

```
##           [,1]      [,2] [,3]      [,4] [,5]      [,6]      [,7]      [,8]
## [1,] 0.5166667 0.6633333 0.72 0.7466667 0.76 0.7766667 0.7466667 0.7066667
##           [,9]
## [1,] 0.6966667
```

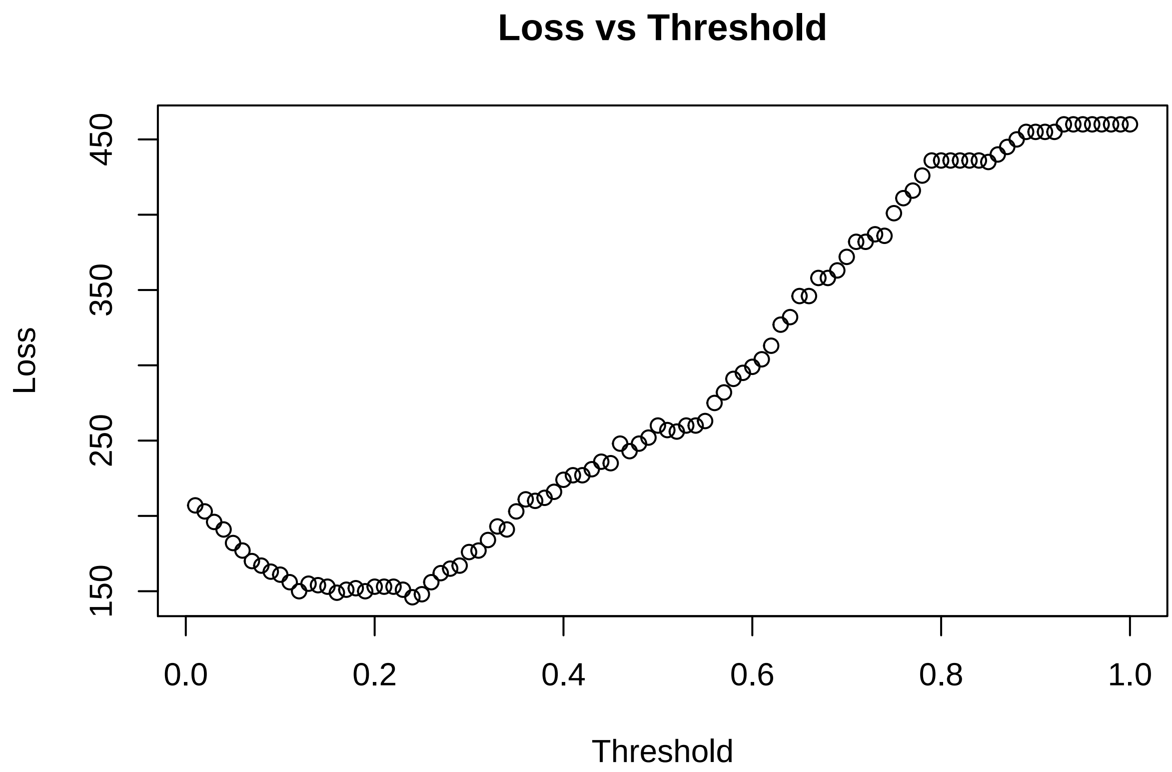
```r
auc
```

```
##           [,1]      [,2]      [,3]     [,4]      [,5]     [,6]      [,7]
## [1,] 0.6393186 0.7178094 0.7283654 0.705163 0.6844691 0.663148 0.5930184
##           [,8]      [,9]
## [1,] 0.5247701 0.5054348
```

So if we're just looking for the highest accuracy, a threshold of 0.5 looks good. If we're judging by AUC, a smaller threshold (0.2 or 0.3) is slightly better. but not by much.

The loss of incorrectly classfying a "bad" customer is 5 times the loss of incorrectly classifying a "good" customer. calulating loss for the value of thresholds ranging from 0.01 to 1.

```r
loss <- c()
for(i in 1:100)
{
  y_hat_round <- as.integer(y_hat > (i/100)) # calculate threshold predictions

  tm <-as.matrix(table(y_hat_round,d.valid$V21))

  if(nrow(tm)>1) { c1 <- tm[2,1] } else { c1 <- 0 }
  if(ncol(tm)>1) { c2 <- tm[1,2] } else { c2 <- 0 }
  loss <- c(loss, c2*5 + c1)
}

plot(c(1:100)/100,loss,xlab = "Threshold",ylab = "Loss",main = "Loss vs Threshold")
```




```
which.min(loss)
```

```
## [1] 24
```

```
loss
```

```
##   [1] 207 203 196 191 182 177 170 167 163 161 156 150 155 154 153 149 151 152
##  [19] 150 153 153 153 151 146 148 156 162 165 167 176 177 184 193 191 203 211
##  [37] 210 212 216 224 227 227 231 236 235 248 243 248 252 260 257 256 260 260
##  [55] 263 275 282 291 295 299 304 313 327 332 346 346 358 358 363 372 382 382
##  [73] 387 386 401 411 416 426 436 436 436 436 436 436 435 440 445 450 455 455
##  [91] 455 455 460 460 460 460 460 460 460 460
```

The threshold probability corresponding to minimum expected loss is 0.13.
The range from 0.07-0.14 is all pretty good. The expected loss at 0.13 is 165 over the 300 validation data points. That compares to 282 for a threshold of 0.5.So accounting for the situation is important.

```
#Here's the accuracy and area-under-curve for the 0.13 threshold:
y_hat_round <- as.integer(y_hat > (which.min(loss)/100)) # find 0/1 predictions
t <- table(y_hat_round,d.valid$V21) # put in table form
acc <- (t[1,1] + t[2,2]) / sum(t) # calculate accuracy
r<-roc(d.valid$V21,y_hat_round) # calculate ROC curve
```

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

```
## Setting direction: controls < cases
```

```r
auc <- r$auc # get AUC
acc
```

```
## [1] 0.7
```

```r
auc
```

```
## Area under the curve: 0.7412
```