GT Introduction to Analytics Modeling - Week 4 HW

Robert Phillips

June 7, 2017

Question 1

Using the same crime data set as in Homework 3 Question 4, apply Principal Component Analysis and then create a regression model using the first 4 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 Homework 3 Question 4. 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.)

We first load the data, and then run PCA on all columns except for the Crime column since Crime is the response. We can then output the principal components to inspect the results.

```
crimes = read.csv('uscrime.txt', header=T, sep='\t')
crimes.pca = prcomp(crimes[1:ncol(crimes)-1], scale=T, center=T)
summary(crimes.pca)
```

```
## Importance of components%s:
##                           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
```

```
#proportion of variance for first 4
crimes.pca.4 = sum(crimes.pca$sdev[1:4]^2)/sum(crimes.pca$sdev^2)
```

We can see that 79.9196788% of the variance is captured by the first 4 principal components.

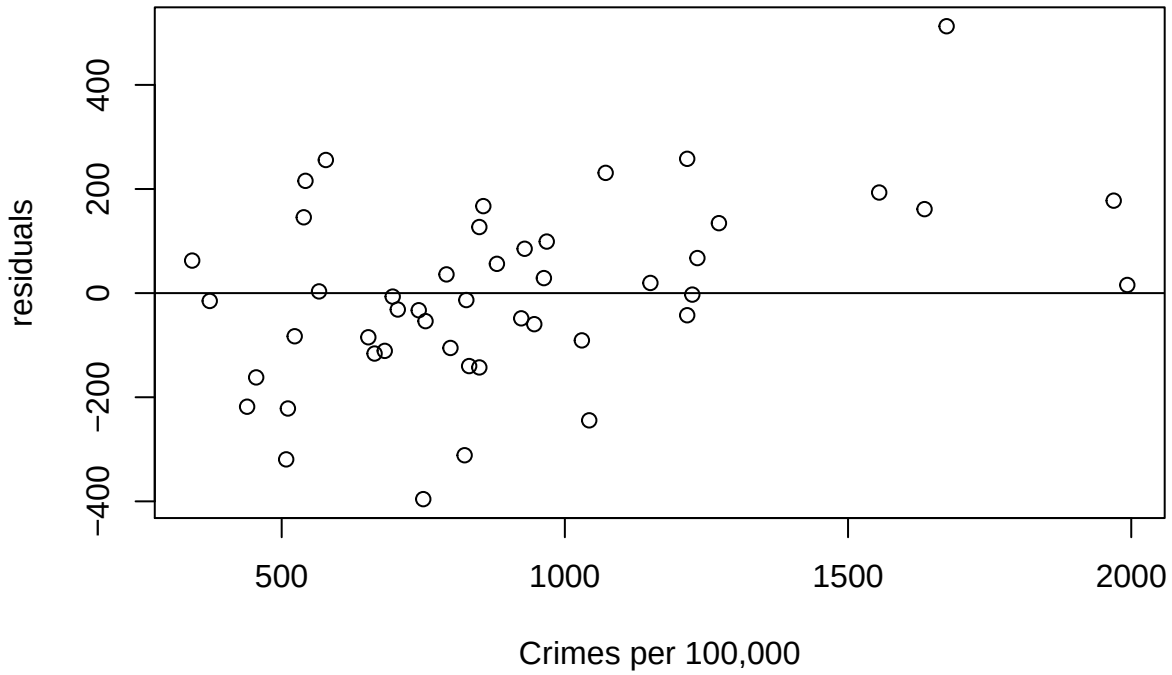
We need our model from the previous homework for comparison. I used all factors in that model, so we'll repeat that here.

```
crimes.fit = lm(Crime ~ ., data=crimes)
summary(crimes.fit)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = crimes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

```
plot(crimes$Crime, resid(crimes.fit), ylab="residuals", xlab="Crimes per 100,000")
abline(0, 0)
```



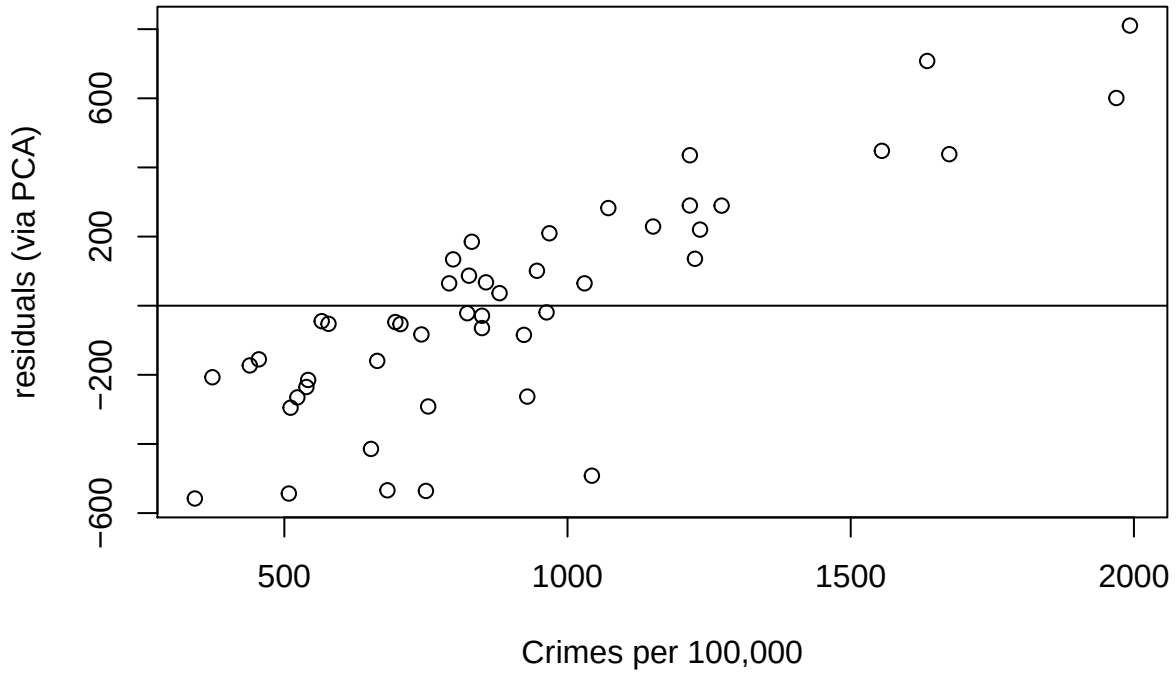
We'll now create a linear regression model using these 4 principal components.

```
crimes.pca.data = data.frame(crimes.pca$x[, 1:4], crime=crimes$Crime)
crimes.pca.fit = lm(crime ~ ., data=crimes.pca.data)
summary(crimes.pca.fit)
```

```
##
## Call:
```

```
## lm(formula = crime ~ ., data = crimes.pca.data)
##
## 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
```

```
plot(crimes$Crime, resid(crimes.pca.fit), ylab="residuals (via PCA)", xlab="Crimes per 100,000")
abline(0, 0)
```



This plot shows a trend in the residuals which is often a sign a linear model may be problemattic. Regardless, we can use the 4 non-intercept coefficents to reclaim the coefficients in the original model.

```
crimes.orig = crimes.pca.fit$coefficients[2:5] %*% t(crimes.pca$rotation[,1:4])
crimes.orig
```

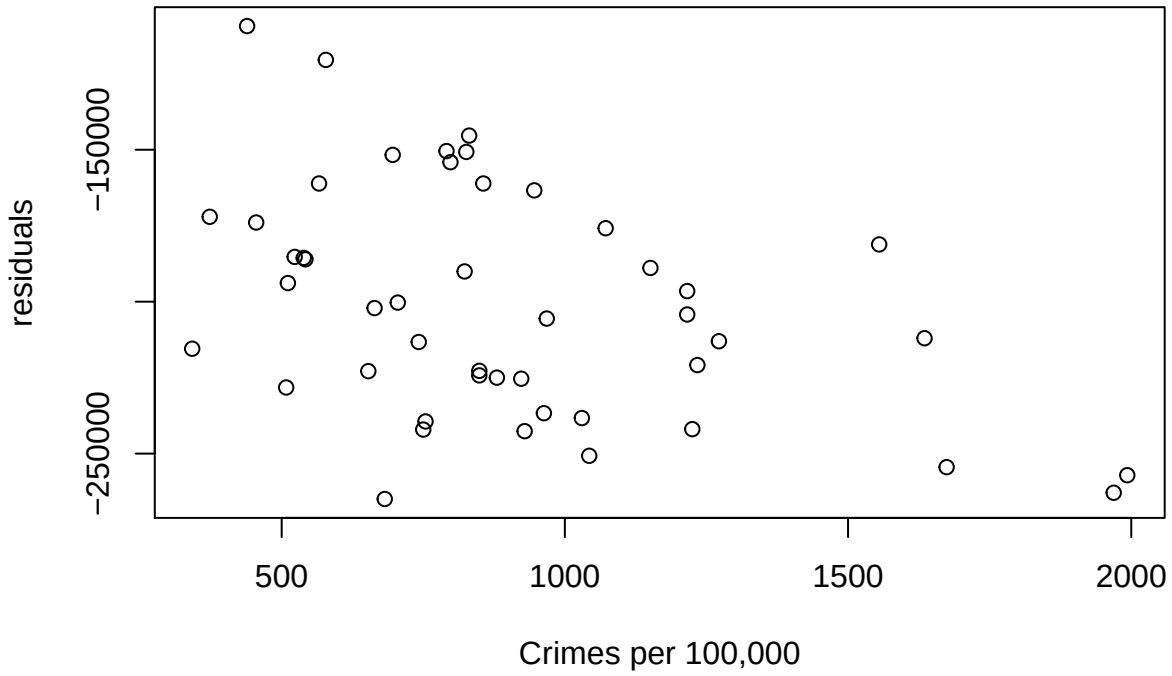
```
##              M       So       Ed      Po1      Po2        LF       M.F
## [1,] -21.27796 10.22309 14.35261 63.45643 64.55797 -14.00535 -24.43757
##           Pop       NW        U1       U2   Wealth      Ineq     Prob
## [1,] 39.83067 15.43455 -27.22228 1.425902 38.60786 -27.53635 3.295707
##           Time
```

```
## [1,] -6.612616
```

We can multiply these by the original crimes values to determine predictions, from which we can compute residuals.

```
crimes.predict = as.matrix(crimes[1:ncol(crimes)-1]) %*% t(crimes.orig)
residuals = crimes$Crime - crimes.predict

plot(crimes$Crime, residuals, ylab="residuals", xlab="Crimes per 100,000")
abline(0, 0)
```



These residuals seem worse. Seems like I am missing a step somewhere.

Question 2

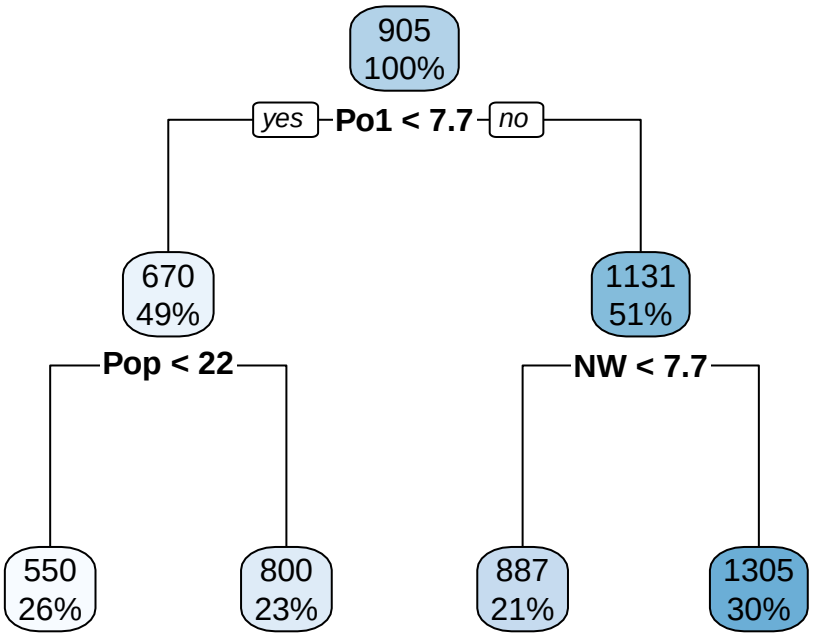
Using the same crime data set as in Homework 3 Question 4, find the best model you can using (a) a regression tree model, and (b) a random forest model. In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).

```
require(rpart)
require(rpart.plot)
require(randomForest)
```

Part a

For part (a) we build a regression tree using rpart.

```
crimes.tree = rpart(Crime ~ ., data=crimes, method="anova")
rpart.plot(crimes.tree)
```



This plot shows that we split on the 3 variables named **Po1**, **Pop**, **NW**. One the left side of the tree, the model predicts cities with an expenditure less than 7.7 and a population less than 22 (in hundred thousands) will yield 550 offenses (per 100,000). However, within this branch, with a population greater than 22 (in hundred thousands) predicts 800 offenses. This increase indicates that population is is signficant.

On the right hand side, where there are greater expenditures, a population with an NW value less than 7.7% of the poulation will yield 887 offenses (per 100,000). However, an NW value greather than 7.7% yeilds 1305 offenses. This increase indicates that the NW attribute is signficant.

Part b

For part (b) we build a random forest model with 500 trees.

```
set.seed(1)
crimes.rf = randomForest(Crime ~ ., data=crimes, ntree=500, type="regression")
crimes.rf.predict = predict(crimes.rf, newdata=crimes, type="response")
crimes.rf.mse = mean((crimes.rf.predict - crimes$Crime)^2)
```

This model gives a mean squared error `1.4983371\times 10^{4}`. We can tune the parameters to attempt to improve the MSE. For example, we can limit the number of variables used in each tree and also increase the number of trees.

```
set.seed(1)
crimes.rf2 = randomForest(Crime ~ ., data=crimes, ntree=1000, type="regression", mtry=5)
crimes.rf2.predict = predict(crimes.rf2, newdata=crimes, type="response")
crimes.rf2.mse = mean((crimes.rf2.predict - crimes$Crime)^2)
```

This model gives a mean squared error `1.5372387\times 10^{4}`. This doesn't to improve the MSE metric. Addtional approaches could be applied to continue to tune other parametetrs.

Question 3

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.

Answer: An appealing aspect of logistic regression is interpreting it as a probability. This can be especially useful in a binary classification scenario (i.e. deciding whether or not to take action). For example, given customer survey data and even unstructured data such as written feedback, we can build a model to determine whether or not an observation represents positive sentiment. We would adjust a threshold property to determine an appropriate value (probability) that indicates positive sentiment. An ROC curve would help do this. We would just need to decide on the impact of false negatives and false positives.

Regarding predictors, we'd likely capture demographic data as well as a number of questions related to sentiment. For example, questions related to satisfaction, quantification of interaction with the service, as well as others.

Question 4

1. *Using the GermanCredit data set at http://archive.ics.uci.edu/ml/machine-learningdatabases/statlog/german/ 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.*
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.*

Part 1

We first load the data. We'll transform the response column 'V21' to 0 and 1 since that is what the *glm* function requires. We'll also split the data into a training and test set to aid in selecting a *good* model.

```
#load the file
credit = read.csv('germancredit.txt', header=F, sep=' ')
#update the response column to 0 = Bad
credit$V21 = sapply(credit$V21, function(x){if (2 == x) 0 else x})

#use randomization to split the data into training and test data
set.seed(1)

m <- dim(credit)[1]
#sample 80% for training
train <- sample(1:m, size = round(m*.8), replace=F)

credit.train <- credit[train,]
credit.test <- credit[-train,]
```

The data consists of numerical and categorical values. Using all of these attributes may lead to a model that is difficult to interpret. But removing attributes may require some domain knowledge. So we can start by training a model on all of the parameterts to determine which are indicated to have signficance.

- Attribute 1: (qualitative) - Status of existing checking account
- Attribute 2: (numerical) - Duration in month
- Attribute 3: (qualitative) - Credit history
- Attribute 4: (qualitative) - Purpose
- Attribute 5: (numerical) - Credit amount
- Attribute 6: (qualitative) - Savings account/bonds
- Attribute 7: (qualitative) - Present employment since

- Attribute 8: (numerical) - Installment rate in percentage of disposable income
- Attribute 9: (qualitative) - Personal status and sex
- Attribute 10: (qualitative) - Other debtors / guarantors
- Attribute 11: (numerical) - Present residence since
- Attribute 12: (qualitative) - Property
- Attribute 13: (numerical) - Age in years
- Attribute 14: (qualitative) - Other installment plans
- Attribute 15: (qualitative) - Housing
- Attribute 16: (numerical) - Number of existing credits at this bank
- Attribute 17: (qualitative) - Job
- Attribute 18: (numerical) - Number of people being liable to provide maintenance for
- Attribute 19: (qualitative) - Telephone
- Attribute 20: (qualitative) - foreign worker
- Attribute 21: (qualitative) - 1 = Good Loan, 2 = Bad Loan

```
credit.train.fit <- glm(V21 ~.,family=binomial(link='logit'),data=credit.train)
summary(credit.train.fit)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = credit.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7244  -0.6431   0.3593   0.6824   2.1598
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.013e+00  1.277e+00  -0.793 0.427550
## V1A12        2.968e-01  2.516e-01   1.179 0.238262
## V1A13        9.400e-01  4.366e-01   2.153 0.031328 *
## V1A14        1.561e+00  2.661e-01   5.866 4.46e-09 ***
## V2          -3.219e-02  1.079e-02  -2.984 0.002842 **
## V3A31        4.462e-01  6.375e-01   0.700 0.483985
## V3A32        8.878e-01  5.020e-01   1.769 0.076957 .
## V3A33        1.350e+00  5.330e-01   2.534 0.011290 *
## V3A34        1.911e+00  5.187e-01   3.683 0.000230 ***
## V4A41        1.958e+00  4.564e-01   4.290 1.79e-05 ***
## V4A410       1.815e+00  8.542e-01   2.125 0.033603 *
## V4A42        8.574e-01  2.980e-01   2.877 0.004011 **
## V4A43        9.136e-01  2.816e-01   3.244 0.001178 **
## V4A44        3.326e-01  7.972e-01   0.417 0.676560
## V4A45        5.729e-01  7.362e-01   0.778 0.436487
## V4A46       -3.958e-01  4.547e-01  -0.871 0.383984
## V4A48        1.986e+00  1.210e+00   1.641 0.100814
## V4A49        7.449e-01  3.798e-01   1.961 0.049879 *
## V5          -1.016e-04  5.168e-05  -1.966 0.049303 *
## V6A62        3.327e-01  3.217e-01   1.034 0.300982
## V6A63        8.453e-01  5.010e-01   1.687 0.091534 .
## V6A64        1.069e+00  5.625e-01   1.900 0.057396 .
## V6A65        1.120e+00  3.044e-01   3.679 0.000234 ***
## V7A72       -1.064e-01  4.763e-01  -0.223 0.823179
## V7A73        2.267e-01  4.529e-01   0.500 0.616742
## V7A74        7.389e-01  5.010e-01   1.475 0.140226
## V7A75        1.769e-01  4.633e-01   0.382 0.702638
```

```
## V8          -2.932e-01  1.021e-01  -2.872 0.004074 **
## V9A92        5.210e-01  4.378e-01   1.190 0.234022
## V9A93        1.034e+00  4.323e-01   2.391 0.016780 *
## V9A94        6.761e-01  5.211e-01   1.298 0.194414
## V10A102     -2.346e-01  4.392e-01  -0.534 0.593208
## V10A103      1.286e+00  5.054e-01   2.544 0.010952 *
## V11         -4.088e-02  9.965e-02  -0.410 0.681656
## V12A122     -1.920e-01  2.917e-01  -0.658 0.510516
## V12A123     -2.337e-01  2.673e-01  -0.874 0.382071
## V12A124     -1.003e+00  4.873e-01  -2.058 0.039627 *
## V13          1.991e-02  1.076e-02   1.851 0.064213 .
## V14A142      2.234e-01  4.795e-01   0.466 0.641277
## V14A143      8.500e-01  2.727e-01   3.117 0.001827 **
## V15A152      5.831e-01  2.709e-01   2.152 0.031363 *
## V15A153      1.156e+00  5.582e-01   2.070 0.038427 *
## V16         -4.555e-01  2.242e-01  -2.031 0.042207 *
## V17A172     -7.326e-01  7.908e-01  -0.926 0.354237
## V17A173     -7.160e-01  7.641e-01  -0.937 0.348785
## V17A174     -5.298e-01  7.689e-01  -0.689 0.490808
## V18         -2.149e-01  2.935e-01  -0.732 0.464051
## V19A192      1.391e-01  2.309e-01   0.602 0.546980
## V20A202      1.716e+00  8.574e-01   2.001 0.045375 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 980.75  on 799  degrees of freedom
## Residual deviance: 694.69  on 751  degrees of freedom
## AIC: 792.69
##
## Number of Fisher Scoring iterations: 5
```

We can see from this output that the attributes a number of the attributes show signficance. We can also see the AIC for this model is `792.6868447`.

To strive for simplier models, we'll select attributes V1, V2, V3, V4, V5, V6, V8, and V14. These attributes appear to be more focused on financial aspects whereas the others are demographic based. This seems like a reasonable approach for this exercise. So let's create a model using these values.

```
credit.train.fit2 <- glm(V21~V1+V2+V3+V4+V5+V6+V8+V14,family=binomial(link='logit'),data=credit.train)
summary(credit.train.fit2)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V14, family = binomial(link = "logit"),
##     data = credit.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5091  -0.7697   0.4228   0.7549   2.3148
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.432e-01  6.186e-01  -1.363 0.172843
## V1A12        2.679e-01  2.303e-01   1.164 0.244572
```

```
## V1A13        8.947e-01  4.059e-01   2.204 0.027511 *
## V1A14        1.514e+00  2.509e-01   6.035 1.59e-09 ***
## V2          -3.189e-02  9.859e-03  -3.235 0.001218 **
## V3A31        6.298e-01  5.815e-01   1.083 0.278785
## V3A32        1.113e+00  4.556e-01   2.443 0.014558 *
## V3A33        1.319e+00  5.095e-01   2.590 0.009611 **
## V3A34        1.934e+00  4.835e-01   4.000 6.35e-05 ***
## V4A41        1.867e+00  4.234e-01   4.410 1.03e-05 ***
## V4A410       1.917e+00  7.940e-01   2.415 0.015751 *
## V4A42        5.345e-01  2.668e-01   2.003 0.045147 *
## V4A43        8.617e-01  2.606e-01   3.307 0.000943 ***
## V4A44        1.609e-01  7.558e-01   0.213 0.831407
## V4A45        6.707e-01  6.858e-01   0.978 0.328071
## V4A46       -6.201e-01  4.310e-01  -1.439 0.150242
## V4A48        1.861e+00  1.199e+00   1.553 0.120435
## V4A49        6.098e-01  3.529e-01   1.728 0.083984 .
## V5          -8.207e-05  4.612e-05  -1.780 0.075140 .
## V6A62        1.503e-01  2.968e-01   0.507 0.612473
## V6A63        9.027e-01  4.802e-01   1.880 0.060145 .
## V6A64        9.437e-01  5.218e-01   1.809 0.070512 .
## V6A65        1.167e+00  2.882e-01   4.048 5.16e-05 ***
## V8          -2.194e-01  9.161e-02  -2.395 0.016643 *
## V14A142      3.102e-01  4.468e-01   0.694 0.487472
## V14A143      7.637e-01  2.540e-01   3.007 0.002641 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 980.75  on 799  degrees of freedom
## Residual deviance: 753.29  on 774  degrees of freedom
## AIC: 805.29
##
## Number of Fisher Scoring iterations: 5
```

This model has an AIC of 805.2934916. Perhaps we have an improved model. However, we see that attributes V5 and V6 may be less signficant. So let's create a new model without them.

```
credit.train.fit3 <- glm(V21~V1+V2+V3+V4+V8+V14,family=binomial(link='logit'),data=credit.train)
summary(credit.train.fit3)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V8 + V14, family = binomial(link = "logit"),
##     data = credit.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3634  -0.8178   0.4470   0.7519   2.2029
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.124714   0.587785  -1.913  0.05569 .
## V1A12        0.403650   0.218368   1.848  0.06453 .
## V1A13        1.009478   0.395812   2.550  0.01076 *
```

```
## V1A14        1.733779   0.240446   7.211 5.57e-13 ***
## V2          -0.038694   0.007564  -5.115 3.13e-07 ***
## V3A31        0.929960   0.565351   1.645  0.09999 .
## V3A32        1.308570   0.444620   2.943  0.00325 **
## V3A33        1.468283   0.499908   2.937  0.00331 **
## V3A34        2.052550   0.473101   4.338 1.43e-05 ***
## V4A41        1.672090   0.406093   4.118 3.83e-05 ***
## V4A410       1.600621   0.743043   2.154  0.03123 *
## V4A42        0.467408   0.261064   1.790  0.07339 .
## V4A43        0.809426   0.252619   3.204  0.00135 **
## V4A44        0.335354   0.752134   0.446  0.65569
## V4A45        0.579588   0.676849   0.856  0.39183
## V4A46       -0.571321   0.418966  -1.364  0.17268
## V4A48        1.852679   1.160574   1.596  0.11041
## V4A49        0.527960   0.341451   1.546  0.12205
## V8          -0.141720   0.082437  -1.719  0.08559 .
## V14A142      0.261762   0.437930   0.598  0.55002
## V14A143      0.718452   0.247932   2.898  0.00376 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 980.75  on 799  degrees of freedom
## Residual deviance: 778.26  on 779  degrees of freedom
## AIC: 820.26
##
## Number of Fisher Scoring iterations: 5
```

This model has an AIC of 820.2596984. Perhaps we have an improved model. Now let's compare AIC on test set.

```
credit.test.fit <- glm(V21~.,family=binomial(link='logit'),data=credit.test)
credit.test.fit2 <- glm(V21~V1+V2+V3+V4+V5+V6+V8+V14,family=binomial(link='logit'),data=credit.test)
credit.test.fit3 <- glm(V21~V1+V2+V3+V4+V8+V14,family=binomial(link='logit'),data=credit.test)
```

This yields AIC values 224.6636386, 212.0112687, and 215.0967629 respectively. This indicates that the original model with all of the parameters may be best as the relative likelihood of that model as compared to the minimum is 0.0083672. However this is small value, therefore we may need explore better techniques for attribute selection in future exercises.

Part 2

We now set out to determine a value of p that minimizes the cost of using our model. We'll use model 1 for this part of the exercise, which is the model with all of the attributes. We'll use the predict function to obtain a predicted response, and then vary p to measure cost. The cost is a sum of loans we should have made + 5 * loans we shouldn't have made.

```
credit.pred = predict(credit.test.fit, credit.test, type="response")
credit.test$predicted.V21 = credit.pred

p = .3
credit.test.fit.t1 = table(credit.test$V21, credit.pred >= p)
credit.test.fit.t1
```

```
##
##     FALSE TRUE
##   0    29   29
##   1     4  138
```

```
cost1 = credit.test.fit.t1[2,1] + credit.test.fit.t1[1,2] * 5

p = .5
credit.test.fit.t2 <- table(credit.test$V21, credit.pred >= p)
credit.test.fit.t2
```

```
##
##     FALSE TRUE
##   0    44   14
##   1    12  130
```

```
cost2 = credit.test.fit.t2[2,1] + credit.test.fit.t2[1,2] * 5

p = .7
credit.test.fit.t3 <- table(credit.test$V21, credit.pred >= p)
credit.test.fit.t3
```

```
##
##     FALSE TRUE
##   0    47   11
##   1    25  117
```

```
cost3 = credit.test.fit.t3[2,1] + credit.test.fit.t3[1,2] * 5
```

This result indicates that a higher threashold is safer (as expected) since cost of p=.7 is 80 whereas the cost of the other 2 is 82 and 149.