ISYE6501 Homework week 4

Dylan Peters

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.

Load the data:

```
uscrime <- read.delim("http://www.statsci.org/data/general/uscrime.txt")
dim(uscrime)
```

```
## [1] 47 16
```

Since we are doing PCA, we dont't need to remove variables with high covariance.

```
Train_X <- uscrime[,-16]
Train_Y <- uscrime[,16]
PCA <- prcomp(Train_X, 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
```

Looking at PC4, the cumulative variance in the first 4 components account for 80% of the variance. Not bad. Now to run a linear regression on those 4 components.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
uscrime.PCA <- data.frame(predict(PCA, Train_X)[,1:4])
model.PCA <- lm(Train_Y ~ ., data=uscrime.PCA)
summary(model.PCA)
```

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

```
## lm(formula = Train_Y ~ ., data = uscrime.PCA)
##
## 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
```

While the first two components have a rather low coefficient, the adjusted R-squred of 0.24 is lower than the model in the previous assignment. Also, the residual standard error is higher (336.4 compared to 213.2). In theory we should remove PC3 and PC4 since they have high p-values, however the assignment says to use all four so we will keep them.

Create the test data frame and generate a prediction for it:

```
test.df <- 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)

# Convert to PCA form first
test.PCA <- data.frame(predict(PCA, test.df))

prediction <- predict(model.PCA, test.PCA)
print ("The model prediction:")
```

```
## [1] "The model prediction:"
```

```
prediction
```

```
##        1
## 1112.678
```

```
# Get the model coefficients based on the original (scaled) parameters:
# a_j = SUM(k=1 to L)b_k * v_j_k
Actual.Coeffs <- PCA$rotation[,1:4] %*% model.PCA$coefficients[2:5]

print ("The model coefficients based on the original (scaled) parameters:")
```

```
## [1] "The model coefficients based on the original (scaled) parameters:"
```

```
Actual.Coeffs
```

```
##              [,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
```

```
model.PCA$coefficients[1]
```

```
## (Intercept)
##    905.0851
```

```
# Verify the results are the same; scale the test data
test.df.scaled <- (test.df - PCA$center) / PCA$scale
print ("The model manual prediction:")
```

```
## [1] "The model manual prediction:"
```

```
sum(Actual.Coeffs * test.df.scaled) + model.PCA$coefficients[[1]] # Intercept
```

```
## [1] 1112.678
```

The predicted value is similar to that predicted in the previous assignment.

Question 2

Using the crime data, find the best model you can using (a) a regression tree model, and (b) a random forest model.

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```
# single decision tree
# use caret train to do cross-validation
```

```r
rpControl <- rpart.control(minbucket = nrow(uscrime) * 0.05)
#rpart.model <- rpart(Crime ~ ., data=uscrime)
trControl <- trainControl(method = "LOOCV", search="grid")
grid <- expand.grid(cp=c(0.01, 0.05, 0.10, 0.20, 0.30))
rpart.model <- train(Crime ~ ., data=uscrime, metric="RMSE", method="rpart", tuneGrid=grid)

rpart.model$results
```

```
##     cp     RMSE  Rsquared   RMSESD RsquaredSD
## 1 0.01 372.4234 0.2337812 75.29900  0.1514778
## 2 0.05 376.1789 0.2206791 73.08329  0.1502921
## 3 0.10 379.9816 0.2075506 72.33427  0.1729494
## 4 0.20 384.8170 0.1855219 67.68823  0.1589540
## 5 0.30 382.6555 0.1685226 54.19219  0.1081097
```

```r
summary(rpart.model$finalModel)
```

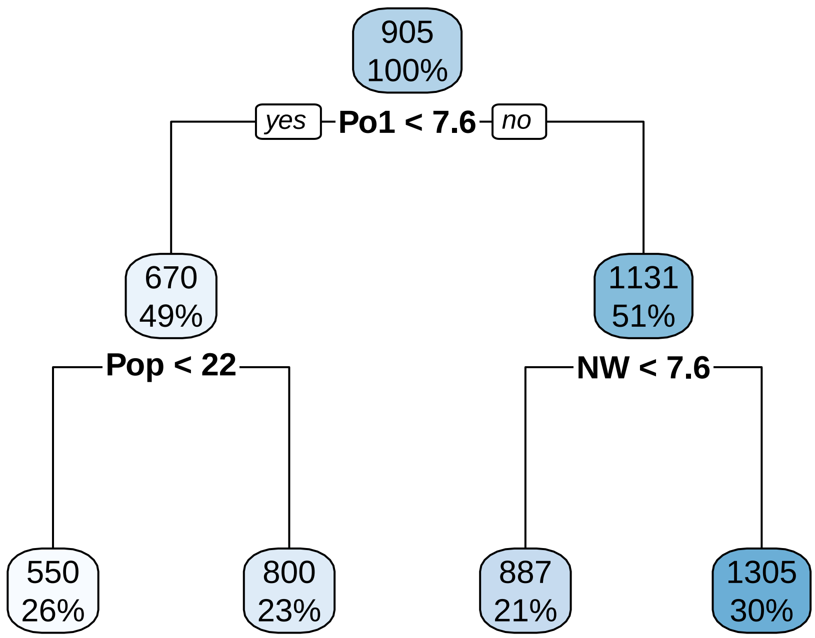
```
## Call:
## rpart(formula = .outcome ~ ., data = list(M = c(15.1, 14.3, 14.2,
## 13.6, 14.1, 12.1, 12.7, 13.1, 15.7, 14, 12.4, 13.4, 12.8, 13.5,
## 15.2, 14.2, 14.3, 13.5, 13, 12.5, 12.6, 15.7, 13.2, 13.1, 13,
## 13.1, 13.5, 15.2, 11.9, 16.6, 14, 12.5, 14.7, 12.6, 12.3, 15,
## 17.7, 13.3, 14.9, 14.5, 14.8, 14.1, 16.2, 13.6, 13.9, 12.6, 13
## ), So = c(1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
## 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
## 1, 1, 0, 0, 1, 0, 1, 0, 0), Ed = c(9.1, 11.3, 8.9, 12.1, 12.1,
## 11, 11.1, 10.9, 9, 11.8, 10.5, 10.8, 11.3, 11.7, 8.7, 8.8, 11,
## 10.4, 11.6, 10.8, 10.8, 8.9, 9.6, 11.6, 11.6, 12.1, 10.9, 11.2,
## 10.7, 8.9, 9.3, 10.9, 10.4, 11.8, 10.2, 10, 8.7, 10.4, 8.8, 10.4,
## 12.2, 10.9, 9.9, 12.1, 8.8, 10.4, 12.1), Po1 = c(5.8, 10.3, 4.5,
## 14.9, 10.9, 11.8, 8.2, 11.5, 6.5, 7.1, 12.1, 7.5, 6.7, 6.2, 5.7,
## 8.1, 6.6, 12.3, 12.8, 11.3, 7.4, 4.7, 8.7, 7.8, 6.3, 16, 6.9,
## 8.2, 16.6, 5.8, 5.5, 9, 6.3, 9.7, 9.7, 10.9, 5.8, 5.1, 6.1, 8.2,
## 7.2, 5.6, 7.5, 9.5, 4.6, 10.6, 9), Po2 = c(5.6, 9.5, 4.4, 14.1,
## 10.1, 11.5, 7.9, 10.9, 6.2, 6.8, 11.6, 7.1, 6, 6.1, 5.3, 7.7,
## 6.3, 11.5, 12.8, 10.5, 6.7, 4.4, 8.3, 7.3, 5.7, 14.3, 7.1, 7.6,
## 15.7, 5.4, 5.4, 8.1, 6.4, 9.7, 8.7, 9.8, 5.6, 4.7, 5.4, 7.4,
## 6.6, 5.4, 7, 9.6, 4.1, 9.7, 9.1), LF = c(0.51, 0.583, 0.533,
## 0.577, 0.591, 0.547, 0.519, 0.542, 0.553, 0.632, 0.58, 0.595,
## 0.624, 0.595, 0.53, 0.497, 0.537, 0.537, 0.536, 0.567, 0.602,
## 0.512, 0.564, 0.574, 0.641, 0.631, 0.54, 0.571, 0.521, 0.521,
## 0.535, 0.586, 0.56, 0.542, 0.526, 0.531, 0.638, 0.599, 0.515,
## 0.56, 0.601, 0.523, 0.522, 0.574, 0.48, 0.599, 0.623), M.F = c(95,
## 101.2, 96.9, 99.4, 98.5, 96.4, 98.2, 96.9, 95.5, 102.9, 96.6,
## 97.2, 97.2, 98.6, 98.6, 95.6, 97.7, 97.8, 93.4, 98.5, 98.4, 96.2,
## 95.3, 103.8, 98.4, 107.1, 96.5, 101.8, 93.8, 97.3, 104.5, 96.4,
## 97.2, 99, 94.8, 96.4, 97.4, 102.4, 95.3, 98.1, 99.8, 96.8, 99.6,
## 101.2, 96.8, 98.9, 104.9), Pop = c(33, 13, 18, 157, 18, 25, 4,
## 50, 39, 7, 101, 47, 28, 22, 30, 33, 10, 31, 51, 78, 34, 22, 43,
## 7, 14, 3, 6, 10, 168, 46, 6, 97, 23, 18, 113, 9, 24, 7, 36, 96,
## 9, 4, 40, 29, 19, 40, 3), NW = c(30.1, 10.2, 21.9, 8, 3, 4.4,
## 13.9, 17.9, 28.6, 1.5, 10.6, 5.9, 1, 4.6, 7.2, 32.1, 0.6, 17,
## 2.4, 9.4, 1.2, 42.3, 9.2, 3.6, 2.6, 7.7, 0.4, 7.9, 8.9, 25.4,
## 2, 8.2, 9.5, 2.1, 7.6, 2.4, 34.9, 4, 16.5, 12.6, 1.9, 0.2, 20.8,
```

```
## 3.6, 4.9, 2.4, 2.2), U1 = c(0.108, 0.096, 0.094, 0.102, 0.091,
## 0.084, 0.097, 0.079, 0.081, 0.1, 0.077, 0.083, 0.077, 0.077,
## 0.092, 0.116, 0.114, 0.089, 0.078, 0.13, 0.102, 0.097, 0.083,
## 0.142, 0.07, 0.102, 0.08, 0.103, 0.092, 0.072, 0.135, 0.105,
## 0.076, 0.102, 0.124, 0.087, 0.076, 0.099, 0.086, 0.088, 0.084,
## 0.107, 0.073, 0.111, 0.135, 0.078, 0.113), U2 = c(4.1, 3.6, 3.3,
## 3.9, 2, 2.9, 3.8, 3.5, 2.8, 2.4, 3.5, 3.1, 2.5, 2.7, 4.3, 4.7,
## 3.5, 3.4, 3.4, 5.8, 3.3, 3.4, 3.2, 4.2, 2.1, 4.1, 2.2, 2.8, 3.6,
## 2.6, 4, 4.3, 2.4, 3.5, 5, 3.8, 2.8, 2.7, 3.5, 3.1, 2, 3.7, 2.7,
## 3.7, 5.3, 2.5, 4), Wealth = c(3940, 5570, 3180, 6730, 5780, 6890,
## 6200, 4720, 4210, 5260, 6570, 5800, 5070, 5290, 4050, 4270, 4870,
## 6310, 6270, 6260, 5570, 2880, 5130, 5400, 4860, 6740, 5640, 5370,
## 6370, 3960, 4530, 6170, 4620, 5890, 5720, 5590, 3820, 4250, 3950,
## 4880, 5900, 4890, 4960, 6220, 4570, 5930, 5880), Ineq = c(26.1,
## 19.4, 25, 16.7, 17.4, 12.6, 16.8, 20.6, 23.9, 17.4, 17, 17.2,
## 20.6, 19, 26.4, 24.7, 16.6, 16.5, 13.5, 16.6, 19.5, 27.6, 22.7,
## 17.6, 19.6, 15.2, 13.9, 21.5, 15.4, 23.7, 20, 16.3, 23.3, 16.6,
## 15.8, 15.3, 25.4, 22.5, 25.1, 22.8, 14.4, 17, 22.4, 16.2, 24.9,
## 17.1, 16), Prob = c(0.084602, 0.029599, 0.083401, 0.015801, 0.041399,
## 0.034201, 0.0421, 0.040099, 0.071697, 0.044498, 0.016201, 0.031201,
## 0.045302, 0.0532, 0.0691, 0.052099, 0.076299, 0.119804, 0.019099,
## 0.034801, 0.0228, 0.089502, 0.0307, 0.041598, 0.069197, 0.041698,
## 0.036099, 0.038201, 0.0234, 0.075298, 0.041999, 0.042698, 0.049499,
## 0.040799, 0.0207, 0.0069, 0.045198, 0.053998, 0.047099, 0.038801,
## 0.0251, 0.088904, 0.054902, 0.0281, 0.056202, 0.046598, 0.052802
## ), Time = c(26.2011, 25.2999, 24.3006, 29.9012, 21.2998, 20.9995,
## 20.6993, 24.5988, 29.4001, 19.5994, 41.6, 34.2984, 36.2993, 21.501,
## 22.7008, 26.0991, 19.1002, 18.1996, 24.9008, 26.401, 37.5998,
## 37.0994, 25.1989, 17.6, 21.9003, 22.1005, 28.4999, 25.8006, 36.7009,
## 28.3011, 21.7998, 30.9014, 25.5005, 21.6997, 37.4011, 44.0004,
## 31.6995, 16.6999, 27.3004, 29.3004, 30.0001, 12.1996, 31.9989,
## 30.0001, 32.5996, 16.6999, 16.0997), .outcome = c(791, 1635,
## 578, 1969, 1234, 682, 963, 1555, 856, 705, 1674, 849, 511, 664,
## 798, 946, 539, 929, 750, 1225, 742, 439, 1216, 968, 523, 1993,
## 342, 1216, 1043, 696, 373, 754, 1072, 923, 653, 1272, 831, 566,
## 826, 1151, 880, 542, 823, 1030, 455, 508, 849)), control = list(
##     minsplit = 20, minbucket = 7, cp = 0, maxcompete = 4, maxsurrogate = 5,
##     usesurrogate = 2, surrogatestyle = 0, maxdepth = 30, xval = 0))
##   n= 47
##
##           CP nsplit rel error
## 1 0.36296293      0 1.0000000
## 2 0.14814320      1 0.6370371
## 3 0.05173165      2 0.4888939
## 4 0.00000000      3 0.4371622
##
## Variable importance
##    Po1    Po2 Wealth   Ineq   Prob      M     NW    Pop   Time     Ed
##     17     17     11     11     10     10      9      5      4      4
##     LF     So
##      1      1
##
## Node number 1: 47 observations,    complexity param=0.3629629
##   mean=905.0851, MSE=146402.7
```

```
##   left son=2 (23 obs) right son=3 (24 obs)
##   Primary splits:
##       Po1    < 7.65      to the left,  improve=0.3629629, (0 missing)
##       Po2    < 7.2       to the left,  improve=0.3629629, (0 missing)
##       Prob   < 0.0418485 to the right, improve=0.3217700, (0 missing)
##       NW     < 7.65      to the left,  improve=0.2356621, (0 missing)
##       Wealth < 6240      to the left,  improve=0.2002403, (0 missing)
##   Surrogate splits:
##       Po2    < 7.2       to the left,  agree=1.000, adj=1.000, (0 split)
##       Wealth < 5330      to the left,  agree=0.830, adj=0.652, (0 split)
##       Prob   < 0.043598  to the right, agree=0.809, adj=0.609, (0 split)
##       M      < 13.25     to the right, agree=0.745, adj=0.478, (0 split)
##       Ineq   < 17.15     to the right, agree=0.745, adj=0.478, (0 split)
##
## Node number 2: 23 observations,    complexity param=0.05173165
##   mean=669.6087, MSE=33880.15
##   left son=4 (12 obs) right son=5 (11 obs)
##   Primary splits:
##       Pop < 22.5      to the left,  improve=0.4568043, (0 missing)
##       M   < 14.5      to the left,  improve=0.3931567, (0 missing)
##       NW  < 5.4       to the left,  improve=0.3184074, (0 missing)
##       Po1 < 5.75      to the left,  improve=0.2310098, (0 missing)
##       U1  < 0.093     to the right, improve=0.2119062, (0 missing)
##   Surrogate splits:
##       NW   < 5.4       to the left,  agree=0.826, adj=0.636, (0 split)
##       M    < 14.5      to the left,  agree=0.783, adj=0.545, (0 split)
##       Time < 22.30055  to the left,  agree=0.783, adj=0.545, (0 split)
##       So   < 0.5       to the left,  agree=0.739, adj=0.455, (0 split)
##       Ed   < 10.85     to the right, agree=0.739, adj=0.455, (0 split)
##
## Node number 3: 24 observations,    complexity param=0.1481432
##   mean=1130.75, MSE=150173.4
##   left son=6 (10 obs) right son=7 (14 obs)
##   Primary splits:
##       NW   < 7.65      to the left,  improve=0.2828293, (0 missing)
##       M    < 13.05     to the left,  improve=0.2714159, (0 missing)
##       Time < 21.9001   to the left,  improve=0.2060170, (0 missing)
##       M.F  < 99.2      to the left,  improve=0.1703438, (0 missing)
##       Po1  < 10.75     to the left,  improve=0.1659433, (0 missing)
##   Surrogate splits:
##       Ed   < 11.45     to the right, agree=0.750, adj=0.4, (0 split)
##       Ineq < 16.25     to the left,  agree=0.750, adj=0.4, (0 split)
##       Time < 21.9001   to the left,  agree=0.750, adj=0.4, (0 split)
##       Pop  < 30        to the left,  agree=0.708, adj=0.3, (0 split)
##       LF   < 0.5885    to the right, agree=0.667, adj=0.2, (0 split)
##
## Node number 4: 12 observations
##   mean=550.5, MSE=20317.58
##
## Node number 5: 11 observations
##   mean=799.5455, MSE=16315.52
##
## Node number 6: 10 observations
##   mean=886.9, MSE=55757.49
```

```
##
## Node number 7: 14 observations
##   mean=1304.929, MSE=144801.8
```

```r
rpart.plot(rpart.model$finalModel)
```



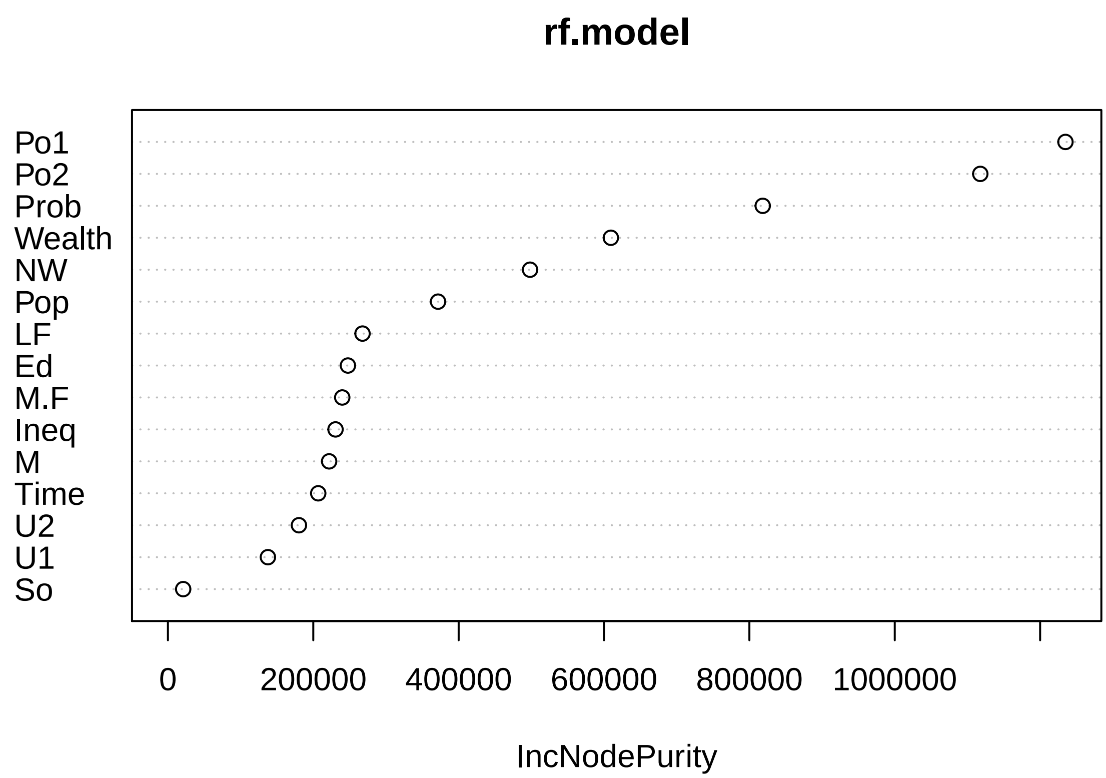

```r
#plot(rpart.model$finalModel, uniform= TRUE, main = "US Crime")
#text(rpart.model$finalModel, use.n= TRUE, all= TRUE, cex = 0.8)
```

And the random forest:

```r
# random forest
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

```r
rf.model <- randomForest(Crime ~ ., data=uscrime)
varImpPlot(rf.model)
```



Question 3

Describe a problem where a logistic regression model would be appropriate:

A good metric for an online class is how likely a learner is to complete a course. Possible variables could be whether the learner is a paid learner, whether they have completed courses before, how many courses they have abandoned, how long is the course, and is the course similar to other courses they have completed.

Question 4

Using german loan data, 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.

Part 1: Load the data:

```
germancredit <- read.delim("germancredit.txt", sep=" ", header=FALSE)
dim(germancredit)
```

```
## [1] 1000   21
```

The glm function requires a target between 0 and 1. In the dataset, V21 of 2 means bad, 1 means good. I will change the 2's to 0's. So 1 means a good loan, 0 means a bad loan.

```r
library(caret)
germancredit$V21[germancredit$V21 == 2] <- 0
germancredit$V21 <- as.factor(germancredit$V21)
```

Create the model

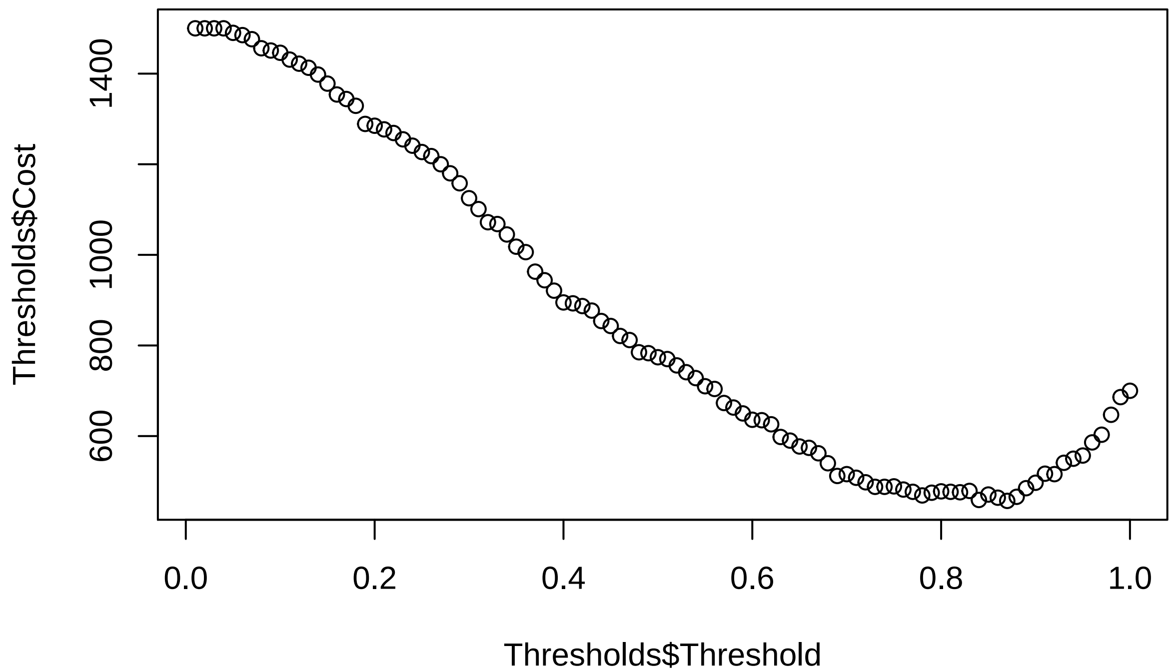
```r
model.glm <- glm(V21 ~ ., data=germancredit, family=binomial(link="logit"))

summary(model.glm)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = germancredit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6116  -0.7095   0.3752   0.6994   2.3410
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.005e-01  1.084e+00  -0.369 0.711869
## V1A12        3.749e-01  2.179e-01   1.720 0.085400 .
## V1A13        9.657e-01  3.692e-01   2.616 0.008905 **
## V1A14        1.712e+00  2.322e-01   7.373 1.66e-13 ***
## V2          -2.786e-02  9.296e-03  -2.997 0.002724 **
## V3A31       -1.434e-01  5.489e-01  -0.261 0.793921
## V3A32        5.861e-01  4.305e-01   1.362 0.173348
## V3A33        8.532e-01  4.717e-01   1.809 0.070470 .
## V3A34        1.436e+00  4.399e-01   3.264 0.001099 **
## V4A41        1.666e+00  3.743e-01   4.452 8.51e-06 ***
## V4A410       1.489e+00  7.764e-01   1.918 0.055163 .
## V4A42        7.916e-01  2.610e-01   3.033 0.002421 **
## V4A43        8.916e-01  2.471e-01   3.609 0.000308 ***
## V4A44        5.228e-01  7.623e-01   0.686 0.492831
## V4A45        2.164e-01  5.500e-01   0.393 0.694000
## V4A46       -3.628e-02  3.965e-01  -0.092 0.927082
## V4A48        2.059e+00  1.212e+00   1.699 0.089297 .
## V4A49        7.401e-01  3.339e-01   2.216 0.026668 *
## V5          -1.283e-04  4.444e-05  -2.887 0.003894 **
## V6A62        3.577e-01  2.861e-01   1.250 0.211130
## V6A63        3.761e-01  4.011e-01   0.938 0.348476
## V6A64        1.339e+00  5.249e-01   2.551 0.010729 *
## V6A65        9.467e-01  2.625e-01   3.607 0.000310 ***
## V7A72        6.691e-02  4.270e-01   0.157 0.875475
## V7A73        1.828e-01  4.105e-01   0.445 0.656049
## V7A74        8.310e-01  4.455e-01   1.866 0.062110 .
## V7A75        2.766e-01  4.134e-01   0.669 0.503410
## V8          -3.301e-01  8.828e-02  -3.739 0.000185 ***
## V9A92        2.755e-01  3.865e-01   0.713 0.476040
## V9A93        8.161e-01  3.799e-01   2.148 0.031718 *
## V9A94        3.671e-01  4.537e-01   0.809 0.418448
## V10A102     -4.360e-01  4.101e-01  -1.063 0.287700
## V10A103      9.786e-01  4.243e-01   2.307 0.021072 *
## V11         -4.776e-03  8.641e-02  -0.055 0.955920
```

```
## V12A122     -2.814e-01  2.534e-01  -1.111 0.266630
## V12A123     -1.945e-01  2.360e-01  -0.824 0.409743
## V12A124     -7.304e-01  4.245e-01  -1.721 0.085308 .
## V13          1.454e-02  9.222e-03   1.576 0.114982
## V14A142      1.232e-01  4.119e-01   0.299 0.764878
## V14A143      6.463e-01  2.391e-01   2.703 0.006871 **
## V15A152      4.436e-01  2.347e-01   1.890 0.058715 .
## V15A153      6.839e-01  4.770e-01   1.434 0.151657
## V16         -2.721e-01  1.895e-01  -1.436 0.151109
## V17A172     -5.361e-01  6.796e-01  -0.789 0.430160
## V17A173     -5.547e-01  6.549e-01  -0.847 0.397015
## V17A174     -4.795e-01  6.623e-01  -0.724 0.469086
## V18         -2.647e-01  2.492e-01  -1.062 0.288249
## V19A192      3.000e-01  2.013e-01   1.491 0.136060
## V20A202      1.392e+00  6.258e-01   2.225 0.026095 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  895.82  on 951  degrees of freedom
## AIC: 993.82
##
## Number of Fisher Scoring iterations: 5
```

Part 2: Determine a good threshold:

```
Threshold <- sequence(100) / 100

Thresholds <- data.frame(Threshold = Threshold, Cost = Threshold * 0,
                         FalsePostives = Threshold * 0,
                         FalseNegatives = Threshold * 0)

#  incorrectly identifying a bad customer as good, is 5 times worse than
#  incorrectly classifying a good customer as bad

for (i in 1:nrow(Thresholds))
{
  predicted <- as.integer(model.glm$fitted.values > Thresholds[i,1])
  # Calculate cost: 1 is a good loan, 0 is a bad loan
  # Therefore classifying 1 when 0 is true is a false positive (cost of 5)
  false_positive_count <- sum(predicted == 1 & germancredit$V21 == 0)
  false_negative_count <- sum(predicted == 0 & germancredit$V21 == 1)

  cost <- false_positive_count * 5 + false_negative_count * 1
  Thresholds[i,2] <- cost
  Thresholds[i,3] <- false_positive_count
  Thresholds[i,4] <- false_negative_count

}

plot(Thresholds$Threshold, Thresholds$Cost)
```



It looks like the lowest cost threshold is about 0.83. Here is a confusion matrix at that threshold:

```
predicted <- as.integer(model.glm$fitted.values > 0.83)

table(germancredit$V21, predicted)
```

```
##    predicted
##       0   1
##   0 266  34
##   1 309 391
```