Homework 7

Richard Albright
ISYE6501
Spring 2018

2/24/2019

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. 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).

Read in the CSV

```
data <-
  read.table(
    "/Users/ralbright/Dropbox/ISYE6501/week3/homework/uscrime.txt",
    header=TRUE,
    sep="\t"
  )
```

Head:

```
table <- xtable(head(data))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
1	15.10	1	9.10	5.80	5.60	0.51	95.00	33	30.10	0.11	4.10	3940	26.10	0.08	26.20	791
2	14.30	0	11.30	10.30	9.50	0.58	101.20	13	10.20	0.10	3.60	5570	19.40	0.03	25.30	1635
3	14.20	1	8.90	4.50	4.40	0.53	96.90	18	21.90	0.09	3.30	3180	25.00	0.08	24.30	578
4	13.60	0	12.10	14.90	14.10	0.58	99.40	157	8.00	0.10	3.90	6730	16.70	0.02	29.90	1969
5	14.10	0	12.10	10.90	10.10	0.59	98.50	18	3.00	0.09	2.00	5780	17.40	0.04	21.30	1234
6	12.10	0	11.00	11.80	11.50	0.55	96.40	25	4.40	0.08	2.90	6890	12.60	0.03	21.00	682

Tail:

```
table <- xtable(tail(data))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
42	14.10	0	10.90	5.60	5.40	0.52	96.80	4	0.20	0.11	3.70	4890	17.00	0.09	12.20	542
43	16.20	1	9.90	7.50	7.00	0.52	99.60	40	20.80	0.07	2.70	4960	22.40	0.05	32.00	823
44	13.60	0	12.10	9.50	9.60	0.57	101.20	29	3.60	0.11	3.70	6220	16.20	0.03	30.00	1030
45	13.90	1	8.80	4.60	4.10	0.48	96.80	19	4.90	0.14	5.30	4570	24.90	0.06	32.60	455
46	12.60	0	10.40	10.60	9.70	0.60	98.90	40	2.40	0.08	2.50	5930	17.10	0.05	16.70	508
47	13.00	0	12.10	9.00	9.10	0.62	104.90	3	2.20	0.11	4.00	5880	16.00	0.05	16.10	849

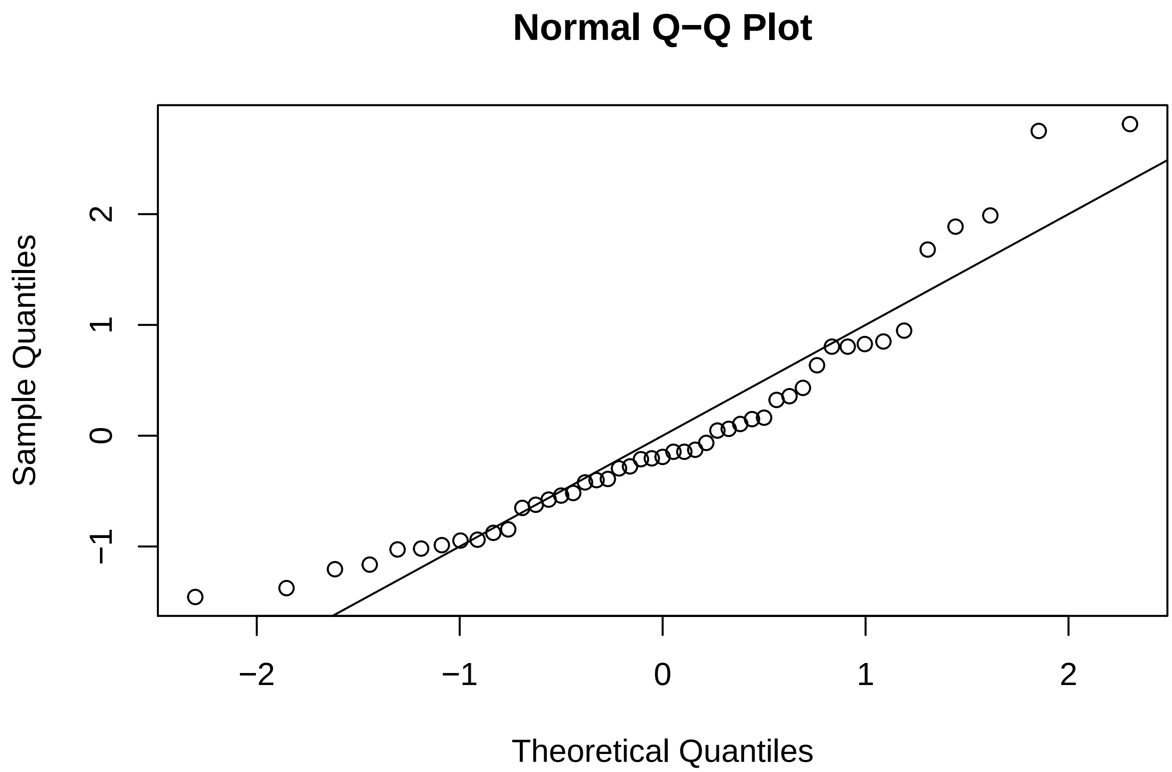
Summary:

```
table <- xtable(summary(data))
print(table, type='latex', comment=FALSE, scalebox='0.4')
```

The plot of the scaled Crime Response Variable using qqnorm also looks like.

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
X	Min. :11.90	Min. :0.0000	Min. : 8.70	Min. : 4.50	Min. : 4.100	Min. :0.4800	Min. : 93.40	Min. : 3.00	Min. : 0.20	Min. :0.07000	Min. :2.000	Min. :2880	Min. :12.60	Min. :0.00690	Min. :12.20	Min. : 342.0
X.1	1st Qu.:13.00	1st Qu.:0.0000	1st Qu.: 9.75	1st Qu.: 6.25	1st Qu.: 5.850	1st Qu.:0.5305	1st Qu.: 96.45	1st Qu.: 10.00	1st Qu.: 2.40	1st Qu.:0.08050	1st Qu.:2.750	1st Qu.:4595	1st Qu.:16.55	1st Qu.:0.03270	1st Qu.:21.60	1st Qu.: 658.5
X.2	Median :13.60	Median :0.0000	Median :10.80	Median : 7.80	Median : 7.300	Median :0.5600	Median : 97.70	Median : 25.00	Median : 7.60	Median :0.09200	Median :3.400	Median :5370	Median :17.60	Median :0.04210	Median :25.80	Median : 831.0
X.3	Mean :13.86	Mean :0.3404	Mean :10.56	Mean : 8.50	Mean : 8.023	Mean :0.5612	Mean : 98.30	Mean : 36.62	Mean :10.11	Mean :0.09547	Mean :3.398	Mean :5254	Mean :19.40	Mean :0.04709	Mean :26.60	Mean : 905.1
X.4	3rd Qu.:14.60	3rd Qu.:1.0000	3rd Qu.:11.45	3rd Qu.:10.45	3rd Qu.: 9.700	3rd Qu.:0.5930	3rd Qu.: 99.20	3rd Qu.: 41.50	3rd Qu.:13.25	3rd Qu.:0.10400	3rd Qu.:3.850	3rd Qu.:5915	3rd Qu.:22.75	3rd Qu.:0.05445	3rd Qu.:30.45	3rd Qu.:1057.5
X.5	Max. :17.70	Max. :1.0000	Max. :12.20	Max. :16.60	Max. :15.700	Max. :0.6410	Max. :107.10	Max. :168.00	Max. :42.30	Max. :0.14200	Max. :5.800	Max. :6890	Max. :27.60	Max. :0.11980	Max. :44.00	Max. :1993.0

```
scaled_crime = scale(data$Crime)
qqnorm(scaled_crime)
abline(0,1)
```



Which seems to indicate that there may outliers in both tails.

Lets take a look at a box plot of our Crime response variable as well.

```
boxplot(data$Crime, main="Crime", boxwex=0.1)
```


```r
possible_outliers <- boxplot.stats(data$Crime)$out
possible_outliers
```

```
## [1] 1969 1674 1993
```

The boxplot points to possible outliers in the upper tail. Output from boxplot.stats indicates that the 3 possible outliers are 1969, 1674, & 1993. We will now use the grubbs.test function to test for the outliers from the data set.

We will use the 1st 2 tests of the The grubbs.test function below (taken directly from the R Documentation).

First test (10) is used to detect if the sample dataset contains one outlier, statistically different than the other values. Test is based by calculating score of this outlier G (outlier minus mean and divided by sd) and comparing it to appropriate critical values. Alternative method is calculating ratio of variances of two datasets - full dataset and dataset without outlier. The obtained value called U is bound with G by simple formula.

Second test (11) is used to check if lowest and highest value are two outliers on opposite tails of sample. It is based on calculation of ratio of range to standard deviation of the sample.

We will loop through the 1st two test types on the Crime column.

```r
tests <- c(10, 11)
for(test in tests) {
  for(truth in c(TRUE,FALSE)) {
    gtest <- grubbs.test(as.vector(data$Crime), type=test, opposite=truth)
    print(paste('Grubbs Test Type:', test, collapse=' '))
    print(gtest)
  }
}
```

```
## [1] "Grubbs Test Type: 10"
##
##  Grubbs test for one outlier
##
## data:  as.vector(data$Crime)
## G = 1.45590, U = 0.95292, p-value = 1
## alternative hypothesis: lowest value 342 is an outlier
```

```
##
## [1] "Grubbs Test Type: 10"
##
##  Grubbs test for one outlier
##
## data:  as.vector(data$Crime)
## G = 2.81290, U = 0.82426, p-value = 0.07887
## alternative hypothesis: highest value 1993 is an outlier
##
## [1] "Grubbs Test Type: 11"
##
##  Grubbs test for two opposite outliers
##
## data:  as.vector(data$Crime)
## G = 4.26880, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
##
## [1] "Grubbs Test Type: 11"
##
##  Grubbs test for two opposite outliers
##
## data:  as.vector(data$Crime)
## G = 4.26880, U = 0.78103, p-value = 1
## alternative hypothesis: 342 and 1993 are outliers
```

Using a 95% confidence interval, We accept the null hypothesis that there are not any outliers in our Crime reponse variable.

Now that we know there are no outliers to contend with, we will buld a regression tree and random forest model.

```r
# Fit a regression tree function to the crime data. Note that the deviance is a quality of fit statisti

data_tree <- tree(Crime~., data=data)
```

```r
summ <- summary(data_tree)
summ
```

```
##
## Regression tree:
## tree(formula = Crime ~ ., data = data)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "LF"  "NW"
## Number of terminal nodes:  7
## Residual mean deviance:  47390 = 1896000 / 40
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -98.300   -1.545    0.000  110.600  490.100
```

The most significant variables used to create branches in the tree are Po1, Pop, LF, and NW. Resulting in 7 leaves. our residual mean deviance is 47390. Let's compute an r^2 for our tree. We can either sum the square of the residuals from the summary object (summ$residuals), the dev variable(summ$dev), or just call the deviance function as provided by the tree package on our model. I opted for the last method.

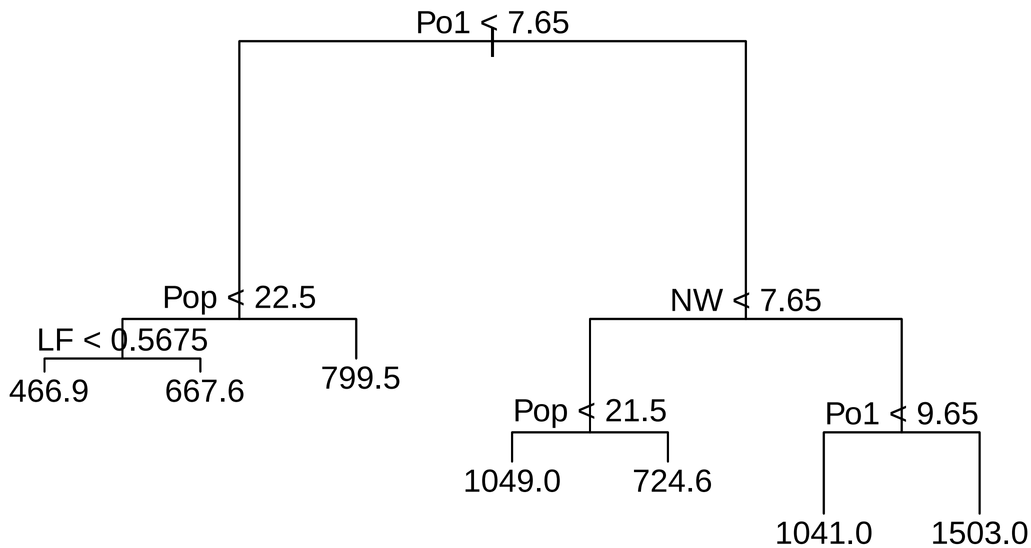
```r
ssr <- deviance(data_tree)
sst <- sum((data$Crime - mean(data$Crime))^2)
r2 = 1 - ssr/sst
```

```r
r2
```

```
## [1] 0.7244962
```

Let's inspect our tree more prior to making predictions and getting an r^2 from those predictions.

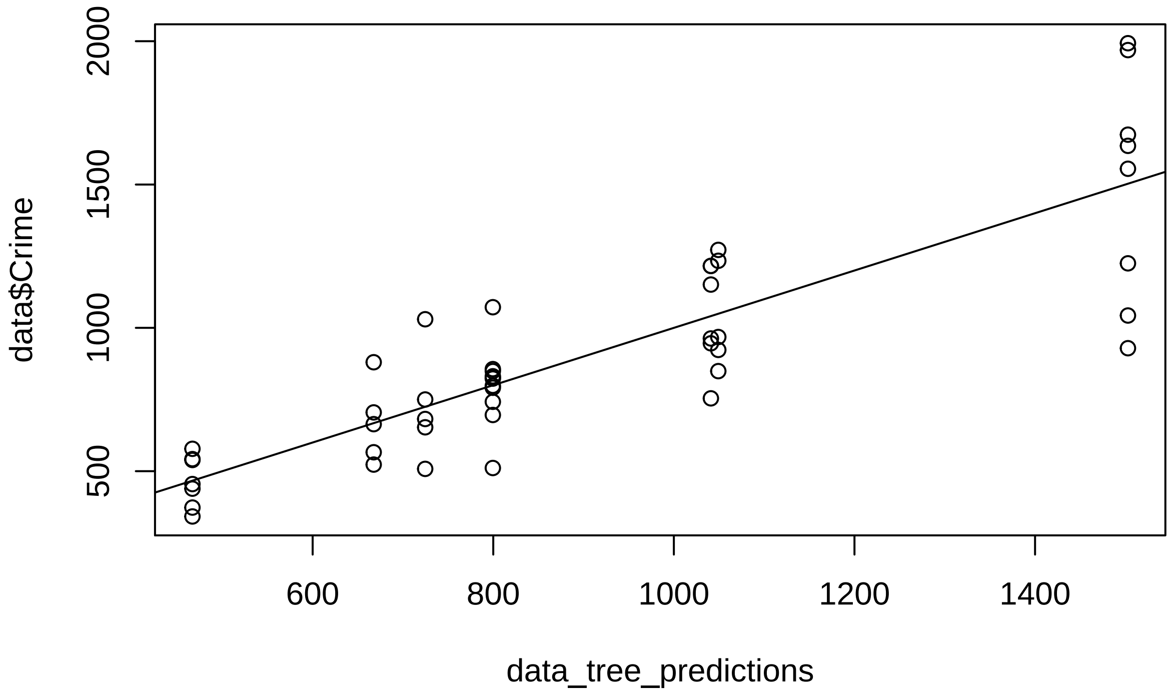
```r
plot(data_tree)
text(data_tree)
```




```r
data_tree$frame
```

```
##        var  n        dev      yval splits.cutleft splits.cutright
## 1      Po1 47 6880927.66  905.0851          <7.65           >7.65
## 2      Pop 23  779243.48  669.6087          <22.5           >22.5
## 4       LF 12  243811.00  550.5000        <0.5675         >0.5675
## 8   <leaf>  7   48518.86  466.8571
## 9   <leaf>  5   77757.20  667.6000
## 5   <leaf> 11  179470.73  799.5455
## 3       NW 24 3604162.50 1130.7500          <7.65           >7.65
## 6      Pop 10  557574.90  886.9000          <21.5           >21.5
## 12 <leaf>  5  146390.80 1049.2000
## 13 <leaf>  5  147771.20  724.6000
## 7      Po1 14 2027224.93 1304.9286          <9.65           >9.65
## 14 <leaf>  6  170828.00 1041.0000
## 15 <leaf>  8 1124984.88 1502.8750
```

```r
# manually compute r-squared. Is this a good measure of the quality of fit? Notice we only use averages
data_tree_predictions <- predict(data_tree)
plot(data_tree_predictions, data$Crime)
abline(0,1)
```




```r
ssr <- sum((data_tree_predictions-data$Crime)^2)
r2 = 1 - ssr/sst
r2
```

```
## [1] 0.7244962
```

Now let's inspect our pruned tree object on our training set, we are looking for the tree whose number of leaves has the least standard error.

```r
prune.tree(data_tree)$size
```

```
## [1] 7 6 5 4 3 2 1
```

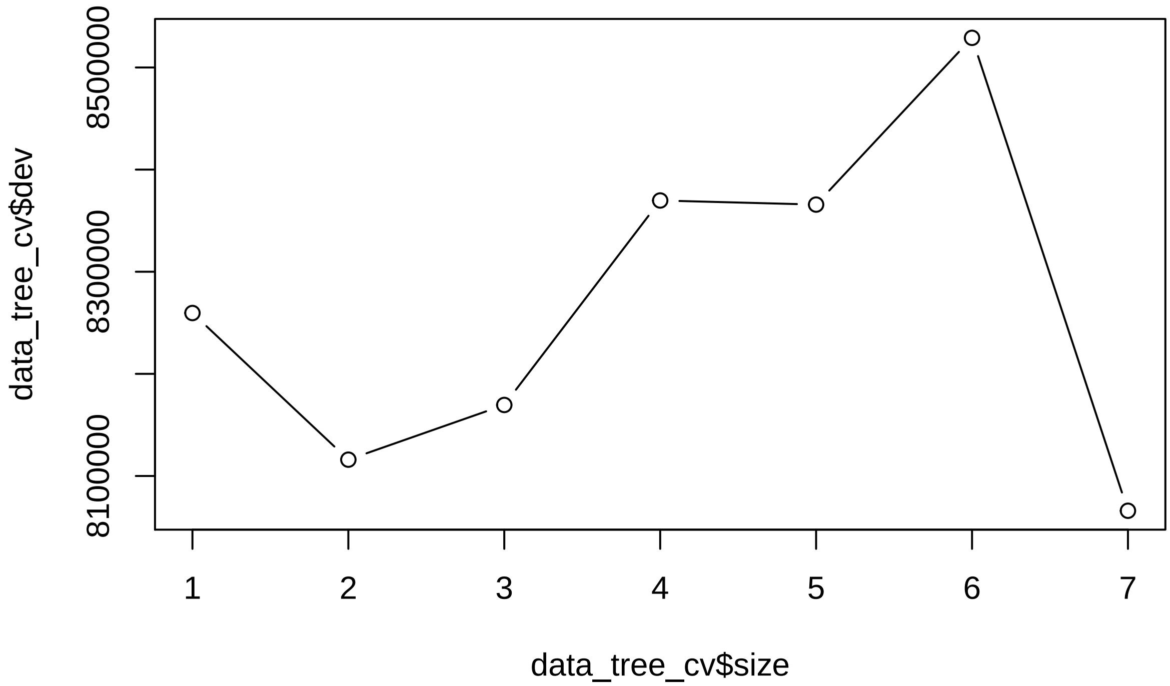
```r
prune.tree(data_tree)$dev
```

```
## [1] 1895722 2013257 2276670 2632631 3364043 4383406 6880928
```

```r
set.seed(1)
data_tree_cv = cv.tree(data_tree)
r2_cv = 1 - sum(data_tree_cv$dev)/sum(data_tree_cv$size) / sst
r2_cv
```

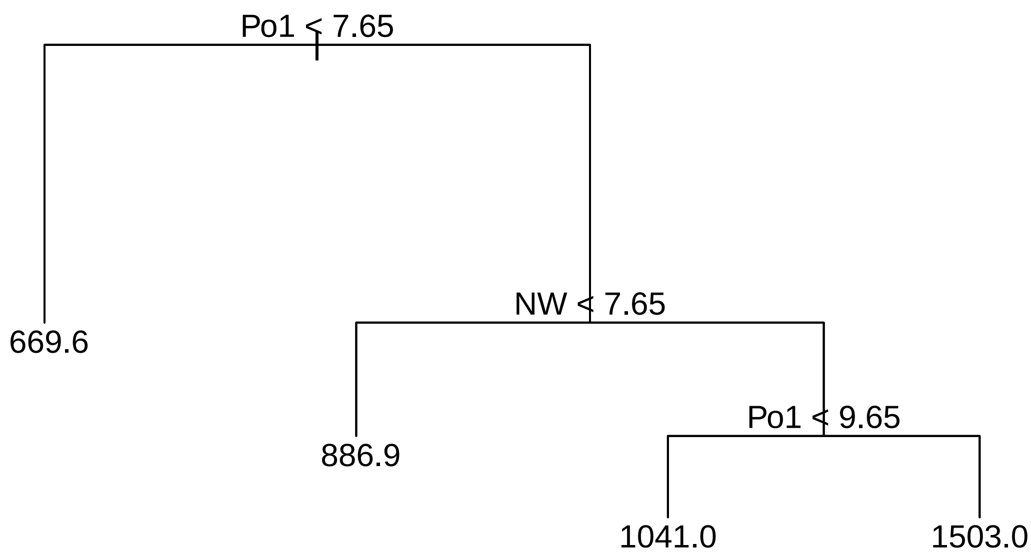
```
## [1] 0.699607
```

```r
plot(data_tree_cv$size, data_tree_cv$dev, type="b")
```



Based upon inspection of our pruned tree, pruning it is not a good idea, as all other error's are higher than without pruning. Let's see what the resulting r^2 would be if we did decide to prune the tree and leave the best 4 leaves.

```
data_tree_pruned <- prune.tree(data_tree, best=4)
plot(data_tree_pruned)
text(data_tree_pruned)
```




```
pruned_predict <- predict(data_tree_pruned)
ssr = sum((pruned_predict-data$Crime)^2)
r2_pruned  = 1 - ssr/sst
r2_pruned
```

```
## [1] 0.6174017
```

Our resulting r^2 drops to 0.6174017. This model is not very good given the amount of data points avaible, since only leaf 2 has enough data to perform a linear regression on it.

```
pruned_data1 <- data[which(data_tree_pruned$where == 1),]
pruned_data2 <- data[which(data_tree_pruned$where == 2),]
pruned_data3 <- data[which(data_tree_pruned$where == 3),]
pruned_data4 <- data[which(data_tree_pruned$where == 4),]

nrow(pruned_data1)
```

```
## [1] 0
```

```
nrow(pruned_data2)
```

```
## [1] 23
```

```
nrow(pruned_data3)
```

```
## [1] 0
```

```
nrow(pruned_data4)
```

```
## [1] 10
```

```
pruned_tree2 = lm(Crime~.,data=pruned_data2)
summary(pruned_tree2)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = pruned_data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -109.147  -52.803   -6.495   53.784  127.196
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -48.5477  2044.9766  -0.024   0.9817
## M              45.8622    58.6256   0.782   0.4597
## So            380.4815   223.1072   1.705   0.1319
## Ed            187.9074    89.5799   2.098   0.0741 .
## Po1            -3.5138   157.7513  -0.022   0.9829
## Po2            44.6382   148.5528   0.300   0.7725
## LF           1059.3652  1187.9722   0.892   0.4021
## M.F           -22.5521    21.4677  -1.051   0.3284
## Pop            10.6413     5.0929   2.089   0.0750 .
## NW              0.1010     7.9019   0.013   0.9902
## U1           4878.2802  4874.8165   1.001   0.3503
## U2             -5.5126   133.5094  -0.041   0.9682
## Wealth         -0.1022     0.1752  -0.583   0.5779
## Ineq            4.7779    35.5290   0.134   0.8968
## Prob        -7317.4407  3280.7511  -2.230   0.0609 .
## Time          -20.0603     7.7287  -2.596   0.0357 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 115.9 on 7 degrees of freedom
## Multiple R-squared:  0.8794, Adjusted R-squared:  0.6209
## F-statistic: 3.403 on 15 and 7 DF,  p-value: 0.0541
```

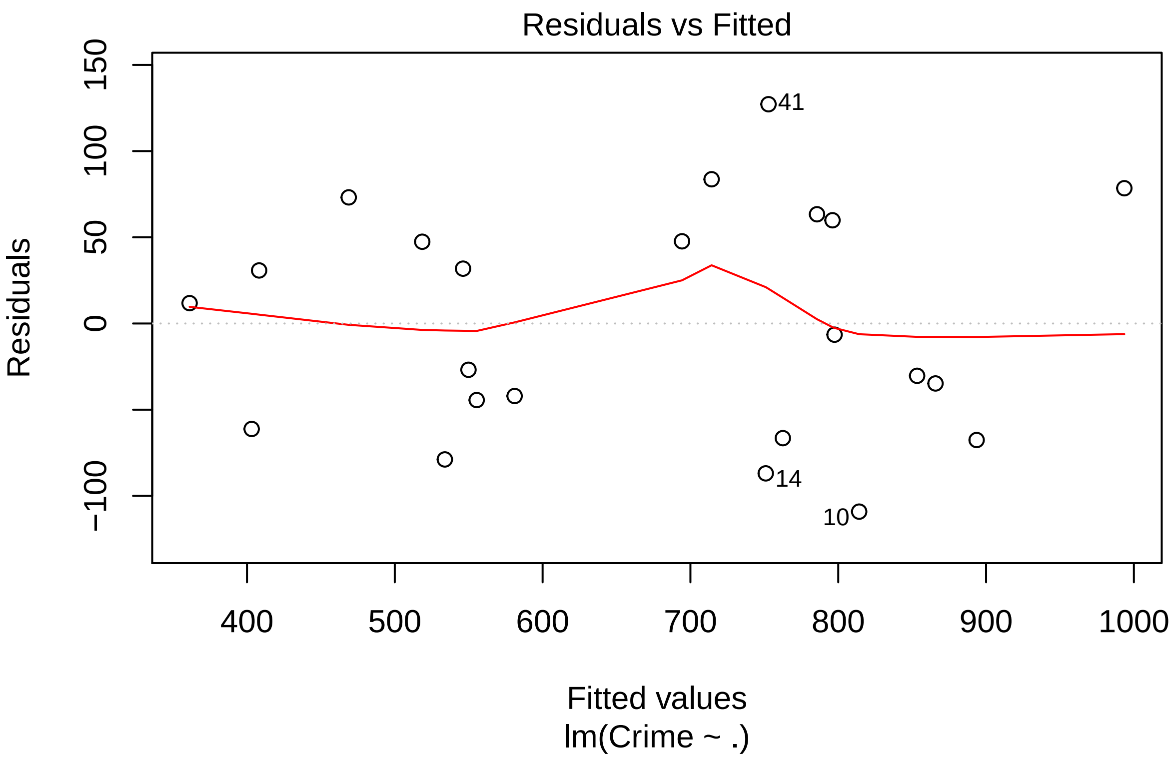
```r
pruned_tree4 = lm(Crime~.,data=pruned_data4)
summary(pruned_tree4)
```

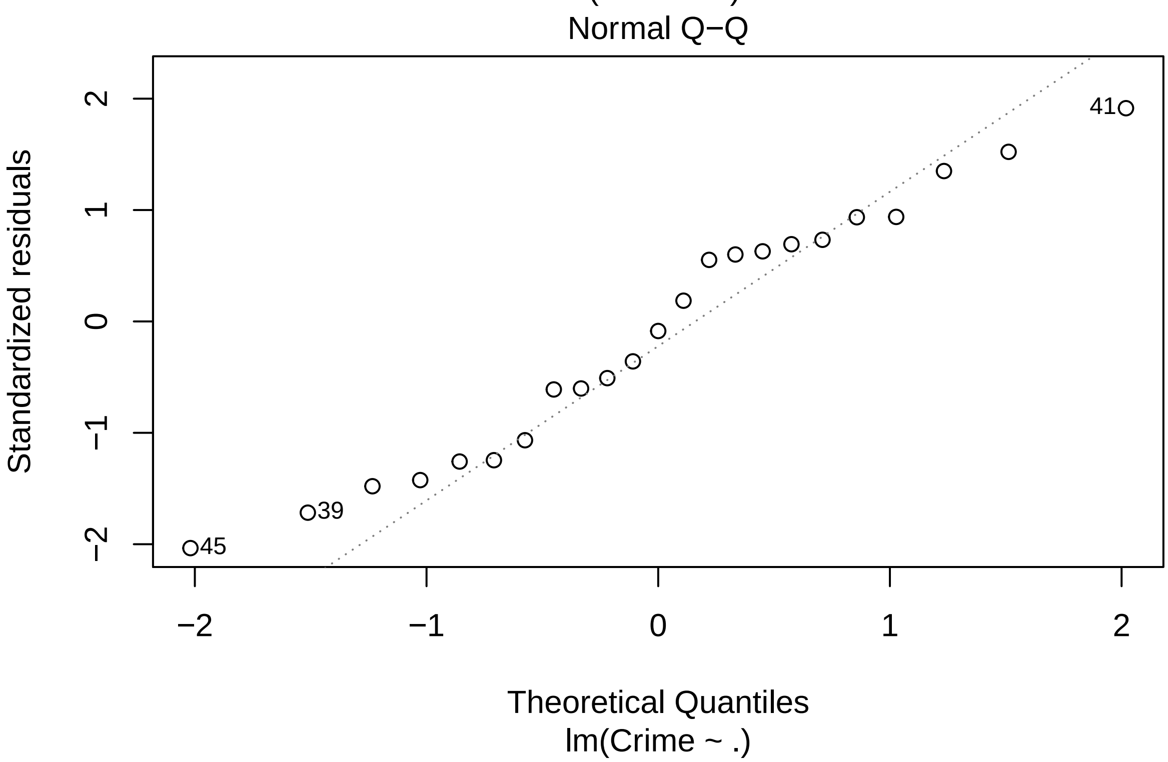
```
##
## Call:
## lm(formula = Crime ~ ., data = pruned_data4)
##
## Residuals:
## ALL 10 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (6 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 32527.85         NA      NA       NA
## M             258.27         NA      NA       NA
## So                NA         NA      NA       NA
## Ed            -46.38         NA      NA       NA
## Po1         -1168.92         NA      NA       NA
## Po2           612.42         NA      NA       NA
## LF          16612.42         NA      NA       NA
## M.F          -384.45         NA      NA       NA
## Pop           -18.22         NA      NA       NA
## NW            124.13         NA      NA       NA
## U1           2064.68         NA      NA       NA
## U2                NA         NA      NA       NA
## Wealth            NA         NA      NA       NA
## Ineq              NA         NA      NA       NA
## Prob              NA         NA      NA       NA
## Time              NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 9 and 0 DF,  p-value: NA
```

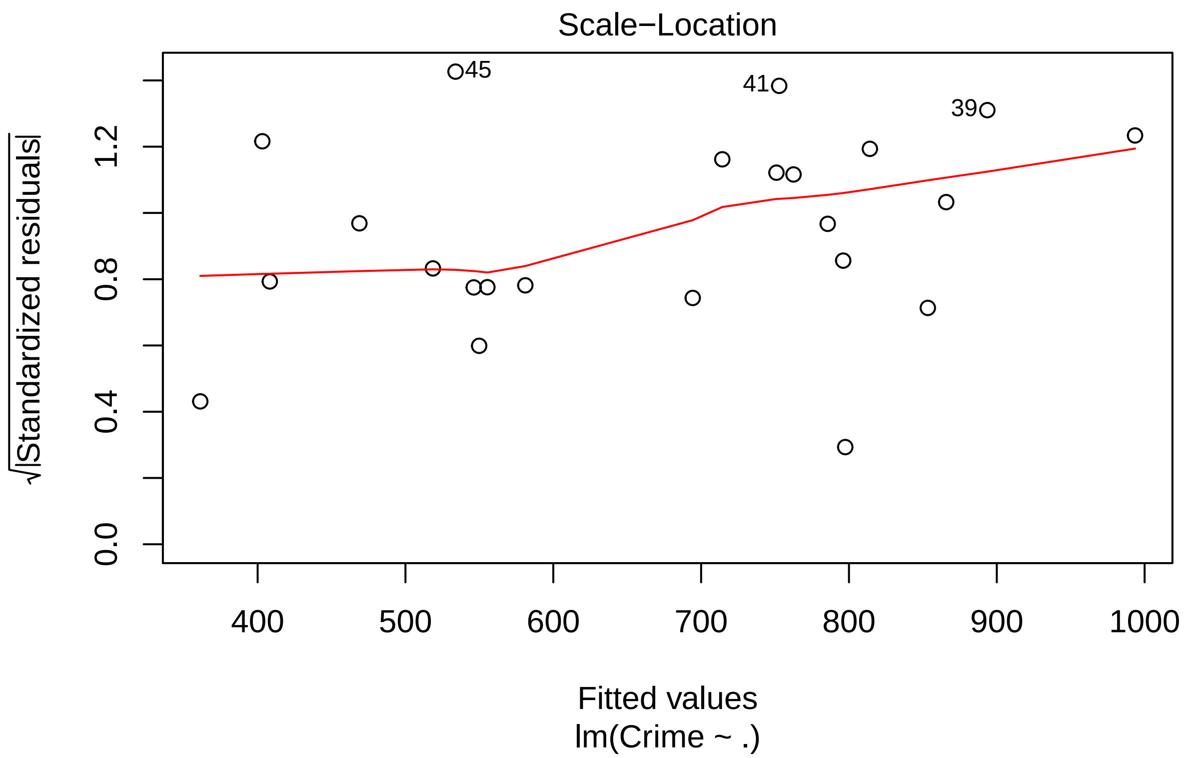
The r^2 on our linear regression of leaf 2 of our pruned tree is 0.8794, and our adjusted r^2 is 0.6209.

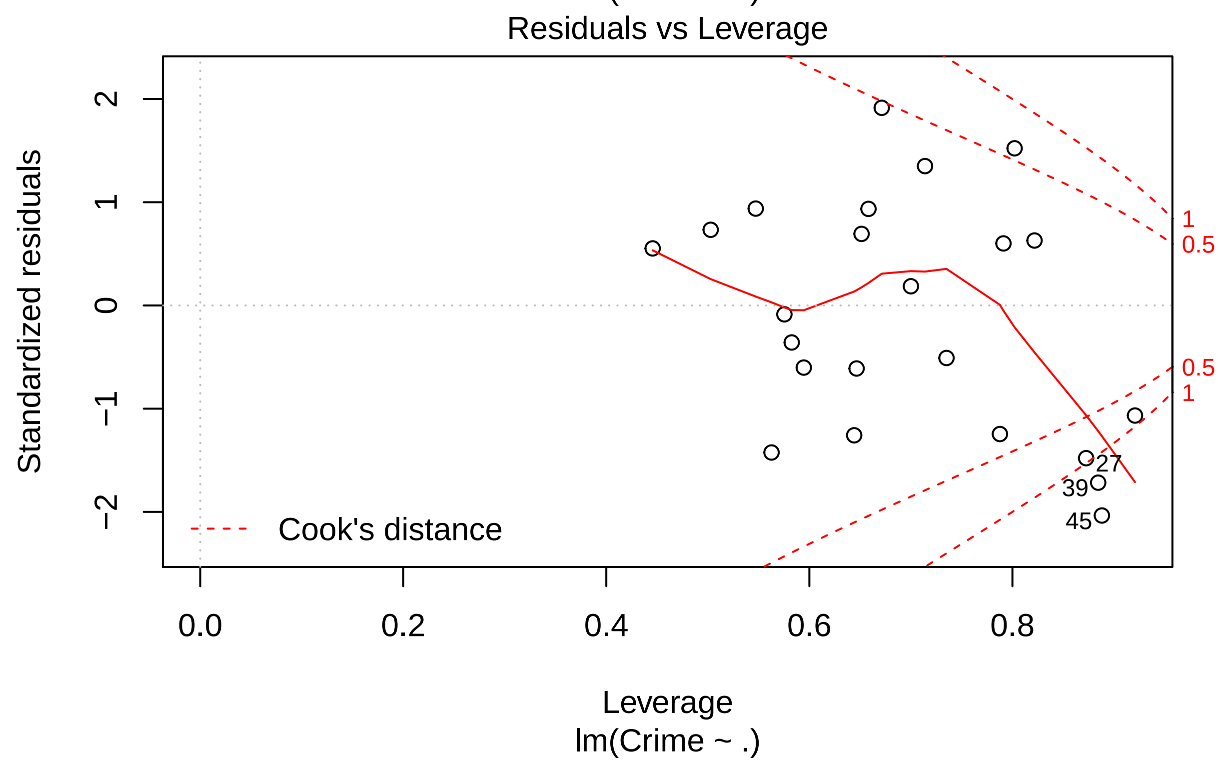
Here is the plot of our linear regression on leaf 2 of our pruned tree.

```r
plot(pruned_tree2)
```


Residuals vs Fitted
Residuals
150
100
50
0
−100
400
500
600
700
800
900
1000
41
14
10
Fitted values
lm(Crime ~ .)


Normal Q−Q
Standardized residuals
2
1
0
−1
−2
41
39
45
−2
−1
0
1
2
Theoretical Quantiles
lm(Crime ~ .)





Using a regeession tree does not yield any improvement on our linear regression model from week 5 Homework. I will now use a random forest to see what those results yield. We will set our number of predictors to 4.

```
numpred <- 4
data_forest <- randomForest(Crime~., data=data, mtry=numpred, importance=TRUE)
data_forest
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = data, mtry = numpred,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 82570.84
##                     % Var explained: 43.6
```

```
data_forest_predicted <- predict(data_forest)
ssr <- sum((data_forest_predicted-data$Crime)^2)
sst <- sum((data$Crime - mean(data$Crime))^2)
r2 <- 1 - ssr/sst
r2
```

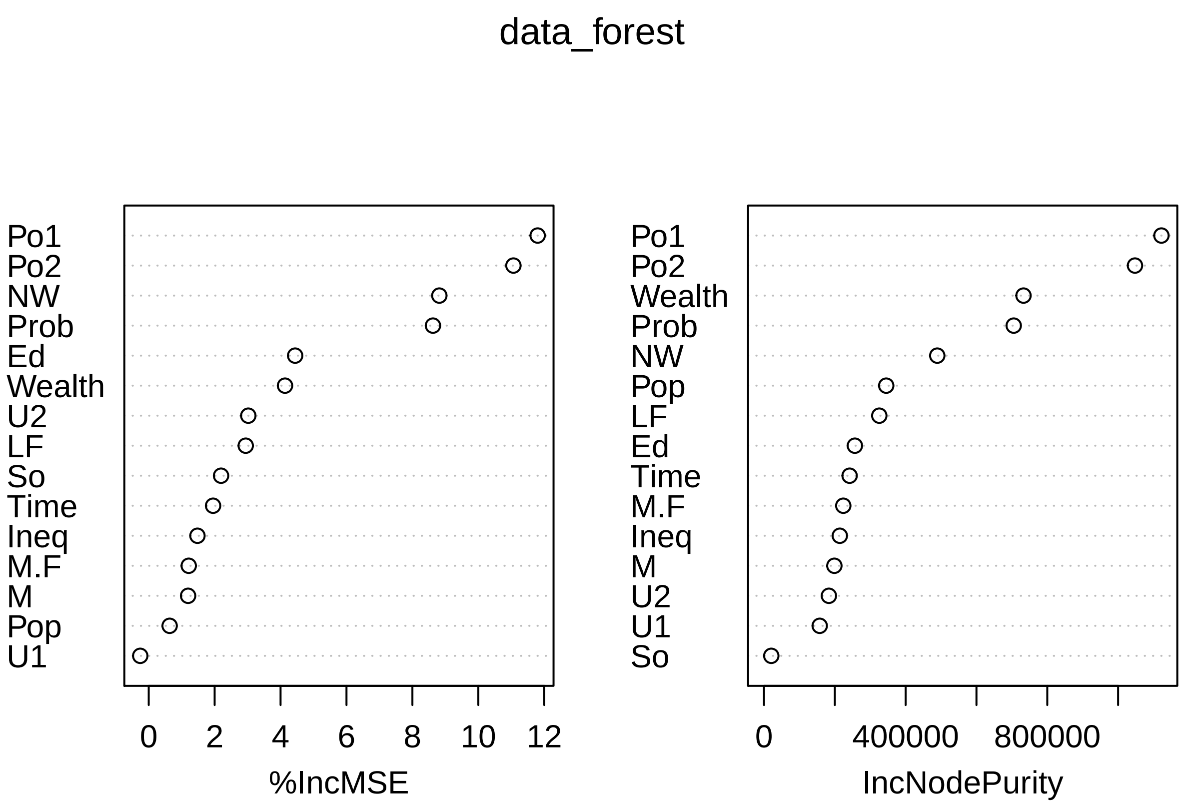
```
## [1] 0.436002
```

Our r^2 using a random forest is also lower than the original linear regression model we used in Homework 5. Here is the importance of our variables ranked and plotted.

```
importance(data_forest)
```

```
##           %IncMSE IncNodePurity
## M       1.1939367     198721.92
## So      2.1944544      20604.85
## Ed      4.4419322     256659.82
## Po1    11.7996620    1122296.45
## Po2    11.0643271    1047627.33
## LF      2.9429189     325612.07
## M.F     1.2132174     223867.07
## Pop     0.6347580     345176.64
## NW      8.8162411     489285.99
## U1     -0.2577539     157350.79
## U2      3.0203457     183263.89
## Wealth  4.1350706     732857.96
## Ineq    1.4781917     214109.70
## Prob    8.6229153     705205.82
## Time    1.9522835     241779.35
```

```
varImpPlot(data_forest)
```



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.

In my position at a FinTech Company analyzing insider trading data, we have numerous binary classifiers for which logit regression is an obvious choice. Some predictors are isthe insider a CEO, is the insider a CFO, does the insider have a 10b5-1 selling plan, are the insiders options expiring within 3 months, etc.

Question 10.3

1. Using 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. 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.

```
data_gc <- read.table("/Users/ralbright/Dropbox/ISYE6501/week7/homework/germancredit.txt", sep=" ")
```

Lets take a peek at the Head.

```r
table <- xtable(head(data_gc))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

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

Then the Tail

```r
table <- xtable(tail(data_gc))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
995	A14	12	A32	A40	2390	A65	A75	4	A93	A101	3	A123	50	A143	A152	1	A173	1	A192	A201	1
996	A14	12	A32	A42	1736	A61	A74	3	A92	A101	4	A121	31	A143	A152	1	A172	1	A191	A201	1
997	A11	30	A32	A41	3857	A61	A73	4	A91	A101	4	A122	40	A143	A152	1	A174	1	A192	A201	1
998	A14	12	A32	A43	804	A61	A75	4	A93	A101	4	A123	38	A143	A152	1	A173	1	A191	A201	1
999	A11	45	A32	A43	1845	A61	A73	4	A93	A101	4	A124	23	A143	A153	1	A173	1	A192	A201	2
1000	A12	45	A34	A41	4576	A62	A71	3	A93	A101	4	A123	27	A143	A152	1	A173	1	A191	A201	1

Here is the summary of the data

```r
table <- xtable(summary(data_gc))
print(table, type='latex', comment=FALSE, scalebox='0.75')
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
X	A11:274	Min. : 4.0	A30: 40	A43 :280	Min. : 250	A61:603	A71: 62	Min. :1.000	A91: 50	A101:907	Min. :1.000	A121:282
X.1	A12:269	1st Qu.:12.0	A31: 49	A40 :234	1st Qu.: 1366	A62:103	A72:172	1st Qu.:2.000	A92:310	A102: 41	1st Qu.:2.000	A122:232
X.2	A13: 63	Median :18.0	A32:530	A42 :181	Median : 2320	A63: 63	A73:339	Median :3.000	A93:548	A103: 52	Median :3.000	A123:332
X.3	A14:394	Mean :20.9	A33: 88	A41 :103	Mean : 3271	A64: 48	A74:174	Mean :2.973	A94: 92		Mean :2.845	A124:154
X.4		3rd Qu.:24.0	A34:293	A49 : 97	3rd Qu.: 3972	A65:183	A75:253	3rd Qu.:4.000			3rd Qu.:4.000	
X.5		Max. :72.0		A46 : 50	Max. :18424			Max. :4.000			Max. :4.000	
X.6				(Other): 55								

In order to perform logit regression in the glm() function we need to convert our response to 0 and 1, where 0 is the positive response.

```r
data_gc$V21[data_gc$V21==1]<-0
data_gc$V21[data_gc$V21==2]<-1
```

In order to test the rosbustness of our final model, we need to split the data into training and test sets.

```r
r = nrow(data_gc)
train_set = sample(1:r, size = round(r * .8), replace = FALSE)
data_gc_train <- data_gc[train_set,]
data_gc_test <- data_gc[-train_set,]
```

We will then create our logit regression model on the training set.

```r
gc_model = glm(V21~., family=binomial(link="logit"),data=data_gc_train)
summary(gc_model)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = data_gc_train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.2610  -0.6889  -0.3564   0.7290   2.5527
##
## Coefficients:
##                Estimate  Std. Error z value       Pr(>|z|)
## (Intercept)  0.44395667  1.20758097   0.368       0.713141
## V1A12       -0.32471700  0.24149590  -1.345       0.178752
## V1A13       -1.08972741  0.47128992  -2.312       0.020765 *
## V1A14       -1.67004779  0.26136778  -6.390 0.000000000166 ***
## V2           0.02380768  0.01030427   2.310       0.020862 *
## V3A31       -0.33993451  0.60824247  -0.559       0.576244
## V3A32       -0.86991098  0.48585539  -1.790       0.073378 .
## V3A33       -1.16290156  0.53873358  -2.159       0.030882 *
## V3A34       -1.71352316  0.49997492  -3.427       0.000610 ***
## V4A41       -1.73834115  0.40809736  -4.260 0.000020477141 ***
## V4A410      -1.04298427  0.83500389  -1.249       0.211637
## V4A42       -0.67576702  0.29100968  -2.322       0.020225 *
## V4A43       -0.98945152  0.28205661  -3.508       0.000452 ***
## V4A44       -0.38164279  0.77008327  -0.496       0.620186
## V4A45       -0.37906961  0.63333717  -0.599       0.549488
## V4A46       -0.22755132  0.43204439  -0.527       0.598412
## V4A48       -1.86072648  1.20149358  -1.549       0.121459
## V4A49       -1.07384055  0.40509048  -2.651       0.008029 **
## V5           0.00015123  0.00005067   2.984       0.002842 **
## V6A62       -0.14748748  0.33290907  -0.443       0.657747
## V6A63       -0.35037002  0.42941766  -0.816       0.414546
## V6A64       -1.14997642  0.59680587  -1.927       0.053994 .
## V6A65       -0.86176803  0.29266281  -2.945       0.003234 **
## V7A72       -0.06419103  0.47837055  -0.134       0.893255
## V7A73        0.04536527  0.46153515   0.098       0.921700
## V7A74       -0.55394737  0.49355071  -1.122       0.261704
## V7A75       -0.05374045  0.45963722  -0.117       0.906924
## V8           0.32222608  0.10042519   3.209       0.001334 **
## V9A92       -0.44492857  0.43763948  -1.017       0.309317
## V9A93       -0.96720367  0.43458576  -2.226       0.026043 *
## V9A94       -0.84565794  0.51941486  -1.628       0.103504
## V10A102      0.47013153  0.50194839   0.937       0.348958
## V10A103     -0.64884724  0.44994719  -1.442       0.149288
## V11          0.04963877  0.09705479   0.511       0.609035
## V12A122      0.43908461  0.28639125   1.533       0.125236
## V12A123      0.25220784  0.27041599   0.933       0.350992
## V12A124      0.70970130  0.47213730   1.503       0.132796
## V13         -0.01166082  0.01018152  -1.145       0.252088
## V14A142     -0.12124711  0.44886607  -0.270       0.787069
## V14A143     -0.83156397  0.27030322  -3.076       0.002095 **
## V15A152     -0.43482785  0.26614954  -1.634       0.102307
## V15A153     -0.75052794  0.51876373  -1.447       0.147963
## V16          0.27204519  0.21599380   1.260       0.207848
## V17A172      0.73371312  0.78630403   0.933       0.350760
## V17A173      0.79698485  0.75825020   1.051       0.293220
## V17A174      0.98685052  0.76749097   1.286       0.198508
## V18          0.21428651  0.28446221   0.753       0.451267
## V19A192     -0.59659394  0.23090802  -2.584       0.009775 **
## V20A202     -1.31674596  0.70114166  -1.878       0.060381 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 977.38  on 799  degrees of freedom
## Residual deviance: 710.14  on 751  degrees of freedom
## AIC: 808.14
##
## Number of Fisher Scoring iterations: 5
```

```
gc_predictions = predict(gc_model)
```

The original model's AIC is 808.14. Using an r^2 of < 0.10, I determined to build a model only using variables V1, V2, V3, V4, V5, V6, V8, V9, V14, V19, and V20.

```
gc_model2 = glm(V21~V1+V2+V3+V4+V5+V6+V8+V9+V14+V19+V20, family=binomial(link="logit"),data=data_gc_tra
summary(gc_model2)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V14 +
##     V19 + V20, family = binomial(link = "logit"), data = data_gc_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2104  -0.7032  -0.3813   0.7716   2.7871
##
## Coefficients:
##                Estimate  Std. Error z value        Pr(>|z|)
## (Intercept)  1.55252473  0.72429919   2.143        0.032074 *
## V1A12       -0.40088550  0.23191275  -1.729        0.083880 .
## V1A13       -1.17256102  0.45851016  -2.557        0.010548 *
## V1A14       -1.67837602  0.25422119  -6.602 0.0000000000406 ***
## V2           0.02170656  0.00969445   2.239        0.025151 *
## V3A31       -0.65612141  0.57407303  -1.143        0.253070
## V3A32       -1.19246700  0.45956819  -2.595        0.009466 **
## V3A33       -1.27222632  0.52505031  -2.423        0.015391 *
## V3A34       -1.87985250  0.48222627  -3.898 0.0000968787924 ***
## V4A41       -1.60751410  0.39154847  -4.106 0.0000403388399 ***
## V4A410      -0.88875421  0.78135768  -1.137        0.255351
## V4A42       -0.57169476  0.27845301  -2.053        0.040062 *
## V4A43       -1.09528771  0.27120535  -4.039 0.0000537730301 ***
## V4A44       -0.50040551  0.75236039  -0.665        0.505978
## V4A45       -0.37553040  0.60595289  -0.620        0.535432
## V4A46       -0.06525157  0.41800164  -0.156        0.875951
## V4A48       -1.92272929  1.21569886  -1.582        0.113745
## V4A49       -1.13096509  0.39296602  -2.878        0.004002 **
## V5           0.00016990  0.00004789   3.548        0.000388 ***
## V6A62        0.02136499  0.31461030   0.068        0.945858
## V6A63       -0.35831244  0.42044216  -0.852        0.394088
## V6A64       -1.04955765  0.56017354  -1.874        0.060982 .
## V6A65       -0.84975197  0.28037525  -3.031        0.002439 **
## V8           0.33759191  0.09581389   3.523        0.000426 ***
## V9A92       -0.40824228  0.41746402  -0.978        0.328119
```

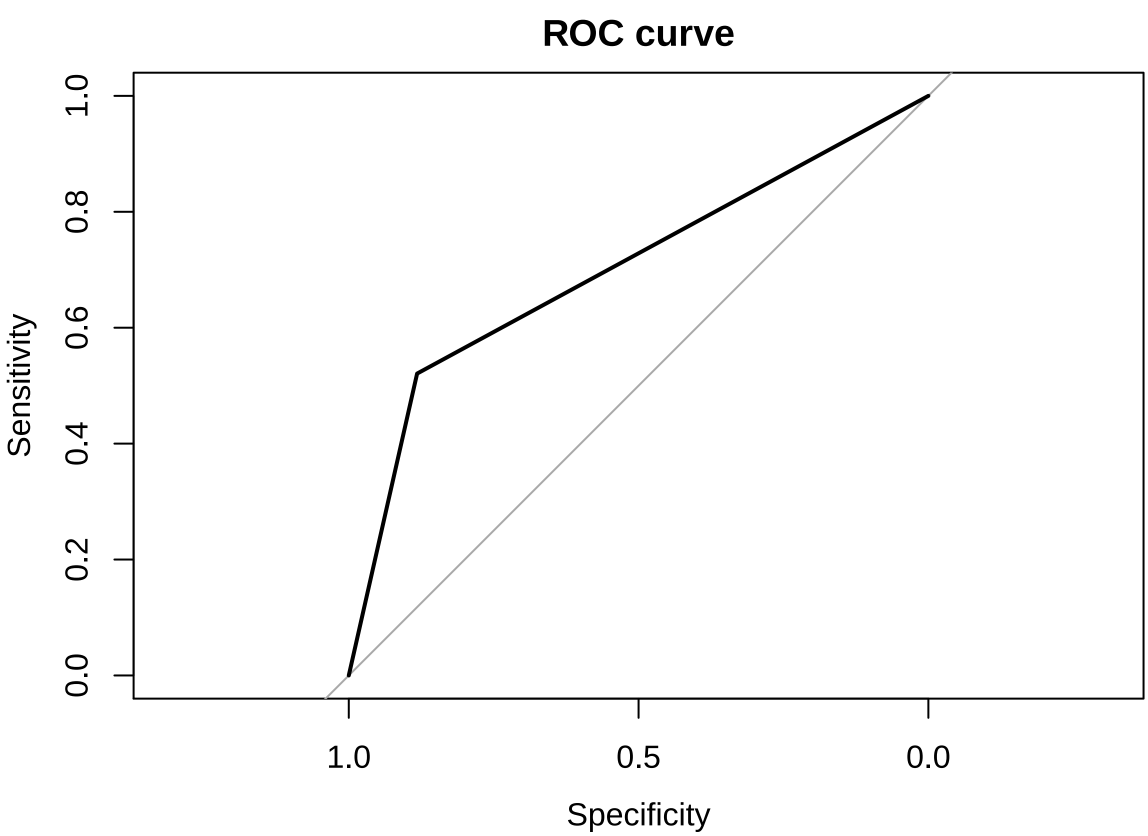
```
## V9A93        -0.97627760  0.41155737  -2.372        0.017685 *
## V9A94        -0.84475075  0.49849725  -1.695        0.090152 .
## V14A142      -0.11011932  0.42935173  -0.256        0.797582
## V14A143      -0.83258977  0.25832903  -3.223        0.001269 **
## V19A192      -0.53171496  0.20666392  -2.573        0.010087 *
## V20A202      -1.33182885  0.67586039  -1.971        0.048773 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 977.38  on 799  degrees of freedom
## Residual deviance: 730.52  on 769  degrees of freedom
## AIC: 792.52
##
## Number of Fisher Scoring iterations: 5
```

The revised model has an AIC of 792.52 I then calculated the predictions from the revised model.

```
gc_predictions2 <- predict(gc_model2, data_gc_train, type="response")
roc2 = roc(data_gc_train$V21,round(gc_predictions2))
auc2 = auc(data_gc_train$V21,round(gc_predictions2))
auc2
```

```
## [1] 0.7014881
```

```
plot(roc2,main="ROC curve")
```



The AUC for the revised model is 0.7014881. Lets see how this model peforms on our testing set.

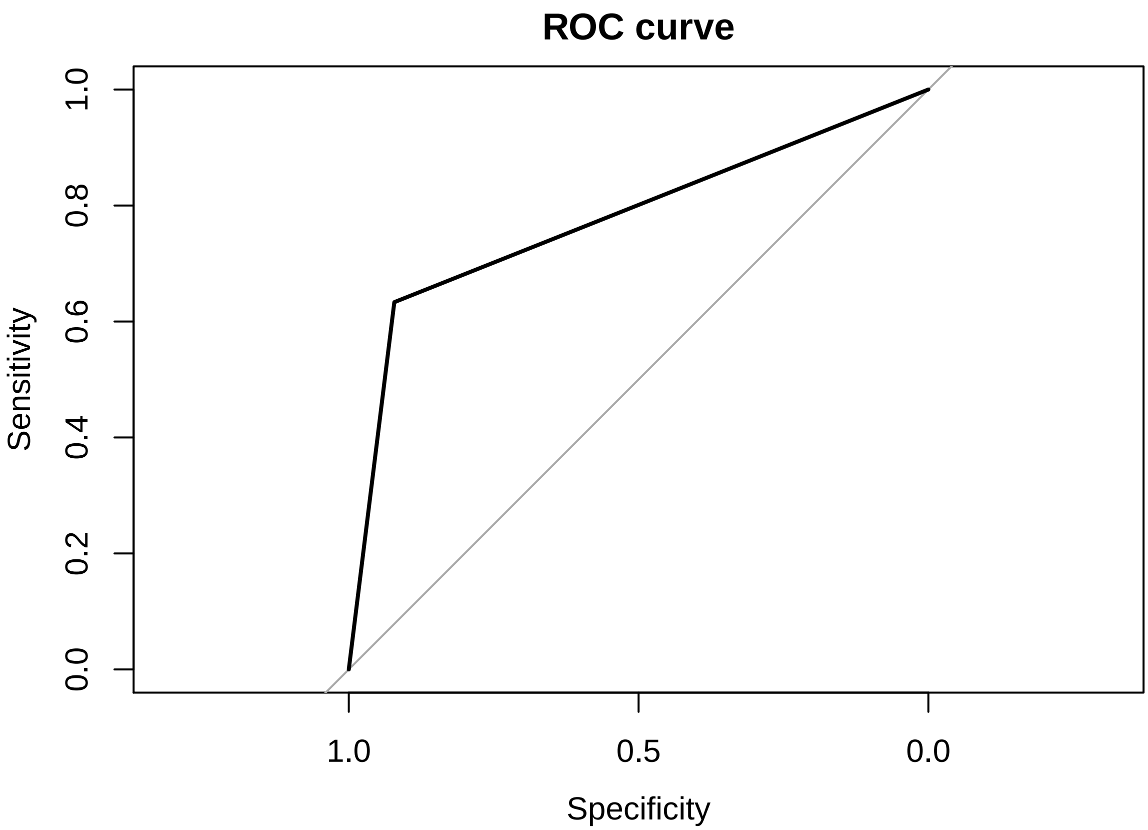
```r
gc_model3 = glm(V21~V1+V2+V3+V4+V5+V6+V8+V9+V14+V19+V20, family=binomial(link="logit"),data=data_gc_tes
summary(gc_model3)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V14 +
##     V19 + V20, family = binomial(link = "logit"), data = data_gc_test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0621  -0.6502  -0.2318   0.3710   2.8629
##
## Coefficients:
##                  Estimate    Std. Error z value  Pr(>|z|)
## (Intercept)   -1.92704595    1.70657463  -1.129    0.2588
## V1A12         -0.65444885    0.61206328  -1.069    0.2850
## V1A13         -0.98302013    0.84384278  -1.165    0.2440
## V1A14         -2.75139105    0.65382447  -4.208 0.0000257 ***
## V2             0.06335050    0.02827923   2.240    0.0251 *
## V3A31          3.10269924    2.04537719   1.517    0.1293
## V3A32          0.81696814    1.11057901   0.736    0.4620
## V3A33          0.67839573    1.27723714   0.531    0.5953
## V3A34         -0.50732119    1.17849806  -0.430    0.6668
## V4A41         -1.18105347    1.21180440  -0.975    0.3297
## V4A410       -19.42983093 2353.40550866  -0.008    0.9934
## V4A42         -1.07856724    0.68811282  -1.567    0.1170
## V4A43         -0.88922783    0.60012045  -1.482    0.1384
## V4A45          1.57845756    1.30701370   1.208    0.2272
## V4A46          3.75327531    1.57067928   2.390    0.0169 *
## V4A48        -13.95077925 3956.18050347  -0.004    0.9972
## V4A49          0.40217276    0.84491392   0.476    0.6341
## V5             0.00002275    0.00012287   0.185    0.8531
## V6A62         -1.74094084    0.73060435  -2.383    0.0172 *
## V6A63          1.42050792    1.34281568   1.058    0.2901
## V6A64         -2.72384138    1.42554598  -1.911    0.0560 .
## V6A65         -1.80549700    0.70544837  -2.559    0.0105 *
## V8             0.32443918    0.21751175   1.492    0.1358
## V9A92         -0.13660611    1.07268169  -0.127    0.8987
## V9A93         -1.36918915    1.08211561  -1.265    0.2058
## V9A94          1.00496879    1.23726943   0.812    0.4166
## V14A142        1.49046478    1.59457304   0.935    0.3499
## V14A143        0.68988443    0.68503231   1.007    0.3139
## V19A192        0.75767966    0.50501842   1.500    0.1335
## V20A202      -17.26820495 1241.00183883  -0.014    0.9889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 244.35  on 199  degrees of freedom
## Residual deviance: 148.57  on 170  degrees of freedom
## AIC: 208.57
##
## Number of Fisher Scoring iterations: 16
```

```r
gc_predictions3 <- predict(gc_model3, data_gc_test, type="response")
roc3 = roc(data_gc_test$V21,round(gc_predictions3))
auc3 = auc(data_gc_test$V21,round(gc_predictions3))
auc3
```

```
## [1] 0.777381
```

```r
plot(roc3,main="ROC curve")
```



Our models AIC on the testing set is 208.57. The AUC is 0.777381. Our resulting models coefficients are

```r
gc_model3$coefficients
```

```
##    (Intercept)          V1A12          V1A13          V1A14
##  -1.92704594789  -0.65444884701  -0.98302013311  -2.75139105467
##             V2          V3A31          V3A32          V3A33
##   0.06335050178   3.10269923522   0.81696813530   0.67839573353
##          V3A34          V4A41         V4A410          V4A42
##  -0.50732119474  -1.18105346847 -19.42983093104  -1.07856723722
##          V4A43          V4A45          V4A46          V4A48
##  -0.88922783230   1.57845756359   3.75327531459 -13.95077924899
##          V4A49             V5          V6A62          V6A63
##   0.40217276157   0.00002275027  -1.74094084340   1.42050791514
##          V6A64          V6A65             V8          V9A92
##  -2.72384137514  -1.80549700205   0.32443917998  -0.13660611255
##          V9A93          V9A94        V14A142        V14A143
##  -1.36918914994   1.00496879493   1.49046477570   0.68988442628
##        V19A192        V20A202
##   0.75767966274 -17.26820494916
```

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.

Let's loop through the thresholds of 0.01 through 0.99 and solve for accuracy and lowest cost.

```
# high cost bad point as good = 5
# low cost good point as bad = 1
costs <- matrix(, 99, ncol = 2)
accuracies <- matrix(, 99, ncol = 2)
i=1
for (threshold in seq(.01, .99, .01))
{

  conf_matrix <- confusion.matrix(data_gc_test$V21, gc_predictions3, threshold = threshold)
  accuracies[i, 1] <- threshold
  accuracies[i,2] <- conf_matrix[2, 1] + conf_matrix[1, 2]
  cost = conf_matrix[2, 1] * 5 + conf_matrix[1, 2] * 1
  costs[i, 1] <- threshold
  costs[i, 2] <- cost
  i <- i + 1
}
```

Our accuracies are as follows.

```
#mininmized misclassification unweighted
acc_thresh = accuracies[which.min(accuracies[, 2]), 1]
acc_thresh
```

```
## [1] 0.53
```

```
acc_cost = accuracies[which.min(accuracies[, 2]), 2]
acc_cost
```

```
## [1] 28
```

The best threshold for accuracy is on our training set is 0.53, the unweighted cost is 28.

Solving for the minimized cost we get the following.

```
#miniminized misclassification threshold and cost
wgt_thresh = costs[which.min(costs[, 2]), 1]
wgt_thresh
```

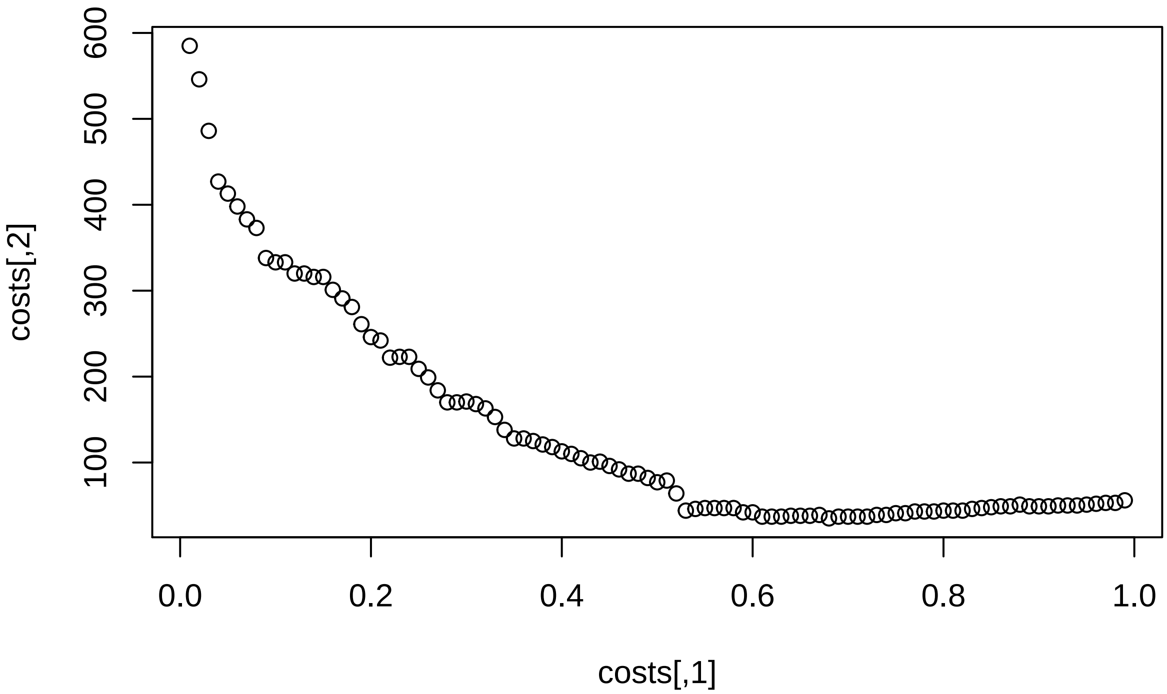
```
## [1] 0.68
```

```
wgt_cost = costs[which.min(costs[, 2]), 2]
wgt_cost
```

```
## [1] 35
```

The best threshold on our test set for minimizing our cost is 0.68, with a minimized cost of 35. Below is a chart of the associated costs vs their thresholds.

```
plot(costs)
```



600
500
400
300
200
100
costs[,2]
0.0
0.2
0.4
0.6
0.8
1.0
costs[,1]