Question 10.1

In [1]:
```
#Setting parts
set.seed(10)
install.packages('tree')
library(randomForest)
library(caret)
library(tree)
```

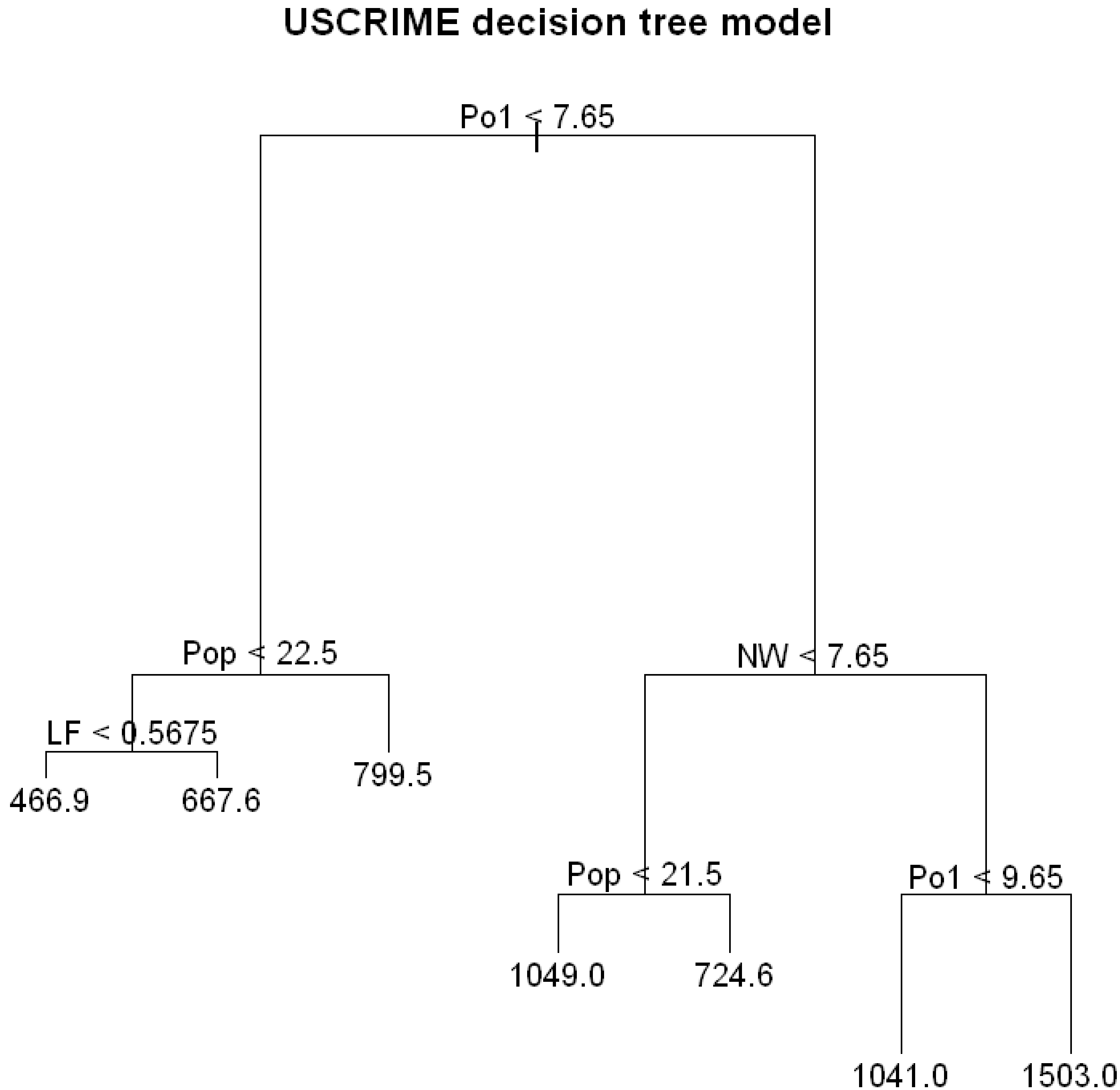
```
package 'tree' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ashka\AppData\Local\Temp\RtmpoteG1R\downloaded_packages

randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang

Attaching package: 'ggplot2'

The following object is masked from 'package:randomForest':

    margin
```

In [2]:
```
#Read the data
data_raw<-read.table(file = 'C:/Users/ashka/Dropbox/GitHub/ISYE6501_Analytics_Modelling/HW7/uscrime.txt', hea
der=TRUE)
```

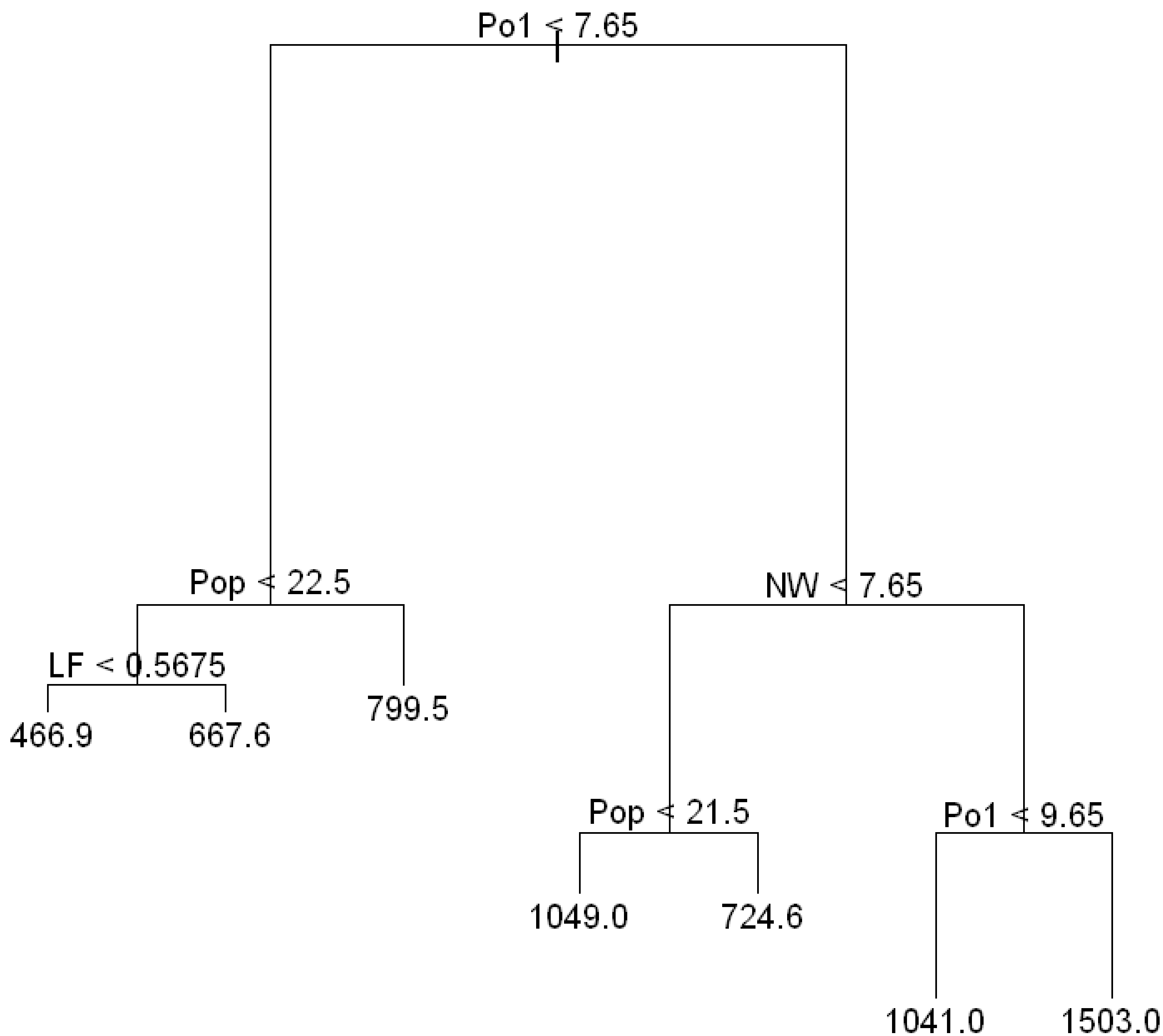
In [3]:
```
#Build the model
# crime_tree_model<-tree(Crime ~. , data=uscrime)
crime_tree_model <- tree(formula = Crime ~. , data = data_raw)
summary(crime_tree_model)
```

```
Regression tree:
tree(formula = Crime ~ ., data = data_raw)
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
```

In [4]:
```
# Test method to check the error rate
plot(crime_tree_model)
text(crime_tree_model)
title('USCRIME decision tree model')
```


USCRIME decision tree model
Po1 < 7.65
Pop < 22.5
NW < 7.65
LF < 0.5675
799.5
466.9
667.6
Pop < 21.5
Po1 < 9.65
1049.0
724.6
1041.0
1503.0

In [5]:
```
# prune the tree
desired_nodes<-7
tree_prune<-prune.tree(tree = crime_tree_model,best = desired_nodes)
plot(tree_prune)
text(tree_prune)
```


Po1 < 7.65
Pop < 22.5
NW < 7.65
LF < 0.5675
799.5
466.9
667.6
Pop < 21.5
Po1 < 9.65
1049.0
724.6
1041.0
1503.0

In [6]:
```
summary(tree_prune)
```

```
Regression tree:
tree(formula = Crime ~ ., data = data_raw)
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
```

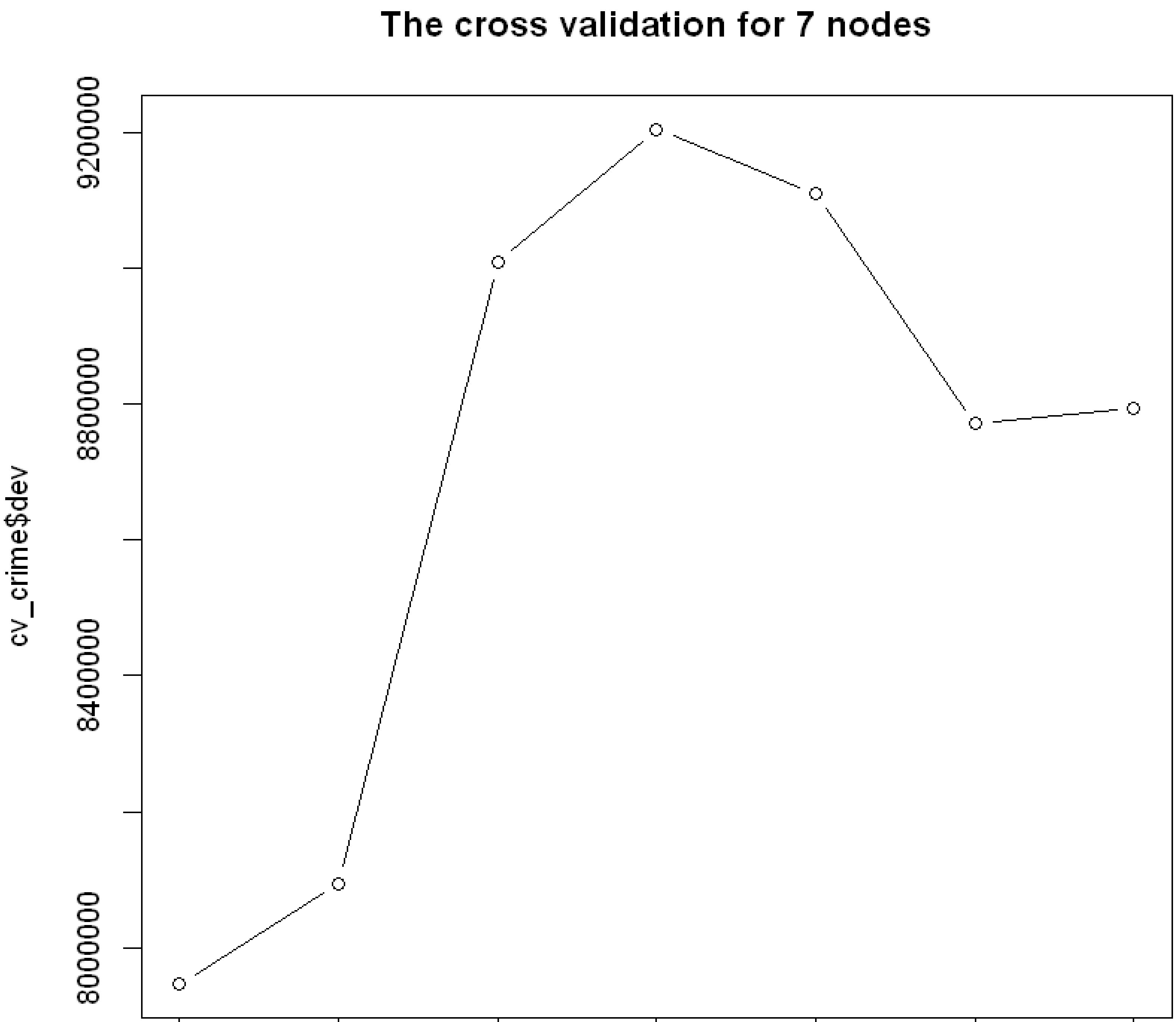
In [7]:
```
prune.tree(crime_tree_model)$dev
```

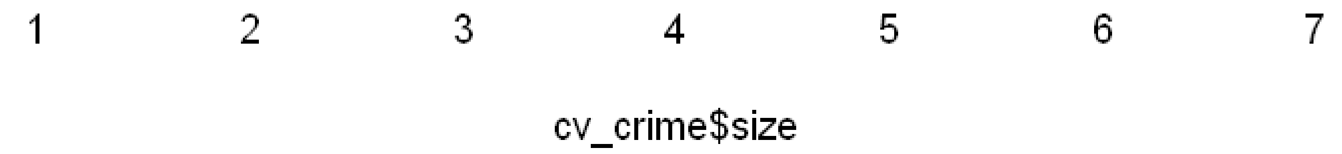
1895721.65941558 2013256.60227273 2276669.50227273 2632631.25326087 3364043.3068323 4383405.97826087 6880927.65957447

In [8]:
```
prune.tree(crime_tree_model)$size
```

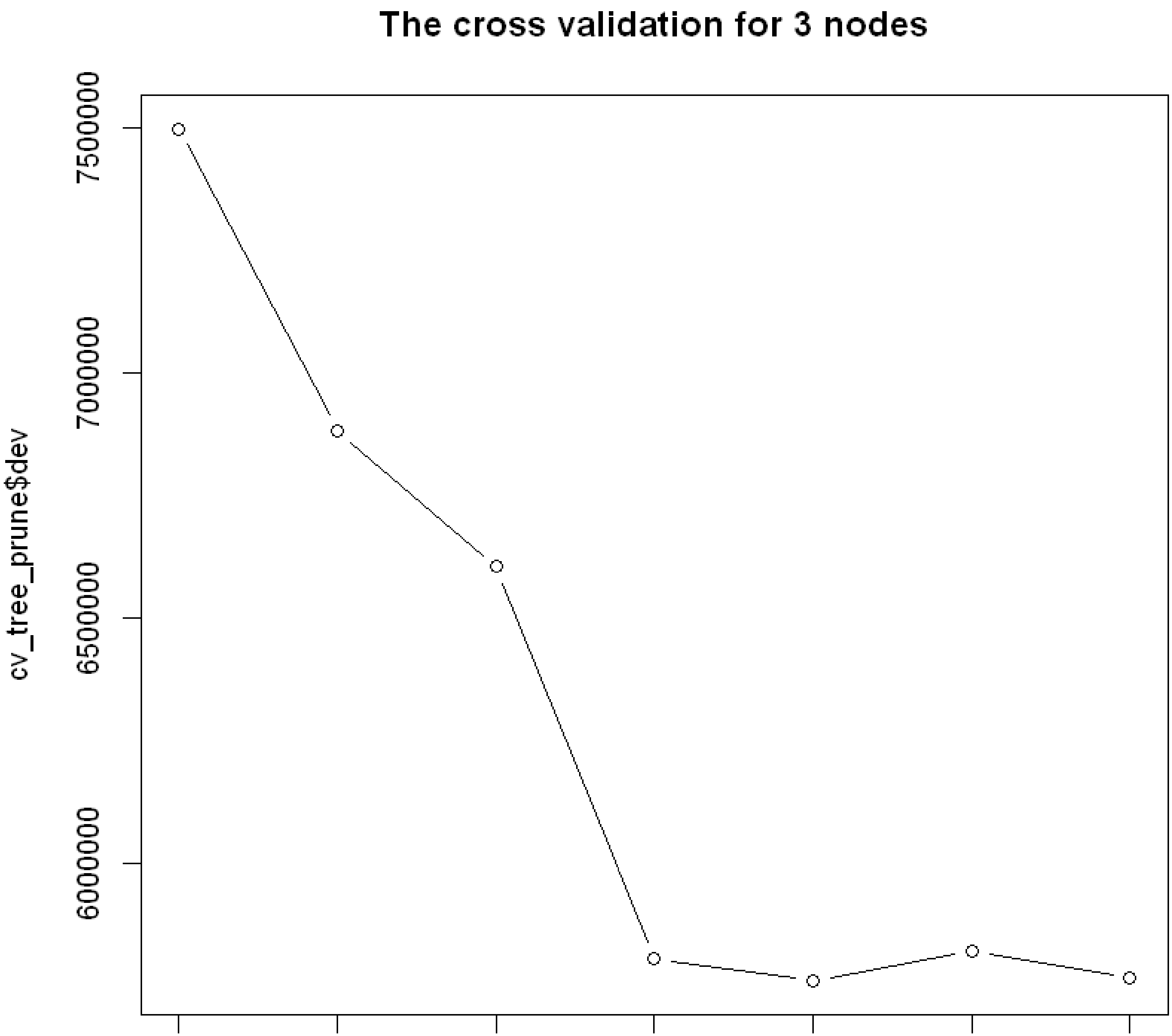
7 6 5 4 3 2 1

In [9]:
```
cv_crime<-cv.tree(crime_tree_model)
plot(cv_crime$size,cv_crime$dev,type='b')
title('The cross validation for 7 nodes')
```


The cross validation for 7 nodes
cv_crime$dev
9200000
8800000
8400000
8000000


1
2
3
4
5
6
7
cv_crime$size

In [10]:
```
cv_tree_prune<-cv.tree(tree_prune)
plot(cv_tree_prune$size,cv_tree_prune$dev,type='b')
title('The cross validation for 3 nodes')
```


The cross validation for 3 nodes
cv_tree_prune$dev
7500000
7000000
6500000
6000000

As per the above graphs it is possible to understand that the 7 node plot is providing the suitable answer comparing with the 3 nodes.

In [11]:
```
#calculating the quality of the fit by calculating the residual sum squared and dividing it by the true skill statistics
crime_tree_prediction<-predict(tree_prune,data=data_raw[,1:15])
RSS <- sum((crime_tree_prediction - data_raw[,16])^2)
TSS <- sum((data_raw[,16] - mean(data_raw[,16]))^2)
R2 <- 1 - RSS/TSS
```

As it is difficult to check all the node sizes manually, it is useful to apply a loop and check the different accuracy for the various node numbers.

In [12]:
```
total_results<-data.frame(matrix(nrow = 5,ncol = 2))
colnames(total_results)<-c('NodeSize','R2')
i=1
for (desired_nodes in 3:7){
    crime_tree_model <- tree(formula = Crime ~. , data = data_raw)
    tree_prune<-prune.tree(tree = crime_tree_model,best = desired_nodes)

    predict <- predict(tree_prune,data=data_raw[,1:15])
    RSS <- sum((predict - data_raw[,16])^2)
    TSS <- sum((data_raw[,16] - mean(data_raw[,16]))^2)
    R2 <- 1 - RSS/TSS
    total_results[i,1]<-desired_nodes
    total_results[i,2]<-R2
    i=i+1
}
```

It sows that increasing the nodes to 7 can greatly improve the performance.

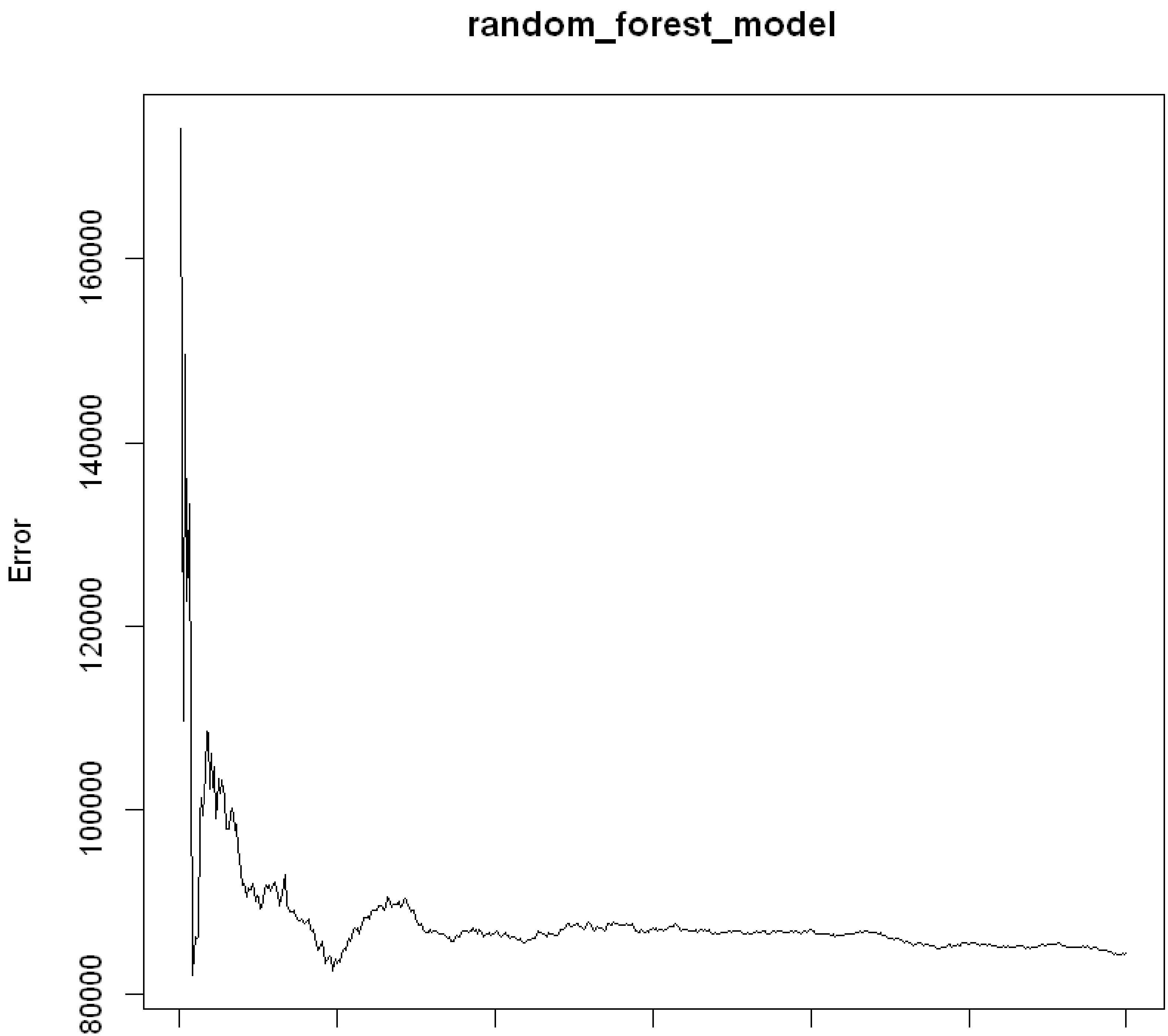
Part b: Random forest

Part B of the assignment will focus on the random forest and compare the results with the decision tree

In [13]:
```
#Use the loaded data in the tree section of the analysis
```

In [14]:
```
number_features <- 1 + log(ncol(data_raw))
random_forest_model <- randomForest(Crime~., data = data_raw, mtry = number_features, importance = T, ntree = 600)
```

In [15]:
```
plot(random_forest_model)
```

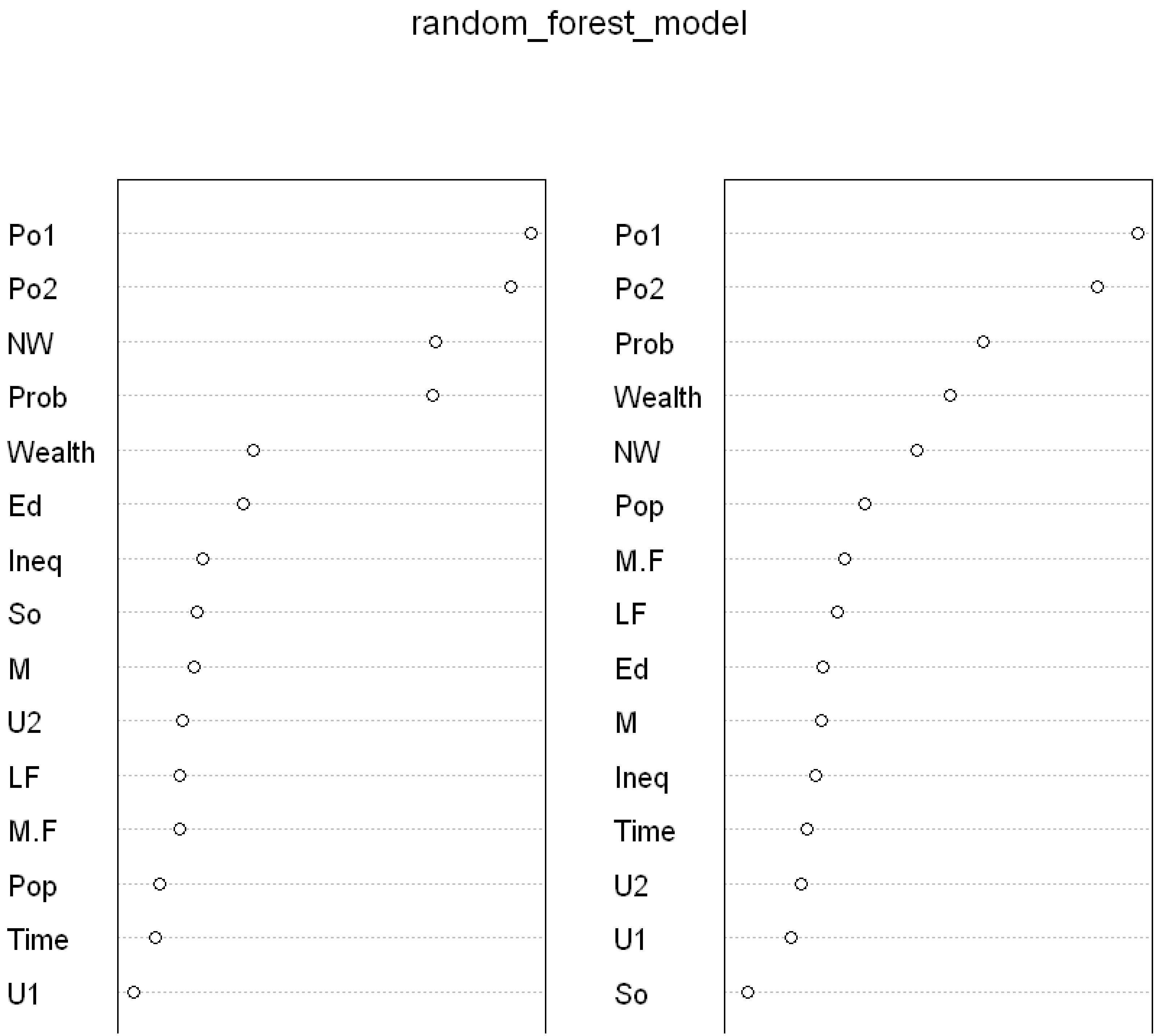

random_forest_model
Error
160000
140000
120000
100000
80000

In [16]:
```
y_predicted <- predict(random_forest_model)
y_reference<-data_raw$Crime
RSS<-sum((y_predicted-y_reference)^2)
TSS<-sum((y_reference-mean(y_reference))^2)
RSQ<-1-RSS/TSS
```

In [17]:
```
importance(random_forest_model)
```

	%IncMSE	IncNodePurity
M	1.9443264	237795.75
So	2.0437821	23196.77
Ed	3.6304361	244507.06
Po1	13.5325017	1170350.26
Po2	12.8656628	1053817.43
LF	1.4690139	287869.95
M.F	1.4650766	307339.41
Pop	0.7565604	368026.69
NW	10.2800395	519784.66
U1	-0.1478492	147789.51
U2	1.5730397	178034.63
Wealth	4.0173065	619315.42
Ineq	2.2614113	223651.39
Prob	10.1595130	716718.24
Time	0.5867046	198041.34

In [18]:
```
varImpPlot(random_forest_model)
```


random_forest_model
Po1
Po2
NW
Prob
Wealth
Ed
Ineq
So
M
U2
LF
M.F
Pop
Time
U1
Po1
Po2
Prob
Wealth
NW
Pop
M.F
LF
Ed
M
Ineq
Time
U2
U1
So

Q 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.

My job involves predicting the quality of the grain by using different data points collected using the internet of things sensors. I need to use the grain condition such as initial weight, protein level, initial moisture, crushing results and possible storage time as input to the developed algorithms and detremine the silo location for the optimum storage periods.

Q 10.3

In [19]:
```
# Loading the data
raw_data<-read.table('C:/Users/ashka/Dropbox/GitHub/ISYE6501_Analytics_Modelling/HW7/german.txt', header = FA
LSE)
```

In [20]:
```
head(raw_data)
```

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

In [21]:
```
#Setting the index value to 0 and 1
raw_data['V21'][raw_data['V21']==2]<-0
```

In [22]:
```
# we need to divide the data into train and test before moving forward with the modelling part.

data_sampling <- sample(1:nrow(raw_data), size = round(0.8*nrow(raw_data)))
train_data <- raw_data[data_sampling,]
test_data <- raw_data[-data_sampling,]
```

In [23]:
```
#Modelling part based on the logistic regression

machine_learning_model<- glm(formula=V21 ~.,family = binomial(link='logit'),data = train_data)
summary(machine_learning_model)
```

```
Call:
glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7645  -0.6282   0.3493   0.6822   2.3979

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.848e-02  1.182e+00   0.033 0.974036
V1A12        1.397e-01  2.481e-01   0.563 0.573295
V1A13        8.700e-01  4.203e-01   2.070 0.038476 *
V1A14        1.556e+00  2.622e-01   5.935 2.94e-09 ***
V2          -2.780e-02  1.064e-02  -2.612 0.008994 **
V3A31       -2.561e-01  6.199e-01  -0.413 0.679534
V3A32        8.096e-01  4.914e-01   1.647 0.099456 .
V3A33        1.383e+00  5.394e-01   2.564 0.010334 *
V3A34        1.807e+00  4.978e-01   3.629 0.000284 ***
V4A41        1.674e+00  4.439e-01   3.770 0.000163 ***
V4A410       1.325e+00  9.077e-01   1.460 0.144379
V4A42        9.020e-01  2.958e-01   3.050 0.002292 **
V4A43        8.490e-01  2.836e-01   2.994 0.002757 **
V4A44        4.583e-01  7.821e-01   0.586 0.557870
V4A45        4.471e-01  6.022e-01   0.742 0.457865
V4A46       -2.204e-01  4.469e-01  -0.493 0.621878
V4A48        2.079e+00  1.250e+00   1.663 0.096313 .
V4A49        1.054e+00  3.834e-01   2.750 0.005952 **
V5          -1.424e-04  5.068e-05  -2.810 0.004957 **
V6A62        5.686e-01  3.345e-01   1.700 0.089182 .
V6A63        1.059e-01  4.401e-01   0.241 0.809889
V6A64        1.490e+00  6.110e-01   2.439 0.014715 *
V6A65        9.638e-01  2.966e-01   3.250 0.001156 **
V7A72        2.277e-01  4.968e-01   0.458 0.646755
V7A73        4.223e-01  4.865e-01   0.868 0.385333
V7A74        1.300e+00  5.296e-01   2.455 0.014089 *
V7A75        6.840e-01  4.872e-01   1.404 0.160373
V8          -3.718e-01  1.022e-01  -3.638 0.000274 ***
V9A92        2.386e-01  4.416e-01   0.540 0.589063
V9A93        8.170e-01  4.374e-01   1.868 0.061780 .
V9A94        1.749e-01  5.234e-01   0.334 0.738284
V10A102     -8.887e-01  4.640e-01  -1.915 0.055470 .
V10A103      8.098e-01  4.928e-01   1.643 0.100340
V11         -6.733e-02  9.801e-02  -0.687 0.492099
```

```
V12A122     -2.814e-01  2.849e-01  -0.987 0.323424
V12A123     -2.614e-01  2.673e-01  -0.978 0.328229
V12A124     -6.162e-01  4.716e-01  -1.307 0.191356
V13          1.822e-02  1.051e-02   1.734 0.082971 .
V14A142      1.391e-01  4.517e-01   0.308 0.758055
V14A143      5.559e-01  2.752e-01   2.020 0.043375 *
V15A152      4.949e-01  2.627e-01   1.884 0.059570 .
V15A153      6.797e-01  5.317e-01   1.278 0.201118
V16         -3.766e-01  2.179e-01  -1.728 0.083918 .
V17A172     -8.021e-01  7.422e-01  -1.081 0.279859
V17A173     -7.326e-01  7.167e-01  -1.022 0.306721
V17A174     -6.265e-01  7.213e-01  -0.869 0.385077
V18         -4.663e-01  2.807e-01  -1.661 0.096687 .
V19A192      3.469e-01  2.353e-01   1.474 0.140358
V20A202      1.212e+00  6.614e-01   1.832 0.066967 .
---
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: 701.08  on 751  degrees of freedom
AIC: 799.08

Number of Fisher Scoring iterations: 5
```

In [24]:
```
#This part will focus on the model prediction

logistic_model_prediction<-predict(object = machine_learning_model,data = test_data,type = 'response')
# yhat1 <- as.integer(yhat_logit > 0.5)
logistic_model_prediction_results<-as.integer(logistic_model_prediction > 0.5)
logistic_model_prediction_results
```

```
1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1
0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1 1 1
1 0 1 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1
0 1 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1
1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1
1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1
0 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1
1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0
1 1 1 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1
1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1
1 0 1 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1
1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1
1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
1 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1
1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1
1 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 0 1 0
```

In [25]:
```
# End of the document
```