Homework 7

Anonymous

2024-02-27

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).

Part a

```
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

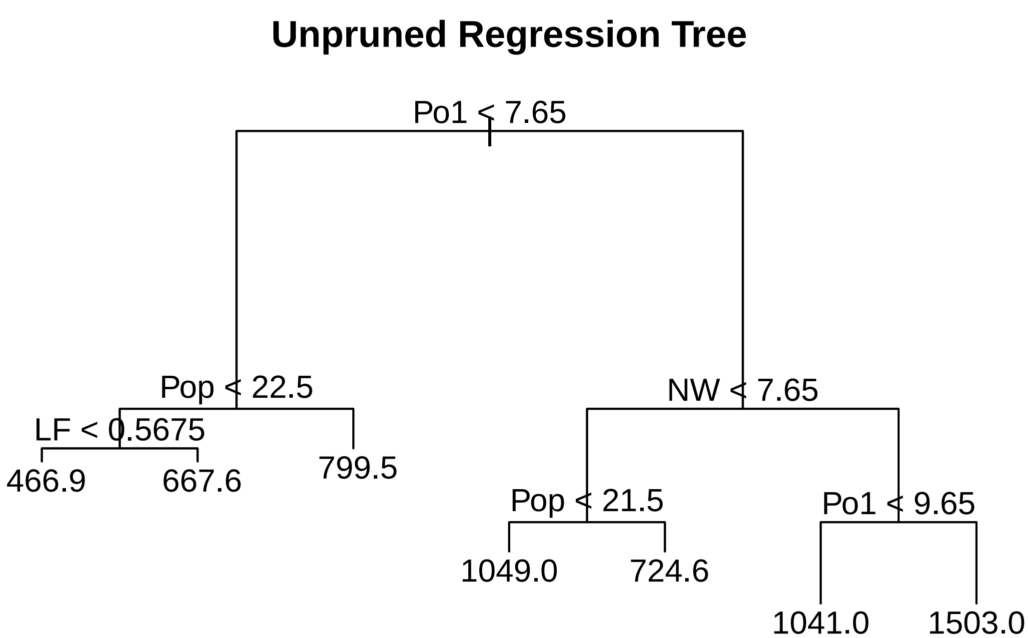
```
##
## Attaching package: 'randomForest'
```

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

```
library(tree)
crime_data <- read.table(file= "C:\\Users\\sheya\\OneDrive\\Desktop\\uscrime.txt",
                         header = TRUE)
test <- 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)

tree_model <- tree(Crime~., data = crime_data)
summary(tree_model)
```

```
##
## Regression tree:
## tree(formula = Crime ~ ., data = crime_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
```

```r
#Visulaization
plot(tree_model)
text(tree_model, pretty = 0)
title(main = "Unpruned Regression Tree")
```

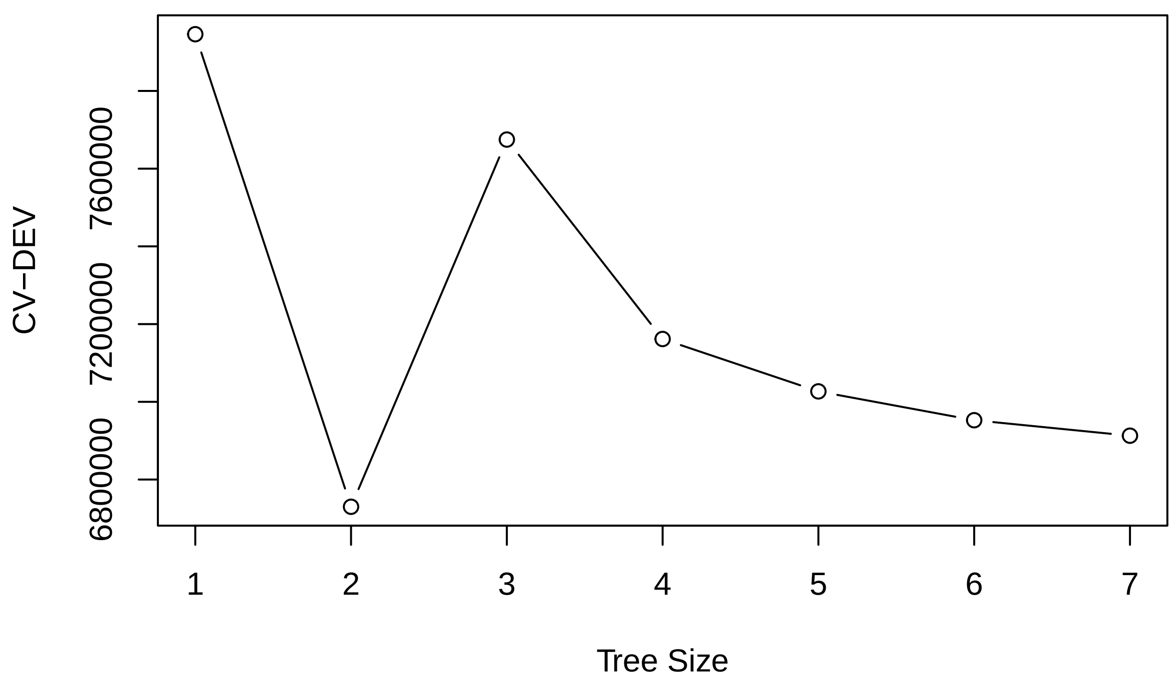



```r
#Prediction on the unpruned model
pred_un <- predict(tree_model, crime_data[,1:15])

#Calculating R2 for the unpruned model
SSE <- sum((pred_un - crime_data[16])^2)
SST <- sum((crime_data$Crime - mean(crime_data$Crime))^2)
R2 <- 1 - SSE/SST

#Classification trees allow the use of cross-validation to select a good
```

```r
#prunning of the tree.
set.seed(18)
tree_cv = cv.tree(tree_model)
summary(tree_cv)
```

```
##        Length Class  Mode
## size   7      -none- numeric
## dev    7      -none- numeric
## k      7      -none- numeric
## method 1      -none- character
```

```r
plot(tree_cv$size, tree_cv$dev, type = "b",
     xlab = "Tree Size", ylab = "CV-DEV")
```




```r
#The plot shows deviation which is a measurement of the errors from 1 to 7.
#A tree size of 5 and 6 with close proximity of their deviation values show little error.
#Therefore, the unpruned model suggests overfitting.
#Now lets try to prune using tree sizes of 5 and 6 and see the R2.

#Pruning the tree using a tree size of 6
tree_prune_6 <- prune.tree(tree_model, best = 6)
summary(tree_prune_6)
```

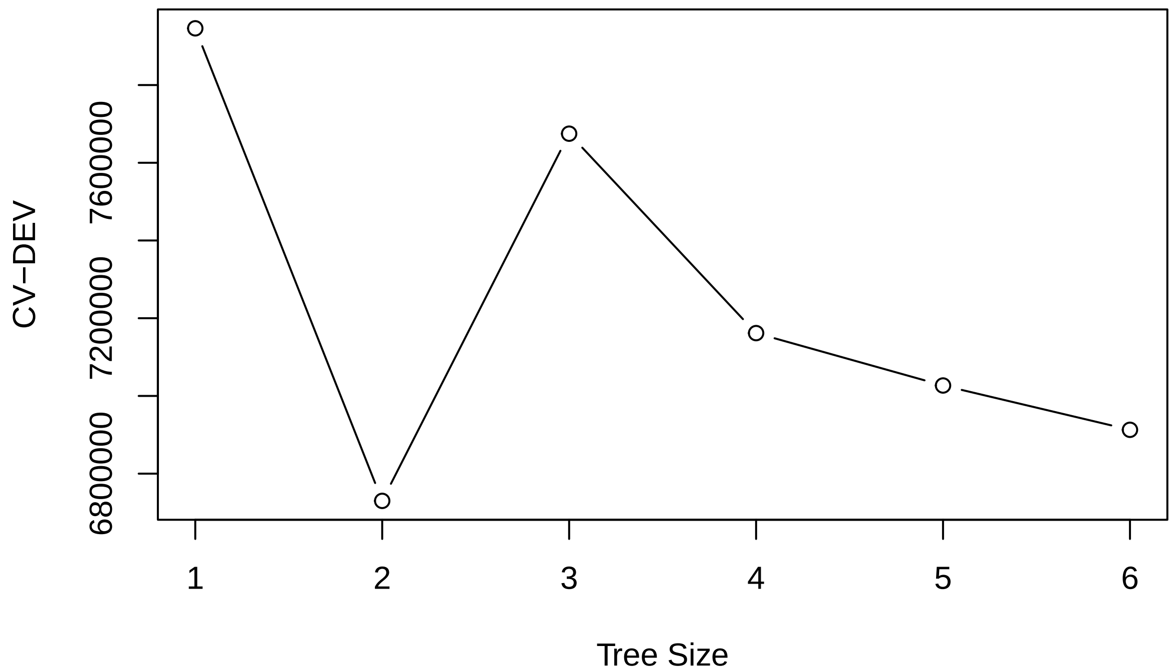
```
##
```

```
## Regression tree:
## snip.tree(tree = tree_model, nodes = 4L)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "NW"
## Number of terminal nodes:  6
## Residual mean deviance:  49100 = 2013000 / 41
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -99.520   -1.545    0.000  122.800  490.100
```

```r
#Prediction of tree
pred_tree_6 <- predict(tree_prune_6, crime_data[,1:15])

#Calculating R2 for the unpruned model
SSE_cv <- sum((pred_tree_6 - crime_data[16])^2)
R2_6 <- 1 - SSE_cv / SST

#Cross-validation on tree
set.seed(18)
tree_cv_6 = cv.tree(tree_prune_6)
summary(tree_cv_6)
```

```
##        Length Class  Mode
## size   6      -none- numeric
## dev    6      -none- numeric
## k      6      -none- numeric
## method 1      -none- character
```

```r
plot(tree_cv_6$size, tree_cv_6$dev, type = 'b',
     xlab = "Tree Size", ylab = "CV-DEV")
```




```r
#For each tree will reflect its deviation

tree_cv_6$size
```

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

```r
tree_cv_6$dev
```

```
## [1] 6912822 7026839 7161659 7674863 6729955 7945960
```

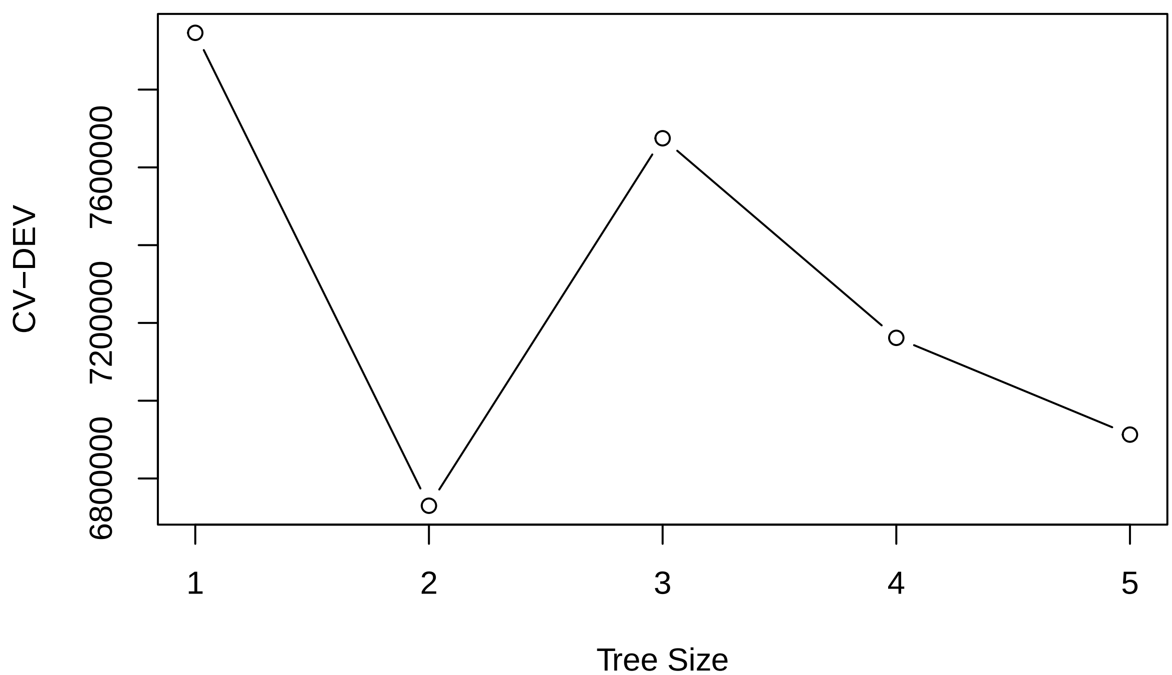
```r
#Now lets try for a tree size of 5
tree_prune_5 <- prune.tree(tree_model, best = 5)
summary(tree_prune_5)
```

```
##
## Regression tree:
## snip.tree(tree = tree_model, nodes = c(4L, 6L))
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "NW"
## Number of terminal nodes:  5
## Residual mean deviance:  54210 = 2277000 / 42
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -573.9  -107.5    15.5     0.0   122.8   490.1
```

```r
#Prediction on tree size 5
pred_tree_5 <- predict(tree_prune_5, crime_data[,1:15])

#Calculating R2 for the unpruned model
SSE_cv <- sum((pred_tree_5 - crime_data[16])^2)
R2_5 <- 1 - SSE_cv / SST

#Cross-validation on tree
set.seed(18)
tree_cv_5 = cv.tree(tree_prune_5)
summary(tree_cv_5)
```

```
##        Length Class  Mode
## size   5      -none- numeric
## dev    5      -none- numeric
## k      5      -none- numeric
## method 1      -none- character
```

```r
plot(tree_cv_5$size, tree_cv_5$dev, type = "b",
     xlab = "Tree Size", ylab = "CV-DEV")
```




```r
#For each tree will reflect its deviation

tree_cv_5$size
```

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

```r
tree_cv_5$dev
```

```
## [1] 6912822 7161659 7674863 6729955 7945960
```

I used a decision tree to evaluate the small data set. The decision tree was fitted to the full data set called un-pruned data set. The R^2 for this model is 72%, close to what I found with a linear regression model in the previous assignment. I ran cross-validation on the un-pruned data set to spot overfitting. The CV process revealed overfitting which was expected in a small data set. Therefore, prunning was performed to create an optimal decision tree and the same process followed but, this time the R^2 did not decrease with a smaller tree size and overfitting was still pronounced.

Part b

```r
#Random Forest model

rf <- randomForest(Crime~., data = crime_data)
summary(rf)
```

```
##                 Length Class  Mode
## call              3    -none- call
## type              1    -none- character
## predicted        47    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times        47    -none- numeric
## importance       15    -none- numeric
## importanceSD      0    -none- NULL
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y                47    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```r
print(rf)
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = crime_data)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 84806.7
##                     % Var explained: 42.07
```

```r
#Prediction on RF model
pred_rf <- predict(rf)

#Calculating R2 for the RF model
SSE_rf <- sum((pred_rf - crime_data[16])^2)
SST <- sum((crime_data$Crime - mean(crime_data$Crime))^2)
R2_rf <- 1 - SSE_rf/SST
R2_rf
```

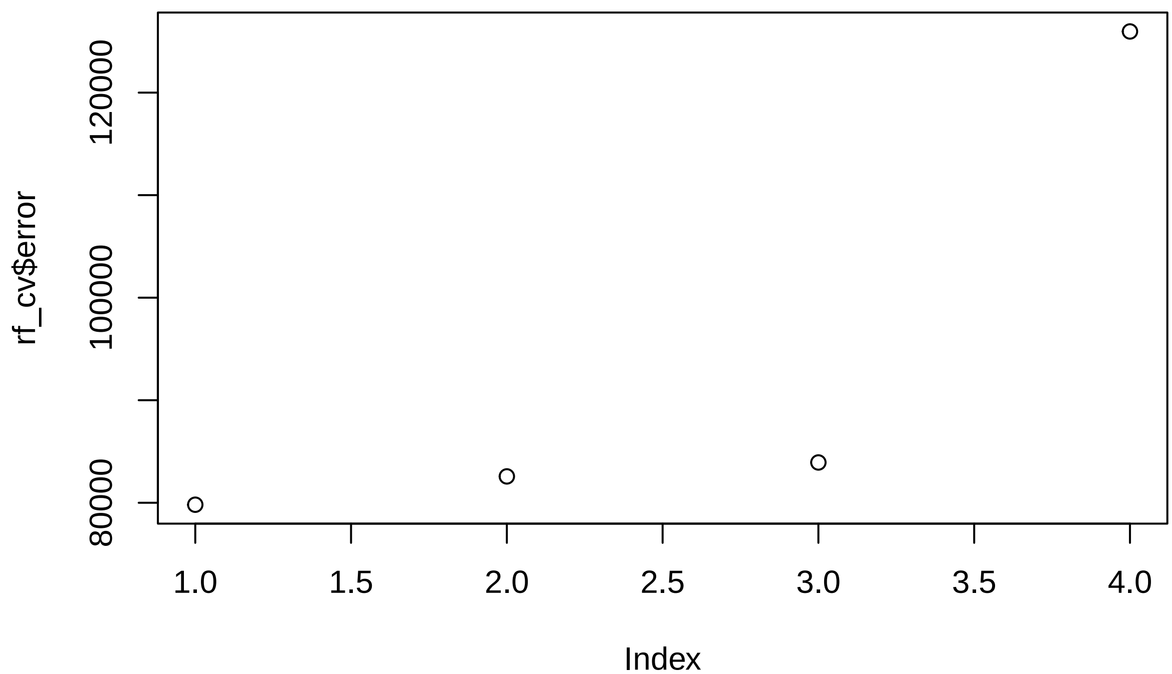
```
## [1] 0.42073
```

```r
#CV on RF model and the R2
rf_cv <- rfcv(trainx = crime_data[,1:15], trainy = crime_data$Crime, cv.fold = 10)
print(rf_cv)
```

```
## $n.var
## [1] 15  8  4  1
##
## $error.cv
##        15         8         4         1
##  79809.64  82569.71  83928.17 125964.94
##
## $predicted
## $predicted$`15`
##  [1]  761.7234 1104.4435  687.6527 1361.4968  974.4451 1178.1952  933.9357
##  [8] 1032.6052  842.8781  764.2576 1225.1793  842.1519  708.5982  720.1272
## [15]  700.5806  893.1612  652.6221 1013.3265 1101.2674 1170.0086  838.4113
## [22]  717.3869  935.9250  907.6164  681.8581 1264.6006  852.6843  977.2391
## [29] 1246.1808  809.2042  750.3706 1096.1583  747.8535  937.9947 1097.5014
## [36] 1060.1829  825.1743  648.5620  787.4838  992.1146  786.3486  587.9489
## [43]  892.0229 1045.9562  656.7548 1005.5758 1078.4402
##
## $predicted$`8`
##  [1]  749.2641 1053.0746  678.1312 1372.5514  969.3615 1275.0023  938.9363
##  [8] 1077.6097  816.8258  723.1820 1253.6596  828.7718  661.9169  678.3712
## [15]  667.1509  954.2401  596.8021 1065.1728 1206.6320 1247.0461  862.6599
## [22]  693.2627  926.1622  960.8124  676.5038 1318.9806  838.9362  919.5391
## [29] 1209.8345  774.7754  710.7930 1139.2371  772.6364  860.5083 1124.6261
## [36] 1015.2740  793.2516  618.6377  747.5356  941.5504  735.5742  562.5441
## [43]  910.0046 1049.2343  636.5604 1033.2314 1015.6866
##
## $predicted$`4`
##  [1]  716.8167 1057.7331  602.3791 1436.5503 1002.1193 1228.1315 1084.2895
##  [8] 1198.2753  781.3632  707.2577 1122.6516  817.4284  625.4396  680.3395
## [15]  643.7114  951.9747  629.2351 1254.1266 1308.6034 1326.2521  773.8185
## [22]  685.5508  965.9302  864.2897  673.2007 1327.2754  817.8120  920.7302
## [29] 1480.1317  729.3598  672.0886 1068.0030  774.8349  928.7939 1189.8076
## [36] 1193.2473  775.9754  516.6584  742.3951  995.0158  712.3654  493.3957
## [43]  927.2906  958.9664  560.8161 1023.3296  930.1657
##
## $predicted$`1`
##  [1]  717.4464  667.3418  490.6725 1083.4521 1137.9607 1186.2215 1197.0720
##  [8] 1142.6937  809.9206  841.0243  923.0994  812.1443  599.1325  682.0892
```

```
## [15] 1091.6210 1068.2805  892.4708 1351.0906 1420.9565 1315.9822  878.0493
## [22]  843.0556  846.3730  838.0067  722.4237 1635.8663  615.0410  928.2220
## [29] 1635.8663  792.2858  699.2301 1049.7360  722.4237  599.0559  958.7147
## [36] 1104.9348  843.0556  971.4194  728.6905  928.2220  793.6019  559.1215
## [43]  839.6947 1305.0263  517.7340 1104.9348  954.6451
```

```r
plot(rf_cv$error)
```




```r
#Prediction using RF_CV model
pred_rf_cv <- rf_cv$predicted[1]

#Calculating R2 for RF_CV model
SSE_rf_cv1 <- sum((pred_rf_cv - crime_data[16])^2)
SST <- sum((crime_data$Crime - mean(crime_data$Crime))^2)
R2_rf_cv <- 1 - SSE_rf_cv1/SST
R2_rf_cv
```

```
## [1] 0.4548623
```

In my findings, the random forest model deals with overfitting sufficiently better than the decision tree in Part a. This can be seen by the errors from the CV model which are close to the initial RF model. Also, the R2 value for the CV model is close to the initial model. Thus, it can be seen that the RF model is good in handling overfitting.

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.

Researchers can use predictors that determine the probability of a student getting accepted into a particular university. The relationship between the predictors and the probability of getting accepted is when a logistic regression model is appropriate. The predictors include:

1. GPA
2. ACT score
3. SAT score
4. Number of AP classes
5. Extra curricular activities/ community service hours

The different predictors and the response is either 1 (will be accepted) and 0 (will not be accepted). So, any student can go through the model and its output will either be 1 or 0.

Question 10.3

Part I *Using the GermanCredit data set germancredit.txt 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.*

```r
library(reshape2)
library(dummy)
```

```
## dummy 0.1.3
```

```
## dummyNews()
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```r
library(e1071)
library(pROC)
```

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

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

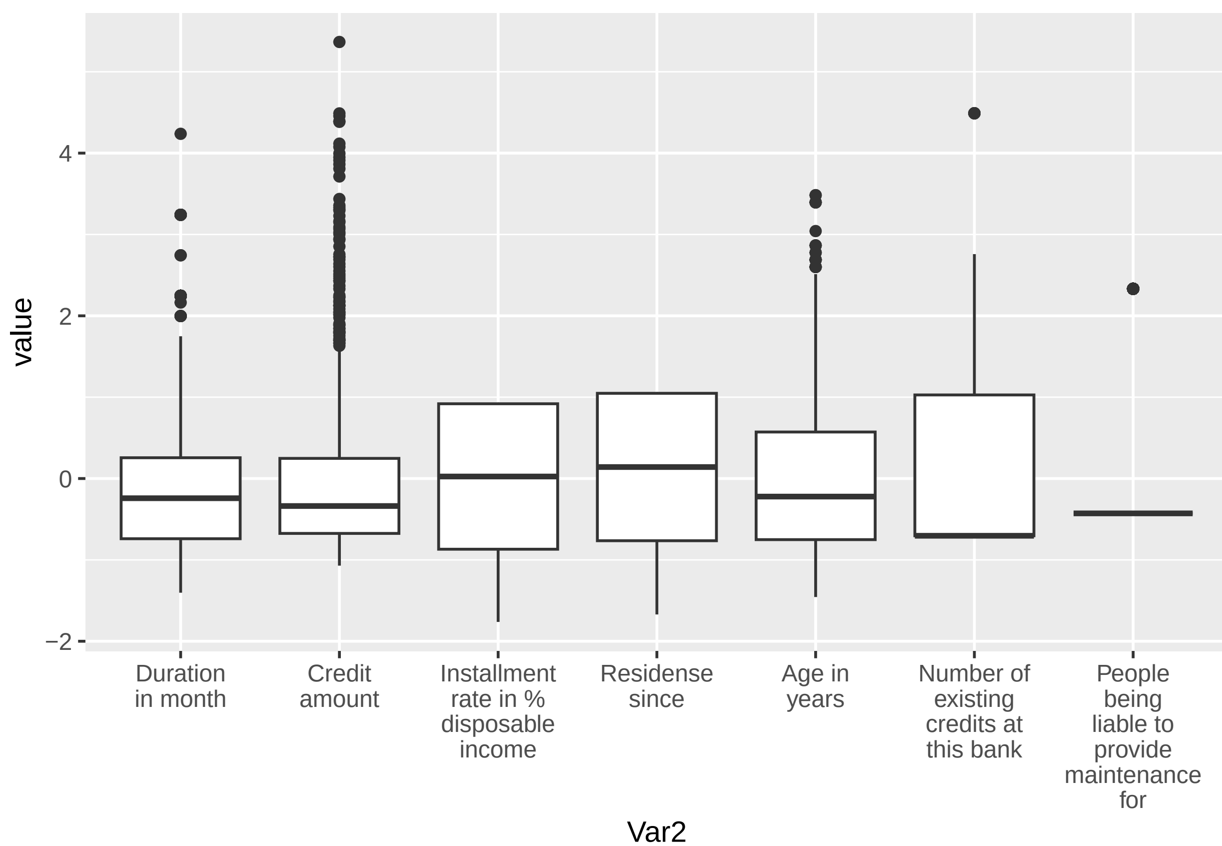
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
german_credit <- read.table(file= "C:\\Users\\sheya\\OneDrive\\Desktop\\germancredit.txt",
                        header = TRUE)
summary(german_credit)
```

```
##      A11                  X6            A34                A43
##  Length:999         Min.   : 4.00   Length:999         Length:999
##  Class :character   1st Qu.:12.00   Class :character   Class :character
##  Mode  :character   Median :18.00   Mode  :character   Mode  :character
##                     Mean   :20.92
##                     3rd Qu.:24.00
##                     Max.   :72.00
##      X1169          A65                A75                  X4
##  Min.   :  250   Length:999         Length:999         Min.   :1.000
##  1st Qu.: 1368   Class :character   Class :character   1st Qu.:2.000
##  Median : 2320   Mode  :character   Mode  :character   Median :3.000
##  Mean   : 3273                                         Mean   :2.972
##  3rd Qu.: 3972                                         3rd Qu.:4.000
##  Max.   :18424                                         Max.   :4.000
##      A93                A101                 X4.1          A121
##  Length:999         Length:999         Min.   :1.000   Length:999
##  Class :character   Class :character   1st Qu.:2.000   Class :character
##  Mode  :character   Mode  :character   Median :3.000   Mode  :character
##                                        Mean   :2.844
##                                        3rd Qu.:4.000
##                                        Max.   :4.000
##       X67           A143               A152                 X2
##  Min.   :19.00   Length:999         Length:999         Min.   :1.000
##  1st Qu.:27.00   Class :character   Class :character   1st Qu.:1.000
##  Median :33.00   Mode  :character   Mode  :character   Median :1.000
##  Mean   :35.51                                         Mean   :1.406
##  3rd Qu.:42.00                                         3rd Qu.:2.000
##  Max.   :75.00                                         Max.   :4.000
##      A173                 X1            A192               A201
##  Length:999         Min.   :1.000   Length:999         Length:999
##  Class :character   1st Qu.:1.000   Class :character   Class :character
##  Mode  :character   Median :1.000   Mode  :character   Mode  :character
##                     Mean   :1.155
##                     3rd Qu.:1.000
##                     Max.   :2.000
##       X1.1
##  Min.   :1.0
##  1st Qu.:1.0
##  Median :1.0
##  Mean   :1.3
##  3rd Qu.:2.0
##  Max.   :2.0
```

```r
#Renaming the columns as per the description in the document
newnames <- c('Checking account Status',
```

```
                 'Duration in month', 'Credit history',
                 'Purpose', 'Credit amount',
                 'Savings account/bonds','Employment since',
                 'Installment rate in % disposable income',
                 'Status& Sex', 'Other debtors / guarantors',
                 'Residense since', 'Property', 'Age in years',
                 'Other installment plans', 'Housing',
                 'Number of existing credits at this bank', 'Job',
                 'People being liable to provide maintenance for',
                 'Telephone', 'foreign worker', 'Customer_Status')
colnames(german_credit) <- newnames

#The document shows there are 7 numeric and 13 categorical predictors
#I will group the numerical variables to visualize the data

num_val <- scale(german_credit[,c(2,5,8,11,13,16,18)])
num_val <- melt(num_val)

#Box plot to view the outliers
ggplot(num_val, aes(x = Var2, y = value)) + geom_boxplot() +
  scale_x_discrete(labels = function(x) stringr::str_wrap(x, width = 10))
```




```
#According to the boxplot, "Credit amount" reveal a significant number of outliers
#However, I cannot remove them for this analysis.
```

```r
#The next step so to create dummy variables for the categorical variables
categorical <- german_credit[,-c(2,5,8,11,13,16,18,21)]
numerical <- german_credit[, c(2,5,8,11,13,16,18,21)]

cat_dummy <- dummy(categorical)

#Now I must drop one variable from each category to avoid multi-collinearity.

cat_dummy_new <- cat_dummy[,-c(1,5,10,20,25,30,34,37,41,44,47,51,53)]

#Combining the numerical and dummy data together
new_data <- cbind(cat_dummy_new, numerical)
new_data$Customer_Status <- ifelse(new_data$Customer_Status == 1, 1, 0)

#Response to 1 and 0, 1 = Good and 0 = Bad

table(new_data$Customer_Status)
```

```
##
##   0   1
## 300 699
```

```r
set.seed(42)
#Create Data Partition
germ_credit_split <- createDataPartition(y = new_data$Customer_Status,
                                         p = 0.7, times = 1, list = FALSE)
head(germ_credit_split,6)
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         7
## [6,]         8
```

```r
#Training data
train_data <- new_data[germ_credit_split,]

#Testing data
test_data <- new_data[-germ_credit_split,]

#Initial model with all variables
initial_model <- glm(train_data$Customer_Status~.,
                     family = binomial(link = "logit"),
                     data = train_data)

summary(initial_model)
```

```
##
## Call:
## glm(formula = train_data$Customer_Status ~ ., family = binomial(link = "logit"),
```

```
##     data = train_data)
##
## Coefficients:
##                                                  Estimate Std. Error z value
## (Intercept)                                     2.423e-01  1.391e+00   0.174
## Checking.account.Status_A121                    2.869e-01  2.634e-01   1.089
## Checking.account.Status_A131                    1.129e+00  5.021e-01   2.249
## Checking.account.Status_A141                    1.616e+00  2.840e-01   5.691
## Credit.history_A311                            -1.171e-01  7.137e-01  -0.164
## Credit.history_A321                             6.137e-01  5.423e-01   1.132
## Credit.history_A331                             8.711e-01  5.943e-01   1.466
## Credit.history_A341                             1.231e+00  5.508e-01   2.236
## Purpose_A411                                    1.494e+00  4.526e-01   3.301
## Purpose_A4101                                   1.073e+00  8.634e-01   1.243
## Purpose_A421                                    8.633e-01  3.235e-01   2.669
## Purpose_A431                                    8.176e-01  3.017e-01   2.710
## Purpose_A441                                    1.101e+00  9.799e-01   1.124
## Purpose_A451                                    1.636e-01  6.599e-01   0.248
## Purpose_A461                                   -6.330e-01  4.879e-01  -1.297
## Purpose_A481                                    1.656e+00  1.197e+00   1.383
## Purpose_A491                                    3.009e-01  3.954e-01   0.761
## Savings.account.bonds_A621                      7.327e-01  3.556e-01   2.060
## Savings.account.bonds_A631                      2.318e-01  4.498e-01   0.515
## Savings.account.bonds_A641                      1.191e+00  6.289e-01   1.894
## Savings.account.bonds_A651                      9.366e-01  3.368e-01   2.781
## Employment.since_A721                          -3.904e-01  5.289e-01  -0.738
## Employment.since_A731                           1.650e-01  5.054e-01   0.326
## Employment.since_A741                           6.894e-01  5.439e-01   1.267
## Employment.since_A751                           3.031e-01  5.096e-01   0.595
## Status..Sex_A921                                3.689e-01  4.449e-01   0.829
## Status..Sex_A931                                7.688e-01  4.357e-01   1.765
## Status..Sex_A941                                6.067e-01  5.484e-01   1.106
## Other.debtors...guarantors_A1021               -1.142e+00  4.906e-01  -2.329
## Other.debtors...guarantors_A1031                7.616e-01  5.020e-01   1.517
## Property_A1221                                 -3.140e-01  3.087e-01  -1.017
## Property_A1231                                 -1.904e-01  2.815e-01  -0.676
## Property_A1241                                 -1.740e-01  5.246e-01  -0.332
## Other.installment.plans_A1421                   1.144e-01  4.937e-01   0.232
## Other.installment.plans_A1431                   6.186e-01  3.000e-01   2.062
## Housing_A1521                                   4.877e-01  2.786e-01   1.750
## Housing_A1531                                  -1.525e-02  5.937e-01  -0.026
## Job_A1721                                      -6.614e-01  9.171e-01  -0.721
## Job_A1731                                      -1.011e+00  8.887e-01  -1.137
## Job_A1741                                      -8.998e-01  8.998e-01  -1.000
## Telephone_A1921                                 2.877e-01  2.525e-01   1.139
## foreign.worker_A2021                            1.750e+00  7.470e-01   2.343
## `Duration in month`                            -3.402e-02  1.132e-02  -3.007
## `Credit amount`                                -1.288e-04  5.519e-05  -2.333
## `Installment rate in % disposable income`      -3.578e-01  1.098e-01  -3.260
## `Residense since`                              -3.172e-02  1.035e-01  -0.306
## `Age in years`                                  1.906e-02  1.133e-02   1.683
## `Number of existing credits at this bank`      -1.935e-01  2.233e-01  -0.866
## `People being liable to provide maintenance for` -2.890e-01  3.054e-01  -0.946
##                                                Pr(>|z|)
```

```
## (Intercept)                                      0.861669
## Checking.account.Status_A121                     0.276038
## Checking.account.Status_A131                     0.024518 *
## Checking.account.Status_A141                     1.27e-08 ***
## Credit.history_A311                              0.869614
## Credit.history_A321                              0.257807
## Credit.history_A331                              0.142684
## Credit.history_A341                              0.025378 *
## Purpose_A411                                     0.000962 ***
## Purpose_A4101                                    0.213958
## Purpose_A421                                     0.007616 **
## Purpose_A431                                     0.006722 **
## Purpose_A441                                     0.260977
## Purpose_A451                                     0.804224
## Purpose_A461                                     0.194522
## Purpose_A481                                     0.166560
## Purpose_A491                                     0.446675
## Savings.account.bonds_A621                       0.039367 *
## Savings.account.bonds_A631                       0.606357
## Savings.account.bonds_A641                       0.058279 .
## Savings.account.bonds_A651                       0.005427 **
## Employment.since_A721                            0.460405
## Employment.since_A731                            0.744072
## Employment.since_A741                            0.205019
## Employment.since_A751                            0.551987
## Status..Sex_A921                                 0.407027
## Status..Sex_A931                                 0.077640 .
## Status..Sex_A941                                 0.268563
## Other.debtors...guarantors_A1021                 0.019880 *
## Other.debtors...guarantors_A1031                 0.129192
## Property_A1221                                   0.309166
## Property_A1231                                   0.498846
## Property_A1241                                   0.740054
## Other.installment.plans_A1421                    0.816690
## Other.installment.plans_A1431                    0.039191 *
## Housing_A1521                                    0.080074 .
## Housing_A1531                                    0.979506
## Job_A1721                                        0.470781
## Job_A1731                                        0.255447
## Job_A1741                                        0.317323
## Telephone_A1921                                  0.254655
## foreign.worker_A2021                             0.019133 *
## `Duration in month`                              0.002641 **
## `Credit amount`                                  0.019649 *
## `Installment rate in % disposable income`        0.001116 **
## `Residense since`                                0.759277
## `Age in years`                                   0.092459 .
## `Number of existing credits at this bank`        0.386261
## `People being liable to provide maintenance for` 0.344055
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 868.33  on 699  degrees of freedom
## Residual deviance: 621.17  on 651  degrees of freedom
## AIC: 719.17
##
## Number of Fisher Scoring iterations: 5
```

```r
#From the summary, there are plenty non-significant p-values.
#I will build a new model with only significant variables and a threshold of 0.05.

new_model <- glm(train_data$Customer_Status ~ Checking.account.Status_A13 +
                   Checking.account.Status_A14 + Credit.history_A34 +
                   Purpose_A41 + Purpose_A42 + Purpose_A43 +
                   Savings.account.bonds_A62 + Savings.account.bonds_A65 +
                   Other.debtors...guarantors_A102 + foreign.worker_A202 +
                   `Duration in month`+ `Credit amount` +
                   `Installment rate in % disposable income`,
                 family = binomial(link = "logit"), data = train_data)

summary(new_model)
```

```
##
## Call:
## glm(formula = train_data$Customer_Status ~ Checking.account.Status_A13 +
##     Checking.account.Status_A14 + Credit.history_A34 + Purpose_A41 +
##     Purpose_A42 + Purpose_A43 + Savings.account.bonds_A62 + Savings.account.bonds_A65 +
##     Other.debtors...guarantors_A102 + foreign.worker_A202 + `Duration in month` +
##     `Credit amount` + `Installment rate in % disposable income`,
##     family = binomial(link = "logit"), data = train_data)
##
## Coefficients:
##                                            Estimate Std. Error z value
## (Intercept)                               1.518e+00  3.851e-01   3.941
## Checking.account.Status_A131              8.649e-01  4.424e-01   1.955
## Checking.account.Status_A141              1.504e+00  2.242e-01   6.710
## Credit.history_A341                       7.287e-01  2.273e-01   3.206
## Purpose_A411                              1.140e+00  3.855e-01   2.958
## Purpose_A421                              4.411e-01  2.573e-01   1.715
## Purpose_A431                              7.852e-01  2.386e-01   3.291
## Savings.account.bonds_A621                6.421e-01  3.269e-01   1.964
## Savings.account.bonds_A651                9.959e-01  3.024e-01   3.293
## Other.debtors...guarantors_A1021         -1.193e+00  4.670e-01  -2.554
## foreign.worker_A2021                      1.375e+00  7.131e-01   1.928
## `Duration in month`                      -4.032e-02  1.033e-02  -3.903
## `Credit amount`                          -7.889e-05  4.802e-05  -1.643
## `Installment rate in % disposable income` -2.832e-01  9.758e-02  -2.902
##                                           Pr(>|z|)
## (Intercept)                               8.10e-05 ***
## Checking.account.Status_A131              0.050544 .
## Checking.account.Status_A141              1.94e-11 ***
## Credit.history_A341                       0.001345 **
## Purpose_A411                              0.003097 **
## Purpose_A421                              0.086384 .
## Purpose_A431                              0.000998 ***
## Savings.account.bonds_A621                0.049514 *
```

```
## Savings.account.bonds_A651                0.000990 ***
## Other.debtors...guarantors_A1021          0.010642 *
## foreign.worker_A2021                      0.053836 .
## `Duration in month`                       9.50e-05 ***
## `Credit amount`                           0.100368
## `Installment rate in % disposable income` 0.003706 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 868.33  on 699  degrees of freedom
## Residual deviance: 685.45  on 686  degrees of freedom
## AIC: 713.45
##
## Number of Fisher Scoring iterations: 5
```

```r
#This new model is my final model as most of the variables are significant.
#I will need to change the test data with the same variables as the ones in the new model
#for predictions

test_new<-test_data[,c('Checking.account.Status_A13',
                     'Checking.account.Status_A14','Credit.history_A34',
                     'Purpose_A41','Purpose_A42','Purpose_A43',
                     'Savings.account.bonds_A62','Savings.account.bonds_A65',
                     'Other.debtors...guarantors_A102',
                     'Other.installment.plans_A143',
                     'foreign.worker_A202','Duration in month','Credit amount',
                     'Installment rate in % disposable income','Customer_Status')]

#Accuracy defines how effective the model is in characterizing bad and good status.

#Calculate the predicted probabilities for the test data using 0.5 as threshold
predicted <- predict(new_model, test_new, type = "response")
predicted_int0.5 <- as.integer(predicted > 0.5)

table <- table(predicted_int0.5, test_new$Customer_Status)
confusion <- confusionMatrix(table, positive = '1')
confusion
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.5   0   1
##                0  34  26
##                1  48 191
##
##                Accuracy : 0.7525
##                  95% CI : (0.6996, 0.8004)
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.16558
##
##                   Kappa : 0.3217
##
```

```
##  Mcnemar's Test P-Value : 0.01464
##
##             Sensitivity : 0.8802
##             Specificity : 0.4146
##          Pos Pred Value : 0.7992
##          Neg Pred Value : 0.5667
##              Prevalence : 0.7258
##          Detection Rate : 0.6388
##    Detection Prevalence : 0.7993
##       Balanced Accuracy : 0.6474
##
##        'Positive' Class : 1
##
```

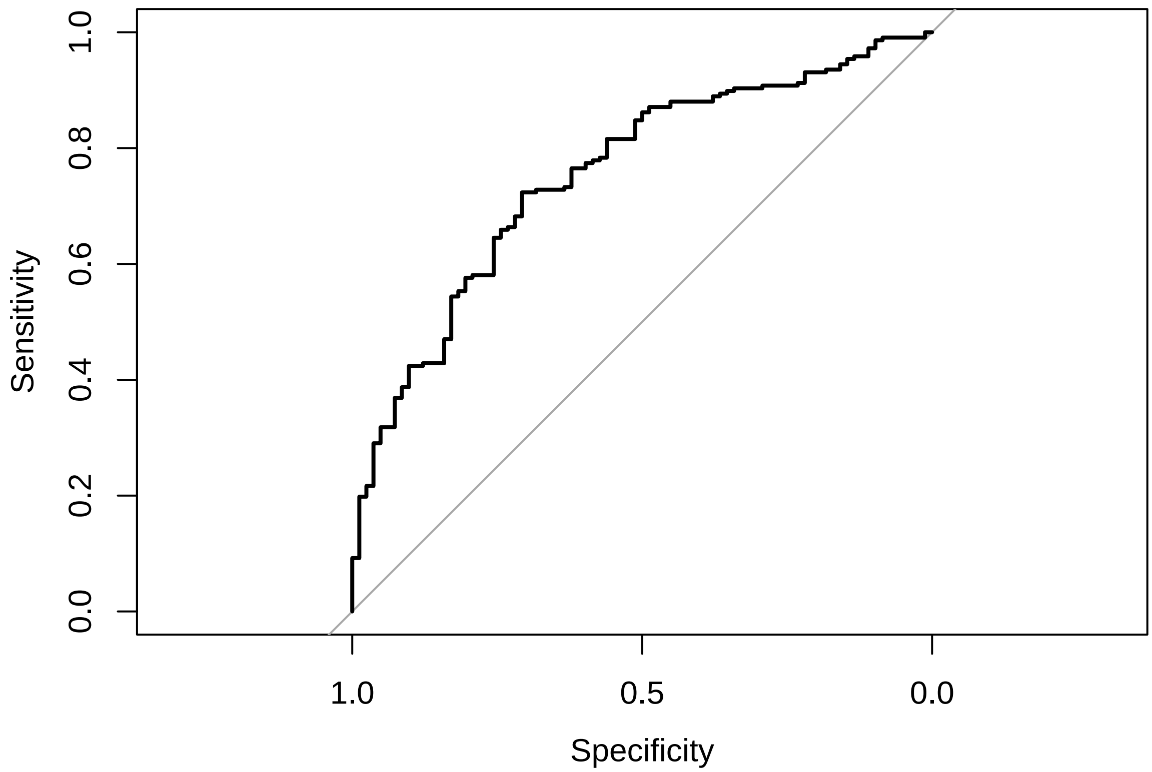
```r
#Accuracy predicted with 0.5 threshold is 75%

#I would like to try 0.6 as a threshold to see the accuracy
predicted_int0.6 <- as.integer(predicted > 0.6)
table2 <- table(predicted_int0.6, test_new$Customer_Status)
confusion2 <- confusionMatrix(table2)
confusion2
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.6   0   1
##                0  46  45
##                1  36 172
##
##                Accuracy : 0.7291
##                  95% CI : (0.6749, 0.7787)
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.4780
##
##                   Kappa : 0.3419
##
##  Mcnemar's Test P-Value : 0.3741
##
##             Sensitivity : 0.5610
##             Specificity : 0.7926
##          Pos Pred Value : 0.5055
##          Neg Pred Value : 0.8269
##              Prevalence : 0.2742
##          Detection Rate : 0.1538
##    Detection Prevalence : 0.3043
##       Balanced Accuracy : 0.6768
##
##        'Positive' Class : 0
##
```

```r
#Accuracy predicted with 0.6 threshold is 71%

#ROC
roc_curve <- roc(test_new$Customer_Status, predicted)
```

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

## Setting direction: controls < cases
```

```r
plot(roc_curve)
```




```r
roc_curve$auc
```

```
## Area under the curve: 0.7538
```

```r
#Therefore, the accuracy is best between 0.5 and 0.6 thresholds since the ROC curve
#reveals the AUC measuring separability.
#Higher the AUC, the better the model is at predicting 0 classes as 0
#and 1 classes as 1.
```

Part II *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.*

```r
#This part asks for the number of loans to false positives are reduced.
#Therefore, the sensitivity needs to be high
#This is accomplished by testing different thresholds and seeing where the
```

```r
#sensitivity is highest

#Previously I tested a threshold of 0.6
predicted_int0.6 <- as.integer(predicted > 0.6)
table2 <- table(predicted_int0.6, test_new$Customer_Status)
    confusion2 <- confusionMatrix(table2, positive = '1')
    confusion2
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.6   0   1
##                0  46  45
##                1  36 172
##
##                Accuracy : 0.7291
##                  95% CI : (0.6749, 0.7787)
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.4780
##
##                   Kappa : 0.3419
##
##  Mcnemar's Test P-Value : 0.3741
##
##             Sensitivity : 0.7926
##             Specificity : 0.5610
##          Pos Pred Value : 0.8269
##          Neg Pred Value : 0.5055
##              Prevalence : 0.7258
##          Detection Rate : 0.5753
##    Detection Prevalence : 0.6957
##       Balanced Accuracy : 0.6768
##
##        'Positive' Class : 1
##
```

```r
#The sensitivity predicted 79% with a threshold of 0.6

#Threshold of 0.4
predicted_int0.4 <- as.integer(predicted > 0.4)
table3 <- table(predicted_int0.4, test_new$Customer_Status)
confusion3 <- confusionMatrix(table3, positive = '1')
confusion3
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.4   0   1
##                0  18  16
##                1  64 201
##
##                Accuracy : 0.7324
##                  95% CI : (0.6784, 0.7818)
```

```
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.4266
##
##                   Kappa : 0.1782
##
##  Mcnemar's Test P-Value : 1.482e-07
##
##             Sensitivity : 0.9263
##             Specificity : 0.2195
##          Pos Pred Value : 0.7585
##          Neg Pred Value : 0.5294
##              Prevalence : 0.7258
##          Detection Rate : 0.6722
##    Detection Prevalence : 0.8863
##       Balanced Accuracy : 0.5729
##
##        'Positive' Class : 1
##
```

```r
#The sensitivity predicted 92% with a threshold of 0.4

#threshold of 0.2
predicted_int0.2 <- as.integer(predicted > 0.2)
table4 <- table(predicted_int0.2, test_new$Customer_Status)
confusion4 <- confusionMatrix(table4, positive = '1')
confusion4
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.2   0   1
##                0   7   2
##                1  75 215
##
##                Accuracy : 0.7425
##                  95% CI : (0.689, 0.7911)
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.2821
##
##                   Kappa : 0.1053
##
##  Mcnemar's Test P-Value : 2.303e-16
##
##             Sensitivity : 0.99078
##             Specificity : 0.08537
##          Pos Pred Value : 0.74138
##          Neg Pred Value : 0.77778
##              Prevalence : 0.72575
##          Detection Rate : 0.71906
##    Detection Prevalence : 0.96990
##       Balanced Accuracy : 0.53807
##
##        'Positive' Class : 1
##
```

```r
#The sensitivity predicted 98% with a threshold of 0.2

#Threshold 0.1
predicted_int0.1 <- as.integer(predicted > 0.1)
table5 <- table(predicted_int0.1, test_new$Customer_Status)
confusion5 <- confusionMatrix(table5, positive = '1')
confusion5
```

```
## Confusion Matrix and Statistics
##
##
## predicted_int0.1   0   1
##                0   2   2
##                1  80 215
##
##                Accuracy : 0.7258
##                  95% CI : (0.6714, 0.7755)
##     No Information Rate : 0.7258
##     P-Value [Acc > NIR] : 0.5297
##
##                   Kappa : 0.0216
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99078
##             Specificity : 0.02439
##          Pos Pred Value : 0.72881
##          Neg Pred Value : 0.50000
##              Prevalence : 0.72575
##          Detection Rate : 0.71906
##    Detection Prevalence : 0.98662
##       Balanced Accuracy : 0.50759
##
##        'Positive' Class : 1
##
```

```r
#The sensitivity predicted 98% with a threshold of 0.1
```

Based on my findings, the threshold increases the sensitivity as the threshold lowers. However, there was no significant change in sensitivity when the threshold was tested below 0.2. In conclusion, the threshold of 0.2 is a good probability based on the model performed.