Homework_7

Claire Kraft

2024-10-08

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).

In the regression tree it seems there are 4 "generations". The children are Po1, Pop, LF, NW. In the model validation section the R^2 result is 72.445%. R^2 is the metric used to measure the fitness of the model. The closer that the R^2 gets to 100 the better fit the model is. 72% is closer to 100% os the model fits nearly perfectly.

In the random forest i chose 5 predictors. The mean of squared residuals is 84022.61 and percent of variances explained is 42.61%. The model doesn't seem that strong as the variance explainability is sub 50%. But the R^2 is 90 which suggests the model fits well. Given the contradiction, poor variances explanation score (42.61) and good R^2 makes me think there could be an overfitting? Perhaps redoing the model may lead to a better result.

```
# Set up
rm(list = ls())

# Helper
#install.packages("caret")
library(tree)
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
library(randomForest)
```

```
## randomForest 4.7-1.2
```

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

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

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

```
#install.packages("devtools") # Look at reference 1 and 2
library(devtools)
```

```
## Loading required package: usethis
```

```
#devtools::install_github('araastat/reprtree') # Look at reference 2
library(reprtree)
```

```
## Loading required package: plotrix
```

```
## Registered S3 method overwritten by 'reprtree':
##   method    from
##   text.tree tree
```

```
#install.packages("dplyr")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#install.packages("magrittr")
library(magrittr)
#install.packages("pROC")
library(pROC)
```

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

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

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

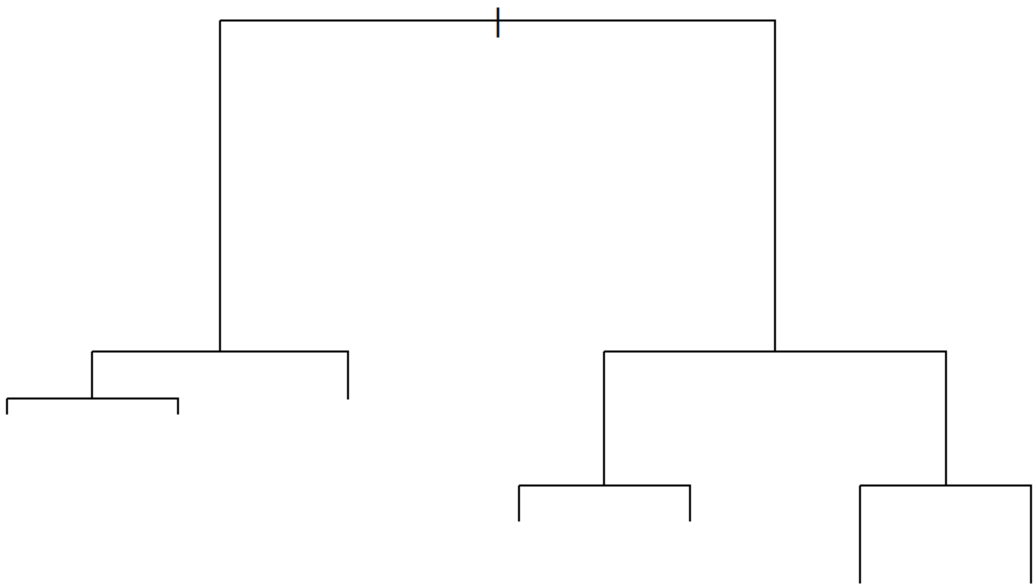
```
# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 07/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
head(uscrime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

```
# Fit a reg tree
crime_tree <- tree(Crime~., data = uscrime)
summary(crime_tree)
```

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

```
# Plot tree
plot(crime_tree)
text(crime_tree, pretty = 0)
```



```
crime_tree
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 47 6881000  905.1
##    2) Po1 < 7.65 23  779200  669.6
##      4) Pop < 22.5 12  243800  550.5
##        8) LF < 0.5675 7   48520  466.9 *
##        9) LF > 0.5675 5   77760  667.6 *
##      5) Pop > 22.5 11  179500  799.5 *
##    3) Po1 > 7.65 24 3604000 1131.0
##      6) NW < 7.65 10  557600  886.9
##       12) Pop < 21.5 5  146400 1049.0 *
##       13) Pop > 21.5 5  147800  724.6 *
##      7) NW > 7.65 14 2027000 1305.0
##       14) Po1 < 9.65 6  170800 1041.0 *
##       15) Po1 > 9.65 8 1125000 1503.0 *
```

```
# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 07/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
head(uscrime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

```
# Fit a reg tree
crime_tree <- tree(Crime~., data = uscrime)

# Make predictions on the training data
predicted_values <- predict(crime_tree, newdata = uscrime)
summary(predicted_values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   466.9   696.1   799.5   905.1  1049.2  1502.9
```

```
# Calculate Mean Squared Error (MSE)
mse <- mean((uscrime$Crime - predicted_values)^2)
mse
```

```
## [1] 40334.5
```

```
# Calculate R-squared
# total sum of squares
sst <- sum((uscrime$Crime - mean(uscrime$Crime))^2)
sst
```

```
## [1] 6880928
```

```
# sum of suqare errors
sse <- sum((uscrime$Crime - predicted_values)^2)
sse
```

```
## [1] 1895722
```

```
r2 <- 1 - (sse / sst) #1- 1895722/40334.5
r2
```

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

```
# Set up
rm(list = ls())

# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 07/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
head(uscrime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

```
# Set seed for reproducibility
set.seed(123)
num_pred <- 5


# Fit a random forest model
crime_forest <- randomForest(Crime~., data = uscrime, mtry = num_pred, importance = TRUE, ntree
= 500)
crime_forest
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = uscrime, mtry = num_pred,      importance = TRUE, n
tree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 84022.61
##                     % Var explained: 42.61
```

```
# Make predictions on the training data
crime_forest_pred <- predict(crime_forest, newdata = uscrime)


# Calculate R-squared
# Calculate Total Sum of Squares (SST)
sst <- sum((uscrime$Crime - mean(uscrime$Crime))^2)
sst
```

```
## [1] 6880928
```

```
# Calculate Sum of Squared Errors (SSE)
sse <- sum((crime_forest_pred - uscrime$Crime)^2)
sse
```

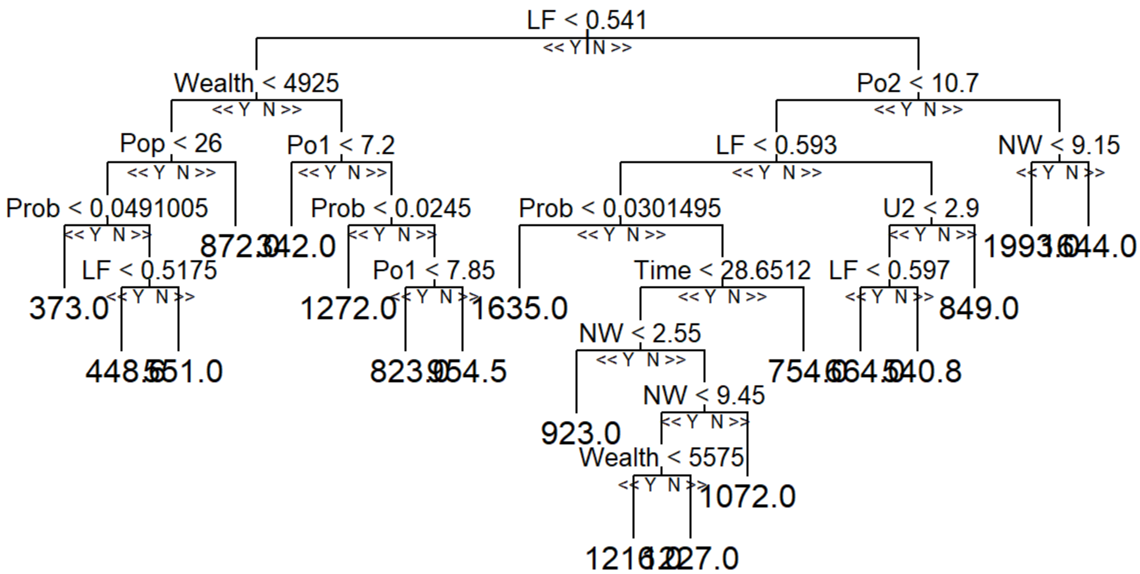
```
## [1] 684962
```

```
# Calculate R-squared
r2 <- 1 - (sse / sst)
r2
```

```
## [1] 0.900455
```

```
# Plot the random forest model
reprtree:::plot.getTree(crime_forest, k = 1, labelVar = TRUE) # Look at reference 1
```

```
## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 * charht, labels =
## stat, : "labelVar" is not a graphical parameter
```



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.

Logistic regression is like linear regression except it is more useful for discrete variables. Discrete variables are like integers where the values are finite. Linear regression is good for continuous variables which are like are like floats. Some possible predictors could be - Work at home (1) or go in the office (0) - Go climbing (1) or stay at home (0) - Travel abroad (1) or stay-cation (0) - Cook at home (1) or take out (0)

Question 10.3.1

Using the GermanCredit data set germancredit.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german (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 (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.

Wow most variables are irrelevant. There is only ne with the twinkle twinkle stars meaning they carry some level of statistical significance. So i will just extract variable V14A143. Looking at the ROC .8 looks to be the appropriate threshold.

```
# Set up
rm(list = ls())

# Read in data
ger_credit <- read.table("~/GitHub/omsa/ISYE 6501/Homework 07/german_credit.txt")

# Convert variables to binary
ger_credit$V21[ger_credit$V21==1]<-0
ger_credit$V21[ger_credit$V21==2]<-1
head(ger_credit)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173   1
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173   1
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172   2
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173   2
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173   2
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172   2
##    V19  V20 V21
## 1 A192 A201   0
## 2 A191 A201   1
## 3 A191 A201   0
## 4 A191 A201   0
## 5 A191 A201   1
## 6 A192 A201   0
```

```
# Split data
ger_credit_train <- ger_credit[1:700,]
ger_credit_test <- ger_credit[701:1000,]
table(ger_credit_train$V21)
```

```
##
##   0   1
## 493 207
```

```
table(ger_credit_test$V21)
```

```
##
##   0   1
## 207  93
```

```
# Create logistical regression model
ger_credit_logreg = glm(V21~., data = ger_credit_train, family = binomial(link = "logit"))
summary(ger_credit_logreg)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = ger_credit_train)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  8.094e-01  1.318e+00   0.614 0.539040
## V1A12       -1.634e-01  2.627e-01  -0.622 0.533900
## V1A13       -1.039e+00  4.368e-01  -2.379 0.017345 *
## V1A14       -1.742e+00  2.908e-01  -5.990 2.09e-09 ***
## V2           2.888e-02  1.109e-02   2.605 0.009177 **
## V3A31        6.540e-01  6.927e-01   0.944 0.345051
## V3A32       -6.821e-01  5.092e-01  -1.339 0.180424
## V3A33       -9.090e-01  5.558e-01  -1.635 0.101968
## V3A34       -1.528e+00  5.305e-01  -2.880 0.003981 **
## V4A41       -1.873e+00  4.937e-01  -3.793 0.000149 ***
## V4A410      -1.556e+00  8.474e-01  -1.836 0.066314 .
## V4A42       -8.275e-01  3.200e-01  -2.586 0.009715 **
## V4A43       -8.720e-01  3.037e-01  -2.872 0.004085 **
## V4A44       -1.269e-01  9.259e-01  -0.137 0.890996
## V4A45       -5.639e-01  6.640e-01  -0.849 0.395753
## V4A46        4.279e-02  4.545e-01   0.094 0.924983
## V4A48       -2.364e+00  1.318e+00  -1.794 0.072847 .
## V4A49       -7.918e-01  4.184e-01  -1.892 0.058447 .
## V5           1.153e-04  5.572e-05   2.069 0.038552 *
## V6A62       -2.399e-01  3.460e-01  -0.693 0.488086
## V6A63       -4.461e-01  5.256e-01  -0.849 0.396045
## V6A64       -1.641e+00  6.364e-01  -2.579 0.009922 **
## V6A65       -8.197e-01  3.163e-01  -2.592 0.009548 **
## V7A72       -2.960e-01  5.418e-01  -0.546 0.584866
## V7A73       -4.289e-01  5.131e-01  -0.836 0.403237
## V7A74       -1.198e+00  5.565e-01  -2.153 0.031289 *
## V7A75       -5.071e-01  5.145e-01  -0.986 0.324340
## V8           3.565e-01  1.079e-01   3.304 0.000954 ***
## V9A92       -4.955e-01  4.594e-01  -1.079 0.280776
## V9A93       -1.299e+00  4.492e-01  -2.891 0.003843 **
## V9A94       -3.874e-01  5.379e-01  -0.720 0.471446
## V10A102      7.803e-01  4.842e-01   1.612 0.107066
## V10A103     -1.099e+00  5.261e-01  -2.090 0.036646 *
## V11          1.389e-02  1.048e-01   0.133 0.894503
## V12A122      4.053e-01  3.135e-01   1.293 0.196068
## V12A123      1.573e-01  2.837e-01   0.555 0.579128
## V12A124      8.694e-01  5.126e-01   1.696 0.089905 .
## V13         -1.376e-02  1.140e-02  -1.208 0.227097
## V14A142      6.347e-02  5.019e-01   0.126 0.899384
## V14A143     -6.359e-01  2.917e-01  -2.180 0.029250 *
## V15A152     -2.121e-01  2.939e-01  -0.722 0.470525
## V15A153     -9.052e-01  5.847e-01  -1.548 0.121597
## V16          3.427e-01  2.253e-01   1.521 0.128271
## V17A172     -7.079e-02  8.640e-01  -0.082 0.934695
## V17A173      1.246e-02  8.326e-01   0.015 0.988064
## V17A174      1.442e-01  8.212e-01   0.176 0.860619
```

```
## V18          4.924e-01  3.096e-01   1.591 0.111703
## V19A192     -3.059e-01  2.500e-01  -1.223 0.221152
## V20A202     -1.418e+00  8.186e-01  -1.732 0.083288 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 850.06  on 699  degrees of freedom
## Residual deviance: 612.64  on 651  degrees of freedom
## AIC: 710.64
##
## Number of Fisher Scoring iterations: 5
```

```
# get the probability for each data point
fitted = predict(ger_credit_logreg, type = 'response')%>%as.data.frame()
fitted
```

```
##                .
## 1   0.0351527293
## 2   0.6631315445
## 3   0.0210935509
## 4   0.1392405844
## 5   0.6971567966
## 6   0.1869923275
## 7   0.0534671612
## 8   0.2072025359
## 9   0.0083122790
## 10  0.8699161579
## 11  0.6849072644
## 12  0.8627703776
## 13  0.1999216130
## 14  0.3495957362
## 15  0.6341273208
## 16  0.5046707557
## 17  0.0297202981
## 18  0.7839498361
## 19  0.5203739223
## 20  0.0668505391
## 21  0.0890886836
## 22  0.1698140083
## 23  0.0770880430
## 24  0.0426810755
## 25  0.0070641448
## 26  0.1233503578
## 27  0.1887037265
## 28  0.1776460429
## 29  0.0674979768
## 30  0.7285553703
## 31  0.1874140780
## 32  0.4885370813
## 33  0.4289784562
## 34  0.0243473942
## 35  0.4594467272
## 36  0.5676275430
## 37  0.1712135466
## 38  0.4191653869
## 39  0.0949980540
## 40  0.2339301437
## 41  0.1627470336
## 42  0.3259181313
## 43  0.2585109099
## 44  0.1146983632
## 45  0.5789759295
## 46  0.2418521158
## 47  0.0884868830
## 48  0.1200843086
## 49  0.0760425725
## 50  0.2283919422
## 51  0.3703116725
```

```
## 52  0.1018104865
## 53  0.0566671803
## 54  0.0087812990
## 55  0.6841374585
## 56  0.0739699142
## 57  0.2264406242
## 58  0.3540586349
## 59  0.4751403538
## 60  0.9168410378
## 61  0.3683594467
## 62  0.0173156007
## 63  0.4891903109
## 64  0.8058079736
## 65  0.2697567201
## 66  0.2717549630
## 67  0.1755145707
## 68  0.2902254599
## 69  0.3755339747
## 70  0.1650827515
## 71  0.1942835290
## 72  0.0136468206
## 73  0.1395260532
## 74  0.3987554847
## 75  0.6077489958
## 76  0.0659230146
## 77  0.6194364075
## 78  0.0917045585
## 79  0.1561335173
## 80  0.6433743166
## 81  0.0594558559
## 82  0.0726541983
## 83  0.1762629670
## 84  0.0877269194
## 85  0.1585195390
## 86  0.0280853711
## 87  0.4960044354
## 88  0.6593344766
## 89  0.4253566070
## 90  0.4127007742
## 91  0.0178086576
## 92  0.0405422693
## 93  0.0950862045
## 94  0.3339439136
## 95  0.0722536432
## 96  0.9404200764
## 97  0.0270755466
## 98  0.3093872844
## 99  0.2201284905
## 100 0.1797368479
## 101 0.3013079772
## 102 0.3595048911
## 103 0.0697264629
```

```
## 104 0.0643904639
## 105 0.0200683684
## 106 0.4304499995
## 107 0.8288421157
## 108 0.2633354743
## 109 0.1114896523
## 110 0.1186279542
## 111 0.5309198927
## 112 0.5728499572
## 113 0.3923373747
## 114 0.6000758163
## 115 0.2800935255
## 116 0.0247823525
## 117 0.4548909821
## 118 0.0892725008
## 119 0.2942851026
## 120 0.2127168533
## 121 0.4845848497
## 122 0.0589823995
## 123 0.0615676372
## 124 0.0864544030
## 125 0.3951678943
## 126 0.4833611141
## 127 0.4220644896
## 128 0.3296644059
## 129 0.1014602445
## 130 0.7166164551
## 131 0.5064655516
## 132 0.8716853777
## 133 0.1668266486
## 134 0.1927535879
## 135 0.2066845025
## 136 0.0241695228
## 137 0.0116418014
## 138 0.1023326291
## 139 0.0312076535
## 140 0.1160716391
## 141 0.0172422944
## 142 0.8340216063
## 143 0.4293459182
## 144 0.2632252290
## 145 0.1250703791
## 146 0.6827085832
## 147 0.3390421522
## 148 0.0469907973
## 149 0.2219159587
## 150 0.0087580344
## 151 0.0467391023
## 152 0.0058620093
## 153 0.3701737286
## 154 0.0862596676
## 155 0.2469909263
```

```
## 156 0.4333941792
## 157 0.0157097480
## 158 0.5300186509
## 159 0.3511699066
## 160 0.0080032796
## 161 0.0282421915
## 162 0.3023816546
## 163 0.0634287642
## 164 0.5594775877
## 165 0.2089570220
## 166 0.0133284912
## 167 0.6481256330
## 168 0.1634675421
## 169 0.0953926673
## 170 0.4567584107
## 171 0.8536793800
## 172 0.1037224652
## 173 0.5510460729
## 174 0.0227593180
## 175 0.6142230536
## 176 0.3622215582
## 177 0.4551536507
## 178 0.0859028972
## 179 0.0867504683
## 180 0.2843912075
## 181 0.4251583476
## 182 0.7798423255
## 183 0.8547330915
## 184 0.0064260299
## 185 0.4748736566
## 186 0.0632447688
## 187 0.5388562993
## 188 0.1263785462
## 189 0.3596572737
## 190 0.6399799564
## 191 0.0259512471
## 192 0.6329974604
## 193 0.4942760974
## 194 0.0429871450
## 195 0.3736529787
## 196 0.3107656312
## 197 0.0230791908
## 198 0.8460417727
## 199 0.1159616069
## 200 0.7367762582
## 201 0.0332478626
## 202 0.6909149064
## 203 0.0762795556
## 204 0.1732500289
## 205 0.1111493926
## 206 0.3718386607
## 207 0.0371364477
```

```
## 208 0.1853640963
## 209 0.7674824684
## 210 0.0004142399
## 211 0.0109800209
## 212 0.0444616489
## 213 0.7908587872
## 214 0.0951918044
## 215 0.0192625169
## 216 0.0070114959
## 217 0.4476015159
## 218 0.1297646191
## 219 0.6108984830
## 220 0.0867123197
## 221 0.3456375951
## 222 0.7354836314
## 223 0.1458715285
## 224 0.0707955270
## 225 0.0239014973
## 226 0.4252081742
## 227 0.4845770768
## 228 0.3777528523
## 229 0.0374829872
## 230 0.6703940818
## 231 0.3243306693
## 232 0.0407607176
## 233 0.0407071598
## 234 0.2353587806
## 235 0.0134793835
## 236 0.7991417065
## 237 0.4627854359
## 238 0.7736131442
## 239 0.1148711019
## 240 0.1500653455
## 241 0.7212857337
## 242 0.0344184126
## 243 0.7013383908
## 244 0.0675110454
## 245 0.1329796945
## 246 0.0364026808
## 247 0.0370322254
## 248 0.2090271068
## 249 0.1025072108
## 250 0.3547275208
## 251 0.0369015237
## 252 0.0968099795
## 253 0.8432329303
## 254 0.0798441591
## 255 0.0311010362
## 256 0.2351946350
## 257 0.0559123758
## 258 0.8779539904
## 259 0.0241341515
```

```
## 260 0.0895997432
## 261 0.1792351695
## 262 0.5029713418
## 263 0.3934316047
## 264 0.2044175455
## 265 0.0215177499
## 266 0.3587706067
## 267 0.0582520326
## 268 0.2862790307
## 269 0.4170618861
## 270 0.0632863630
## 271 0.0164568250
## 272 0.0141921899
## 273 0.9767670153
## 274 0.4432979179
## 275 0.8697956361
## 276 0.0388780340
## 277 0.0706145872
## 278 0.1938395967
## 279 0.1041070721
## 280 0.0940818376
## 281 0.0064830080
## 282 0.0533152240
## 283 0.1733856362
## 284 0.0135766970
## 285 0.6266404239
## 286 0.8066360088
## 287 0.4262261789
## 288 0.3200139098
## 289 0.1657332366
## 290 0.6627904223
## 291 0.0164092123
## 292 0.4384154887
## 293 0.2385092730
## 294 0.0582063069
## 295 0.5663297549
## 296 0.6806568277
## 297 0.0239551694
## 298 0.0361059512
## 299 0.0417588461
## 300 0.0167264166
## 301 0.0751961976
## 302 0.7389305916
## 303 0.1307022905
## 304 0.3117625783
## 305 0.2435459444
## 306 0.0406217471
## 307 0.0517264053
## 308 0.5634941813
## 309 0.3497328686
## 310 0.6783823150
## 311 0.3461344843
```

```
## 312 0.2442869353
## 313 0.3019085102
## 314 0.4809351489
## 315 0.0169125866
## 316 0.8843160514
## 317 0.1137078808
## 318 0.1616558558
## 319 0.0993102957
## 320 0.4081051082
## 321 0.8565178854
## 322 0.7871287929
## 323 0.0495821224
## 324 0.1174036992
## 325 0.1927748686
## 326 0.0203120691
## 327 0.0110650206
## 328 0.0607876633
## 329 0.2993015341
## 330 0.3380039145
## 331 0.0867922663
## 332 0.1878336986
## 333 0.9600446670
## 334 0.2549440406
## 335 0.9372305028
## 336 0.1725769903
## 337 0.1965734956
## 338 0.5814613816
## 339 0.5614024806
## 340 0.5676290157
## 341 0.5928059405
## 342 0.2625610211
## 343 0.2928324506
## 344 0.3906936110
## 345 0.1009230438
## 346 0.0618081681
## 347 0.0784496022
## 348 0.6084189978
## 349 0.0171006189
## 350 0.1350751112
## 351 0.1062710362
## 352 0.0633734954
## 353 0.0097652460
## 354 0.7296740984
## 355 0.0610611826
## 356 0.7228625441
## 357 0.0058761181
## 358 0.1875932346
## 359 0.1419505886
## 360 0.5315244567
## 361 0.2572596634
## 362 0.0415982593
## 363 0.4207808035
```

```
## 364 0.1025307078
## 365 0.5115334072
## 366 0.0139920757
## 367 0.0094234859
## 368 0.4818015693
## 369 0.6120289280
## 370 0.2206213900
## 371 0.1440817186
## 372 0.0450445923
## 373 0.0518006666
## 374 0.2265113849
## 375 0.9615003428
## 376 0.6854348769
## 377 0.1564583430
## 378 0.0142864483
## 379 0.8204265767
## 380 0.0972967833
## 381 0.0840376467
## 382 0.5535440261
## 383 0.1692903827
## 384 0.2838499643
## 385 0.1002178357
## 386 0.0693606628
## 387 0.0848725593
## 388 0.6377369261
## 389 0.1238744379
## 390 0.0795674792
## 391 0.2418020805
## 392 0.0719382556
## 393 0.6001819224
## 394 0.0604783000
## 395 0.0312909770
## 396 0.6489049467
## 397 0.5308459641
## 398 0.4635443014
## 399 0.5587058176
## 400 0.0193988541
## 401 0.0998733908
## 402 0.3779929221
## 403 0.2076255143
## 404 0.1716483507
## 405 0.3785740709
## 406 0.3782835842
## 407 0.0058951696
## 408 0.2297572328
## 409 0.1608691957
## 410 0.1315320538
## 411 0.5165698973
## 412 0.0163033935
## 413 0.1150942022
## 414 0.0631899522
## 415 0.6484509810
```

```
## 416 0.0379585697
## 417 0.6389242633
## 418 0.5421421153
## 419 0.1250650392
## 420 0.5656150349
## 421 0.0355485562
## 422 0.0642738993
## 423 0.1127768745
## 424 0.0413851775
## 425 0.2233954784
## 426 0.1528152433
## 427 0.0787755939
## 428 0.0076001176
## 429 0.0281148321
## 430 0.5577324663
## 431 0.0063569559
## 432 0.6849128392
## 433 0.3137922385
## 434 0.1843836889
## 435 0.2370390865
## 436 0.1245364114
## 437 0.0329058837
## 438 0.0371873444
## 439 0.7063189943
## 440 0.5808851057
## 441 0.1268266412
## 442 0.6071921533
## 443 0.1852986243
## 444 0.3038183596
## 445 0.4169147167
## 446 0.1147617793
## 447 0.8103498649
## 448 0.0484559322
## 449 0.0542749318
## 450 0.2772058408
## 451 0.0281183394
## 452 0.0762490276
## 453 0.1355061173
## 454 0.0352371122
## 455 0.5214647230
## 456 0.1763033382
## 457 0.4046585156
## 458 0.0910331238
## 459 0.6199544619
## 460 0.0741804345
## 461 0.2145234707
## 462 0.3595981853
## 463 0.4994823151
## 464 0.0528310586
## 465 0.0916164647
## 466 0.1689513271
## 467 0.5388152298
```

```
## 468 0.3057924354
## 469 0.1532254099
## 470 0.0142173348
## 471 0.5188883572
## 472 0.7637180782
## 473 0.4727328973
## 474 0.0167523563
## 475 0.3207347897
## 476 0.6770720441
## 477 0.0506652009
## 478 0.2512905680
## 479 0.0695052369
## 480 0.2339496669
## 481 0.1245950043
## 482 0.6654181227
## 483 0.3297717009
## 484 0.0174939540
## 485 0.0270173055
## 486 0.4541547865
## 487 0.0263595426
## 488 0.4470217697
## 489 0.0218017762
## 490 0.0547308427
## 491 0.0486110298
## 492 0.7549257386
## 493 0.0312341705
## 494 0.1240415637
## 495 0.1482321742
## 496 0.3435374934
## 497 0.8919388551
## 498 0.0375272881
## 499 0.2137138472
## 500 0.2479079163
## 501 0.8655181778
## 502 0.2651174226
## 503 0.1142713568
## 504 0.6325511096
## 505 0.8739243430
## 506 0.0326023212
## 507 0.0066753500
## 508 0.8352909894
## 509 0.2741097245
## 510 0.0691590146
## 511 0.3014655256
## 512 0.0344553243
## 513 0.0608680924
## 514 0.7317360060
## 515 0.1218913557
## 516 0.1323172721
## 517 0.0587501683
## 518 0.2474476167
## 519 0.2685124685
```

```
## 520 0.0147835257
## 521 0.0784219749
## 522 0.4081542504
## 523 0.8661943726
## 524 0.0335487808
## 525 0.3204875567
## 526 0.2703254953
## 527 0.1450091924
## 528 0.0095610742
## 529 0.8256627459
## 530 0.3648465940
## 531 0.6097475110
## 532 0.5805369273
## 533 0.0530498396
## 534 0.0945340209
## 535 0.0961415128
## 536 0.3785993887
## 537 0.4525318864
## 538 0.2571235603
## 539 0.9205320430
## 540 0.2915532532
## 541 0.3094534407
## 542 0.1356746947
## 543 0.4334918096
## 544 0.1245968722
## 545 0.0534534089
## 546 0.6110853459
## 547 0.1631020274
## 548 0.1012282234
## 549 0.8567976198
## 550 0.0082889598
## 551 0.0274048051
## 552 0.1933391390
## 553 0.2145135379
## 554 0.2860182128
## 555 0.4609596516
## 556 0.3268306841
## 557 0.8351050905
## 558 0.4005795876
## 559 0.6873216343
## 560 0.1203873264
## 561 0.1778646208
## 562 0.4847779827
## 563 0.2346648931
## 564 0.4151570311
## 565 0.4471507233
## 566 0.3181475261
## 567 0.4624840228
## 568 0.0046831744
## 569 0.3061085190
## 570 0.7345306484
## 571 0.6368239984
```

```
## 572 0.0592355161
## 573 0.0672494708
## 574 0.6076357926
## 575 0.3093626930
## 576 0.1315871035
## 577 0.1732296864
## 578 0.0824369640
## 579 0.8577132343
## 580 0.0871567155
## 581 0.1870939900
## 582 0.3410171659
## 583 0.1625726432
## 584 0.7508651531
## 585 0.1113374547
## 586 0.5286226479
## 587 0.3609234961
## 588 0.1736915761
## 589 0.5683812535
## 590 0.1527282608
## 591 0.1700030925
## 592 0.5551855065
## 593 0.0615556483
## 594 0.7546288686
## 595 0.4330694947
## 596 0.8175721170
## 597 0.7630343188
## 598 0.3824824879
## 599 0.0816536317
## 600 0.0429927973
## 601 0.0705768595
## 602 0.5023419531
## 603 0.9085270102
## 604 0.2911515245
## 605 0.3650697503
## 606 0.7413500385
## 607 0.0206172644
## 608 0.7358520550
## 609 0.0878650349
## 610 0.0334513938
## 611 0.7801409524
## 612 0.1999665555
## 613 0.4166551649
## 614 0.1645432873
## 615 0.0759879390
## 616 0.6896964337
## 617 0.4459402365
## 618 0.3611148549
## 619 0.7034830803
## 620 0.2833782128
## 621 0.1178921981
## 622 0.1897732998
## 623 0.4374313678
```

```
## 624 0.6399102039
## 625 0.3664644319
## 626 0.0348234655
## 627 0.0399286544
## 628 0.5530505781
## 629 0.0595126599
## 630 0.0171540492
## 631 0.4964869633
## 632 0.7793502017
## 633 0.2323922945
## 634 0.3155524190
## 635 0.6388389762
## 636 0.4691926607
## 637 0.0715165195
## 638 0.2689390341
## 639 0.2477371856
## 640 0.6358416207
## 641 0.7918449310
## 642 0.3900754399
## 643 0.1423474522
## 644 0.0194970397
## 645 0.2959028261
## 646 0.1718603197
## 647 0.6203122812
## 648 0.1556280961
## 649 0.4114020602
## 650 0.6403609797
## 651 0.6625037910
## 652 0.3192764759
## 653 0.7069011614
## 654 0.7035867281
## 655 0.0091305387
## 656 0.4940215438
## 657 0.6739374596
## 658 0.1505741962
## 659 0.7546936698
## 660 0.1770696144
## 661 0.2526978553
## 662 0.5586144741
## 663 0.0931815566
## 664 0.5338981005
## 665 0.1252649490
## 666 0.4113673323
## 667 0.5991056048
## 668 0.6236853354
## 669 0.5861157249
## 670 0.1211148488
## 671 0.0496661867
## 672 0.0504975823
## 673 0.4821583187
## 674 0.0431563873
## 675 0.1374787417
```

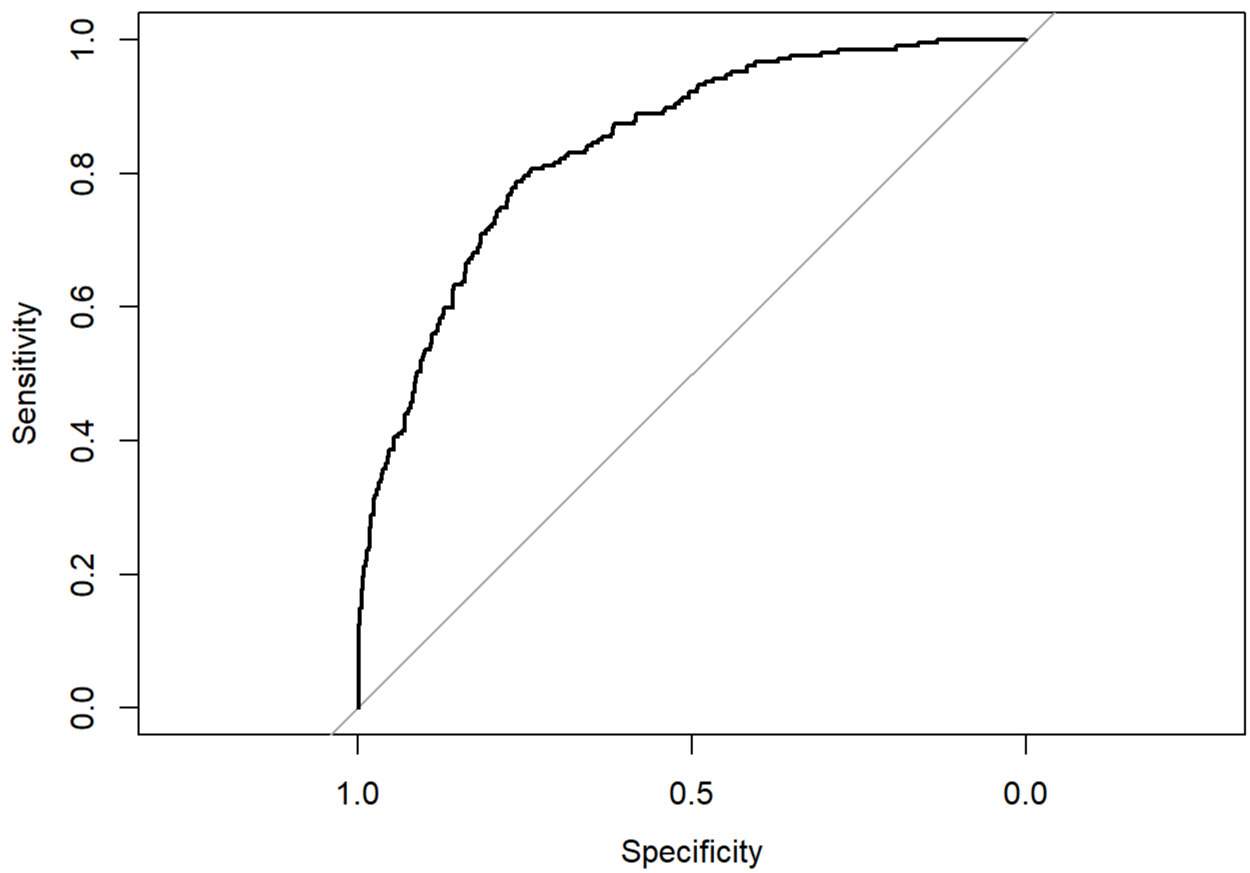
```
## 676 0.0895521636
## 677 0.0507627220
## 678 0.7904106978
## 679 0.4457845777
## 680 0.1340196793
## 681 0.1779301879
## 682 0.0270518497
## 683 0.2819362362
## 684 0.1525426231
## 685 0.3114221327
## 686 0.4253600636
## 687 0.0183651944
## 688 0.5723507651
## 689 0.0641291644
## 690 0.1672671878
## 691 0.4326351612
## 692 0.4113411424
## 693 0.2259759100
## 694 0.1333236418
## 695 0.0427781295
## 696 0.0094894598
## 697 0.0173293421
## 698 0.0954448322
## 699 0.0624075477
## 700 0.2897646341
```

```
# plot the ROC
# Look at reference 4
datas = data.frame(orignial = ger_credit_train$V21, prob=fitted$.)
roc = roc(orignial~prob, data = datas)
```

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

```
## Setting direction: controls < cases
```

```
plot(roc)
```



Question 10.3.2

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.

Looking at the ROC .8 looks to be the appropriate threshold.

References:

[1] alejandro_hagan. (2024). How to visualize random forest model output in R? Stack Overflow. https://stackoverflow.com/questions/73898275/how-to-visualize-random-forest-model-output-in-r (https://stackoverflow.com/questions/73898275/how-to-visualize-random-forest-model-output-in-r)

[2] araastat. (2018, July 17). Error in library("reprtree") : there is no package called "reprtree" · Issue #12 · araastat/reprtree. GitHub. https://github.com/araastat/reprtree/issues/12 (https://github.com/araastat/reprtree/issues/12)

[3] Hofmann, H. (1994). Statlog (German Credit Data) [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5NC77 (https://doi.org/10.24432/C5NC77).

[4] roc function - RDocumentation. (n.d.). Www.rdocumentation.org. https://www.rdocumentation.org/packages/pROC/versions/1.18.5/topics/roc (https://www.rdocumentation.org/packages/pROC/versions/1.18.5/topics/roc)

[5] APA citation