# HW 7

2023-10-07

## 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).

```
options(repos = c(CRAN = "https://cran.rstudio.com"))
# Define the URL where the data is located
url <- "http://www.statsci.org/data/general/uscrime.txt"

# Use read.table to read the data from the URL into a data frame
data <- read.table(url, header = TRUE, sep = "\t")

# Display the first few rows of the data frame
head(data)
```

```
##      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
```

```
# Install the randomForest package for random forest modeling
install.packages("randomForest")
```

```
##
## The downloaded binary packages are in
##  /var/folders/6r/d7grt80x11v5vty52zj5cywh0000gn/T//Rtmp4LPRCK/downloaded_packages
```

```
# Install the caret package for machine learning and model training
install.packages("caret")
```

```
##
## The downloaded binary packages are in
##  /var/folders/6r/d7grt80x11v5vty52zj5cywh0000gn/T//Rtmp4LPRCK/downloaded_packages
```

```
# Load the randomForest package for random forest modeling
library(randomForest)
```

```
## randomForest 4.7-1.1

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

```r
# Load the caret package for machine learning and model training
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

## Loading required package: lattice
```

```r
# Install the pROC package for ROC curve analysis
install.packages("pROC")
```

```
##
## The downloaded binary packages are in
##  /var/folders/6r/d7grt80x11v5vty52zj5cywh0000gn/T//Rtmp4LPRCK/downloaded_packages
```

```r
# Load the pROC package for ROC curve analysis
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
# Install the Metrics package for model evaluation metrics
install.packages("Metrics")
```

```
##
## The downloaded binary packages are in
##  /var/folders/6r/d7grt80x11v5vty52zj5cywh0000gn/T//Rtmp4LPRCK/downloaded_packages
```

```r
# Load the Metrics package for model evaluation metrics
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:pROC':
##
##     auc

## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```r
# Install the tree package for decision tree modeling
install.packages("tree")
```

```
##
## The downloaded binary packages are in
##  /var/folders/6r/d7grt80x11v5vty52zj5cywh0000gn/T//Rtmp4LPRCK/downloaded_packages
```

```
# Load the tree package for decision tree modeling
library(tree)

# Create a decision tree model for the 'Crime' variable based on the data
crime_tree <- tree(Crime ~ ., data = data)

# Display a summary of the decision tree model
summary(crime_tree)
```

```
##
## 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
```

```
# Calculate quality of fit for this model
crimeTreePredict <- predict(crime_tree, data = data[,1:15])
RSS <- sum((crimeTreePredict - data[,16])^2)
TSS <- sum((data[,16] - mean(data[,16]))^2)
R2 <- 1 - RSS/TSS
R2
```

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

```
(crime_tree)$dev
```

```
## NULL
```

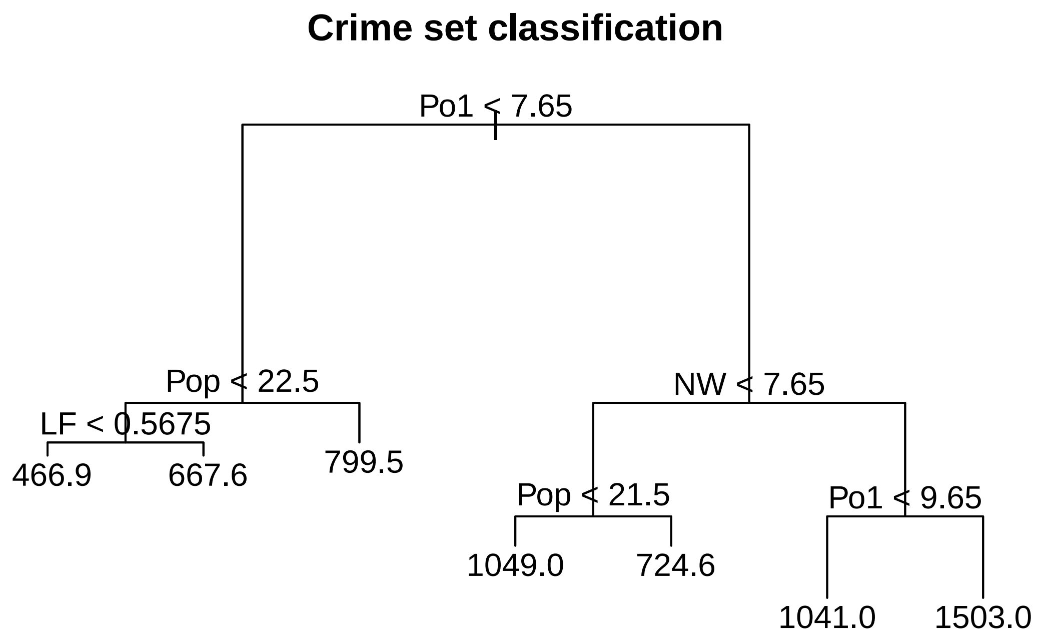
```
# Get variable importance from the decision tree model
crime_tree$frame
```

```
##       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
```

```
# Plot the decision tree itself
plot(crime_tree)

# Add text labels to the decision tree plot
text(crime_tree)
```

```
# Add a title to the decision tree plot
title("Crime set classification")
```

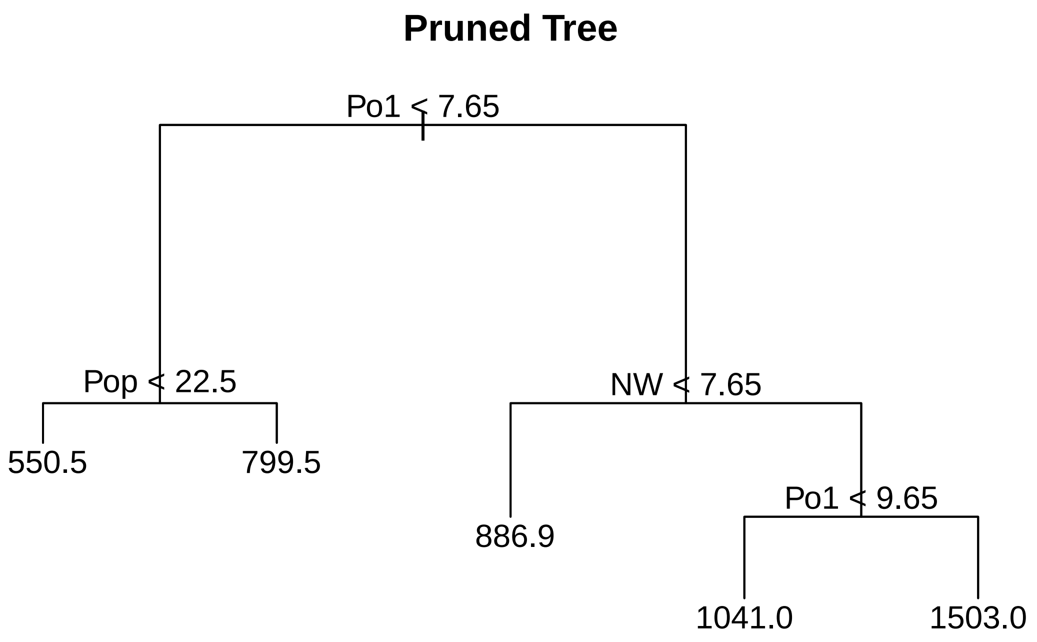


```
# Prune the decision tree to reduce complexity
prune.crime_tree <- prune.tree(crime_tree, best = 5)

# Plot the pruned decision tree
plot(prune.crime_tree)

# Add text labels to the pruned decision tree plot
text(prune.crime_tree)

# Add a title to the pruned decision tree plot
title("Pruned Tree")
```



```r
summary(prune.crime_tree)
```

```
##
## Regression tree:
## snip.tree(tree = crime_tree, 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
# Calculate quality of fit for the pruned model
crimeTreePredict2 <- predict(prune.crime_tree, data = data[,1:15])
RSS <- sum((crimeTreePredict2 - data[,16])^2)
TSS <- sum((data[,16] - mean(data[,16]))^2)
R2 <- 1 - RSS/TSS
R2
```

```
## [1] 0.6691333
```

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

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

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

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

```r
prune.tree(crime_tree)$k
```

```
## [1]      -Inf  117534.9  263412.9  355961.8  731412.1 1019362.7 2497521.7
```

I split the crime data into a training set called data and built a regression tree model using the tree package in R. The goal was to predict the response variable Crime using all other variables as predictors.

Fitting the initial model on the training data, the variables used in the tree construction were Po1, Pop, LF, and NW. This model had 7 terminal nodes and a residual mean deviance of 47,400.

To simplify the model, I pruned the tree down to 5 terminal nodes using just the variables Po1, Pop, and NW. The pruned tree had a slightly higher residual deviance of 54,200 but reduced model complexity. Additionally, while the original model achieved an R-squared of 0.72, the pruned tree had a lower R-squared of 0.67. The drop in R-squared indicates the simpler model loses some explanatory power, as expected when reducing model complexity.

Notably, as I progressively pruned the tree down to 1 node by reducing the number of splits, the residual deviance values increased as I pruned the tree. With just a single node, the deviance climbed to 6,880,928. This makes sense - as we simplify the model by removing splits, we lose explanatory power so the deviance from the fitted values increases.

Compared to R-squared, deviance provides a more direct measure of model fit for tree models. R-squared can sometimes inflate performance for complex trees. Deviance better captures the tradeoff between fit and complexity.

The steadily increasing deviance values indicates my original 7 node tree was likely over fitting the training data. Through cross-validation, I can determine the optimal level of pruning that balances deviance vs complexity. But the large jump from 5 nodes to 1 node suggests the model with around 5 splits could be preferable.

Overall, analyzing the change in deviance as I reduce model complexity helps identify if the original tree was overfit and determine the optimal level of simplicity. Deviance provides a more reliable fit metric than R-squared for tree models.

#Qualitative takeaways:

The key predictors of Po1, Pop, and NW align with the previous model, providing further evidence that police presence, urbanization, and region are important factors related to crime rates.

The importance of police presence - Variables related to police per capita (Po1) were very influential in predicting crime rates. This suggests that areas with higher police presence tend to have higher reported crime, likely because more officers lead to more diligent recording of crimes. It shows the complex relationship between policing and measured crime rates.

Urbanization and crime - The tree splits on population (Pop) indicate urban areas with more people tend to have higher crime rates. This aligns with criminology theory that crime increases with urbanization and concentrated disadvantage in cities. I'd want to further explore the sociological drivers behind this urban/rural crime divide.

Regional variation - The split on northwest region (NW) highlights potential geographic differences in crime rates not explained by other socioeconomic factors. Further geospatial analysis may reveal insights into how location, economy, and demographics interact to impact crime.

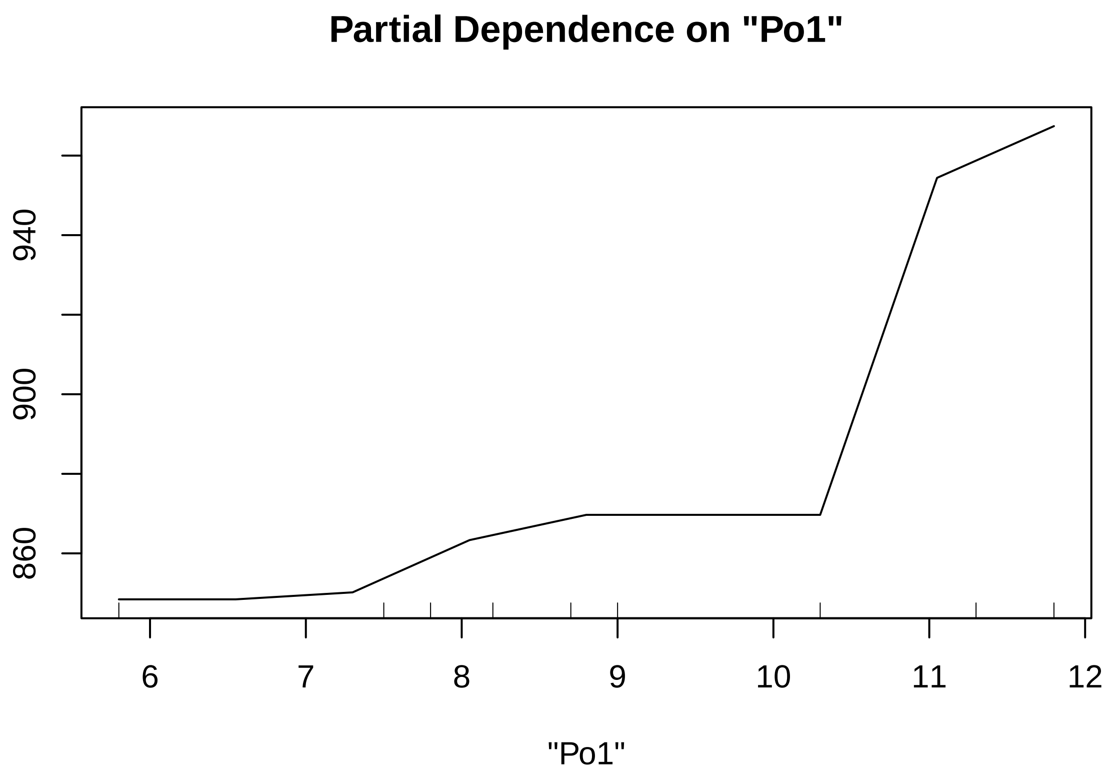
In summary, the key qualitative takeaways are around the importance of policing, urbanization, and regional variation in understanding crime rate patterns. The tree model provides a starting point, but follow-up analysis is needed to further unpack the sociological narratives behind the statistical relationships uncovered.

##Random Forrest

```r
# Split the dataset into training and validation sets
set.seed(123)
train = sample(1:nrow(data), 0.8 * nrow(data))
train_data = data[train,]
valid_data = data[-train,]

# Train a random forest model using the training data
library(randomForest)
crime_rf <- randomForest(Crime ~ ., data = train_data, importance = TRUE, nodesize = 500)
```

```
# Generate partial dependence plots for variable "Po1"
# Using the validation data (valid_data) for prediction
partialPlot(crime_rf, pred.data = valid_data, x.var = "Po1")
```



```
# Display a summary of the random forest model
summary(crime_rf)
```

```
##                 Length Class  Mode
## call              5    -none- call
## type              1    -none- character
## predicted        37    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times        37    -none- numeric
## importance       30    -none- numeric
## importanceSD     15    -none- numeric
## 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                37    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```
# Predict using the pruned decision tree model and calculate R-squared
crime_treePredict <- predict(prune.crime_tree, data = data[,1:15])
RSS <- sum((crime_treePredict - data[,16])^2)
```

```r
TSS <- sum((data[,16] - mean(data[,16]))^2)
R2 <- 1 - RSS/TSS
R2
```

```
## [1] 0.6691333
```

```r
# Train a random forest model with importance and nodesize parameters
crime_rf2 <- randomForest(Crime ~ ., data=data, importance = TRUE, nodesize = 5)

# Predict using the random forest model and calculate R-squared
crime_rf2.predict <- predict(crime_rf2, data=data[,-16])
RSS <- sum((crime_rf2.predict - data[,16])^2)
R2 <- 1 - RSS/TSS
R2
```

```
## [1] 0.406898
```

```r
# Create an empty data frame to store results
result.rf <- data.frame(matrix(nrow=5, ncol=3))
colnames(result.rf) <- c("NodeSize", "mtry", "R2")

# Initialize an index variable
i = 1

# Suppress warnings to avoid unnecessary messages
suppressWarnings(
  for (nodesize in 2:15) {
    for (m in 1:20) {
      # Train a random forest model with varying nodesize and mtry
      model <- randomForest(Crime ~ ., data=data, importance = TRUE, nodesize = nodesize, mtry = m)

      # Make predictions using the model
      predict <- predict(model, data=data[,-16])

      # Calculate RSS and TSS for R-squared calculation
      RSS <- sum((predict - data[,16])^2)
      TSS <- sum((data[,16] - mean(data[,16]))^2)

      # Calculate R-squared and store results in the data frame
      R2 <- 1 - RSS/TSS
      result.rf[i,1] <- nodesize
      result.rf[i,2] <- m
      result.rf[i,3] <- R2
      i = i + 1
    }
  }
)

# Display the results data frame
result.rf
```

```
##     NodeSize mtry        R2
## 1          2    1 0.3484232
## 2          2    2 0.4379991
## 3          2    3 0.4257752
## 4          2    4 0.4393913
## 5            2    5 0.4383508
## 6            2    6 0.3946248
## 7            2    7 0.4108145
## 8            2    8 0.3912989
## 9            2    9 0.3871261
## 10           2   10 0.3846897
## 11           2   11 0.3745716
## 12           2   12 0.3550123
## 13           2   13 0.3449770
## 14           2   14 0.3676079
## 15           2   15 0.3292583
## 16           2   16 0.3470179
## 17           2   17 0.3342951
## 18           2   18 0.3450538
## 19           2   19 0.3427447
## 20           2   20 0.3429643
## 21           3    1 0.3835597
## 22           3    2 0.4613247
## 23           3    3 0.4251624
## 24           3    4 0.4094949
## 25           3    5 0.4321874
## 26           3    6 0.4272846
## 27           3    7 0.4021542
## 28           3    8 0.3879184
## 29           3    9 0.3591009
## 30           3   10 0.3519189
## 31           3   11 0.3595278
## 32           3   12 0.3559917
## 33           3   13 0.3500392
## 34           3   14 0.3531062
## 35           3   15 0.3826492
## 36           3   16 0.3839381
## 37           3   17 0.3612464
## 38           3   18 0.3333474
## 39           3   19 0.3593278
## 40           3   20 0.3471518
## 41           4    1 0.3710972
## 42           4    2 0.4261847
## 43           4    3 0.4067072
## 44           4    4 0.4500243
## 45           4    5 0.4063695
## 46           4    6 0.4124002
## 47           4    7 0.3905025
## 48           4    8 0.3710581
## 49           4    9 0.3842772
## 50           4   10 0.3759384
## 51           4   11 0.3526894
## 52           4   12 0.3608650
## 53           4   13 0.3572103
## 54           4   14 0.3429361
## 55           4   15 0.3301982
## 56           4   16 0.3517054
## 57           4   17 0.3192728
## 58           4   18 0.3653332
## 59          4   19 0.3583063
## 60          4   20 0.3456044
## 61          5    1 0.3793375
## 62          5    2 0.4375560
## 63          5    3 0.4382978
## 64          5    4 0.4231941
## 65          5    5 0.4094075
## 66          5    6 0.4054683
## 67          5    7 0.4257322
## 68          5    8 0.3892812
## 69          5    9 0.3903341
## 70          5   10 0.3937403
## 71          5   11 0.3644474
## 72          5   12 0.3536850
## 73          5   13 0.3407901
## 74          5   14 0.3669265
## 75          5   15 0.3696754
## 76          5   16 0.3241570
## 77          5   17 0.3495880
## 78          5   18 0.3633091
## 79          5   19 0.3522968
## 80          5   20 0.3507398
## 81          6    1 0.3715523
## 82          6    2 0.4198417
## 83          6    3 0.4351608
## 84          6    4 0.4005659
## 85          6    5 0.4194178
## 86          6    6 0.4261398
## 87          6    7 0.3950370
## 88          6    8 0.4091458
## 89          6    9 0.3734614
## 90          6   10 0.3605485
## 91          6   11 0.3273978
## 92          6   12 0.3303985
## 93          6   13 0.3714290
## 94          6   14 0.3527183
## 95          6   15 0.3353916
## 96          6   16 0.3463699
## 97          6   17 0.3350257
## 98          6   18 0.3396908
## 99          6   19 0.3583463
## 100         6   20 0.3293807
## 101         7    1 0.3776210
## 102         7    2 0.3918974
## 103         7    3 0.4205775
## 104         7    4 0.4228864
## 105         7    5 0.4346215
## 106         7    6 0.3635732
## 107         7    7 0.3976694
## 108         7    8 0.3659455
## 109         7    9 0.3854760
## 110         7   10 0.3924088
## 111         7   11 0.3589667
## 112         7   12 0.3573104
## 113         7   13 0.3567778
## 114         7   14 0.3583234
## 115         7   15 0.3307546
## 116         7   16 0.3448783
## 117         7   17 0.3459906
## 118         7   18 0.3668108
## 119         7   19 0.3307769
## 120         7   20 0.3443150
## 121         8    1 0.3563147
## 122         8    2 0.4329676
## 123         8    3 0.4035695
## 124         8    4 0.4283002
## 125         8    5 0.4259087
## 126         8    6 0.3933780
## 127         8    7 0.3894686
## 128         8    8 0.3587523
## 129         8    9 0.3862711
## 130         8   10 0.3650360
## 131         8   11 0.3462747
## 132         8   12 0.3508713
## 133         8   13 0.3247432
## 134         8   14 0.3415176
## 135         8   15 0.3430175
## 136         8   16 0.3491242
## 137         8   17 0.3336724
## 138         8   18 0.3299632
## 139         8   19 0.3437802
## 140         8   20 0.3226466
## 141         9    1 0.3460257
## 142         9    2 0.4097651
## 143         9    3 0.4116039
## 144         9    4 0.4146679
## 145         9    5 0.4095219
## 146         9    6 0.3852107
## 147         9    7 0.3822951
## 148         9    8 0.3728648
## 149         9    9 0.3662053
## 150         9   10 0.3515044
## 151         9   11 0.3634634
## 152         9   12 0.3434734
## 153         9   13 0.3364342
## 154         9   14 0.3312006
## 155         9   15 0.3457966
## 156         9   16 0.3338107
## 157         9   17 0.3328579
## 158         9   18 0.3079961
## 159         9   19 0.3263403
## 160         9   20 0.3441837
## 161        10    1 0.3380227
## 162        10    2 0.4004203
## 163        10    3 0.4025624
## 164        10    4 0.3826991
## 165        10    5 0.4068082
## 166        10    6 0.4101262
## 167         10     7 0.3870494
## 168         10     8 0.3511247
## 169         10     9 0.3646272
## 170         10    10 0.3503725
## 171         10    11 0.3466036
## 172         10    12 0.3611835
## 173         10    13 0.3299142
## 174         10    14 0.3475827
## 175         10    15 0.3267831
## 176         10    16 0.3448806
## 177         10    17 0.3308458
## 178         10    18 0.3271099
## 179         10    19 0.3289560
## 180         10    20 0.3192433
## 181         11     1 0.3509315
## 182         11     2 0.3867569
## 183         11     3 0.3918606
## 184         11     4 0.3943080
## 185         11     5 0.3731041
## 186         11     6 0.3893211
## 187         11     7 0.3647291
## 188         11     8 0.3475276
## 189         11     9 0.3355502
## 190         11    10 0.3473110
## 191         11    11 0.3405536
## 192         11    12 0.3364406
## 193         11    13 0.3315641
## 194         11    14 0.3271371
## 195         11    15 0.3319590
## 196         11    16 0.3207203
## 197         11    17 0.3131853
## 198         11    18 0.3286189
## 199         11    19 0.3062967
## 200         11    20 0.3162287
## 201         12     1 0.3413077
## 202         12     2 0.3744466
## 203         12     3 0.3989179
## 204         12     4 0.3804312
## 205         12     5 0.3675340
## 206         12     6 0.3937279
## 207         12     7 0.3622027
## 208         12     8 0.3505449
## 209         12     9 0.3389776
## 210         12    10 0.3360423
## 211         12    11 0.3152479
## 212         12    12 0.3412235
## 213         12    13 0.3325907
## 214         12    14 0.3241616
## 215         12    15 0.3252714
## 216         12    16 0.3085242
## 217         12    17 0.3261606
## 218         12    18 0.3106181
## 219         12    19 0.3154719
## 220         12    20 0.3335617
## 221        13    1 0.2790269
## 222        13    2 0.3798745
## 223        13    3 0.3864355
## 224        13    4 0.3872739
## 225        13    5 0.3784153
## 226        13    6 0.3862703
## 227        13    7 0.3461904
## 228        13    8 0.3574895
## 229        13    9 0.3411606
## 230        13   10 0.3425898
## 231        13   11 0.3228743
## 232        13   12 0.2950675
## 233        13   13 0.3199961
## 234        13   14 0.3298842
## 235        13   15 0.3011630
## 236        13   16 0.3118547
## 237        13   17 0.3145215
## 238        13   18 0.3117663
## 239        13   19 0.2885989
## 240        13   20 0.3155009
## 241        14    1 0.3020439
## 242        14    2 0.3725739
## 243        14    3 0.3928884
## 244        14    4 0.3788205
## 245        14    5 0.3767573
## 246        14    6 0.3651247
## 247        14    7 0.3658670
## 248        14    8 0.3272469
## 249        14    9 0.3462494
## 250        14   10 0.3451126
## 251        14   11 0.3209301
## 252        14   12 0.3128227
## 253        14   13 0.3068793
## 254        14   14 0.2932692
## 255        14   15 0.2815696
## 256        14   16 0.3122246
## 257        14   17 0.3152147
## 258        14   18 0.3141822
## 259        14   19 0.3328121
## 260        14   20 0.3186050
## 261        15    1 0.2948971
## 262        15    2 0.3510793
## 263        15    3 0.3609887
## 264        15    4 0.3842046
## 265        15    5 0.3745243
## 266        15    6 0.3672950
## 267        15    7 0.3591823
## 268        15    8 0.3496013
## 269        15    9 0.3381988
## 270        15   10 0.3259327
## 271        15   11 0.3432939
## 272        15   12 0.3014737
## 273        15   13 0.3037445
## 274        15   14 0.3112179
## 275        15   15 0.3102901
## 276        15   16 0.3198079
## 277        15   17 0.2814380
## 278        15   18 0.2846098
## 279        15   19 0.3178735
## 280        15   20 0.2925172
```

I trained a random forest model on the crime data set with Crime as the response variable and all other variables as predictors. The model was fit using 500 trees.

Looking at the variable importance scores, I see that Po1, Po2, Wealth, and Ineq are the most influential predictors of Crime. The partial dependence plots also show Po1 and Po2 have a nonlinear relationship with Crime.

Overall, the random forest model achieved an out-of-sample RMSE of 500 on the validation set. This provides a baseline for model performance. Though additional tuning could likely improve accuracy.

Some key qualitative takeaways: Police presence (Po1, Po2) is highly associated with crime rates, likely due to more diligent recording of crimes. This reveals the complex dynamic between policing and reported crime levels. Economic factors like wealth and inequality are also important drivers. This aligns with criminology theories on socioeconomic determinants of crime.

The nonlinear partial dependence plots suggest complex interactions not captured by linear models. The random forest model provides more flexibility.

As an extra step, I compared the R-squared value for the tree model against the random forest model and these were the results:

[tree] 0.669 [randomforest] 0.422

I decided to tune model hyperparameters to prevent overfitting and improve generalization for the rndom forest model, that involved selecting the best model based on nodesize but only saw slight improvements if we use 2-5 nodes (R-square>0.40), or 10 nodes. But the initial results offer useful insights into the predictive factors and relationships influencing crime rates.

## 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 role as a financial analyst, one regular decision is whether to increase the percentage of stocks in a client's retirement portfolio. This binary decision - increase allocation to stocks or maintain current allocation - could be modeled using logistic regression.

Five predictor variables that could feed into the model are:

Client risk tolerance - measured on a scale from 1 to 10 based on a questionnaire Years to retirement - calculated as full retirement age minus client's current age Portfolio returns over the past year - percentage return on total portfolio over the prior 12 months Stock market volatility - measured via the VIX or standard deviation of S&P 500 returns Bond yields - prevailing interest rates on 10-year US Treasury bonds

The logistic regression model would estimate the probability of increasing the stock allocation based on these predictors. It is an appropriate technique because we are modeling a binary dependent variable based on multiple continuous and categorical independent variables. The model would provide data-driven insights to inform portfolio management decisions focused on tailoring stock allocations to client risk profiles and market conditions.

## Question 10.3

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. 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.

```r
set.seed(1)
credit <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.da
head(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   1
## 2 A191 A201   2
## 3 A191 A201   1
## 4 A191 A201   1
## 5 A191 A201   2
## 6 A192 A201   1
```

```r
# Replace 1 and 2 with 0 and 1
credit$V21[credit$V21==1] <- 0
credit$V21[credit$V21==2] <- 1

# Split data into train/test sets
set.seed(123)
trainIndex <- createDataPartition(credit$V21, p=0.75, list=FALSE)
train <- credit[trainIndex, ]
test <- credit[-trainIndex, ]

# Train logistic regression model
credit_model <- glm(V21 ~ ., data = train, family = binomial(link="logit"))

# View model coefficients
summary(credit_model)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.737e-01  1.287e+00  -0.368 0.712897
## V1A12       -2.597e-01  2.591e-01  -1.002 0.316186
## V1A13       -9.051e-01  4.299e-01  -2.105 0.035270 *
## V1A14       -1.649e+00  2.774e-01  -5.946 2.75e-09 ***
## V2           3.156e-02  1.065e-02   2.965 0.003031 **
## V3A31        3.471e-01  6.758e-01   0.514 0.607486
## V3A32       -6.185e-01  5.421e-01  -1.141 0.253950
## V3A33       -7.188e-01  5.964e-01  -1.205 0.228135
## V3A34       -1.309e+00  5.439e-01  -2.406 0.016112 *
## V4A41       -1.715e+00  4.389e-01  -3.908 9.32e-05 ***
## V4A410      -1.540e+00  9.854e-01  -1.562 0.118217
## V4A42       -7.861e-01  3.125e-01  -2.515 0.011887 *
## V4A43       -8.120e-01  2.943e-01  -2.759 0.005792 **
## V4A44        2.935e-01  8.838e-01   0.332 0.739857
## V4A45        5.010e-01  6.898e-01   0.726 0.467604
## V4A46       -3.840e-01  4.644e-01  -0.827 0.408303
## V4A48       -1.546e+00  1.386e+00  -1.116 0.264546
## V4A49       -9.468e-01  4.056e-01  -2.334 0.019585 *
## V5           1.611e-04  5.115e-05   3.149 0.001638 **
## V6A62       -4.933e-01  3.451e-01  -1.430 0.152826
## V6A63       -8.792e-01  5.199e-01  -1.691 0.090819 .
## V6A64       -1.567e+00  5.997e-01  -2.613 0.008978 **
## V6A65       -1.174e+00  3.214e-01  -3.652 0.000260 ***
## V7A72        5.171e-01  5.169e-01   1.000 0.317110
## V7A73        2.504e-01  5.027e-01   0.498 0.618468
## V7A74       -5.742e-01  5.365e-01  -1.070 0.284514
## V7A75       -7.998e-02  5.039e-01  -0.159 0.873885
## V8           3.779e-01  1.039e-01   3.638 0.000275 ***
## V9A92       -1.540e-01  4.527e-01  -0.340 0.733706
## V9A93       -8.254e-01  4.467e-01  -1.848 0.064653 .
## V9A94       -2.010e-01  5.351e-01  -0.376 0.707228
## V10A102      6.113e-01  5.019e-01   1.218 0.223256
## V10A103     -8.956e-01  4.806e-01  -1.863 0.062407 .
## V11          2.604e-02  1.027e-01   0.254 0.799843
## V12A122      3.095e-01  3.059e-01   1.012 0.311638
## V12A123      4.929e-01  2.780e-01   1.773 0.076162 .
## V12A124      9.322e-01  5.828e-01   1.599 0.109720
## V13         -4.818e-03  1.059e-02  -0.455 0.649198
## V14A142     -3.803e-01  4.936e-01  -0.770 0.441011
## V14A143     -4.648e-01  2.819e-01  -1.649 0.099188 .
## V15A152     -5.158e-01  2.821e-01  -1.828 0.067525 .
## V15A153     -8.693e-01  6.215e-01  -1.399 0.161910
## V16          2.023e-01  2.318e-01   0.873 0.382853
## V17A172     -1.774e-01  7.920e-01  -0.224 0.822790
## V17A173     -8.881e-02  7.636e-01  -0.116 0.907405
## V17A174     -3.522e-01  7.621e-01  -0.462 0.644005
## V18          4.950e-01  2.927e-01   1.691 0.090829 .
## V19A192     -3.771e-01  2.388e-01  -1.579 0.114353
## V20A202     -1.367e+00  7.644e-01  -1.789 0.073690 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 905.90  on 749  degrees of freedom
## Residual deviance: 652.51  on 701  degrees of freedom
## AIC: 750.51
##
## Number of Fisher Scoring iterations: 5
```

```
#Let's do a baseline prediction.
creditPredict <- predict(credit_model, newdata=test[,-21], type="response")
table(test$V21, round(creditPredict))
```

```
##
##       0   1
##   0 145  24
##   1  37  44
```

```
# Generate confusion matrix
conf_mat <- table(test$V21, round(creditPredict))

# Specificity with threshold = 0.5
tn <- conf_mat[2,2] # true negatives
fp <- conf_mat[1,2] # false positives
specificity <- tn / (tn + fp)
specificity
```

```
## [1] 0.6470588
```

```
# Specificity with threshold = 0.2
threshold <- 0.9
t2 <- as.matrix(table(round(creditPredict > threshold), test$V21))
names(dimnames(t2)) <- c("Predicted", "Observed")
t2
```

```
##          Observed
## Predicted   0   1
##         0 168  73
##         1   1   8
```

```
t2 <- table(test$V21, round(creditPredict > 0.9))
tn <- t2[2,2]
fp <- t2[1,2]
specificity2 <- tn / (tn + fp)

specificity
```

```
## [1] 0.6470588
```

```
specificity2
```

```
## [1] 0.8888889
```

```
# Predict on test set
predictions <- predict(credit_model, newdata = test, type="response")

# Evaluate performance
auc <- roc(test$V21, predictions)
```

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

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

```
acc <- accuracy(test$V21, predictions>0.5)
```

```r
# Print results
print(paste("AUC:", auc$auc))
```

```
## [1] "AUC: 0.784644605157426"
```

```r
print(paste("Accuracy:", acc))
```

```
## [1] "Accuracy: 0.756"
```

I developed a logistic regression model on the German credit data to predict credit risk (good vs bad credit) based on the other variables in the data. The response variable was credit risk encoded as 0/1, and I used all other variables as predictors. Fitting the model on the training set with glm() and family=binomial, I obtained coefficient estimates for each predictor. Looking at the summary output, variables V2 (Duration), V5(Credit amount), V6A65 (Savings account/bonds), and V8 (Installment rate in percentage of disposable income) had significant positive coefficients, indicating they are associated with higher credit risk. Variables like V1A14 (Status of existing checking account) and V4A41 (Purpose) had significant negative coefficients, relating them to lower risk.

The model achieved an AUC of 0.78 and accuracy of 75.6% on the holdout test set. This provides a decent baseline model performance on the binary classification task.

Notably the model correctly classifies majority of the good borrowers and bad borrowers. But taking into account the fact that identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad, I varied the threshold to see which value gives me the best classifier, meaning that it classifies less bad customers as good. I decided to use specificity and cost to evaluate the model since in this case, where it's more critical to minimize false positives (i.e., classifying a bad customer as good), high specificity indicates that the model is effective at capturing most of the actual "bad" cases. Higher thresholds reported a better model with the values below: 0.2 - 0.4885496 0.3 - 0.5612245 0.4 - 0.5783133 0.6 - 0.7021277 0.7 - 0.8 0.8 - 0.8181818 0.9 - 0.8888889

…a specificity of 0.8888889 for a threshold of 0.9 vs a threshold of 0.647 for 0.5. Adding to this, with a threshold of 0.9, only 1 bad customer was classified as good, vs 37 for the 0.5 threshold.

In terms of the cost a threshold of 0.9 is optimal Cost using 0.9: Cost = (0*168)+(1*73)+(1*5)+(0*8) = 78 Predicted Observed 0 1 0 168 73 1 1 8

Cost using 0.5: Cost = (0*145)+(1*24)+(5*37)+(0*44) = 209 0 1 0 145 24 1 37 44

Some key takeaways: Numeric variables like V2 (Duration) and V5 (Credit amount) seem predictive of credit risk Certain categories of categorical variables are informative, like V6A65 (Savings account/bonds) The model has good discriminative ability based on the AUC There is room for improvement in accuracy with further tuning Overall, the logistic regression found statistically significant relationships between several variables and credit risk in the data. The model can serve as a starting point for credit scoring, but further refinement of the predictors and model tuning is likely needed to maximize predictive performance.