Lab 7

When a guest arrives they will count how many sides it has on

Alice Chang

Growing the full classification tree

```
# Load data
library(MASS)
library(dplyr)
```

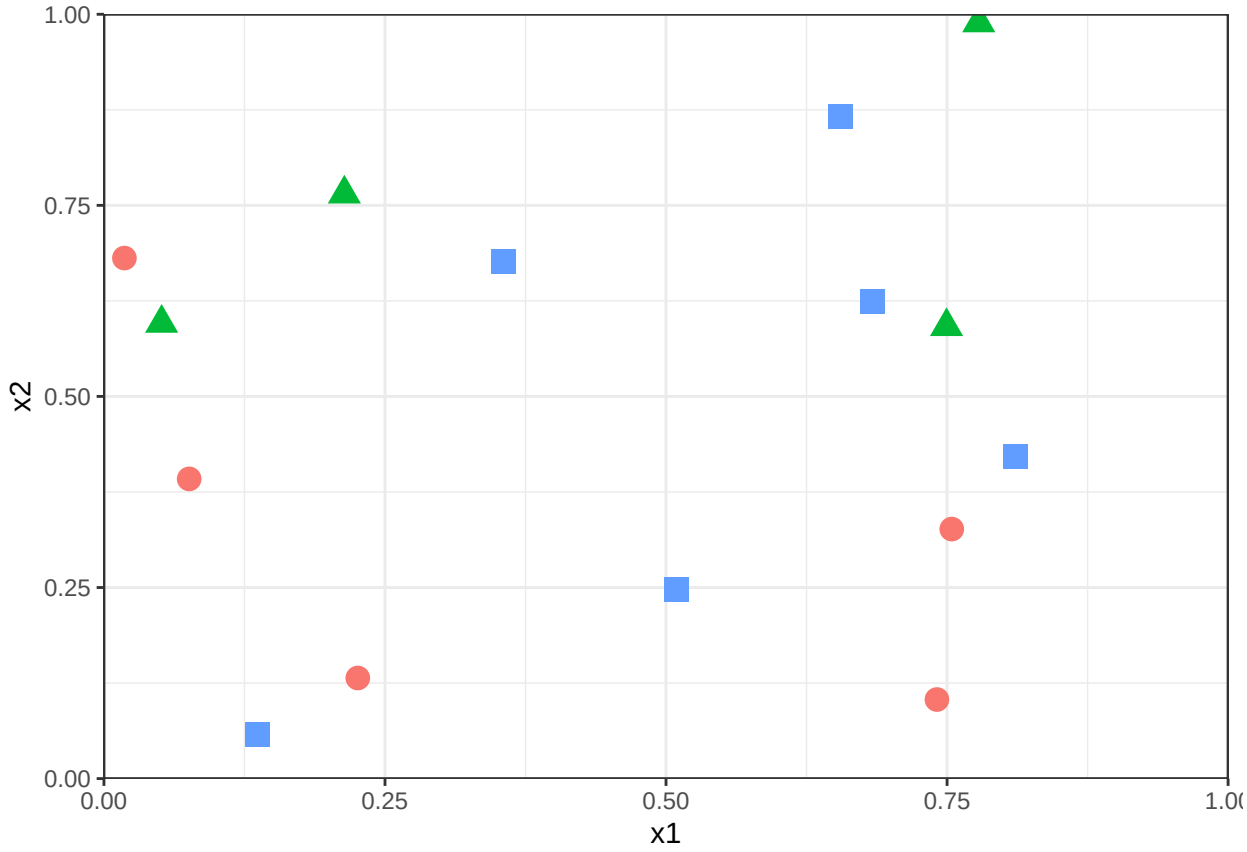
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

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

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

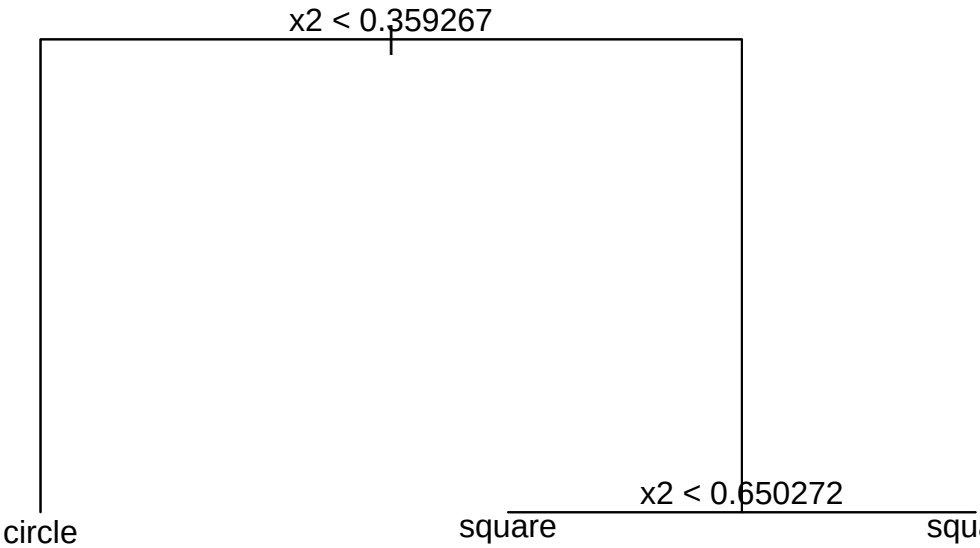
```
library(tree)
set.seed(75)
n <- 16
x1 <- runif(n)
x2 <- runif(n)
group <- as.factor(sample(1:3, n, replace = TRUE))
levels(group) <- c("circle", "triangle", "square")
df <- data.frame(x1, x2, group)
df[1, 2] <- .765 # tweaks to make a more interesting configuration
df[9, 1] <- .741
df <- df[-7, ]

library(ggplot2)
ggplot(df, aes(x = x1, y = x2, col = group, shape = group)) +
  geom_point(size = 4) +
  scale_x_continuous(expand = c(0, 0) , limits = c(0, 1)) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 1)) +
  scale_color_discrete(guide = FALSE) +
  scale_shape_discrete(guide = FALSE) +
  theme_bw()
```




```
#Fit tree w/ Gini
t1 <- tree(group ~ .,
           data = df, split = "gini")
summary(t1)
```

```
##
## Classification tree:
## tree(formula = group ~ ., data = df, split = "gini")
## Variables actually used in tree construction:
## [1] "x2"
## Number of terminal nodes:  3
## Residual mean deviance:  2.319 = 27.83 / 12
## Misclassification error rate: 0.5333 = 8 / 15
```

```
plot(t1)
text(t1, pretty = 0)
```



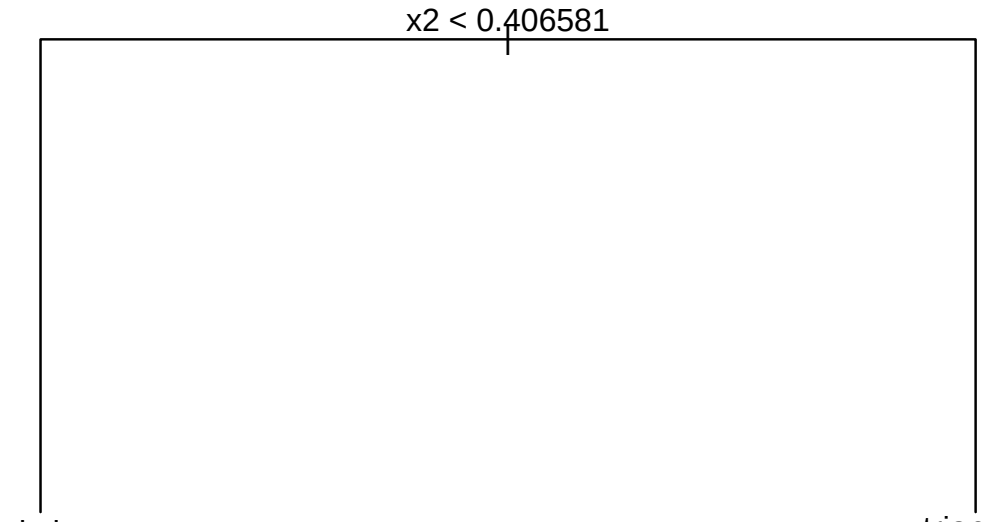
1. Neither the horizontal split around x2 = 0.5 nor the vertical split around x1 = 0.30 were decided upon by my classification tree as the first split. However, the classification tree's identification of the first split at x2 = 0.352967 is somewhat close to the first horizontal split commonly identified in class. 2. As a "greedy" model, the tree decides splits to ensure optimal node purity, so at times it does not consider every possible split. While the second split of the tree at x2 = 0.650272 predicts sqare for x2 > 0.650272 as well as x2 < 0.650272, the split ensures optimal node purity in each region. 3. It would predict a square.

An alternate metric

```
t2 <- tree(group ~ .,
           data = df, split = "deviance")
summary(t2)
```

```
##
## Classification tree:
## tree(formula = group ~ ., data = df, split = "deviance")
## Variables actually used in tree construction:
## [1] "x2"
## Number of terminal nodes:  2
## Residual mean deviance:  1.924 = 25.01 / 13
## Misclassification error rate: 0.4667 = 7 / 15
```

```
plot(t2)
text(t2, pretty = 0)
```



The Gini index measures the probabiltiy that a variable will be wrongly classified if it is randomly chosen. As the Gini index measures total variance across the K classes, a model based on the Gini index decides optimal splits that ensure node purity. In contrast to the Gini Index, the deviance metric is able to perform model comparison and select a model that minimizes the size of the tree while also optimizing the goodness of fit to the training data.

Crime and Communities, revisited

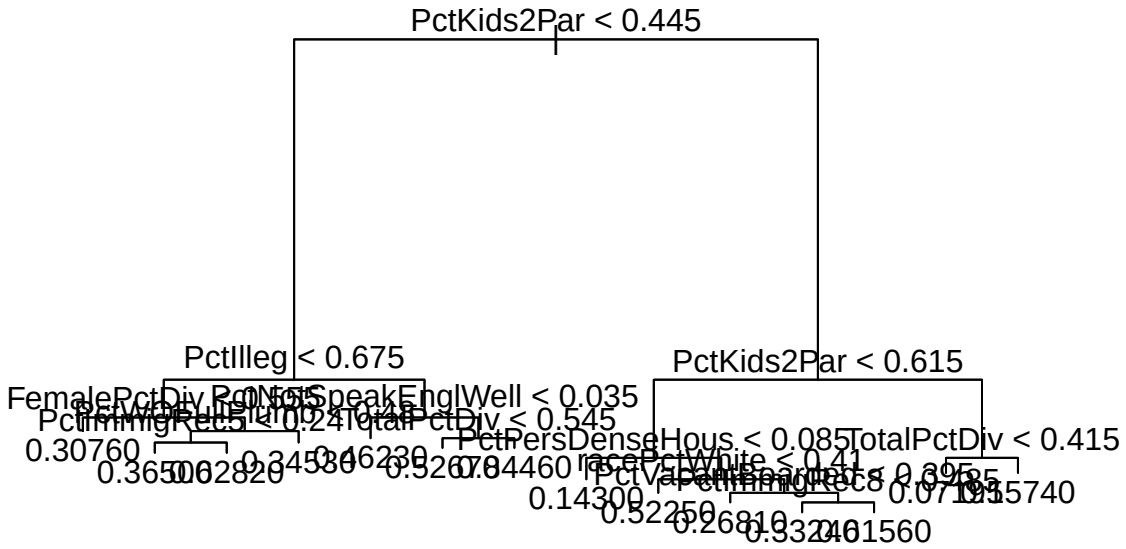
Growing a pruned regression tree

```
d <- read.csv("http://andrewpbray.github.io/data/crime-train.csv")

d_cleaned <- data.frame("blank" = rep(0, nrow(d)))
counter <- 1

for(i in 5:ncol(d)){
  if(d[1, i] != "?"){
    d_cleaned[[counter]] <- d[[i]]
    colnames(d_cleaned)[counter] <- colnames(d)[i]
    counter <- counter + 1
  }
}
t3 <- tree(ViolentCrimesPerPop ~ .,
           data = d_cleaned)

summary(t3)
```

```
##
## Regression tree:
## tree(formula = ViolentCrimesPerPop ~ ., data = d_cleaned)
## Variables actually used in tree construction:
##  [1] "PctKids2Par"         "PctIlleg"            "FemalePctDiv"
##  [4] "PctWOFullPlumb"      "PctImmigRec5"        "PctNotSpeakEnglWell"
##  [7] "TotalPctDiv"         "PctPersDenseHous"    "racePctWhite"
## [10] "PctVacantBoarded"    "PctImmigRec8"
## Number of terminal nodes:  14
## Residual mean deviance:  0.01572 = 12.36 / 786
```

```
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.38820 -0.06195 -0.02195  0.00000  0.05261  0.55260
```

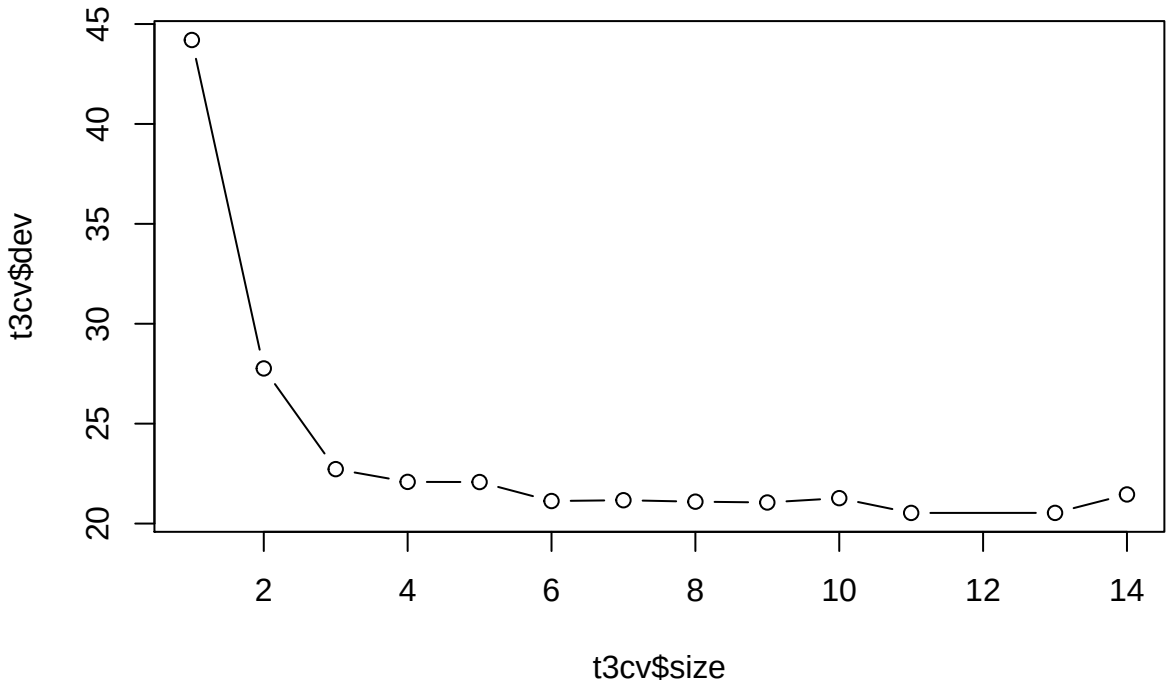
```
plot(t3)
text(t3, pretty = 0)
```




```
set.seed(40)
t3cv <- cv.tree(t3)
t3cv
```

```
## $size
##  [1] 14 13 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 21.45633 20.53357 20.53246 21.27094 21.05452 21.09503 21.16761
##  [8] 21.12668 22.07763 22.08556 22.72322 27.76323 44.20042
##
## $k
##  [1]       -Inf  0.5176420  0.5222261  0.5544045  0.6118008  0.7251080
##  [7]  0.7326129  0.8050502  1.1126087  1.1522867  2.0273988  4.1941945
## [13] 18.2672161
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

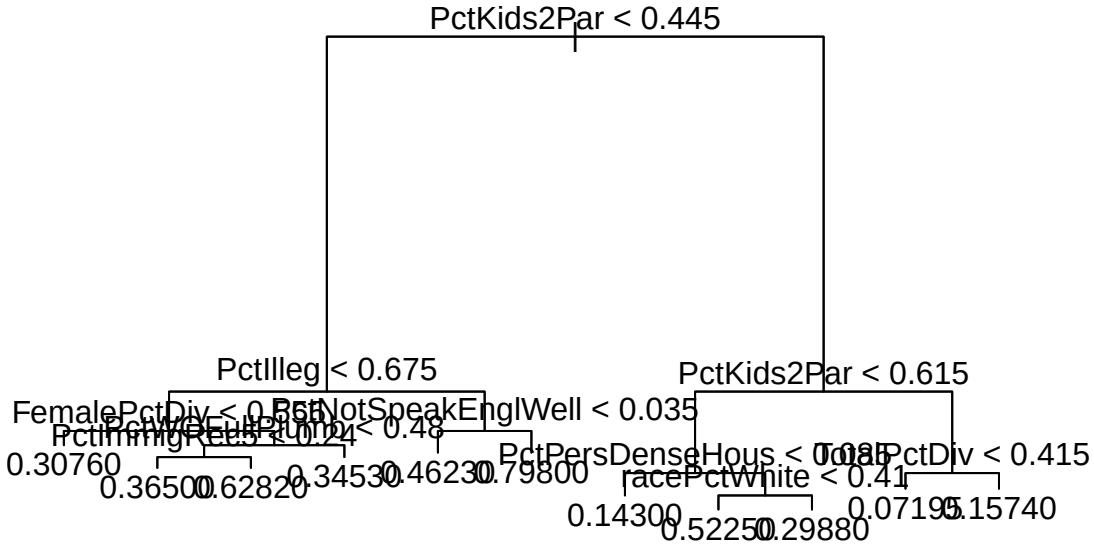
```
plot(t3cv$size, t3cv$dev, type = "b")
```




```
t3cv$size[which.min(t3cv$dev)]
```

```
## [1] 11
```

```
prune.t3 = prune.tree(t3, best = 11)
plot(prune.t3)
text(prune.t3, pretty = 0)
```



Comparing predictive performance

```
#load test data
test_data <- read.csv("https://bit.ly/2PYS8Ap")
#clean test_data
```

```
crime_test <- data.frame("blank" = rep(0, nrow(test_data)))
counter <- 1

for(i in 5:ncol(test_data)){
  if(test_data[1, i] != "?"){
    crime_test[[counter]] <- test_data[[i]]
    colnames(crime_test)[counter] <- colnames(test_data)[i]
    counter <- counter + 1
  }
}
#predict into test data
yhat_tree = predict(t3, crime_test)

tree_MSE <- mean((yhat_tree-crime_test$ViolentCrimesPerPop)^2)
tree_MSE
```

```
## [1] 0.01544466
```

```
#group A test MSE

lm <- lm(sqrt(ViolentCrimesPerPop)~ PctIlleg +
racePctWhite +
TotalPctDiv +
PctVacantBoarded:racePctWhite +
LemasPctOfficDrugUn +
MedRentPctHousInc +
PctNotHSGrad, data = d_cleaned)

lm_test_mse <- mean((sqrt(crime_test$ViolentCrimesPerPop) - predict(lm, crime_test))^2)

lm_test_mse
```

```
## [1] 0.01647594
```

Using the pruned regression tree, the test MSE is 0.01544466. This is slightly lower than the test MSE resulting from our group's regression model which was around 0.01647594.

Growing a random forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

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

```
#bagging
library(dplyr)
```

```r
#glimpse(d_cleaned)
set.seed(1)
bag.crime = randomForest(ViolentCrimesPerPop ~., data = d_cleaned, mtry = 100, importance = TRUE)

yhat.bag = predict(bag.crime, newdata = test_data)
mean((yhat.bag - test_data$ViolentCrimesPerPop)^2)
```

```
## [1] 0.003120125
```

```r
set.seed(1)
rf.crime = randomForest(ViolentCrimesPerPop ~., data = d_cleaned, mtry = 100/3, importance = TRUE)

yhat.rf = predict(rf.crime, newdata = test_data)
mean((yhat.rf - test_data$ViolentCrimesPerPop)^2)
```

```
## [1] 0.003194281
```

Both test MSEs were an improvement over the vanilla pruned regression tree (0.01544466) as well as our group's regression model(0.01647594). The test MSE from bagging was 0.003120125, while the test MSE from the second random forest model was slightly higher at 0.003194281.

Variable importance

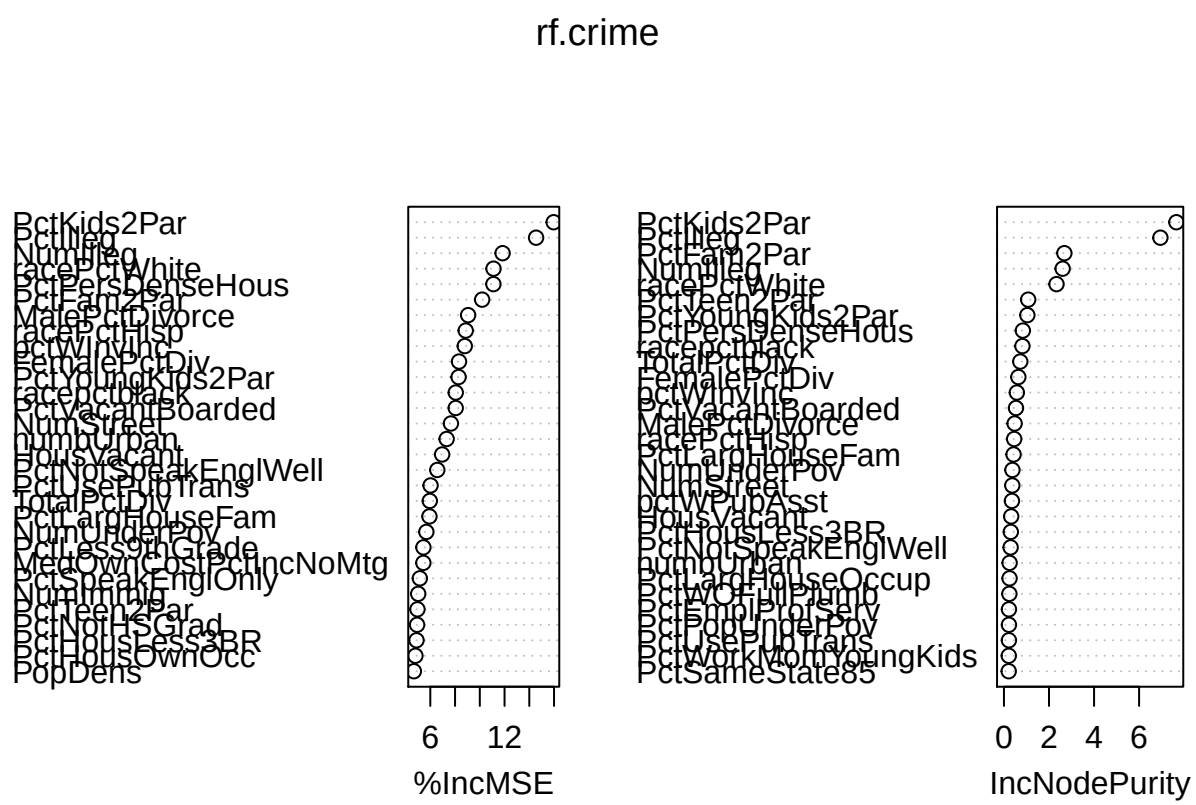
```r
importance(rf.crime)
```

```
##                         %IncMSE IncNodePurity
## population            3.7767532   0.129735912
## householdsize         3.7044873   0.107791331
## racepctblack          8.0553325   0.819356478
## racePctWhite         11.1083682   2.333703420
## racePctAsian          2.7776905   0.118166167
## racePctHisp           8.8631010   0.454163456
## agePct12t21           2.5853914   0.134709461
## agePct12t29           3.5792516   0.124325695
## agePct16t24           2.5619712   0.105624441
## agePct65up            2.2752883   0.113472526
## numbUrban             7.3195592   0.251459420
## pctUrban              4.1806051   0.113858954
## medIncome             3.5077708   0.070277949
## pctWWage              1.0364704   0.114539867
## pctWFarmSelf          1.3047395   0.192026964
## pctWInvInc            8.7855884   0.572235181
## pctWSocSec            3.5312314   0.131670080
## pctWPubAsst           4.4809679   0.354062385
## pctWRetire            1.2025063   0.132057802
## medFamInc             3.0559132   0.093314582
## perCapInc             2.0636072   0.091973576
## whitePerCap           1.7112283   0.100605989
## blackPerCap           3.5656202   0.157307712
## indianPerCap          2.4641833   0.161739946
## AsianPerCap           1.8761981   0.190662171
## OtherPerCap           3.9140895   0.199963110
## HispPerCap            0.4511918   0.149438053
## NumUnderPov           5.6809751   0.375198491
## PctPopUnderPov        3.9479037   0.221154600
```

```
## PctLess9thGrade        5.4421598   0.196069989
## PctNotHSGrad           4.9339928   0.165404389
## PctBSorMore            4.3768439   0.206303369
## PctUnemployed          3.4506582   0.186324933
## PctEmploy              2.0823900   0.118165081
## PctEmplManu            0.2420780   0.198444600
## PctEmplProfServ        1.4229789   0.221894923
## PctOccupManu           1.2897384   0.153216506
## PctOccupMgmtProf       1.7170440   0.203759127
## MalePctDivorce         9.0583256   0.473631250
## MalePctNevMarr         1.4351484   0.167141508
## FemalePctDiv           8.3218593   0.638982864
## TotalPctDiv            5.9590375   0.724605200
## PersPerFam             3.7131163   0.209048057
## PctFam2Par            10.1906737   2.681195098
## PctKids2Par           15.9701920   7.654037769
## PctYoungKids2Par       8.2860024   1.037903373
## PctTeen2Par            4.9612799   1.075134752
## PctWorkMomYoungKids    2.6064289   0.214589184
## PctWorkMom             3.3819036   0.198120058
## NumIlleg              11.8434235   2.606300843
## PctIlleg              14.5500310   6.944915990
## NumImmig               5.0248012   0.159469475
## PctImmigRecent         2.5577906   0.162315609
## PctImmigRec5           4.5745402   0.182704137
## PctImmigRec8           3.2063170   0.152660646
## PctImmigRec10          2.1778833   0.179231964
## PctRecentImmig         2.3534177   0.117420827
## PctRecImmig5           2.7121929   0.086446603
## PctRecImmig8           3.5714434   0.083391511
## PctRecImmig10          3.4492904   0.118476666
## PctSpeakEnglOnly       5.1542909   0.186821521
## PctNotSpeakEnglWell    6.5641843   0.295861578
## PctLargHouseFam        5.9306218   0.425844718
## PctLargHouseOccup      3.8056334   0.249100616
## PersPerOccupHous       2.5239261   0.087745700
## PersPerOwnOccHous      3.8389334   0.133433640
## PersPerRentOccHous     1.6314198   0.146046839
## PctPersOwnOccup        4.6331223   0.142262916
## PctPersDenseHous      11.0993825   0.838681884
## PctHousLess3BR         4.8851549   0.304074503
## MedNumBR               1.6002324   0.004449586
## HousVacant             6.9564787   0.320524685
## PctHousOccup           2.4208835   0.192699593
## PctHousOwnOcc          4.7974275   0.131057806
## PctVacantBoarded       8.0531495   0.534177944
## PctVacMore6Mos         1.2368519   0.132476255
## MedYrHousBuilt         3.4683494   0.170533697
## PctHousNoPhone         3.6126151   0.147875093
## PctWOFullPlumb         2.5829270   0.240285984
## OwnOccLowQuart         2.8406953   0.095924238
## OwnOccMedVal           1.9607854   0.078054861
## OwnOccHiQuart          3.3991148   0.073080414
## RentLowQ               2.9244871   0.106160208
```

```
## RentMedian              1.6720072   0.087354760
## RentHighQ               3.0999750   0.082778128
## MedRent                 3.0429773   0.089828498
## MedRentPctHousInc       2.6825195   0.186032580
## MedOwnCostPctInc        2.0489555   0.152315219
## MedOwnCostPctIncNoMtg   5.4380786   0.200876804
## NumInShelters           3.2915133   0.161725554
## NumStreet               7.6670020   0.369993793
## PctForeignBorn          2.6647163   0.151467689
## PctBornSameState        3.9002411   0.179665198
## PctSameHouse85          2.5494862   0.123195084
## PctSameCity85           0.9527450   0.138396171
## PctSameState85          1.9620418   0.211405592
## LandArea                2.8343560   0.129491803
## PopDens                 4.6704881   0.141419894
## PctUsePubTrans          6.0208490   0.220831943
## LemasPctOfficDrugUn     3.4614876   0.053454846
```

```
varImpPlot(rf.crime)
```




```
summary(lm)
```

```
##
## Call:
## lm(formula = sqrt(ViolentCrimesPerPop) ~ PctIlleg + racePctWhite +
##     TotalPctDiv + PctVacantBoarded:racePctWhite + LemasPctOfficDrugUn +
##     MedRentPctHousInc + PctNotHSGrad, data = d_cleaned)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -0.45886 -0.08039 -0.01261  0.07474  0.51942
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.29282    0.03821   7.663 5.29e-14 ***
## PctIlleg                       0.19180    0.03946   4.861 1.41e-06 ***
## racePctWhite                  -0.28866    0.03286  -8.784  < 2e-16 ***
## TotalPctDiv                    0.38817    0.03053  12.713  < 2e-16 ***
## LemasPctOfficDrugUn            0.07830    0.01970   3.975 7.68e-05 ***
## MedRentPctHousInc              0.10998    0.02921   3.765 0.000179 ***
## PctNotHSGrad                   0.11382    0.02917   3.902 0.000103 ***
## racePctWhite:PctVacantBoarded  0.09887    0.03557   2.780 0.005564 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.129 on 792 degrees of freedom
## Multiple R-squared:  0.669,  Adjusted R-squared:  0.666
## F-statistic: 228.6 on 7 and 792 DF,  p-value: < 2.2e-16
```

The results indicate that racePctWhite, PctIlleg, NumIlleg, PctKids2Par, and PctFam2Par are particularly important variables. In accordance with these results, the regression coefficients from our group's model also show that the proportion of whites, racePctWhite, and undocumented residents, PctIlleg, in the population are particularly influential in the decrease in violent crime in the area (β_2 = -0.28866 and β_1 = 0.19180). However, our logistic regression model identifies TotalPctDiv (β_3 = 0.38817) as the most influential variable, while the variable importance plot does not register its particular importance. LemasPctOfficeDrugUn, MedRentPctHousInc, PctNotHSGrad have smaller regression coefficients in the logistic regression model (β_4 = 0.07830,β_5 = 0.10998, β_6 =0.11382) and do not register as particularly important variables in the variable importance plot. While the effect of the interaction between racePctWhite and PctVacantBoarded is significant in the logistic regression model, it is not accounted for in the variable importance plot for the random forest model.