Title: Using machine learning to detect determinants of Violence Against Women

This project looks at data from the Nepal Demographic and Health Survey (DHS) 2022 to better understand issues of violence against women. It uses information from women who took part in the Domestic Violence section of the survey.

The main goal is to organize and prepare the data so we can identify patterns — for example, whether experiences of violence are linked to factors like education, household income, or where someone lives.

To explore these patterns, the project uses a Classification and Regression Tree (CART) method — a type of machine-learning approach that helps identify which factors are most strongly associated with experiences of violence.

The notebook shows, step by step, how to:

1. Clean and organize the raw DHS data,
2. Combine related pieces of information (such as family details or household size), and
3. Create a single, ready-to-use dataset for further analysis or computer modeling using CART or similar statistical methods.
4. Output of the analysis

DHS: Nepal 2020 - Any Type of VAW

Preparing the environment

Installing necessary packages

Sourcing the packages

```
library(rpart)
library(rpart.plot)
library(haven)
library(data.table)
library(dplyr)
```

Step 1: Preparing the Data

```
rm(list=ls()) #remove all objects
```

Define indicator and circumstances, add in variable names

```
# define desired indicator and circumstances
indicator <- "AllViolence"
# Wealth (v190), residence (v025), respondent’s education (v106), number of children under 5 yea
rs old2 (b8* and v201), respondent’s
# circumstances <- c("PoorerHousehold", "Residence", "Education", "NumberBirths")
circumstances <- c("PoorerHousehold", "Residence", "aGroup", "NUnder5", "Education")

# The following are codes for "Any type of VAW", "SampleWeight", "PoorerHousehold", "Residence",
"Education" which will be columns of dataframe fetched from DHS file
# Any type of VAW (d104, d106, d107, d108)
# requiredVarNames=c("D104","D106","D107","D108","D005","V044","V190","V025","V106","B8","V201")
requiredVarNames=c("D104","D106","D107","D108","D103A","D103B","D103C","D105A","D105B","D105
C","D105D","D105E","D105F","D105H","D105I","D105J","D105K","D005","V044","V502","V190","V025","V
012","V106","V001","V002")
```

Read the DHS Data (Stata version)

```
# DHS provides datasets in different formats. Here we use the Stata version.
# Please ensure your .dta files are in the dta folder. This command defines where to access and
read the data.
ir_data_file <- "C:/Users/alamichhane_unfpa/OneDrive/Archive/Ashish Work/4. UNFPA/Data modellin
g/CART/NepalTechnicalRTraining/Nepal_DHS2022/dta/NPIR82FL.DTA"
df <- read_dta(ir_data_file,
               #n_max = 10), #this argument specifies to read only 10 rows, handy when checking
a big dataset preliminarily before spending the time loading the full version
               col_select = tolower(requiredVarNames))
df <- zap_labels(df) #this removes data labels. sometimes labels from other data formats can bri
ng bugs.
names(df) <- toupper(names(df)) #convert variable names to upper cases to be consistent with  da
ta dictionary
```

Step 2: Generating variables

Preprocess Sample Weight column to the dataframe

```
df$SampleWeight<-as.numeric(as.character(df$D005))/1000000
df$SampleWeight[is.na(df$SampleWeight)] <- 0

sum(df$SampleWeight)
```

```
## [1] 5177.121
```

Construct VAW Variable (any type of VAW)

```
#First assign datause as dataframe that will be used for the CART algorithm. In datause create t
he different columns that process the data and prepare it for CART. Datause is exactly df.
datause<-df

# Filter the denominator
ViV<-"V044" #flag of whether there is domestic violence module
ViK<-match(ViV, colnames(datause))
datause<-datause[datause[, ViK]==1, ] ### keep only women selected for the module and interviewe
d
datause<-datause[datause$V502 %in% c(1,2), ] # keep women currently or formerly in partnership

sum(datause$SampleWeight)
```

```
## [1] 4030.736
```

```
# code for AllViolence
VarName<-c("D103A","D103B","D103C","D105A","D105B","D105C","D105D","D105E","D105F","D105H","D105
I","D105J","D105K")
k<-match(VarName, colnames(datause), nomatch = 0)
print(k)
```

```
##  [1] 10 11 12 14 15 16 17 18 19 20 21 22 23
```

```
l<-length(k)

datause$var2tab<- 0 #assign initial value to zero and then replace if any type of violence repor
ted

for(i in k){ #iterate over each type of violence experienced often/sometimes in the past 12 mont
hs
  datause$v <- unlist(datause[,i])
  datause$var2tab[!is.na(datause$v) & (datause$v  %in% c(1, 2)) ] <- 1
}
```

Generate Circumstances - NUnder 5 (PR)

```
#To generate the number of children under 5 in a household (NUnder5) we must create a separate d
ataframe (df2) and aggregate the data by household ID and cluster ID.

# Need to load the PR dataset
vars <- c("HV105","HV001", "HV002")
pr_data_file <- "dta/NPPR82FL.DTA"
df2 <- read_dta(pr_data_file,
                col_select = tolower(vars))
df2 <- zap_labels(df2)
names(df2) <- toupper(names(df2))


#define the variable used to identify age of each child in household
ageV<-"HV105"

#create a new data frame
df2 <- df2[,c("HV001", "HV002", ageV)]

#filter out children under 5 in household
df2 <- df2[df2$HV105<=5, ]

#create count column that will be aggregated
df2$ct<-1
Under5<-aggregate(df2$ct, list(df2$HV001, df2$HV002), sum)
# aggregate by Household number and cluster ID into a vector Under5
colnames(Under5)<-c("V001", "V002", "NUnder5")

#merge Under5 vector into datause dataframe for analysis
datause<-merge(datause, Under5, by=c("V001", "V002"), all.x=T)


#filter out na's
datause$NUnder5[is.na(datause$NUnder5)]<-0
```

Generate Circumstances - Wealth, Residence, Education, Age

Group

```
# Create Wealth Circumstance
VarName<-"V190"
k<-match(VarName, colnames(datause))
datause$PoorerHousehold <- "0"
datause$PoorerHousehold[datause$V190 %in% c(0,1,2)] <- "1"

#Create Residence Circumstance
VarName<-"V025"
k<-match(VarName, colnames(datause))
datause$Residence<-"Rural"
datause$Residence[datause[, k]==1]<-"Urban"
datause$Residence<-factor(datause$Residence, levels = c("Urban" , "Rural"))

# code for Education
VarName<-"V106"
k<-match(VarName, colnames(datause))

datause$Education<-"Lower"
datause$Education[datause[, k]== 0] <-"Lower"
datause$Education[datause[, k]== 1] <-"Lower"
datause$Education[datause[, k]== 2] <-"Secondary"
datause$Education[datause[, k]== 3] <-"Higher"
datause$Education<-factor(datause$Education, levels=c("Lower", "Secondary", "Higher"), ordered =
TRUE)

# code for aGroup
VarName<-"V012"
k<-match(VarName, colnames(datause))

datause$Age<-datause[ , k]
datause$aGroup<-"Missing"
datause$aGroup[!is.na(datause$Age) & datause$Age<15 ]="0-14"
datause$aGroup[!is.na(datause$Age) & datause$Age>=15 & datause$Age<25 ]="15-24"
datause$aGroup[!is.na(datause$Age) & datause$Age>=25 & datause$Age<35  ]="25-34"
datause$aGroup[!is.na(datause$Age) & datause$Age>=35 & datause$Age<98]="35+"
datause$aGroup<-factor(datause$aGroup, levels=c("0-14", "15-24", "25-34", "35+"))


head(datause,10)
```

```
##    V001 V002 V012 V025 V044 V106 V190 V502   D005 D103A D103B D103C D104 D105A
## 1     1    1   29    2    1    1    1    1 664034     0     0     0    0     0
## 2     1    8   33    2    1    1    1    1 664034     0     0     0    0     0
## 3     1   11   23    2    1    1    2    1 664034     0     0     0    0     0
## 4     1   23   24    2    1    1    1    1 664034     0     0     0    0     0
## 5     1   26   41    2    1    2    2    1 664034     0     0     0    0     0
## 6     1   33   22    2    1    2    1    1 664034     0     0     0    0     0
## 7     1   45   27    2    1    2    1    1 664034     0     0     0    0     0
## 8     2    4   33    2    1    1    1    1 702137     0     0     0    0     3
## 9     2    7   35    2    1    1    1    1 702137     0     0     0    0     0
## 10    2    9   32    2    1    2    2    1 702137     0     0     0    0     0
##    D105B D105C D105D D105E D105F D105H D105I D105J D105K D106 D107 D108
## 1      0     0     0     0     0     0     0     0     0    0    0    0
## 2      0     0     0     0     0     0     0     0     0    0    0    0
## 3      0     0     0     0     0     0     0     0     0    0    0    0
## 4      0     0     0     0     0     0     0     0     0    0    0    0
## 5      0     0     0     0     0     0     0     0     0    0    0    0
## 6      0     0     0     0     0     0     0     0     0    0    0    0
## 7      0     0     0     0     0     0     0     0     0    0    0    0
## 8      3     0     0     0     0     0     0     0     0    1    0    0
## 9      0     0     0     0     0     0     0     0     0    0    0    0
## 10     0     0     0     0     0     0     0     0     0    0    0    0
##    SampleWeight var2tab v NUnder5 PoorerHousehold Residence Education Age
## 1      0.664034       0 0       1               1     Rural     Lower  29
## 2      0.664034       0 0       2               1     Rural     Lower  33
## 3      0.664034       0 0       1               1     Rural     Lower  23
## 4      0.664034       0 0       0               1     Rural     Lower  24
## 5      0.664034       0 0       0               1     Rural Secondary  41
## 6      0.664034       0 0       1               1     Rural Secondary  22
## 7      0.664034       0 0       2               1     Rural Secondary  27
## 8      0.702137       0 0       1               1     Rural     Lower  33
## 9      0.702137       0 0       0               1     Rural     Lower  35
## 10     0.702137       0 0       1               1     Rural Secondary  32
##    aGroup
## 1   25-34
## 2   25-34
## 3   15-24
## 4   15-24
## 5     35+
## 6   15-24
## 7   25-34
## 8   25-34
## 9     35+
## 10  25-34
```

Step 3: Conducting Descriptive Analysis

National Level

Get average access rate across all households

```
# calculate overall mean
avg_access_rate<-weighted.mean(datause$var2tab, datause$SampleWeight)
print(paste("Avg national access rate:", avg_access_rate))
```

```
## [1] "Avg national access rate: 0.174034474876231"
```

Get average access rate for households from top 60 in wealth quantiles (one circumstance)

```
access_rate<-weighted.mean(datause[datause$PoorerHousehold==0,]$var2tab, datause[datause$PoorerH
ousehold==0,]$SampleWeight)
print(paste("Access rate for top60 wealth households", access_rate))
```

```
## [1] "Access rate for top60 wealth households 0.153401942940462"
```

Get average access rate for households from below 40 wealth and urban residence (two circumstances)

```
access_rate<-weighted.mean(datause[datause$PoorerHousehold==1 & datause$Residence=="Urban",]$var
2tab, datause[datause$PoorerHousehold==1 & datause$Residence=="Urban",]$SampleWeight)
print(paste("Access rate for below40 wealth households and lower education", access_rate))
```

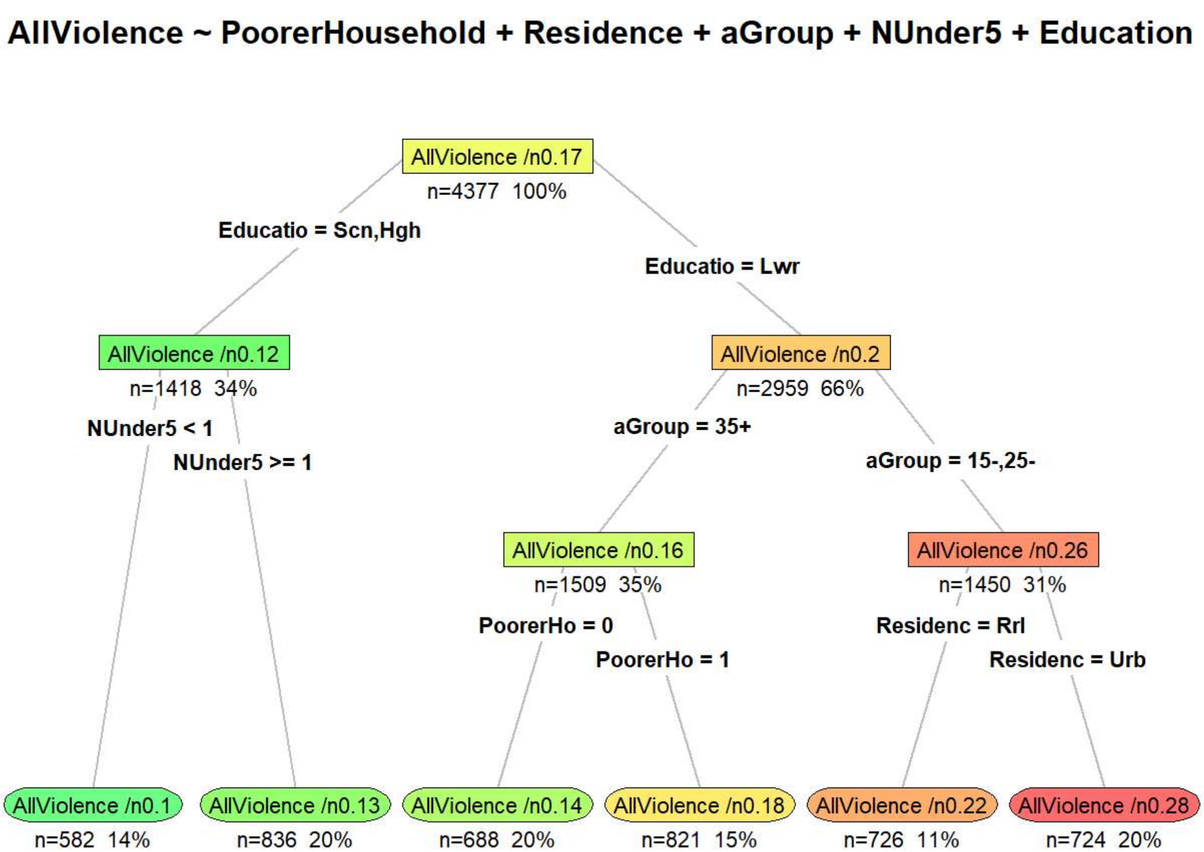
```
## [1] "Access rate for below40 wealth households and lower education 0.252658113349665"
```

Step 4: Running the CART

```
# construct formula_string to use in CART algorithm and D index calculation
formula_string<-paste("var2tab", paste(circumstances, collapse=" + "), sep=" ~ ")
# construct title string to use in title of generated tree
title_string<-paste(indicator, paste(circumstances, collapse=" + "), sep=" ~ ")
print(formula_string)
```

```
## [1] "var2tab ~ PoorerHousehold + Residence + aGroup + NUnder5 + Education"
```

```
# defind variables required in CART
cp_chosen<- 1
minb_chosen = 11

min_node<-max(49, nrow(datause)/minb_chosen)

# defind tree method and generate CART tree
treemethod<-"anova"
treefit <- rpart(as.formula(formula_string),
                 data = datause,  weights=SampleWeight,
                 method=treemethod, control = rpart.control(cp = cp_chosen/nrow(datause), maxdep
th=6,
                                                            minbucket = min_node, minsplit=2*min
_node))

# plot the tree
sub_string<-NULL
treeplot<- prp(treefit, main=title_string, sub=sub_string,
               type=4, fallen=T, branch=.3, round=0, leaf.round=9,
               clip.right.labs=F, under.cex=1,
               box.palette="GnYlRd",
               prefix=paste("AllViolence", "/n"), branch.col="gray", branch.lwd=2,
               extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1.0, cex.sub=0.7)
```



Step 5: Conducting Sensitivity Analysis (WIP)

National

Deduct wealth from circumstances

```
deducted_title_string<-paste("AllViolence ~ Residence + aGroup + NUnder5 + Education")
deducted_formula_string<-"var2tab ~ Residence + aGroup + NUnder5 + Education"
```

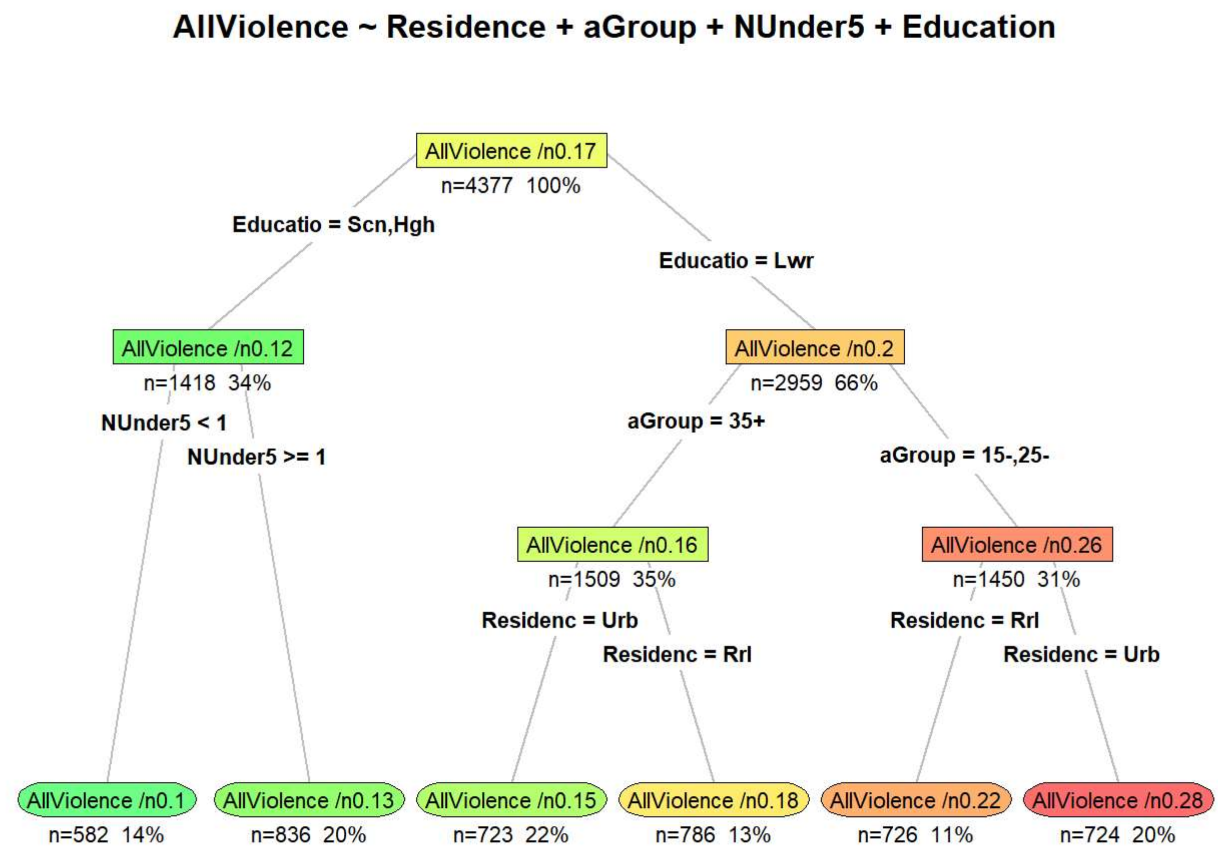
Build new model

```
deducted_treefit <- rpart(as.formula(deducted_formula_string), data = datause, weights=SampleWei
ght, method=treemethod, control = rpart.control(cp = cp_chosen/nrow(datause), maxdepth=6, minbuc
ket = min_node, minsplit=2*min_node))

print(deducted_treefit[["frame"]][["yval"]])
```

```
##  [1] 0.1740345 0.1167848 0.1001506 0.1289492 0.2037600 0.1579615 0.1464601
##  [8] 0.1778985 0.2551220 0.2155427 0.2779762
```

Plot the tree

```
# plot the tree
sub_string<-NULL
treeplot_deducted<- prp(deducted_treefit, main=deducted_title_string, sub=sub_string,
                        type=4, fallen=T, branch=.3, round=0, leaf.round=9,
                        clip.right.labs=F, under.cex=1,
                        box.palette="GnYlRd",
                        prefix=paste("AllViolence", "/n"), branch.col="gray", branch.lwd=2,
                        extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1.0, cex.sub=0.7)
```



###

change level of confidence/significance? (cp?)

```
# construct formula_string to use in CART algorithm
formula_string<-paste("var2tab", paste(circumstances, collapse=" + "), sep=" ~ ")
# construct title string to use in title of generated tree
title_string<-paste(indicator, paste(circumstances, collapse=" + "), sep=" ~ ")
print(formula_string)
```

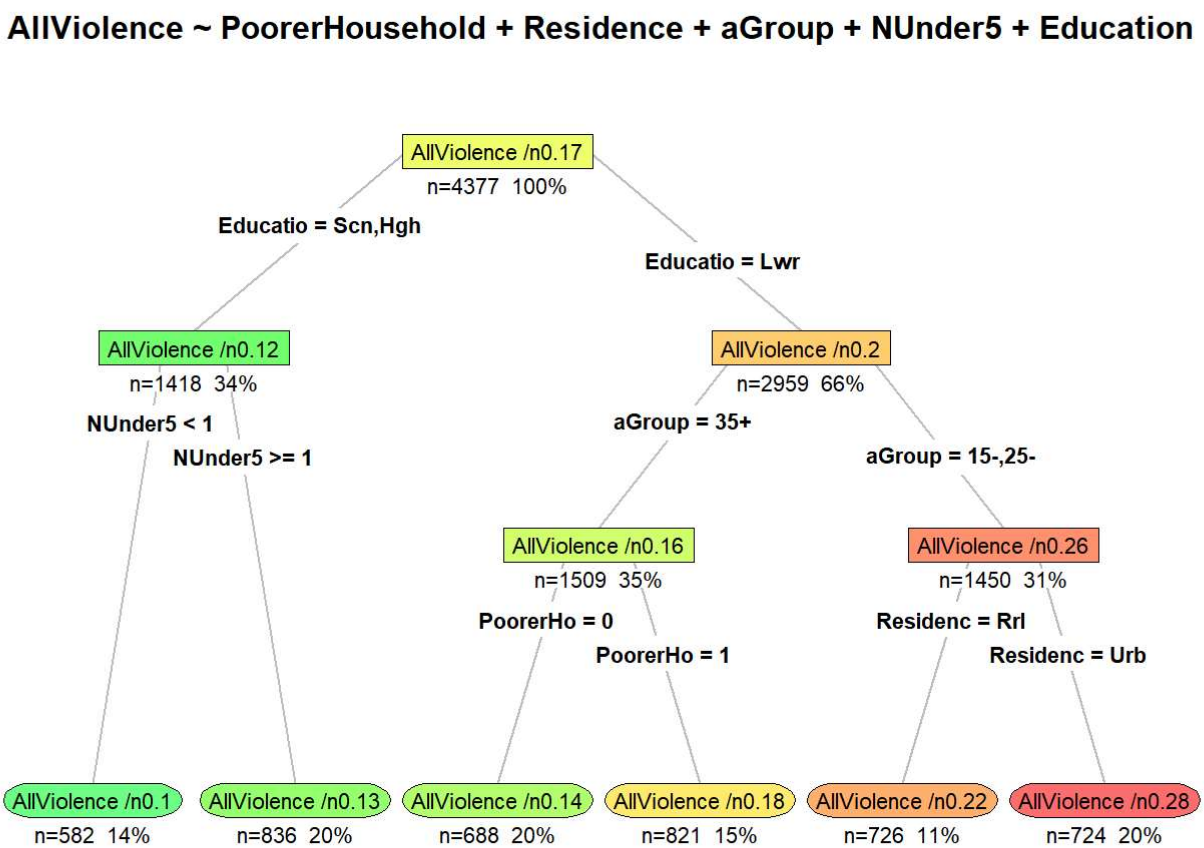
```
## [1] "var2tab ~ PoorerHousehold + Residence + aGroup + NUnder5 + Education"
```

```
# defind variables required in CART
cp_chosen<- 1.2 # MODIFY cp
minb_chosen = 11

min_node<-max(49, nrow(datause)/minb_chosen)

# defind tree method and generate CART tree
treemethod<-"anova"
treefit <- rpart(as.formula(formula_string),
                 data = datause,  weights=SampleWeight,
                 method=treemethod, control = rpart.control(cp = cp_chosen/nrow(datause), maxdep
th=6,
                                                            minbucket = min_node, minsplit=2*min
_node))

print(summary(treefit))
```

```
## Call:
## rpart(formula = as.formula(formula_string), data = datause, weights = SampleWeight,
##     method = treemethod, control = rpart.control(cp = cp_chosen/nrow(datause),
##         maxdepth = 6, minbucket = min_node, minsplit = 2 * min_node))
##   n= 4377
##
##             CP nsplit rel error    xerror       xstd
## 1 0.0118387461      0 1.0000000 1.0009059 0.02711034
## 2 0.0107714371      1 0.9881613 0.9995160 0.02689869
## 3 0.0019524316      2 0.9773898 0.9797671 0.02639044
## 4 0.0007321306      3 0.9754374 0.9798141 0.02641770
## 5 0.0004810919      4 0.9747053 0.9810258 0.02642987
## 6 0.0002741604      5 0.9742242 0.9806478 0.02642500
##
## Variable importance
##       Education          aGroup         NUnder5       Residence PoorerHousehold
##              39              36              15               7               3
##
## Node number 1: 4377 observations,    complexity param=0.01183875
##   mean=0.1740345, MSE=0.1437465
##   left son=2 (1418 obs) right son=3 (2959 obs)
##   Primary splits:
##       Education       splits as  RLL,     improve=1.183875e-02, (0 missing)
##       NUnder5         < 1.5 to the left,  improve=6.611779e-03, (0 missing)
##       PoorerHousehold splits as  LR,      improve=4.728422e-03, (0 missing)
##       aGroup          splits as  -RRL,    improve=3.309796e-03, (0 missing)
##       Residence       splits as  LR,      improve=2.850888e-06, (0 missing)
##
## Node number 2: 1418 observations,    complexity param=0.0004810919
##   mean=0.1167848, MSE=0.1031461
##   left son=4 (582 obs) right son=5 (836 obs)
##   Primary splits:
##       NUnder5         < 0.5 to the left,  improve=0.0019617270, (0 missing)
##       aGroup          splits as  -RLL,    improve=0.0018923490, (0 missing)
##       PoorerHousehold splits as  LR,      improve=0.0003418410, (0 missing)
##       Residence       splits as  LR,      improve=0.0000227508, (0 missing)
##   Surrogate splits:
##       aGroup splits as  -RRL, agree=0.709, adj=0.311, (0 split)
##
## Node number 3: 2959 observations,    complexity param=0.01077144
##   mean=0.20376, MSE=0.1622419
##   left son=6 (1509 obs) right son=7 (1450 obs)
##   Primary splits:
##       aGroup          splits as  -RRL,    improve=0.0144987400, (0 missing)
##       NUnder5         < 1.5 to the left,  improve=0.0080107800, (0 missing)
##       PoorerHousehold splits as  LR,      improve=0.0026365380, (0 missing)
##       Residence       splits as  RL,      improve=0.0002340662, (0 missing)
##   Surrogate splits:
##       NUnder5         < 0.5 to the left,  agree=0.700, adj=0.363, (0 split)
##       PoorerHousehold splits as  LR,      agree=0.541, adj=0.027, (0 split)
##
## Node number 4: 582 observations
```

```
##   mean=0.1001506, MSE=0.09012048
##
## Node number 5: 836 observations
##   mean=0.1289492, MSE=0.1123213
##
## Node number 6: 1509 observations,    complexity param=0.0007321306
##   mean=0.1579615, MSE=0.1330097
##   left son=12 (688 obs) right son=13 (821 obs)
##   Primary splits:
##       PoorerHousehold splits as  LR, improve=0.002273906, (0 missing)
##       Residence       splits as  LR, improve=0.001723954, (0 missing)
##   Surrogate splits:
##       Residence splits as  LR, agree=0.621, adj=0.131, (0 split)
##
## Node number 7: 1450 observations,    complexity param=0.001952432
##   mean=0.255122, MSE=0.1900348
##   left son=14 (726 obs) right son=15 (724 obs)
##   Primary splits:
##       Residence       splits as  RL,      improve=4.759933e-03, (0 missing)
##       PoorerHousehold splits as  LR,      improve=1.391711e-03, (0 missing)
##       aGroup          splits as  -RL-,    improve=1.222730e-03, (0 missing)
##       NUnder5         < 0.5 to the right, improve=9.717552e-05, (0 missing)
##
## Node number 12: 688 observations
##   mean=0.1426797, MSE=0.1223222
##
## Node number 13: 821 observations
##   mean=0.1777531, MSE=0.1461569
##
## Node number 14: 726 observations
##   mean=0.2155427, MSE=0.169084
##
## Node number 15: 724 observations
##   mean=0.2779762, MSE=0.2007054
##
## n= 4377
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 4377 579.40410 0.1740345
##    2) Education=Secondary,Higher 1418 142.09240 0.1167848
##      4) NUnder5< 0.5 582  52.43996 0.1001506 *
##      5) NUnder5>=0.5 836  89.37374 0.1289492 *
##    3) Education=Lower 2959 430.45220 0.2037600
##      6) aGroup=35+ 1509 186.55100 0.1579615
##       12) PoorerHousehold=0 688  96.81046 0.1426797 *
##       13) PoorerHousehold=1 821  89.31629 0.1777531 *
##      7) aGroup=15-24,25-34 1450 237.66020 0.2551220
##       14) Residence=Rural 726  77.40584 0.2155427 *
##       15) Residence=Urban 724 159.12310 0.2779762 *
```

```
# plot the tree
sub_string<-NULL
treeplot<- prp(treefit, main=title_string, sub=sub_string,
               type=4, fallen=T, branch=.3, round=0, leaf.round=9,
               clip.right.labs=F, under.cex=1,
               box.palette="GnYlRd",
               prefix=paste("AllViolence", "/n"), branch.col="gray", branch.lwd=2,
               extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1.0, cex.sub=0.7)
```



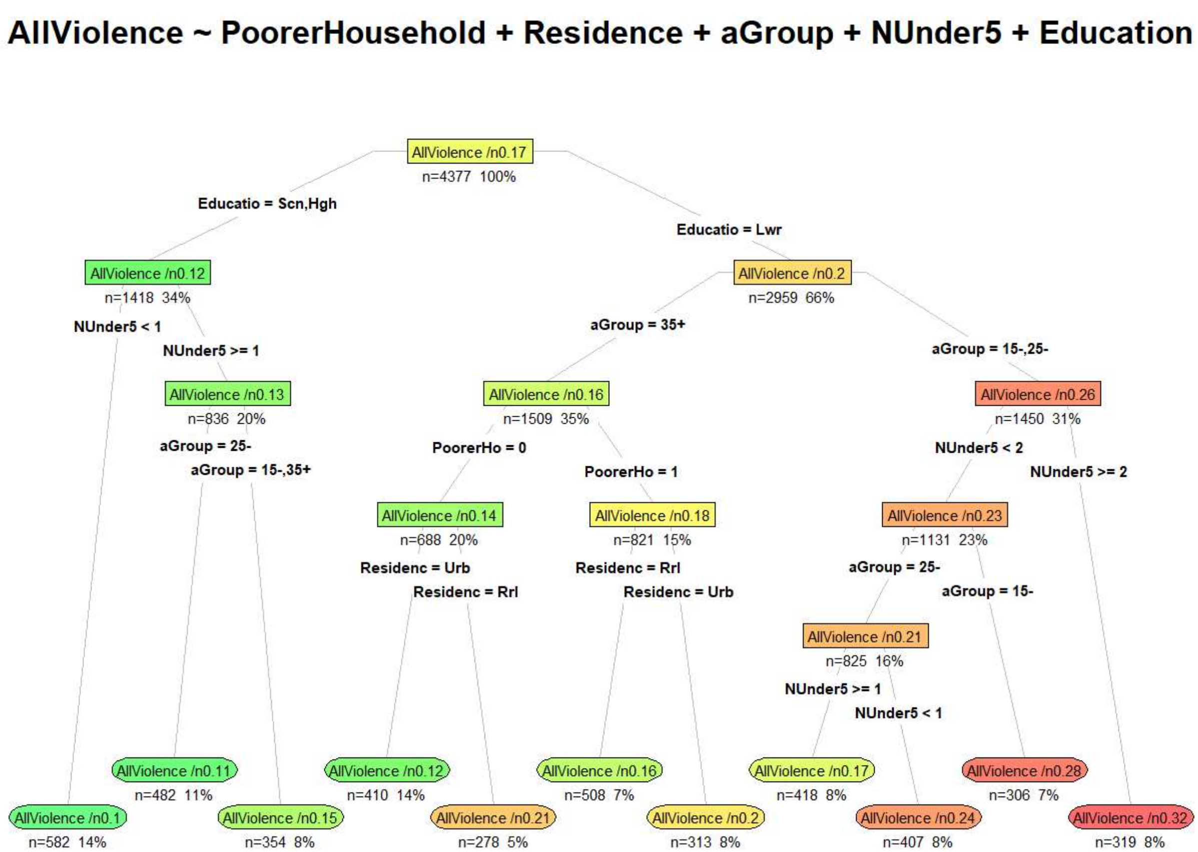
###

change thresholds for the minimum number of households: 9% to 5% or even 15%

```
# construct formula_string to use in CART algorithm
formula_string<-paste("var2tab", paste(circumstances, collapse=" + "), sep=" ~ ")
# construct title string to use in title of generated tree
title_string<-paste(indicator, paste(circumstances, collapse=" + "), sep=" ~ ")
print(formula_string)
```

```
## [1] "var2tab ~ PoorerHousehold + Residence + aGroup + NUnder5 + Education"
```

```
# defind variables required in CART
cp_chosen<- 1
minb_chosen = 19 # Modify minimum number of households

min_node<-max(49, nrow(datause)/minb_chosen)

# defind tree method and generate CART tree
treemethod<-"anova"
treefit <- rpart(as.formula(formula_string),
                 data = datause,  weights=SampleWeight,
                 method=treemethod, control = rpart.control(cp = cp_chosen/nrow(datause), maxdep
th=6,
                                                            minbucket = min_node, minsplit=2*min
_node))

# plot the tree
sub_string<-NULL
treeplot<- prp(treefit, main=title_string, sub=sub_string,
               type=4, fallen=T, branch=.3, round=0, leaf.round=9,
               clip.right.labs=F, under.cex=1,
               box.palette="GnYlRd",
               prefix=paste("AllViolence", "/n"), branch.col="gray", branch.lwd=2,
               extra=101, under=T, lt=" < ", ge=" >= ", cex.main=1.0, cex.sub=0.7)
```



Step 6: Calculating D Index

```
# We first assign initial overall d-index as zero for each input.
Overall_D = 0

opp_datause <- datause
opp_datause$var2tab<- 1 - opp_datause$var2tab

# Next we make sure that we have defined a list of circumstances and define the formula string f
or calculating the D-index.

if(length(circumstances)>0){
  Dformula_string<-paste("SampleWeight", paste(circumstances, collapse=" + "), sep=" ~ ")

  # Here we create an aggregated sum of all the different combinations of the circumstances.
  circum_sum<-aggregate(as.formula(Dformula_string), data=opp_datause, sum)

  # We create a data frame to calculate d-index from data use and multiply sample weights by the
var2tab to create an aggregate sum of for "yes" access.
  D_datause<-opp_datause
  D_datause$SampleWeight<-D_datause$SampleWeight*D_datause$var2tab
  indic_sum<-aggregate(as.formula(Dformula_string), data=D_datause, sum)

  #We create a new data frame that combines all of our values. SW.x is all of our inputs. SW.y i
s only the values for "yes" access.
  total_sw<-sum(opp_datause$SampleWeight)
  tab_sum<-merge(circum_sum, indic_sum, by=circumstances, all.x=T)

  #Defining empty inputs as 0
  tab_sum$SampleWeight.y[is.na(tab_sum$SampleWeight.y)]<-0

  #Calculating overall_mean
  overall_mean<-sum(opp_datause$SampleWeight*opp_datause$var2tab)/sum(opp_datause$SampleWeight)

  #We rewrite the table excluding values that are 0.
  tab_sum<-tab_sum[tab_sum$SampleWeight.x>0, ]

  # D-Index calculation
  if(!is.na(overall_mean) && overall_mean>0) {

    tab_sum$SampleP<-tab_sum$SampleWeight.x/total_sw
    tab_sum$SampleM<-tab_sum$SampleWeight.y/tab_sum$SampleWeight.x-overall_mean
    tab_sum$abs_mean<-abs(tab_sum$SampleM)* tab_sum$SampleP
    D<-sum(tab_sum$abs_mean)/(2*overall_mean)

    Overall_D = D
  }
}
print(Overall_D)
```

```
## [1] 0.04589199
```