Assignment_1_Decision_Trees

Sushmitha Meduri

2024-04-12

```
# Loading necessary libraries
library(dplyr)
```

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

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

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

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ readr     2.1.4
## ✓ ggplot2   3.4.3     ✓ stringr   1.5.0
## ✓ lubridate 1.9.2     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
# Load cleaned data, "youth_data.Rdata" file from my local device

filepath <- "D:\\1_MASTERS\\3rd quarter-Spring 2024\\Machine learning 2\\Youth_Parse_da
ta_assignment_1\\youth_data.Rdata"

load(filepath)
```

Load cleaned data file, “youth_data.csv” from my local device

#youth_data <- read_csv(“D:/1_MASTERS/3rd quarter-Spring 2024/Machine learning 2/Youth_Parse_data_assignment_1/youth_data.csv”, show_col_types = FALSE)

#view the head of the file #head(youth_data)

```
# Handling Missing Values (Explore the data to decide the best approach)**
df <- df %>% na.omit()
```

```
# Quick data exploration
summary(df)
```

```
##     iralcfy          irmjfy         ircigfm      IRSMKLSS30N       iralcfm
##  Min.   :  1.0   Min.   :  1.0   Min.   : 1.0   Min.   : 1.00   Min.   : 1.00
##  1st Qu.:991.0   1st Qu.:991.0   1st Qu.:91.0   1st Qu.:91.00   1st Qu.:91.00
##  Median :991.0   Median :991.0   Median :91.0   Median :91.00   Median :91.00
##  Mean   :777.8   Mean   :870.8   Mean   :89.5   Mean   :90.33   Mean   :82.88
##  3rd Qu.:991.0   3rd Qu.:991.0   3rd Qu.:91.0   3rd Qu.:91.00   3rd Qu.:91.00
##  Max.   :993.0   Max.   :993.0   Max.   :93.0   Max.   :93.00   Max.   :93.00
##
##      irmjfm         ircigage      irsmklsstry       iralcage
##  Min.   : 1.00   Min.   :  5.0   Min.   :  5.0   Min.   :  1.0
##  1st Qu.:91.00   1st Qu.:991.0   1st Qu.:991.0   1st Qu.: 16.0
##  Median :91.00   Median :991.0   Median :991.0   Median :991.0
##  Mean   :85.23   Mean   :912.7   Mean   :960.8   Mean   :726.6
##  3rd Qu.:91.00   3rd Qu.:991.0   3rd Qu.:991.0   3rd Qu.:991.0
##  Max.   :93.00   Max.   :991.0   Max.   :991.0   Max.   :991.0
##
##     irmjage      mrjflag  alcflag  tobflag     alcydays        mrjydays
##  Min.   :  6.0   0:3595   0:3114   0:3806   Min.   :1.000   Min.   :1.000
##  1st Qu.:991.0   1: 674   1:1155   1: 463   1st Qu.:6.000   1st Qu.:6.000
##  Median :991.0                             Median :6.000   Median :6.000
##  Mean   :836.8                             Mean   :5.031   Mean   :5.521
##  3rd Qu.:991.0                             3rd Qu.:6.000   3rd Qu.:6.000
##  Max.   :991.0                             Max.   :6.000   Max.   :6.000
##
##     alcmdays        mrjmdays        cigmdays       smklsmdays    schfelt
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   1:3094
##  1st Qu.:5.000   1st Qu.:5.000   1st Qu.:6.000   1st Qu.:5.000   2:1175
##  Median :5.000   Median :5.000   Median :6.000   Median :5.000
##  Mean   :4.664   Mean   :4.812   Mean   :5.927   Mean   :4.976
##  3rd Qu.:5.000   3rd Qu.:5.000   3rd Qu.:6.000   3rd Qu.:5.000
##  Max.   :5.000   Max.   :5.000   Max.   :6.000   Max.   :5.000
##
##  tchgjob  avggrade stndscig stndsmj  stndalc  stnddnk  parchkhw parhlphw
##  1:3135   1: 175   1: 386   1: 981   1:1186   1: 361   1:3506   1:3402
##  2:1134   2:4094   2:3883   2:3288   2:3083   2:3908   2: 763   2: 867
##
##
##
##
##
##  PRCHORE2 PRLMTTV2 parlmtsn PRGDJOB2 PRPROUD2 argupar  YOFIGHT2 YOGRPFT2
##  1:3829   1:1833   1:2756   1:3661   1:3588   1:3388   1: 595   1: 409
##  2: 440   2:2436   2:1513   2: 608   2: 681   2: 881   2:3674   2:3860
##
##
##
##
##
##  YOHGUN2  YOSELL2  YOSTOLE2 YOATTAK2 PRPKCIG2 PRMJEVR2 prmjmo   PRALDLY2
##  1: 196   1:  57   1: 132   1: 183   1:4061   1:3526   1:3699   1:3926
##  2:4073   2:4212   2:4137   2:4086   2: 208   2: 743   2: 570   2: 343
##
##
##
##
```

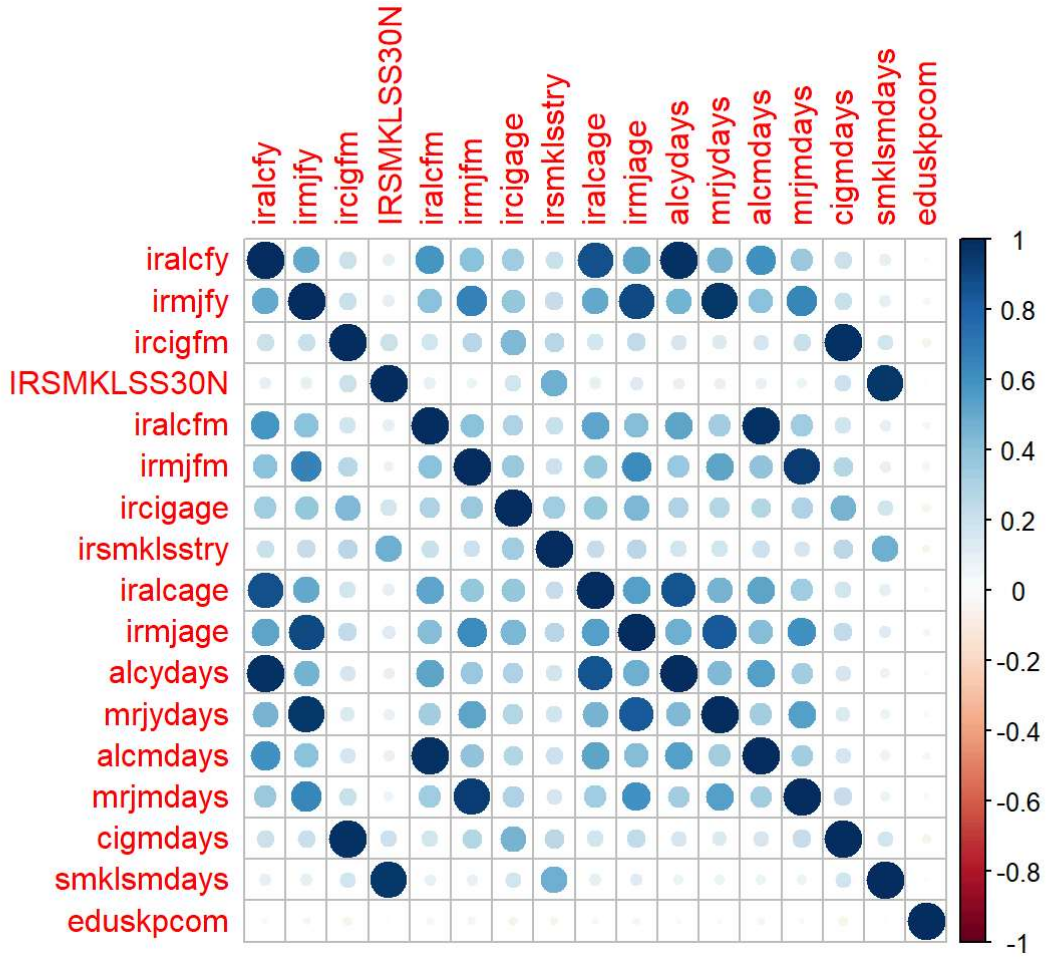
```
##
##  YFLPKCG2 YFLTMRJ2 yflmjmo  YFLADLY2 FRDPCIG2 FRDMEVR2 frdmjmon FRDADLY2
##  1:4037   1:3270   1:3319   1:3909   1:4026   1:3311   1:3397   1:3890
##  2: 232   2: 999   2: 950   2: 360   2: 243   2: 958   2: 872   2: 379
##
##
##
##
##
##  talkprob PRTALK3  PRBSOLV2 PREVIOL2 PRVDRGO2 GRPCNSL2 PREGPGM2 YTHACT2
##  1: 219   1:2645   1:1166   1: 414   1: 438   1: 156   1: 225   1: 601
##  2:4050   2:1624   2:3103   2:3855   2:3831   2:4113   2:4044   2:3668
##
##
##
##
##
##  DRPRVME3 ANYEDUC3 rlgattd  rlgimpt  rlgdcsn  rlgfrnd  irsex    NEWRACE2
##  1:3172   1:3071   1:1179   1:2734   1:2411   1:1050   1:2162   1:2456
##  2:1097   2:1198   2:3090   2:1535   2:1858   2:3219   2:2107   2: 425
##                                                                 3:  51
##                                                                 4:  14
##                                                                 5: 235
##                                                                 6: 258
##                                                                 7: 830
##  HEALTH2  eduschlgo   EDUSCHGRD2    eduskpcom     imother  ifather  income
##  1:1490   1 :3699   6      :680   Min.   : 0.00   1:3957   1:3285   1: 467
##  2:1754   2 : 536   5      :663   1st Qu.: 0.00   2: 299   2: 966   2: 899
##  3: 850   11:   2   4      :617   Median : 0.00   3:  13   3:  18   3: 575
##  4: 175   85:   2   7      :609   Mean   :19.93                     4:2328
##           94:  20   3      :577   3rd Qu.: 2.00
##           97:   5   99     :536   Max.   :99.00
##           98:   5   (Other):587
##  govtprog POVERTY3 PDEN10   COUTYP4
##  1: 781   1: 640   1:1762   1:1896
##  2:3488   2: 754   2:2212   2:1565
##           3:2875   3: 295   3: 808
##
##
##
##
```

```
#View structure of dataframe
str(df)
```

```
## 'data.frame':    4269 obs. of  79 variables:
##  $ iralcfy    : int  993 991 993 991 991 991 12 991 993 993 ...
##  $ irmjfy     : int  991 991 993 991 991 991 991 228 3 3 ...
##  $ ircigfm    : num  91 91 93 91 91 91 91 1 91 91 ...
##  $ IRSMKLSS30N: num  91 91 91 91 91 91 93 91 91 91 ...
##  $ iralcfm    : num  93 91 93 91 91 91 1 91 93 93 ...
##  $ irmjfm     : num  91 91 93 91 91 91 91 1 93 93 ...
##  $ ircigage   : int  991 991 13 991 991 991 991 13 991 991 ...
##  $ irsmklsstry: int  991 991 991 991 991 991 13 991 991 991 ...
##  $ iralcage   : int  12 991 13 991 991 991 13 991 13 15 ...
##  $ irmjage    : int  991 991 13 991 991 991 991 11 13 15 ...
##  $ mrjflag    : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 2 2 ...
##  $ alcflag    : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 2 2 ...
##  $ tobflag    : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 2 2 1 ...
##  $ alcydays   : int  6 6 6 6 6 6 2 6 6 6 ...
##  $ mrjydays   : int  6 6 6 6 6 6 6 4 1 1 ...
##  $ alcmdays   : int  5 5 5 5 5 5 1 5 5 5 ...
##  $ mrjmdays   : int  5 5 5 5 5 5 5 1 5 5 ...
##  $ cigmdays   : int  6 6 6 6 6 6 6 1 6 6 ...
##  $ smklsmdays : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ schfelt    : Factor w/ 2 levels "1","2": 1 1 1 1 1 2 1 2 1 2 ...
##  $ tchgjob    : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ avggrade   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ stndscig   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 1 2 ...
##  $ stndsmj    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 1 1 2 ...
##  $ stndalc    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 1 1 1 1 ...
##  $ stnddnk    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 1 2 ...
##  $ parchkhw   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ parhlphw   : Factor w/ 2 levels "1","2": 1 2 1 1 2 1 1 1 1 1 ...
##  $ PRCHORE2   : Factor w/ 2 levels "1","2": 2 1 1 1 1 1 1 1 1 1 ...
##  $ PRLMTTV2   : Factor w/ 2 levels "1","2": 2 2 2 1 1 1 2 1 1 1 ...
##  $ parlmtsn   : Factor w/ 2 levels "1","2": 2 2 1 2 2 1 1 1 1 1 ...
##  $ PRGDJOB2   : Factor w/ 2 levels "1","2": 2 1 1 1 1 1 1 1 1 1 ...
##  $ PRPROUD2   : Factor w/ 2 levels "1","2": 2 1 1 1 1 1 1 1 1 1 ...
##  $ argupar    : Factor w/ 2 levels "1","2": 1 1 1 1 1 2 1 1 1 1 ...
##  $ YOFIGHT2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 1 1 2 1 2 ...
##  $ YOGRPFT2   : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 2 2 ...
##  $ YOHGUN2    : Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 2 2 2 ...
##  $ YOSELL2    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ YOSTOLE2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 1 2 2 2 ...
##  $ YOATTAK2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ PRPKCIG2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ PRMJEVR2   : Factor w/ 2 levels "1","2": 1 2 1 1 1 1 1 1 2 2 ...
##  $ prmjmo     : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 2 ...
##  $ PRALDLY2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ YFLPKCG2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ YFLTMRJ2   : Factor w/ 2 levels "1","2": 1 2 1 1 1 1 1 2 2 2 ...
##  $ yflmjmo    : Factor w/ 2 levels "1","2": 1 2 1 1 1 1 1 1 2 2 ...
##  $ YFLADLY2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 2 1 ...
##  $ FRDPCIG2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 2 1 ...
##  $ FRDMEVR2   : Factor w/ 2 levels "1","2": 1 2 1 1 1 1 1 2 2 2 ...
##  $ frdmjmon   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 2 2 2 ...
##  $ FRDADLY2   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 2 1 ...
##  $ talkprob   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ PRTALK3    : Factor w/ 2 levels "1","2": 2 1 2 1 2 1 1 1 2 1 ...
```

```
##  $ PRBSOLV2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 2 1 2 ...
##  $ PREVIOL2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 2 1 2 ...
##  $ PRVDRGO2   : Factor w/ 2 levels "1","2": 2 1 2 2 2 1 2 2 2 2 ...
##  $ GRPCNSL2   : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ PREGPGM2   : Factor w/ 2 levels "1","2": 2 1 2 2 2 2 2 2 2 1 ...
##  $ YTHACT2    : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 1 2 2 ...
##  $ DRPRVME3   : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 2 1 1 ...
##  $ ANYEDUC3   : Factor w/ 2 levels "1","2": 1 1 1 1 2 1 1 2 1 1 ...
##  $ rlgattd    : Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 2 1 2 ...
##  $ rlgimpt    : Factor w/ 2 levels "1","2": 2 2 1 1 2 1 2 1 1 2 ...
##  $ rlgdcsn    : Factor w/ 2 levels "1","2": 2 2 1 1 2 1 2 1 1 2 ...
##  $ rlgfrnd    : Factor w/ 2 levels "1","2": 2 2 1 2 2 1 2 2 2 2 ...
##  $ irsex      : Factor w/ 2 levels "1","2": 1 1 1 2 1 1 1 1 2 1 ...
##  $ NEWRACE2   : Factor w/ 7 levels "1","2","3","4",..: 1 1 5 5 6 1 1 2 2 1 ...
##  $ HEALTH2    : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 3 2 2 1 2 1 1 3 2 1 ...
##  $ eduschlgo  : Factor w/ 7 levels "1","2","11","85",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ EDUSCHGRD2 : Ord.factor w/ 12 levels "1"<"2"<"3"<"4"<..: 5 5 4 7 8 3 6 5 6 6 ...
##  $ eduskpcom  : int  0 0 0 0 0 0 0 1 0 2 ...
##  $ imother    : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 2 1 1 2 1 ...
##  $ ifather    : Factor w/ 3 levels "1","2","3": 1 1 1 1 2 1 1 2 2 1 ...
##  $ income     : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 4 4 4 2 4 4 3 1 1 3 ...
##  $ govtprog   : Factor w/ 2 levels "1","2": 2 2 1 2 2 2 2 1 1 2 ...
##  $ POVERTY3   : Ord.factor w/ 3 levels "1"<"2"<"3": 3 3 3 1 3 3 3 1 1 3 ...
##  $ PDEN10     : Factor w/ 3 levels "1","2","3": 2 1 1 2 1 2 2 1 1 1 ...
##  $ COUTYP4    : Factor w/ 3 levels "1","2","3": 2 1 1 2 1 2 2 1 1 1 ...
##  - attr(*, "var.labels")= chr [1:79] "ALCOHOL FREQUENCY PAST YEAR - IMPUTATION REVIS
ED" "MARIJUANA FREQUENCY PAST YEAR - IMPUTATION REVISED" "CIG FREQUENCY PAST MONTH - IM
PUTATION REVISED" "SMOKELESS TOBACCO FREQUENCY PAST MONTH - IMPUTATION REVISED" ...
##  - attr(*, "na.action")= 'omit' Named int [1:1231] 5 7 12 17 19 20 28 35 36 43 ...
##   ..- attr(*, "names")= chr [1:1231] "42" "53" "67" "99" ...
```

```
# Correlation plot for all the numeric variables
df %>%
   select(where(is.numeric)) %>%
   cor() %>%
   corrplot::corrplot()
```



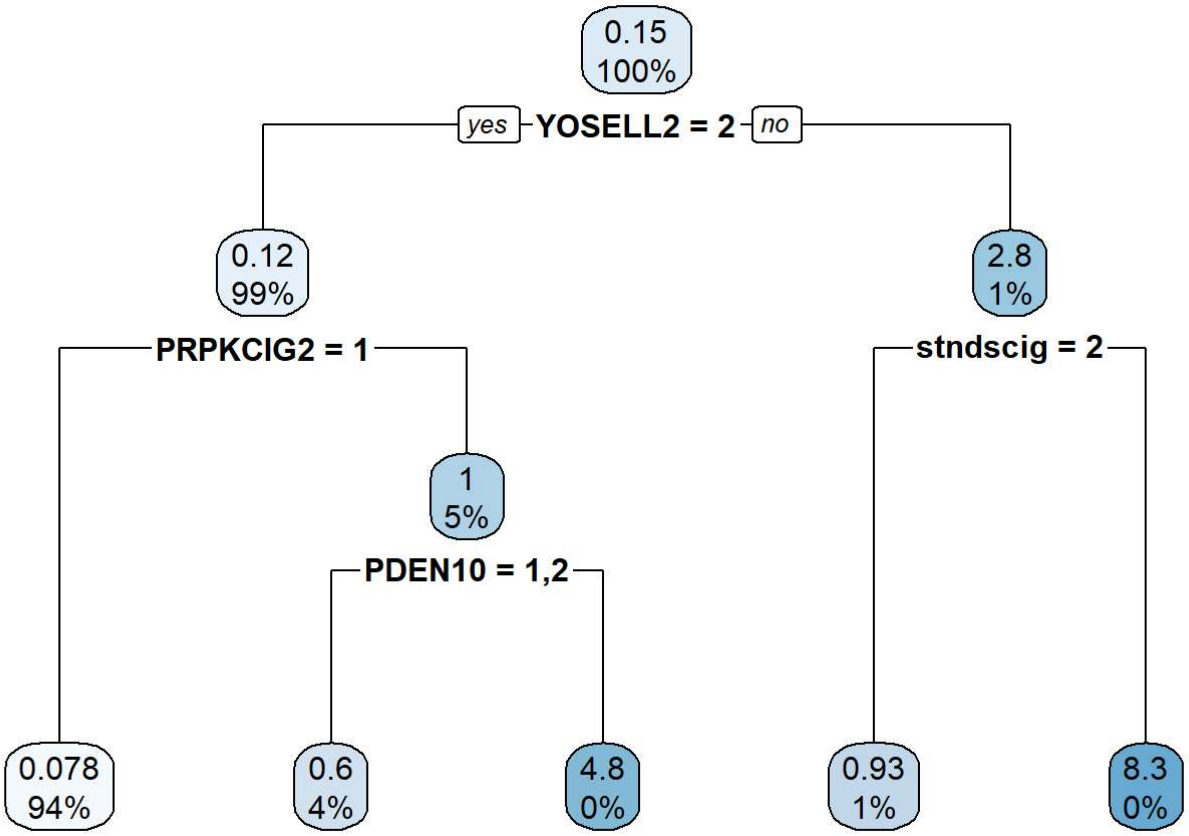

```
# Quantitative variable: Regression

# Recode IRCIGFM 91 and 93 to 0 as they are never used cigarettes for convince as the d
ata is just for 1 year, both of the code represet non-usage of cigarette.

df$ircigfm <- ifelse(df$ircigfm %in% c(91, 93), 0, df$ircigfm)
```

```
# Divide data into train and test sets
set.seed(123)  # for reproducibility
train_index <- sample(1:nrow(df), 0.8 * nrow(df))  # 80% for training
train_df <- df[train_index, ]
test_df <- df[-train_index, ]
```

```
# Choosing predictors from "df"
predictor_variables <- c(demographic_cols, youth_experience_cols)
```

```
set.seed(1)
# Define target and predictor variables for ircigfm with train data
y_reg_train <- train_df$ircigfm
X_reg_train <- train_df[, predictor_variables]


# Test data
y_reg_test <- test_df$ircigfm
X_reg_test <- test_df[, predictor_variables]
```

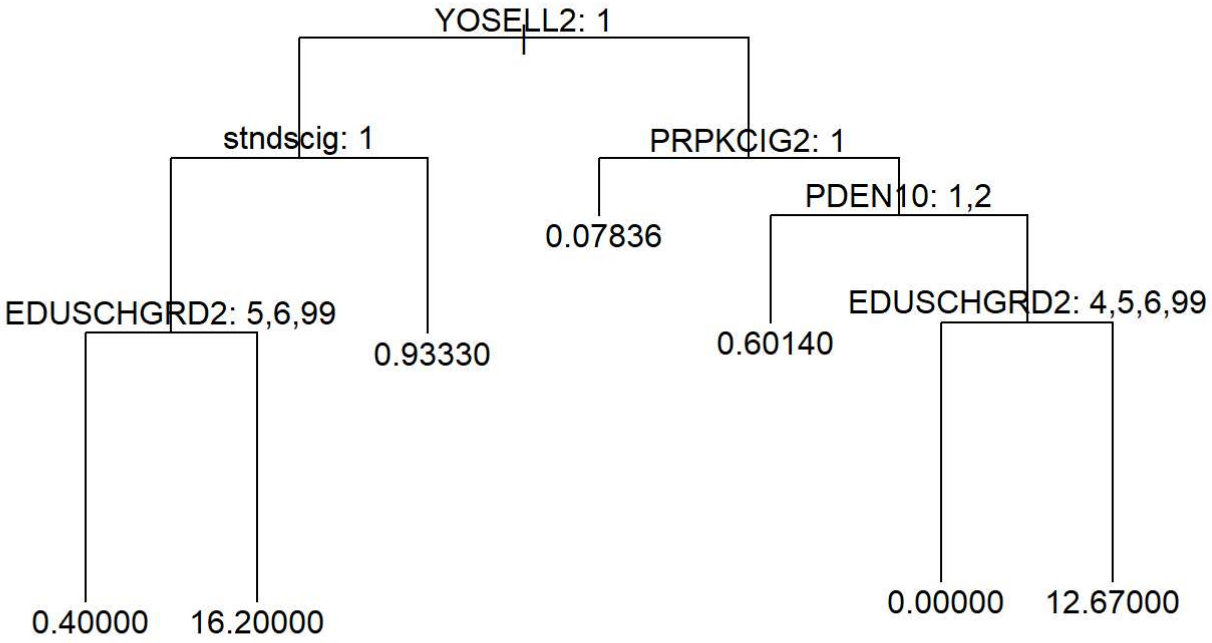
```
# Plotting tree using rpart.plot
tree_1 <- rpart(y_reg_train ~ ., data = X_reg_train)
rpart.plot(tree_1)
```




```
# Train decision tree model
tree_model_reg <- tree(y_reg_train ~ ., data = X_reg_train)
summary(tree_model_reg)
```

```
##
## Regression tree:
## tree(formula = y_reg_train ~ ., data = X_reg_train)
## Variables actually used in tree construction:
## [1] "YOSELL2"    "stndscig"   "EDUSCHGRD2" "PRPKCIG2"   "PDEN10"
## Number of terminal nodes:  7
## Residual mean deviance:  2.081 = 7094 / 3408
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -16.20000  -0.07836  -0.07836   0.00000  -0.07836  29.92000
```

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



As we see the tree “YOSELL2” which is Youth sold illegal drugs seems to be a potential variable and the tree start from his predictor.

```
tree_model_reg
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 3415 9386.0  0.15370
##    2) YOSELL2: 1 40 2023.0  2.77500
##      4) stndscig: 1 10 1276.0  8.30000
##        8) EDUSCHGRD2: 5,6,99 5    3.2  0.40000 *
##        9) EDUSCHGRD2: 4,7,8 5  648.8 16.20000 *
##      5) stndscig: 2 30  339.9  0.93330 *
##    3) YOSELL2: 2 3375 7085.0  0.12270
##      6) PRPKCIG2: 1 3216 4236.0  0.07836 *
##      7) PRPKCIG2: 2 159 2715.0  1.01900
##       14) PDEN10: 1,2 143  900.3  0.60140 *
##       15) PDEN10: 3 16 1567.0  4.75000
##         30) EDUSCHGRD2: 4,5,6,99 10    0.0  0.00000 *
##         31) EDUSCHGRD2: 7,9 6  965.3 12.67000 *
```

```
# Make predictions on the test set
pred_reg <- predict(tree_model_reg, newdata = X_reg_test)

# Calculate mean squared error (MSE)
mse_ircigfm <- mean((pred_reg - y_reg_test)^2)
print(paste("Mean Squared Error for Cigarette Frequency:", mse_ircigfm))
```

```
## [1] "Mean Squared Error for Cigarette Frequency: 3.44273581402767"
```

The test MSE is 3.4427 for Cigarette frequency using decision trees for regression.

```
# Find optimal tree size
tree_size <- cv.tree(tree_model_reg)$size[which.min(cv.tree(tree_model_reg)$dev)]
cat("Optimal Tree Size:", tree_size, "\n")
```

```
## Optimal Tree Size: 1
```

```
# Cross-validation for optimal tree size
cv_reg <- cv.tree(tree_model_reg)
optimal_tree_size <- cv_reg$size[which.min(cv_reg$dev)]
cat("Optimal Tree Size:", optimal_tree_size, "\n")
```

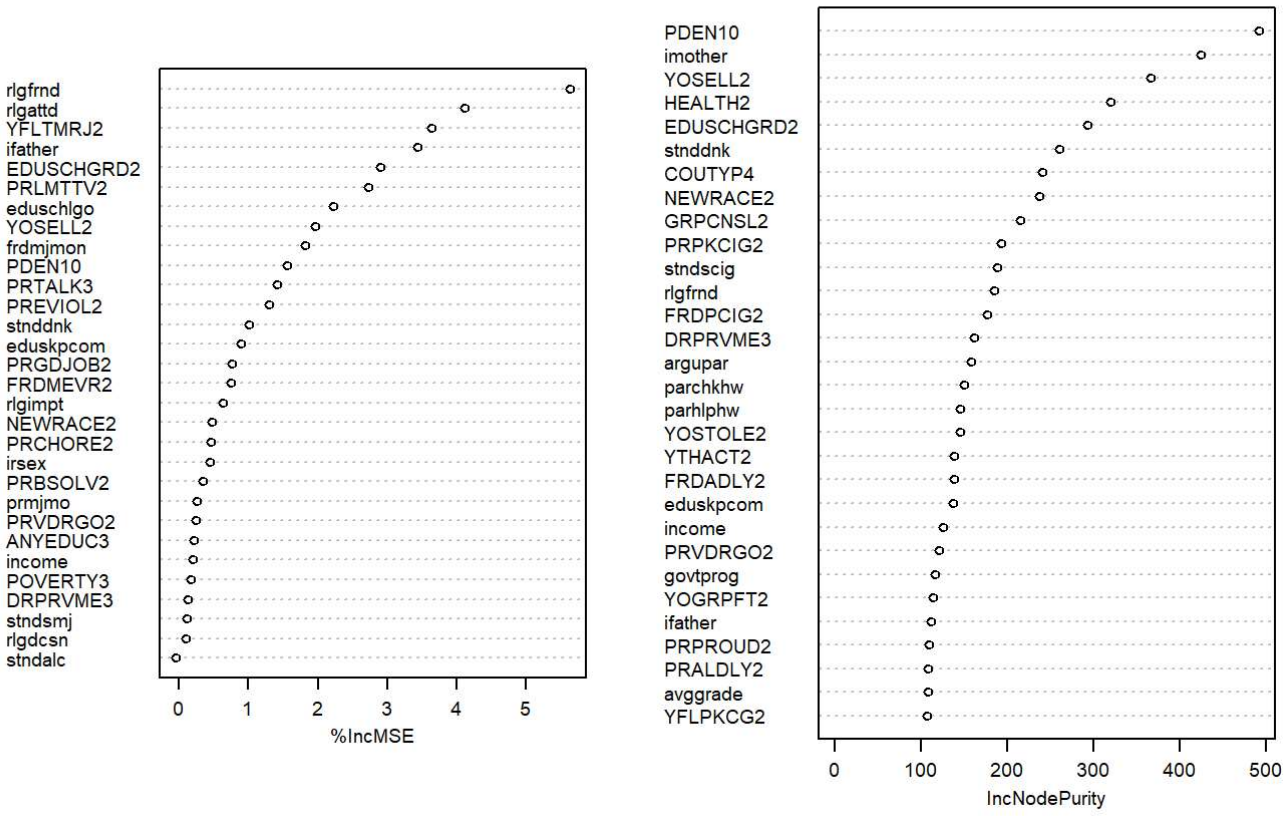
```
## Optimal Tree Size: 4
```

#random forest

```
# Train bagging model with all predictors and importance
rf_model_reg <- randomForest(y_reg_train ~ .,
                             data = X_reg_train,
                             importance = TRUE,
                             ntree = 500)
```

```
# Variable importance
varImpPlot(rf_model_reg, cex = 0.6)
```

rf_model_reg




```
# Make predictions on the test set
pred_reg_rf <- predict(rf_model_reg, newdata = X_reg_test)

# Calculate mean squared error (MSE) on the test set
mse_ircigfm_rf <- mean((pred_reg_rf - y_reg_test)^2)
print(paste("Mean Squared Error for Cigarette Frequency (Random Forest):", mse_ircigfm_
rf))
```

```
## [1] "Mean Squared Error for Cigarette Frequency (Random Forest): 2.675180197419998"
```

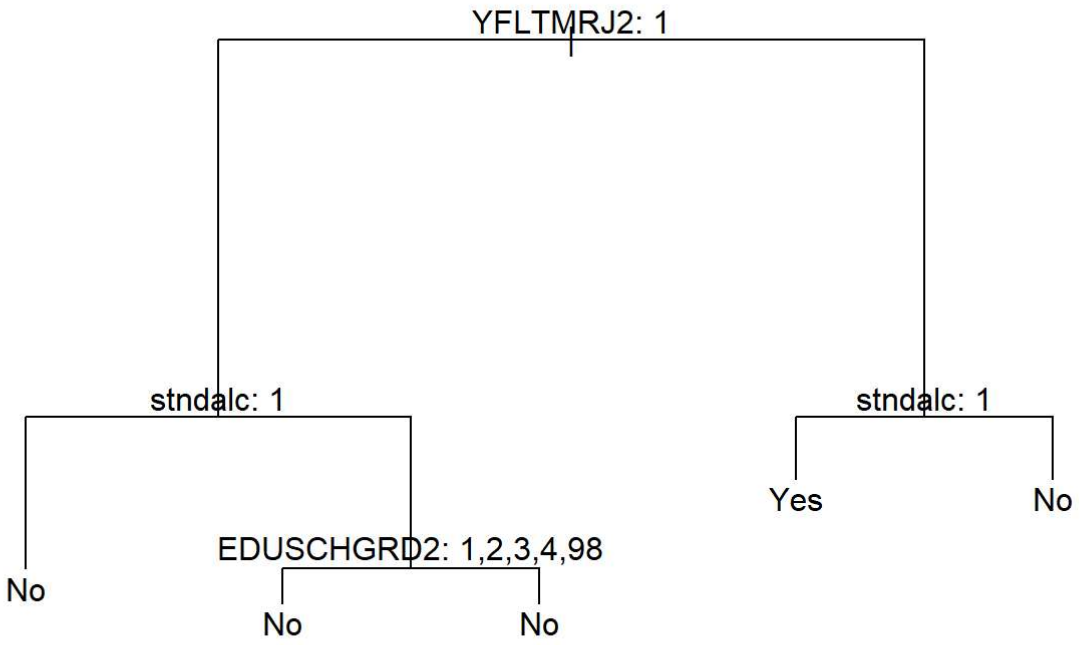
Using random forest regression, the test MSE value is decreased to 2.677. This means ensemble methods have improved our accuracy by 1%.

```
# Binary Classification: Modeling for alcflag (Any Alcohol Ever Used)
# Define target and predictor variables for alcflag
set.seed(123)

predictors <- c(demographic_cols, youth_experience_cols)

# Create a binary variable indicating alcohol use
df_binary <- df %>%
  mutate(alcohol_used = ifelse(alcflag == 1, "Yes", "No"))

# Split data into training and testing sets (80-20 split)
train_indices <- sample(1:nrow(df_binary), 0.8 * nrow(df_binary))
train_df <- df_binary[train_indices, ]
test_df <- df_binary[-train_indices, ]


# Convert alcohol_used to factor with two levels
train_df$alcohol_used <- factor(train_df$alcohol_used, levels = c("No", "Yes"))
```

```
# Train decision tree classification model
alc_tree_model <- tree(alcohol_used ~ ., data = train_df[, c(predictors, "alcohol_use
d")])
summary(alc_tree_model)
```

```
##
## Classification tree:
## tree(formula = alcohol_used ~ ., data = train_df[, c(predictors,
##     "alcohol_used")])
## Variables actually used in tree construction:
## [1] "YFLTMRJ2"   "stndalc"    "EDUSCHGRD2"
## Number of terminal nodes:  5
## Residual mean deviance:  0.9245 = 3152 / 3410
## Misclassification error rate: 0.2138 = 730 / 3415
```

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



Based on the tree YFLTMRJ2:1 which represents “How youth feel when they try Marijuana” is considered as a strong predictor when we predict “alcflag”.

```
alc_tree_model
```

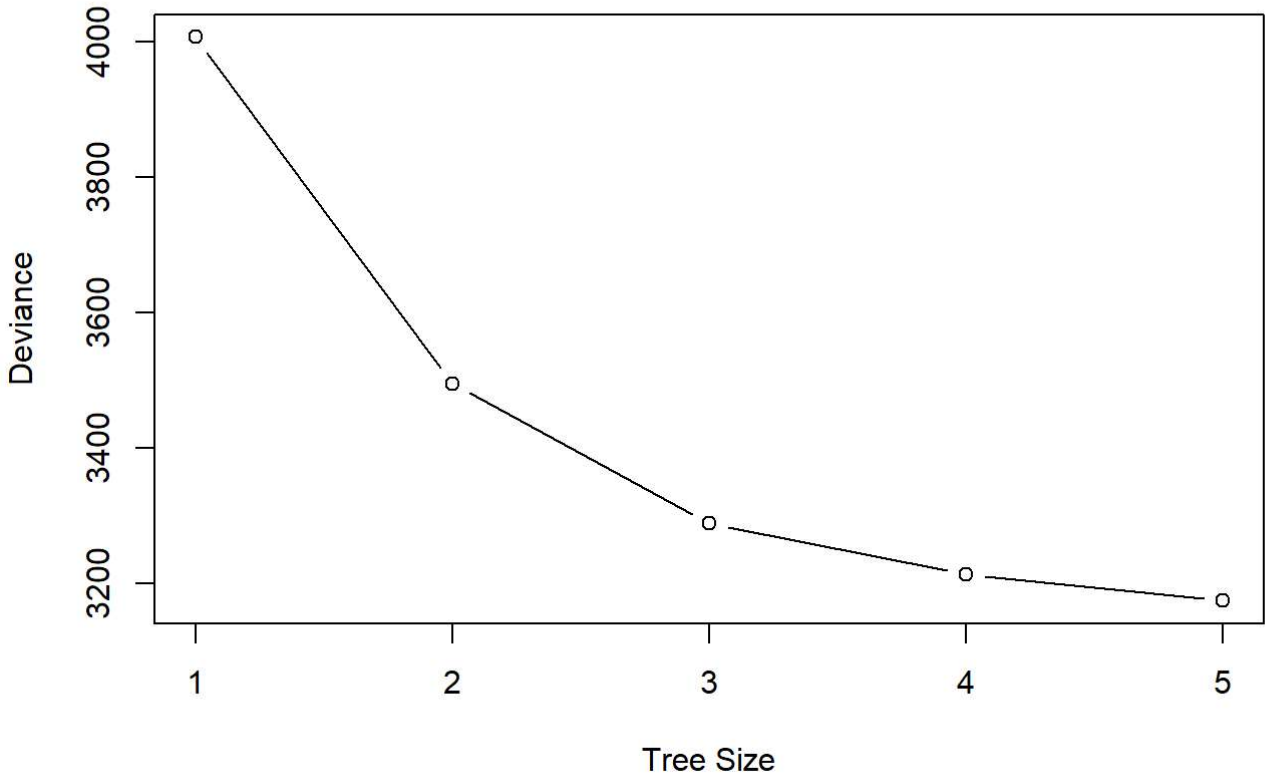
```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3415 4005.0 No ( 0.72679 0.27321 )
##    2) YFLTMRJ2: 1 2629 2435.0 No ( 0.82541 0.17459 )
##      4) stndalc: 1 569  762.0 No ( 0.60808 0.39192 ) *
##      5) stndalc: 2 2060 1467.0 No ( 0.88544 0.11456 )
##       10) EDUSCHGRD2: 1,2,3,4,98 985  473.7 No ( 0.93503 0.06497 ) *
##       11) EDUSCHGRD2: 5,6,7,8,9,99 1075  945.3 No ( 0.84000 0.16000 ) *
##    3) YFLTMRJ2: 2 786 1056.0 Yes ( 0.39695 0.60305 )
##      6) stndalc: 1 381  414.2 Yes ( 0.23360 0.76640 ) *
##      7) stndalc: 2 405  557.3 No ( 0.55062 0.44938 ) *
```

```
# Make predictions on the test set
alc_tree_pred <- predict(alc_tree_model,test_df, type = "class")


# Calculate accuracy
accuracy_alc_tree <- mean(alc_tree_pred == test_df$alcohol_used)
cat("Alcohol binary classification accuracy (Decision Tree):", accuracy_alc_tree)
```

```
## Alcohol binary classification accuracy (Decision Tree): 0.7997658
```

Using decision trees, the accuracy is 79.97%.

```
# Optimize decision tree using cross-validation
cv_bin <- cv.tree(alc_tree_model)
plot(cv_bin$size, cv_bin$dev, type = 'b',xlab = "Tree Size", ylab = "Deviance")
```



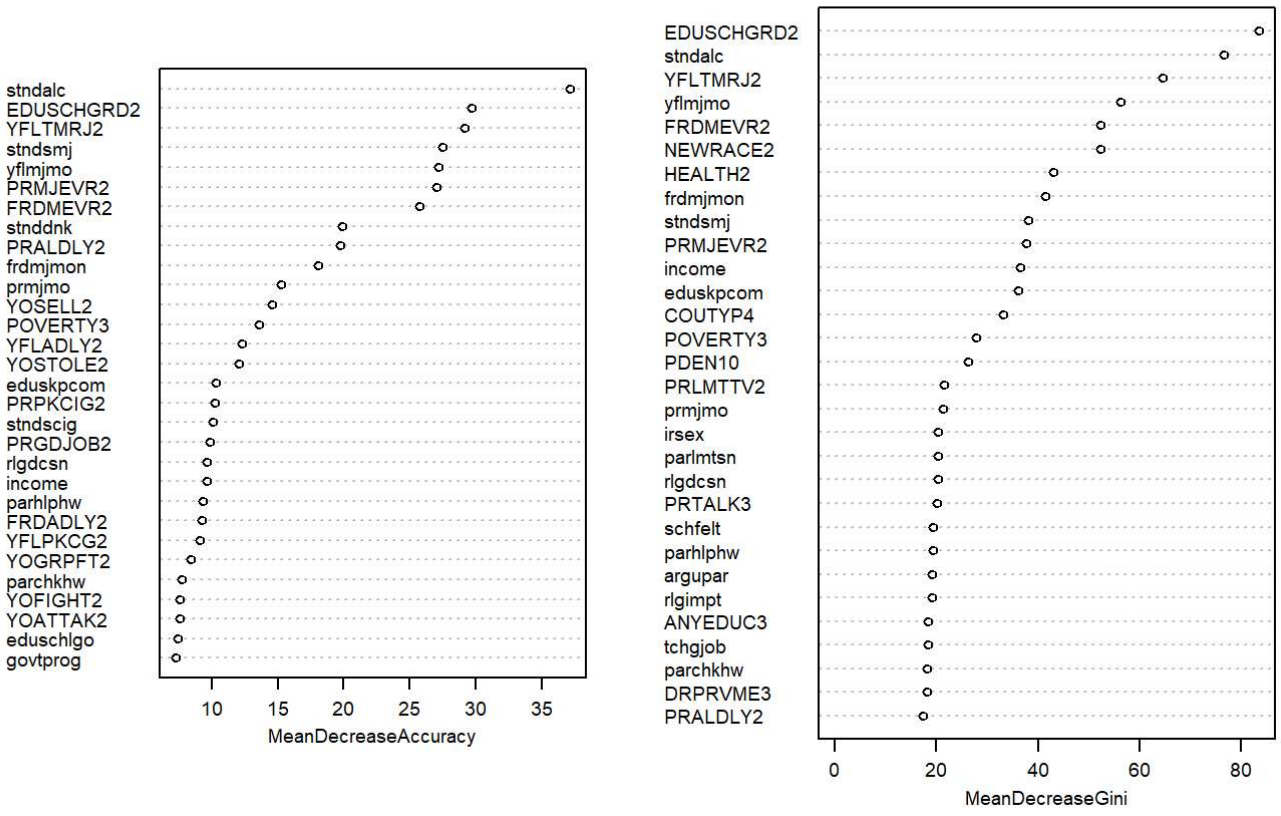

```
optimal_tree_size <- cv_bin$size[which.min(cv_bin$dev)]
cat("Optimal tree size:", optimal_tree_size, "\n")
```

```
## Optimal tree size: 5
```

Here the optimal tree size is 5

```
# Train Random Forest model

rf_model <- randomForest(alcohol_used ~ ., data = train_df[, c(predictors, "alcohol_use
d")], mtry = sqrt(ncol(train_df)), importance = TRUE)

varImpPlot(rf_model, cex=0.6)
```

rf_model




```
# Predict test set using Random Forest
rf_pred <- predict(rf_model, newdata = test_df)

# Calculate accuracy
accuracy_rf <- mean(rf_pred == test_df$alcohol_used)
cat("Alcohol binary classification accuracy (Random Forest):", accuracy_rf)
```

```
## Alcohol binary classification accuracy (Random Forest): 0.8161593
```

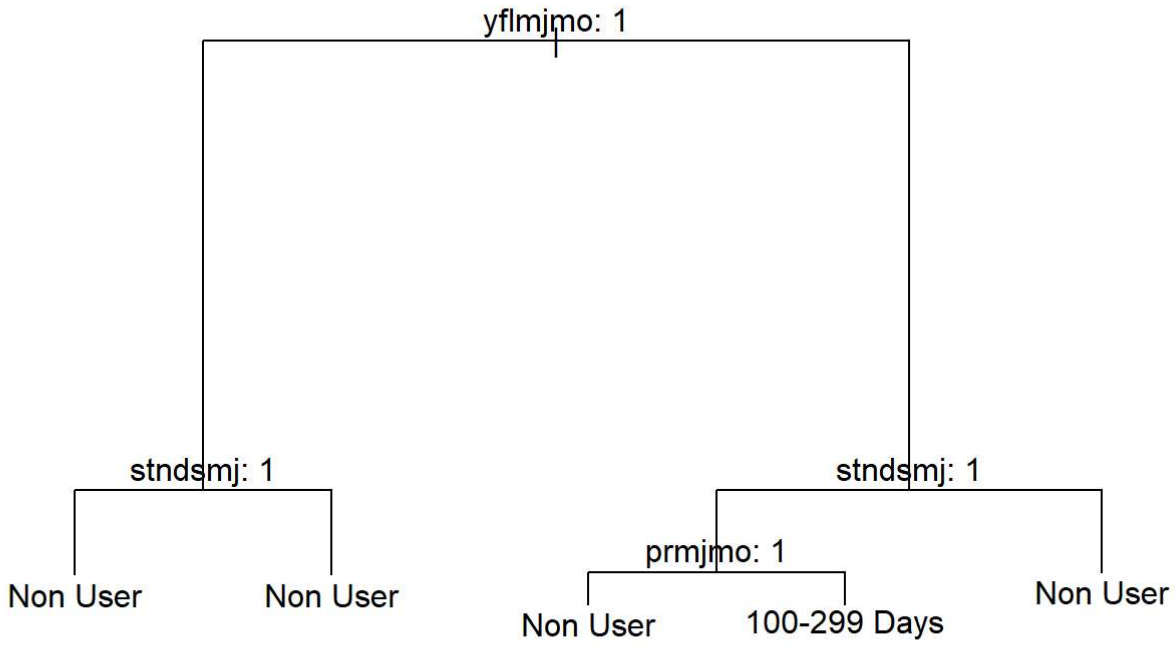
Using, ensemble methods, the accuracy has increased to 81.26%.

```
# Marijuana Use Frequency Multi-class Classification
marijuana_multiclass_train <- train_df %>%
  mutate(mrj_use_category = factor(mrjydays, levels = 1:6, labels = c("1-11 Days", "12-
49 Days", "50-99 Days", "100-299 Days", "300-365 Days", "Non User")))

marijuana_multiclass_test <- test_df %>%
  mutate(mrj_use_category = factor(mrjydays, levels = 1:6, labels = c("1-11 Days", "12-
49 Days", "50-99 Days", "100-299 Days", "300-365 Days", "Non User")))
```

```
# Train decision tree classification model
mrj_tree <- tree(mrj_use_category ~ ., data = marijuana_multiclass_train[, c(predictor
s, "mrj_use_category")])
summary(mrj_tree)
```

```
##
## Classification tree:
## tree(formula = mrj_use_category ~ ., data = marijuana_multiclass_train[,
##     c(predictors, "mrj_use_category")])
## Variables actually used in tree construction:
## [1] "yflmjmo" "stndsmj" "prmjmo"
## Number of terminal nodes:  5
## Residual mean deviance:  0.852 = 2905 / 3410
## Misclassification error rate: 0.1291 = 441 / 3415
```

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



It

seems like the predictor yflmjmo:1 HOW YTH FEELS: PEERS USING MARIJUANA MONTHLY strongly or partially disapprove as a strong predictor.

```
mrj_tree_pred <- predict(mrj_tree, marijuana_multiclass_test, type = "class")
mrj_tree_acc <- mean(mrj_tree_pred == marijuana_multiclass_test$mrj_use_category)
cat("Marijuana multi-class classification accuracy (Decision Tree):", mrj_tree_acc,
"\n")
```

```
## Marijuana multi-class classification accuracy (Decision Tree): 0.8641686
```

The accuracy for classifying the multi-class classification is 86.4%.

```
# Train random forest classification model using random forest
mrj_rf <- randomForest(
  mrj_use_category ~ .,
  data = marijuana_multiclass_train[, c(predictors, "mrj_use_category")],
  mtry = sqrt(ncol(marijuana_multiclass_train) - 1), # Number of variables randomly sam
pled as candidates at each split
  ntree = 400 # Number of trees to grow
)
mrj_rf
```

```
##
## Call:
##  randomForest(formula = mrj_use_category ~ ., data = marijuana_multiclass_train[,
c(predictors, "mrj_use_category")], mtry = sqrt(ncol(marijuana_multiclass_train) -
1), ntree = 400)
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 12.97%
## Confusion matrix:
##              1-11 Days 12-49 Days 50-99 Days 100-299 Days 300-365 Days Non User
## 1-11 Days            6          1          0            8            0      164
## 12-49 Days          10          3          0            7            2       76
## 50-99 Days           1          1          0            3            0       26
## 100-299 Days         6          2          0           14            0       81
## 300-365 Days         5          2          0            7            0       20
## Non User            12          3          0            5            1     2949
##              class.error
## 1-11 Days    0.966480447
## 12-49 Days   0.969387755
## 50-99 Days   1.000000000
## 100-299 Days 0.864077670
## 300-365 Days 1.000000000
## Non User     0.007070707
```

```
mrj_rf_pred <- predict(mrj_rf, marijuana_multiclass_test, type = "class")
mrj_rf_acc <- mean(mrj_rf_pred == marijuana_multiclass_test$mrj_use_category)
cat("Marijuana multi-class classification accuracy (Random Forest):", mrj_rf_acc, "\n")
```

```
## Marijuana multi-class classification accuracy (Random Forest): 0.8653396
```

Accuracy after using Random Forest is 86.3%.

Examining ensemble methods like random forest the accuracy remains the same at 86%.