Practical Homework 1 Decision Trees_Revised

Erdenetuya Namsrai

2025-04-23

1. Data preprocessing

```
# 1. Load Libraries
library(tree)
library(randomForest)
library(e1071)
library(caret)
library(tidyverse)
library(ggplot2)
library(reshape2)
library(RColorBrewer)
library(rpart)
library(rpart.plot)
library(ipred)

# 2. Load "Youth Dataset"
url <- "https://raw.githubusercontent.com/mendible/5322/main/Homework%201/youth_data.Rdata"
download.file(url, destfile = "youth_data.Rdata", mode = "wb")

loaded_dataset <- load("youth_data.Rdata")
youth_data_df <- get(loaded_dataset[1])

youth <- youth_data_df
head(youth, 3)
```

```
##   IRALCFY IRMJFY IRCIGFM IRSMKLSS30N IRALCFM IRMJFM IRCIGAGE IRSMKLSSTRY
## 1     991    991      91          91      91     91      991         991
## 2     991     60      91          91      91      2      991         991
## 3       1    991      91          91      93     91      991         991
##   IRALCAGE IRMJAGE MRJFLAG ALCFLAG TOBFLAG ALCYDAYS MRJYDAYS ALCMDAYS MRJMDAYS
## 1      991     991       0       0       0        6        6        5        5
## 2      991      14       1       0       0        6        3        5        1
## 3       11     991       0       1       0        1        6        5        5
##   CIGMDAYS SMKLSMDAYS SCHFELT TCHGJOB AVGGRADE STNDSCIG STNDSMJ STNDALC STNDDNK
## 1        6          5       1       1        2        2       2       2       2
## 2        6          5       2       1        2        1       1       1       1
## 3        6          5       1       1        2        2       2       2       2
##   PARCHKHW PARHLPHW PRCHORE2 PRLMTTV2 PARLMTSN PRGDJOB2 PRPROUD2 ARGUPAR
## 1        1        1        1        2        1        1        1       1
## 2        1        1        2        2        1        1        1       1
## 3        1        1        1        1        1        2        1       1
##   YOFIGHT2 YOGRPFT2 YOHGUN2 YOSELL2 YOSTOLE2 YOATTAK2 PRPKCIG2 PRMJEVR2 PRMJMO
## 1        2        2       2       2        2        2        1        1      1
## 2        2        2       2       2        2        2        1        2      2
## 3        1        1       2       2        2        2        1        1      1
##   PRALDLY2 YFLPKCG2 YFLTMRJ2 YFLMJMO YFLADLY2 FRDPCIG2 FRDMEVR2 FRDMJMON
## 1        1        1        1       1        1        1        1        1
## 2        1        1        1       2        1        2        2        2
## 3        1        1        1       1        1        1        1        1
##   FRDADLY2 TALKPROB PRTALK3 PRBSOLV2 PREVIOL2 PRVDRGO2 GRPCNSL2 PREGPGM2
## 1        1        2       1        1        1        2        2        2
## 2        2        2       1        2        2        2        2        2
## 3        1        2       1        2        2        2        2        2
##   YTHACT2 DRPRVME3 ANYEDUC3 RLGATTD RLGIMPT RLGDCSN RLGFRND IRSEX NEWRACE2
## 1       2        1        1       2       1       1       1     1        7
## 2       1        2        1       2       2       2       2     2        1
## 3       2        1        1       2       1       1       2     1        6
##   HEALTH2 EDUSCHLGO EDUSCHGRD2 EDUSKPCOM IMOTHER IFATHER INCOME GOVTPROG
## 1       3         1          3         0       1       1      2        2
## 2       4         1          6         0       1       1      2        2
## 3       1         1          2         1       1       1      4        2
##   POVERTY3 PDEN10 COUTYP4
## 1        1      2       2
## 2        1      2       2
## 3        3      1       1
```

```
# 3. Check missing values
missing_values <- colSums(is.na(youth))
print(missing_values)
```

```
##     IRALCFY      IRMJFY     IRCIGFM IRSMKLSS30N     IRALCFM      IRMJFM
##           0           0           0           0           0           0
##    IRCIGAGE IRSMKLSSTRY    IRALCAGE     IRMJAGE     MRJFLAG     ALCFLAG
##           0           0           0           0           0           0
##     TOBFLAG    ALCYDAYS    MRJYDAYS    ALCMDAYS    MRJMDAYS    CIGMDAYS
##           0           0           0           0           0           0
##  SMKLSMDAYS     SCHFELT     TCHGJOB    AVGGRADE    STNDSCIG     STNDSMJ
##           0           0          43         716         377         445
##     STNDALC     STNDDNK    PARCHKHW    PARHLPHW    PRCHORE2    PRLMTTV2
##         454         550          77          89          35          68
##    PARLMTSN    PRGDJOB2    PRPROUD2     ARGUPAR    YOFIGHT2    YOGRPFT2
##         259          63          77         180          67          64
##     YOHGUN2     YOSELL2    YOSTOLE2    YOATTAK2    PRPKCIG2    PRMJEVR2
##          46          25          40          41          96         101
##      PRMJMO    PRALDLY2    YFLPKCG2    YFLTMRJ2     YFLMJMO    YFLADLY2
##         100          91          90          89          92          93
##    FRDPCIG2    FRDMEVR2    FRDMJMON    FRDADLY2    TALKPROB     PRTALK3
##         146         152         152         147         336         199
##    PRBSOLV2    PREVIOL2    PRVDRGO2    GRPCNSL2    PREGPGM2     YTHACT2
##         286         142         102         120          98          68
##    DRPRVME3    ANYEDUC3     RLGATTD     RLGIMPT     RLGDCSN     RLGFRND
##         193         167         288         321         297         322
##       IRSEX    NEWRACE2     HEALTH2   EDUSCHLGO  EDUSCHGRD2   EDUSKPCOM
##           0           0          13           0           0           0
##     IMOTHER     IFATHER      INCOME    GOVTPROG    POVERTY3      PDEN10
##           0           0           0           0           0           0
##     COUTYP4
##           0
```

```
# 4. Define variable groups

# substance related variables
substance_variables <- c(
  "IRALCFY", "IRMJFY", "IRCIGFM", "IRSMKLSS30N", "IRALCFM", "IRMJFM",
  "IRCIGAGE", "IRSMKLSSTRY", "IRALCAGE", "IRMJAGE",
  "MRJFLAG", "ALCFLAG", "TOBFLAG",
  "ALCYDAYS", "MRJYDAYS", "ALCMDAYS", "MRJMDAYS", "CIGMDAYS", "SMKLSMDAYS"
)

# demographic related variables
demographic_variables <- c(
  "IRSEX", "NEWRACE2", "HEALTH2", "EDUSCHLGO", "EDUSCHGRD2", "EDUSKPCOM",
  "IMOTHER", "IFATHER", "INCOME", "GOVTPROG", "POVERTY3", "PDEN10", "COUTYP4"
)

# select variables from the original dataset
df_substance <- youth %>% select(all_of(substance_variables)) # select specific substance variab
les of interest
df_demog <- youth %>% select(all_of(demographic_cols))  # select specific demographic variables
of interest
df_youth_exper <- youth %>% select(SCHFELT:RLGFRND) # use all youth questions, start with schfel
t and go through rlgfrnd



# 5. Combine all into a single dataset
df_youth <- cbind(df_substance, df_demog, df_youth_exper) #combine into one data frame

dim(df_youth)
```

```
## [1] 10561    79
```

```
# 6. Remove missing values
cleaned_youth <- na.omit(df_youth)
youthdf <- cleaned_youth

dim(youthdf)
```

```
## [1] 8249   79
```

2.Binary classification “MRJFLAG”

2.1 Feature Selection: The Most Important Variables for Predicting “MRJFLAG”

```
# 1.Select youth experience variables and add MRJFLAG
df_corr <- youthdf %>%
  select(SCHFELT:RLGFRND, MRJFLAG)

# 2.Encode all categorical variables (if not numeric, factorize as integer)
df_corr[] <- lapply(df_corr, function(col) {
  if (is.factor(col) || is.character(col)) {
    as.integer(factor(col))
  } else {
    col
  }
})

# 3. Compute correlations
corr_matrix <- cor(df_corr, use = "complete.obs")
corr_mrjflag <- sort(corr_matrix[, "MRJFLAG"][-which(names(corr_matrix[, "MRJFLAG"]) == "MRJFLA
G")], decreasing = TRUE)

# 4.Print top and bottom correlations
cat("\nTop correlated features with MRJFLAG:\n")
```

```
##
## Top correlated features with MRJFLAG:
```

```
print(head(corr_mrjflag, 10))
```

```
##  FRDMJMON   YFLMJMO  FRDMEVR2  YFLTMRJ2    PRMJMO  PRMJEVR2  FRDADLY2  YFLADLY2
## 0.4571310 0.4317141 0.4225345 0.4042225 0.3619291 0.3600818 0.1928574 0.1791655
##  PRPROUD2  PRLMTTV2
## 0.1524149 0.1477842
```

```
cat("\nLeast correlated features with MRJFLAG:\n")
```

```
##
## Least correlated features with MRJFLAG:
```

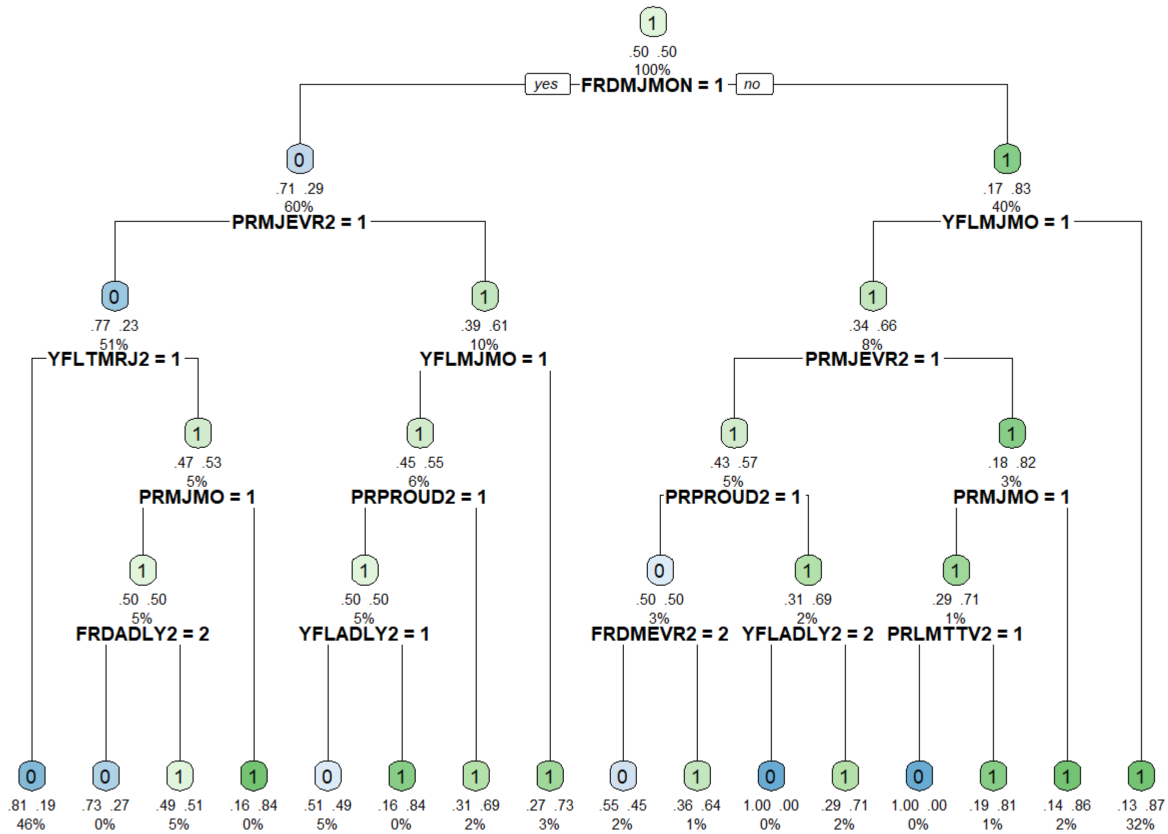
```
print(tail(corr_mrjflag, 10))
```

```
##    YOHGUN2   YOGRPFT2   AVGGRADE   YOFIGHT2   YOATTAK2    STNDDNK    YOSELL2
## -0.1098613 -0.1127250 -0.1136333 -0.1241215 -0.1298840 -0.2065708 -0.2112300
##   YOSTOLE2    STNDALC    STNDSMJ
## -0.2281383 -0.2848982 -0.4029783
```

2.2 Desicion Tree

Young people use marijuana? If so, what factors influence their use? “Target variable = MRJFLAG” (0 = no marijuana use, 1 = marijuana use)

```
# 1. Important variables and dataset
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

youth_binary <- youthdf %>%
  select(all_of(important_vars), MRJFLAG)

# Convert MRJFLAG to factor
youth_binary$MRJFLAG <- as.factor(youth_binary$MRJFLAG)

# 2. Split into training/testing sets
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 3. Handle class imbalance (class_weight = "balanced")
class_counts <- table(training_data$MRJFLAG)
total <- sum(class_counts)
weights <- total / (length(class_counts) * class_counts)
sample_weights <- weights[training_data$MRJFLAG]

# 4. Train a larger tree with max depth = 5 and minbucket = 5
tree_rpart <- rpart(
  MRJFLAG ~ .,
  data = training_data,
  method = "class",
  weights = sample_weights,
  control = rpart.control(
    maxdepth = 5,
    cp = 0.0001,
    minbucket = 5    # Minimum samples per leaf (like nodesize)
  )
)

# 5. Plot the tree
rpart.plot(tree_rpart, type = 2, extra = 104, under = TRUE, cex = 0.6)
```




```
# 6. Ensure test data is a data frame
testing_data <- as.data.frame(testing_data)

# 7. Predict on test data
test_pred <- predict(tree_rpart, testing_data, type = "class")

# 8. Ensure predictions and actuals have same factor levels
test_pred <- factor(test_pred, levels = levels(testing_data$MRJFLAG))
actual <- testing_data$MRJFLAG

# 9. Confusion matrix
confusion_matrix_dt <- table(Predicted = test_pred, Actual = actual)
print(confusion_matrix_dt)
```

```
##          Actual
## Predicted    0    1
##         0 1688  109
##         1  391  287
```

```
# 10. Evaluation metrics
accuracy_dt <- mean(test_pred == actual)
test_error_rate_dt <- 1 - accuracy_dt
precision_dt <- mean(test_pred == "1" & actual == "1") / mean(test_pred == "1")
recall_dt <- mean(test_pred == "1" & actual == "1") / mean(actual == "1")
f1_dt <- 2 * precision_dt * recall_dt / (precision_dt + recall_dt)

cat("\nEvaluation Metrics:\n")
```

```
##
## Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_dt, 4), "\n")
```

```
## Accuracy: 0.798
```

```
cat("Test Error Rate:", round(test_error_rate_dt, 4), "\n")
```

```
## Test Error Rate: 0.202
```

```
cat("Precision:", round(precision_dt, 4), "\n")
```

```
## Precision: 0.4233
```

```
cat("Recall:", round(recall_dt, 4), "\n")
```

```
## Recall: 0.7247
```

```
cat("F1-Score:", round(f1_dt, 4), "\n")
```

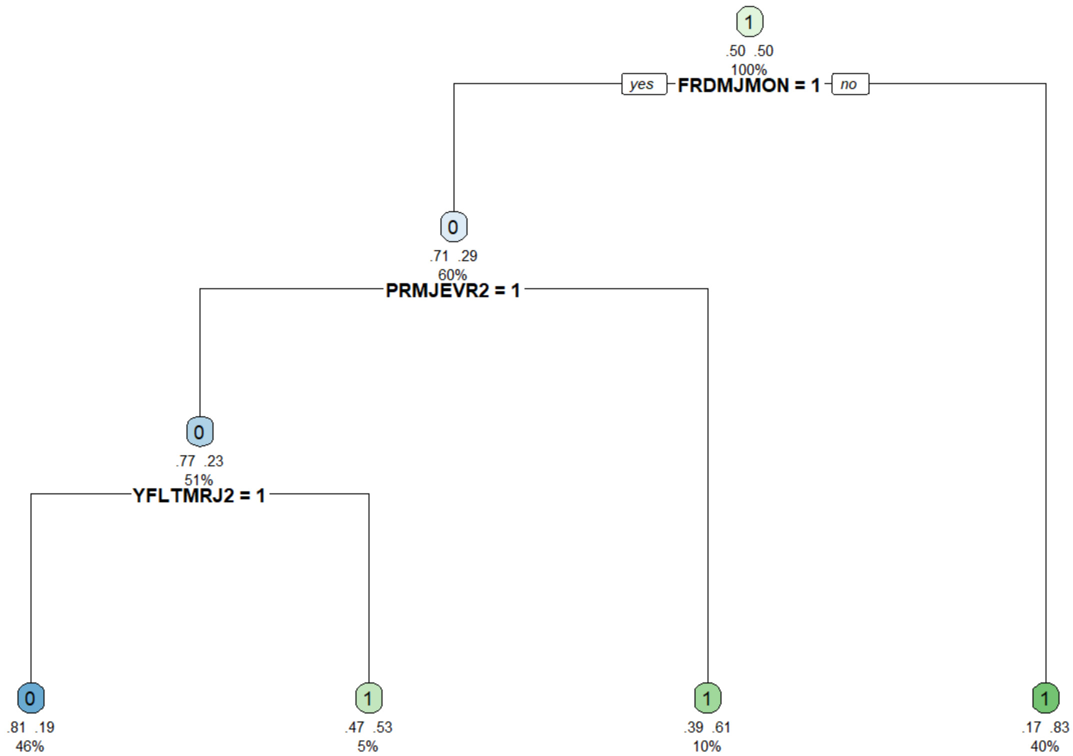
```
## F1-Score: 0.5345
```

2.3 Pruned Tree

```
# 1. Prepare dataset
youth_binary <- youthdf %>%
  select(all_of(important_vars), MRJFLAG)

youth_binary$MRJFLAG <- as.factor(youth_binary$MRJFLAG)

# 2. Train-test split
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 3. Handle class imbalance
class_counts <- table(training_data$MRJFLAG)
total <- sum(class_counts)
weights <- total / (length(class_counts) * class_counts)
sample_weights <- weights[training_data$MRJFLAG]

# 4. Train a larger tree with nodesize (minbucket) and maxdepth
tree_large <- rpart(
  MRJFLAG ~ .,
  data = training_data,
  method = "class",
  weights = sample_weights,
  control = rpart.control(
    maxdepth = 5,
    cp = 0.0001,
    minbucket = 5
  )
)

# 5. Cross-validation and pruning
printcp(tree_large)
```

```
##
## Classification tree:
## rpart(formula = MRJFLAG ~ ., data = training_data, weights = sample_weights,
##     method = "class", control = rpart.control(maxdepth = 5, cp = 1e-04,
##         minbucket = 5))
##
## Variables actually used in tree construction:
##  [1] FRDADLY2 FRDMEVR2 FRDMJMON PRLMTTV2 PRMJEVR2 PRMJMO   PRPROUD2 YFLADLY2
##  [9] YFLMJMO  YFLTMRJ2
##
## Root node error: 2887/5774 = 0.5
##
## n= 5774
##
##           CP nsplit rel error  xerror     xstd
## 1 0.51770259      0   1.00000 1.04820 0.013145
## 2 0.04086282      1   0.48230 0.48230 0.011259
## 3 0.00673145      2   0.44143 0.45592 0.011042
## 4 0.00117278      3   0.43470 0.44178 0.010919
## 5 0.00091875      8   0.42877 0.44409 0.010939
## 6 0.00082799     10   0.42693 0.44381 0.010937
## 7 0.00081650     12   0.42528 0.44357 0.010935
## 8 0.00010000     15   0.42283 0.44623 0.010958
```

```
optimal_cp <- tree_large$cptable[which.min(tree_large$cptable[, "xerror"]), "CP"]

# Prune the tree
tree_pruned <- prune(tree_large, cp = optimal_cp)

# 6. Plot the pruned tree
rpart.plot(tree_pruned, type = 2, extra = 104, under = TRUE, cex = 0.6)
```




```
# 7. Evaluation on test set
test_pred <- predict(tree_pruned, newdata = testing_data, type = "class")
test_pred <- factor(test_pred, levels = levels(testing_data$MRJFLAG))
actual <- testing_data$MRJFLAG

confusion_matrix_pru <- table(Predicted = test_pred, Actual = actual)
print(confusion_matrix_pru)
```

```
##          Actual
## Predicted    0    1
##         0 1540   76
##         1  539  320
```

```
accuracy_pru <- mean(test_pred == actual)
test_error_rate_pru <- 1 - accuracy_pru
precision_pru <- mean(test_pred == "1" & actual == "1") / mean(test_pred == "1")
recall_pru <- mean(test_pred == "1" & actual == "1") / mean(actual == "1")
f1_score_pru <- 2 * precision_pru * recall_pru / (precision_pru + recall_pru)

# 8. Print results
cat("\nPruned Tree Evaluation Metrics:\n")
```

```
##
## Pruned Tree Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_pru, 4), "\n")
```

```
## Accuracy: 0.7515
```

```
cat("Test Error Rate:", round(test_error_rate_pru, 4), "\n")
```

```
## Test Error Rate: 0.2485
```

```
cat("Precision:", round(precision_pru, 4), "\n")
```

```
## Precision: 0.3725
```

```
cat("Recall:", round(recall_pru, 4), "\n")
```

```
## Recall: 0.8081
```

```
cat("F1-Score:", round(f1_score_pru, 4), "\n")
```

```
## F1-Score: 0.51
```

2.4 Bagging

```
# 1. Important variables
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

# 2. Prepare dataset with selected variables
youth_binary <- youthdf %>%
  select(all_of(important_vars), MRJFLAG)

# Convert target to factor
youth_binary$MRJFLAG <- as.factor(youth_binary$MRJFLAG)

# 3. Split into train and test sets
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 4. Train the bagging model (mtry = all predictors)
set.seed(42)
bagging_model <- randomForest(
  MRJFLAG ~ .,
  data = training_data,
  mtry = length(important_vars),
  ntree = 1000,
  nodesize = 5,
  maxnodes = 30,
  importance = TRUE
)

# 5. Predict on test data
bagging_pred <- predict(bagging_model, newdata = testing_data, type = "class")
actual <- testing_data$MRJFLAG

# 6. Confusion matrix
confusion_matrix_bag <- table(Predicted = bagging_pred, Actual = actual)
print(confusion_matrix_bag)
```

```
##          Actual
## Predicted    0    1
##         0 1950  220
##         1  129  176
```

```
# 7. Evaluation metrics
accuracy_bag <- mean(bagging_pred == actual)
test_error_rate_bag <- 1 - accuracy_bag
precision_bag <- mean(bagging_pred == "1" & actual == "1") / mean(bagging_pred == "1")
recall_bag <- mean(bagging_pred == "1" & actual == "1") / mean(actual == "1")
f1_bag <- 2 * precision_bag * recall_bag / (precision_bag + recall_bag)

# 8. Print results
cat("\nBagging Model Evaluation Metrics:\n")
```

```
##
## Bagging Model Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_bag, 4), "\n")
```

```
## Accuracy: 0.859
```

```
cat("Test Error Rate:", round(test_error_rate_bag, 4), "\n")
```

```
## Test Error Rate: 0.141
```

```
cat("Precision:", round(precision_bag, 4), "\n")
```

```
## Precision: 0.577
```

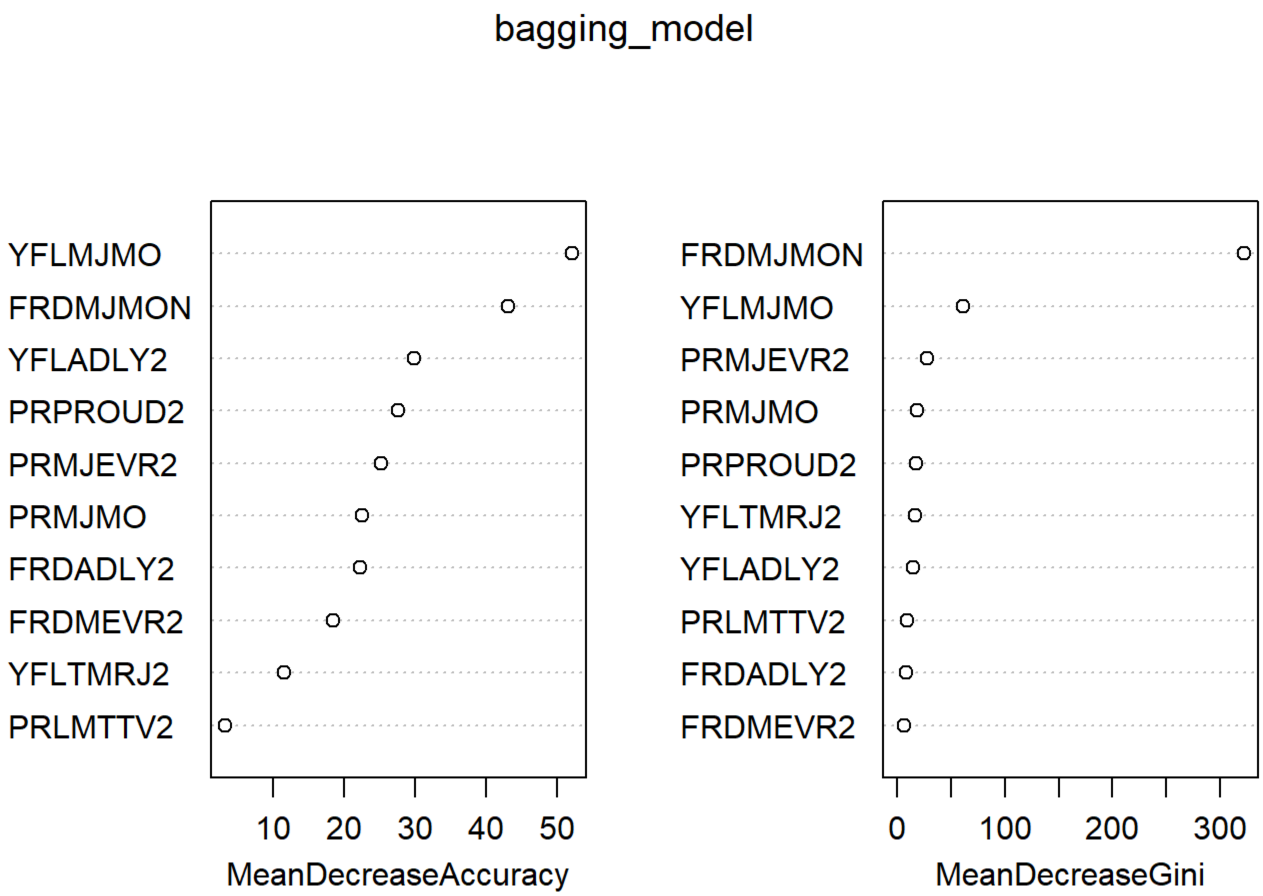
```
cat("Recall:", round(recall_bag, 4), "\n")
```

```
## Recall: 0.4444
```

```
cat("F1-Score:", round(f1_bag, 4), "\n")
```

```
## F1-Score: 0.5021
```

```
# 9. Feature importance plot
varImpPlot(bagging_model)
```



2.5 Random Forest

```
# 1. Important variables
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

# 2. Prepare dataset
youth_binary <- youthdf %>%
  select(all_of(important_vars), MRJFLAG)

# Convert MRJFLAG to factor
youth_binary$MRJFLAG <- as.factor(youth_binary$MRJFLAG)

# 3. Train/test split
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 4. Train Random Forest with nodesize and maxnodes
set.seed(42)
rf_model <- randomForest(
  MRJFLAG ~ .,
  data = training_data,
  mtry = 3,
  ntree = 5000,
  nodesize = 5,
  maxnodes = 30,
  importance = TRUE
)

# 5. Predict on test data
rf_pred <- predict(rf_model, newdata = testing_data, type = "class")
actual <- testing_data$MRJFLAG

# 6. Confusion matrix
confusion_matrix_rf <- table(Predicted = rf_pred, Actual = actual)
print(confusion_matrix_rf)
```

```
##          Actual
## Predicted    0    1
##         0 1977  238
##         1  102  158
```

```
# 7. Evaluation metrics
accuracy_rf <- mean(rf_pred == actual)
test_error_rate_rf <- 1 - accuracy_rf
precision_rf <- mean(rf_pred == "1" & actual == "1") / mean(rf_pred == "1")
recall_rf <- mean(rf_pred == "1" & actual == "1") / mean(actual == "1")
f1_rf <- 2 * precision_rf * recall_rf / (precision_rf + recall_rf)

# 8. Print results
cat("\nRandom Forest Evaluation Metrics:\n")
```

```
##
## Random Forest Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_rf, 4), "\n")
```

```
## Accuracy: 0.8626
```

```
cat("Test Error Rate:", round(test_error_rate_rf, 4), "\n")
```

```
## Test Error Rate: 0.1374
```

```
cat("Precision:", round(precision_rf, 4), "\n")
```

```
## Precision: 0.6077
```

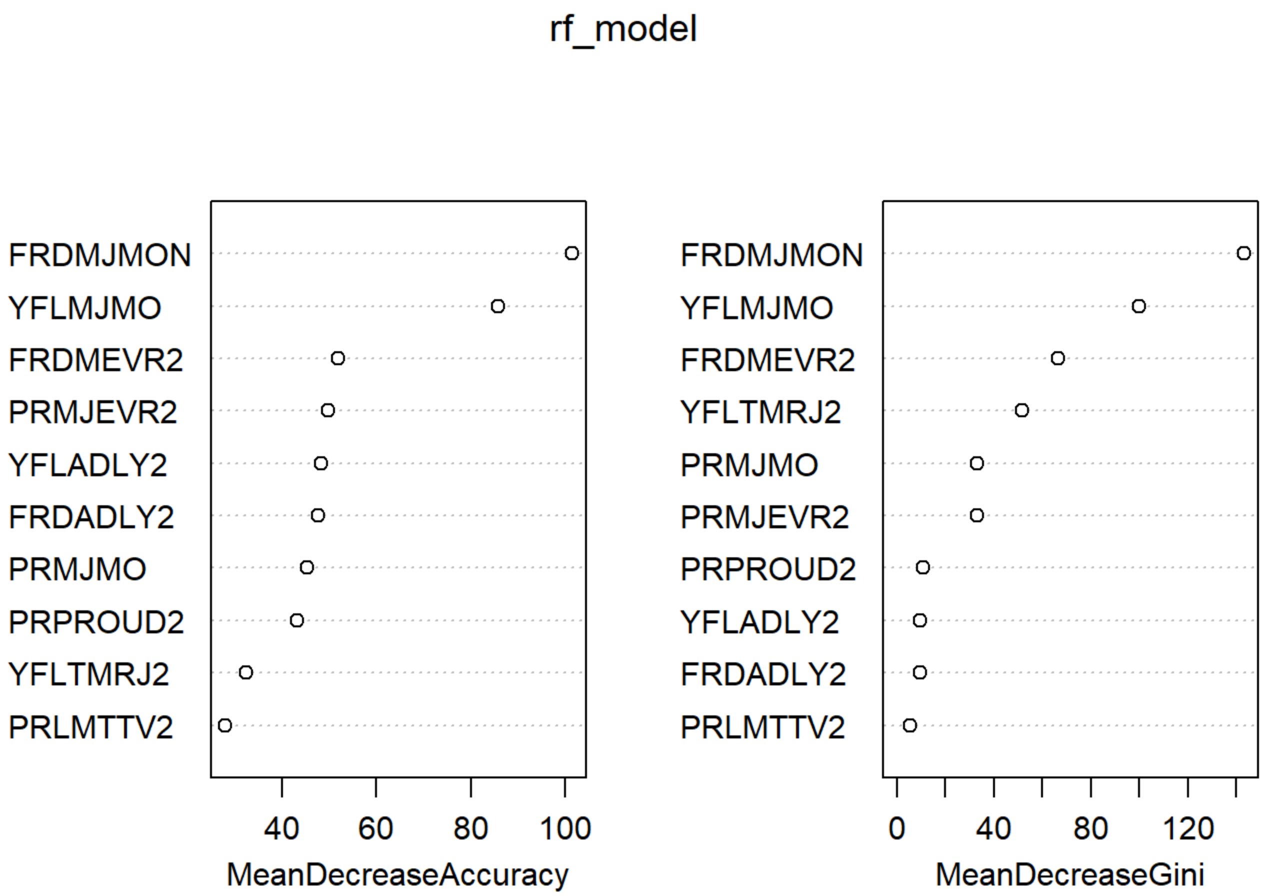
```
cat("Recall:", round(recall_rf, 4), "\n")
```

```
## Recall: 0.399
```

```
cat("F1-Score:", round(f1_rf, 4), "\n")
```

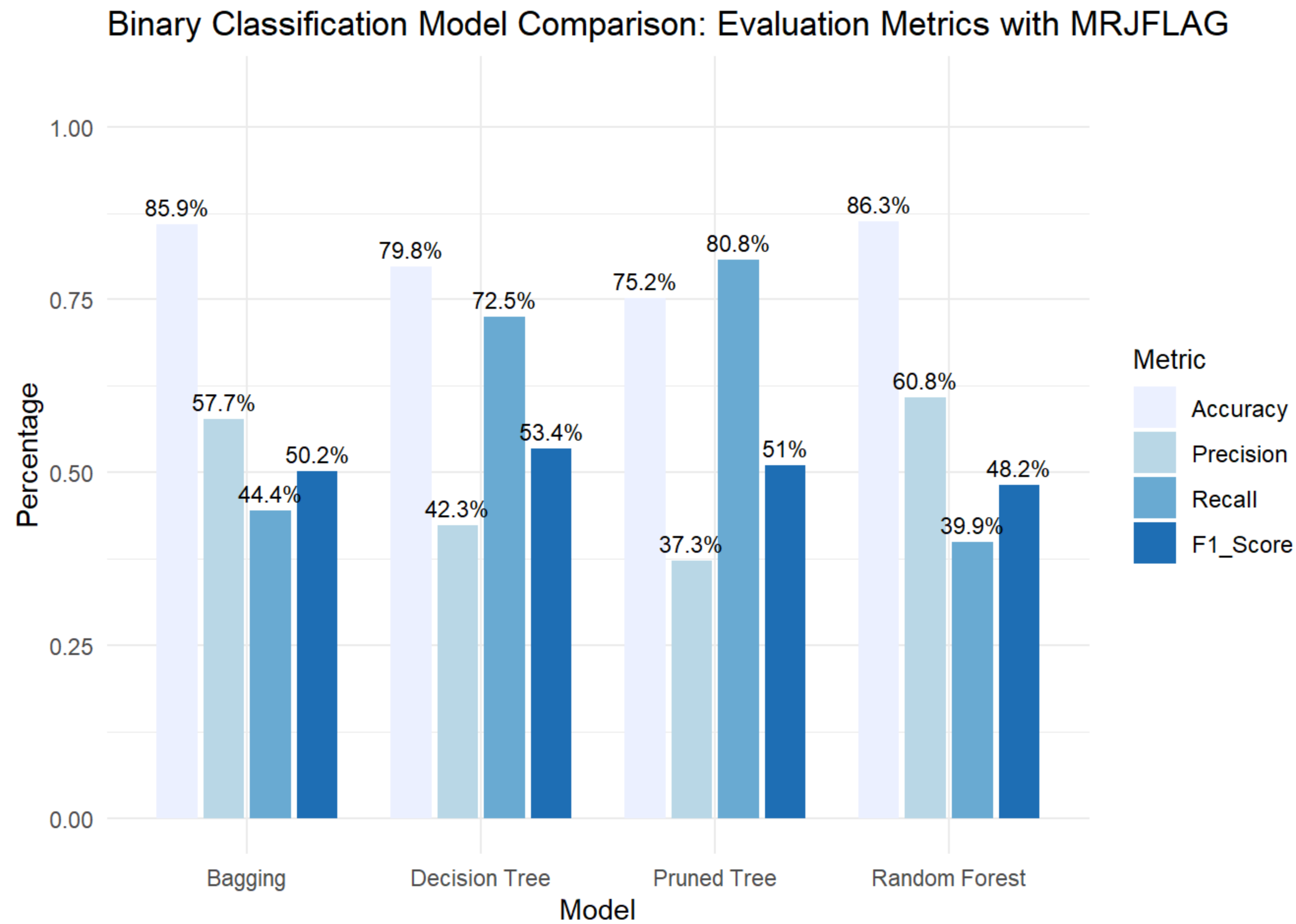
```
## F1-Score: 0.4817
```

```
# 9. Variable importance plot
varImpPlot(rf_model)
```



2.6 Comparison of Binary Classification Methods Decision Tree, Pruned Tree, Bagging and random Forest with “MRJFLAG”

```
# 1. Define model names and evaluation metrics
model_names <- c("Decision Tree", "Pruned Tree", "Bagging", "Random Forest")

accuracy_values_marijuana <- c(accuracy_dt, accuracy_pru, accuracy_bag, accuracy_rf)
precision_values_marijuana <- c(precision_dt, precision_pru, precision_bag, precision_rf)
recall_values_marijuana <- c(recall_dt, recall_pru, recall_bag, recall_rf)
f1_score_values_marijuana <- c(f1_dt, f1_score_pru, f1_bag, f1_rf)

# 2. Create full metrics data frame for reshaping
metrics_df <- data.frame(
  Model = model_names,
  Accuracy = accuracy_values_marijuana,
  Precision = precision_values_marijuana,
  Recall = recall_values_marijuana,
  F1_Score = f1_score_values_marijuana
)

# 3. Reshape to long format
metrics_long <- melt(metrics_df, id.vars = "Model",
                     variable.name = "Metric", value.name = "Score")

# 4. Plot all metrics (no Accuracy vs Error Rate plot)
ggplot(metrics_long, aes(x = Model, y = Score, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_text(aes(label = paste0(round(Score * 100, 1), "%")),
            position = position_dodge(width = 0.8),
            vjust = -0.5, size = 3) +
  scale_fill_brewer(palette = "Blues") +
  labs(title = "Binary Classification Model Comparison: Evaluation Metrics with MRJFLAG",
       x = "Model", y = "Percentage") +
  ylim(0, 1.05) +
  theme_minimal() +
  theme(legend.title = element_text(size = 10),
        legend.text = element_text(size = 9),
        axis.text.x = element_text(angle = 0, hjust = 0.5))
```



3.Binary classification “TOBFLAG”

3.1 Feature Selection: The Most Important Variables for Predicting “TOBFLAG”

```
# 1.Select youth experience variables and add TOBFLAG
df_corr <- youthdf %>%
  select(SCHFELT:RLGFRND, TOBFLAG)

# 2.Encode all categorical variables (if not numeric, factorize as integer)
df_corr[] <- lapply(df_corr, function(col) {
  if (is.factor(col) || is.character(col)) {
    as.integer(factor(col))
  } else {
    col
  }
})

# 3. Compute correlations
corr_matrix <- cor(df_corr, use = "complete.obs")
corr_mrjflag <- sort(corr_matrix[, "TOBFLAG"][-which(names(corr_matrix[, "TOBFLAG"]) == "TOBFLA
G")], decreasing = TRUE)

# 4.Print top and bottom correlations
cat("\nTop correlated features with TOBFLAG:\n")
```

```
##
## Top correlated features with TOBFLAG:
```

```
print(head(corr_mrjflag, 10))
```

```
##   YFLMJMO  FRDMJMON  YFLTMRJ2  FRDMEVR2    PRMJMO  PRMJEVR2  FRDADLY2  YFLADLY2
## 0.2620369 0.2598866 0.2460419 0.2402495 0.2147514 0.2117051 0.1764211 0.1566074
##  PRALDLY2  PRPROUD2
## 0.1160421 0.1125972
```

```
cat("\nLeast correlated features with TOBFLAG:\n")
```

```
##
## Least correlated features with TOBFLAG:
```

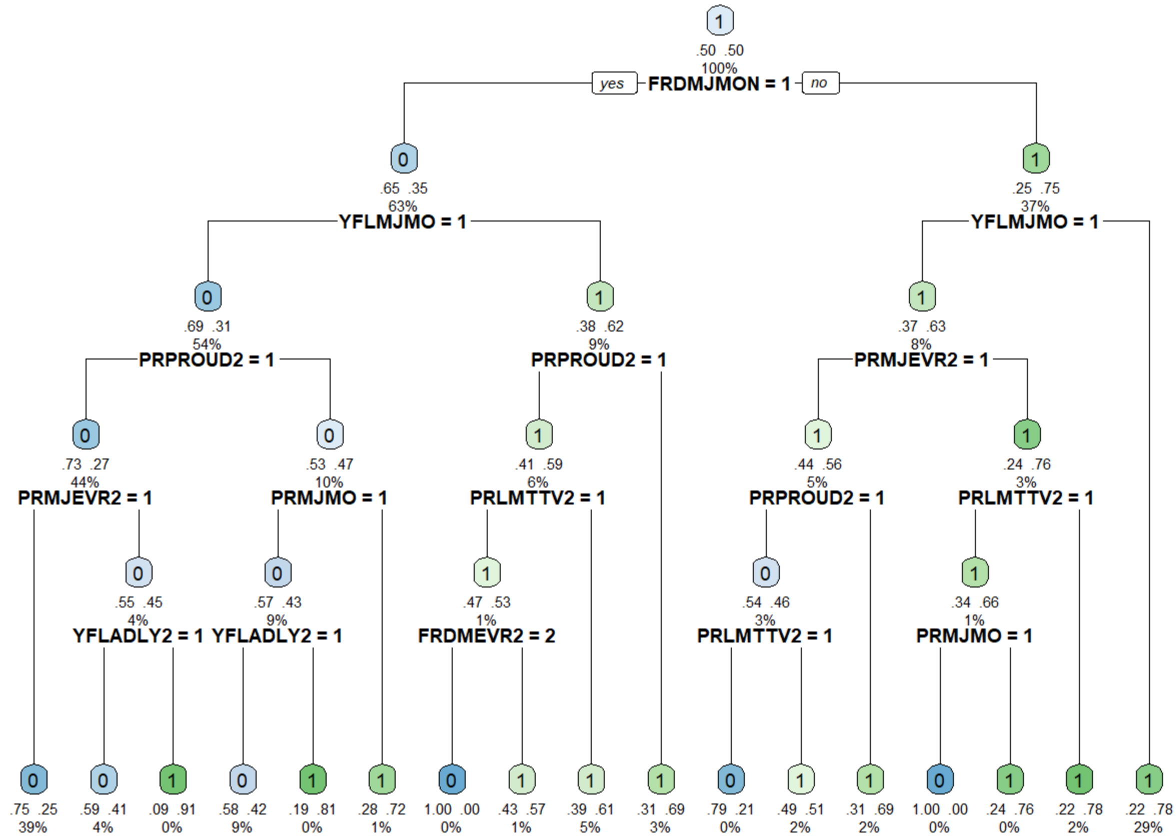
```
print(tail(corr_mrjflag, 10))
```

```
##   AVGGRADE   YOATTAK2   YOFIGHT2   YOGRPFT2    YOHGUN2    STNDDNK    STNDALC
## -0.1115246 -0.1237177 -0.1261375 -0.1269909 -0.1288916 -0.1475899 -0.1905028
##   YOSTOLE2    YOSELL2    STNDSMJ
## -0.2004238 -0.2120911 -0.2217796
```

3.2 Desicion Tree

Young people use tobacco? If so, what factors influence their use? “Target variable = TOBFLAG” (0 = no tobacco use, 1 = tobacco use)

```
# 1. Important variables and dataset
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

youth_binary <- youthdf %>%
  select(all_of(important_vars), TOBFLAG)

# Convert TOBFLAG to factor
youth_binary$TOBFLAG <- as.factor(youth_binary$TOBFLAG)

# 2. Split into training/testing sets
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 3. Handle class imbalance (class_weight = "balanced")
class_counts <- table(training_data$TOBFLAG)
total <- sum(class_counts)
weights <- total / (length(class_counts) * class_counts)
sample_weights <- weights[training_data$TOBFLAG]

# 4. Train a larger tree with max depth = 5 and minbucket = 5
tree_rpart <- rpart(
  TOBFLAG ~ .,
  data = training_data,
  method = "class",
  weights = sample_weights,
  control = rpart.control(
    maxdepth = 5,
    cp = 0.0001,
    minbucket = 5    # Minimum samples per leaf (like nodesize)
  )
)

# 5. Plot the tree
rpart.plot(tree_rpart, type = 2, extra = 104, under = TRUE, cex = 0.6)
```




```
# 6. Ensure test data is a data frame
testing_data <- as.data.frame(testing_data)

# 7. Predict on test data
test_pred <- predict(tree_rpart, testing_data, type = "class")

# 8. Ensure predictions and actuals have same factor levels
test_pred <- factor(test_pred, levels = levels(testing_data$TOBFLAG))
actual <- testing_data$TOBFLAG

# 9. Confusion matrix
confusion_matrix_dt <- table(Predicted = test_pred, Actual = actual)
print(confusion_matrix_dt)
```

```
##          Actual
## Predicted    0    1
##         0 1656   91
##         1  553  175
```

```
# 10. Evaluation metrics
accuracy_dt2 <- mean(test_pred == actual)
test_error_rate_dt2 <- 1 - accuracy_dt2
precision_dt2 <- mean(test_pred == "1" & actual == "1") / mean(test_pred == "1")
recall_dt2 <- mean(test_pred == "1" & actual == "1") / mean(actual == "1")
f1_dt2 <- 2 * precision_dt2 * recall_dt2 / (precision_dt2 + recall_dt2)

cat("\nEvaluation Metrics:\n")
```

```
##
## Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_dt2, 4), "\n")
```

```
## Accuracy: 0.7398
```

```
cat("Test Error Rate:", round(test_error_rate_dt2, 4), "\n")
```

```
## Test Error Rate: 0.2602
```

```
cat("Precision:", round(precision_dt2, 4), "\n")
```

```
## Precision: 0.2404
```

```
cat("Recall:", round(recall_dt2, 4), "\n")
```

```
## Recall: 0.6579
```

```
cat("F1-Score:", round(f1_dt2, 4), "\n")
```

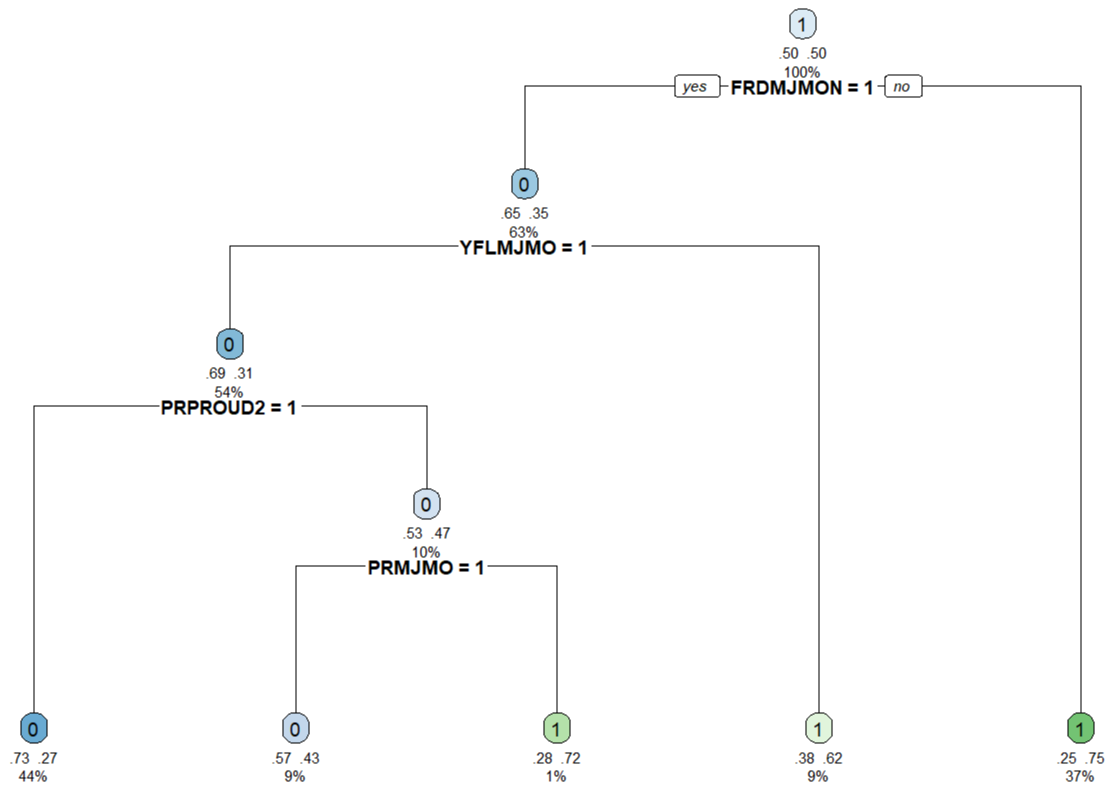
```
## F1-Score: 0.3521
```

3.3 Pruned Tree

```
youth_binary <- youthdf %>%
  select(all_of(important_vars), TOBFLAG)

youth_binary$TOBFLAG <- as.factor(youth_binary$TOBFLAG)

# 2. Train-test split
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 3. Handle class imbalance
class_counts <- table(training_data$TOBFLAG)
total <- sum(class_counts)
weights <- total / (length(class_counts) * class_counts)
sample_weights <- weights[training_data$TOBFLAG]

# 4. Train a larger tree
tree_large <- rpart(
  TOBFLAG ~ .,
  data = training_data,
  method = "class",
  weights = sample_weights,
  control = rpart.control(maxdepth = 5, cp = 0.0001)
)

# 5. Cross-validation and pruning
printcp(tree_large)  # view cp table
```

```
##
## Classification tree:
## rpart(formula = TOBFLAG ~ ., data = training_data, weights = sample_weights,
##     method = "class", control = rpart.control(maxdepth = 5, cp = 1e-04))
##
## Variables actually used in tree construction:
## [1] FRDADLY2 FRDMEVR2 FRDMJMON PRLMTTV2 PRMJEVR2 PRMJMO   PRPROUD2 YFLADLY2
## [9] YFLMJMO
##
## Root node error: 2887/5774 = 0.5
##
## n= 5774
##
##           CP nsplit rel error  xerror     xstd
## 1  0.36890319      0   1.00000 1.06038 0.013136
## 2  0.04232099      1   0.63110 0.65583 0.012356
## 3  0.00577843      2   0.58878 0.58878 0.011996
## 4  0.00424995      4   0.57722 0.58629 0.011981
## 5  0.00231592      5   0.57297 0.58682 0.011984
## 6  0.00149369      7   0.56834 0.58887 0.011997
## 7  0.00076380     10   0.56386 0.59336 0.012023
## 8  0.00053089     12   0.56233 0.60458 0.012088
## 9  0.00050920     13   0.56180 0.60458 0.012088
## 10 0.00010000     16   0.56027 0.61461 0.012144
```

```
optimal_cp <- tree_large$cptable[which.min(tree_large$cptable[, "xerror"]), "CP"]

# Prune the tree
tree_pruned <- prune(tree_large, cp = optimal_cp)

# 6. Plot the pruned tree
rpart.plot(tree_pruned, type = 2, extra = 104, under = TRUE, cex = 0.6)
```




```
# 7. Evaluation on test set
test_pred <- predict(tree_pruned, newdata = testing_data, type = "class")
test_pred <- factor(test_pred, levels = levels(testing_data$TOBFLAG))
actual <- testing_data$TOBFLAG

confusion_matrix_pru <- table(Predicted = test_pred, Actual = actual)
print(confusion_matrix_pru)
```

```
##          Actual
## Predicted    0    1
##         0 1640   88
##         1  569  178
```

```
accuracy_pru2 <- mean(test_pred == actual)
test_error_rate_pru2 <- 1 - accuracy_pru2
precision_pru2 <- mean(test_pred == "1" & actual == "1") / mean(test_pred == "1")
recall_pru2 <- mean(test_pred == "1" & actual == "1") / mean(actual == "1")
f1_score_pru2 <- 2 * precision_pru2 * recall_pru2 / (precision_pru2 + recall_pru2)

# 8. Print results
cat("\nPruned Tree Evaluation Metrics:\n")
```

```
##
## Pruned Tree Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_pru2, 4), "\n")
```

```
## Accuracy: 0.7345
```

```
cat("Test Error Rate:", round(test_error_rate_pru2, 4), "\n")
```

```
## Test Error Rate: 0.2655
```

```
cat("Precision:", round(precision_pru2, 4), "\n")
```

```
## Precision: 0.2383
```

```
cat("Recall:", round(recall_pru2, 4), "\n")
```

```
## Recall: 0.6692
```

```
cat("F1-Score:", round(f1_score_pru2, 4), "\n")
```

```
## F1-Score: 0.3514
```

3.4 Bagging

```
# 1. Important variables
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

# 2. Prepare dataset with selected variables
youth_binary <- youthdf %>%
  select(all_of(important_vars), TOBFLAG)

# Convert target to factor
youth_binary$TOBFLAG <- as.factor(youth_binary$TOBFLAG)

# 3. Split into train and test sets
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 4. Train the bagging model (mtry = all predictors)
set.seed(42)
bagging_model <- randomForest(
  TOBFLAG ~ .,
  data = training_data,
  mtry = length(important_vars),
  ntree = 1000,
  nodesize = 5,
  maxnodes = 30,
  importance = TRUE,
  strata = training_data$TOBFLAG,
  sampsize = rep(min(table(training_data$TOBFLAG)), 2)
)

# 5. Predict on test data
bagging_pred <- predict(bagging_model, newdata = testing_data, type = "class")
actual <- testing_data$TOBFLAG

# 6. Confusion matrix
confusion_matrix_bag <- table(Predicted = bagging_pred, Actual = actual)
print(confusion_matrix_bag)
```

```
##          Actual
## Predicted    0    1
##         0 1676   88
##         1  533  178
```

```
# 7. Evaluation metrics with safety checks
tp <- sum(bagging_pred == "1" & actual == "1")
fp <- sum(bagging_pred == "1" & actual == "0")
fn <- sum(bagging_pred == "0" & actual == "1")

precision_bag2 <- ifelse((tp + fp) == 0, NA, tp / (tp + fp))
recall_bag2 <- ifelse((tp + fn) == 0, 0, tp / (tp + fn))
f1_bag2 <- ifelse(is.na(precision_bag2) || (precision_bag2 + recall_bag2) == 0, NA,
                 2 * precision_bag2 * recall_bag2 / (precision_bag2 + recall_bag2))

accuracy_bag2 <- mean(bagging_pred == actual)
error_rate_bag <- 1 - accuracy_bag2

# 8. Print results
cat("\nBagging Model Evaluation Metrics (TOBFLAG):\n")
```

```
##
## Bagging Model Evaluation Metrics (TOBFLAG):
```

```
cat("Accuracy:", round(accuracy_bag2, 4), "\n")
```

```
## Accuracy: 0.7491
```

```
cat("Test Error Rate:", round(error_rate_bag, 4), "\n")
```

```
## Test Error Rate: 0.2509
```

```
cat("Precision:", ifelse(is.na(precision_bag2), "NaN", round(precision_bag2, 4)), "\n")
```

```
## Precision: 0.2504
```

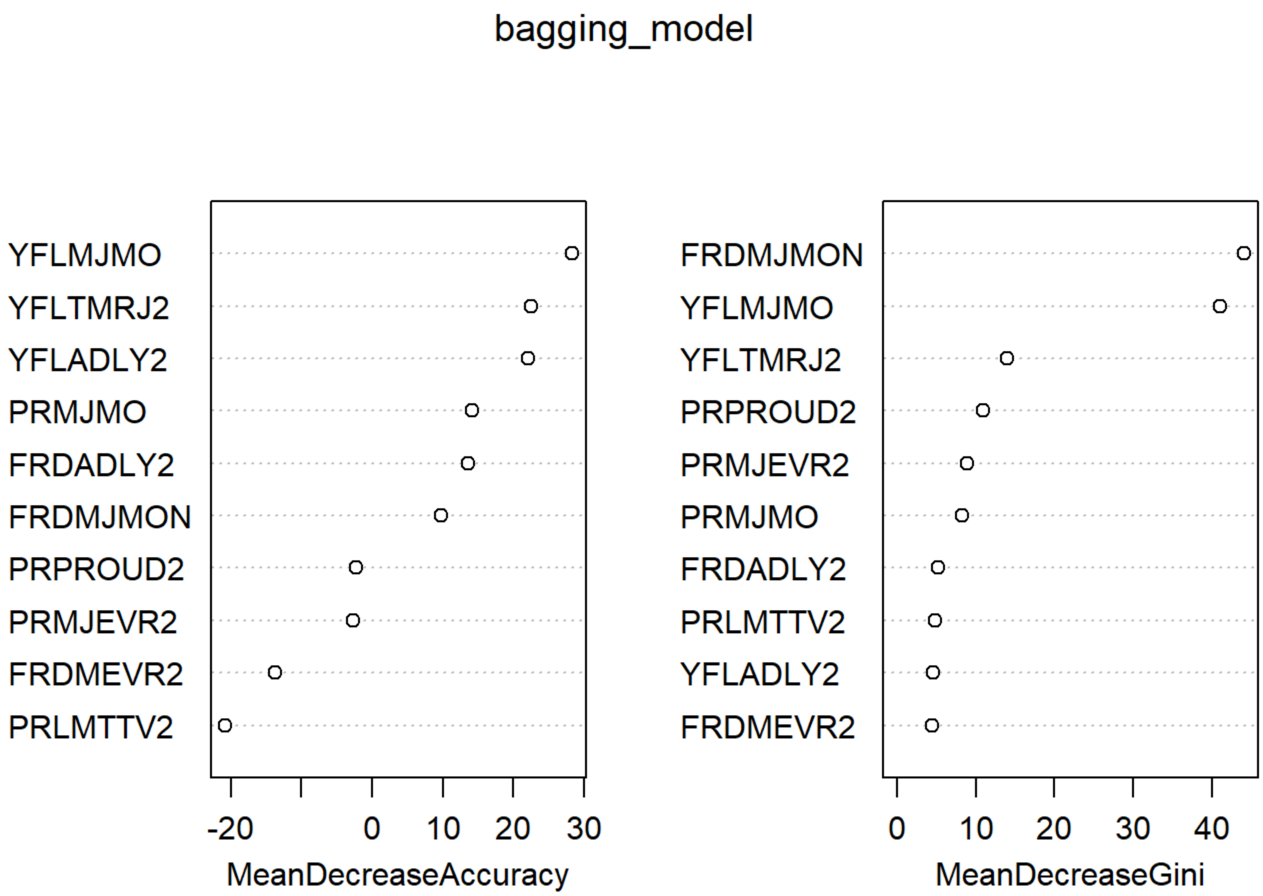
```
cat("Recall:", round(recall_bag2, 4), "\n")
```

```
## Recall: 0.6692
```

```
cat("F1-Score:", ifelse(is.na(f1_bag2), "NaN", round(f1_bag2, 4)), "\n")
```

```
## F1-Score: 0.3644
```

```
# 9. Feature importance plot
varImpPlot(bagging_model)
```



3.5 Random Forest

```
# 1. Important variables
important_vars <- c(
  "FRDMJMON", "YFLMJMO", "FRDMEVR2", "YFLTMRJ2", "PRMJMO",
  "PRMJEVR2", "FRDADLY2", "YFLADLY2", "PRPROUD2", "PRLMTTV2"
)

# 2. Prepare dataset
youth_binary <- youthdf %>%
  select(all_of(important_vars), TOBFLAG)

# Convert TOBFLAG to factor
youth_binary$TOBFLAG <- as.factor(youth_binary$TOBFLAG)

# 3. Train/test split
set.seed(42)
train_idx <- sample(1:nrow(youth_binary), 0.7 * nrow(youth_binary))
training_data <- youth_binary[train_idx, ]
testing_data <- youth_binary[-train_idx, ]

# 4. Train Random Forest with nodesize and maxnodes
set.seed(42)
rf_model <- randomForest(
  TOBFLAG ~ .,
  data = training_data,
  mtry = 3,
  ntree = 5000,
  nodesize = 5,
  maxnodes = 30,
  importance = TRUE
)

# 5. Predict on test data
rf_pred <- predict(rf_model, newdata = testing_data, type = "class")
actual <- testing_data$TOBFLAG

# 6. Confusion matrix
confusion_matrix_rf <- table(Predicted = rf_pred, Actual = actual)
print(confusion_matrix_rf)
```

```
##          Actual
## Predicted    0    1
##         0 2209  266
##         1    0    0
```

```
# 7. Evaluation metrics with safety checks
tp <- sum(rf_pred == "1" & actual == "1")
fp <- sum(rf_pred == "1" & actual == "0")
fn <- sum(rf_pred == "0" & actual == "1")

precision_rf2 <- ifelse((tp + fp) == 0, 0, tp / (tp + fp))
recall_rf2 <- ifelse((tp + fn) == 0, 0, tp / (tp + fn))
f1_rf2 <- ifelse((precision_rf2 + recall_rf2) == 0, 0,
                2 * precision_rf2 * recall_rf2 / (precision_rf2 + recall_rf2))

accuracy_rf2 <- mean(rf_pred == actual)
test_error_rate_rf2 <- 1 - accuracy_rf2

# 8. Print results
cat("\nRandom Forest Evaluation Metrics:\n")
```

```
##
## Random Forest Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_rf2, 4), "\n")
```

```
## Accuracy: 0.8925
```

```
cat("Test Error Rate:", round(test_error_rate_rf2, 4), "\n")
```

```
## Test Error Rate: 0.1075
```

```
cat("Precision:", round(precision_rf2, 4), "\n")
```

```
## Precision: 0
```

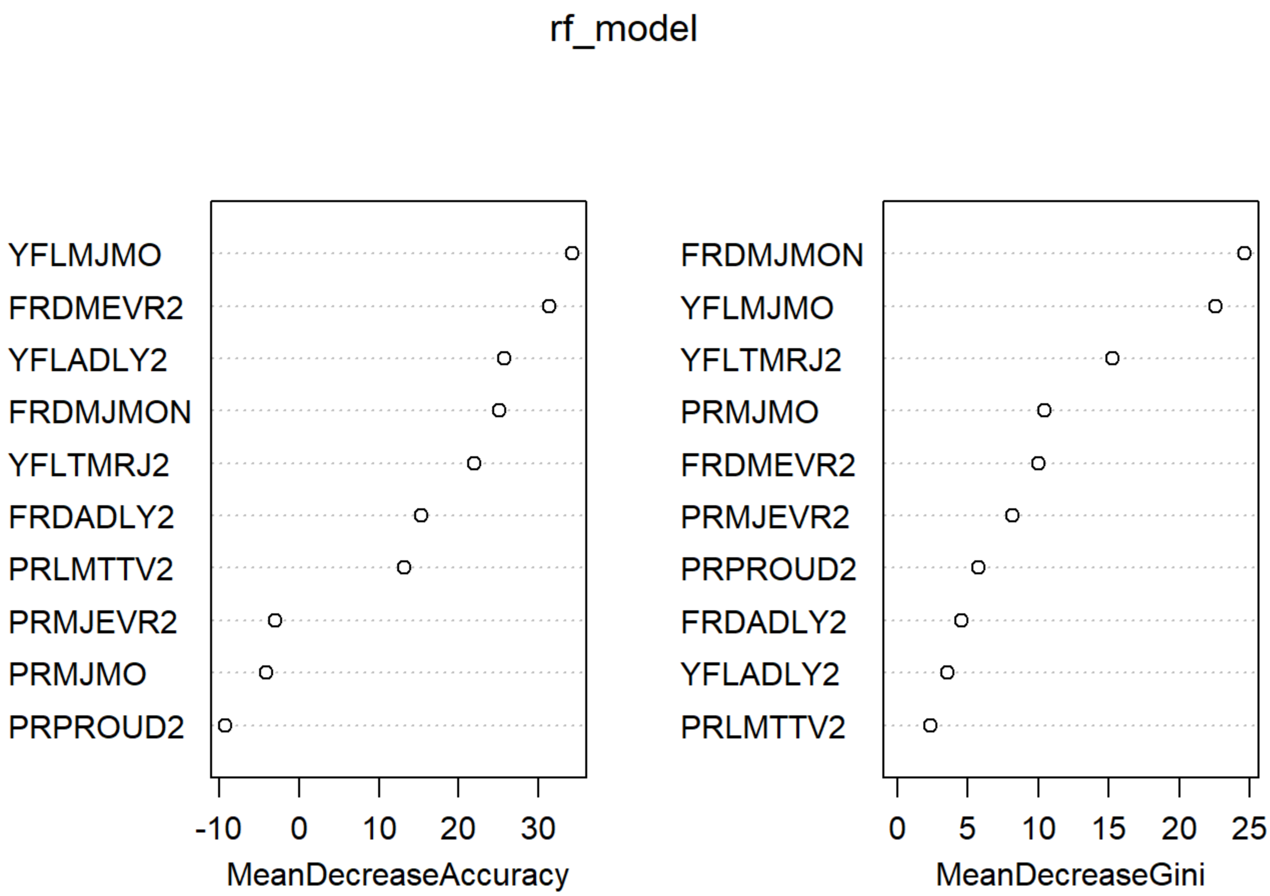
```
cat("Recall:", round(recall_rf2, 4), "\n")
```

```
## Recall: 0
```

```
cat("F1-Score:", round(f1_rf2, 4), "\n")
```

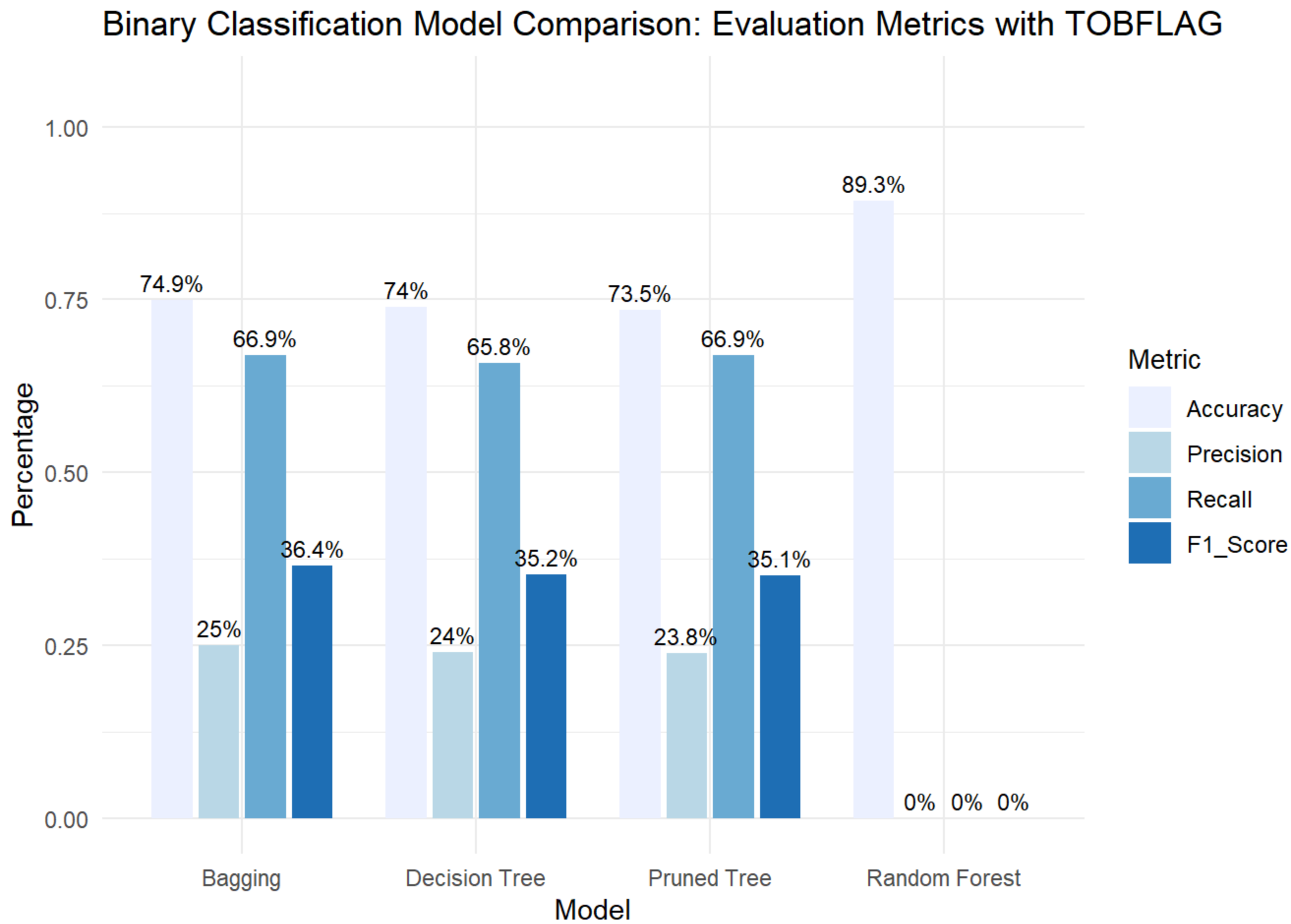
```
## F1-Score: 0
```

```
# 9. Optional: Variable importance plot
varImpPlot(rf_model)
```



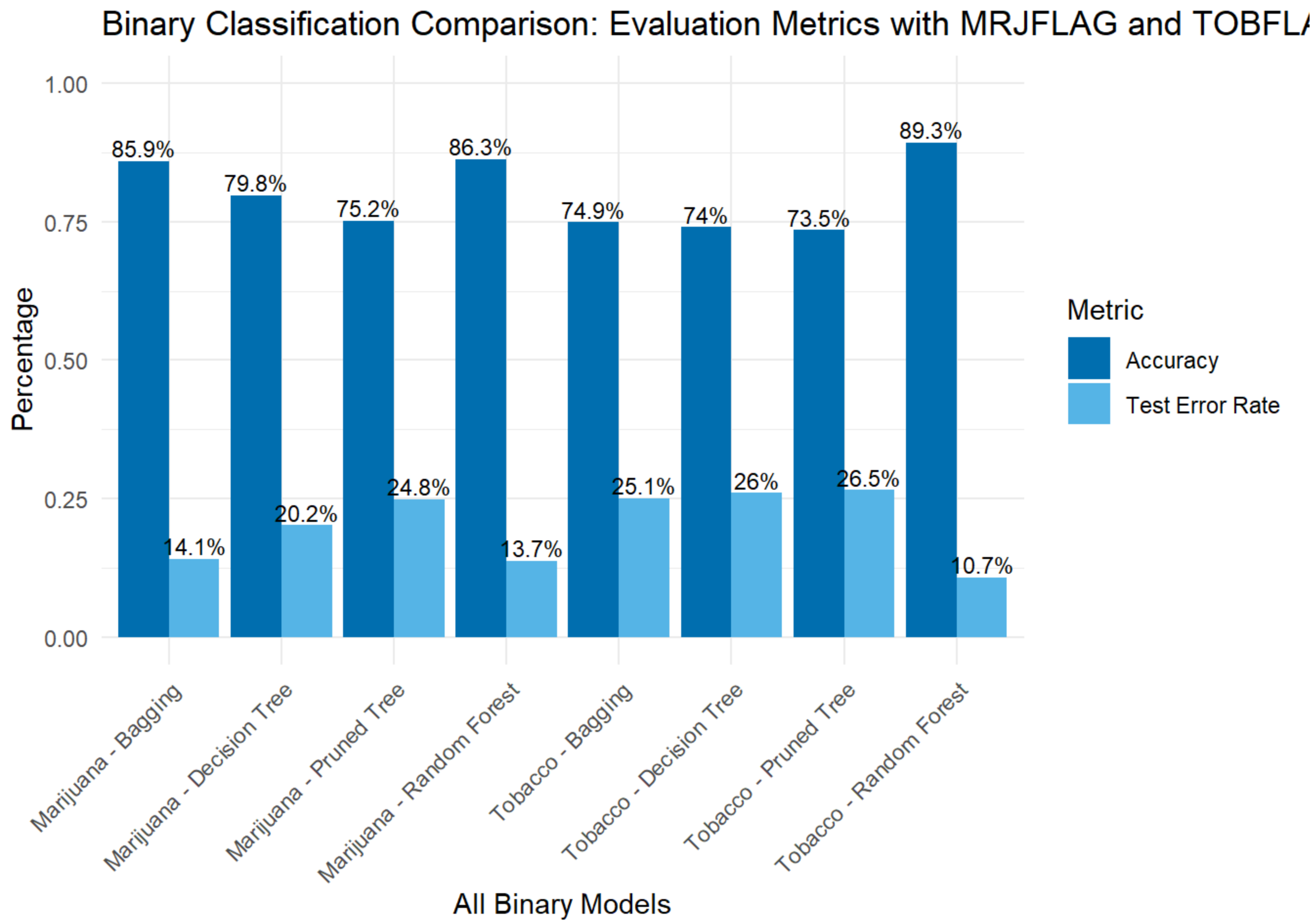
3.6 Comparison of Binary Classification Methods Decision Tree, Pruned Tree, Bagging and random Forest with “TOBFLAG”

```
# 1. Define model names and evaluation metrics
model_names <- c("Decision Tree", "Pruned Tree", "Bagging", "Random Forest")

accuracy_values_tobacco <- c(accuracy_dt2, accuracy_pru2, accuracy_bag2, accuracy_rf2)
precision_values_tobacco <- c(precision_dt2, precision_pru2, precision_bag2, precision_rf2)
recall_values_tobacco <- c(recall_dt2, recall_pru2, recall_bag2, recall_rf2)
f1_score_values_tobacco <- c(f1_dt2, f1_score_pru2, f1_bag2, f1_rf2)

# 2. Create full metrics data frame for reshaping
metrics_df <- data.frame(
  Model = model_names,
  Accuracy = accuracy_values_tobacco,
  Precision = precision_values_tobacco,
  Recall = recall_values_tobacco,
  F1_Score = f1_score_values_tobacco
)

# 3. Reshape to long format
metrics_long <- melt(metrics_df, id.vars = "Model",
                     variable.name = "Metric", value.name = "Score")

# 4. Plot all metrics (no Accuracy vs Error Rate plot)
ggplot(metrics_long, aes(x = Model, y = Score, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_text(aes(label = paste0(round(Score * 100, 1), "%")),
            position = position_dodge(width = 0.8),
            vjust = -0.5, size = 3) +
  scale_fill_brewer(palette = "Blues") +
  labs(title = "Binary Classification Model Comparison: Evaluation Metrics with TOBFLAG",
       x = "Model", y = "Percentage") +
  ylim(0, 1.05) +
  theme_minimal() +
  theme(legend.title = element_text(size = 10),
        legend.text = element_text(size = 9),
        axis.text.x = element_text(angle = 0, hjust = 0.5))
```



4. Comparison of Binary Classification Methods Decision Tree, Pruned Tree, Bagging and random Forest with “MRJFLAG” and “TOBFLAG”

```
# 1. Model names and substances
model_names <- c("Decision Tree", "Pruned Tree", "Bagging", "Random Forest")
substances <- c("Marijuana", "Tobacco")

# 2. Metrics for each substance
accuracy_values_marijuana <- c(accuracy_dt, accuracy_pru, accuracy_bag, accuracy_rf)
precision_values_marijuana <- c(precision_dt, precision_pru, precision_bag, precision_rf)
recall_values_marijuana <- c(recall_dt, recall_pru, recall_bag, recall_rf)
f1_score_values_marijuana <- c(f1_dt, f1_score_pru, f1_bag, f1_rf)
error_values_marijuana <- 1 - accuracy_values_marijuana

accuracy_values_tobacco <- c(accuracy_dt2, accuracy_pru2, accuracy_bag2, accuracy_rf2)
precision_values_tobacco <- c(precision_dt2, precision_pru2, precision_bag2, precision_rf2)
recall_values_tobacco <- c(recall_dt2, recall_pru2, recall_bag2, recall_rf2)
f1_score_values_tobacco <- c(f1_dt2, f1_score_pru2, f1_bag2, f1_rf2)
error_values_tobacco <- 1 - accuracy_values_tobacco

# 3. Create labels: "Marijuana - Decision Tree", etc.
x_labels <- paste(rep(substances, each = length(model_names)),
                  rep(model_names, times = length(substances)),
                  sep = " - ")

# 4. Create comparison data frame
comparison_df <- data.frame(
  Method = rep(x_labels, times = 2),
  Metric = rep(c("Accuracy", "Test Error Rate"), each = length(x_labels)),
  Value = c(accuracy_values_marijuana, accuracy_values_tobacco,
            error_values_marijuana, error_values_tobacco)
)

# 5. Custom color palette
custom_palette <- c("Accuracy" = "#0072B2", "Test Error Rate" = "#56B4E9")

# 6. Plot
ggplot(comparison_df, aes(x = Method, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_text(aes(label = paste0(round(Value * 100, 1), "%")),
            position = position_dodge(width = 0.9),
            vjust = -0.25, size = 3) +
  scale_fill_manual(values = custom_palette) +
  labs(title = "Binary Classification Comparison: Evaluation Metrics with MRJFLAG and TOBFLAG",
       x = "All Binary Models", y = "Percentage") +
  theme_minimal() +
  ylim(0, 1) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



5. Multi-class Classification “MRJYDAYS”

Number of days of marijuana in past year (1-5 categories, 6 = none)

5.1 Feature Selection: The Most Important Variables for Predicting “MRJYDAYS”

```
# 1.Select youth experience variables and add MRJYDAYS
df_corr <- youthdf %>%
  select(SCHFELT:RLGFRND, MRJYDAYS)

# 2.Encode all categorical variables (if not numeric, factorize as integer)
df_corr[] <- lapply(df_corr, function(col) {
  if (is.factor(col) || is.character(col)) {
    as.integer(factor(col))
  } else {
    col
  }
})

# 3. Compute correlations
corr_matrix <- cor(df_corr, use = "complete.obs")
corr_mrjydays <- sort(corr_matrix[, "MRJYDAYS"][-which(names(corr_matrix[, "MRJYDAYS"]) == "MRJY
DAYS")], decreasing = TRUE)

# 4.Print top and bottom correlations
cat("\nTop correlated features with MRJYDAYS:\n")
```

```
##
## Top correlated features with MRJYDAYS:
```

```
print(head(corr_mrjydays, 10))
```

```
##    STNDSMJ    STNDALC   YOSTOLE2    STNDDNK    YOSELL2   YOATTAK2   YOFIGHT2
## 0.32048430 0.23141043 0.17120623 0.14184577 0.12778990 0.09700066 0.09004114
##   YOGRPFT2    YOHGUN2   AVGGRADE
## 0.08259105 0.07860397 0.07406352
```

```
cat("\nLeast correlated features with MRJYDAYS:\n")
```

```
##
## Least correlated features with MRJYDAYS:
```

```
print(tail(corr_mrjydays, 10))
```

```
##   PRGDJOB2   PRPROUD2   YFLADLY2   FRDADLY2     PRMJMO   PRMJEVR2   YFLTMRJ2
## -0.1238442 -0.1246756 -0.1337351 -0.1409722 -0.2772494 -0.2836012 -0.3362317
##   FRDMEVR2    YFLMJMO   FRDMJMON
## -0.3455380 -0.3584304 -0.3726979
```

5.2 Decision Tree

Target Classes: Light (0), Moderate (1), Heavy (2)

```
# 1. Define peer features
important_vars <- c(
  "STNDSMJ", "STNDALC", "YOSTOLE2", "STNDDNK", "YOSELL2",
  "YOATTAK2", "YOFIGHT2", "YOGRPFT2", "YOHGUN2", "AVGGRADE"
)

# 2. Filter data and drop missing values
df_model <- youthdf %>%
  filter(MRJYDAYS %in% 1:6) %>%
  select(all_of(important_vars), MRJYDAYS) %>%
  na.omit()

# 3. Map MRJYDAYS to 3-class target variable
df_model$MRJYDAYS_GROUP <- case_when(
  df_model$MRJYDAYS %in% c(1, 2) ~ 0,  # Light
  df_model$MRJYDAYS %in% c(3, 4) ~ 1,  # Moderate
  df_model$MRJYDAYS %in% c(5, 6) ~ 2   # Heavy
)

# 4. Convert character columns to numeric if needed
for (col in important_vars) {
  if (is.character(df_model[[col]])) {
    df_model[[col]] <- as.integer(as.factor(df_model[[col]]))
  }
}

# 5. Convert target to factor with labels
df_model$MRJYDAYS_GROUP <- factor(df_model$MRJYDAYS_GROUP,
                                  levels = c(0, 1, 2),
                                  labels = c("Light", "Moderate", "Heavy"))

# 6. Train/test split with stratification
set.seed(42)
train_idx <- createDataPartition(df_model$MRJYDAYS_GROUP, p = 0.7, list = FALSE)
train_data <- df_model[train_idx, ]
test_data  <- df_model[-train_idx, ]

# 7. Create weights for training data
class_counts <- table(train_data$MRJYDAYS_GROUP)
total <- sum(class_counts)
weights_lookup <- total / (length(class_counts) * class_counts)
train_weights <- weights_lookup[train_data$MRJYDAYS_GROUP]

# 8. Train decision tree with class weights
tree_model <- rpart(
  MRJYDAYS_GROUP ~ . -MRJYDAYS,
  data = train_data,
  method = "class",
  weights = train_weights,
  control = rpart.control(cp = 0.001, maxdepth = 6)
)
```

```
# 9. Predict on test data
preds <- predict(tree_model, newdata = test_data, type = "class")

# 10. Evaluate model
conf_matrix <- confusionMatrix(preds, test_data$MRJYDAYS_GROUP)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Light Moderate Heavy
##   Light       87       34   501
##   Moderate    50       56   206
##   Heavy       54       22  1463
##
## Overall Statistics
##
##                Accuracy : 0.6494
##                  95% CI : (0.6302, 0.6682)
##     No Information Rate : 0.8775
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1824
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Light Class: Moderate Class: Heavy
## Sensitivity               0.45550         0.50000       0.6742
## Specificity               0.76556         0.89157       0.7492
## Pos Pred Value            0.13987         0.17949       0.9506
## Neg Pred Value            0.94381         0.97409       0.2430
## Prevalence                0.07723         0.04529       0.8775
## Detection Rate            0.03518         0.02264       0.5916
## Detection Prevalence      0.25152         0.12616       0.6223
## Balanced Accuracy         0.61053         0.69579       0.7117
```

```
# 11. Extract key evaluation metrics
accuracy_dt3 <- conf_matrix$overall["Accuracy"]
test_error_rate_dt3 <- 1 - accuracy_dt3
precision_dt3 <- conf_matrix$byClass[,"Pos Pred Value"]
recall_dt3 <- conf_matrix$byClass[,"Sensitivity"]
f1_dt3 <- 2 * (precision_dt3 * recall_dt3) / (precision_dt3 + recall_dt3)

# Display metrics
cat("\nModel Evaluation Metrics:\n")
```

```
##
## Model Evaluation Metrics:
```

```
cat("Accuracy:", round(accuracy_dt3, 4), "\n")
```

```
## Accuracy: 0.6494
```

```
cat("Test Error Rate:", round(test_error_rate_dt3, 4), "\n\n")
```

```
## Test Error Rate: 0.3506
```

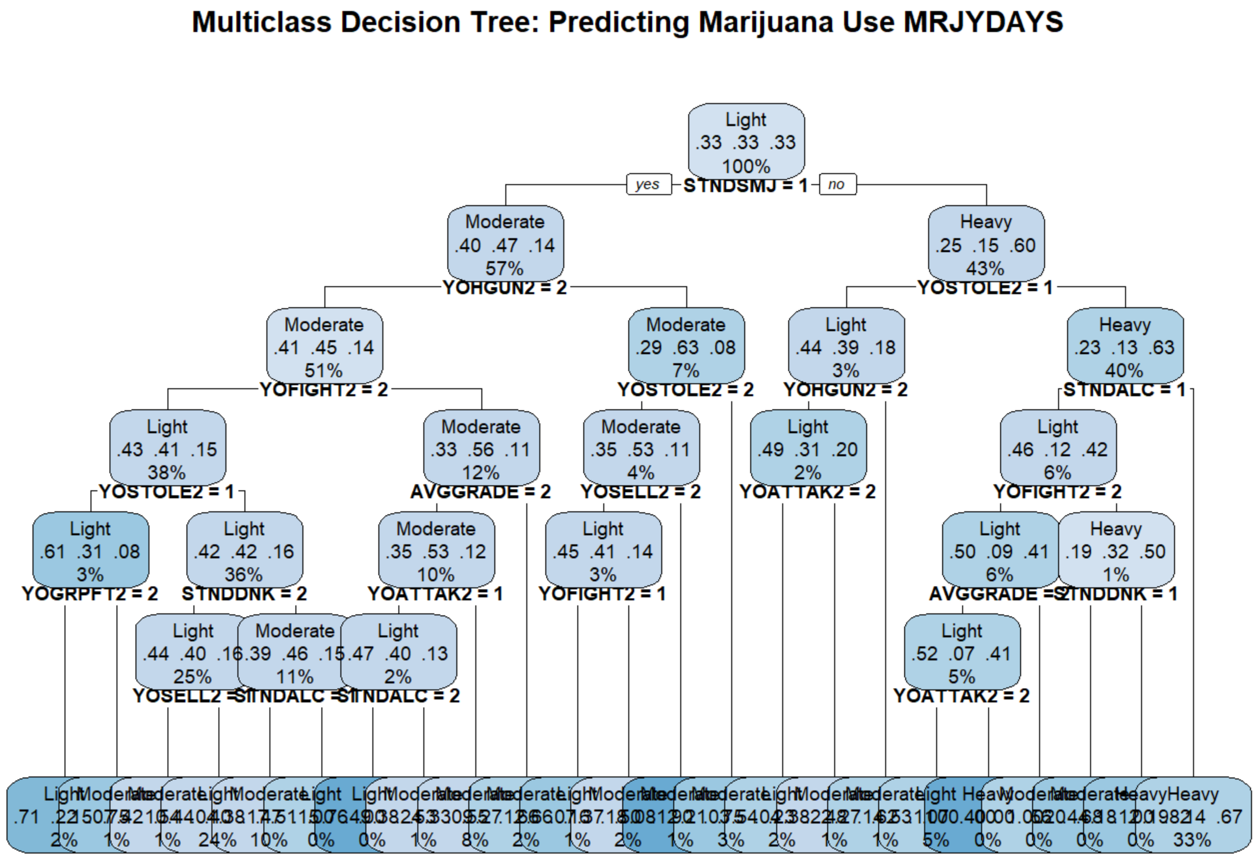
```
cat("Class-wise Metrics:\n")
```

```
## Class-wise Metrics:
```

```
metrics_df <- data.frame(
  Class = rownames(conf_matrix$byClass),
  Precision = round(precision_dt3, 4),
  Recall = round(recall_dt3, 4),
  F1_Score = round(f1_dt3, 4)
)
print(metrics_df)
```

```
##                            Class Precision Recall F1_Score
## Class: Light       Class: Light    0.1399 0.4555   0.2140
## Class: Moderate Class: Moderate    0.1795 0.5000   0.2642
## Class: Heavy       Class: Heavy    0.9506 0.6742   0.7889
```

```
# 12. Visualize decision tree
rpart.plot(
  tree_model,
  type = 2,
  extra = 104,
  box.palette = "Blues",
  fallen.leaves = TRUE,
  cex = 0.6,
  main = "Multiclass Decision Tree: Predicting Marijuana Use MRJYDAYS"
)
```



5.3 Pruned Tree

```
# 1. Define important features
important_vars <- c(
  "STNDSMJ", "STNDALC", "YOSTOLE2", "STNDDNK", "YOSELL2",
  "YOATTAK2", "YOFIGHT2", "YOGRPFT2", "YOHGUN2", "AVGGRADE"
)

# 2. Filter and prepare the data
df_model <- youthdf %>%
  filter(MRJYDAYS %in% 1:6) %>%
  select(all_of(important_vars), MRJYDAYS) %>%
  na.omit()

# 3. Map MRJYDAYS to 3 classes
df_model$MRJYDAYS_GROUP <- case_when(
  df_model$MRJYDAYS %in% c(1, 2) ~ 0,
  df_model$MRJYDAYS %in% c(3, 4) ~ 1,
  df_model$MRJYDAYS %in% c(5, 6) ~ 2
)

# 4. Convert character columns to numeric
for (col in important_vars) {
  if (is.character(df_model[[col]])) {
    df_model[[col]] <- as.integer(as.factor(df_model[[col]]))
  }
}

# 5. Convert target to factor
df_model$MRJYDAYS_GROUP <- factor(df_model$MRJYDAYS_GROUP,
                                  levels = c(0, 1, 2),
                                  labels = c("Light", "Moderate", "Heavy"))

# 6. Train-test split
set.seed(42)
train_idx <- createDataPartition(df_model$MRJYDAYS_GROUP, p = 0.7, list = FALSE)
train_data <- df_model[train_idx, ]
test_data  <- df_model[-train_idx, ]

# 7. Compute class weights
class_counts <- table(train_data$MRJYDAYS_GROUP)
total <- sum(class_counts)
weights_lookup <- total / (length(class_counts) * class_counts)
train_weights <- weights_lookup[train_data$MRJYDAYS_GROUP]

# 8. Train the full tree with maxdepth and minbucket
tree_model <- rpart(
  MRJYDAYS_GROUP ~ . -MRJYDAYS,
  data = train_data,
  method = "class",
  weights = train_weights,
  control = rpart.control(cp = 0.001, maxdepth = 6, minbucket = 20)
)
```

```
# 9. Prune the tree using optimal cp
optimal_cp <- tree_model$cptable[which.min(tree_model$cptable[,"xerror"]), "CP"]
cat("Optimal CP:", optimal_cp, "\n")
```

```
## Optimal CP: 0.01261531
```

```
pruned_tree <- prune(tree_model, cp = optimal_cp)

# 10. Predict using pruned tree
preds_pruned <- predict(pruned_tree, newdata = test_data, type = "class")

# 11. Evaluate pruned tree
conf_matrix_pruned <- confusionMatrix(preds_pruned, test_data$MRJYDAYS_GROUP)
print(conf_matrix_pruned)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Light Moderate Heavy
##   Light        0        0     0
##   Moderate   115       83   479
##   Heavy       76       29  1691
##
## Overall Statistics
##
##                Accuracy : 0.7173
##                  95% CI : (0.6991, 0.735)
##     No Information Rate : 0.8775
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1932
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Light Class: Moderate Class: Heavy
## Sensitivity               0.00000         0.74107       0.7793
## Specificity               1.00000         0.74841       0.6535
## Pos Pred Value                NaN         0.12260       0.9415
## Neg Pred Value            0.92277         0.98385       0.2925
## Prevalence                0.07723         0.04529       0.8775
## Detection Rate            0.00000         0.03356       0.6838
## Detection Prevalence      0.00000         0.27376       0.7262
## Balanced Accuracy         0.50000         0.74474       0.7164
```

```
# 12. General metrics
accuracy_pru3 <- conf_matrix_pruned$overall["Accuracy"]
test_error_rate_pru3 <- 1 - accuracy_pru3
precision_pru3 <- mean(conf_matrix_pruned$byClass[,"Pos Pred Value"], na.rm = TRUE)
recall_pru3 <- mean(conf_matrix_pruned$byClass[,"Sensitivity"], na.rm = TRUE)
f1_pru3 <- mean(2 * (precision_pru3 * recall_pru3) / (precision_pru3 + recall_pru3), na.rm = TRU
E)

cat("\n===== Pruned Tree Summary Metrics =====\n")
```

```
##
## ===== Pruned Tree Summary Metrics =====
```

```
cat("Accuracy:", round(accuracy_pru3, 4), "\n")
```

```
## Accuracy: 0.7173
```

```
cat("Test Error Rate:", round(test_error_rate_pru3, 4), "\n")
```

```
## Test Error Rate: 0.2827
```

```
cat("Macro Precision:", round(precision_pru3, 4), "\n")
```

```
## Macro Precision: 0.5321
```

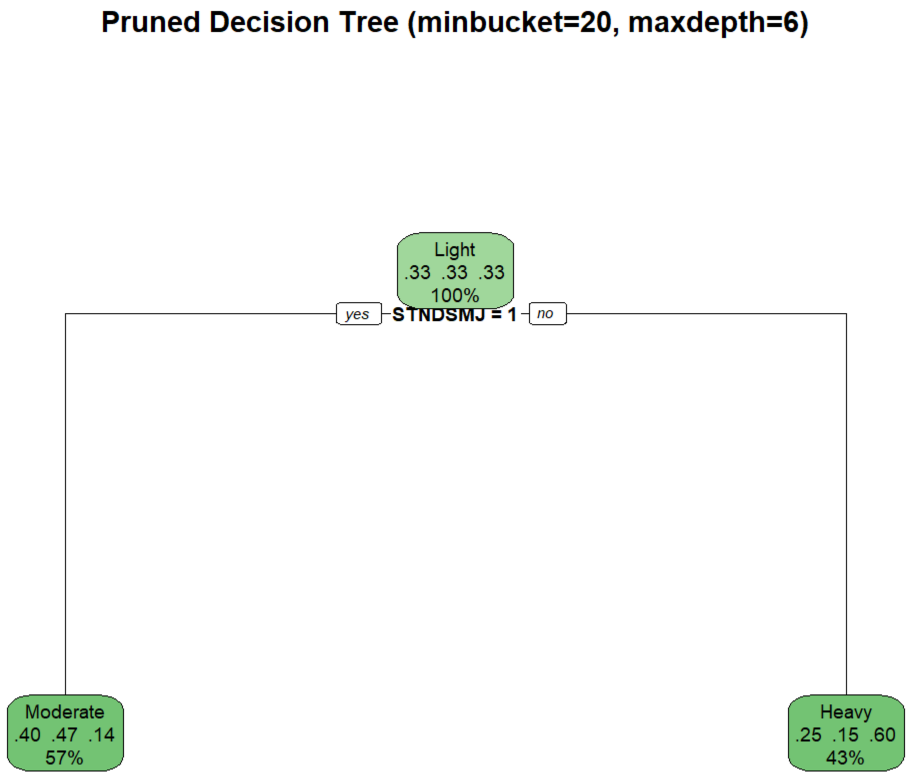
```
cat("Macro Recall:", round(recall_pru3, 4), "\n")
```

```
## Macro Recall: 0.5068
```

```
cat("Macro F1-Score:", round(f1_pru3, 4), "\n")
```

```
## Macro F1-Score: 0.5191
```

```
# 13. Plot the pruned tree
rpart.plot(
  pruned_tree,
  type = 2,
  extra = 104,
  box.palette = "Greens",
  fallen.leaves = TRUE,
  cex = 0.6,
  main = "Pruned Decision Tree (minbucket=20, maxdepth=6)"
)
```


Pruned Decision Tree (minbucket=20, maxdepth=6)
Light
.33 .33 .33
100%
yes
STNDSMJ = 1
no
Moderate
.40 .47 .14
57%
Heavy
.25 .15 .60
43%

5.4

Bagging

```
# 1. Prepare the data
important_vars <- c(
  "STNDSMJ", "STNDALC", "YOSTOLE2", "STNDDNK", "YOSELL2",
  "YOATTAK2", "YOFIGHT2", "YOGRPFT2", "YOHGUN2", "AVGGRADE"
)

# Create model dataset (exclude MRJYDAYS after creating class)
df_model <- youthdf %>%
  filter(MRJYDAYS %in% 1:6) %>%
  select(all_of(important_vars), MRJYDAYS) %>%
  na.omit()

# 2. Create multiclass outcome variable
df_model$MRJYDAYS_GROUP <- case_when(
  df_model$MRJYDAYS %in% c(1, 2) ~ "Light",
  df_model$MRJYDAYS %in% c(3, 4) ~ "Moderate",
  df_model$MRJYDAYS %in% c(5, 6) ~ "Heavy"
)

df_model$MRJYDAYS_GROUP <- as.factor(df_model$MRJYDAYS_GROUP)

# Remove MRJYDAYS to prevent leakage
df_model <- df_model %>% select(-MRJYDAYS)

# 3. Convert character columns to integer if needed
for (col in important_vars) {
  if (is.character(df_model[[col]])) {
    df_model[[col]] <- as.integer(as.factor(df_model[[col]]))
  }
}

# 4. Train-test split (70/30 stratified)
set.seed(42)
train_idx <- createDataPartition(df_model$MRJYDAYS_GROUP, p = 0.7, list = FALSE)
train_data <- df_model[train_idx, ]
test_data  <- df_model[-train_idx, ]

# 5. Set safe tree parameters
tree_control <- rpart.control(
  maxdepth = 5,        # Conservative tree depth
  minbucket = 15,      # Min obs per leaf to prevent overfitting
  cp = 0.01            # Prune small branches
)

# 6. Train Bagging model with 100 trees (reduced for safety)
set.seed(42)
bag_model <- bagging(
  MRJYDAYS_GROUP ~ .,
  data = train_data,
  nbagg = 100,             # Reduced from 1000 for generalization
  coob = TRUE,             # OOB error estimate
  control = tree_control
```

```
)

# 7. Predict on test data
bag_preds <- predict(bag_model, newdata = test_data, type = "class")

# 8. Evaluate
conf_matrix_bag <- confusionMatrix(bag_preds, test_data$MRJYDAYS_GROUP)

# 9. Macro-averaged metrics
accuracy_bag3 <- conf_matrix_bag$overall["Accuracy"]
test_error_rate_bag3 <- 1 - accuracy_bag3
precision_bag3 <- mean(conf_matrix_bag$byClass[,"Pos Pred Value"], na.rm = TRUE)
recall_bag3 <- mean(conf_matrix_bag$byClass[,"Sensitivity"], na.rm = TRUE)
f1_bag3 <- mean(2 * (precision_bag3 * recall_bag3) / (precision_bag3 + recall_bag3), na.rm = TRU
E)

# 10. Print results
cat("\n===== Bagging (nbagg = 100, Safe Tree Settings) Summary Metrics =====\n")
```

```
##
## ===== Bagging (nbagg = 100, Safe Tree Settings) Summary Metrics =====
```

```
cat("Accuracy:", round(accuracy_bag3, 4), "\n")
```

```
## Accuracy: 0.8775
```

```
cat("Test Error Rate:", round(test_error_rate_bag3, 4), "\n")
```

```
## Test Error Rate: 0.1225
```

```
cat("Macro Precision:", round(precision_bag3, 4), "\n")
```

```
## Macro Precision: 0.8775
```

```
cat("Macro Recall:", round(recall_bag3, 4), "\n")
```

```
## Macro Recall: 0.3333
```

```
cat("Macro F1-Score:", round(f1_bag3, 4), "\n")
```

```
## Macro F1-Score: 0.4831
```

5.5 Random Forest

```
# Step 1: Prepare the data
important_vars <- c(
  "STNDSMJ", "STNDALC", "YOSTOLE2", "STNDDNK", "YOSELL2",
  "YOATTAK2", "YOFIGHT2", "YOGRPFT2", "YOHGUN2", "AVGGRADE"
)

df_model <- youthdf %>%
  filter(MRJYDAYS %in% 1:6) %>%
  select(all_of(important_vars), MRJYDAYS) %>%
  na.omit()

df_model$MRJYDAYS_GROUP <- case_when(
  df_model$MRJYDAYS %in% c(1, 2) ~ "Light",
  df_model$MRJYDAYS %in% c(3, 4) ~ "Moderate",
  df_model$MRJYDAYS %in% c(5, 6) ~ "Heavy"
)

df_model$MRJYDAYS_GROUP <- as.factor(df_model$MRJYDAYS_GROUP)

# Convert character variables to integers if needed
for (col in important_vars) {
  if (is.character(df_model[[col]])) {
    df_model[[col]] <- as.integer(as.factor(df_model[[col]]))
  }
}

# Step 2: Train-test split
set.seed(42)
train_idx <- createDataPartition(df_model$MRJYDAYS_GROUP, p = 0.7, list = FALSE)
training_data <- df_model[train_idx, ]
testing_data  <- df_model[-train_idx, ]

# Step 3: Train Random Forest model
set.seed(42)
rf_model <- randomForest(
  MRJYDAYS_GROUP ~ . -MRJYDAYS,
  data = training_data,
  mtry = length(important_vars),  # Use all features at each split
  ntree = 1000,                   # Number of trees
  nodesize = 5,                   # Min samples per leaf
  maxnodes = 30,                  # Max leaf nodes
  importance = TRUE
)

# Step 4: Predict on test data
rf_preds <- predict(rf_model, newdata = testing_data)

# Step 5: Evaluate performance
conf_matrix_rf3 <- confusionMatrix(rf_preds, testing_data$MRJYDAYS_GROUP)

# Step 6: General metrics (macro)
accuracy_rf3 <- conf_matrix_rf3$overall["Accuracy"]
```

```
test_error_rate_rf3 <- 1 - accuracy_rf3
precision_rf3 <- mean(conf_matrix_rf3$byClass[,"Pos Pred Value"], na.rm = TRUE)
recall_rf3 <- mean(conf_matrix_rf3$byClass[,"Sensitivity"], na.rm = TRUE)
f1_rf3 <- mean(2 * (precision_rf3 * recall_rf3) / (precision_rf3 + recall_rf3), na.rm = TRUE)

# Step 7: Print metrics
cat("\n===== Random Forest (1000 Trees) Summary Metrics =====\n")
```

```
##
## ===== Random Forest (1000 Trees) Summary Metrics =====
```

```
cat("Accuracy:", round(accuracy_rf3, 4), "\n")
```

```
## Accuracy: 0.8795
```

```
cat("Test Error Rate:", round(test_error_rate_rf3, 4), "\n")
```

```
## Test Error Rate: 0.1205
```

```
cat("Macro Precision:", round(precision_rf3, 4), "\n")
```

```
## Macro Precision: 0.7304
```

```
cat("Macro Recall:", round(recall_rf3, 4), "\n")
```

```
## Macro Recall: 0.3651
```

```
cat("Macro F1-Score:", round(f1_rf3, 4), "\n")
```

```
## Macro F1-Score: 0.4869
```

```
# Step 8 (Optional): Variable importance
cat("\nTop 10 Important Variables:\n")
```

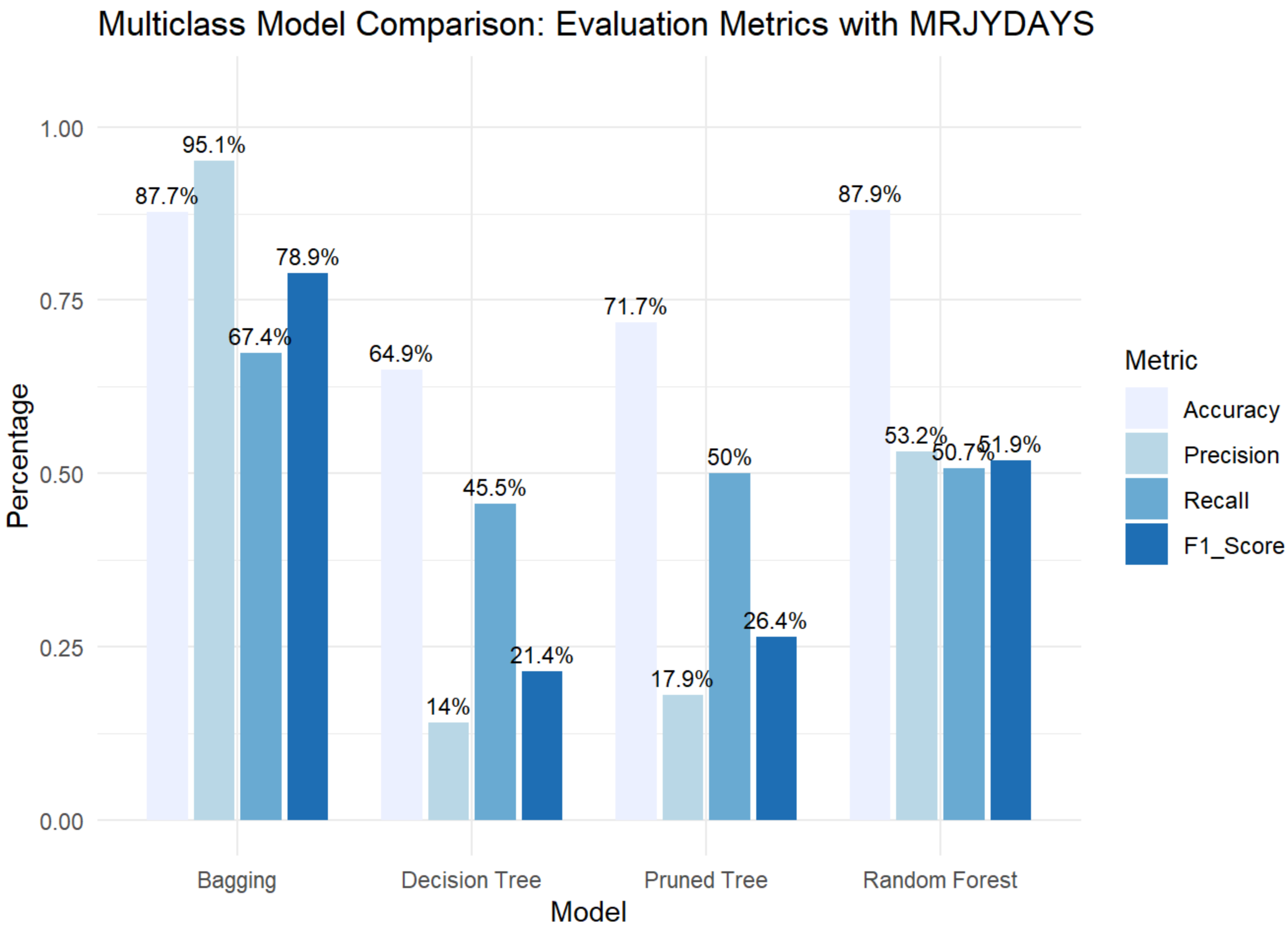
```
##
## Top 10 Important Variables:
```

```
importance_df <- importance(rf_model)
print(head(importance_df[order(importance_df[,"MeanDecreaseGini"], decreasing = TRUE), ], 10))
```

```
##                Heavy      Light  Moderate MeanDecreaseAccuracy MeanDecreaseGini
## STNDSMJ  -4.9587608  9.9272842 48.599409            23.374726       117.285402
## YOSTOLE2 13.9492702 18.5919457 25.679784            27.541974        16.594124
## YOSELL2  30.6345464 -0.5284028 32.629257            40.319894        15.319445
## YOHGUN2  -7.8027312 -4.1000411 32.852177            15.078981         9.555168
## STNDALC  -1.0324880 -3.1642622 13.953178             6.658407         8.744566
## YOATTAK2  9.6016267 -6.9398029  1.669372             9.063831         7.505459
## YOFIGHT2  6.5801682 -8.4468995 -6.920794             2.042117         7.490202
## YOGRPFT2  1.4627038 -4.5744403 12.150014             5.486276         6.694045
## AVGGRADE -7.9811324 -0.6370057  7.874096            -4.331908         5.568865
## STNDDNK   0.4329341 -6.1236066 -4.385434            -2.972418         4.247826
```

5.6 Comparison of Multiclass Classification Methods Decision Tree, Pruned Tree, Bagging and random Forest with “MRJYDAYS”

```
# 1. Define model names and evaluation metrics
model_names <- c("Decision Tree", "Pruned Tree", "Bagging", "Random Forest")

# 2. Define metrics (make sure each has exactly 4 values!)
accuracy_values_mrjydays  <- c(accuracy_dt3, accuracy_pru3, accuracy_bag3, accuracy_rf3)[1:4]
precision_values_mrjydays <- c(precision_dt3, precision_pru3, precision_bag3, precision_rf3)[1:
4]
recall_values_mrjydays    <- c(recall_dt3, recall_pru3, recall_bag3, recall_rf3)[1:4]
f1_score_values_mrjydays  <- c(f1_dt3, f1_pru3, f1_bag3, f1_rf3)[1:4]

# 3. Create full metrics data frame
metrics_df <- data.frame(
  Model = model_names,
  Accuracy = accuracy_values_mrjydays,
  Precision = precision_values_mrjydays,
  Recall = recall_values_mrjydays,
  F1_Score = f1_score_values_mrjydays
)

# 4. Reshape to long format for ggplot
library(reshape2)
metrics_long <- melt(metrics_df, id.vars = "Model",
                     variable.name = "Metric", value.name = "Score")

# 5. Plot metrics
library(ggplot2)
ggplot(metrics_long, aes(x = Model, y = Score, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_text(aes(label = paste0(round(Score * 100, 1), "%")),
            position = position_dodge(width = 0.8),
            vjust = -0.5, size = 3) +
  scale_fill_brewer(palette = "Blues") +
  labs(
    title = "Multiclass Model Comparison: Evaluation Metrics with MRJYDAYS",
    x = "Model", y = "Percentage"
  ) +
  ylim(0, 1.05) +
  theme_minimal() +
  theme(
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    axis.text.x = element_text(angle = 0, hjust = 0.5)
  )
```



6. Regression "IRCIGAGE"

Cigarette age of first use (1-55), 991 = never used

```
required_columns <- c(youth_experience_cols, "IRCIGAGE")
df_req <- df[, required_columns]
df_req <- na.omit(df_req)
dim(df_req)
```

```
## [1] 8252   48
```

```
df_req <- df_req[df_req$IRCIGAGE >= 7, ]
dim(df_req)
```

```
## [1] 8235   48
```

```
df_req <- df_req[df_req$IRCIGAGE != 991, ]
dim(df_req)
```

```
## [1] 616  48
```

6.1 Feature Selection: The Most Important Variables for Predicting “IRCIGAGE”

```
# 1. Select variables
df_corr <- df_req
df_corr$IRCIGAGE <- df_req$IRCIGAGE

# 2. Encode categorical variables as numeric using factor levels
df_corr_encoded <- df_corr %>%
  mutate(across(where(is.character), ~ as.numeric(factor(.)))) %>%
  mutate(across(where(is.factor), ~ as.numeric(.)))

# 3. Remove rows with NA values
df_corr_encoded <- na.omit(df_corr_encoded)

# 4. Compute correlations with IRCIGAGE
correlations <- cor(df_corr_encoded, use = "complete.obs")
ircigage_corr <- correlations[, "IRCIGAGE"]
ircigage_corr <- ircigage_corr[!names(ircigage_corr) %in% "IRCIGAGE"]

# 5. Sort and display top and bottom correlations
sorted_corr <- sort(ircigage_corr, decreasing = TRUE)

cat("\nTop correlated features with IRCIGAGE:\n")
```

```
##
## Top correlated features with IRCIGAGE:
```

```
print(head(sorted_corr, 10))
```

```
##   FRDMJMON   FRDMEVR2   YOFIGHT2   YFLTMRJ2    YFLMJMO   YOGRPFT2    YOHGUN2
## 0.16116072 0.14249077 0.13726380 0.12298058 0.12088910 0.11187725 0.09567001
##   PREVIOL2   PRBSOLV2   YOATTAK2
## 0.09463679 0.07231738 0.06824687
```

```
cat("\nLeast correlated features with IRCIGAGE:\n")
```

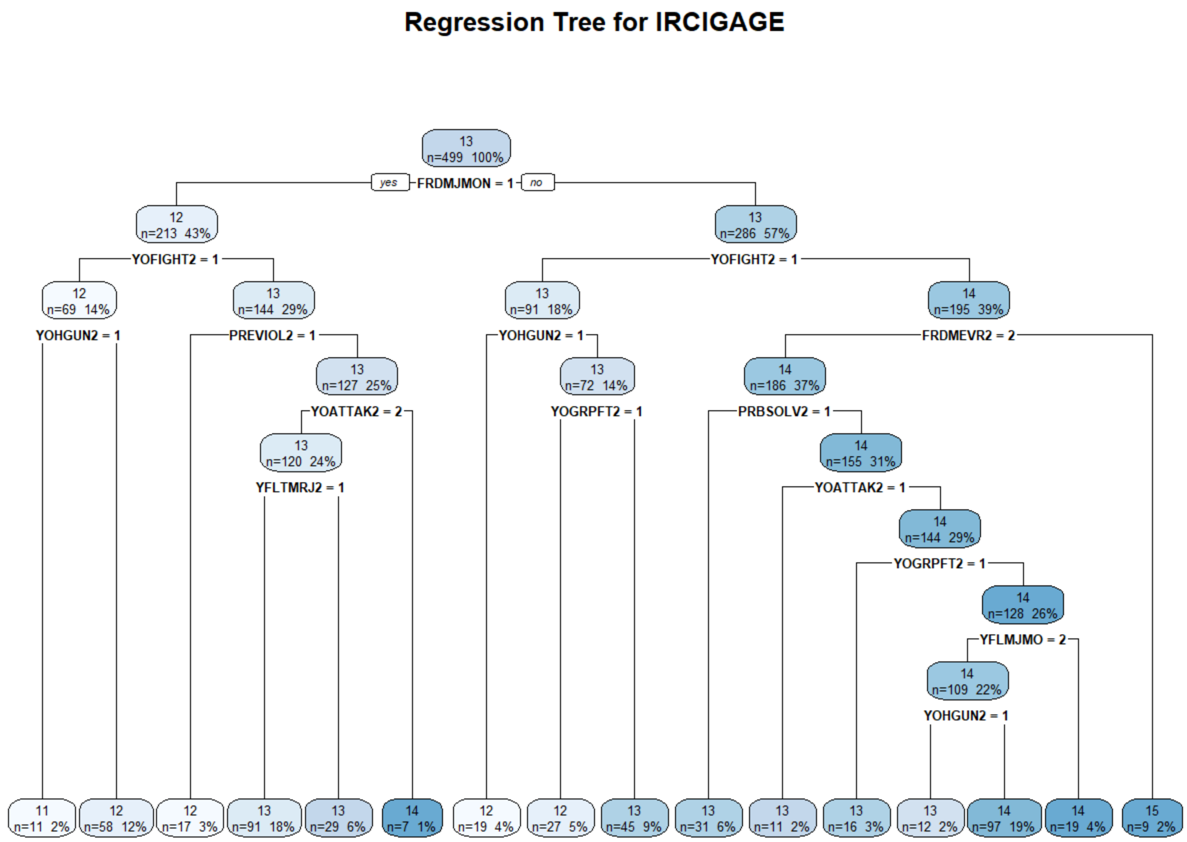
```
##
## Least correlated features with IRCIGAGE:
```

```
print(tail(sorted_corr, 10))
```

```
##     SCHFELT    YFLADLY2     STNDSMJ    PRGDJOB2    PRPROUD2    PRPKCIG2
## -0.04284459 -0.04343169 -0.05048760 -0.07973280 -0.08824700 -0.09450648
##     ARGUPAR     PRTALK3    DRPRVME3     STNDALC
## -0.10119278 -0.12777569 -0.13424944 -0.14463567
```

6.2 Decision Tree

```
# 1. Define important predictor variables and target
important_vars <- c("FRDMJMON", "FRDMEVR2", "YOFIGHT2", "YFLTMRJ2",
                    "YFLMJMO", "YOGRPFT2", "YOHGUN2", "PREVIOL2",
                    "PRBSOLV2", "YOATTAK2")
target_var <- "IRCIGAGE"

# 2. Prepare and clean data
youth_reg <- df[, c(important_vars, target_var)] %>%
  # Encode character columns as numeric
  mutate(across(where(is.character), ~ as.numeric(factor(.)))) %>%
  # Filter valid values for IRCIGAGE
  filter(!is.na(IRCIGAGE), IRCIGAGE >= 7, IRCIGAGE != 991) %>%
  # Drop rows with NA in predictors
  filter(if_all(all_of(important_vars), ~ !is.na(.)))

# 3. Train-test split
set.seed(123)
train_indices <- sample(1:nrow(youth_reg), 0.7 * nrow(youth_reg))
train_reg <- youth_reg[train_indices, ]
test_reg <- youth_reg[-train_indices, ]

# 4. Fit full regression decision tree
tree_reg <- rpart(IRCIGAGE ~ ., data = train_reg, method = "anova", cp = 0.001)

# 5. Plot the tree
rpart.plot(tree_reg, type = 2, extra = 101, fallen.leaves = TRUE,
           main = "Regression Tree for IRCIGAGE")
```




```
# 6. Predict and evaluate
pred_reg <- predict(tree_reg, newdata = test_reg)
mse_tree <- mean((pred_reg - test_reg$IRCIGAGE)^2)
rmse_tree <- sqrt(mse_tree)

cat("Mean Squared Error (MSE):", round(mse_tree, 4), "\n")
```

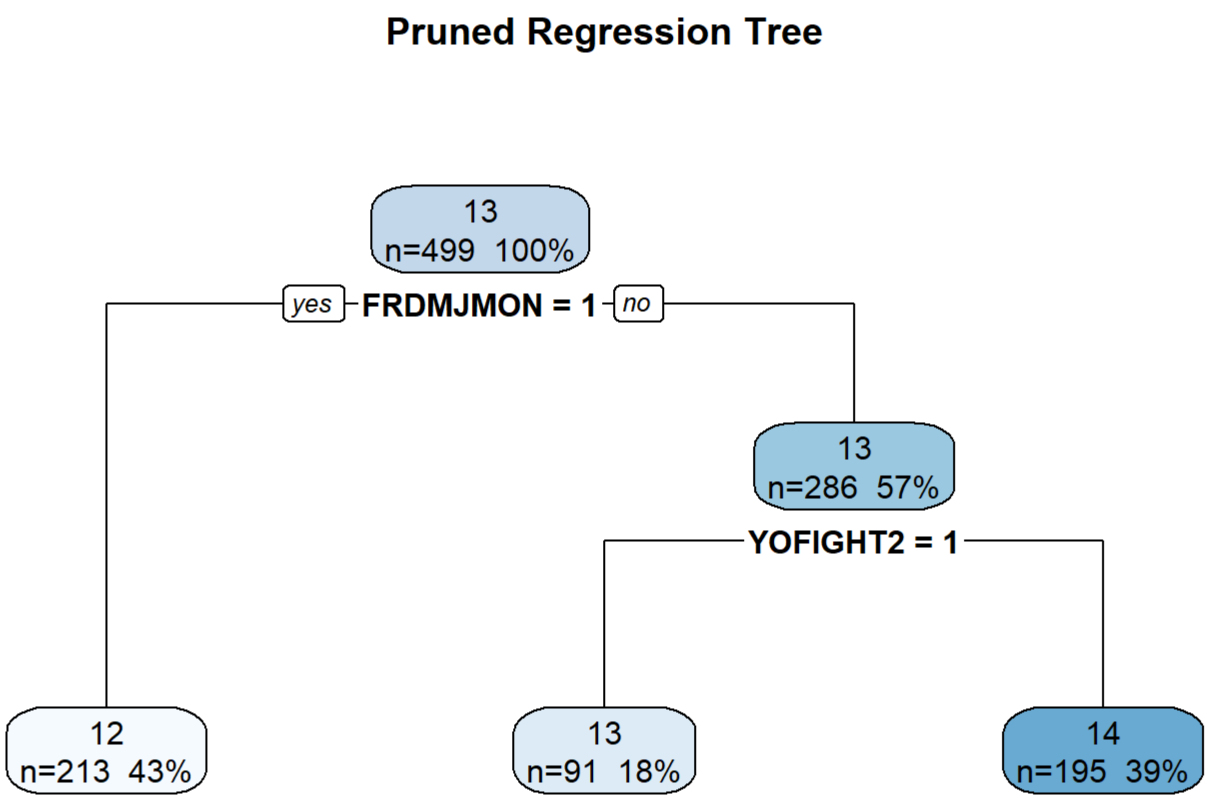
```
## Mean Squared Error (MSE): 5.8389
```

```
cat("Root Mean Squared Error (RMSE):", round(rmse_tree, 4), "\n")
```

```
## Root Mean Squared Error (RMSE): 2.4164
```

6.3 Pruned Tree

```
# 1. Prepare the data
important_vars <- c("FRDMJMON", "FRDMEVR2", "YOFIGHT2", "YFLTMRJ2",
                    "YFLMJMO", "YOGRPFT2", "YOHGUN2", "PREVIOL2",
                    "PRBSOLV2", "YOATTAK2")
target_var <- "IRCIGAGE"

youth_reg <- df[, c(important_vars, target_var)] %>%
  mutate(across(where(is.character), ~ as.numeric(factor(.)))) %>%
  filter(!is.na(IRCIGAGE), IRCIGAGE >= 7, IRCIGAGE != 991) %>%
  filter(if_all(all_of(important_vars), ~ !is.na(.)))

# 2. Train/test split
set.seed(123)
train_indices <- sample(1:nrow(youth_reg), 0.7 * nrow(youth_reg))
train_reg <- youth_reg[train_indices, ]
test_reg <- youth_reg[-train_indices, ]

# 3. Grow deep tree to enable pruning
tree_deep <- rpart(IRCIGAGE ~ ., data = train_reg, method = "anova", cp = 0)

# 4. Select best cp from cptable
cp_table <- tree_deep$cptable
cp_best <- cp_table[which.min(cp_table[, "xerror"]), "CP"]

# 5. Prune the tree using best cp
tree_pruned <- prune(tree_deep, cp = cp_best)

# 6. Plot the pruned tree
rpart.plot(tree_pruned, main = "Pruned Regression Tree", type = 2, extra = 101,
           fallen.leaves = TRUE)
```




```
# 7. Predict and evaluate
pred_pruned <- predict(tree_pruned, newdata = test_reg)
mse_pruned <- mean((pred_pruned - test_reg$IRCIGAGE)^2)
rmse_pruned <- sqrt(mse_pruned)

cat("Mean Squared Error (MSE):", round(mse_pruned, 4), "\n")
```

```
## Mean Squared Error (MSE): 5.4639
```

```
cat("Root Mean Squared Error (RMSE):", round(rmse_pruned, 4), "\n")
```

```
## Root Mean Squared Error (RMSE): 2.3375
```

6.4 Bagging

```
# Define tuning grid with large mtry values
tune_grid <- expand.grid(mtry = c(6, 7, 8, 9, 10))

# Set cross-validation
control <- trainControl(method = "cv", number = 5)

# Train with caret using full bootstrapping (bagging)
set.seed(123)
bag_tuned <- train(
  IRCIGAGE ~ .,
  data = train_reg,
  method = "rf",
  tuneGrid = tune_grid,
  trControl = control,
  ntree = 5000
)

# Evaluate
pred_bag_tuned <- predict(bag_tuned, newdata = test_reg)
mse_bag_tuned <- mean((pred_bag_tuned - test_reg$IRCIGAGE)^2)
rmse_bag_tuned <- sqrt(mse_bag_tuned)

cat("MSE:", round(mse_bag_tuned, 4), "\n")
```

```
## MSE: 5.7098
```

```
cat("RMSE:", round(rmse_bag_tuned, 4), "\n")
```

```
## RMSE: 2.3895
```

6.5 Random Forest

```
# 1. Prepare the data
important_vars <- c("FRDMJMON", "FRDMEVR2", "YOFIGHT2", "YFLTMRJ2",
                    "YFLMJMO", "YOGRPFT2", "YOHGUN2", "PREVIOL2",
                    "PRBSOLV2", "YOATTAK2")
target_var <- "IRCIGAGE"

# Filter, encode, and clean
youth_reg <- df[, c(important_vars, target_var)] %>%
  mutate(across(where(is.character), ~ as.numeric(factor(.)))) %>%
  filter(!is.na(IRCIGAGE), IRCIGAGE >= 7, IRCIGAGE != 991) %>%
  filter(if_all(all_of(important_vars), ~ !is.na(.)))

# 2. Train/test split
set.seed(123)
train_indices <- sample(seq_len(nrow(youth_reg)), 0.7 * nrow(youth_reg))
train_reg <- youth_reg[train_indices, ]
test_reg <- youth_reg[-train_indices, ]

# 3. Random Forest Tuning using caret
# Define tuning grid
tune_grid <- expand.grid(
  mtry = c(2, 3, 4, 5, 6, 7, 8, 9, 10)
)

# Set up cross-validation
control <- trainControl(method = "cv", number = 5)

# Train the tuned random forest model
set.seed(123)
rf_tuned <- train(
  IRCIGAGE ~ .,
  data = train_reg,
  method = "rf",
  trControl = control,
  tuneGrid = tune_grid,
  ntree = 500,
  importance = TRUE
)

# 4. Evaluate the model on test set
pred_rf <- predict(rf_tuned, newdata = test_reg)
mse_rf <- mean((pred_rf - test_reg$IRCIGAGE)^2)
rmse_rf <- sqrt(mse_rf)

cat("Mean Squared Error (MSE):", round(mse_rf, 4), "\n")
```

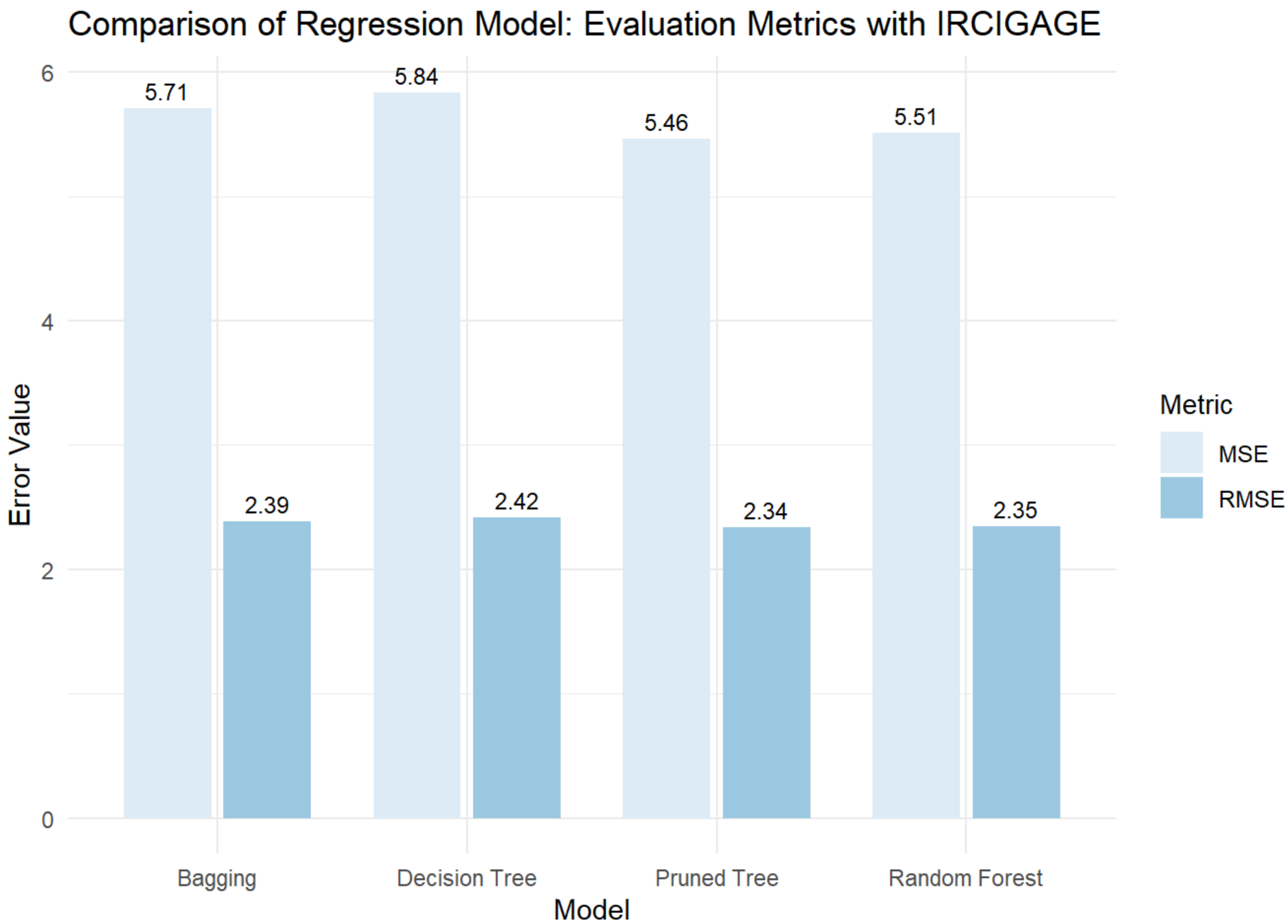
```
## Mean Squared Error (MSE): 5.5101
```

```
cat("Root Mean Squared Error (RMSE):", round(rmse_rf, 4), "\n")
```

```
## Root Mean Squared Error (RMSE): 2.3474
```

6.6 Comparison of Regression Decision Tree, Pruned Tree, Bagging and random Forest with “IRCIGAGE”

```
# 1. Define model names and evaluation metrics
model_names <- c("Decision Tree", "Pruned Tree", "Bagging", "Random Forest")

# 2. Define metrics
mean_squared_error_ircigage <- c(mse_tree, mse_pruned, mse_bag_tuned, mse_rf)
root_mean_squared_error_ircigage <- c(rmse_tree, rmse_pruned, rmse_bag_tuned, rmse_rf)

# 3. Create full metrics data frame
metrics_df <- data.frame(
  Model = model_names,
  MSE = mean_squared_error_ircigage,
  RMSE = root_mean_squared_error_ircigage
)

# 4. Reshape to long format for ggplot
metrics_long <- melt(metrics_df, id.vars = "Model",
                     variable.name = "Metric", value.name = "Score")

# 5. Plot metrics
ggplot(metrics_long, aes(x = Model, y = Score, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.7) +
  geom_text(aes(label = round(Score, 2)),  # No percentage, just numeric
            position = position_dodge(width = 0.8),
            vjust = -0.5, size = 3) +
  scale_fill_brewer(palette = "Blues") +
  labs(
    title = "Comparison of Regression Model: Evaluation Metrics with IRCIGAGE",
    x = "Model", y = "Error Value"
  ) +
  theme_minimal() +
  theme(
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    axis.text.x = element_text(angle = 0, hjust = 0.5)
  )
```


Comparison of Regression Model: Evaluation Metrics with IRCIGAGE
Error Value
0
2
4
6
5.71
2.39
5.84
2.42
5.46
2.34
5.51
2.35
Bagging
Decision Tree
Pruned Tree
Random Forest
Model
Metric
MSE
RMSE