Investigating factors influencing teen drug use

Name: Zahra Ahmadi Angali

Abstract

Teen drug usage is a matter of concern for society that needs to be addressed. In this study we hope to find the elements that contribute to some teenagers' drug usage by evaluating the National Survey on Drug Use and Health (NSDUH) data by utilizing decision trees as a guide. The data is explored through classification and regression analysis that addresses the following questions; whether teens used drugs or not, and how frequently was their usage through quantitative and categorical variables. Each question reveals an additional component of the picture concerning youth drug use. We use tree diagrams to visually trace paths that lead to answers, making sense of the complexities of teen decision-making. Investigating why youth may use more or fewer drugs. By identifying what is most important in predicting youth drug usage, we obtain insight into what impacts teens' drug decisions. These results are approached with caution, hoping to communicate them in a way that is beneficial and respectful to teenagers. At each step various models are trained leading to, Boosting with an accuracy of 90% giving the best accuracy when the exploration is to identify the elements that trigger the use of marijuana on a binary response variable. Furthermore, the Decision Tree is chosen as the best model with an accuracy of 88% presenting the best insight on factors that influence the levels of marijuana usage. As per the regression analysis, among all modeling methods conducted Random Forest is seen to out outperform the others through having the lowest Mean Squared Error. Finally, this study is a journey of exploration, utilizing decision trees and its ensemble methods that utilize decision trees as their base models to shed light on the variables influencing youth drug use, allowing us to better help them in making healthy choices.

Introduction

The NSDUH dataset, with over 5000 data through 79 variables in forms of categorical and continuous, provides a comprehensive view of the experiences that teenagers have, the data

contains substance-related columns ranging from their use of drugs, alcohol, and tobacco to their mental health state with youth-related questions and their demographic characteristics. I am interested in finding out why and how frequently some teenagers use drugs. My intention is to analyze the data in order to identify the critical elements that contribute to some teenagers' greater chance of drug usage than others. Utilizing particular tools like Decision Trees, Boosting, Random Forest and Bagging enables us to identify the key factors influencing teen marijuana consumption Finding information that can help adolescents avoid drugs and make better decisions is our main objective.
As we explore the findings, be sure to explain them in a friendly and supportive way, keeping the focus on helping teens navigate tough situations.

Theoretical Background

Decision Tree:

Decision trees are a core machine learning technique used for classification and regression tasks, in which decision nodes are placed progressively based on data characteristics. Their understanding and simplicity make them useful for understanding complex relationships, but they are susceptible to overfitting noisy data. Pruning and limiting tree depth can help minimize overfitting, and ensemble methods like Random Forests and Gradient Boosting include several decision trees to improve the accuracy of predictions. Given constraints, decision trees are frequently used to provide easy insights into patterns of data. Pruning the tree though typically not considered as a tuning parameter, cross-validation can be applied after training to remove parts of the tree that do not contribute significantly to its predictive performance.

Bagging:

Bagging is a learning technique that increases predictive performance by integrating multiple models. It bootstraps multiple subsets of the training data and trains decision trees on each one by itself. Bagging reduces variance and overfitting by an average of every one of the model predictions for regression or using a majority vote for classification, especially when used with high-variance models such as decision trees. While bagging might not improve performance if the base models are already accurate, it remains an effective approach to boost accuracy in predictions in machine learning applications.

Random Forest:

Random Forest generates multiple decision trees from random subsets of features and train data. It combines the predictions from these trees to produce a final prediction. Random Forest improves generalization by including variability in the model-building process. It has been known for being powerful and doing well across a variety of datasets. In addition, Random Forest provides information about feature importance, making it useful for analyzing patterns of data. Compromising some interpretation for its ability to predict, it is still one of the most widely used and efficient machine learning algorithms. Tuning the mtry parameter in Random Forest involves finding the optimal number of variables to consider at each split when building each tree in the ensemble. This parameter controls the randomness in feature selection during tree construction, high values for this parameter could lead to overfitting, and on the contrary low values could lead to bias therefore this is an important parameter that needs to be tuned to ensure model performance.

Boosting:

Boosting trains multiple weak models in order, leading to a strong model. It corrects previous modeling errors by focusing on incorrectly classified cases. Boosting continuously enhances the accuracy of models until an endpoint is met. Boosting is useful for reducing bias and improving generalization, but it is sensitive to noisy data and prone to overfitting. Despite these challenges, boosting is still a popular method for increasing accuracy in predictions in machine learning tasks. To optimize a boosting model, the shrinkage parameter is an important parameter to consider. This parameter determines the effect of each tree to the ensemble. A smaller shrinkage parameter reduces each tree's impact on the final predictions, which helps to prevent overfitting by reducing the influence of irrelevant features or noisy data. Instead of making significant modifications to the predictions, each tree makes smaller, more gradual changes, leading to a more robust and generalized model.

Variable selection using random forest

The huge data set is decreased to a selection of 79 potential variables that influence the teen's drug usage. Further variable selection is performed in each step whether it was binary classification multi-classification or a regression a random forest was performed on the data set with the designated response variable to identify the most influential variables. In this process, the variables with the highest influence are set aside as their influence is already proven and we proceed the analysis of other potential variables.

Methodology

The NHDUS dataset contains a vast number of variables to to name a few, variables indicating the use of various substances or, their age of first use, their frequency of use, and dived into their mental state with youth experience-related questions, their parental presence, and many more variables. The aim of this study is to pinpoint the underlying factors contributing to adolescent marijuana use, which is a significant issue in public health. Prior to our analysis, the data is preprocessed, for instance, missing values were identified and statistical imputation was performed. For categorical variables the median is assigned to the missing values and for continuous variables the mean is assigned to their missing values. Additionally, variable types are verified and converted to their appropriate types to ensure accuracy. Moreover, variable names are also changed in the binary classification and the regression analysis, for a better representation.

With the command ***duplicate()*** the dataset is carefully examined for data redundancy, with the examination it can be concluded that the data is clean of redundancy. Because the dataset is significantly huge through the feature selection method the elements that have the most weight on marijuana consumption are detected and set aside and the main objective of our analysis is to explore other variables excluding the top 5. Whether the objective was to investigate if marijuana was used or not, to determine the frequency of marijuana usage or to identify the levels of the majority of students who use marijuana, a Random Forest was implemented on the data set with the correct type of response variable that is compatible with the type of question we are investigating.

Furthermore, to ensure the accuracy of our models, the selected data set is split into training and testing sets. To make sure our model is not overfitting we have assigned 70% of the data to the training set, which is a significant amount assigned for the model to learn, and the rest to the testing set to examine how the model is performing. The models are tuned and trained on the training set and their performance tested on the testing set.

Marijuana used or not used: Binary Classification

When the objective of our model is to predict whether marijuana is used or not we are unintentionally dividing it into two categories therefore, we are exploring a binary classification. The most associated variables are excluded so that we can focus on identifying other elements influencing the use of marijuana. All models performed well but the most outstanding performance is the Boosting model with 90% accuracy on the test data set and stating variables associated with teen’s prior substance abuse and peer pressure playing a key role in their decision.

Exploring the number of days marijuana was used: Multi-Classification

To examine the number of days a teen has used marijuana throughout the range of one year the variable contains information on marijuana use in five categories with distinguished time periods. We can elaborate on the importance of tobacco use and peer pressure with our selected model, a decision tree with 88% accuracy. Our selected modeling approach, allows us to shed light on the complex interactions between these variables and teenage marijuana use. Giving us a better grasp of the varied elements that shape teenage marijuana consumption patterns.

Estimating the frequency of marijuana usage: Regression

To predict the frequency of marijuana consumption the variable ‘mrjyday’ is chosen as our response variable, and the feature selection and the model selection is assessed based on this variable. The random forest with the lowest mean squared error is selected as our chosen model predicting that most teens' choice to use marijuana is mostly influenced by their parents' lack of attention prior use of tobacco and lack of support from their friends.

Computational Results

Binary classification: Marijuana used or not

A variety of models of the decision tree, bagging, random forest, and boosting have been trained on our training set to examine the use of Marijuana on our subsetted data set. The boosting model with an accuracy of almost 91% on the test set is proven to give us the best results.

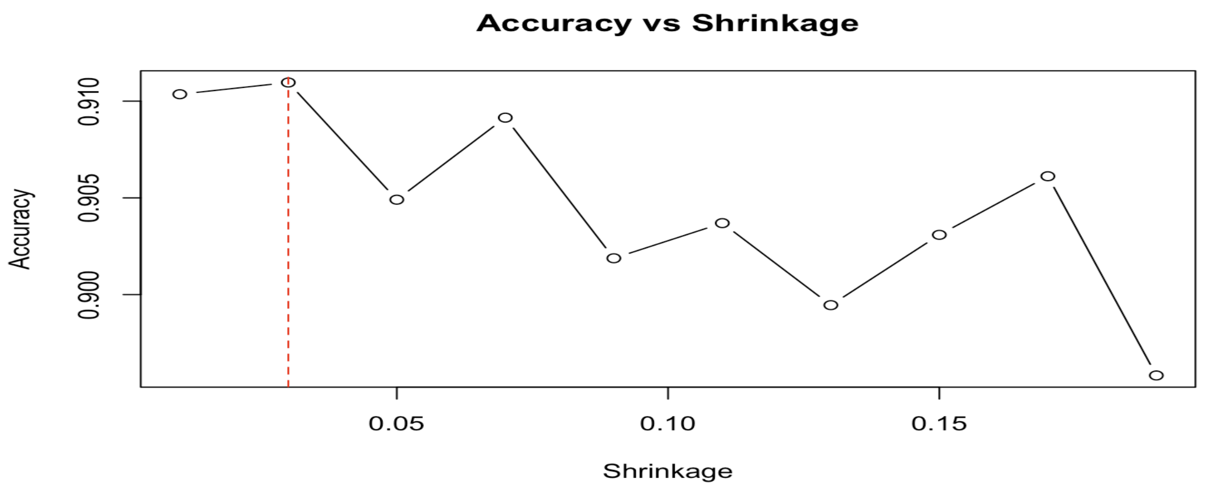


Figure1:best shrinkage value

Through experimenting with multiple shrinkage values also known as the learning rate parameter for our Boosting model, we reached an optimal value of 0.01, giving us the best performance.

The Bagging model with a mtry of 13 was implemented because in our subset for binary classification, only a subset of 14 variables was chosen. As for the Random Forest we tried various values of mtry using the ***tuneRF()*** command, and we concluded that our best mtry would be 3. As for the accuracy of the three other models, Random Forest was not far behind at 90% accuracy and Decision Tree was at almost 90% accuracy with no change in the results even with pruning. Among all of the models executed, the bagging model with an accuracy of 89% comes at last place.

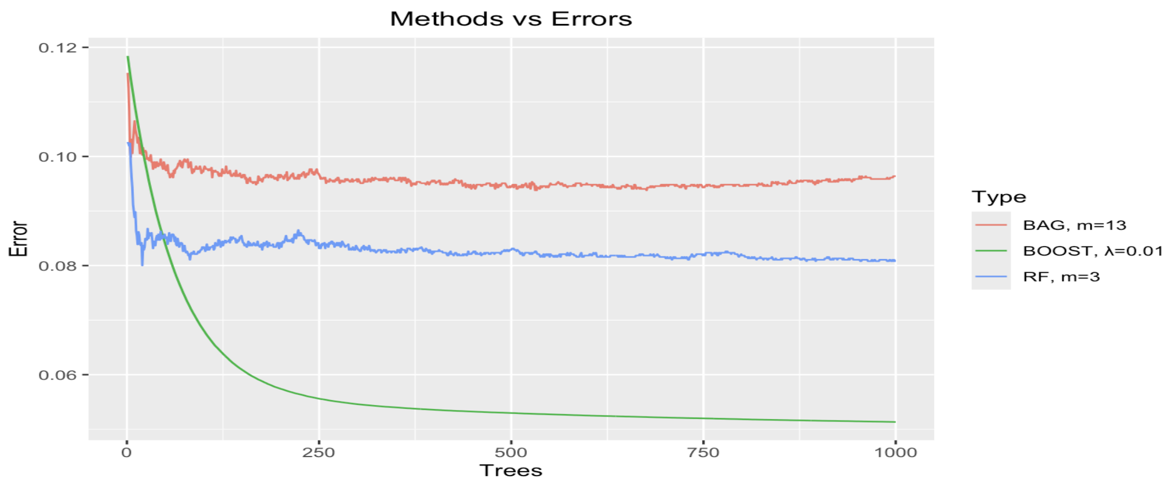


Figure 2: performance of different models based the error rate

The error rate for the three following models Bagging, Boosting and Random Forest on the training set is shown in the above figure. It can be concluded that boosting has the lowest error rate even on the training data set with 1000 trees.

Multi classification: Number of days Marijuana was used in the past year

For the analysis of the multi-classification model using the feature selection method a subset of 21 variables was used. The model performs to estimate the use of marijuana in the past. Our response has been divided in 5 categories based on the number of days throughout one year and based on the category it falls into we can determine the number of days a teen has used marijuana.

1 = 1-11 Days
2 = 12-49 Days
3 = 50-99 Days
4 = 100-299 Days
5= 300-365 Days
6 = Non-User or No Past Year Use

As a result of carrying out multiple methods, the Decision Tree with an accuracy of 88% on our test set is presented to be the best model. We furthered our exploration on the decision tree model with pruning, to see if we could obtain better results. As a result of our exploration, the pruning did not improve the performance of the model. Notably, Random Forest and Bagging models achieved accuracies of 87%, closely matching the performance of our selected model. Conversely, Boosting performed the poorest, with an accuracy of 2%. For the Boosting method, our approach involved testing various learning rates to determine the optimal value, ultimately selecting a learning rate of 0.2 with 1000 trees.

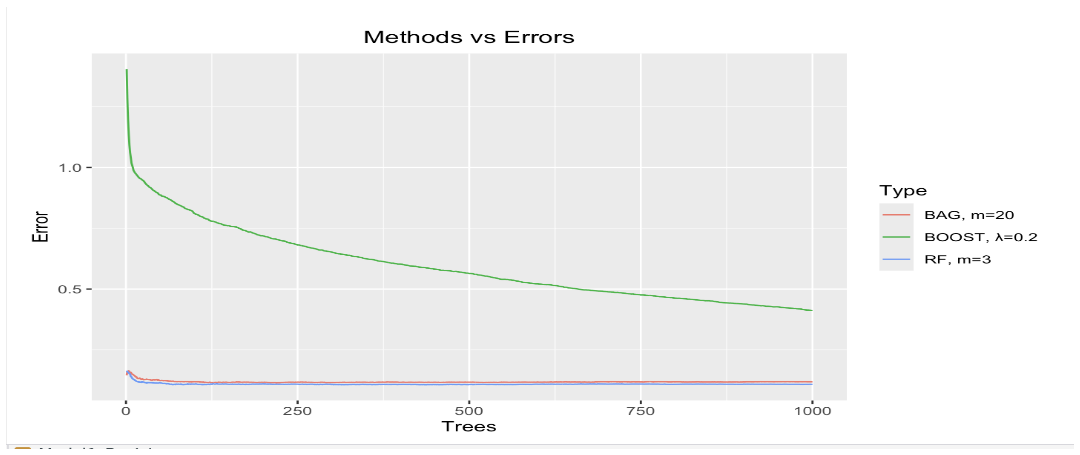
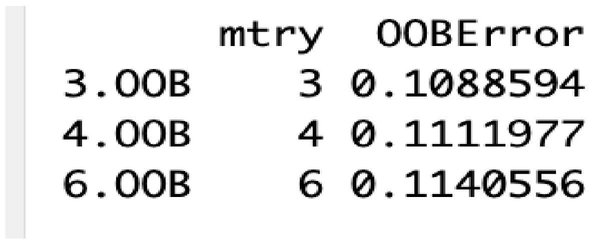


Figure 3: Methods vs error rate of the other models performed on the training set

The above demonstrates the error of the three models Bagging, Boosting, and Random Forest on the training set, from the graph we understand the Random Forest has the lowest error rate on the training set.



```
      mtry  OOBError
3.OOB    3 0.1088594
4.OOB    4 0.1111977
6.OOB    6 0.1140556
```

Figure 4: results of trying out Different mtry values for the Random forest model

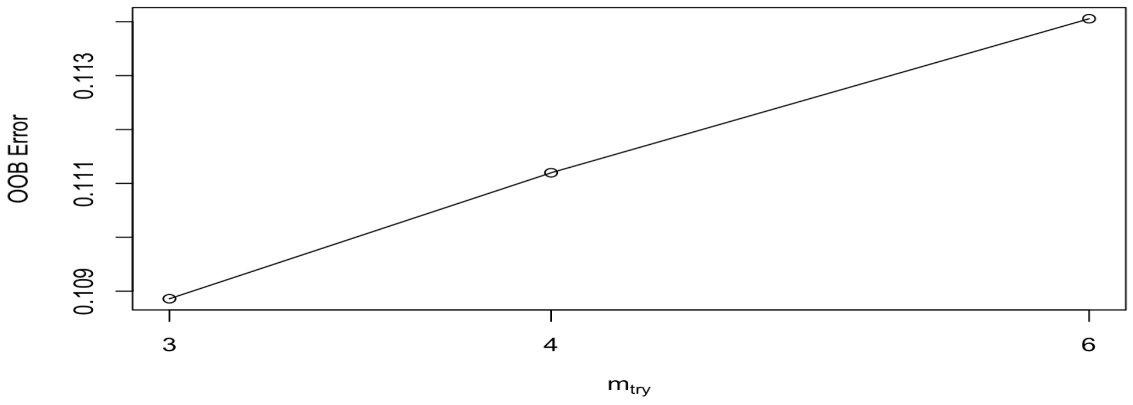


Figure 5: plotting the mtry values vs the OOB error

In an effort to discover the best random forest model, an attempt to optimize the through adjusting the parameter mtry using the tuneRF command is conducted. This command identifies the most effective mtry values and presents its findings through plots and tables, as depicted above. As the result of tuning this parameter the value of 3 is chosen as it gives the lowest error.

Regression: Predicting the frequency of marijuana usage

The regression analysis is employed to determine the frequency of marijuana use, ranging from 1 to 365 days. Initially, a subset of the dataset is selected using feature selection techniques. Subsequently, Decision Tree, Bagging, Random Forest, and Boosting models are applied to the training set of this subset. Our analysis reveals that Random Forest outperforms the other models, exhibiting a lower mean squared error of 1238 compared to the other models.

To enhance the performance of our random forest model, we fine-tuned the hyperparameter mtry using the ***tuneRF()*** command with 1000 trees. As a result, a mtry value of 5 yielded the lowest error and performed the best. Additionally, we adjusted the learning rate for other models, such as pruning the decision tree and determining the optimal shrinkage value for the Boosting model. However, despite these adjustments, none of the other models were able to surpass the performance of the Random Forest model.

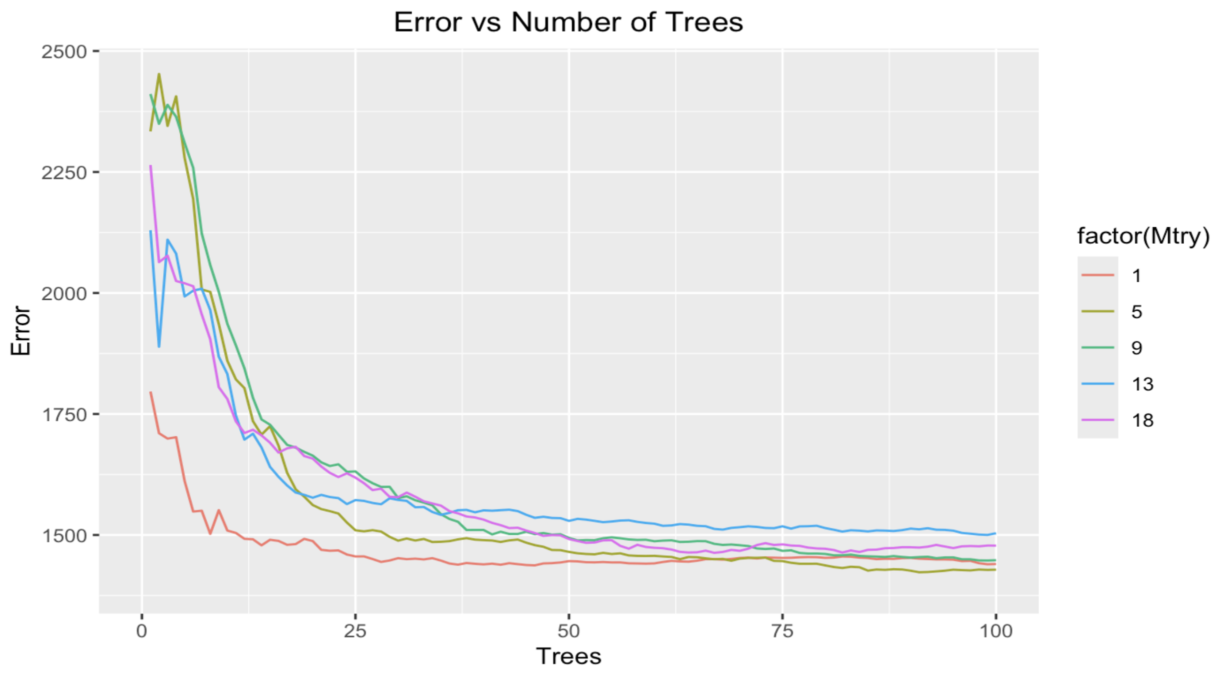


Figure6: error vs the number trees on different mtry values

Additionally, we adjusted the learning rate for other models, such as pruning the decision tree and determining the optimal shrinkage value for the Boosting model. However, despite these adjustments, none of the other models were able to surpass the performance of the Random Forest model on the testing set.

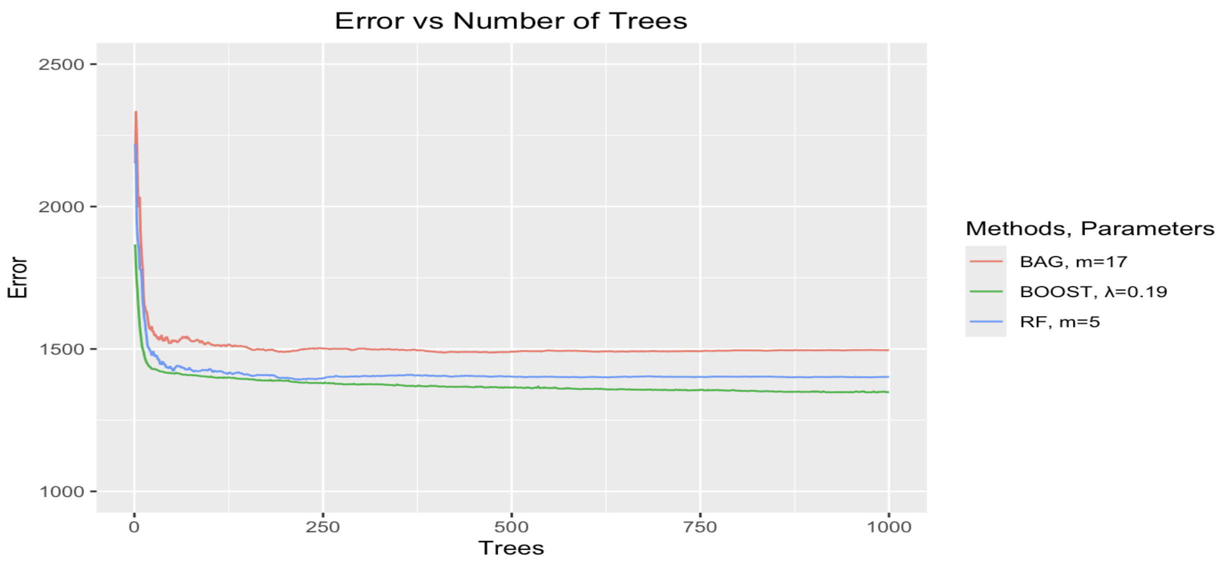


Figure 7:error rate of different methods on the training set

The error rate of the models is also seen in the above figure to see how the models are performing on the training set, indicating Boosting performed the best on the training set and has the lowest error with the shrinkage value of 0.19. As mentioned before on the efforts of tuning the shrinkage parameter for the boosting model a sequence of 0.01 to 0.2 with the step of 0.02 is trained on the model resulting in a 0.19 value to have the lowest error rate on the testing set. Therefore the boosting model is once again performed with the optimal shrinkage value, indicating it does have a better performance on the training set but does not improve its performance on the testing set.

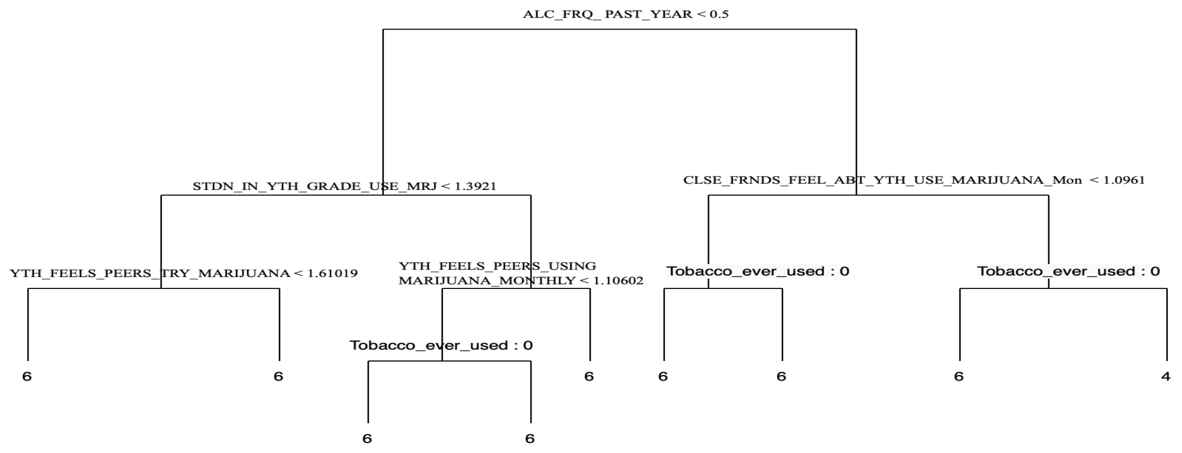
Discussion

The analysis of the NSDUH dataset has brought valuable insights into the drug-use behaviors among adolescents. Key findings drawn from the data demonstrate the significance of certain factors like prior use of alcohol, parental attitudes, and peer pressure on teenagers' decisions regarding marijuana consumption. The analysis comparing different modeling approaches revealed the tendency of a model to outperform the others based on the factors selected as our response variables, the key findings from the top-performing models are outlined below.

	var	rel.inf
	<chr>	<dbl>
tobaco_ever_used	tobaco_ever_used	21.3915938
days_used_alchohol	days_used_alchohol	17.9287418
alchohol_ever_used	alchohol_ever_used	17.2521671
student_y_grade_marijuan	student_y_grade_marijuan	9.0143609
close_youth_marijuana_mon	close_youth_marijuana_mon	8.7409676
teen_feels_peers_marijuana_monthly	teen_feels_peers_marijuana_monthly	7.7600628
close_friend_teen_mrijuana	close_friend_teen_mrijuana	6.0939712
peers_try_marijuana	peers_try_marijuana	3.8282836
race	race	3.5938634
yth_sold_drug	yth_sold_drug	1.7289648

Figure 8: top ten elements that have influenced to use of marijuana

Primary discoveries from the binary classification are indicated in the above figure, demonstrating teens who have been exposed to tobacco and alcohol are likely to try marijuana and therefore we can mostly conclude the drugs alcohol and tobacco are likely to be interdependent. Other factors that play a role in teen's use of marijuana are stated to be relevant to the grade that they are in and how their peers think about their monthly consumption of marijuana and how the teens feel about their friend's monthly use of the substance. From the overall figure above, we can state that teen's usage of marijuana is highly due to peer pressure.



ALC_FRQ_ PAST_YEAR < 0.5
STDN_IN_YTH_GRADE_USE_MRJ < 1.3921
CLSE_FRNDS_FEEL_ABT_YTH_USE_MARIJUANA_Mon < 1.0961
YTH_FEELS_PEERS_TRY_MARIJUANA < 1.61019
YTH_FEELS_PEERS_USING MARIJUANA_MONTHLY < 1.10602
Tobacco_ever_used : 0
Tobacco_ever_used : 0
Tobacco_ever_used : 0
6
6
6
6
6
6
4
6
6

Figure 9: decision tree model

With Decision Tree giving the best performance on the multi-classification approach, it can be stated from the tree above that teens who consumed alcohol and whom their friends neither approved or disapproved of their monthly use of marijuana and have used tobacco are likely to use marijuana 100 to 299 days of the year.

Insights into the influential factors driving teenage marijuana usage frequency are derived through implementing a Random Forest on the variable containing information on the frequency of a yearly marijuana intake as. As a result, factors such as tobacco usage, friends' perceptions of marijuana, and parental attitudes toward marijuana are observed to significantly influence the frequency of marijuana usage among teens.

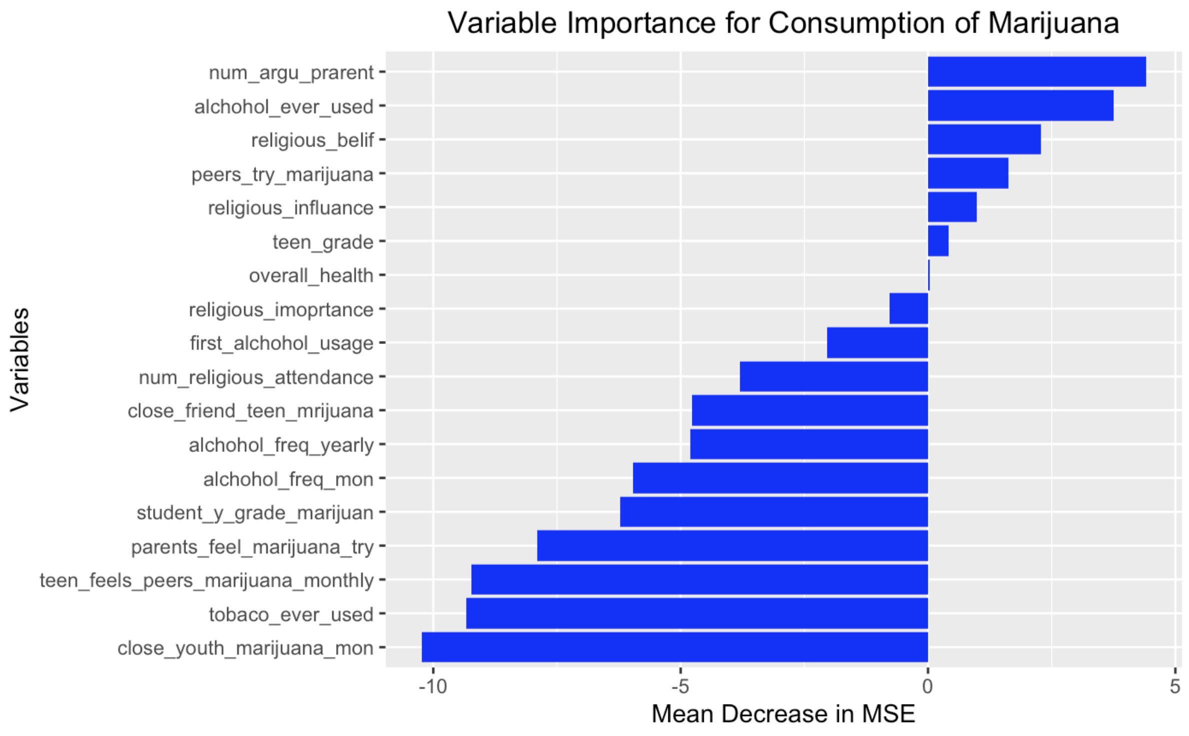


Figure 10: variable importance for consumption of marijuana

Conclusion

It can be concluded from this research that in order to address the issue of adolescent marijuana use we need to identify the factors influencing its use. This identification procedure was conducted through binary classification, a multi-classification, and a regression approach. To explore each approach the following models: Decision Trees, Bagging, Boosting, and Random forest were implemented on their appropriate response variables. Measures were taken to optimize the models. After thorough evaluation and identification of influential elements, it is evident that an annual overview of marijuana consumption frequency provides valuable insights into teen behavior, triggers, and consumption patterns. That being said, I strongly believe Addressing this issue through regression analysis offers a deeper understanding and therefore, provides better insights for future strategies to tackle this social problem.

References

1. https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-022-01939-x#Sec2

2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5026681/

3. https://www.datafiles.samhsa.gov/dataset/national-survey-drug-use-and-health-2020-nsduh-2020-ds0001

4. https://github.com/mendible/5322/blob/main/Homework%201/YouthParse.R

5. https://rdrr.io/cran/randomForest/man/tuneRF.html

Code Appendix

```
data["irmjfy"][data["irmjfy"] == 991 | data["irmjfy"] == 993] <- 0
data["iralcfy"][data["iralcfy"] == 991 | data["iralcfy"] == 993] <- 0
data["irmjfm"][data["irmjfm"] == 91 | data["irmjfm"] == 93] <- 0

# Convert only non-factor columns to factors
unordered_factor_cols <- c('alcmdays', 'mrjydays',
                           'mrjflag', 'alcflag', 'tobflag', # binary flag columns from substance
                           'irsex', 'NEWRACE2', 'eduschlgo', 'imother', 'ifather', 'govtprog', 'PDEN10', 'COUTYP4') # unordered categories for demographics

ordered_factor_cols <- c('EDUSCHGRD2','HEALTH2','POVERTY3','income','mrjmdays',"alcydays")

# Check if columns are already factors
unordered_factor_cols <- unordered_factor_cols[!sapply(data[unordered_factor_cols], is.factor)]
ordered_factor_cols <- ordered_factor_cols[!sapply(data[ordered_factor_cols], is.factor)]

# Convert to factors
data[unordered_factor_cols] <- lapply(data[unordered_factor_cols], factor) # Correct columns to unordered factors (e.g. yes, no)
data[ordered_factor_cols] <- lapply(data[ordered_factor_cols], factor, ordered=TRUE) # Correct columns to ordered factors (e.g. small, medium, large)

#looping through all columns
for (col in names(data)) {

  if (is.factor(data[[col]])) {

    data[[col]][is.na(data[[col]])] <- names(sort(table(data[[col]]), decreasing = TRUE))[1]
  }

  else if (is.numeric(data[[col]])) {

    data[[col]][is.na(data[[col]])] <- mean(data[[col]], na.rm = TRUE)
  }
}

#there is no longer any missing data in our data set

sum(is.na(data))

duplicated(data)
```

Binary Classification

```
featureselection<- randomForest(mrjflag~., data= data, ntree= 1000 , importance = TRUE)
```

```
important_variables<-importance(featureselection, type=1)

important_variables[order(important_variables, decreasing = TRUE), ]
```

With feature selection we were able to get the most important variables for our response variable we will set aside the first variables as they already show high importance on our response variable

```
df<-data%>%select(c('mrjflag','tobflag',
'alcflag','FRDMEVR2','YOSELL2','yflmjmo','frdmjmon','YFLTMRJ2','stndsmj', 'alcydays', 'irsex', 'cigmdays',
'ANYEDUC3','NEWRACE2'))

#View(df)

str(df)

new_names1 <- c(

    "marijuana_ever_used", "tobaco_ever_used", "alchohol_ever_used", "close_friend_teen_mrijuana",
"yth_sold_drug",

      "teen_feels_peers_marijuana_monthly", "close_youth_marijuana_mon", "peers_try_marijuana",
"student_y_grade_marijuan", "days_used_alchohol",

  "sex", "cig_use_mon", "substance_ed", "race")


# Rename variables

df <- df %>% rename(!!!setNames(names(.), new_names1))


#View(df)

str(df)
```

Splitting the selected data into train and test

```
set.seed(123)

train.index<- sample(1:nrow(df), nrow(df)*0.7)

df.train<- df[train.index,]

df.test<- df[-train.index,]

```
```
```

~~~
# Model_1 Decision tree

```{r}
tree.model<-tree(marijuana_ever_used~. ,data= df.train)
summary(tree.model)
plot(tree.model)
text(tree.model, pretty = 0)
predict.tree<- predict(tree.model, newdata= df.test, type= "class")
table(predict.tree, df.test$marijuana_ever_used)
mean((predict.tree == df.test$marijuana_ever_used))
#we got an accuracy of 89% ~ 90%
~~~

cross validating to find the best

```
cv.mrj <- cv.tree(tree.model, FUN = prune.misclass)
names(cv.mrj)
cv.mrj
par(mfrow = c(1, 2))
plot(cv.mrj$size, cv.mrj$dev, type = "b")
plot(cv.mrj$k, cv.mrj$dev, type = "b")
```

pruned tree

```
prune.mrj <- prune.misclass(tree.model, best = 4)
plot(prune.mrj)
text(prune.mrj, pretty = 0)

tree.pred.prune <- predict(prune.mrj, df.test,
```

```
    type = "class")

table(tree.pred.prune, df.test$marijuana_ever_used)

mean(tree.pred.prune== df.test$marijuana_ever_used)

#pruning has no effeect on the improvment of the model
```

The prunning has no improvment on the model

Model_2 Bagging

```
#Bagging

all_predictors<- length(df.train)-1

bag.model <- randomForest(marijuana_ever_used ~ ., data = df.train, mtry =all_predictors ,ntree= 1000,
importance = TRUE)

bag.model

yhat.bag <- predict(bag.model, newdata = df.test,type="class")

mean(yhat.bag == df.test$marijuana_ever_used)

table(yhat.bag, df.test$marijuana_ever_used)

#there is 89% accuracy with bagging slightly worse

importance(bag.model)

varImpPlot(bag.model)

# Get variable importance

importance_df <- as.data.frame(importance(bag.model))

importance_df <- importance_df[order(-importance_df$MeanDecreaseAccuracy), ]

top_predictors <- head(importance_df, 20)

top_predictors       <-       top_predictors[,       c("MeanDecreaseAccuracy",
"MeanDecreaseGini")]

top_predictors$PREDICTORS <- rownames(top_predictors)

#with  decision  tree-based  models  and  want  to  understand  the  importance  of
variables in creating informative splits, MDG may provide more insights.
```

```
ggplot(data = top_predictors, aes(x = MeanDecreaseGini, y =
reorder(PREDICTORS, MeanDecreaseGini))) +
  geom_bar(stat = "identity", fill = "blue") +
  ggtitle("Variable Importance for Consumption of Marijuana") +
  ylab("Variables") +
  xlab("Mean Decrease in ") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.caption = element_text(hjust = 0.5))
# Assuming df.train contains your training data
# Assuming marijuana_ever_used is your target variable

# Define the number of predictors
n_predictors <- ncol(df.train) - 1  # Exclude the target variable

# Use tuneRF to tune mtry
rf_model <- randomForest(marijuana_ever_used ~ ., mtry= n_predictors, data =
df.train)
tuned_rf <- tuneRF(x = df.train[, -which(names(df.train) ==
"marijuana_ever_used")],
                   y = df.train$marijuana_ever_used,
                   ntreeTry = 1000,  # Number of trees to grow
                              stepFactor = 1.5,   # Multiplicative factor to
increase/decrease mtry
                     improve = 0.05,  # Minimum improvement in node purity to
consider splitting
```

```
                          trace = TRUE, plot = TRUE)  # Print progress and plot mean
decrease accuracy

# Optimal mtry value

print(tuned_rf)
```

as it is shown the best mtry would be 3 since it is giving us the lowest

```
rf.binary<-randomForest(marijuana_ever_used~  .,  mtry=  3,ntree=1000,  data  =
df.train)

yhat.binary <- predict(rf.binary, newdata = df.test,type="class")

mean(yhat.binary ==df.test$marijuana_ever_used)

#random forest the best with a 90% accu
```

Boosting

```
#Boosting

boost.mrj <- gbm(marijuana_ever_used ~ ., data = df.train,

  distribution = "gaussian", n.trees = 1000,

  interaction.depth = 4, shrinkage = 0.2, verbose = F, cv =10)

yhat.boost <- predict(boost.mrj,

  newdata = df.test, n.trees = 1000, type = "response")

pred.gbm <- predict(boost.mrj,

  newdata = df.test, n.trees = 1000)

pred_direction <- ifelse(pred.gbm > 1.5, 1, 0)

table(pred_direction, df.test$marijuana_ever_used)
```

As it is clear the accuracy with the shirnkage of 0.2 is at 89%, in order to find the optimal shrinkage a loop is designed so that we can find the optimal shirnkage value that will give us the highest accuracy.

```
# Define shrinkage parameter values
shrinkage_vals <- seq(0.01, 0.2, by = 0.02)

# Initialize accuracy vector to store accuracy values
acc <- numeric(length(shrinkage_vals))
```

```
# Loop through shrinkage values
for (i in seq_along(shrinkage_vals)) {
  shrinkage <- shrinkage_vals[i]

  # Fit gradient boosting model
  gbm_model <- gbm(marijuana_ever_used ~ ., data = df.train,
                   distribution = "gaussian", n.trees = 1000,
                   shrinkage = shrinkage, interaction.depth = 4)
    # Compute test predictions
  test_preds <- predict(gbm_model, newdata = df.test, n.trees = 1000)
   # Binarize predictions based on threshold
 pred_direction <- ifelse(test_preds > 1.5, 1, 0)
   # Compute accuracy
  acc[i] <- mean(pred_direction == df.test$marijuana_ever_used)
}
# Print accuracy vector
print(acc)

# Plot accuracy vs shrinkage

plot(shrinkage_vals, acc, type = "b", xlab = "Shrinkage",

     ylab = "Accuracy", main = "Accuracy vs Shrinkage")

# Find shrinkage value with highest accuracy

best_shrinkage <- shrinkage_vals[which.max(acc)]

Best_shrinkage

# Add abline to indicate the shrinkage value with highest accuracy

abline(v = best_shrinkage, col = "red", lty = 2)
```

the optimal shrinkage value would be 0.01 with the accuracy of 91%

```
boost.binary<- gbm(marijuana_ever_used ~ ., data = df.train,

                   distribution = "gaussian", n.trees = 1000,

                   shrinkage = 0.01, interaction.depth = 4)

yhat.boost.binary <- predict(boost.binary,

    newdata = df.test, n.trees = 1000, type = "response")

pred.gbm <- predict(boost.mrj,

    newdata = df.test, n.trees = 1000)
```

```r
pred_binary <- ifelse(yhat.boost.binary > 1.5, 1, 0)

table(pred_binary, df.test$marijuana_ever_used)

mean(pred_binary == df.test$marijuana_ever_used)

#we can conclude that we get the best results with boosting

summary.gbm(boost.binary)

rf_err2 <- data.frame(Trees = 1:1000, Error = rf.binary$err.rate[,"OOB"], Type = "RF, m=3")

bag_err2 <- data.frame(Trees = 1:1000, Error = bag.model$err.rate[,"OOB"], Type = "BAG, m=13")

boost_err2 <- data.frame(Trees = 1:1000, Error = boost.binary$train.error, Type = "BOOST, λ=0.01")

# Combine all error data frames

model_err2 <- rbind(rf_err2, bag_err2, boost_err2)

# Plot the error rate for each model

ggplot(data=model_err2, aes(x=Trees, y=Error)) + geom_line(aes(color=Type)) + ggtitle("Methods vs
Errors") +xlim(0,1000)+theme(plot.title = element_text(hjust=0.5))
```

Multi Class

understanding the classes

```r
featureselection_multi<- randomForest(mrjydays~., data= data, ntree= 1000 , importance = TRUE)

important_variables1<-importance(featureselection_multi, type=1)

important_variables1[order(important_variables1, decreasing = TRUE), ]

df_multi<- data %>% select(c('FRDMEVR2', 'yflmjmo','mrjydays',
'iralcage','iralcfy','EDUSCHGRD2','frdmjmon','rlgfrnd','alcflag','PRLMTTV2', 'HEALTH2', 'NEWRACE2',
'rlgattd', 'POVERTY3','income','stndsmj', 'stndalc','COUTYP4', 'PRVDRGO2', 'tobflag', 'YFLTMRJ2'))

#splitting to train and test

set.seed(123)

train.index1<- sample(1:nrow(df_multi), nrow(df_multi)*0.7)

df_multi.train<- df_multi[train.index1,]

df_multi.test<- df_multi[-train.index1,]

#Model1_Desicion tree
```

```
set.seed(123)

tree_multi.model<-tree(mrjydays~. ,data= df_multi.train)

summary(tree_multi.model)

plot(tree_multi.model)

text(tree_multi.model, pretty = 0)

predict.tree_multi<- predict(tree_multi.model, newdata= df_multi.test, type= "class")

table(predict.tree_multi, df_multi.test$mrjydays)

mean((predict.tree_multi == df_multi.test$mrjydays))
```

There is about 88% accuracy next we will try to pruning to see if we can improve the accuracy

```
cv.mrjday <- cv.tree(tree_multi.model, FUN = prune.misclass)

names(cv.mrjday)

cv.mrjday

par(mfrow = c(1, 2))

plot(cv.mrjday$size, cv.mrjday$dev, type = "b")

plot(cv.mrjday$k, cv.mrjday$dev, type = "b")

prune.mrjday <- prune.misclass(tree_multi.model, best = 4)

plot(prune.mrjday)

text(prune.mrjday, pretty = 0)

tree_multi.pred.prune <- predict(prune.mrjday, df_multi.test,

    type = "class")

table(tree_multi.pred.prune, df_multi.test$mrjydays)

mean(tree_multi.pred.prune== df_multi.test$mrjydays)

#the pruning has not increase the accuracy
```

Model2_Bagging

```
#Bagging

all_predictors1<- length(df_multi.train)-1
```

```
bag.model_multi <- randomForest(mrjydays ~ ., data = df_multi.train, mtry =all_predictors1 ,ntree= 1000, importance = TRUE)

bag.model_multi

yhat_multi.bag <- predict(bag.model_multi, newdata = df_multi.test,type="class")

mean(yhat_multi.bag == df_multi.test$mrjydays)


table(yhat_multi.bag, df_multi.test$mrjydays)

#there is 87% accuracy with bagging which is less than when we were using the desicion tree

importance(bag.model_multi)

varImpPlot(bag.model_multi)

# Get variable importance

importance_df_multi <- as.data.frame(importance(bag.model_multi))

importance_df_multi <- importance_df_multi[order(-importance_df$MeanDecreaseAccuracy), ]

top_predictors_multi <- head(importance_df_multi, 20)

top_predictors_multi <- top_predictors_multi[, c("MeanDecreaseAccuracy", "MeanDecreaseGini")]

top_predictors_multi$PREDICTORS <- rownames(top_predictors_multi)

#with decision tree-based models and want to understand the importance of variables in creating informative splits, MDG may provide more insights.

ggplot(data = top_predictors_multi, aes(x = MeanDecreaseGini, y = reorder(PREDICTORS, MeanDecreaseGini))) +

 geom_bar(stat = "identity", fill = "blue") +

 ggtitle("Variable Importance for Consumption of Marijuana") +

 ylab("Variables") +

 xlab("Mean Decrease in ") +

 theme(plot.title = element_text(hjust = 0.5),

       plot.caption = element_text(hjust = 0.5))

n_predictors_multi <- ncol(df_multi.train) - 1  # Exclude the target variable
```

```r
# Use tuneRF to tune mtry

rf_model1 <- randomForest(mrjydays ~ ., mtry= n_predictors_multi, data = df_multi.train)

tuned_rf1 <- tuneRF(x = df_multi.train[, -which(names(df_multi.train) == "mrjydays")],

                    y = df_multi.train$mrjydays,

                    ntreeTry = 1000,  # Number of trees to grow

                    stepFactor = 1.5,  # Multiplicative factor to increase/decrease mtry

                    improve = 0.05,  # Minimum improvement in node purity to consider splitting

                    trace = TRUE, plot = TRUE)  # Print progress and plot mean decrease accuracy

# Optimal mtry value

print(tuned_rf1)


rf.model2 <- randomForest(mrjydays ~ ., data = df_multi.train, mtry = 3,ntree=1000,importance = TRUE)

yhat.rf <- predict(rf.model2, newdata = df_multi.test,type="class")


table(yhat.rf,df_multi.test$mrjydays)

mean(yhat.rf==df_multi.test$mrjydays)


boost1<-gbm(mrjydays ~ ., data = df_multi.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.01 , interaction.depth = 4)

boost2<-gbm(mrjydays ~ ., data = df_multi.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.05 , interaction.depth = 4)

boost3<-gbm(mrjydays ~ ., data = df_multi.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.1 , interaction.depth = 4)

boost4<-gbm(mrjydays ~ ., data = df_multi.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.15 , interaction.depth = 4)

boost5<-gbm(mrjydays ~ ., data = df_multi.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.2 , interaction.depth = 4)
```

```r
boost1_err <- data.frame(Trees = 1:1000, Error = boost1$train.error, Type = "Boost1, λ=0.01")

boost2_err <- data.frame(Trees = 1:1000, Error = boost2$train.error, Type = "Boost2, λ=0.05")

boost3_err <- data.frame(Trees = 1:1000, Error = boost3$train.error, Type = "Boost3, λ=0.1")

boost4_err <- data.frame(Trees = 1:1000, Error = boost4$train.error, Type = "Boost4, λ=0.15")

boost5_err <- data.frame(Trees = 1:1000, Error = boost5$train.error, Type = "Boost5, λ=0.2")

# Combine all error data frames

model_err <- rbind(boost1_err, boost2_err, boost3_err,boost4_err,boost5_err)


# Plot the error rate for each model

ggplot(data=model_err, aes(x=Trees, y=Error)) +

  geom_line(aes(color=Type)) + ggtitle(" Train Error vs Number of Trees") +

  xlim(0,200)+theme(plot.title = element_text(hjust=0.5))

boost.multi<- gbm(mrjydays ~ ., data = df_multi.train,

                   distribution = "gaussian", n.trees = 1000,

                   shrinkage = 0.2, interaction.depth = 4)

boost.pred <- predict(boost.multi, newdata = df_multi.test, n.trees = 1000)


  classify <- function(x) {

  cut(x, breaks = c(-Inf, 0.5, 1.5, 2.5, 3.5, Inf), labels = c(0, 1, 2, 3, 4))

}

# Apply the classification function to predictions

class_test <- sapply(boost.pred, classify)

table(class_test , df_multi.test$mrjydays)

# Make sure both factors have the same levels

class_test <- factor(class_test, levels = levels(df_multi.test$mrjydays))


# Compute accuracy
```

```
accuracy <- mean(class_test == df_multi.test$mrjydays)

accuracy

Create data frames for each model's error

rf_err1 <- data.frame(Trees = 1:1000, Error = rf.model2$err.rate[,"OOB"], Type = "RF, m=3")

bag_err1 <- data.frame(Trees = 1:1000, Error = bag.model_multi$err.rate[,"OOB"], Type = "BAG, m=20")

boost_err1 <- data.frame(Trees = 1:1000, Error = boost.multi$train.error, Type = "BOOST, λ=0.2")


# Combine all error data frames

model_err1 <- rbind(rf_err1, bag_err1, boost_err1)


# Plot the error rate for each model

ggplot(data=model_err1, aes(x=Trees, y=Error)) + geom_line(aes(color=Type)) + ggtitle("Methods vs
Errors") +xlim(0,1000)+theme(plot.title = element_text(hjust=0.5))

featureselection_reg<- randomForest(irmjfm~., data= data, ntree= 1000 , importance = TRUE)

important_variables2<-importance(featureselection_reg, type=1)

important_variables2[order(important_variables2, decreasing = TRUE), ]


#now we subset the variables we want to select for the analysis.

df.reg<- data %>% select(c('irmjfy','EDUSCHGRD2','yflmjmo','rlgfrnd', 'YFLTMRJ2','iralcage',
'FRDMEVR2','alcflag', 'iralcfy', 'frdmjmon', 'rlgimpt', 'PRMJEVR2', 'stndsmj', 'rlgattd', 'tobflag', 'rlgdcsn',
'iralcfm', 'argupar', 'HEALTH2'))

new_names <- c(

    "marijuana_freq_yearly", "teen_grade", "teen_feels_peers_marijuana_monthly", "religious_belif",
"peers_try_marijuana",

  "first_alchohol_usage", "close_friend_teen_mrijuana", "alchohol_ever_used", "alchohol_freq_yearly",
"close_youth_marijuana_mon",

          "religious_imoprtance", "parents_feel_marijuana_try", "student_y_grade_marijuan",
"num_religious_attendance", "tobaco_ever_used",
  "religious_influance", "alchohol_freq_mon", "num_argu_prarent", "overall_health"

)


# Rename variables

df.reg <- df.reg %>% rename(!!!setNames(names(.), new_names))


set.seed(123)

train.index2<- sample(1:nrow(df.reg), nrow(df.reg)*0.7)

df.reg.train<- df.reg[train.index2,]

df.reg.test<- df.reg[-train.index2,]

#Model1_Desicion tree

set.seed(123)

tree.reg.model<-tree(marijuana_freq_yearly~. ,data= df.reg.train)

summary(tree.reg.model)

plot(tree.reg.model)

text(tree.reg.model, pretty = 0)

predict.tree.reg<- predict(tree.reg.model, newdata= df.reg.test)

mean((predict.tree.reg -df.reg.test$marijuana_freq_yearly)^2)

cv.reg <- cv.tree(tree.reg.model)

names(cv.reg)

cv.reg

par(mfrow = c(1, 2))

plot(cv.reg$size, cv.reg$dev, type = "b")

plot(cv.reg$k, cv.reg$dev, type = "b")

prune.reg <- prune.tree(tree.reg.model, best = 4)

plot(prune.reg)

text(prune.reg, pretty = 0)
```

```
tree.reg.pred.prune <- predict(prune.reg, df.reg.test)

mean((tree.reg.pred.prune- df.reg.test$marijuana_freq_yearly)^2)

#we can see that pruning has made an increase in the test mse

#Bagging

all_predictors2<- length(df.reg.train)-1

bag.model.reg  <-  randomForest(marijuana_freq_yearly  ~  .,  data  =  df.reg.train,  mtry  =all_predictors2
,ntree= 1000, importance = TRUE)

bag.model.reg

yhat.reg.bag <- predict(bag.model.reg, newdata = df.reg.test)

mean((yhat.reg.bag- df.reg.test$marijuana_freq_yearly)^2)

#the mse with boosting is 1363

importance(bag.model.reg)

varImpPlot(bag.model.reg)

# Define the values for mtry

mtry_values <- c(1, 5, 9,13, 18)

ntrees <- 1000

# Create empty vectors to store error rates and models

error_rates <- numeric()

rf_models <- list()

# Loop through each mtry value

for (mtry_val in mtry_values) {

 # Fit a random forest model

  rf_model <- randomForest(marijuana_freq_yearly ~ ., data = df.reg.train, mtry = mtry_val, ntree = ntrees,
importance = TRUE)

  # Store the model

 rf_models[[as.character(mtry_val)]] <- rf_model
```

```
  # Store the error rate
 error_rates <- c(error_rates, rf_model$mse)
}


# Combine error rates and mtry values into a data frame
model_err <- data.frame(Mtry = rep(mtry_values, each = ntrees),
                        Trees = rep(1:ntrees, length(mtry_values)),
                        Error = error_rates)


# Plot the error rate for each model
ggplot(data = model_err, aes(x = Trees, y = Error, color = factor(Mtry))) +
 geom_line() +
 ggtitle("Error vs Number of Trees") +
 xlim(0, 100) +
 theme(plot.title = element_text(hjust = 0.5))
rf.reg <- randomForest(marijuana_freq_yearly ~ ., data = df.reg.train, mtry =5 ,ntree= 1000, importance =
TRUE)
rf.reg
yhat.rf <- predict(rf.reg,newdata= df.reg.test)
mean((yhat.rf - df.reg.test$marijuana_freq_yearly)^2)


#comparing the random forest with the two other models seem to give us a better mse
importance_df.reg <- as.data.frame(importance(rf.reg))


importance_df.reg <- importance_df.reg[order(-importance_df.reg$`%IncMSE`), ]
top_predictors.reg <- head(importance_df.reg, 20)
```

```r
# Correcting the column names to match the ones in the dataframetop_predictors.reg <-
top_predictors.reg[, c("%IncMSE", "IncNodePurity")]

top_predictors.reg$PREDICTORS <- rownames(top_predictors.reg)

# Plotting the most important variables

ggplot(data = top_predictors.reg, aes(x = `%IncMSE`, y = reorder(PREDICTORS, `%IncMSE`))) +

  geom_bar(stat = "identity", fill = "blue") +

  ggtitle("Variable Importance for Consumption of Marijuana") +

  ylab("Variables") +

  xlab("Mean Decrease in MSE") +

  theme(plot.title = element_text(hjust = 0.5),

        plot.caption = element_text(hjust = 0.5))

#Boosting

# plotting the training error for each number of trees (up to 1000) for each shrinkage value.

# Define the values for shrinkage

shrinkage_vals <- seq(0.01, 0.2, by = 0.02)

ntrees <- 1000

# Create empty vectors to store error rates and models

error_rates <- numeric()

gbm_models <- list()

# Loop through each shrinkage value

for (shrinkage_val in shrinkage_vals) {

  # Fit a gradient boosting model

  gbm_reg <- gbm(marijuana_freq_yearly ~ ., data = df.reg.train, distribution = "gaussian",

                   n.trees = ntrees, shrinkage = shrinkage_val)

   # Store the model

  gbm_models[[as.character(shrinkage_val)]] <- gbm_reg

   # Compute the predictions
  yhat <- predict(gbm_reg, newdata = df.reg.test, n.trees = ntrees)

   # Compute the MSE

  mse <- mean((yhat - df.reg.test$marijuana_freq_yearly)^2)

   # Store the error rate

  error_rates <- c(error_rates, gbm_reg$train.error)

}

# Combine error rates and shrinkage values into a data frame

model_err <- data.frame(Shrinkage = rep(shrinkage_vals, each = ntrees),

                        Trees = rep(1:ntrees, length(shrinkage_vals)),

                        Error = error_rates)



# Plot the error rate for each model

ggplot(data = model_err, aes(x = Trees, y = Error, color = factor(Shrinkage))) +

  geom_line() +

  ggtitle("Error vs Number of Trees") +

  xlim(0, 100) +

  theme(plot.title = element_text(hjust = 0.5))

boost.reg<-  gbm(marijuana_freq_yearly  ~.  ,  data  =  df.reg.train,distribution  =  "gaussian",  n.trees  =
1000,shrinkage =0.19)

boost.pred.reg <- predict(boost.reg,df.reg.test,n.trees = 1000)

mean((boost.pred.reg- df.reg.test$marijuana_freq_yearly)^2)

Create data frames for each model's error

rf_err <- data.frame(Trees = 1:1000, Error = rf.reg$mse, Type = "RF, m=5")

bag_err <- data.frame(Trees = 1:1000, Error = bag.model.reg$mse, Type = "BAG, m=17")

boost_err <- data.frame(Trees = 1:1000, Error = boost.reg$train.error, Type = "BOOST, λ=0.19")

# Combine all error data frames

model_err <- rbind(rf_err, bag_err, boost_err)
```

```r
# Plot the error rate for each model

ggplot(data = model_err, aes(x = Trees, y = Error, color = Type)) +

  geom_line() +

  ggtitle("Error vs Number of Trees") +

  xlim(0, 1000) +

  ylim(1000, 2500) +

  theme(plot.title = element_text(hjust = 0.5)) +

  guides(color = guide_legend(title = "Methods, Parameters"))
```