Psychedelic Inclusion for Improved Health

Deborah Young

2023-08-08

Data Exploration and Cleaning:

```
nrow(data)
```

```
## [1] 605
```

```
ncol(data)
```

```
## [1] 260
```

```
duplicated_rows <- duplicated(data)
unique_data <- data[!duplicated_rows, ]
nrow(unique_data)
```

```
## [1] 605
```

```
# Calculate the sum of NA values per column
na_sum_per_column <- colSums(is.na(data))
#print(na_sum_per_column)

na_sum <- sum(is.na(data))
#print(na_sum)

missing_percentage <- colMeans(is.na(data)) * 100
#print(missing_percentage)
```

```
# Calculate the percentage of missing data for each column
missing_percentage <- colMeans(is.na(data)) * 100

# Identify columns with missing data greater than 50%
columns_to_keep <- missing_percentage <= 5

# Subset the data frame to keep only the columns with missing data less than or equal to 5%
data_filtered <- data[, columns_to_keep]
nrow(data_filtered)
```

```
## [1] 605
```

```r
ncol(data_filtered)
```

```
## [1] 199
```

```r
# Print the filtered data frame
#head(data_filtered)
```

CREATE DATAFRAME FOR ONLY CATEGORICAL DATA

```r
# Use apply() to check for repeating values in each column
columns_with_duplicates <- apply(data_filtered, 2, function(x) any(duplicated(x)))

# Alternatively, you can use sapply() for a simplified output
columns_with_duplicates <- sapply(data_filtered, function(x) any(duplicated(x)))

# Get the names of columns with repeating values
columns_with_repeats <- names(data_filtered)[columns_with_duplicates]

# Print the columns with repeating values
#print(columns_with_repeats)

data2 <- data_filtered %>%
  dplyr::select(dplyr::all_of(columns_with_repeats))
```

```r
categorical_data <- data2 %>%
  dplyr::select(-StartDate, -EndDate, -`Duration (in seconds)`, -Q7)

#head(categorical_data)
```

VIEW CATEGORICAL VARIABLES

CREATE NEW VARIABLE FOR CLEANED DATA

```r
#drop columns without ordinal data
data <- categorical_data %>% dplyr::select(-c("Q4", "Q6", "Q8", "Q9", "Q11", "Q12", "Q13", "Use_locatio
```

HOLDING DATA HERE:

```r
preserved_data <- data
```

Modeling:

MODELING FOR ALL CLASSES OF TARGET VARIABLE

```
table(data$Q88)
```

```
##
##         1-2%       11-25%       26-50%         3-5%       51-75%         6-10%
##            3           11           28            4          122             3
##      76-100% Less than 1%    None (0%)
##          429            2            1
```

Going to change "None (0%)" to "Less than 1%" so that I don't have any classes with only one value for splitting later on.

```
# Replacing "None (0%)" with "Less than 1%" in the target column "Q88"
data$Q88[data$Q88 == "None (0%)"] <- "Less than 1%"
```

```
table(data$Q88)
```

```
##
##         1-2%       11-25%       26-50%         3-5%       51-75%         6-10%
##            3           11           28            4          122             3
##      76-100% Less than 1%
##          429            3
```

Convert data to **factor** before one hot encoding:

```
table(target_variable)
```

```
## target_variable
##         1-2%       11-25%       26-50%         3-5%       51-75%         6-10%
##            3           11           28            4          122             3
##      76-100% Less than 1%
##          429            3
```

```
#convert columns to factors

# Loop through the columns of the data
for (col_name in names(feature_data)) {
  # Extract the unique responses for the column
  unique_res <- unique(feature_data[[col_name]])

  # If the number of unique responses is less than or equal to 10
  if (length(unique_res) <= 10) {
    # Convert the column to an ordered factor using the unique responses as levels
    feature_data[[col_name]] <- factor(feature_data[[col_name]], levels = unique_res, ordered = TRUE)
  }
}

# Now, 'feature_data' is updated so that the relevant columns are ordered factors
```

FEATURE ENGINEERING: ONE-HOT ENCODING

```
data_onehot <- dummyVars("~ .", data = feature_data)
data_encoded <- data.frame(predict(data_onehot, newdata = feature_data))
```

```
final_data <- data.frame(target = target_variable, data_encoded)
```

FEATURE SELECTION

```
# Identify rows with any missing values
rows_with_na <- rowSums(is.na(final_data)) > 0

# Subset data to view only rows with missing values
missing_data_rows <- final_data[rows_with_na,]

# Print or view the rows with missing values
#print(missing_data_rows)
```

```
#ommitting NAs because there are only 4 records with them and two of them appear to be NA all across.
final_data <- na.omit(final_data)
```

Tried RFE but it was too computationally intensive.

```
#library(caret)
#control <- rfeControl(functions=rfFuncs, method="cv", number=10)
#results <- rfe(final_data[, -which(names(final_data) == "target")], final_data$target, sizes=c(1:ncol(
#selected_features <- predictors(results)
```

Stepwise Regression Try #1 - Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC' cannot proceed

```
#full_model <- lm(target ~ ., data = final_data)

#library(MASS)
#stepwise_model <- stepAIC(full_model, direction="both") # full_model is the initial model with all fea
#selected_features <- names(coef(stepwise_model))
#final_data <- final_data[, c("target", selected_features)]
```

Stepwise Regression Try #2 - Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC' cannot proceed

```
#library(MASS)

# Fit the full model with all predictors
#full_model <- lm(target ~ ., data = final_data)

# Apply stepwise selection
#stepwise_model <- stepAIC(full_model, direction = "both")

# The final selected model
#final_model <- lm(formula(stepwise_model), data = final_data)
```

CORRELATION MATRIX FOR FEATURE SELECTION

```
correlation_matrix <- cor(final_data[, -which(names(final_data) == "target")])
highly_correlated <- findCorrelation(correlation_matrix, cutoff = 0.75)
final_data <- final_data[, -highly_correlated]
```

BEGIN MODELING

Split Data

```
set.seed(123)
splitIndex <- createDataPartition(final_data$target, p = .8, list = FALSE)
train_data <- final_data[splitIndex,]
test_data <- final_data[-splitIndex,]
```

Tried **linear model** but I knew it wasn't right, I just wanted to see and this is the warning I received: "Warning: using type ="numeric" with a factor response will be ignoredWarning: '-' not meaningful for factors"

```
#linear_model <- lm(target ~ ., data = train_data)
```

SVM MODEL

```
svm_model <- svm(target ~ ., data=train_data)
summary(svm_model)
```

```
##
## Call:
## svm(formula = target ~ ., data = train_data)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  458
##
##  ( 23 98 316 9 4 3 3 2 )
##
##
## Number of Classes:  8
##
## Levels:
##  1-2% 11-25% 26-50% 3-5% 51-75% 6-10% 76-100% Less than 1%
```

Evaluate SVM Model

```
predictions <- predict(svm_model, newdata = test_data)
```

```r
conf_matrix <- confusionMatrix(predictions, test_data$target)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     1-2% 11-25% 26-50% 3-5% 51-75% 6-10% 76-100% Less than 1%
##   1-2%            0      0      0    0      0     0       0             0
##   11-25%          0      0      0    0      0     0       0             0
##   26-50%          0      0      0    0      0     0       0             0
##   3-5%            0      0      0    0      0     0       0             0
##   51-75%          0      0      0    0      0     0       0             0
##   6-10%           0      0      0    0      0     0       0             0
##   76-100%         0      2      5    0     24     0      85             0
##   Less than 1%    0      0      0    0      0     0       0             0
##
## Overall Statistics
##
##                Accuracy : 0.7328
##                  95% CI : (0.6426, 0.8107)
##     No Information Rate : 0.7328
##     P-Value [Acc > NIR] : 0.5481
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1-2% Class: 11-25% Class: 26-50% Class: 3-5%
## Sensitivity                   NA       0.00000        0.0000          NA
## Specificity                    1       1.00000        1.0000           1
## Pos Pred Value                NA           NaN           NaN          NA
## Neg Pred Value                NA       0.98276        0.9569          NA
## Prevalence                     0       0.01724        0.0431           0
## Detection Rate                 0       0.00000        0.0000           0
## Detection Prevalence           0       0.00000        0.0000           0
## Balanced Accuracy             NA       0.50000        0.5000          NA
##                      Class: 51-75% Class: 6-10% Class: 76-100%
## Sensitivity                 0.0000           NA         1.0000
## Specificity                 1.0000            1         0.0000
## Pos Pred Value                 NaN           NA         0.7328
## Neg Pred Value              0.7931           NA            NaN
## Prevalence                  0.2069            0         0.7328
## Detection Rate              0.0000            0         0.7328
## Detection Prevalence        0.0000            0         1.0000
## Balanced Accuracy           0.5000           NA         0.5000
##                      Class: Less than 1%
## Sensitivity                           NA
## Specificity                            1
## Pos Pred Value                        NA
## Neg Pred Value                        NA
## Prevalence                             0
```

```
## Detection Rate                          0
## Detection Prevalence                    0
## Balanced Accuracy                      NA
```

```
accuracy <- sum(diag(conf_matrix$table)) / sum(conf_matrix$table)
print(paste("Accuracy:", round(accuracy, 3)))
```

```
## [1] "Accuracy: 0.733"
```

RANDOM FOREST MODEL

```
set.seed(123)
splitIndex <- createDataPartition(final_data$target, p = .8, list = FALSE)
train_data <- final_data[splitIndex,]
test_data <- final_data[-splitIndex,]
```

```
rf_model <- randomForest(target ~ ., data = train_data)
```

```
predictions <- predict(rf_model, test_data)
confusionMatrix(predictions, test_data$target)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     1-2% 11-25% 26-50% 3-5% 51-75% 6-10% 76-100% Less than 1%
##   1-2%            0      0      0    0      0     0       0            0
##   11-25%          0      0      0    0      0     0       0            0
##   26-50%          0      0      0    0      0     0       0            0
##   3-5%            0      0      0    0      0     0       0            0
##   51-75%          0      0      0    0      2     0       0            0
##   6-10%           0      0      0    0      0     0       0            0
##   76-100%         0      2      5    0     22     0      85            0
##   Less than 1%    0      0      0    0      0     0       0            0
##
## Overall Statistics
##
##                Accuracy : 0.75
##                  95% CI : (0.6611, 0.8257)
##     No Information Rate : 0.7328
##     P-Value [Acc > NIR] : 0.3822
##
##                   Kappa : 0.0952
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1-2% Class: 11-25% Class: 26-50% Class: 3-5%
## Sensitivity                   NA       0.00000        0.0000          NA
## Specificity                    1       1.00000        1.0000           1
```

```
## Pos Pred Value                 NA           NaN           NaN          NA
## Neg Pred Value                 NA       0.98276        0.9569          NA
## Prevalence                      0       0.01724        0.0431           0
## Detection Rate                  0       0.00000        0.0000           0
## Detection Prevalence            0       0.00000        0.0000           0
## Balanced Accuracy              NA       0.50000        0.5000          NA
##                       Class: 51-75% Class: 6-10% Class: 76-100%
## Sensitivity                 0.08333           NA        1.00000
## Specificity                 1.00000            1        0.06452
## Pos Pred Value              1.00000           NA        0.74561
## Neg Pred Value              0.80702           NA        1.00000
## Prevalence                  0.20690            0        0.73276
## Detection Rate              0.01724            0        0.73276
## Detection Prevalence        0.01724            0        0.98276
## Balanced Accuracy           0.54167           NA        0.53226
##                       Class: Less than 1%
## Sensitivity                            NA
## Specificity                             1
## Pos Pred Value                         NA
## Neg Pred Value                         NA
## Prevalence                              0
## Detection Rate                          0
## Detection Prevalence                    0
## Balanced Accuracy                      NA
```

MODELING FOR SPECIFIED BINARY TARGET VARIABLE

```r
table(preserved_data$Q88)
```

```
##
##          1-2%        11-25%        26-50%          3-5%        51-75%          6-10%
##             3            11            28             4           122              3
##      76-100% Less than 1%    None (0%)
##           429             2             1
```

TRYING BINARY

```r
# Creating a binary target variable for whether Q88 is in the 76-100% range
preserved_data$target_variable <- ifelse(preserved_data$Q88 == "76-100%", 1, 0)

# Checking the table for the new target variable
table(preserved_data$target_variable)
```

```
##
##   0   1
## 174 429
```

```r
#separate target variable and convert to factor
target_variable <- as.factor(preserved_data$target_variable)

#Remove the Target Variable from the Data
feature_data <- preserved_data[, -which(names(preserved_data) == "target_variable")]
```

```r
table(target_variable)
```

```
## target_variable
##   0   1
## 174 429
```

```r
#convert columns to factors

# Loop through the columns of the data
for (col_name in names(feature_data)) {
  # Extract the unique responses for the column, assuming they are in the desired order
  unique_res <- unique(feature_data[[col_name]])

  # If the number of unique responses is less than or equal to 10
  if (length(unique_res) <= 10) {
    # Convert the column to an ordered factor using the unique responses as levels
    feature_data[[col_name]] <- factor(feature_data[[col_name]], levels = unique_res, ordered = TRUE)
  }
}

# Now, 'feature_data' is updated so that the relevant columns are ordered factors
```

FEATURE ENGINEERING: ONE-HOT ENCODING

```r
data_onehot <- dummyVars("~ .", data = feature_data)
data_encoded <- data.frame(predict(data_onehot, newdata = feature_data))
```

```r
final_data <- data.frame(target = target_variable, data_encoded)
```

FEATURE SELECTION

```r
# Identify rows with any missing values
rows_with_na <- rowSums(is.na(final_data)) > 0

# Subset data to view only rows with missing values
missing_data_rows <- final_data[rows_with_na,]

# Print or view the rows with missing values
#print(missing_data_rows)
```

```r
#ommitting NAs because they can't be present for most models and there are only 4 records with them and
final_data <- na.omit(final_data)
```

STARTEDITING HERE

Tried RFE but it was too computationally intensive.

```r
#library(caret)
#control <- rfeControl(functions=rfFuncs, method="cv", number=10)
#results <- rfe(final_data[, -which(names(final_data) == "target")], final_data$target, sizes=c(1:ncol(
#selected_features <- predictors(results)
```

Stepwise Regression Try #1 - Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC' cannot proceed

```
#error: Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC

#full_model <- lm(target ~ ., data = final_data)

#library(MASS)
#stepwise_model <- stepAIC(full_model, direction="both") # full_model is the initial model with all fea
#selected_features <- names(coef(stepwise_model))
#final_data <- final_data[, c("target", selected_features)]
```

Stepwise Regression Try #2 - Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC' cannot proceed

```
#error: Error in stepAIC(full_model, direction = "both") : AIC is -infinity for this model, so 'stepAIC

#library(MASS)

# Fit the full model with all predictors
#full_model <- lm(target ~ ., data = final_data)

# Apply stepwise selection
#stepwise_model <- stepAIC(full_model, direction = "both")

# The final selected model
#final_model <- lm(formula(stepwise_model), data = final_data)
```

CORRELATION MATRIX FOR FEATURE SELECTION

```
correlation_matrix <- cor(final_data[, -which(names(final_data) == "target")])
highly_correlated <- findCorrelation(correlation_matrix, cutoff = 0.75)
final_data <- final_data[, -highly_correlated]
```

Split data

```
set.seed(123)
splitIndex <- createDataPartition(final_data$target, p = .8, list = FALSE)
train_data <- final_data[splitIndex,]
test_data <- final_data[-splitIndex,]
```

Tried linear model but I knew it wasn't right, I just wanted to see and this is the warning I received: "Warning: using type ="numeric" with a factor response will be ignoredWarning: '-' not meaningful for factors"

```
#linear_model <- lm(target ~ ., data = train_data)
```

SVM MODEL

```r
svm_model <- svm(target ~ ., data=train_data)
summary(svm_model)
```

```
##
## Call:
## svm(formula = target ~ ., data = train_data)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  356
##
##  ( 139 217 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Evaluate SVM Model

```r
predictions <- predict(svm_model, newdata = test_data)
```

```r
conf_matrix <- confusionMatrix(predictions, test_data$target)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  0  0
##          1 34 85
##
##                Accuracy : 0.7143
##                  95% CI : (0.6243, 0.7933)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 0.5461
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.519e-08
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.7143
##              Prevalence : 0.2857
```

```
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
accuracy <- sum(diag(conf_matrix$table)) / sum(conf_matrix$table)
print(paste("Accuracy:", round(accuracy, 3)))
```

```
## [1] "Accuracy: 0.714"
```

KNN MODEL

```
set.seed(123)
splitIndex <- createDataPartition(final_data$target, p = .8, list = FALSE)
train_data <- final_data[splitIndex,]
test_data <- final_data[-splitIndex,]
```

```
knn_model <- knn(train_data[, -which(names(train_data) == "target")], test_data[, -which(names(test_data
```

```
confusion <- confusionMatrix(test_data$target, knn_model)
print(confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 29
##          1  5 80
##
##                Accuracy : 0.7143
##                  95% CI : (0.6243, 0.7933)
##     No Information Rate : 0.916
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1119
##
##  Mcnemar's Test P-Value : 7.998e-05
##
##             Sensitivity : 0.50000
##             Specificity : 0.73394
##          Pos Pred Value : 0.14706
##          Neg Pred Value : 0.94118
##              Prevalence : 0.08403
##          Detection Rate : 0.04202
##    Detection Prevalence : 0.28571
##       Balanced Accuracy : 0.61697
```

```
##
##        'Positive' Class : 0
##
```

```
accuracy <- sum(test_data$target == knn_model) / length(test_data$target)
print(paste("Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy: 71.43 %"
```

```
print(confusion$byClass)
```

```
##          Sensitivity          Specificity       Pos Pred Value
##           0.50000000           0.73394495           0.14705882
##       Neg Pred Value            Precision               Recall
##           0.94117647           0.14705882           0.50000000
##                   F1           Prevalence       Detection Rate
##           0.22727273           0.08403361           0.04201681
## Detection Prevalence    Balanced Accuracy
##           0.28571429           0.61697248
```

```
roc_obj <- roc(test_data$target, as.numeric(knn_model))
```

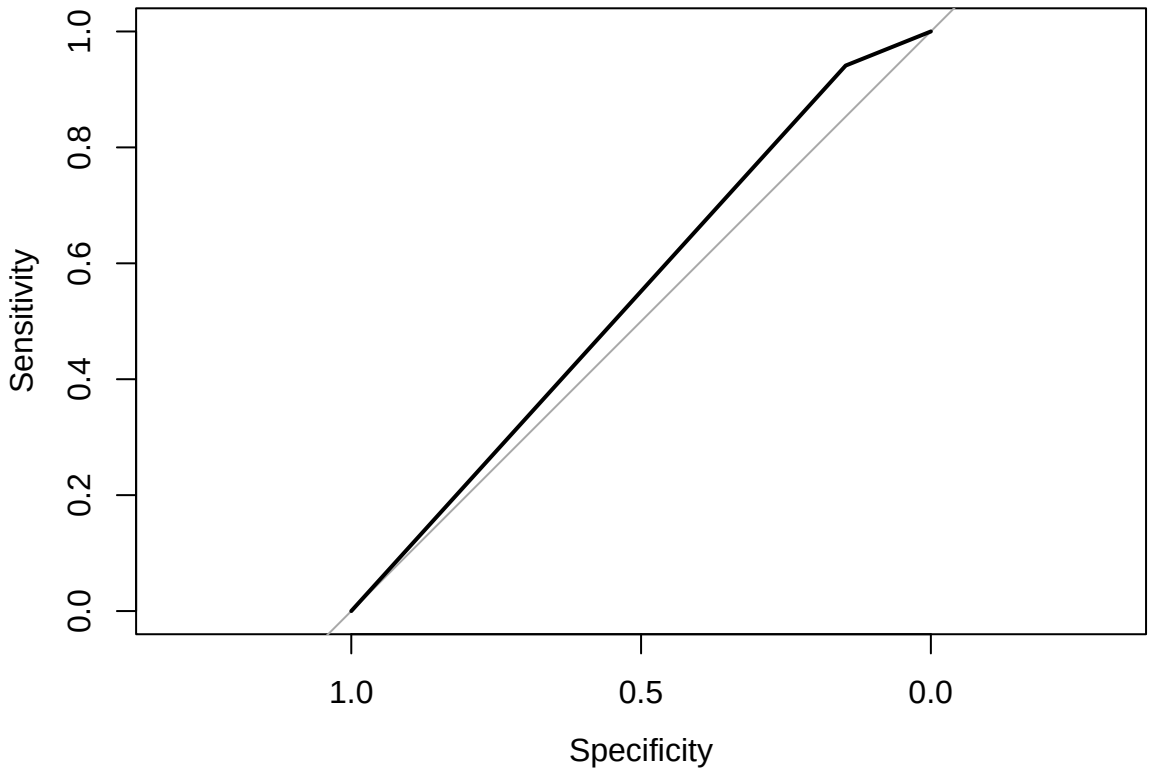
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc(roc_obj)
```

```
## Area under the curve: 0.5441
```

```
plot(roc_obj)
```



RANDOMFOREST MODEL

```
set.seed(123)
splitIndex <- createDataPartition(final_data$target, p = .8, list = FALSE)
train_data <- final_data[splitIndex,]
test_data <- final_data[-splitIndex,]
```

```
rf_model <- randomForest(target ~ ., data = train_data)
```

```
predictions <- predict(rf_model, test_data)
confusionMatrix(predictions, test_data$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32  0
##          1  2 85
##
##                Accuracy : 0.9832
##                  95% CI : (0.9406, 0.998)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 4.783e-15
##
```

```
##                   Kappa : 0.9581
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9412
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9770
##              Prevalence : 0.2857
##          Detection Rate : 0.2689
##    Detection Prevalence : 0.2689
##       Balanced Accuracy : 0.9706
##
##        'Positive' Class : 0
##
```

```
roc_obj <- roc(test_data$target, as.numeric(predictions))
```

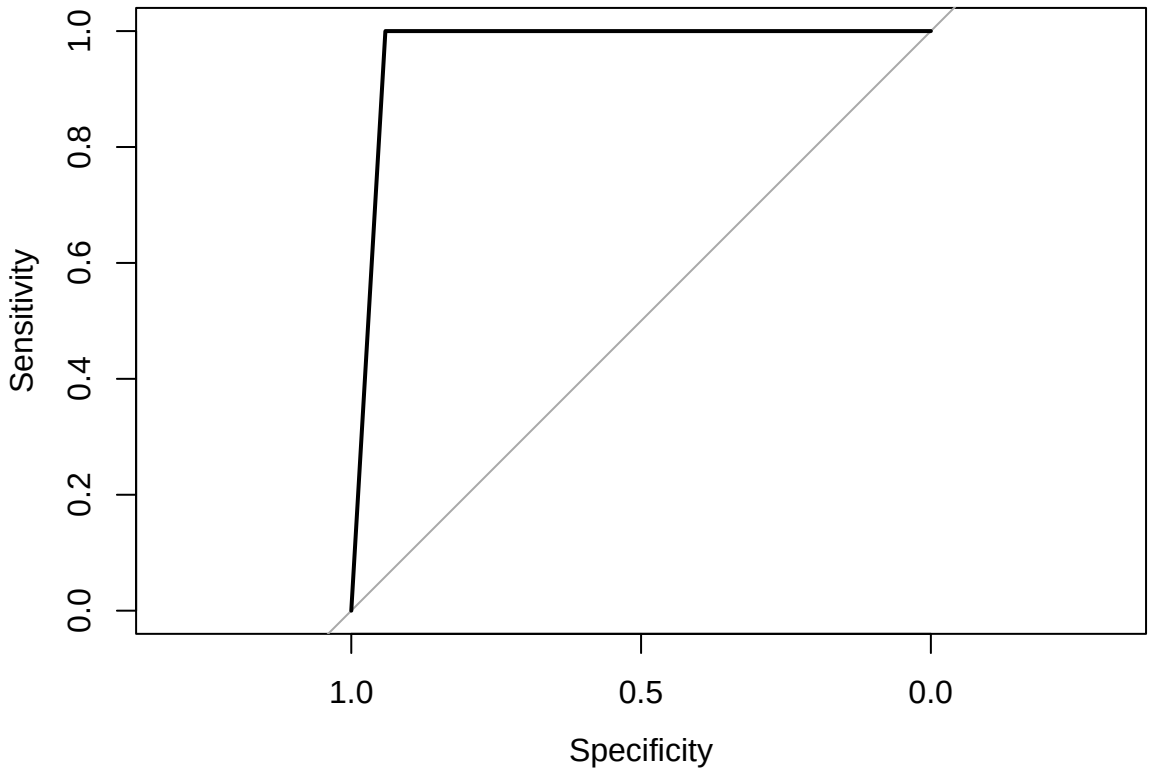
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc(roc_obj)
```

```
## Area under the curve: 0.9706
```

```
plot.roc(roc_obj)
```



GBM Model did not function properly.

WORD CLOUDS:

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

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

Q91: "What do you believe are psychedelics most important contribution to society? (optional)" - Participants were then presented with a text box to answer

```
df <- read_excel("/Users/debane/Documents/MS Data Science/630 Predictive Analytics/Term Project/An Explo

head(df)
```

```
## # A tibble: 6 x 260
##   `Interview Order` StartDate           EndDate
##               <dbl> <dttm>              <dttm>
## 1                NA 2020-11-02 16:04:10 2020-11-02 16:28:08
## 2                NA 2020-11-03 03:51:32 2020-11-03 04:18:14
## 3                NA 2020-11-03 06:11:37 2020-11-03 06:47:50
## 4                NA 2020-11-03 09:08:34 2020-11-03 10:08:49
## 5                NA 2020-11-03 15:34:18 2020-11-04 09:22:06
## 6                NA 2020-11-04 03:35:40 2020-11-04 04:58:42
## # i 257 more variables: `Duration (in seconds)` <dbl>, Q4 <chr>, Q5 <chr>,
## #   Q6 <chr>, Q7 <dbl>, Q8 <chr>, Q9 <chr>, Q10 <chr>, Q11 <chr>,
## #   Q11_8_TEXT <chr>, Q12 <chr>, Q13 <chr>, Q13_12_TEXT <chr>,
## #   Q13_13_TEXT <chr>, Q14 <chr>, Q15 <chr>, Q16 <chr>, Q17 <chr>, Q18 <dbl>,
## #   Q19 <dbl>, Q20 <chr>, Shroom_1Year <chr>, LSD_1Year <chr>, DMT_1Year <chr>,
## #   `5-MeO_1Year` <chr>, Aya_1Year <chr>, Mescal_1Year <chr>,
## #   Iboga_1Year <chr>, RC_1Year <chr>, Salvia_1Year <chr>, ...
```

```
text_data <- paste(df$Q91, collapse = " ")
```

```
library(tm)
text_data <- tolower(text_data)
text_data <- removePunctuation(text_data)
text_data <- removeNumbers(text_data)
text_data <- removeWords(text_data, stopwords("en"))
```

```
text_corpus <- Corpus(VectorSource(text_data))
text_tdm <- TermDocumentMatrix(text_corpus)
text_m <- as.matrix(text_tdm)
word_freqs <- sort(rowSums(text_m), decreasing=TRUE)
word_cloud_data <- data.frame(word=names(word_freqs), freq=word_freqs)
```

```
# Open a PNG graphics device
png(filename="/Users/debane/Documents/MS Data Science/630 Predictive Analytics/Term Project/Q91wordcloud

# Your existing code to create the word cloud
text_corpus <- Corpus(VectorSource(text_data))
text_tdm <- TermDocumentMatrix(text_corpus)
text_m <- as.matrix(text_tdm)
word_freqs <- sort(rowSums(text_m), decreasing=TRUE)
word_cloud_data <- data.frame(word=names(word_freqs), freq=word_freqs)

wordcloud(words = word_cloud_data$word, freq = word_cloud_data$freq, min.freq = 1, max.words=200, random

# Close the PNG graphics device
dev.off()
```

```
## pdf
##   2
```

Q93: "Can you describe your most powerful positive psychedelic experience and any enduring changes you attribute to having that experience? (optional)" - Participants could then answer using a text box

```
text_data <- paste(df$Q93, collapse = " ")
```

```
library(tm)
text_data <- tolower(text_data)
text_data <- removePunctuation(text_data)
text_data <- removeNumbers(text_data)
text_data <- removeWords(text_data, stopwords("en"))
```

```
text_corpus <- Corpus(VectorSource(text_data))
text_tdm <- TermDocumentMatrix(text_corpus)
text_m <- as.matrix(text_tdm)
word_freqs <- sort(rowSums(text_m), decreasing=TRUE)
word_cloud_data <- data.frame(word=names(word_freqs), freq=word_freqs)
```

```
# Open a PNG graphics device
png(filename="/Users/debane/Documents/MS Data Science/630 Predictive Analytics/Term Project/Q93wordcloud

# Your existing code to create the word cloud
text_corpus <- Corpus(VectorSource(text_data))
text_tdm <- TermDocumentMatrix(text_corpus)
text_m <- as.matrix(text_tdm)
word_freqs <- sort(rowSums(text_m), decreasing=TRUE)
word_cloud_data <- data.frame(word=names(word_freqs), freq=word_freqs)
```

```r
wordcloud(words = word_cloud_data$word, freq = word_cloud_data$freq, min.freq = 1, max.words=200, rando

# Close the PNG graphics device
dev.off()
```

```
## pdf
##   2
```

References:

A Short Introduction to the caret Package. (n.d.). Retrieved August 9, 2023, from https://cran.r-project.org/web/packages/caret/vignettes/caret.html An Exploration of Naturalistic Psychedelic use. (2023). figshare. https://doi.org/10.6084/m9.figshare.21708044.v1 Bento, C. (2018, December 3). K-Means in Real Life: Clustering Workout Sessions. Medium. https://towardsdatascience.com/k-means-in-real-life-clustering-workout-sessions-119946f9e8dd Bhalla, D. (n.d.). Random Forest in R. ListenData. Retrieved August 9, 2023, from https://www.listendata.com/2014/11/random-forest-with-r.html Gandhi, R. (2018, July 5). Support Vector Machine — Introduction to Machine Learning Algorithms. Medium. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 KNN Algorithm | KNN In R | KNN Algorithm Example. (n.d.). Retrieved August 9, 2023, from https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/ Top 5 Predictive Analytics Models and Algorithms - insightsoftware. (n.d.). Retrieved June 18, 2023, from https://insightsoftware.com/blog/top-5-predictive-analytics-models-and-algorithms/ Understanding K-means Clustering with Examples. (2014, July 25). Edureka. https://www.edureka.co/blog/k-means-clustering/