Duncan McKinnon

West

HW 6

QA

```
suppressPackageStartupMessages( {
  library(tidyverse)
  library(NHANES)
  library(partykit)
  library(rpart)
} )
```

1).

```
set.seed(1847)

n <- nrow(NHANES)
train_pct <- 0.8


# get sample indices
train_ind <- sample.int(n, n * train_pct, replace = F)

# create train and test datasets
train <- NHANES[train_ind, ]
test <- NHANES[-train_ind, ]
```

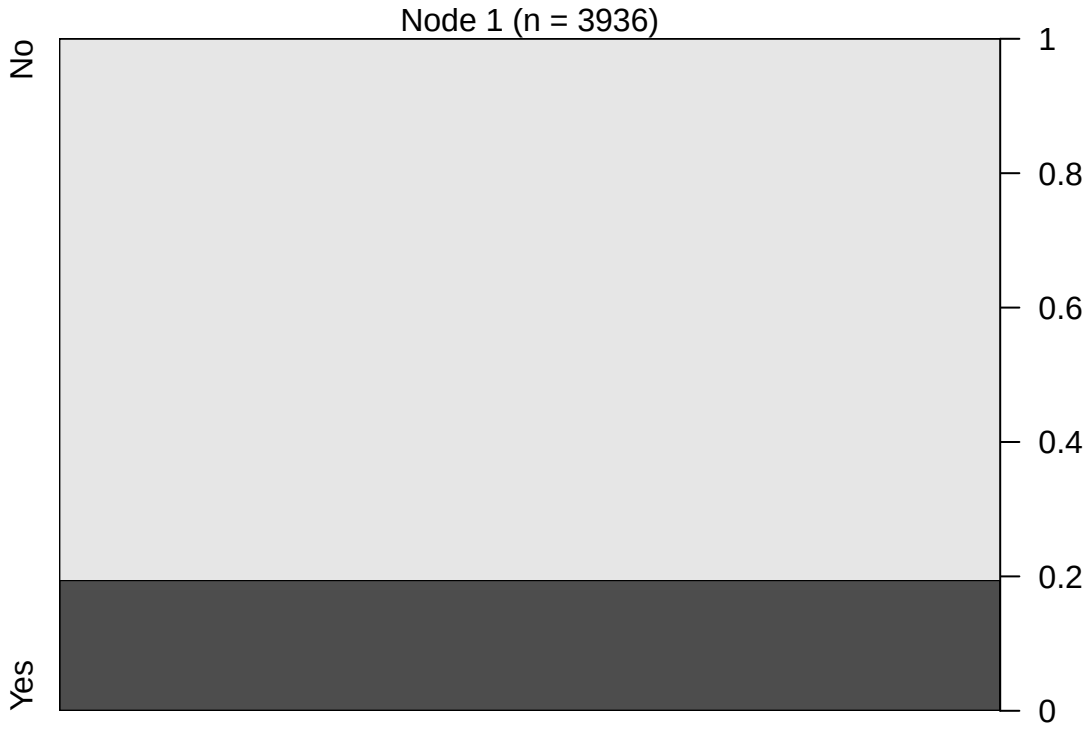
2).

```
# model of hard drug use
mod_marij <- rpart(HardDrugs ~ RegularMarij, train)

mod_marij
```

```
## n=3936 (4064 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3936 765 No (0.8056402 0.1943598) *
```

3).

```
# party kit plot
plot(as.party(mod_marij))
```



4).

```
# Predict hardrug usage for test data
mod_marij_test_result <- test
mod_marij_test_result$pred <- predict(mod_marij, newdata=test,type="class")

# Confusion matrix
table(mod_marij_test_result$pred, mod_marij_test_result$HardDrugs)
```

```
##
##        No Yes
##   No  941 240
##   Yes   0   0
```

5).

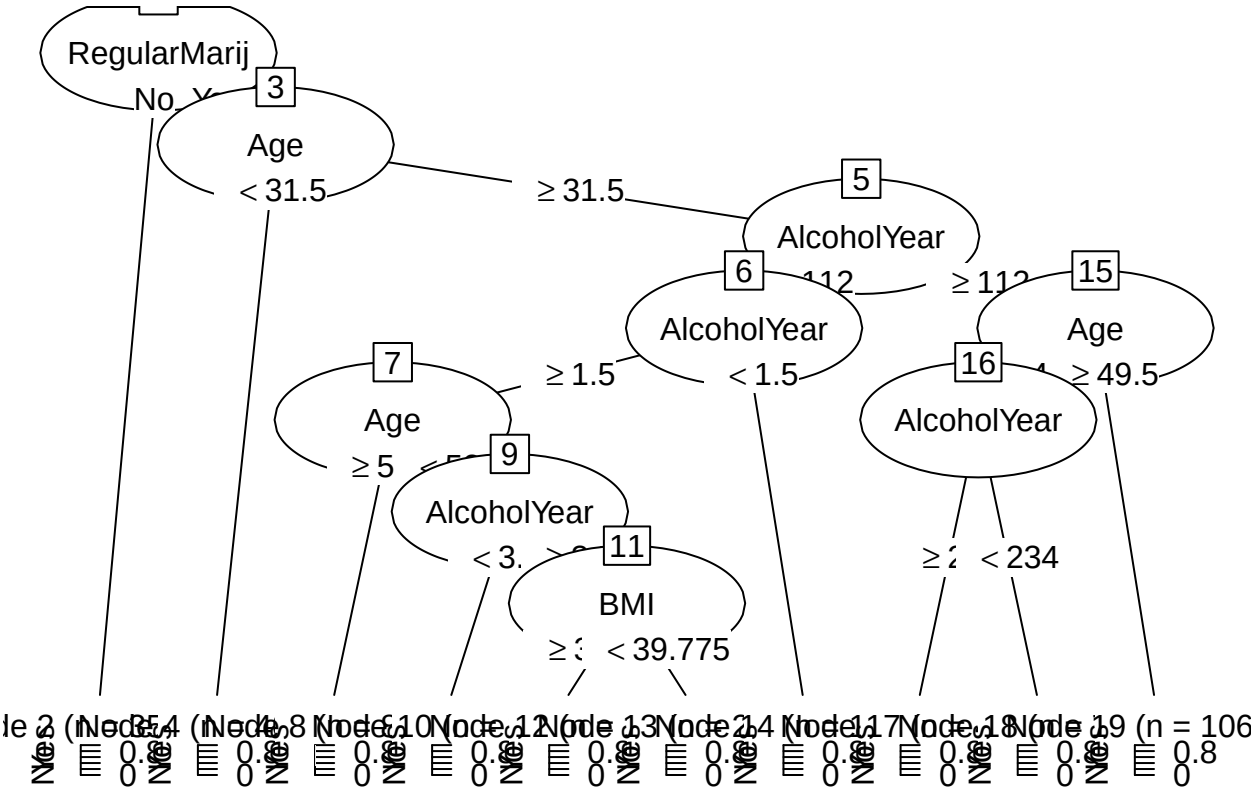
```
# create decision tree using multiple variables
mod_multi <- rpart(HardDrugs ~ RegularMarij + Age + AlcoholYear + BMI, data = train)

mod_multi
```

```
## n=4584 (3416 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
```

```
##       * denotes terminal node
##
##   1) root 4584 825 No (0.82002618 0.17997382)
##     2) RegularMarij=No 3502 289 No (0.91747573 0.08252427) *
##     3) RegularMarij=Yes 1082 536 No (0.50462107 0.49537893)
##       6) Age< 31.5 406 146 No (0.64039409 0.35960591) *
##       7) Age>=31.5 676 286 Yes (0.42307692 0.57692308)
##        14) AlcoholYear< 112 450 213 Yes (0.47333333 0.52666667)
##          28) AlcoholYear>=1.5 338 163 No (0.51775148 0.48224852)
##            56) Age>=52.5 83  26 No (0.68674699 0.31325301) *
##            57) Age< 52.5 255 118 Yes (0.46274510 0.53725490)
##             114) AlcoholYear< 3.5 17   3 No (0.82352941 0.17647059) *
##             115) AlcoholYear>=3.5 238 104 Yes (0.43697479 0.56302521)
##               230) BMI>=39.775 17   4 No (0.76470588 0.23529412) *
##               231) BMI< 39.775 221  91 Yes (0.41176471 0.58823529) *
##          29) AlcoholYear< 1.5 112  38 Yes (0.33928571 0.66071429) *
##        15) AlcoholYear>=112 226  73 Yes (0.32300885 0.67699115)
##          30) Age< 49.5 120  55 Yes (0.45833333 0.54166667)
##            60) AlcoholYear>=234 42  12 No (0.71428571 0.28571429) *
##            61) AlcoholYear< 234 78  25 Yes (0.32051282 0.67948718) *
##          31) Age>=49.5 106  18 Yes (0.16981132 0.83018868) *
```

6).

```
# Decision tree plot
plot(as.party(mod_multi))
```



7).

```
# Predict hardrug usage for test data
mod_multi_test_result <- test
mod_multi_test_result$pred <- predict(mod_multi, newdata=test,type="class")

# Confusion matrix
table(mod_multi_test_result$pred, mod_multi_test_result$HardDrugs)
```

```
##
##        No Yes
##   No  894 143
##   Yes  47  97
```

8).

The partition tree model above predicts whether an individual is a hard drug user using their age, bmi, alcohol habits, and whether they regularly use marijuana. This partition tree allows us to pick out an individual and determine if they are likely to be a hard drug user. If we identified that someone was 34 years old, had a BMI of 10, drank 200 servings of alcohol/year and regularly used marijuana, this model would predict that they were a hard-drug user. In fact, this model would predict that any regular marijuana user who was older than 31.5 and who drinks between 112 and 234 servings of alcohol a year is a hard drug user, regardless of their BMI.

9).

```
# complexity parameter / relative error for decision tree
printcp(mod_multi)
```

```
##
## Classification tree:
## rpart(formula = HardDrugs ~ RegularMarij + Age + AlcoholYear +
##     BMI, data = train)
##
## Variables actually used in tree construction:
## [1] Age          AlcoholYear  BMI          RegularMarij
##
## Root node error: 825/4584 = 0.17997
##
## n=4584 (3416 observations deleted due to missingness)
##
##         CP nsplit rel error  xerror     xstd
## 1 0.063030      0   1.00000 1.00000 0.031527
## 2 0.012525      2   0.87394 0.88848 0.030079
## 3 0.010909      6   0.82303 0.88242 0.029996
## 4 0.010000      9   0.79030 0.87273 0.029861
```

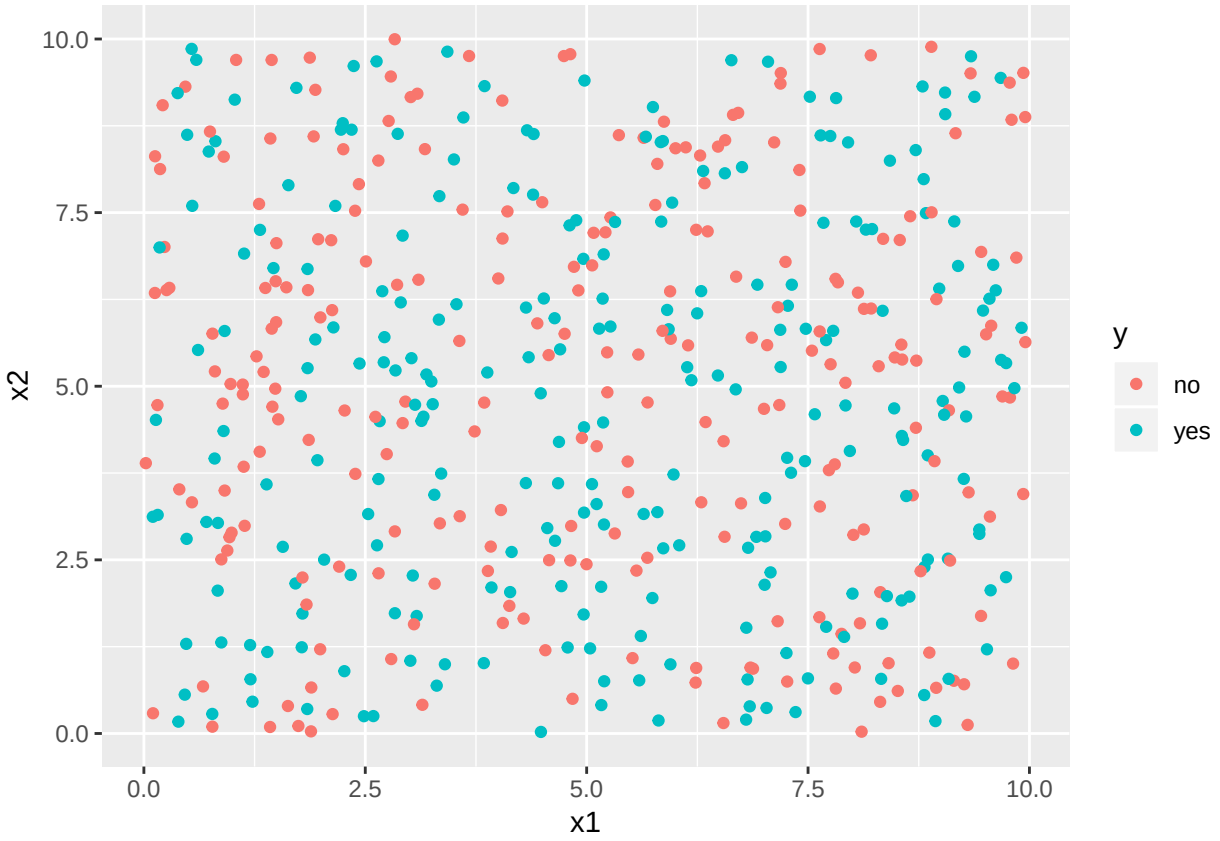
QB

1, 2, 3).

```
# set random seed
set.seed(1847)

# create simulated dataset
sim_data <- data.frame(x1 = runif(500, 0, 10), x2 = runif(500, 0, 10), y = sample(c('yes', 'no'), 500, r
```

4).

```
# Plot predictors and response
ggplot(sim_data) +
  geom_point(aes(x = x1, y = x2, color = y))
```

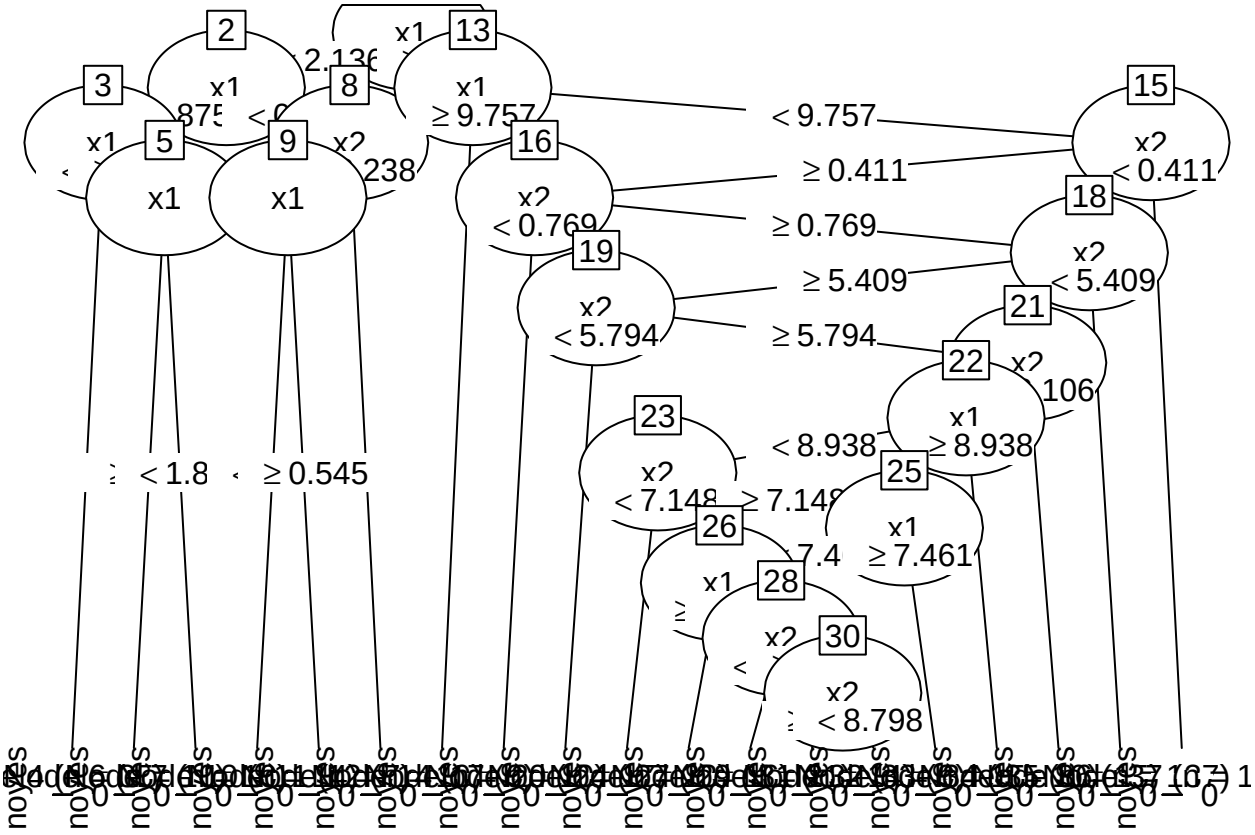


5).

```
# reset random seed
set.seed(1847)
n_sim <- nrow(sim_data)
train_pct_sim <- 0.8

# create training indices
train_ind_sim <- sample(n_sim, n_sim * train_pct_sim, replace = F)

# create train and test sets
train_sim <- sim_data[train_ind_sim, ]
test_sim <- sim_data[-train_ind_sim, ]
```

6).

```
# fit decision tree
mod_sim <- rpart(y ~ x1 + x2, data = train_sim)

# plot decision tree
plot(as.party(mod_sim))
```



7).

```
# predict test results from decision tree
pred_sim <- test_sim
pred_sim$pred <- predict(mod_sim, newdata = test_sim, type="class")

# create confusion matrix for test predictions
conf <- table(pred_sim$pred, pred_sim$y)
conf
```

```
##
##       no yes
##   no  15  21
##   yes 31  33
```

```
# calculate accuracy of predictions
(conf[1,1] + conf[2,2]) / sum(conf)
```

```
## [1] 0.48
```

8).

Because there is no relationship between the predictors and the binary response, it makes sense that the accuracy of the model is around 50%. When there is no relationship and only 2 possible outcomes, we would expect the prediction to be correct about half the time due solely to random variability. An accuracy

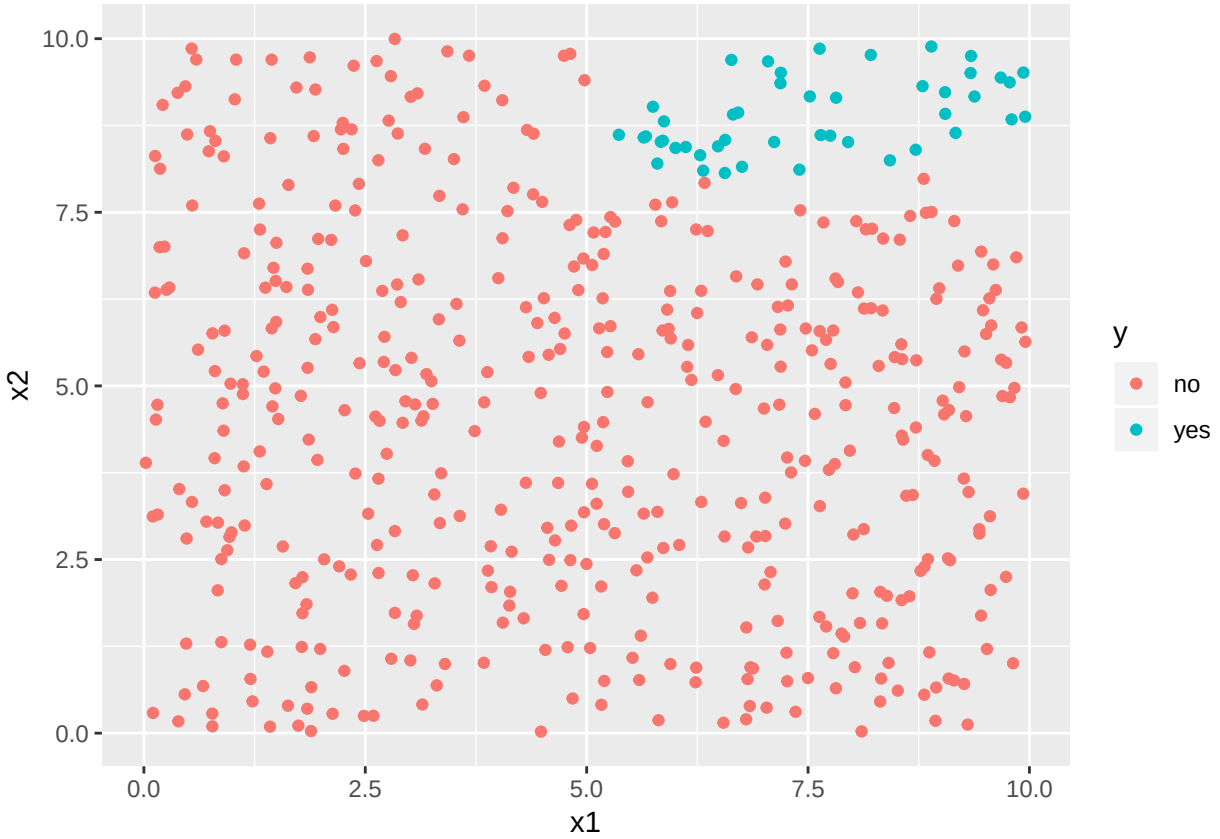
significantly below 50 percent would imply an inverse correlation between prediction and response, and an accuracy level significantly above 50 percent would imply a direct correlation, so with no correlation we should expect to see around 50% accuracy.

9, 10, 11).

```
# reset random seed
set.seed(1847)

# create simulated dataset
sim_data2 <- data.frame(x1 = runif(500, 0, 10), x2 = runif(500, 0, 10))
sim_data2$y <- ifelse((sim_data2$x1 > 5 & sim_data2$x2 > 8), 'yes', 'no')
```

12).

```
# Plot predictors and response
ggplot(sim_data2) +
  geom_point(aes(x = x1, y = x2, color = y))
```

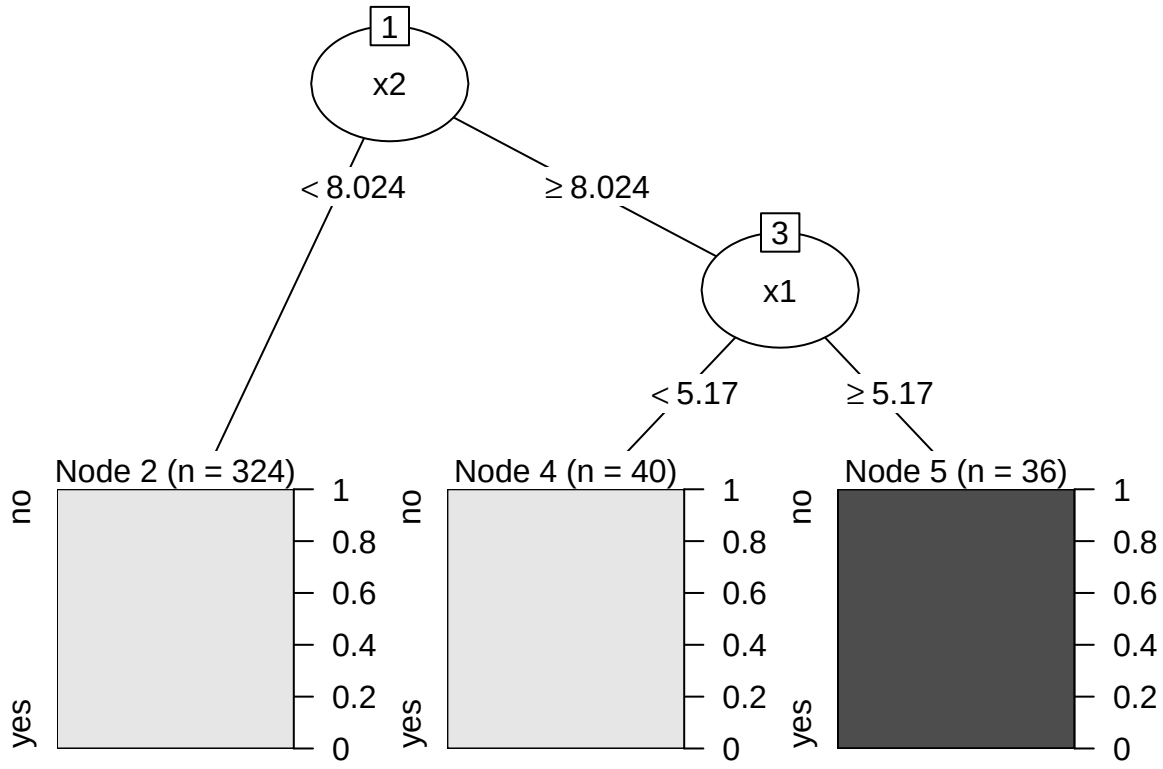


13).

```r
# reset random seed
set.seed(1847)
n_sim2 <- nrow(sim_data2)
train_pct_sim2 <- 0.8

# create training indices
train_ind_sim2 <- sample(n_sim2, n_sim2 * train_pct_sim2, replace = F)

# create train and test sets
train_sim2 <- sim_data2[train_ind_sim2, ]
test_sim2 <- sim_data2[-train_ind_sim2, ]
```

14).

```r
# fit decision tree
mod_sim2 <- rpart(y ~ x1 + x2, data = train_sim2)

# plot decision tree
plot(as.party(mod_sim2))
```



15).

```
# predict test results from decision tree
pred_sim2 <- test_sim2
pred_sim2$pred <- predict(mod_sim2, newdata = test_sim2, type="class")

# create confusion matrix for test predictions
conf2 <- table(pred_sim2$pred, pred_sim2$y)
conf2
```

```
##
##       no yes
##   no  90   0
##   yes  0  10
```

```
# calculate accuracy of predictions
(conf2[1,1] + conf2[2,2]) / sum(conf2)
```

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

16).

In generating the decision tree the algorithm first splits the data based on the value of the x2 variable before accounting for the value of x1. This first split makes sense given the relationship between the response variable y, and the predictors x1 and x2. By splitting on x2 first, the model can immediately categorize about 80% of the training data because y can only be 'yes' if $x2 > 8$ (where x2 is uniformly distributed between 0 and 10). This accounts for more of the data than if the model split on x1 first, because y can only be 'yes' if $x1 > 5$, meaning that splitting on x1 first would only allow the model to categorize about 50% of the data (as x1 is also uniformly distributed between 0 and 10).

17).

This simulation was created to model the scenario where the response was 'yes' if $x1 > 5$ and $x2 > 8$, and 'no' otherwise. The decision tree found branching points very close to the actual values of the threshold (5.17 and 8.024 respectively). The difference between the thresholds found by the model and the actual thresholds used to generate the data is a consequence of the relative size and coverage of the dataset used to simulate this relationship. If all the possibilities in the uniform space between 0 and 10 that could be occupied by x1 and x2 were accounted for (once), then the thresholds the model identified would match the thresholds used to generate the simulation. It would be able to account for every equally weighted possibility in determining the structure of the system. Since only a limited number of real values for x1 and x2 were randomly generated within the uniform space between 0 and 10, the model can only account for the given subset of the possibilities it has been presented (sample), leading to thresholds that don't necessarily match the exact (ideal) function used when labeling the responses from given predictors.