

# **Week 11: Binary Classification Decision Trees**

## **Predicting Classes rather than Numeric Values**

Scott Schwartz

May 18, 2021

# Decision Trees

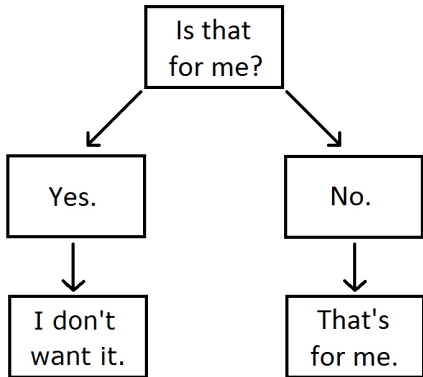


## Decision Trees



# Binary Classification Decision Trees

My Cat's Decision-Making Tree.



## Statistical Inference

- Hypothesis Testing
  - One sample simulation p-values
  - Two sample permutation p-values
- Estimation
  - Bootstrap Confidence Intervals
- Prediction
  - Simple Linear Regression: real-valued outcome predictions with one predictor
  - Multivariate Linear Regression: real-valued outcome predictions with many predictors
  - **Classification**: class prediction with many predictors

# Statistics and Data Science

## Statistics: Inference

- Hypothesis Testing
- Estimation (Confidence Intervals)

## Data Science: Machine Learning

- Prediction
  - Regression
  - Classification

- Hypothesis Testing and Estimation are both aspects of Simple Linear and Multivariate Regression
  - The same is true for a related Classification method called Logistic Regression
- Our Machine Learning (ML) focus is on the Decision Tree Classification
  - but there are many other ML Classification methods and also many ML Regression methods beyond Simple Linear and Multivariate Regression models from Statistics

## Trouble Sleeping?

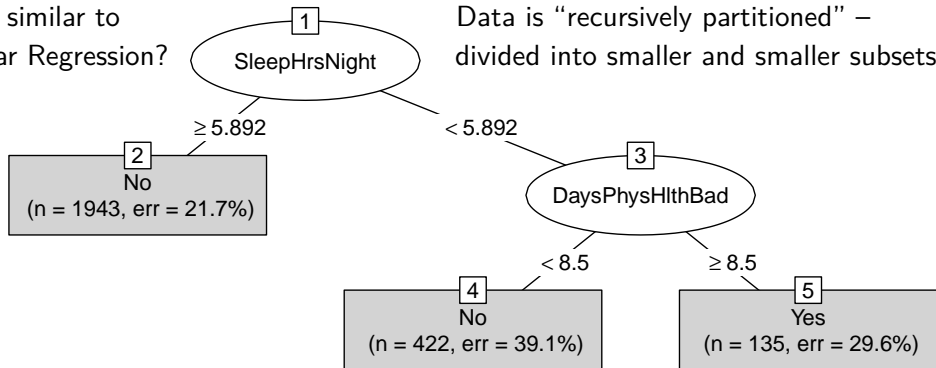
```
# https://cran.r-project.org/web/packages/NHANES/NHANES.pdf
# Random sample of n=2500 observations from the
# US National Health and Nutrition Examination Study (NHANES)
library(tidyverse); NHANES <- read_csv("NHANES_data.csv")
NHANES %>% select(Age, Race3, Work, DaysMentHlthBad, DaysPhysHlthBad,
                  Depressed, SleepHrsNight, SleepTrouble) %>% glimpse()

## Rows: 2,500
## Columns: 8
## $ Age          <dbl> 64, 32, 67, 43, 75, 21, 30, 79, 37, 60, 27, 45, 26, 47~
## $ Race3        <chr> "Hispanic", NA, NA, NA, "White", "Hispanic", "White", ~
## $ Work         <chr> "NotWorking", "Working", "Working", "NotWorking", "Not~
## $ DaysMentHlthBad <dbl> 0, 0, 0, 30, 0, 0, 2, 0, 30, 3, 4, 26, 3, 0, 30, 2, 30~
## $ DaysPhysHlthBad <dbl> 0, 0, 0, 21, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 10, 3~
## $ Depressed     <chr> "None", "None", "None", "Most", "None", "None", "None"~
## $ SleepHrsNight <dbl> 5.477751, 6.819995, 5.852940, 5.721259, 8.071489, 6.39~
## $ SleepTrouble  <chr> "Yes", "No", "No", "Yes", "Yes", "No", "Yes", "No", "N~
```

## Who reports sleep trouble?

```
library(rpart) #install.packages("rpart") # for the `rpart()` function
tree <- rpart(SleepTrouble ~ SleepHrsNight + DaysPhysHlthBad, data=NHANES)
library(partykit) #install.packages("partykit") for `as.party()` & `plot()`
plot(as.party(tree), gp=gpar(cex=1), type="simple")
```

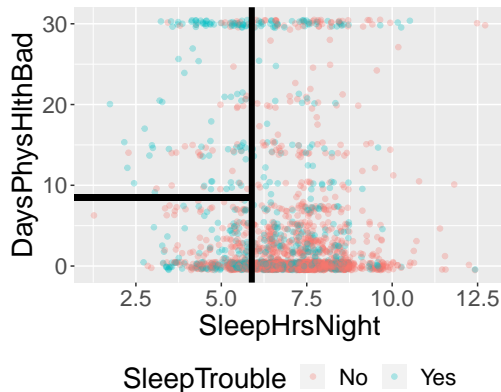
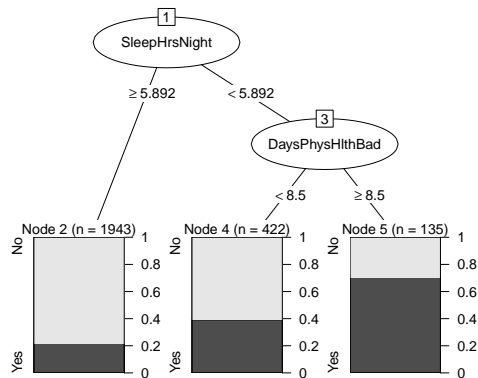
Very similar to  
Linear Regression?



# Geometric Interpretation

Data is "recursively partitioned"

Divided into smaller and smaller subsets...

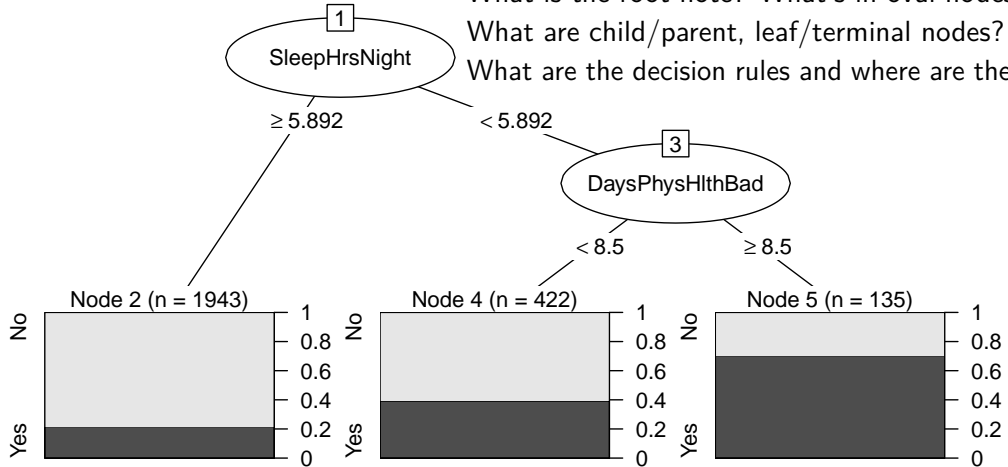




# Who reports sleep trouble?

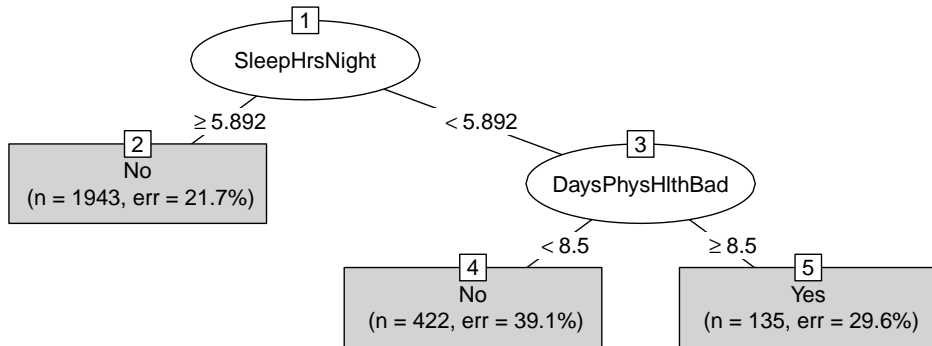
```
plot(as.party(tree), type="extended")
```

What is the root node? What's in oval nodes?  
What are child/parent, leaf/terminal nodes?  
What are the decision rules and where are they?



## What kind of things could be different here?

```
plot(as.party(tree), gp=gpar(cex=1), type="simple")
```



**Does someone have having trouble sleeping if**

- they typically gets 5.2 hours of sleep on weeknights
- report having 4 days of poor physical health over the past month ?

# What kind of things could be different here?

## Nonstandard Binary Classification Decision Trees?

- Why are there two branches only?
- Why are decision rules only a single cutoff?
- Why are decision rules based on one variable only?
- Are “composite” decision rules possible?

How would we use a nonbinary categorical variable?

```
MaritalStatus <-  
  as.factor(NHANES$MaritalStatus)  
  
MaritalStatus2 <-  
  as.numeric(MaritalStatus)  
  
tibble(level_text=MaritalStatus,  
        level_numeric=MaritalStatus2) %>%  
  unique() %>% arrange(level_numeric)
```

```
## # A tibble: 6 x 2  
##   level_text    level_numeric  
##   <fct>          <dbl>  
## 1 Divorced             1  
## 2 LivePartner          2  
## 3 Married              3  
## 4 NeverMarried         4  
## 5 Separated            5  
## 6 Widowed              6
```

## How are decision trees constructed?

- 1 All possible **variables** and all their possible **decision rules** are considered
  - Each **variable** and **decision rule** creates two new child nodes
- 2 All **variable** and **decision rule** violating **stopping rule** are discarded
  - node must be large enough to split, resulting nodes must be large enough
  - maximum tree depth, and maximum node count must not be exceeded
- 3 Of those remaining, the **variable** and **decision rule** producing the lowest *entropy/gini impurity* nodes is selected (for possible addition into the tree)
  - $\text{entropy}(p) = -p \log(p) - (1 - p) \log(1 - p)$     •  $\text{gini}(p) = 1 - p^2 - (1 - p)^2$   
(*entropy* and *gini impurity* are highest if  $p = 0.5$  and lowest if  $p = 1$  or  $p = 0$ )
- 4 A **variable** and **decision rule** giving “sufficient improvement” is added to the tree
  - The `cp` parameter in the `rpart()` function defines “sufficient improvement”:  
 $\text{Error}(T) + \text{cp} \times \text{splits} \times \text{Error}(T_0)$  must improve ( $T_0$  is the no splits tree)

## How are decision trees evaluated?

The outcome/response  $y_i$  (and hence  $\hat{y}$ ) is re-coded as 0 (negative) or 1 (positive)

- $\hat{y}$  either *predicts* or *does not predict* the positive event
- $\hat{y}$  for “Do you have sleep trouble? Yes or No?” is 1 if “Yes” is predicted

- |                        |   |   |
|------------------------|---|---|
| 1. Accuracy:           | $\frac{\sum_{i=1}^n 1_{y_i}(\hat{y}_i)}{\sum_{i=1}^n 1_{y_i}(\hat{y}_i) \times 1_1(\hat{y}_i)}$ | where $1_{y_i}(\hat{y}_i) = 1$ if $\hat{y}_i = y_i$ |
| 2. Precision:          | $\frac{\sum_{i=1}^n \hat{y}_i}{\sum_{i=1}^n 1_{y_i}(\hat{y}_i) \times 1_1(\hat{y}_i)}$          | where $1_1(\hat{y}_i) = 1$ if $\hat{y}_i = 1$       |
| 3. Sensitivity/Recall: | $\frac{\sum_{i=1}^n y_i}{\sum_{i=1}^n 1_{y_i}(\hat{y}_i) \times 1_1(\hat{y}_i)}$                |   |
| 4. Specificity:        | $\frac{\sum_{i=1}^n 1_{y_i}(\hat{y}_i) \times 1_0(\hat{y}_i)}{\sum_{i=1}^n (1 - y_i)}$          | where $1_0(\hat{y}_i) = 1$ if $\hat{y}_i = 0$       |

For even more versions see: [https://en.wikipedia.org/wiki/Sensitivity\\_and\\_specificity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity)

**What is the meaning of Accuracy, Precision, Sensitivity/Recall, and Specificity?**

What do the expressions mean?  $\sum_{i=1}^n y_i$ ,  $\sum_{i=1}^n (1 - y_i)$ ,  $\sum_{i=1}^n 1_{y_i}(\hat{y}_i) \times 1_1(\hat{y}_i)$ , etc.?

## How are decision trees evaluated?

The outcome/response  $y_i$  (and hence  $\hat{y}$ ) is re-coded as 0 (negative) or 1 (positive)

- $\hat{y}$  either *predicts* or *does not predict* the positive event
- $\hat{y}$  for “Do you have sleep trouble? Yes or No?” is 1 if “Yes” is predicted

### TP, TN, FP, FN, and Confusion Matrices – What are each of these things?

	$y_i = 0$	$y_i = 1$
$\hat{y}_i = 0$	$n_{TN}$	$n_{FN}$
$\hat{y}_i = 1$	$n_{FP}$	$n_{TP}$

	$y_i = 0$	$y_i = 1$
$\hat{y}_i = 0$	✓	×
$\hat{y}_i = 1$	×	✓

$$n = n_{TN} + n_{FN} + n_{FP} + n_{TP}$$

This is like *Type I* and *Type II* errors from the *Hypothesis Testing* context!

- What do FP, FN, TP, or TN predictions mean practically in different contexts?

## How are decision trees evaluated?

The outcome/response  $y_i$  (and hence  $\hat{y}$ ) is re-coded as 0 (negative) or 1 (positive)

- $\hat{y}$  either *predicts* or *does not predict* the positive event
- $\hat{y}$  for “Do you have sleep trouble? Yes or No?” is 1 if “Yes” is predicted

### TP, TN, FP, FN, and Confusion Matrices – What are each of these things?

	$y_i = 0$	$y_i = 1$
$\hat{y}_i = 0$	$n_{TN}$	$n_{FN}$
$\hat{y}_i = 1$	$n_{FP}$	$n_{TP}$

$$n = n_{TN} + n_{FN} + n_{FP} + n_{TP}$$

$$1. \text{ Accuracy: } \frac{n_{TP} + n_{TN}}{n}$$

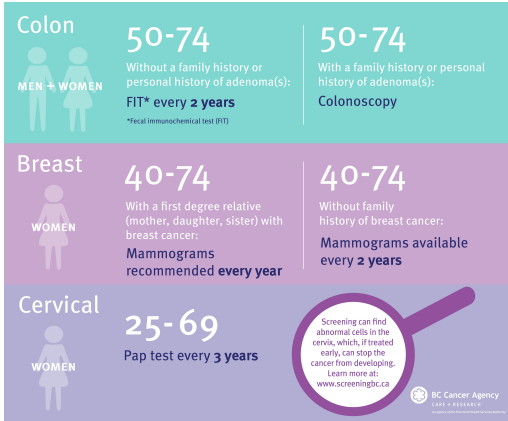
$$2. \text{ Precision: } \frac{n_{TP}}{n_{TP} + n_{FP}}$$

$$3. \text{ Sensitivity/Recall: } \frac{n_{TP}}{n_{TP} + n_{FN}}$$

$$4. \text{ Specificity: } \frac{n_{TN}}{n_{TN} + n_{FP}}$$

For more information see: [https://en.wikipedia.org/wiki/Confusion\\_matrix#Example](https://en.wikipedia.org/wiki/Confusion_matrix#Example)

## What's Worse? FN or FP?



Is it more costly to wrongly ignore cancer (FN)?



Is it more costly to approve bad applicants (FP)?



## What's Worse? FN or FP?

Sometimes it's not immediately clear which may be better or worse

Regarding self-quarantine, some questions

- Does FN meaningfully increase transmission?
- Does FP meaningfully decrease well-being?

Regarding hospitalization, some questions

- Does FN significantly worsen outcomes?
- Does FP prohibitively burden medical care?



This depends on the costs and benefits of the intervention/failure to intervene

## What's Worse? FN or FP?

- **Sensitivity/Recall** is also called the **True Positive Rate (TPR)**
  - The **FALSE Positive Rate (FPR)** is  $1 - \text{TPR}$   
FPR tells you the proportion of positives that are missed
- **Specificity** is also called the **True Negative Rate (TNR)**
  - The **FALSE Negative Rate (FNR)** is  $1 - \text{TNR}$   
FNR tells you the proportion of negatives that are missed

For more information see: [https://en.wikipedia.org/wiki/Confusion\\_matrix#Example](https://en.wikipedia.org/wiki/Confusion_matrix#Example)

## How are decision trees evaluated?

- 1 Decide what matters: accuracy, precision, sensitivity/recall, or specificity?

Accuracy is often a good choice, but not for some **class imbalance** contexts

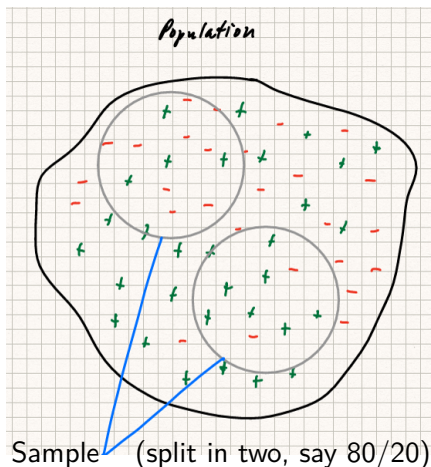
- If you care about FP but they're rare, then accuracy overemphasizes FN
  - If most predictions are negative, then most errors are FN, so FN drives accuracy...
- If you care about FN but they're rare, then accuracy overemphasizes FP
  - If most predictions are positive, then most errors are FP, so FP drives accuracy...

- 2 Use the 80-20 train-test split methodology to compare competing models

- Build the classification model with 80% of the data
- Score the predictions on the remaining 20% of the data
- This suggests how well a model will actually generalize and perform on new data

## Review: The 80/20 Train-Test Split

- Classification is about doing something (predicting) in the sample
- *So you should try to see how well you can do that thing in the population...*



← **Here we split a "representative" population sample into two "representative" samples**

- 1 You fit the model based on a "representative" sample
- 2 So subsamples are "representative of the population"
- 3 Use 80% of the data to fit the "representative" model
- 4 Use 20% to see if the model's actually "representative"

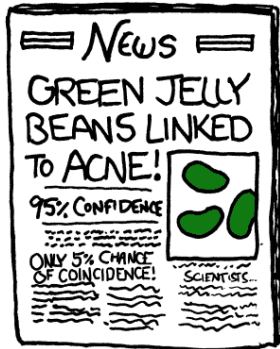
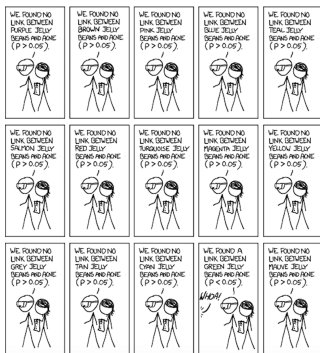
**This strategy shows when the model for 80% of the data doesn't work well for the remaining 20% of the data, which could happen if**

- The subsamples aren't "representative" to begin with
- The model is overly specific to 80% of the data

# Review: The 80/20 Train-Test Split

Fit a model on 80% of the data → "score" the model on the remaining 20%

- The train-test method is a wonderful tool in LARGE data contexts
  - when there's enough data so the random train-test split isn't just "lucky"
- In its more advanced (data science) forms, train-test is a powerful model tuning tool

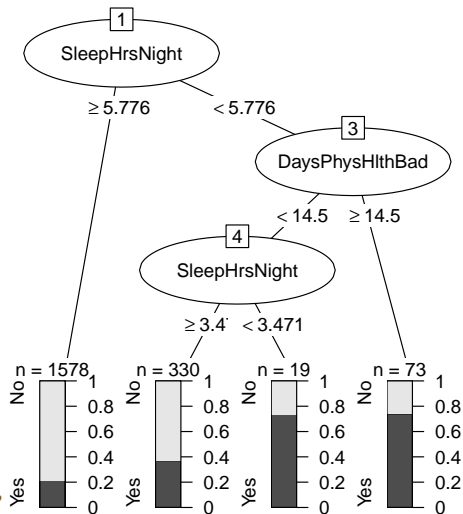


- Like Hypothesis Testing, it is subject to "random chance" (of the test-train split)
- Unlike Hypothesis Testing, it is based on observed out of sample generalizability, rather than tests based on modeling assumptions
- It's not about parameters or 'right' or 'wrong', but picking models predicting new data 'well'

## Trouble Sleeping: 80-20 Train-Test Split

```
NHANES %>% rowid_to_column() ->  
  NHANES; n <- nrow(NHANES)  
set.seed(1002); train_ids <-  
  sample(1:n, size=round(0.8*n))  
train <- NHANES %>%  
  filter(rowid %in% train_ids)  
test <- NHANES %>%  
  filter(!(rowid %in% train_ids))
```

```
tree <- rpart(SleepTrouble ~  
  SleepHrsNight + DaysPhysHlthBad,  
  data=train)  
tree %>% as.party() %>%  
plot(type="extended", #gp=gpar(cex=0.8),  
  tp_args=list(id=FALSE))
```



## Trouble Sleeping: Confusion Matrices

```
tree_train_pred <- predict(  
  tree, type="class")  
train_conf_mtx <- table(  
  `y-hat`=tree_train_pred,  
  `observed y`=train$SleepTrouble)  
train_conf_mtx
```

```
##           observed y  
## y-hat      No  Yes  
##   No  1447  461  
##   Yes   24   68
```

```
train_conf_mtx/sum(train_conf_mtx)
```

```
##           observed y  
## y-hat      No    Yes  
##   No  0.7235 0.2305  
##   Yes 0.0120 0.0340
```

```
tree_test_pred <- predict(  
  tree, type="class", newdata=test)  
test_conf_mtx <- table(  
  `y-hat`=tree_test_pred,  
  `observed y`=test$SleepTrouble)  
test_conf_mtx
```

```
##           observed y  
## y-hat      No  Yes  
##   No   339  133  
##   Yes    9   19
```

```
test_conf_mtx/sum(test_conf_mtx)
```

```
##           observed y  
## y-hat      No    Yes  
##   No  0.678 0.266  
##   Yes 0.018 0.038
```

## Trouble Sleeping: Metrics (Accuracy, etc.)

```
n_TN <- train_conf_mtrx[1,1]
n_FN <- train_conf_mtrx[1,2]
n_FP <- train_conf_mtrx[2,1]
n_TP <- train_conf_mtrx[2,2]
# accuracy
(n_TP+n_TN)/sum(train_conf_mtrx)
```

```
## [1] 0.7575
```

```
# precision
(n_TP)/(n_TP+n_FP)
```

```
## [1] 0.7391304
```

```
# sensitivity/recall:  $TPR = 1-FPR$ 
(n_TP)/(n_TP+n_FN)
```

```
## [1] 0.1285444
```

```
# specificity:  $TNR = 1-FNR$ 
(n_TN)/(n_TN+n_FP)
```

```
## [1] 0.9836846
```

```
n_TN <- test_conf_mtrx[1,1]
n_FN <- test_conf_mtrx[1,2]
n_FP <- test_conf_mtrx[2,1]
n_TP <- test_conf_mtrx[2,2]
# accuracy
(n_TP+n_TN)/sum(test_conf_mtrx)
```

```
## [1] 0.716
```

```
# precision
(n_TP)/(n_TP+n_FP)
```

```
## [1] 0.6785714
```

```
# sensitivity/recall:  $TPR = 1-FPR$ 
(n_TP)/(n_TP+n_FN)
```

```
## [1] 0.125
```

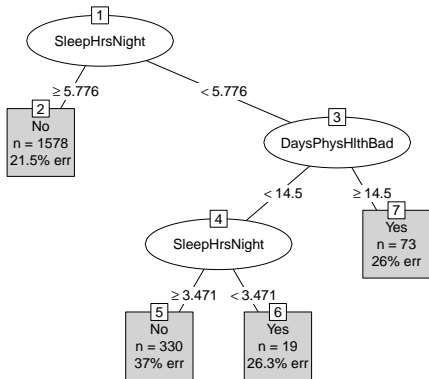
```
# specificity:  $TNR = 1-FNR$ 
(n_TN)/(n_TN+n_FP)
```

```
## [1] 0.9741379
```

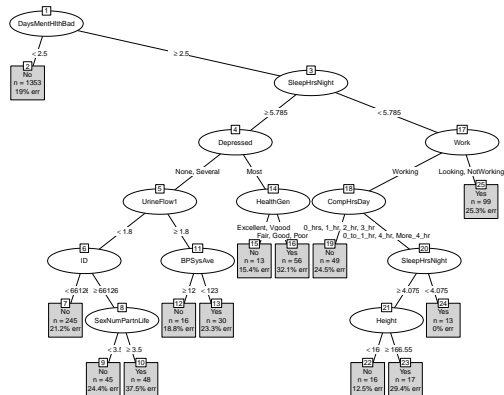


# Trouble Sleeping: Comparing Models

```
tree1 <- rpart(SleepTrouble~SleepHrsNight+DaysPhysHlthBad,
  data=train)
tree1 %>% as.party() %>% plot(type="simple",
  tp_args = list(FUN = function(info)
    list(format(info$prediction), format(paste("n =",info$n)),
      format(paste(round(info$err,1), "% err", sep=""))))))
```



```
tree2 <- rpart(SleepTrouble ~ ., data=train)
tree2 %>% as.party() %>% plot(type="simple",
  ep_args = list(justmin=15),
  tp_args = list(FUN = function(info)
    list(format(info$prediction), format(paste("n =",info$n)),
      format(paste(round(info$err,1), "% err", sep=""))))))
```



# Trouble Sleeping: Comparing Models

Let's just compare the out of sample generalizability

```
tree1_test_pred <-  
  predict(tree1, type="class",  
          newdata=test)  
tree1_test_confmtrx <-  
  table(`y-hat`=tree1_test_pred,  
        `observed y`=test$SleepTrouble)  
tree1_test_confmtrx
```

```
##      observed y  
## y-hat  No Yes  
##   No  339 133  
##   Yes   9  19
```

*# accuracy*

```
n <- sum(tree1_test_confmtrx)  
sum(diag(tree1_test_confmtrx))/n
```

```
## [1] 0.716
```

```
tree2_test_pred <-  
  predict(tree2, type="class",  
          newdata=test)  
tree2_test_confmtrx <-  
  table(`y-hat`=tree2_test_pred,  
        `observed y`=test$SleepTrouble)  
tree2_test_confmtrx
```

```
##      observed y  
## y-hat  No Yes  
##   No  320 116  
##   Yes  28  36
```

*# accuracy*

```
n <- sum(tree2_test_confmtrx)  
sum(diag(tree2_test_confmtrx))/n
```

```
## [1] 0.712
```

# Trouble Sleeping: Comparing Models

Let's just compare the out of sample generalizability

```
tree1_test_pred <-  
  predict(tree1, type="class",  
          newdata=test)  
tree1_test_confmtrx <-  
  table(`y-hat`=tree1_test_pred,  
        `observed y`=test$SleepTrouble)  
tree1_test_confmtrx/sum(tree1_test_confmtrx)
```

```
##      observed y  
## y-hat    No    Yes  
##   No  0.678 0.266  
##   Yes 0.018 0.038
```

*# accuracy*

```
n <- sum(tree1_test_confmtrx)  
sum(diag(tree1_test_confmtrx))/n
```

```
## [1] 0.716
```

```
tree2_test_pred <-  
  predict(tree2, type="class",  
          newdata=test)  
tree2_test_confmtrx <-  
  table(`y-hat`=tree2_test_pred,  
        `observed y`=test$SleepTrouble)  
tree2_test_confmtrx/sum(tree2_test_confmtrx)
```

```
##      observed y  
## y-hat    No    Yes  
##   No  0.640 0.232  
##   Yes 0.056 0.072
```

*# accuracy*

```
n <- sum(tree2_test_confmtrx)  
sum(diag(tree2_test_confmtrx))/n
```

```
## [1] 0.712
```

## Trouble Sleeping: Comparing Models

```
n_TN <- tree1_test_confmtrx[1,1]
n_FN <- tree1_test_confmtrx[1,2]
n_FP <- tree1_test_confmtrx[2,1]
n_TP <- tree1_test_confmtrx[2,2]
# accuracy
(n_TP+n_TN)/sum(tree1_test_confmtrx)
```

```
## [1] 0.716
```

```
# precision
(n_TP)/(n_TP+n_FP)
```

```
## [1] 0.6785714
```

```
# sensitivity/recall: TPR = 1-FPR
(n_TP)/(n_TP+n_FN)
```

```
## [1] 0.125
```

```
# specificity: TNR = 1-FNR
(n_TN)/(n_TN+n_FP)
```

```
## [1] 0.9741379
```

```
n_TN <- tree2_test_confmtrx[1,1]
n_FN <- tree2_test_confmtrx[1,2]
n_FP <- tree2_test_confmtrx[2,1]
n_TP <- tree2_test_confmtrx[2,2]
# accuracy
(n_TP+n_TN)/sum(tree2_test_confmtrx)
```

```
## [1] 0.712
```

```
# precision
(n_TP)/(n_TP+n_FP)
```

```
## [1] 0.5625
```

```
# sensitivity/recall: TPR = 1-FPR
(n_TP)/(n_TP+n_FN)
```

```
## [1] 0.2368421
```

```
# specificity: TNR = 1-FNR
(n_TN)/(n_TN+n_FP)
```

```
## [1] 0.9195402
```

# Comparing Models

Which of the two models above do we prefer? [Use generalizability on *test* data...]

## How to choose between models: Rationale 1

- If a FP is costly, then low FPR (high sensitivity/recall) is important
  - $\text{sensitivity/recall} = \text{TPR} = 1 - \text{FPR}$
- If a FN is costly, then low FNR (high specificity) is important
  - $\text{specificity} = \text{TNR} = 1 - \text{FNR}$
- Accuracy is a weighted average of sensitivity/recall and specificity
  - sensitivity/recall gets more weight if positive outcomes are more prevalent
  - specificity gets more weight if negative outcomes are more prevalent

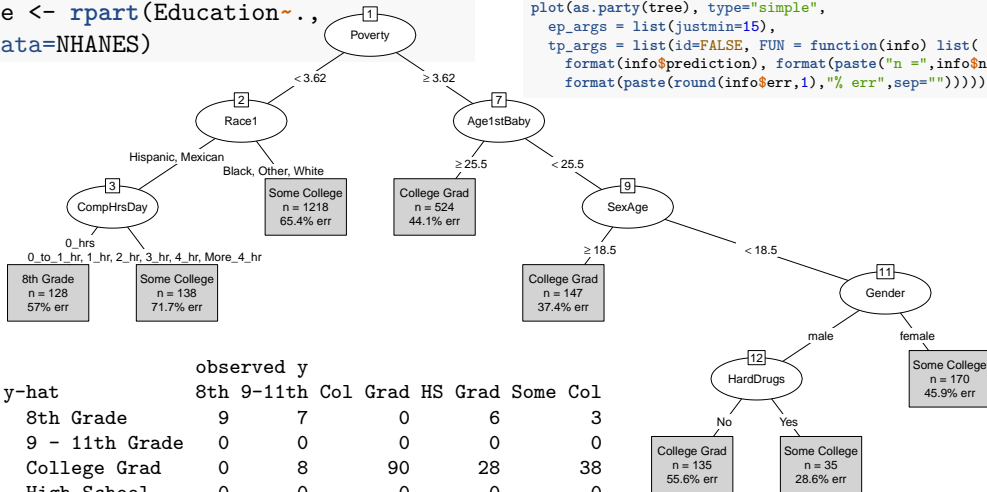
## How to choose between models: Rationale 2

- Less complex models are less likely to overfit the data
  - Less complex model performance tends to generalize better
- Less complex models tends be easier to interpret than more complex models

# Multi-Class (Non-Binary) Outcomes

```
tree <- rpart(Education~.,
  data=NHANES)
```

```
plot(as.party(tree), type="simple",
  ep_args = list(justmin=15),
  tp_args = list(id=FALSE, FUN = function(info) list(
    format(info$prediction), format(paste("n =",info$n)),
    format(paste(round(info$err,1),"% err",sep="")))))
```



##		observed y				
##	y-hat	8th	9-11th	Col	Grad	HS Grad
##	8th Grade	9	7	0	6	3
##	9 - 11th Grade	0	0	0	0	0
##	College Grad	0	8	90	28	38
##	High School	0	0	0	0	0
##	Some College	18	49	54	72	117

## Controlling FP and FN with Thresholding

```
tree2_test_pred <-  
  predict(tree2, type="class",  
          newdata=test)  
table(`y-hat`=tree2_test_pred,  
      `observed y`=test$SleepTrouble)
```

```
##      observed y  
## y-hat  No Yes      FPR: 8%  
##   No  320 116      TPR: 24%  
##   Yes   28  36
```

```
predict(tree2, newdata=test) %>%  
  as_tibble() -> tree2_test_pred  
tree2_test_pred %>% head(3)
```

```
## # A tibble: 3 x 2  
##       No    Yes  
##   <dbl> <dbl>  
## 1 0.810 0.190  
## 2 0.253 0.747  
## 3 0.810 0.190
```

```
threshold_20 <- tree2_test_pred %>%  
  mutate(prediction=ifelse(  
    Yes>=0.20, "Yes", "No"))  
table(`y-hat`=threshold_20$prediction,  
      `observed y`=test$SleepTrouble)
```

```
##      observed y  
## y-hat  No Yes      FPR: 25%  
##   No  261  94      TPR: 38%  
##   Yes   87  58
```

```
threshold_70 <- tree2_test_pred %>%  
  mutate(prediction=ifelse(  
    Yes>=0.70, "Yes", "No"))  
table(`y-hat`=threshold_70$prediction,  
      `observed y`=test$SleepTrouble)
```

```
##      observed y  
## y-hat  No Yes      FPR: 5%  
##   No  333 127      TPR: 17%  
##   Yes   15  25
```