Decision Trees: Next Level

Measuring Success

Gini Index and Cost Complexity

Recall: Our simplest cannabis tree

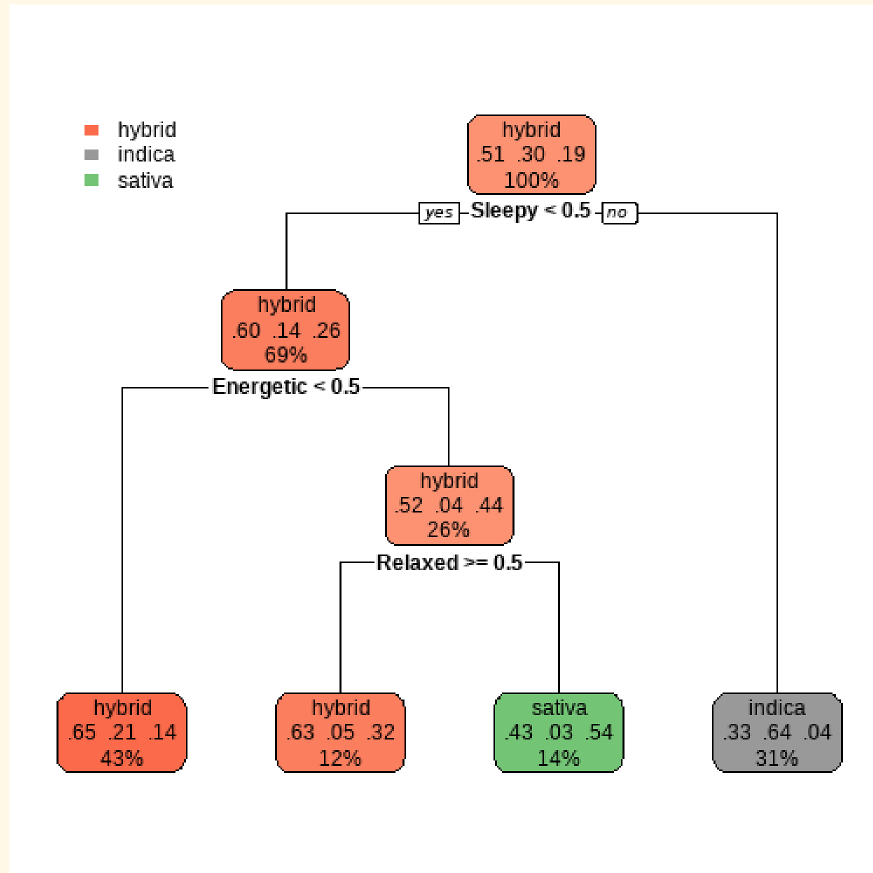
- Which of the final nodes (or leaves) is most pure?
- Which is least pure?

--

- Could we split a node further for better purity?
- almost certainly, yes! It's highly unlikely that *all* of the unused variable have *exactly* the same prevalance across categories.

--

- Should we do it, or is that overfitting?



Cost Complexity revisted

- What is the classification error of each leaf?

```
rpart.plot(tree_fitted$fit)
```

--

- (Left to right)
  - 0.35
  - 0.37
  - 0.46
  - 0.36

--

- The Gini Index for a particular node is the average of errors in each class:

$$(0.35 * 0.65) + (0.21 * 0.79) + (0.14 * 0.86) = 0.5138$$

```
+ small values if the classification errors are close to 0, i.e., high node purity
+ large values (near 1) if the errors are high
+ this is related to the *variance* of the node
```

Gini Index

To calculate the Gini Index average across all leaves:

```
cann %>%
  bind_cols(
    predict(tree_fitted, cann, type = "prob")
  ) %>%
  gain_capture(truth = Type,
               .pred_hybrid, .pred_indica, .pred_sativa)
```

```
## # A tibble: 1 x 3
##   .metric      .estimator .estimate
##   <chr>        <chr>          <dbl>
## 1 gain_capture macro          0.482
```

Cost Complexity revisited

So, when should we split the tree further?

Only if the new splits improve the Gini Index by a certain amount.

This is the `cost_complexity` parameter!

--

But wait! This is a **penalized** metric, using an arbitrary penalty α to avoid overfitting.

Don't we like cross-validation better?

Well... yes.

But imagine fitting *every possible tree* and cross-validating.... yikes.

We have to limit our options and cut our losses somehow!

Bagging

Tree variability

Suppose I took two random subsamples of my cannabis dataset:

```
set.seed(9374534)
splits <- cann %>%
  initial_split(0.5, strata = Type)

cann_1 <- splits %>% training()
cann_2 <- splits %>% testing()

dim(cann_1)
```

```
## [1] 1154   69
```

```
dim(cann_2)
```

```
## [1] 1151   69
```

Tree variability

Then I fit a **decision tree** to each:

```
tree_1 <- tree_wflow %>%
  fit(cann_1)

tree_2 <- tree_wflow %>%
  fit(cann_2)
```

How similar will the results be?

Tree variability

```
tree_1 %>%
  pull_workflow_fit() %$%
  fit %>%
  rpart.plot()
```

Tree variability

```
tree_2 %>%
  pull_workflow_fit() %$%
  fit %>%
  rpart.plot()
```

Tree variability

So... which should we believe?

Tree variability

Let's take several **subsamples** of the data, and make trees from each.

Then, to classify a new observation, we run it through *all* the trees and let them vote!

(It's a bit like a KNN for trees!)

This is called **bagging**

Bagging

```
library(baguette)

bag_tree_spec <- bag_tree() %>%
  set_engine("rpart", times = 5) %>%
  set_mode("classification")
```

Bagging

```
bag_tree_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(bag_tree_spec)

bag_tree_fit <- bag_tree_wflow %>%
  fit(cann)

## (this code may take a while!)
```

Bagging

What variables were most important to the trees?

```
bag_tree_fit %>% pull_workflow_fit()
```

Random Forests

Random Forests

What if some important variables are being masked by *more important* variables?

Remember, we have **65** predictors - yikes!

Let's give some of the other predictors a chance to shine.

Random Forests

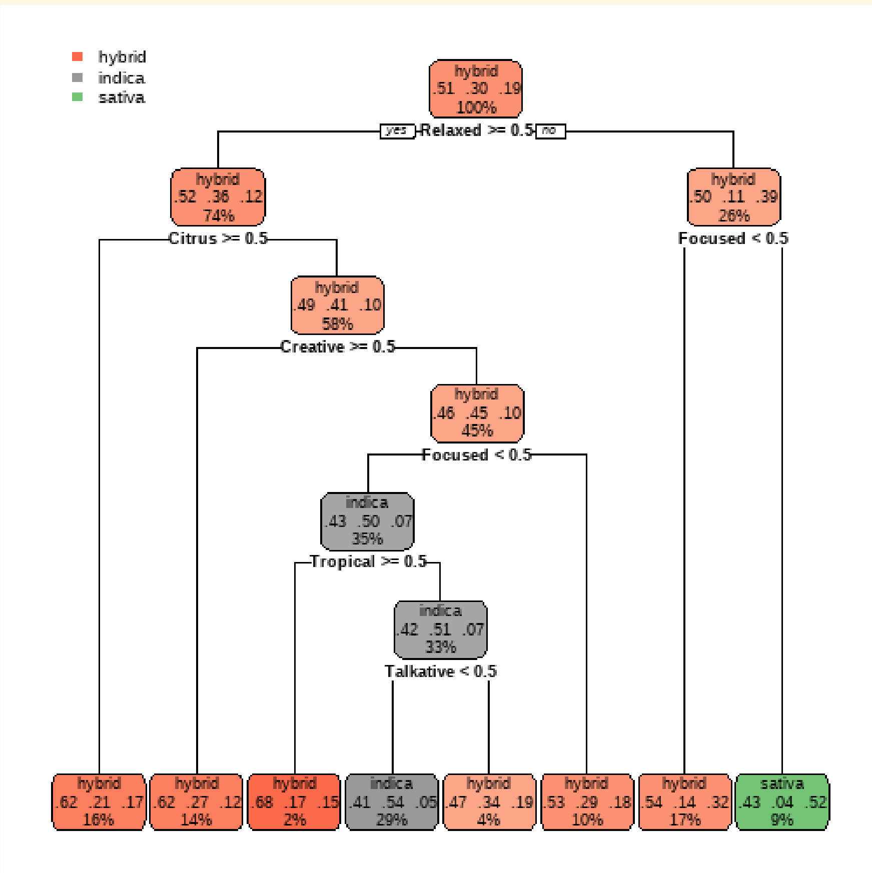
Randomly choose a few of the predictors to include in the data:

```
cann_reduced <- cann %>%
  select(1,2, sample(5:65, 30))

cann_reduced
```

```
## # A tibble: 2,305 x 32
##    Strain   Type  Tobacco   Dry Honey Ammonia Giggly Apple Relaxed Flowery Grape
##    <chr>    <fct>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>   <dbl>   <dbl> <dbl>
##  1 100-Og   hybr~       0     0     0       0      0     0       1       0     0
##  2 98-Whit~ hybr~       0     0     0       0      0     0       1       1     0
##  3 1024     sati~       0     0     0       0      0     0       1       0     0
##  4 13-Dawgs hybr~       0     0     0       0      0     0       1       0     1
##  5 24K-Gold hybr~       0     0     0       0      0     0       1       0     0
##  6 3-Bears~ indi~       0     0     0       0      0     0       0       0     0
##  7 3-Kings  hybr~       0     0     0       0      0     0       1       0     0
##  8 303-Og   indi~       0     0     0       0      1     0       1       0     0
##  9 3D-Cbd   sati~       0     0     0       0      0     0       1       1     0
## 10 3X-Crazy indi~       0     0     0       0      0     0       1       0     1
## # ... with 2,295 more rows, and 21 more variables: Tea <dbl>, Blueberry <dbl>,
## #   Creative <dbl>, Menthol <dbl>, Tropical <dbl>, Woody <dbl>, Earthy <dbl>,
## #   Skunk <dbl>, Citrus <dbl>, Mouth <dbl>, Berry <dbl>, Pungent <dbl>,
## #   Chestnut <dbl>, Blue <dbl>, Happy <dbl>, Vanilla <dbl>, Focused <dbl>,
```

Random Forests



Random Forests

After making many random reduced trees, we then **bag** the results to end up with a **random forest**.

The advantage of this is that more unique variables are involved in the process.

This way, we don't accidentally overfit to a variable that *happens* to be extremely relevant to our particular dataset.

Model spec: `rand_forest()`


I love democracy.

Your turn

Open the activity file

Find the best *bagged* model for the cannabis data

Find the best *random forest* model for the cannabis data

Report some metrics on your models