Decision Trees

class: center, middle, inverse

Decision Trees

Setup

Today's data concerns strains of cannabis, which have the types of `sativa`, `indica`, or `hybrid`:

```
## # A tibble: 6 x 69
##   Strain  Type  Rating Effects Flavor Creative Energetic Tingly Euphoric Relaxed
##   <chr>   <chr>  <dbl> <chr>   <chr>     <dbl>     <dbl>  <dbl>    <dbl>   <dbl>
## 1 100-Og  hybr~    4   Creati~ Earth~        1         1      1        1       1
## 2 98-Whi~ hybr~    4.7 Relaxe~ Flowe~        1         1      0        0       1
## 3 1024    sati~    4.4 Uplift~ Spicy~        1         1      0        0       1
## 4 13-Daw~ hybr~    4.2 Tingly~ Apric~        1         0      1        0       1
## 5 24K-Go~ hybr~    4.6 Happy,~ Citru~        0         0      0        1       1
## 6 3-Bear~ indi~    0   None    None          0         0      0        0       0
## # ... with 59 more variables: Aroused <dbl>, Happy <dbl>, Uplifted <dbl>,
## #   Hungry <dbl>, Talkative <dbl>, Giggly <dbl>, Focused <dbl>, Sleepy <dbl>,
## #   Dry <dbl>, Mouth <dbl>, Earthy <dbl>, Sweet <dbl>, Citrus <dbl>,
## #   Flowery <dbl>, Violet <dbl>, Diesel <dbl>, Spicy/Herbal <dbl>, Sage <dbl>,
## #   Woody <dbl>, Apricot <dbl>, Grapefruit <dbl>, Orange <dbl>, Pungent <dbl>,
## #   Grape <dbl>, Pine <dbl>, Skunk <dbl>, Berry <dbl>, Pepper <dbl>,
## #   Menthol <dbl>, Blue <dbl>, Cheese <dbl>, Chemical <dbl>, Mango <dbl>,
## #   Lemon <dbl>, Peach <dbl>, Vanilla <dbl>, Nutty <dbl>, Chestnut <dbl>,
## #   Tea <dbl>, Tobacco <dbl>, Tropical <dbl>, Strawberry <dbl>,
## #   Blueberry <dbl>, Mint <dbl>, Apple <dbl>, Honey <dbl>, Lavender <dbl>,
## #   Lime <dbl>, Coffee <dbl>, Ammonia <dbl>, Minty <dbl>, Tree <dbl>,
## #   Fruit <dbl>, Butter <dbl>, Pineapple <dbl>, Tar <dbl>, Rose <dbl>,
## #   Plum <dbl>, Pear <dbl>
```

Setup

```
cann <- cann %>%
  mutate(
    Type = factor(Type)
  ) %>%
  drop_na()

cann_cvs <- vfold_cv(cann, v = 5)

cann_recipe <- recipe(Type ~ .,
                      data = cann) %>%
  step_rm(Strain, Effects, Flavor)
```

Setup

```
cann_recipe
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         68
##
## Operations:
##
## Delete terms Strain, Effects, Flavor
```

Logistic Regression

```
logit_mod <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

logit_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(logit_mod)

logit_fit <- logit_wflow %>%
  fit_resamples(cann_cvs)
```

```
## ! Fold1: model (predictions): prediction from a rank-deficient fit may be misleading

## x Fold1: internal: Error: In metric: `roc_auc`

## ! Fold2: model (predictions): prediction from a rank-deficient fit may be misleading

## x Fold2: internal: Error: In metric: `roc_auc`

## ! Fold3: model (predictions): prediction from a rank-deficient fit may be misleading

## x Fold3: internal: Error: In metric: `roc_auc`

## ! Fold4: model (predictions): prediction from a rank-deficient fit may be misleading
```

Logistic Regression

Problem 1: The model is trying to fit 65 predictor coefficients! That's a LOT.

Problem 2: There are three categories in `Type`. How do we interpret the log-odds for multiple groups?

Discriminant Analysis

```
lda_mod <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

lda_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(lda_mod)

lda_fit <- lda_wflow %>%
  fit_resamples(cann_cvs)
```

```
## ! Fold1: model: variables are collinear

## x Fold2: model: Error in lda.default(x, grouping, ...): variables 15 16 appear to...

## ! Fold3: model: variables are collinear

## ! Fold4: model: variables are collinear

## ! Fold5: model: variables are collinear
```

Discriminant Analysis

Problem: There are still 65 predictors, i.e., 65 dimensions!

Some of these contain duplicate information.

```
## # A tibble: 2,305 x 2
##      Dry Mouth
##    <dbl> <dbl>
##  1     1     1
##  2     0     0
##  3     0     0
##  4     0     0
##  5     0     0
##  6     0     0
##  7     0     0
##  8     0     0
##  9     0     0
## 10     0     0
## # ... with 2,295 more rows
```

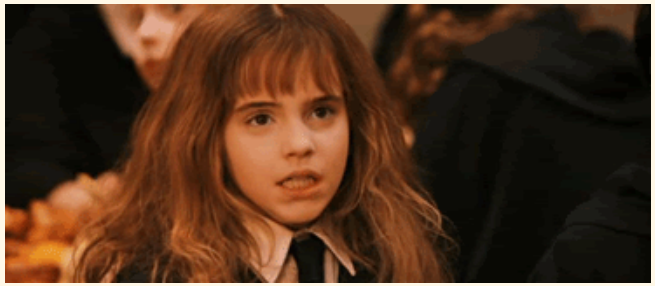
KNN

```
knn_mod <- nearest_neighbor(neighbors = 5) %>%
  set_engine("kknn") %>%
  set_mode("classification")

knn_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(knn_mod)

knn_fit <- knn_wflow %>%
  fit_resamples(cann_cvs)
```

KNN

```
knn_fit <- knn_wflow %>%
  fit_resamples(cann_cvs,
                metrics = metric_set(accuracy, roc_auc, precision, recall))


knn_fit %>% collect_metrics()
```

```
## # A tibble: 4 x 5
##   .metric   .estimator  mean     n std_err
##   <chr>     <chr>      <dbl> <int>   <dbl>
## 1 accuracy  multiclass 0.486     5 0.00908
## 2 precision macro      0.453     5 0.0111
## 3 recall    macro      0.455     5 0.0113
## 4 roc_auc   hand_till  0.668     5 0.00600
```



Decision Trees

Let's play 20 questions.

- Is this strain described as "Energetic"?

Let's play 20 questions.

- Is this strain described as tasting like "Pineapple"?


I think of
"Sativa"
© Copyright / IP Policy
Yes
No

Decision Trees

```
tree_mod <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("classification")

tree_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(tree_mod)
```

Decision Trees

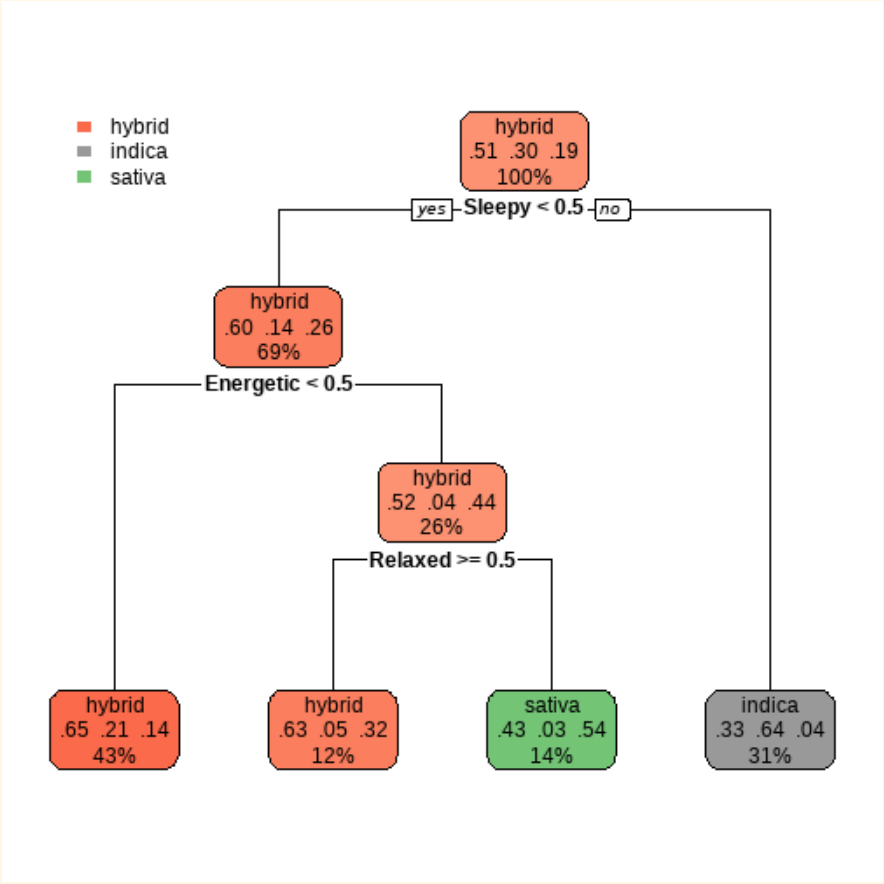
```
tree_fit <- tree_wflow %>%
  fit_resamples(cann_cvs,
                metrics = metric_set(accuracy, roc_auc, precision, recall))


tree_fit %>% collect_metrics()
```

```
## # A tibble: 4 x 5
##   .metric   .estimator  mean     n std_err
##   <chr>     <chr>      <dbl> <int>   <dbl>
## 1 accuracy  multiclass 0.623     5 0.00797
## 2 precision macro      0.597     5 0.00957
## 3 recall    macro      0.572     5 0.0110
## 4 roc_auc   hand_till  0.750     5 0.00631
```

Decision Trees

```
## $actions
## $actions$model
## $spec
## Decision Tree Model Specification (classification)
##
## Computational engine: rpart
##
##
## $formula
## NULL
##
## attr(,"class")
## [1] "action_model" "action_fit"   "action"
##
##
## $fit
## parsnip model object
##
## Fit time:  311ms
## n= 2305
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2305 1118 hybrid (0.51496746 0.29804772 0.18698482)
```

Decision Trees



Decision Trees

(code for that plot, requires `rpart.plot` package.)

```
tree_fitted <- tree_fit_1 %>%
  pull_workflow_fit()

rpart.plot(tree_fitted$fit)
```

What might we change?

Decision Trees - Hyperparameters

- `tree_depth`: How many splits will we "allow" the tree to make?
  - If we allowed infinite splits, we'd end up with only on observation in each "leaf".... overfitting!.
  - If we allow only one split, our accuracy won't be that great.
  - Default in `rpart`: Up to 30
- `min_n`: How many observations have to be in a "leaf" for us to be allowed to split it further?
  - If `min_n` is too small, we're overfitting.
  - If `min_n` is too big, we're not allowing enough flexibility.
  - Default in `rpart`: 20

Decision Trees - Hyperparameters

Let's try varying `min_n` between 2 and 20.

Tuning with cross-validation takes a long time! Do yourself a favor and start with a wide grid.

```
tree_grid <- grid_regular(min_n(c(2,20)),
                          levels = 4)

tree_grid
```

```
## # A tibble: 4 x 1
##   min_n
##   <int>
## 1     2
## 2     8
## 3    14
## 4    20
```

Decision Trees - Hyperparameters

```
tree_mod <- decision_tree(min_n = tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

tree_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(tree_mod)

tree_grid_search <-
  tune_grid(
    tree_wflow,
    resamples = cann_cvs,
    grid = tree_grid
  )

tuning_metrics <- tree_grid_search %>% collect_metrics()
```

Decision Trees - Hyperparameters

```
tuning_metrics
```

```
## # A tibble: 8 x 7
##   min_n .metric  .estimator  mean     n std_err .config
##   <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1     2 accuracy multiclass 0.623     5 0.00797 Model1
## 2     2 roc_auc  hand_till  0.750     5 0.00631 Model1
## 3     8 accuracy multiclass 0.623     5 0.00797 Model2
## 4     8 roc_auc  hand_till  0.750     5 0.00631 Model2
## 5    14 accuracy multiclass 0.623     5 0.00797 Model3
## 6    14 roc_auc  hand_till  0.750     5 0.00631 Model3
## 7    20 accuracy multiclass 0.623     5 0.00797 Model4
## 8    20 roc_auc  hand_till  0.750     5 0.00631 Model4
```

What else can we change?

How is `rpart` choosing to stop splitting?

- cost complexity = how much metric gain is "worth it" to do another split?
    - Default: Split must increase `accuracy` by at least 0.01.

Cost complexity

```
tree_grid <- grid_regular(cost_complexity(),
                          tree_depth(),
                          min_n(),
                          levels = 2)

tree_grid
```

```
## # A tibble: 8 x 3
##   cost_complexity tree_depth min_n
##             <dbl>      <int> <int>
## 1    0.0000000001          1     2
## 2    0.1                   1     2
## 3    0.0000000001         15     2
## 4    0.1                  15     2
## 5    0.0000000001          1    40
## 6    0.1                   1    40
## 7    0.0000000001         15    40
## 8    0.1                  15    40
```

Cost Complexity

```
tree_mod <- decision_tree(cost_complexity = tune(),
                          tree_depth = tune(),
                          min_n = tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

tree_wflow <- workflow() %>%
  add_recipe(cann_recipe) %>%
  add_model(tree_mod)

tree_grid_search <-
  tune_grid(
    tree_wflow,
    resamples = cann_cvs,
    grid = tree_grid
  )

tuning_metrics <- tree_grid_search %>% collect_metrics()
```

Tuning

```
tuning_metrics
```

```
## # A tibble: 16 x 9
##    cost_complexity tree_depth min_n .metric  .estimator  mean     n std_err
##              <dbl>      <int> <int> <chr>    <chr>      <dbl> <int>   <dbl>
##  1    0.0000000001          1     2 accuracy multiclass 0.613     5 0.0122
##  2    0.0000000001          1     2 roc_auc  hand_till  0.680     5 0.0103
##  3    0.1                   1     2 accuracy multiclass 0.613     5 0.0122
##  4    0.1                   1     2 roc_auc  hand_till  0.680     5 0.0103
##  5    0.0000000001         15     2 accuracy multiclass 0.546     5 0.00805
##  6    0.0000000001         15     2 roc_auc  hand_till  0.638     5 0.0160
##  7    0.1                  15     2 accuracy multiclass 0.613     5 0.0122
##  8    0.1                  15     2 roc_auc  hand_till  0.680     5 0.0103
##  9    0.0000000001          1    40 accuracy multiclass 0.613     5 0.0122
## 10    0.0000000001          1    40 roc_auc  hand_till  0.680     5 0.0103
## 11    0.1                   1    40 accuracy multiclass 0.613     5 0.0122
## 12    0.1                   1    40 roc_auc  hand_till  0.680     5 0.0103
## 13    0.0000000001         15    40 accuracy multiclass 0.588     5 0.0130
## 14    0.0000000001         15    40 roc_auc  hand_till  0.753     5 0.0129
## 15    0.1                  15    40 accuracy multiclass 0.613     5 0.0122
## 16    0.1                  15    40 roc_auc  hand_till  0.680     5 0.0103
## # ... with 1 more variable: .config <chr>
```

Tuning

```
tuning_metrics %>%
  filter(.metric == "accuracy") %>%
  slice_max(mean)
```

```
## # A tibble: 6 x 9
##   cost_complexity tree_depth min_n .metric  .estimator  mean     n std_err
##             <dbl>      <int> <int> <chr>    <chr>      <dbl> <int>   <dbl>
## 1    0.0000000001          1     2 accuracy multiclass 0.613     5  0.0122
## 2    0.1                   1     2 accuracy multiclass 0.613     5  0.0122
## 3    0.1                  15     2 accuracy multiclass 0.613     5  0.0122
## 4    0.0000000001          1    40 accuracy multiclass 0.613     5  0.0122
## 5    0.1                   1    40 accuracy multiclass 0.613     5  0.0122
## 6    0.1                  15    40 accuracy multiclass 0.613     5  0.0122
## # ... with 1 more variable: .config <chr>
```

```
tuning_metrics %>%
  filter(.metric == "roc_auc") %>%
  slice_max(mean)
```

```
## # A tibble: 1 x 9
##   cost_complexity tree_depth min_n .metric .estimator  mean     n std_err
##             <dbl>      <int> <int> <chr>   <chr>      <dbl> <int>   <dbl>
## 1    0.0000000001         15    40 roc_auc hand_till  0.753     5  0.0129
```

Try it!

Open Activity-Decision-Tree

Fit a final model with the selected hyperparameters

Report some metrics for the final model

Plot the tree (code is provided)

Interpret the first two levels of splits in plain English.