Classification and Regression Trees

Nate Wells

Math 243: Stat Learning

November 12th, 2021

Outline

In today's class, we will...

- Discuss classification trees for classification problems.
- Build handmade classification tree models

Section 1

Classification Trees

Classification Trees

Classification trees are very similar to regression trees, except the terminal nodes predict levels of a categorical variable, rather than values of a quantitative variable

Classification Trees

Classification trees are very similar to regression trees, except the terminal nodes predict levels of a categorical variable, rather than values of a quantitative variable

- To *grow* a classification tree, we need to make cuts based on a metric other than RSS (why?)

Classification Trees

Classification trees are very similar to regression trees, except the terminal nodes predict levels of a categorical variable, rather than values of a quantitative variable

- To *grow* a classification tree, we need to make cuts based on a metric other than RSS (why?)
- For each split candidate, we average the value of the metric on the two proposed subregions, and select the split that minimizes the average value of the metric.

Classification Trees

Classification trees are very similar to regression trees, except the terminal nodes predict levels of a categorical variable, rather than values of a quantitative variable

- To *grow* a classification tree, we need to make cuts based on a metric other than RSS (why?)
- For each split candidate, we average the value of the metric on the two proposed subregions, and select the split that minimizes the average value of the metric.
- The most natural choice is to use *Classification Error Rate E* (i.e. proportion of obs. in region not in most common class)

$$E = 1 - \max_k(p_{mk}) \quad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

Classification Trees

Classification trees are very similar to regression trees, except the terminal nodes predict levels of a categorical variable, rather than values of a quantitative variable

- To *grow* a classification tree, we need to make cuts based on a metric other than RSS (why?)
- For each split candidate, we average the value of the metric on the two proposed subregions, and select the split that minimizes the average value of the metric.
- The most natural choice is to use *Classification Error Rate E* (i.e. proportion of obs. in region not in most common class)

$$E = 1 - \max_k(p_{mk}) \quad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - But because of the greedy algorithm used to split trees, E tends to overfit to noise in the training data

Other Decision Metric

Two common alternatives for decision metric:

Other Decision Metric

Two common alternatives for decision metric:

- The *Gini index* G:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

Other Decision Metric

Two common alternatives for decision metric:

- The *Gini index* G:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the rate that a random element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the region

Other Decision Metric

Two common alternatives for decision metric:

- The *Gini index* G:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the rate that a random element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the region
  - The Gini index is small if all $\hat{p}_{mk}$ are close to 0 or 1.

- The *information* or *entropy* D:

$$D = -\sum_{i=1}^{K} \hat{p}_{mk} \log_2 \hat{p}_{mk} \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

Other Decision Metric

Two common alternatives for decision metric:

- The *Gini index* G:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the rate that a random element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the region
  - The Gini index is small if all $\hat{p}_{mk}$ are close to 0 or 1.

- The *information* or *entropy* D:

$$D = -\sum_{i=1}^{K} \hat{p}_{mk} \log_2 \hat{p}_{mk} \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the average amount of information conveyed by knowing the region of an observation.

Other Decision Metric

Two common alternatives for decision metric:

- The *Gini index* G:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the rate that a random element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the region
  - The Gini index is small if all $\hat{p}_{mk}$ are close to 0 or 1.

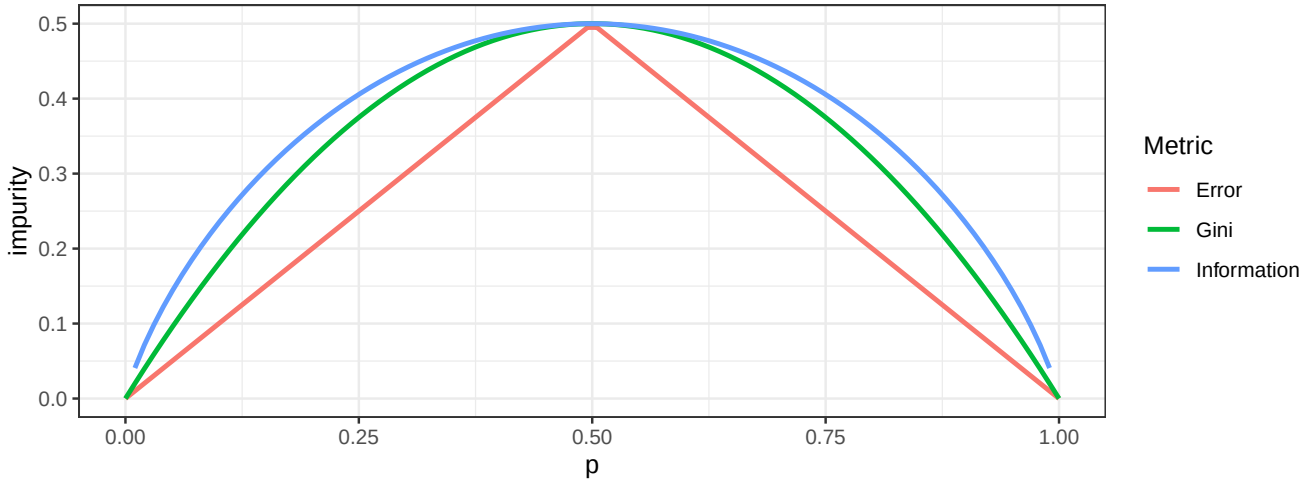
- The *information* or *entropy* D:

$$D = -\sum_{i=1}^{K} \hat{p}_{mk} \log_2 \hat{p}_{mk} \qquad \text{where } \hat{p}_{mk} = \text{prop. obs. in region m in class k}$$

  - It measures the average amount of information conveyed by knowing the region of an observation.
  - The entropy is small if all $\hat{p}_{mk}$ are close to 0 or 1.

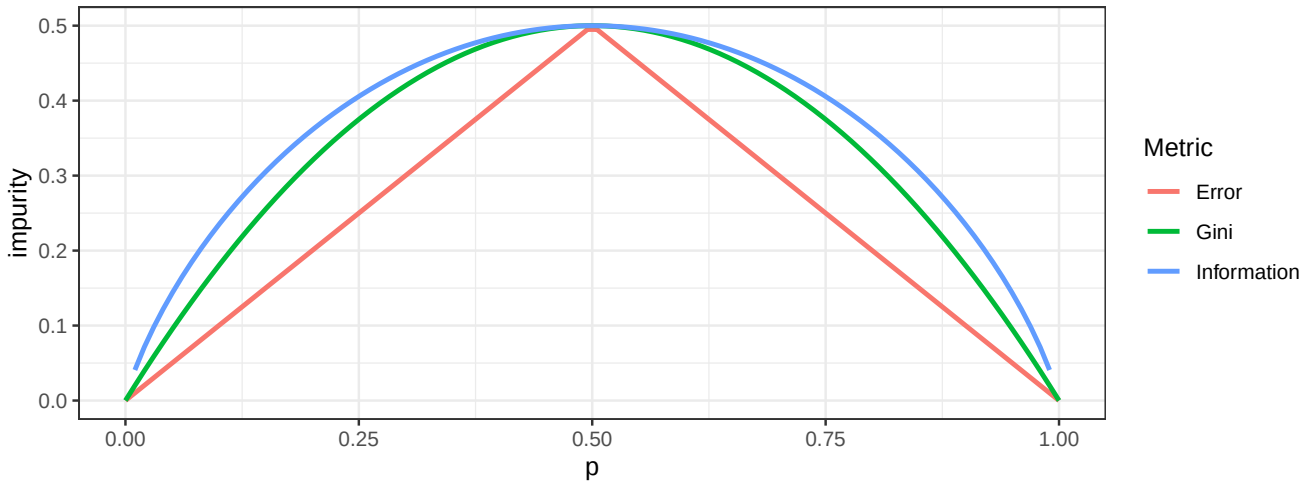
Metrics

- The following plot demonstrates sensitivity of metrics E, G, D to changes in class proportion p.



Metrics

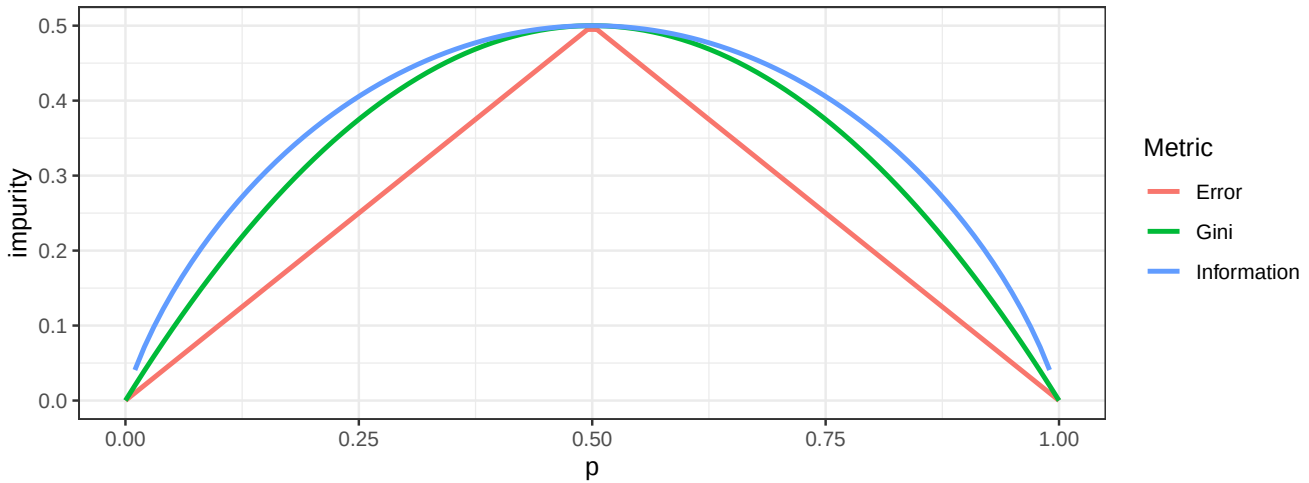
- The following plot demonstrates sensitivity of metrics E, G, D to changes in class proportion p.



- The Gini Index and Information are both more sensitive to changes in node purity than Error (represented by convexity of curves)

Metrics

- The following plot demonstrates sensitivity of metrics E, G, D to changes in class proportion p.



- The Gini Index and Information are both more sensitive to changes in node purity than Error (represented by convexity of curves)
  - Suppose we have an initial class balance of [300, 500] and make a single split into nodes [0,100] and [300,400]
  - The misclassification rate is constant, although node purity has increased

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.
- But this conversion has a significant downside! The algorithm is biased toward making early splits on categorical variables with many levels.

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.
- But this conversion has a significant downside! The algorithm is biased toward making early splits on categorical variables with many levels.
    - Since trees are already prone to high variance, this additional bias can lead to unwanted increases in MSE.

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.
- But this conversion has a significant downside! The algorithm is biased toward making early splits on categorical variables with many levels.
  - Since trees are already prone to high variance, this additional bias can lead to unwanted increases in MSE.
- The "simple" fix is to lump together levels before building a tree, using domain knowledge

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.
- But this conversion has a significant downside! The algorithm is biased toward making early splits on categorical variables with many levels.
  - Since trees are already prone to high variance, this additional bias can lead to unwanted increases in MSE.
- The "simple" fix is to lump together levels before building a tree, using domain knowledge
- An alternative is to allow the model algorithm to lump together values as necessary at each node (order levels in increasing frequency, then make appropriate cut)

Dealing with Categorical Variables

Both regression and classification trees can easily hand either quantitative or binary categorical variables.

- But with some modification, trees can also be used with multi-level categorical variables.
- To do so, we recode all multilevel categorical variables as a sequence of dummy binary variables. Then proceed as usual.
- But this conversion has a significant downside! The algorithm is biased toward making early splits on categorical variables with many levels.
  - Since trees are already prone to high variance, this additional bias can lead to unwanted increases in MSE.
- The "simple" fix is to lump together levels before building a tree, using domain knowledge
- An alternative is to allow the model algorithm to lump together values as necessary at each node (order levels in increasing frequency, then make appropriate cut)
  - But this generally leads to less interpretable models

Section 2

Classification Trees in R

Mushroom Hunting

Mushroom Hunting

Can I eat this?

Mushrooms

- The `mushrooms` data set contains information on `edibility` and 22 other features on 8124 samples of Mushrooms. We'll do a 80-20 training-test split.

Mushrooms

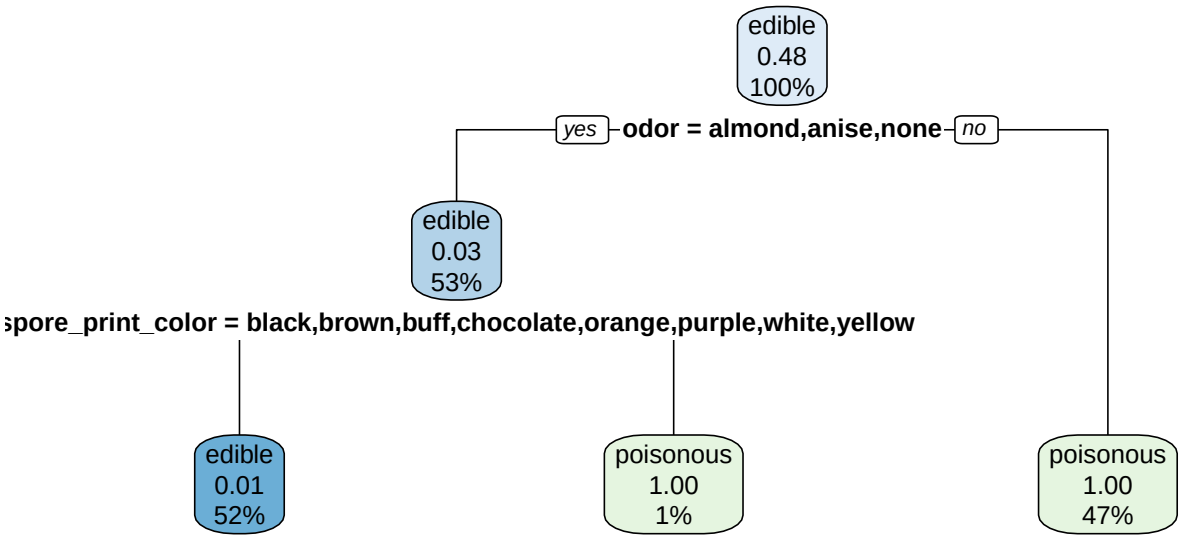
- The `mushrooms` data set contains information on `edibility` and 22 other features on 8124 samples of Mushrooms. We'll do a 80-20 training-test split.

```
## Rows: 6,498
## Columns: 23
## $ edibility                <fct> edible, edible, edible, edible, edible, edibl~
## $ cap_shape                <fct> convex, bell, convex, convex, bell, bell, bel~
## $ cap_surface              <fct> scaly, scaly, scaly, smooth, scaly, smooth, s~
## $ cap_color                <fct> yellow, white, gray, yellow, white, white, ye~
## $ bruises                  <fct> yes, yes, no, yes, yes, yes, yes, yes, yes, y~
## $ odor                     <fct> almond, anise, none, almond, almond, anise, a~
## $ gill_attachement         <fct> free, free, free, free, free, free, free, fre~
## $ gill_spacing             <fct> close, close, crowded, close, close, close, c~
## $ gill_size                <fct> broad, broad, broad, broad, broad, broad, bro~
## $ gill_color               <fct> black, brown, black, brown, gray, brown, gray~
## $ stalk_shape              <fct> enlarging, enlarging, tapering, enlarging, en~
## $ stalk_root               <fct> club, club, equal, club, club, club, club, cl~
## $ stalk_surface_above_ring <fct> smooth, smooth, smooth, smooth, smooth, smoot~
## $ stalk_surface_below_ring <fct> smooth, smooth, smooth, smooth, smooth, smoot~
## $ stalk_color_above_ring   <fct> purple, purple, purple, purple, purple, purpl~
## $ stalk_color_below_ring   <fct> purple, purple, purple, purple, purple, purpl~
## $ veil_type                <fct> partial, partial, partial, partial, partial, ~
## $ veil_color               <fct> white, white, white, white, white, white, whi~
## $ ring_number              <fct> one, one, one, one, one, one, one, one, one, ~
## $ ring_type                <fct> pendant, pendant, evanescent, pendant, pendan~
## $ spore_print_color        <fct> brown, brown, brown, black, black, brown, bla~
## $ population               <fct> numerous, numerous, abundant, numerous, numer~
## $ habitat                  <fct> grasses, meadows, grasses, grasses, meadows, ~
```

Implementing classfication trees in R

As with regression trees, we use the ‘rpart“ package.

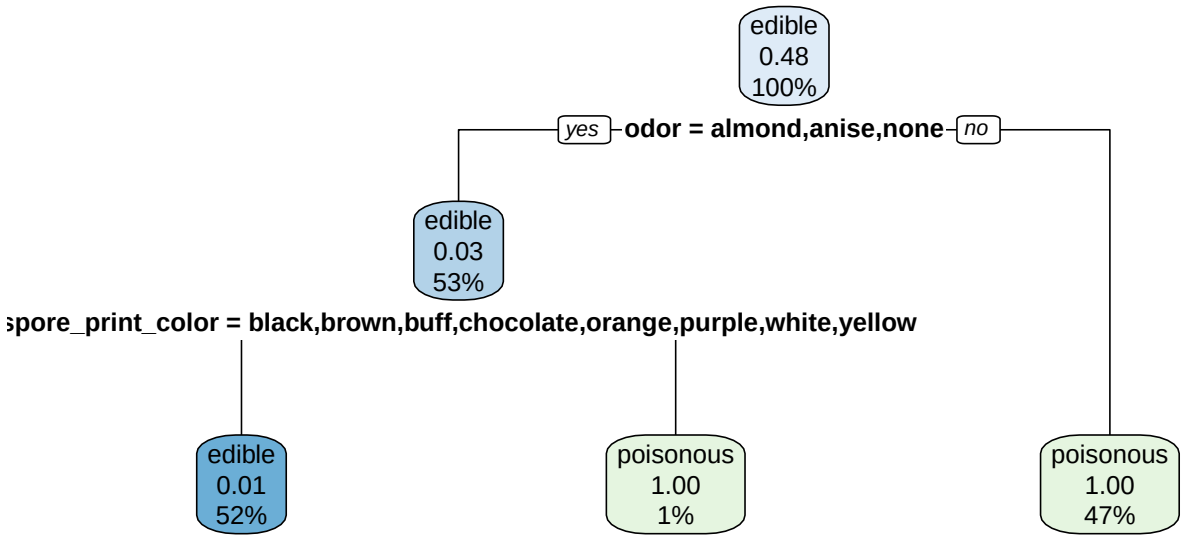
```
library(rpart)
library(rpart.plot)
mushroom_tree<-rpart(edibility ~ ., data = mushrooms_train)
rpart.plot(mushroom_tree)
```



Implementing classfication trees in R

As with regression trees, we use the ‘rpart“ package.

```
library(rpart)
library(rpart.plot)
mushroom_tree<-rpart(edibility ~ ., data = mushrooms_train)
rpart.plot(mushroom_tree)
```



- The default parameters created data with relatively few terminal nodes.
  - And it seems like we obtained good class purity!

Model Accuracy

- How well did we do on test data?

Model Accuracy

- How well did we do on test data?

```
library(yardstick)
mushroom_preds <- predict(mushroom_tree, mushrooms_test, type = "class")
mushroom_probs <- predict(mushroom_tree, mushrooms_test, type = "prob")[,"edible"]

results <- data.frame(obs = mushrooms_test$edibility,preds = mushroom_preds,
                      probs = mushroom_probs)

accuracy(results, truth = obs, estimate = preds)
```

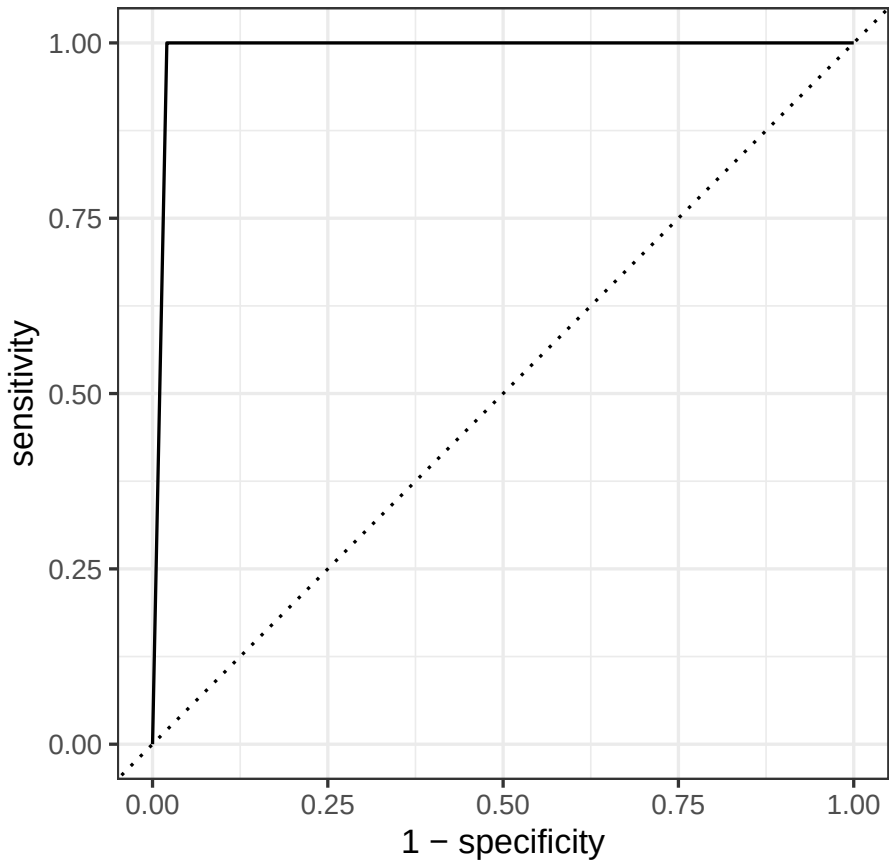
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.990
```

- Looks like we have fantastic accuracy!

ROC Curve

Look at that ROC curve!

```
roc_curve(results, truth = obs, probs) %>%
autoplot()
```



Confusion Matrix

- Just one more thing to check:

Confusion Matrix

- Just one more thing to check:

```
conf_mat(results, truth = obs, estimate = preds)
```

Confusion Matrix

- Just one more thing to check:

```
conf_mat(results, truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       842        16
##   poisonous      0       768
```

Confusion Matrix

- Just one more thing to check:

```
conf_mat(results, truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       842        16
##   poisonous      0       768
```

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!
  - Downside: No tasty mushrooms :(
- *Option 2:* change classification threshhold

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!
  - Downside: No tasty mushrooms :(
- *Option 2:* change classification threshhold
  - I.e. classify as edible only if $P(\text{edible}) > 99.9\%$

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!
  - Downside: No tasty mushrooms :(
- *Option 2:* change classification threshhold
  - I.e. classify as edible only if $P(\text{edible}) > 99.9\%$
- *Option 3:* Incorporate relative loss in Gini index.

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!
  - Downside: No tasty mushrooms :(
- *Option 2:* change classification threshhold
  - I.e. classify as edible only if $P(\text{edible}) > 99.9\%$
- *Option 3:* Incorporate relative loss in Gini index.

$$G = \sum_i \sum_j L(i,j) p_i p_j$$

Improvements

How can we reduce the **type II error** of our classifier? (rate of poison mushrooms identified as `edible`)

- *Option 1:* Everything is poisonous!
  - Downside: No tasty mushrooms :(
- *Option 2:* change classification threshhold
  - I.e. classify as edible only if $P(\text{edible}) > 99.9\%$
- *Option 3:* Incorporate relative loss in Gini index.

$$G = \sum_i \sum_j L(i,j) p_i p_j$$

- Here, $L(i,j)$ is the loss occurred when predicting level j when the truth is level i.

Additional Parameters

- To incorporate loss, create a penalty matrix and add to the `parms` argument in `rpart`:

```
penalty_matrix <- matrix(c(0,1,20,0), byrow = T, nrow = 2)
penalty_matrix
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]   20    0
```

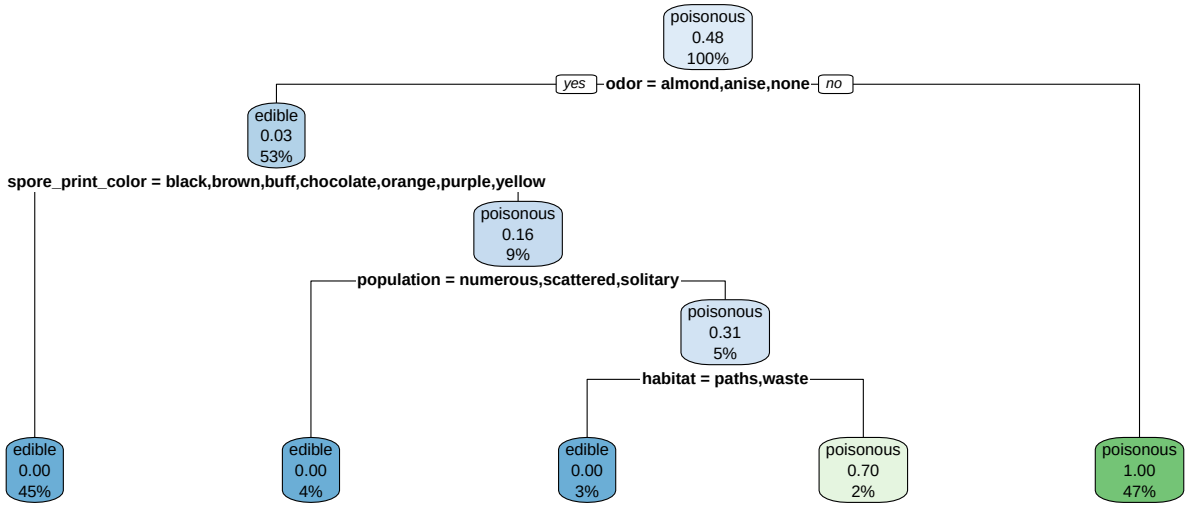
Additional Parameters

- To incorporate loss, create a penalty matrix and add to the `parms` argument in `rpart`:

```
penalty_matrix <- matrix(c(0,1,20,0), byrow = T, nrow = 2)
penalty_matrix
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]   20    0
```

```
mushroom_no_poison <- rpart(edibility ~., data = mushrooms_train,
                            parms = list(loss = penalty_matrix))
rpart.plot(mushroom_no_poison)
```



New Results

- Now how did we do?

New Results

- Now how did we do?

```
results %>% group_by(model) %>% accuracy( truth = obs, estimate = preds)
```

```
## # A tibble: 2 x 4
##   model        .metric  .estimator .estimate
##   <chr>        <chr>    <chr>          <dbl>
## 1 with loss    accuracy binary         0.994
## 2 without loss accuracy binary         0.990
```

New Results

- Now how did we do?

```
results %>% group_by(model) %>% accuracy( truth = obs, estimate = preds)
```

```
## # A tibble: 2 x 4
##   model        .metric  .estimator .estimate
##   <chr>        <chr>    <chr>          <dbl>
## 1 with loss    accuracy binary         0.994
## 2 without loss accuracy binary         0.990
```

```
results %>% filter(model == "with loss") %>% conf_mat(truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       833         0
##   poisonous      9       784
```

New Results

- Now how did we do?

```
results %>% group_by(model) %>% accuracy( truth = obs, estimate = preds)
```

```
## # A tibble: 2 x 4
##   model        .metric  .estimator .estimate
##   <chr>        <chr>    <chr>          <dbl>
## 1 with loss    accuracy binary         0.994
## 2 without loss accuracy binary         0.990
```

```
results %>% filter(model == "with loss") %>% conf_mat(truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       833         0
##   poisonous      9       784
```

- But can we now improve that Type I error?

New Results

- Now how did we do?

```
results %>% group_by(model) %>% accuracy( truth = obs, estimate = preds)
```

```
## # A tibble: 2 x 4
##   model        .metric  .estimator .estimate
##   <chr>        <chr>    <chr>          <dbl>
## 1 with loss    accuracy binary         0.994
## 2 without loss accuracy binary         0.990
```

```
results %>% filter(model == "with loss") %>% conf_mat(truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       833         0
##   poisonous      9       784
```

- But can we now improve that Type I error?
  - To reclaim some of those "poisonous" mushrooms, we'll need to build a deeper tree.

Deeper Trees

- We can control tree depth by setting the minimum `cp` parameter in `rpart.control`

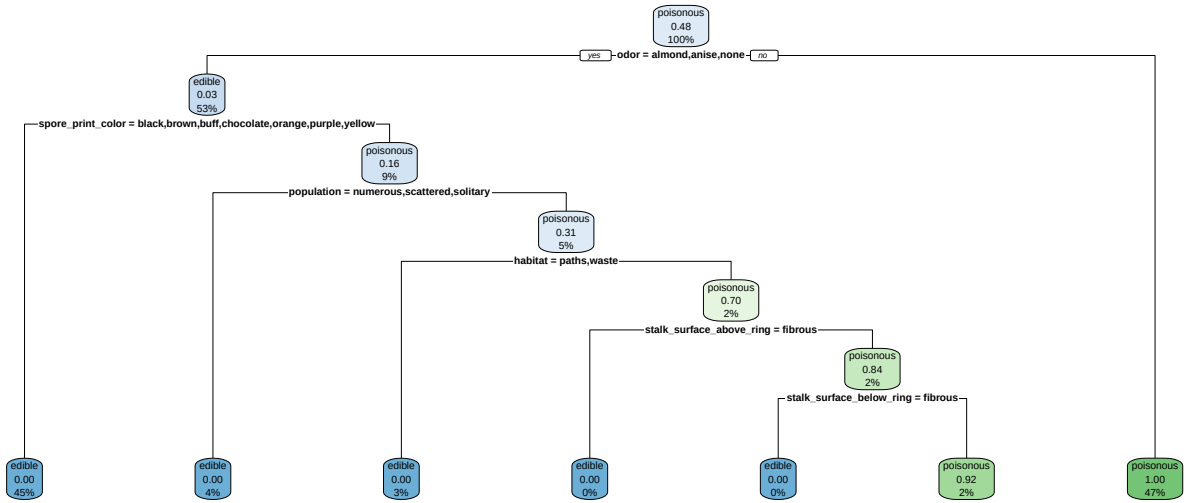
Deeper Trees

- We can control tree depth by setting the minimum `cp` parameter in `rpart.control`
  - Any split that does not decrease overall lack of fit by a factor of `cp` is not attemped.
  - Setting low values of `cp` lead to deeper trees

Deeper Trees

- We can control tree depth by setting the minimum cp parameter in `rpart.control`
  - Any split that does not decrease overall lack of fit by a factor of cp is not attemped.
  - Setting low values of cp lead to deeper trees

```
mushroom_deep <- rpart(edibility ~., data = mushrooms_train,
                       parms = list(loss = penalty_matrix),
                       control = rpart.control(cp = .00001))
rpart.plot(mushroom_deep)
```

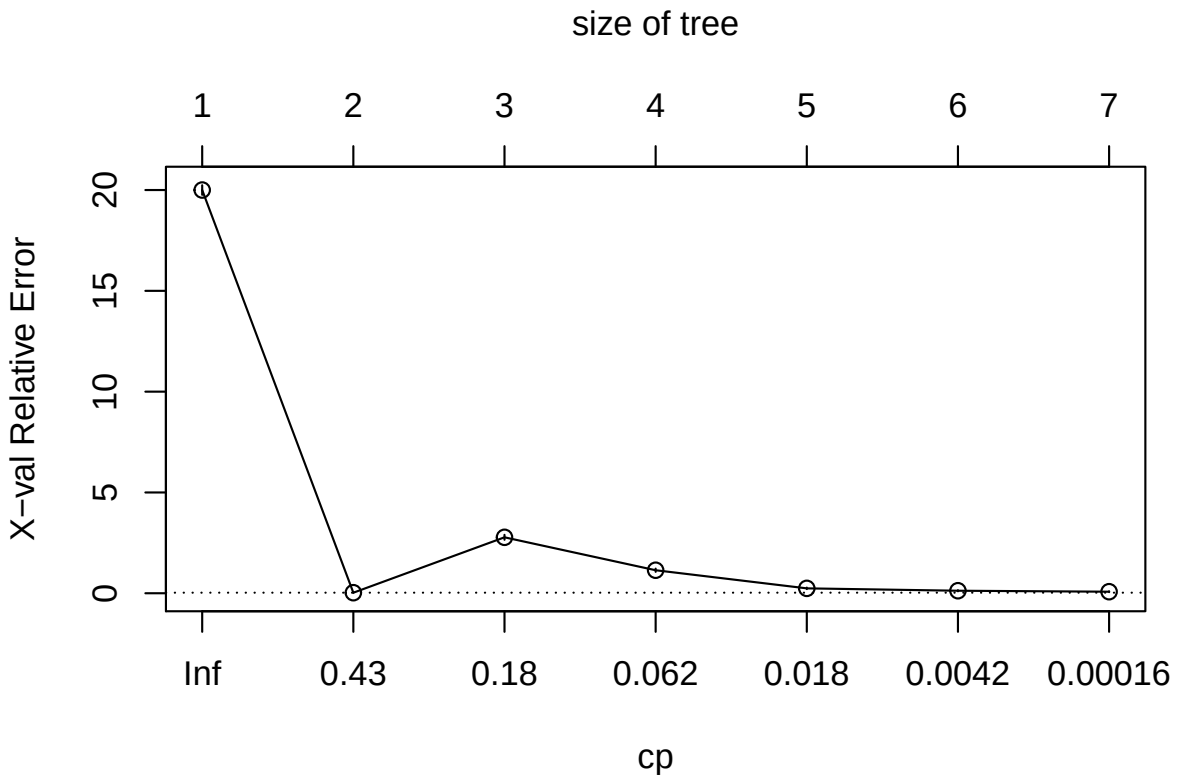


Pruning

- Let's look at cross-validated relative error

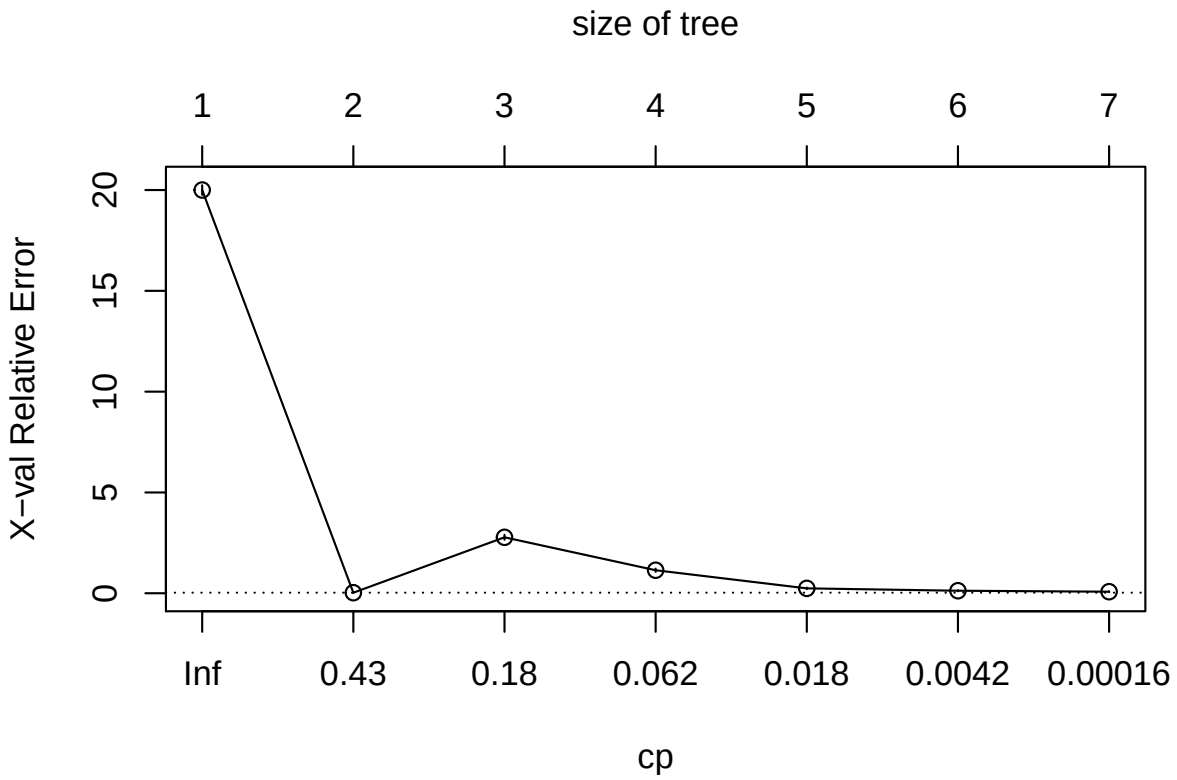
Pruning

- Let's look at cross-validated relative error



Pruning

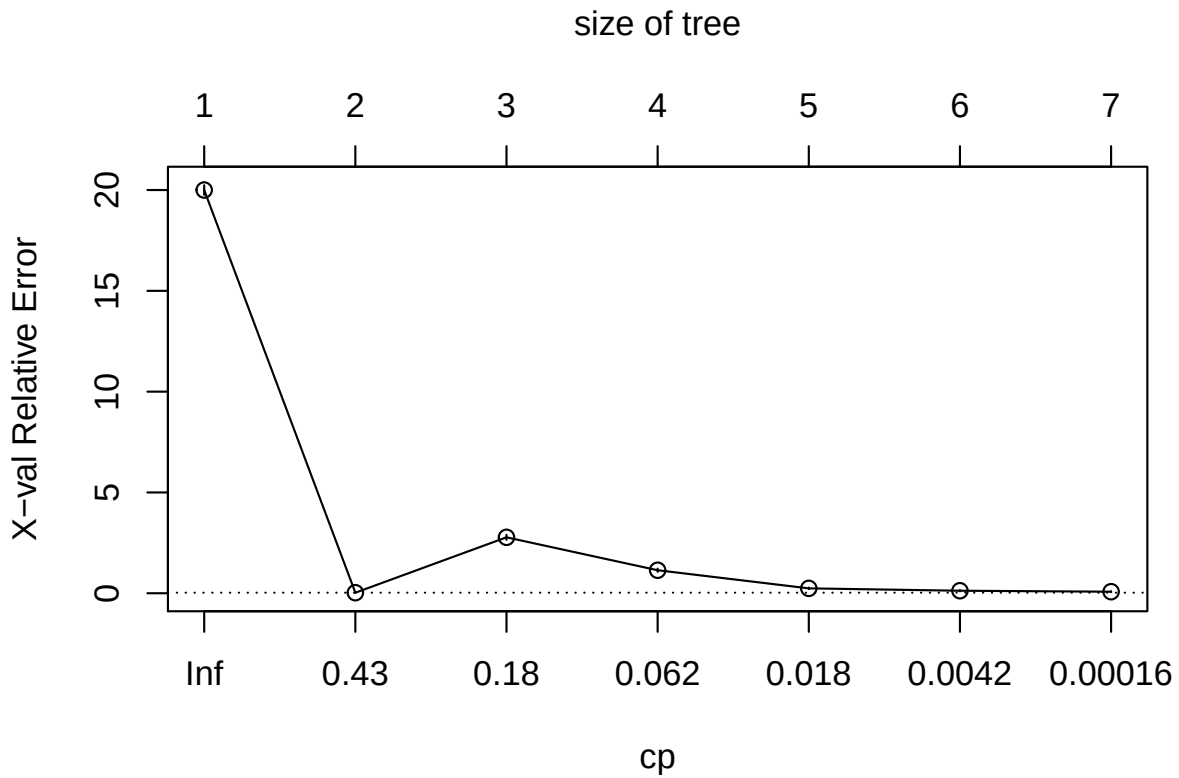
- Let's look at cross-validated relative error



- It's possible we are now overfitting. It may be best to reduce to tree with 6 leaves.

Pruning

- Let's look at cross-validated relative error



- It's possible we are now overfitting. It may be best to reduce to tree with 6 leaves.

```
mushroom_prune <- prune(mushroom_deep, cp = 0.0042)
```

Final Results

- How do our deep and pruned models do?

Final Results

- How do our deep and pruned models do?

```
results %>% group_by(model) %>% accuracy( truth = obs, estimate = preds)
```

```
## # A tibble: 4 x 4
##   model        .metric  .estimator .estimate
##   <chr>        <chr>    <chr>          <dbl>
## 1 deep         accuracy binary         0.998
## 2 pruned       accuracy binary         0.996
## 3 with loss    accuracy binary         0.994
## 4 without loss accuracy binary         0.990
```

```
results %>% filter(model == "deep") %>% conf_mat(truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       838         0
##   poisonous      4       784
```

```
results %>% filter(model == "pruned") %>% conf_mat(truth = obs, estimate = preds)
```

```
##            Truth
## Prediction  edible poisonous
##   edible       835         0
##   poisonous      7       784
```