Continuous outcomes

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Sandro Raabe
Data Scientist

The dataset

```r
head(chocolate, 5)
```

```
final_grade review_date   cocoa_percent  company_location  bean_type                broad_bean_origin
<dbl>       <int>         <dbl>          <fct>             <fct>                    <fct>
3           2009          0.8            U.K.              "Criollo, Trinitario"    "Madagascar"
3.75        2012          0.7            Guatemala         "Trinitario"             "Madagascar"
2.75        2009          0.75           Colombia          "Forastero (Nacional)"   "Colombia"
3.5         2014          0.74           Zealand           ""                       "Papua New Guinea"
3.75        2011          0.72           Australia         ""                       "Bolivia"
```

Construct the regression tree

```r
spec <- decision_tree() %>%
  set_mode("regression") %>%
  set_engine("rpart")

print(spec)
```

```
Decision Tree Model Specification
(regression)

Computational engine: rpart
```

```r
model <- spec %>%
  fit(formula = final_grade ~ .,
      data = chocolate_train)

print(model)
```

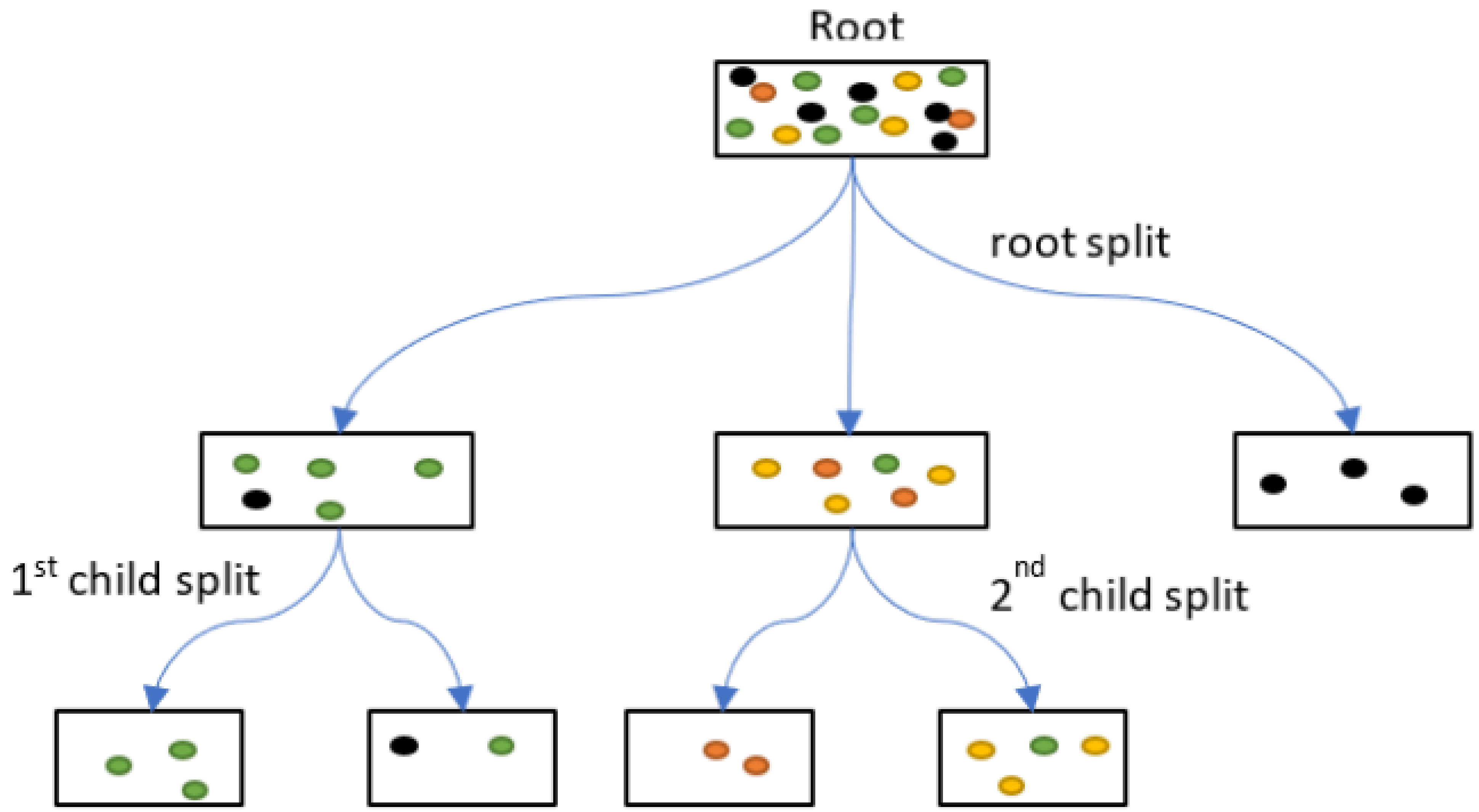
```
parsnip model object

Fit time:  20ms
n= 1437

node), split, n, deviance, yval
      * denotes terminal node
```

Predictions using a regression tree

```r
# Model predictions on new data
predict(model, new_data = chocolate_test)
```

```
.pred
<dbl>
3.281915
3.435234
3.281915
3.833931
3.281915
3.514151
3.273864
3.514151
```

Divide & conquer



Hyperparameters

Goal for regression trees:

- Low variance or deviation from the mean within groups

Design decisions:

- `min_n` : number of data points in a node needed for further split (default: 20)
- `tree_depth` : maximum depth of a tree (default: 30)
- `cost_complexity` : penalty for complexity (default: 0.01)

Set them in very first step:

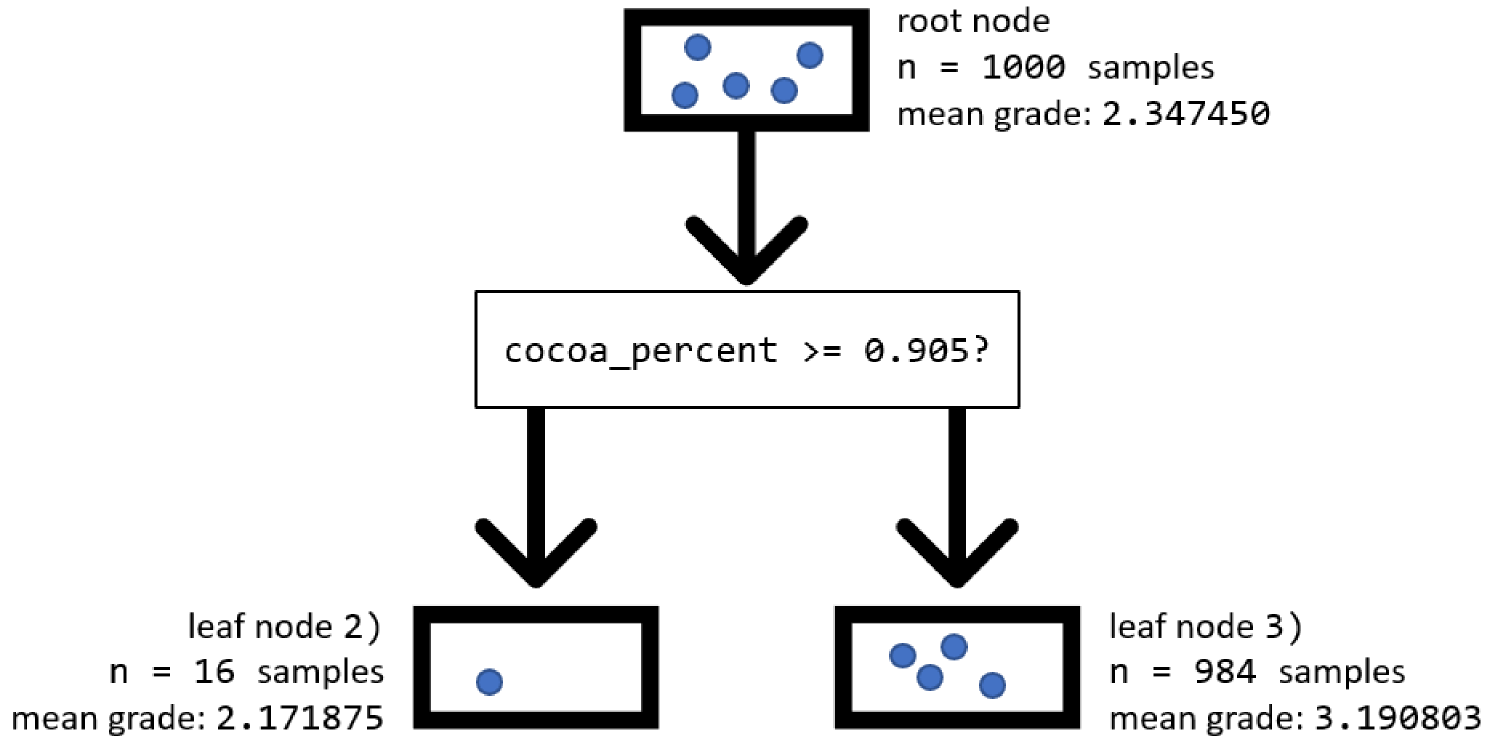
```
decision_tree(tree_depth = 4, cost_complexity = 0.05) %>%
    set_mode("regression")
```

Understanding model output

```
decision_tree(tree_depth = 1) %>%
  set_mode("regression") %>%
  set_engine("rpart")  %>%
  fit(formula = final_grade ~ .,
      data = chocolate_train)
```

```
parsnip model object

Fit time:  1ms
n= 1000

node), split, n, yval

1) root                    1000  2.347450
2) cocoa_percent>=0.905      16  2.171875 *
3) cocoa_percent<0.905      984  3.190803 *
```

- Model with `tree_depth = 1`
- Visualization:



Let's do regression!

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Performance metrics for regression trees

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Sandro Raabe
Data Scientist

How to measure performance?

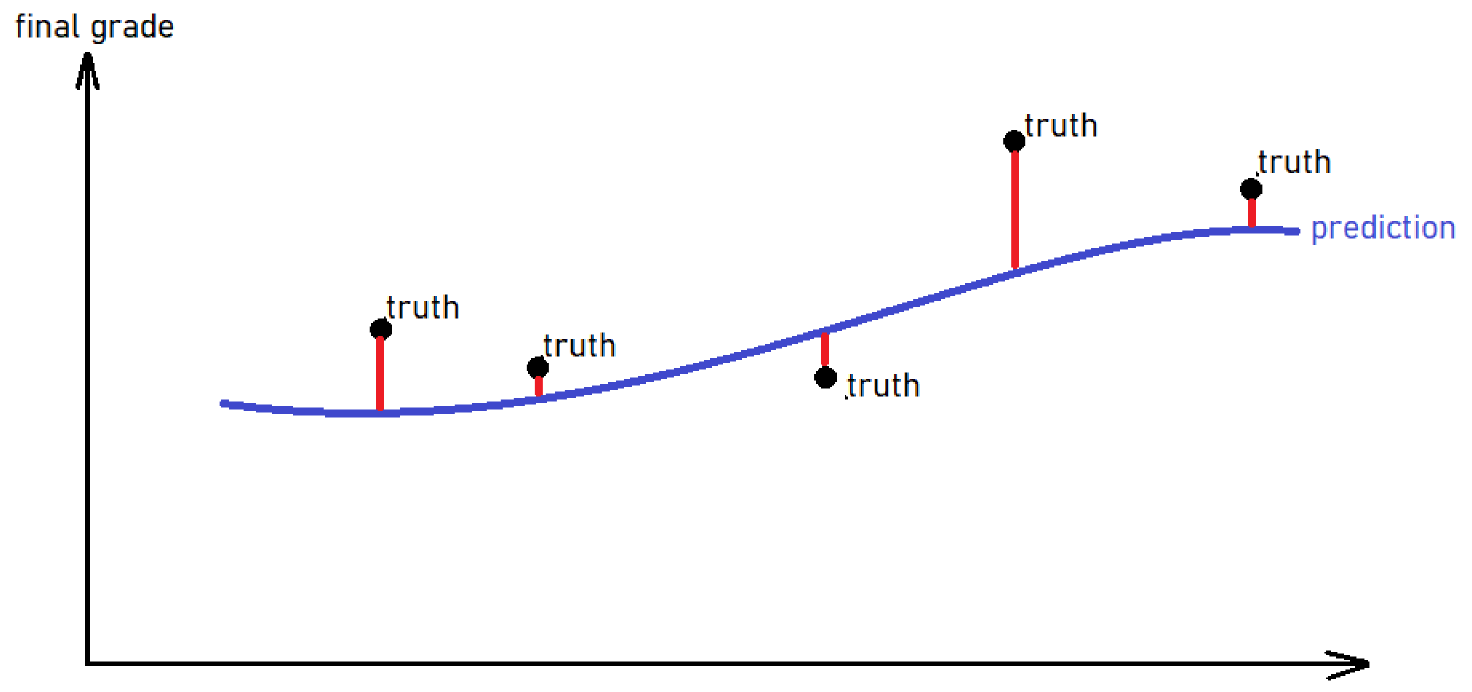
- Classification problems: accuracy (confusion matrix)
- Regression problems: "correct" is relative, no binary correctness

$\Rightarrow$ Measure how far predictions are away from truth

Common metrics for regression

- **M**ean **A**bsolute **E**rror (**MAE**)
- **R**oot **M**ean **S**quare **E**rror (**RMSE**)

MAE intuition:



MAE = average length of the red bars

Formulas and intuition

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |actual_i - predicted_i|$$

- "Sum of absolute deviations divided by the number of predictions"

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (actual_i - predicted_i)^2$$

- "Mean squared error"

Formulas and intuition

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|actual_i - predicted_i|$$

- "Sum of absolute deviations divided by the number of predictions"

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(actual - predicted)^2}$$

- "Root of the mean squared error"
- Large errors get higher weight

Coding: predictions

```r
# parsnip and yardstick are included in tidymodels
library(tidymodels)
```

```r
# Make predictions and add to test data
predictions <- predict(model, new_data = chocolate_test) %>%
  bind_cols(chocolate_test)
```

```
# A tibble: 358 x 7
   .pred final_grade review_date cocoa_percent company_location
   <dbl>       <dbl>       <int>         <dbl> <fct>
 1  2.5         2.75        2013          0.7  France
 2  3.64        3.25        2014          0.8  France
 3  3.3         3.5         2012          0.7  France
 4  3.25        3.5         2011          0.72 Fiji
# ... with 354 more rows, and 2 more variables: bean_type <fct>, broad_bean_origin <fct>
```

Coding: mae() and rmse()

```r
# Evaluate using mae()
mae(predictions,
    estimate = .pred,
    truth = final_grade)
```

```
# A tibble: 1 x 2
  .metric   .estimate
  <chr>         <dbl>
1 mae           0.363
```

```r
# Evaluate using rmse()
rmse(predictions,
     estimate = .pred,
     truth = final_grade)
```

```
# A tibble: 1 x 2
  .metric   .estimate
  <chr>         <dbl>
1 rmse          0.457
```

Let's evaluate!

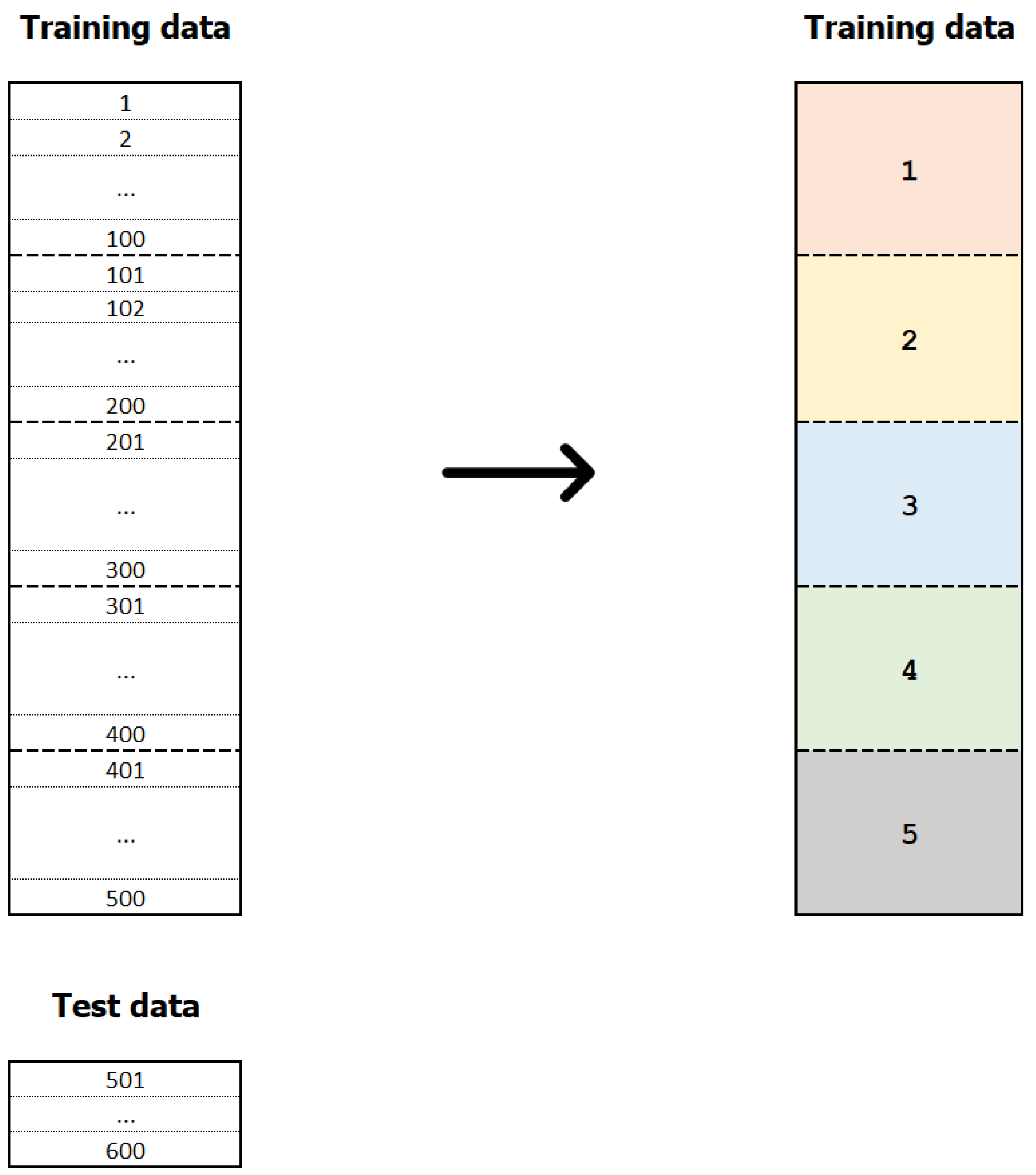
MACHINE LEARNING WITH TREE-BASED MODELS IN R

Cross-validation

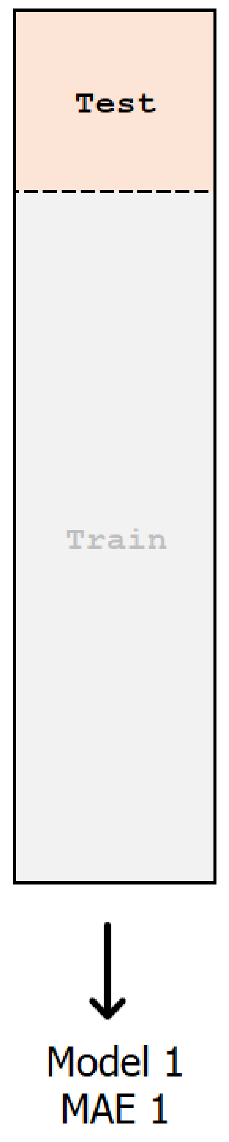
MACHINE LEARNING WITH TREE-BASED MODELS IN R

Sandro Raabe
Data Scientist

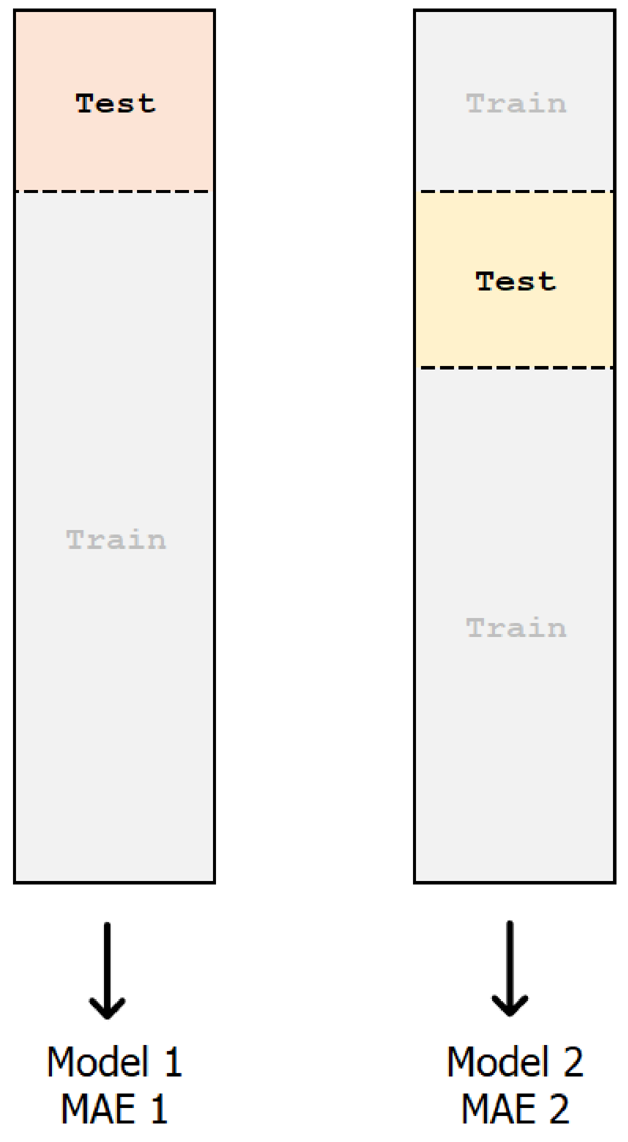
k-fold cross-validation



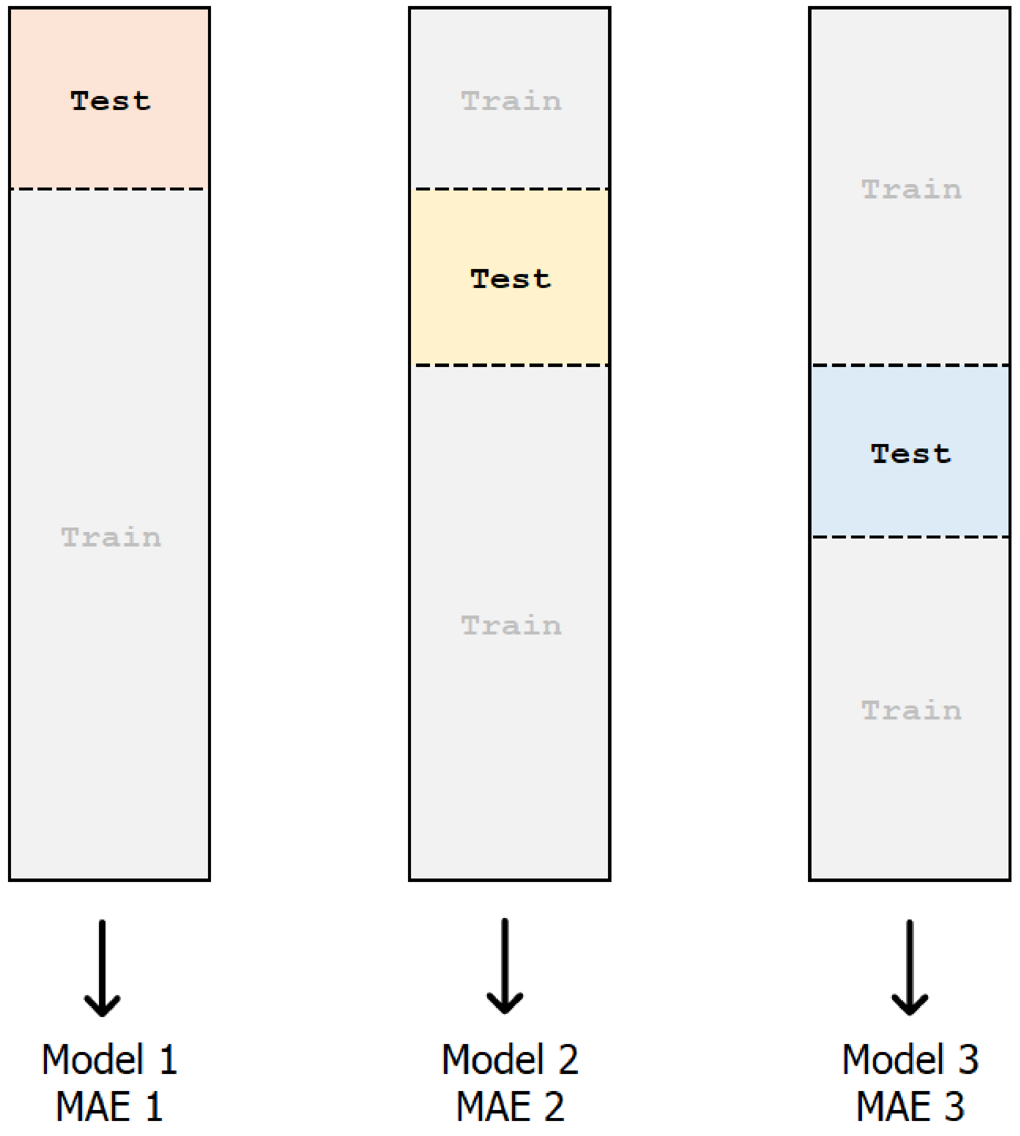
k-fold cross-validation



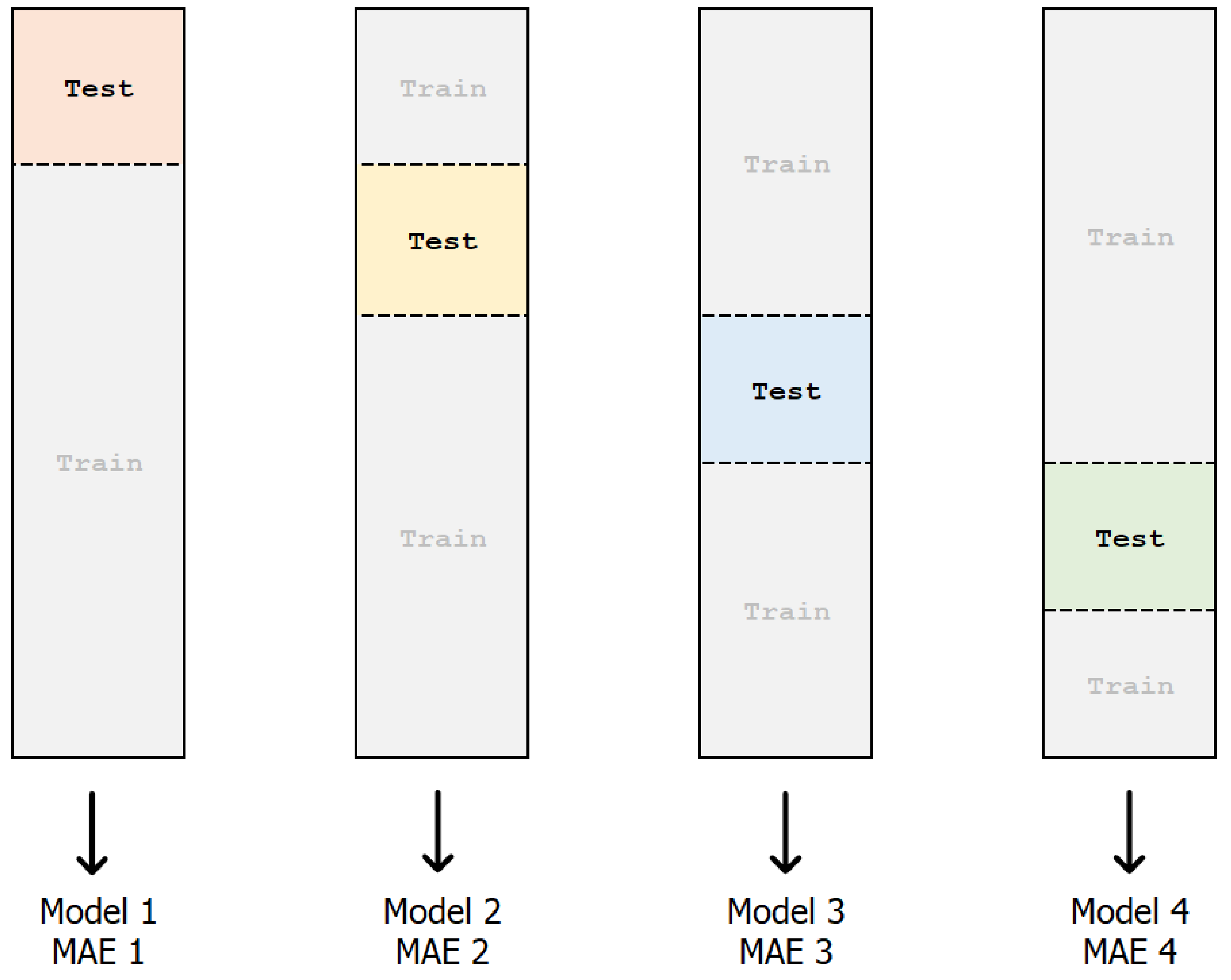
k-fold cross-validation



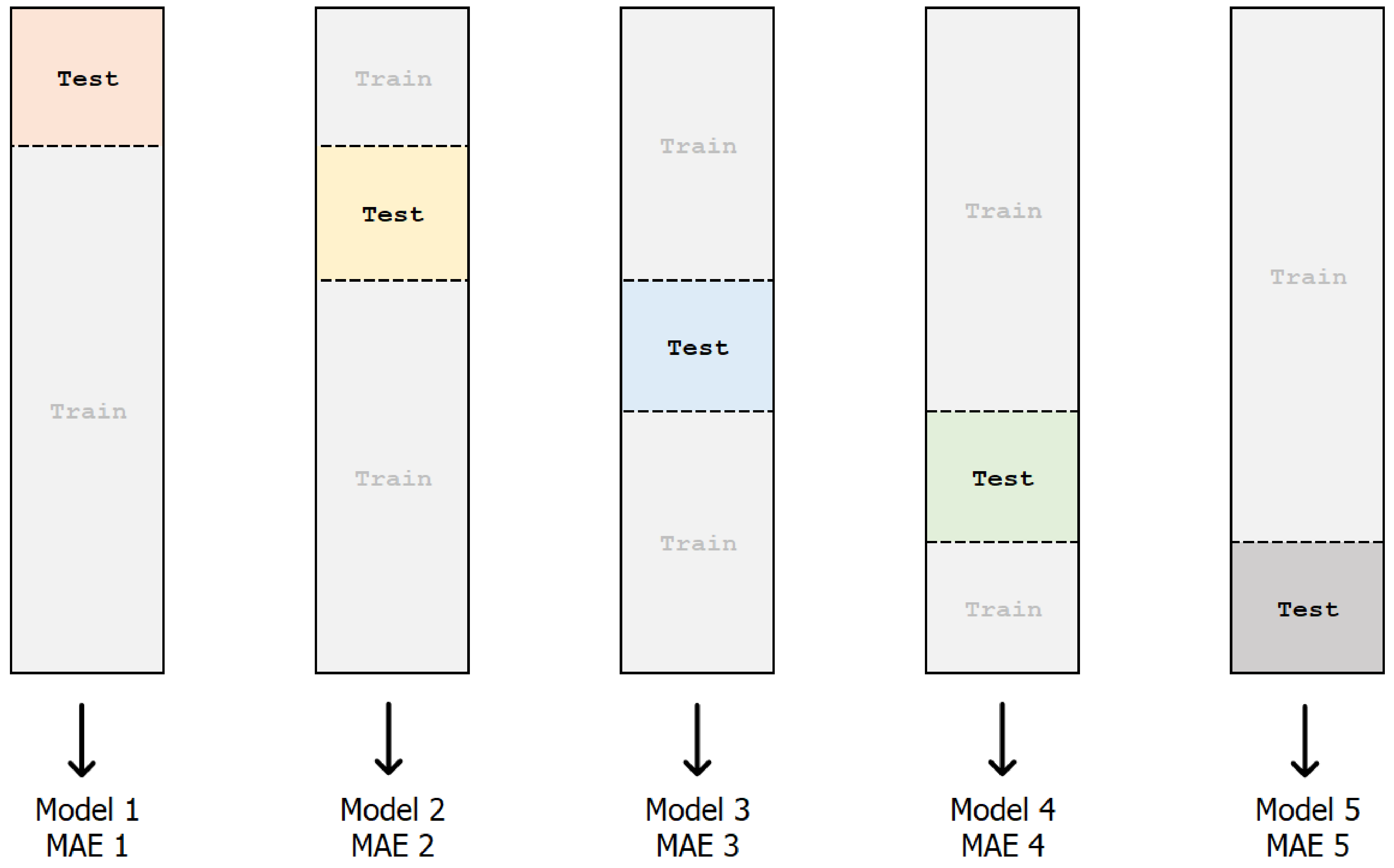
k-fold cross-validation



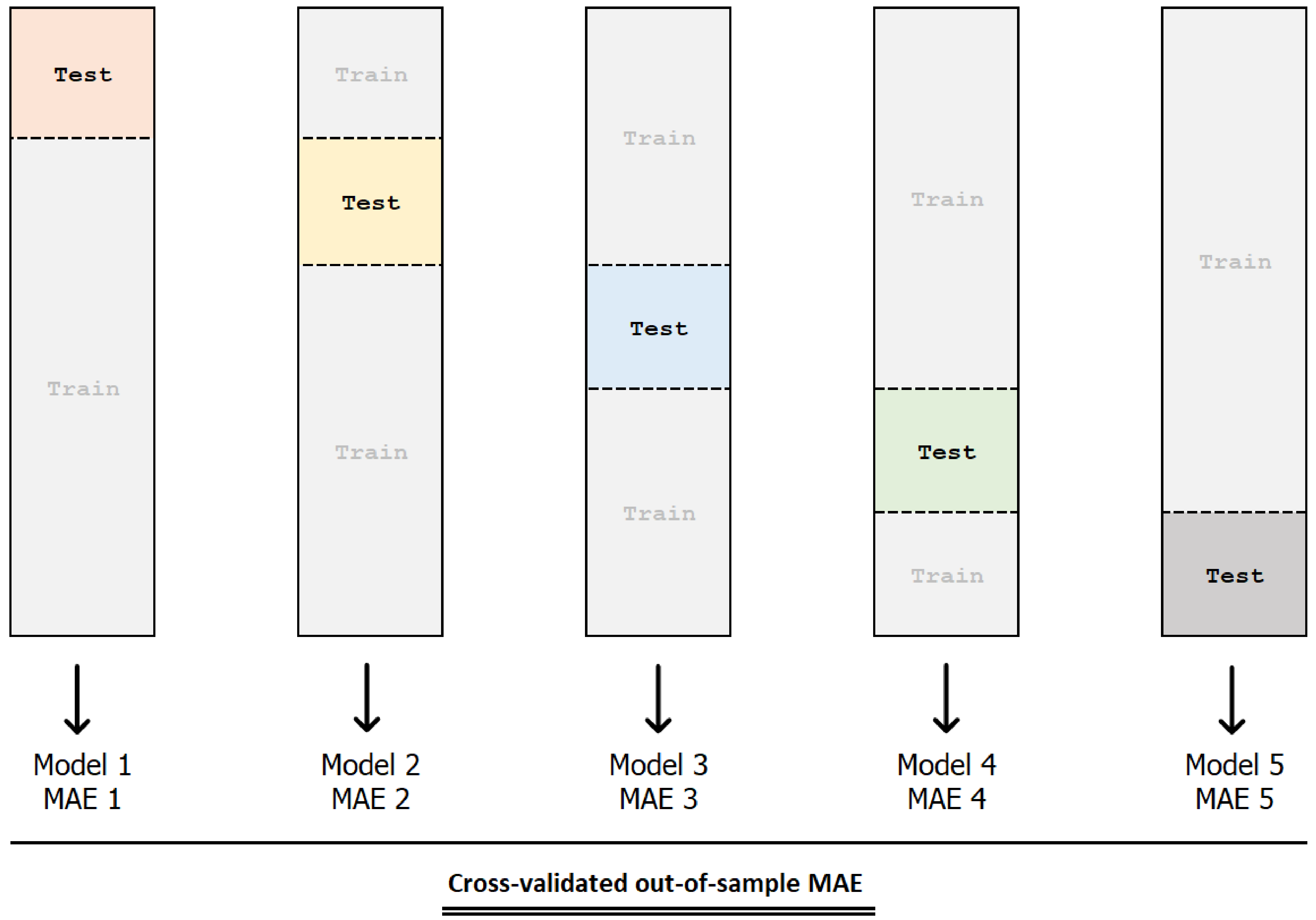
k-fold cross-validation



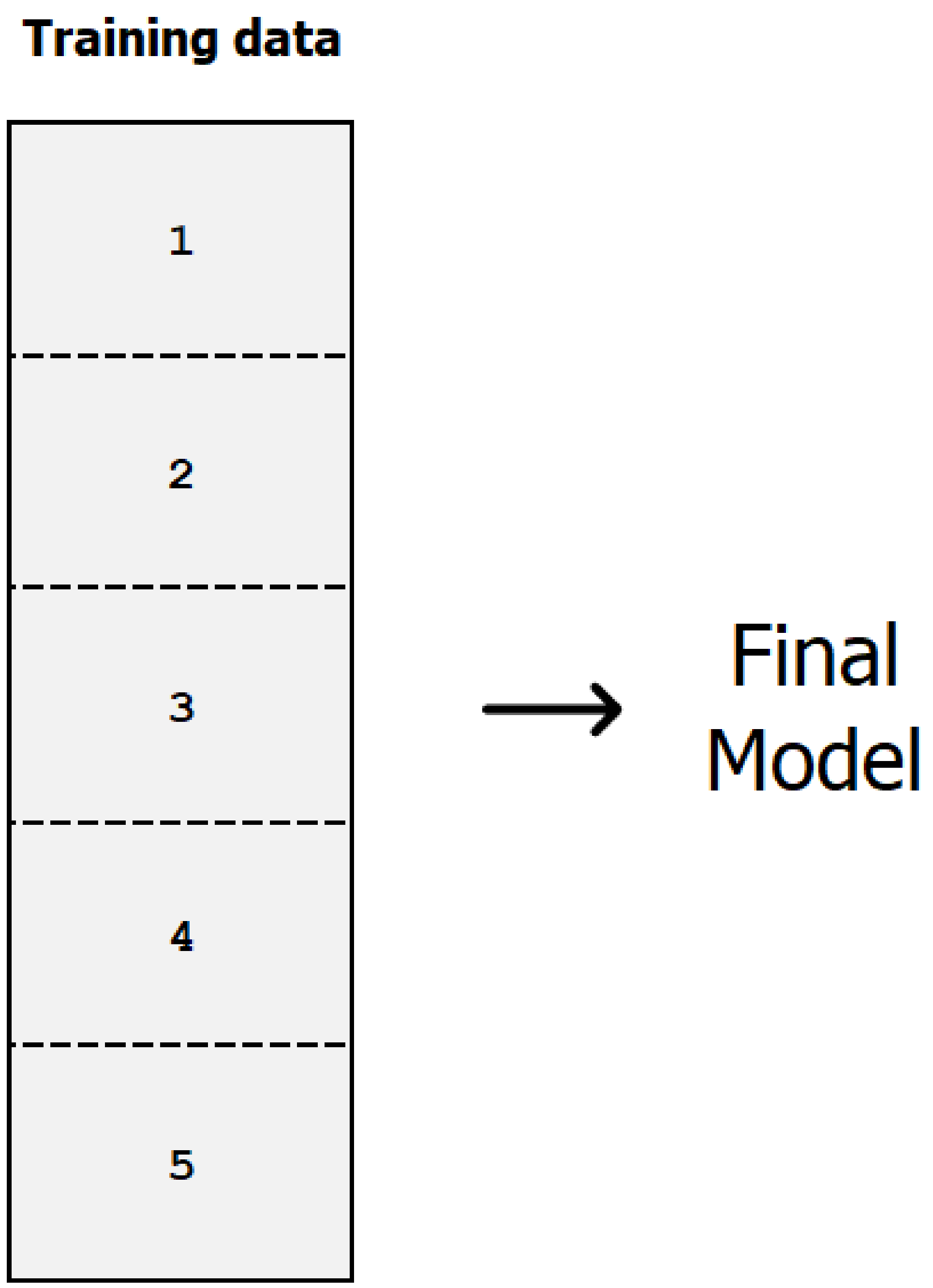
k-fold cross-validation



k-fold cross-validation



Fit final model on the full dataset



Coding - Split the data 10 times

```r
# Random seed for reproducibility
set.seed(100)

# Create 10 folds of the dataset
chocolate_folds <- vfold_cv(chocolate_train, v = 10)
```

```
#  10-fold cross-validation
# A tibble: 10 x 2
  splits             id
1 <split [1293/144]> Fold1
2 <split [1293/144]> Fold2
3 <split [1293/144]> Fold3
4 ...
```

Coding - Fit the folds

```r
# Fit a model for every fold and calculate MAE and RMSE
fits_cv <- fit_resamples(tree_spec,
                         final_grade ~ .,
                         resamples = chocolate_folds,
                         metrics = metric_set(mae, rmse))
```

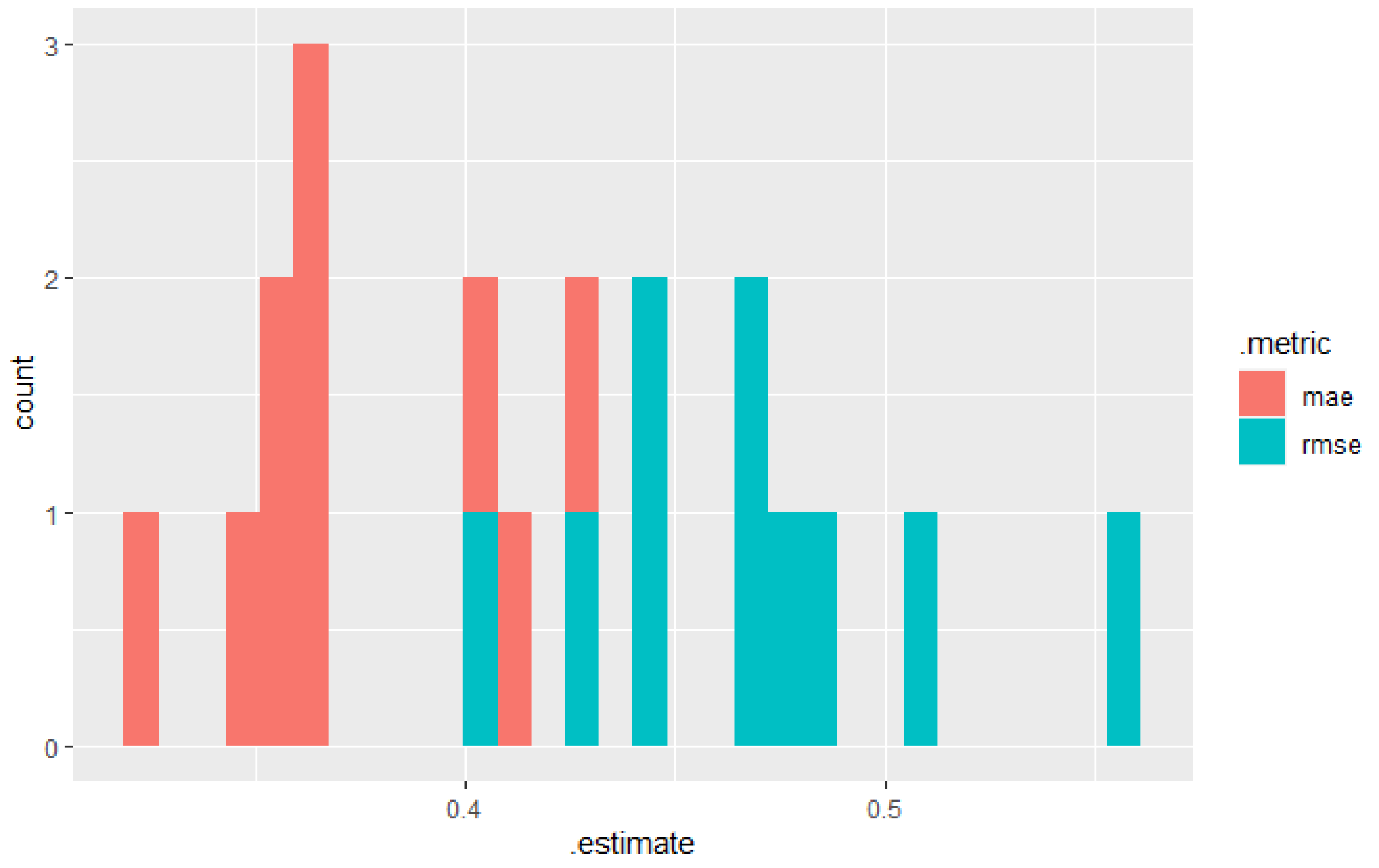
```
# Resampling results
# 10-fold cross-validation
# A tibble: 10 x 4
  splits             id    .metrics
  <list>             <chr> <list>
1 <split [1293/144]> Fold1 <tibble [2 x 4]>
2 <split [1293/144]> Fold2 <tibble [2 x 4]>
3 <split [1293/144]> Fold3 <tibble [2 x 4]>
4 ...
```

Coding - Collect all errors

```r
# Collect raw errors of all model runs
all_errors <- collect_metrics(fits_cv,
                              summarize = FALSE)
print(all_errors)
```

```
# A tibble: 20 x 3
   id       .metric  .estimate
   <chr>    <chr>        <dbl>
 1 Fold01   mae          0.362
 2 Fold01   rmse         0.442
 3 Fold02   mae          0.385
 4 Fold02   rmse         0.504
 5 ...
```

```r
library(ggplot2)
ggplot(all_errors, aes(x = .estimate,
                       fill = .metric)) +
  geom_histogram()
```



Coding - Summarize training sessions

```r
# Collect and summarize errors of all model runs
collect_metrics(fits_cv)
```

```
# A tibble: 2 x 3
  .metric   mean     n
  <chr>    <dbl> <int>
1 mae      0.383    10
2 rmse     0.477    10
```

Let's cross-validate!

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Bias-variance tradeoff

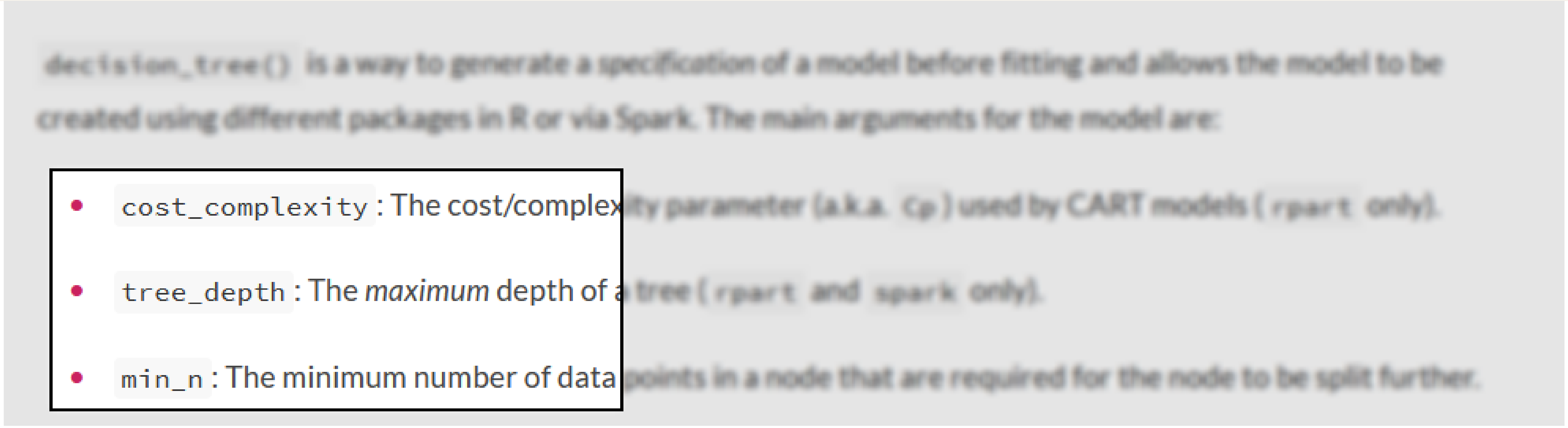
MACHINE LEARNING WITH TREE-BASED MODELS IN R

Sandro Raabe
Data Scientist

Hyperparameters

- Chosen by modeler
- e.g. `tree_depth`
- Check documentation!

```
?decision_tree
```

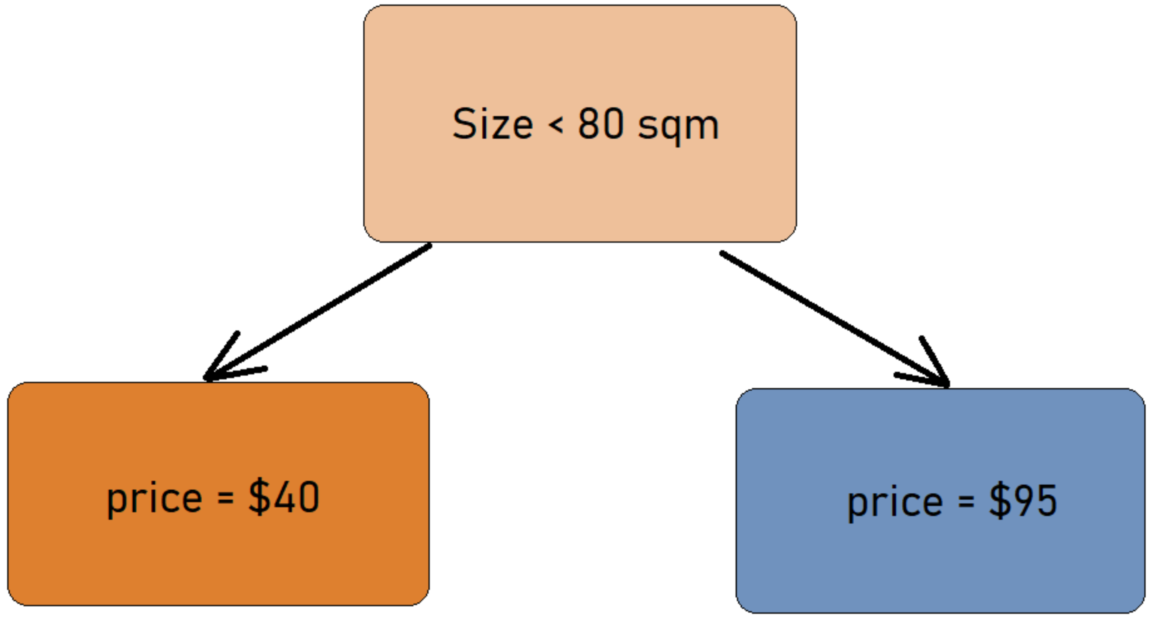


`decision_tree()` is a way to generate a specification of a model before fitting and allows the model to be created using different packages in R or via Spark. The main arguments for the model are:

- `cost_complexity` : The cost/complexity parameter (a.k.a. `Cp`) used by CART models (`rpart` only).
- `tree_depth` : The *maximum* depth of a tree (`rpart` and `spark` only).
- `min_n` : The minimum number of data points in a node that are required for the node to be split further.

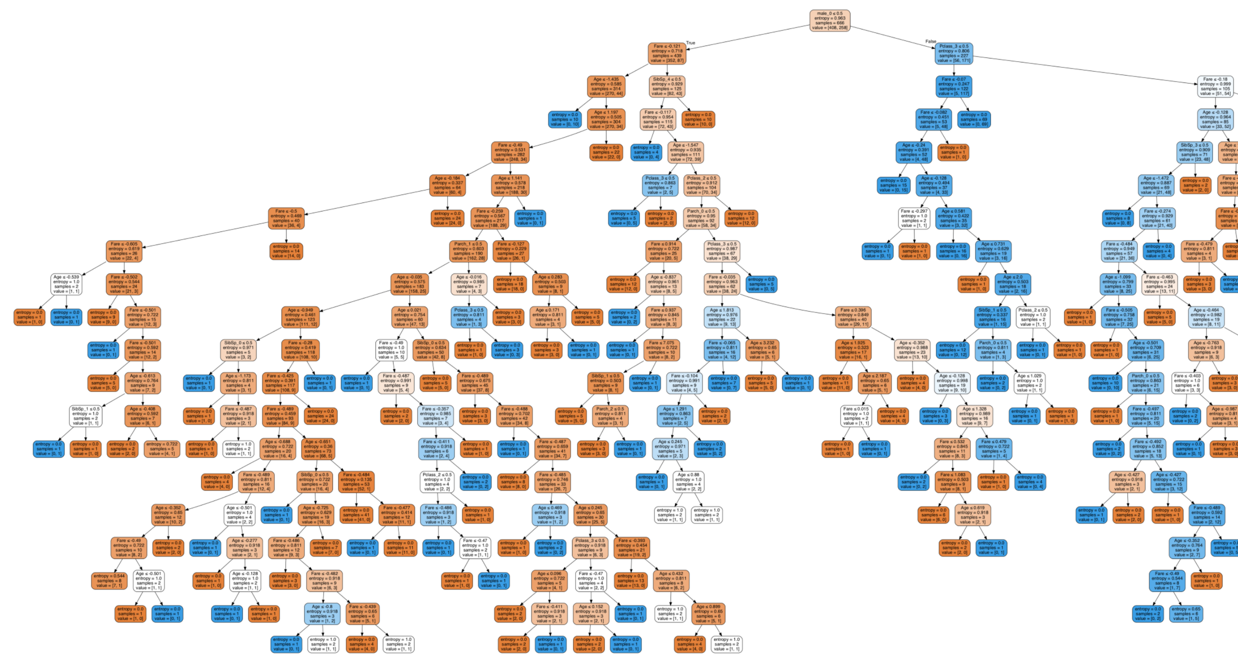
Simple model

```
simple_spec <- decision_tree(tree_depth = 2) %>%
    set_mode("regression")

simple_spec %>% fit(final_grade ~ .,
                    data = training_data)
```



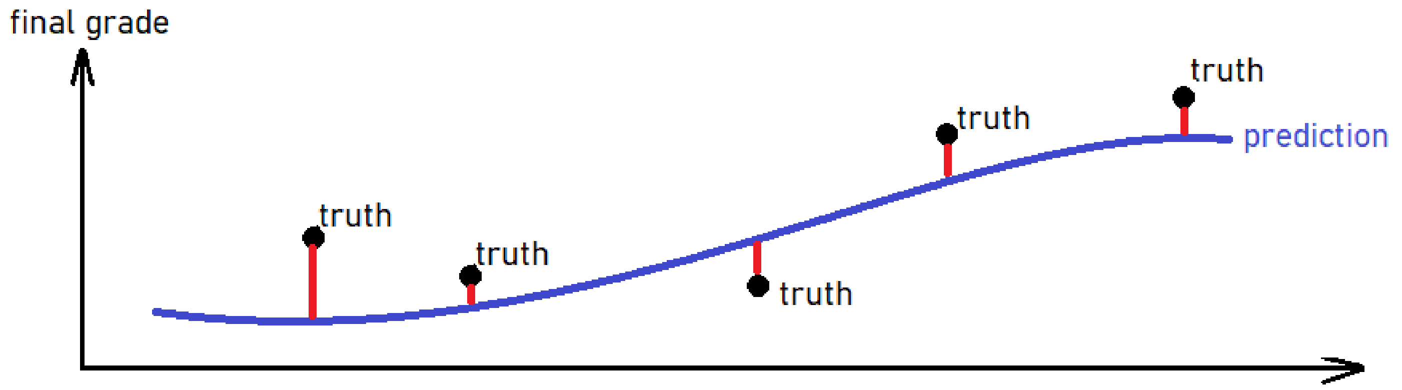
Complex model

```
complex_spec <- decision_tree(tree_depth = 15) %>
    set_mode("regression")

complex_spec %>% fit(final_grade ~ .,
                     data = training_data)
```



Complex model - overfitting - high variance

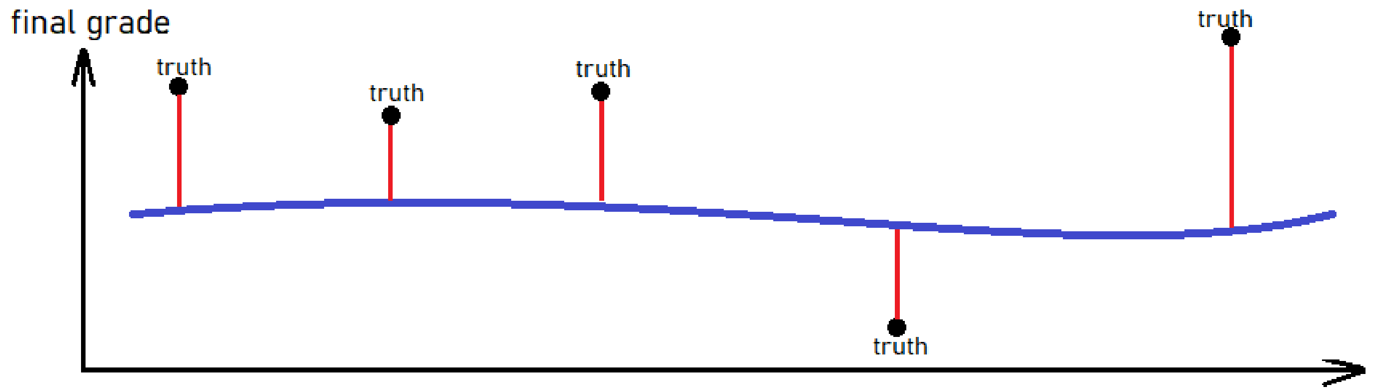
Predictions on training set: well done!



```
mae(train_results,
    estimate = .pred,
    truth = final_grade)
```

```
# A tibble: 1 x 3
  .metric  .estimate
1 mae          0.204
```

Predictions on test set: not even close!



```
mae(test_results,
    estimate = .pred,
    truth = final_grade)
```

```
# A tibble: 1 x 3
  .metric  .estimate
1 mae          0.947
```

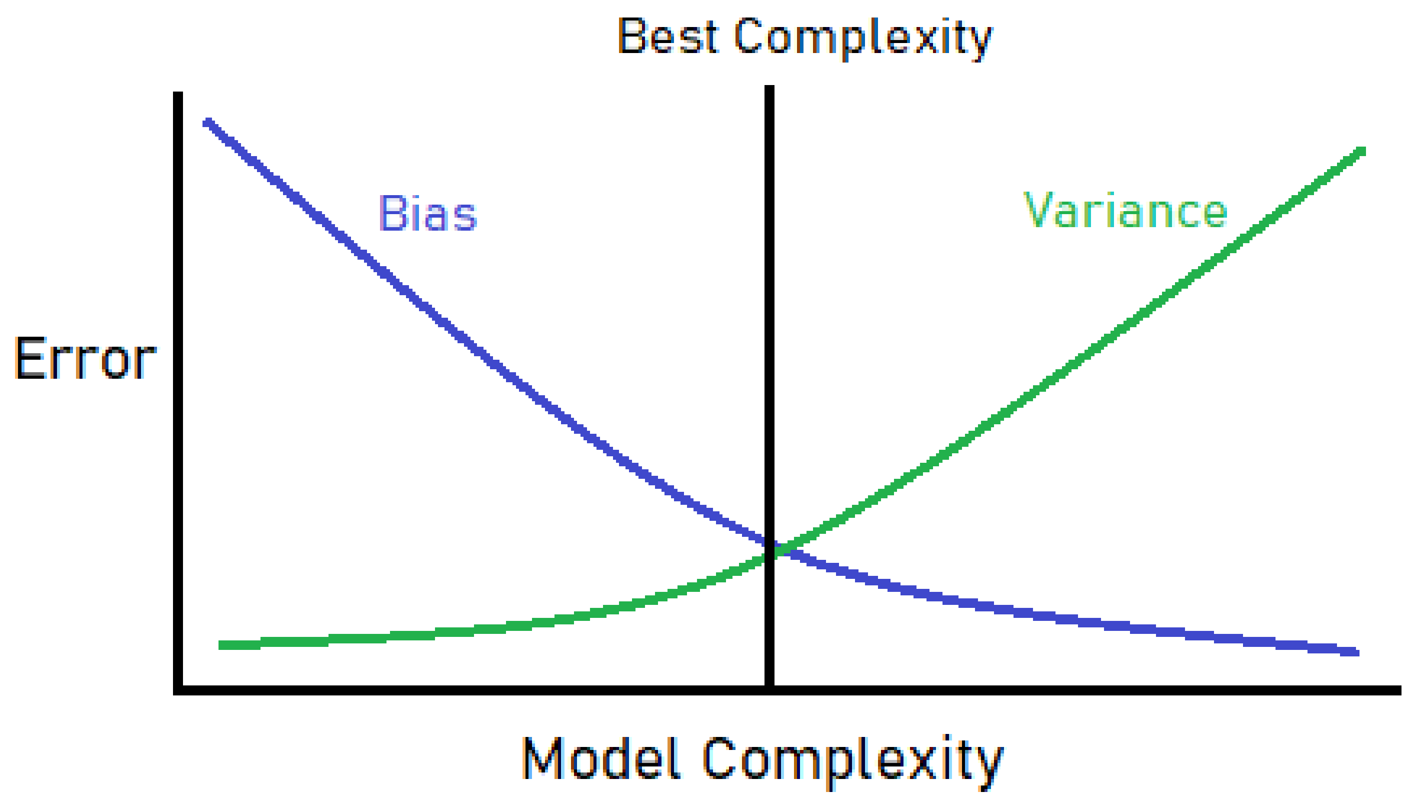
Simple model - underfitting - high bias

Large errors on training **and** test set:

```
bind_rows(training = mae(train_results, estimate = .pred, truth = final_grade),
          test     = mae(test_results,  estimate = .pred, truth = final_grade),
          .id = "dataset")
```

```
# A tibble: 2 x 4
  dataset    .metric  .estimate
  <chr>      <chr>        <dbl>
1 training   mae          0.754
2 test       mae          0.844
```

The bias-variance tradeoff



- Simple models -> high bias
- Complex models -> high variance
- Tradeoff between bias and variance
- Build models around the *sweet spot*

Detecting overfitting

Out-of-sample/CV:

```
collect_metrics(cv_fits)
```

```
# A tibble: 1 x 3
  .metric    mean     n
1 mae       2.432     5
```

- High CV error
- **Overfit / high variance**
- Reduce complexity!

In-sample:

```
mae(training_pred,
    estimate = .pred,
    truth = final_grade)
```

```
# A tibble: 1 x 2
  .metric  .estimate
1 mae          0.228
```

- Small training error

Detecting underfitting

In-sample:

```
mae(training_pred, estimate = .pred, truth = final_grade)
```

```
# A tibble: 1 x 2
  .metric .estimate
  <chr>       <dbl>
1 mae         2.432
```

- Large in-sample/training error
- **Underfit / high bias**
- Increase complexity!

Let's trade off!

MACHINE LEARNING WITH TREE-BASED MODELS IN R