Introduction to regression trees

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Erin LeDell
Instructor

Train a Regression Tree in R

```
rpart(formula = ___,
      data = ___,
      method = ___)
```

formula	is in the format: outcome ~ predictor1+predictor2+etc
data=	specifies the dataframe
method	"class" for classification tree "anova" for regression tree
control=	*optional* parameters for controlling the tree growth

Train/Validation/Test Split

- training set
- validation set
- test set

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Performance metrics for regression

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Erin LeDell
Instructor

Common metrics for regression

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum | \, actual - predicted \, |$$

- Root Mean Square Error (RMSE)

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

Evaluate a regression tree model

```
pred <- predict(object = model,  # model object
                newdata = test)  # test dataset
```

```
library(Metrics)


# Compute the RMSE
rmse(actual = test$response, # the actual values
     predicted = pred)       # the predicted values
```

```
2.278249
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN R

What are the hyperparameters for a decision tree?

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Erin LeDell
Instructor

Decision tree hyperparameters

```
?rpart.control
```

rpart.control {rpart}

Control for Rpart Fits

Description

Various parameters that control aspects of the `rpart` fit.

Usage

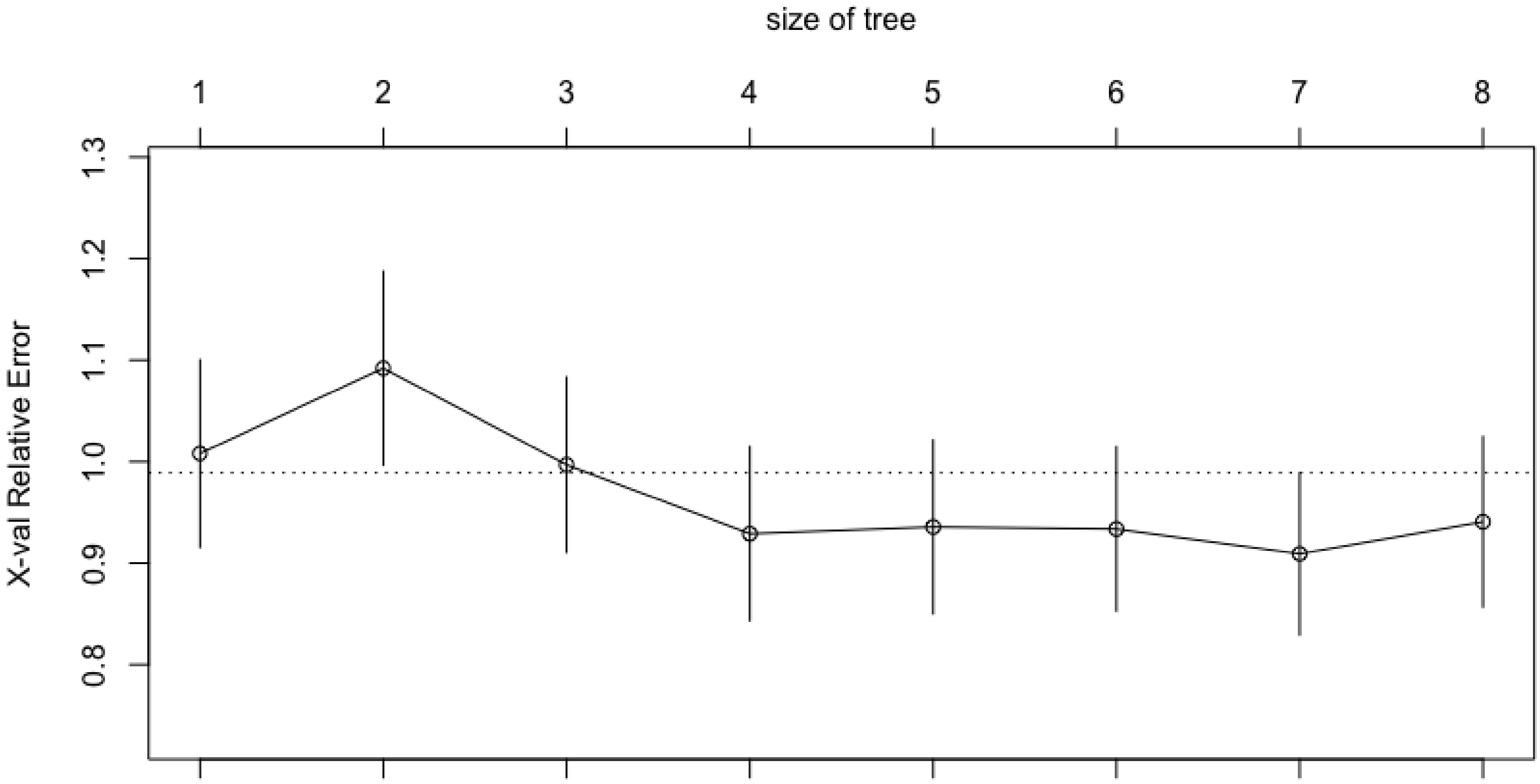
```
rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01,
              maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10,
              surrogatestyle = 0, maxdepth = 30, ...)
```

Decision tree hyperparameters

- **minsplit**: minimum number of data points required to attempt a split
- **cp**: complexity parameter
- **maxdepth**: depth of a decision tree

Cost-Complexity Parameter (CP)

```
plotcp(grade_model)
```



Cost-Complexity Parameter (CP)

```
print(model$cptable)
```

```
          CP nsplit rel error    xerror       xstd
1 0.06839852      0 1.0000000 1.0080595 0.09215642
2 0.06726713      1 0.9316015 1.0920667 0.09543723
3 0.03462630      2 0.8643344 0.9969520 0.08632297
4 0.02508343      3 0.8297080 0.9291298 0.08571411
5 0.01995676      4 0.8046246 0.9357838 0.08560120
6 0.01817661      5 0.7846679 0.9337462 0.08087153
7 0.01203879      6 0.7664912 0.9092646 0.07982862
8 0.01000000      7 0.7544525 0.9407895 0.08399125
```

Cost-Complexity Parameter (CP)

```
# Prune the model to optimized cp value
model_opt <- prune(tree = model, cp = cp_opt)
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Grid Search for model selection

MACHINE LEARNING WITH TREE-BASED MODELS IN R

Erin LeDell
Instructor

Grid Search

- What is a model hyperparameter?
- What is a "grid"?
- What is the goal of a grid search?
- How is the best model chosen?

Set up the grid

```
# Establish a list of possible
# values for minsplit & maxdepth

splits <- seq(1, 30, 5)
depths <- seq(5, 40, 10)
```

```
# Create a data frame containing
# all combinations

hyper_grid <- expand.grid(
               minsplit = splits
               maxdepth = depths
```

```
hyper_grid[1:10,]
```

```
   minsplit maxdepth
1         1        5
2         6        5
3        11        5
4        16        5
5        21        5
6        26        5
7         1       15
8         6       15
9        11       15
10       16       15
```

Grid Search in R: Train models

```
# Create an empty list to store models
models <- list()
```

```
# Execute the grid search
for (i in 1:nrow(hyper_grid)) {
    # Get minsplit, maxdepth values at row i
    minsplit <- hyper_grid$minsplit[i]
    maxdepth <- hyper_grid$maxdepth[i]

    # Train a model and store in the list
    models[[i]] <- rpart(formula = response ~ .,
                         data = train,
                         method = "anova",
                         minsplit = minsplit,
                         maxdepth = maxdepth)
```

```r
# Create an empty vector to store RMSE values
rmse_values <- c()
```

```r
# Compute validation RMSE
for (i in 1:length(models)) {

    # Retreive the i^th model from the list
    model <- models[[i]]

    # Generate predictions on grade_valid
    pred <- predict(object = model,
                    newdata = valid)

    # Compute validation RMSE and add to the
    rmse_values[i] <- rmse(actual = valid$response,
                           predicted = pred)
}
```

Let's practice!

MACHINE LEARNING WITH TREE-BASED MODELS IN R