Regression and Classification Trees

Günter J. Hitsch

February 21, 2017

Regression trees

As an example, we use household-level private label share data aggregated at the *year* level.

```
load("./Data/PL_shares_annual.RData")
```

Convert all categorical variables to factors. The `rpart` package will then understand that these variables are categorical.

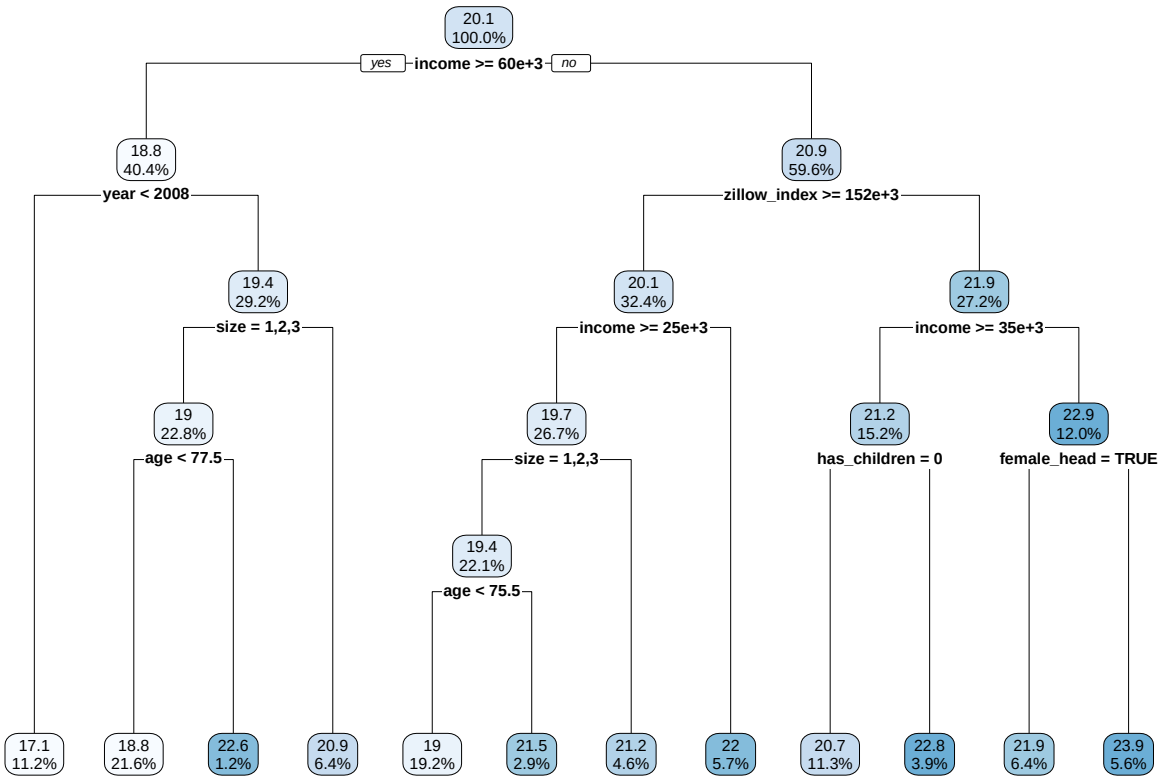
```
categorical_vars = c("unemployed", "has_children", "female_head", "size")
PL_shares_DT[, (categorical_vars) := lapply(.SD, as.factor), .SDcols = categorical_vars]
```

Now build a regression tree (`method = "anova"`).

```
tree = rpart(PL_share ~ ., data = PL_shares_DT, method = "anova",
             control = rpart.control(minsplit = 20, cp = 0.001, maxdepth = 5))
```

The cost-complexity parameter `cp` is key: If the algorithm does not split, make the parameter smaller. `minsplit` determines the minimum number of observations in a node such that a split is attempted, `maxdepth` determines the depth (length) of a tree.

```
N_terminal_nodes = length(unique(tree$where))
rpart.plot(tree, tweak = 0.8, digits = 3)
```



Cost-complexity pruning

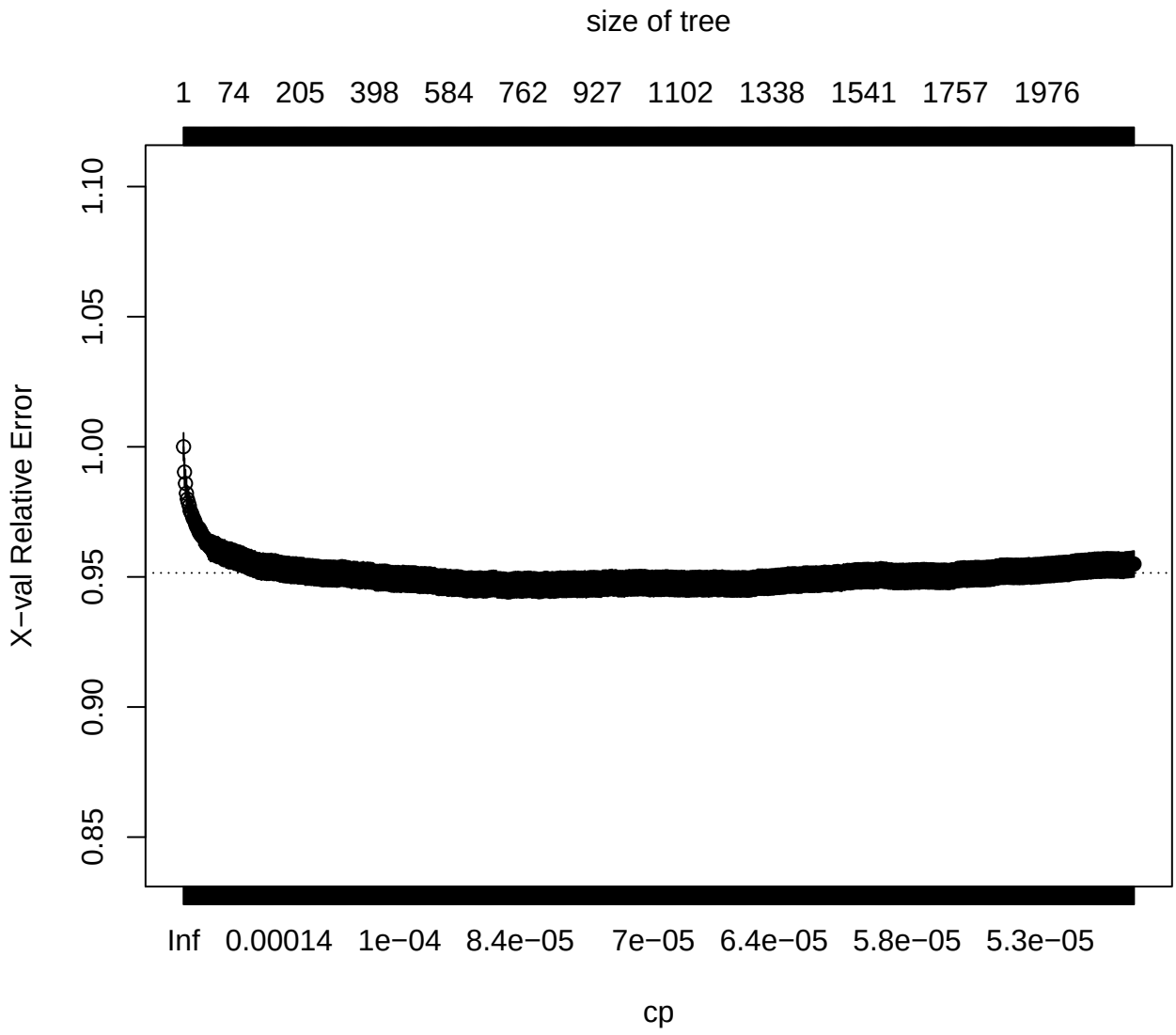
Grow a large tree:

```
tree = rpart(PL_share ~ ., data = PL_shares_DT, method = "anova",
             control = rpart.control(minsplit = 20, cp = 0.00005))

N_terminal_nodes = length(unique(tree$where))
# rpart.plot(tree, tweak = 0.8, digits = 3)
```

Visualize the cross-validation results:

```
plotcp(tree)
```



Find the optimal complexity parameter:

```
index_optimal = which.min(tree$cptable[, "xerror"])
cp_optimal    = tree$cptable[index_optimal, "CP"]
size_optimal  = tree$cptable[index_optimal, "nsplit"] + 1
```

Prune the tree:

```
pruned_tree = prune(tree, cp = cp_optimal)
# rpart.plot(pruned_tree)
```

Classification trees

Use the RFM data to predict if a customer buys.

```
load("./Data/RFM-Data.RData")
```

Use the `method = "class"` option for classification.

```
tree = rpart(buyer ~ ., data = rfm_DT, method = "class",
             control = rpart.control(minsplit = 20, cp = 0.00005))

rpart.plot(tree, tweak = 0.8, digits = 3)
```