Decision Trees

Data Sets

Attrition

```
attrition <- attrition %>% mutate_if(is.ordered, factor, order = F)
attrition.h2o <- as.h2o(attrition)

churn <- initial_split(attrition, prop = .7, strata = "Attrition")
churn.train <- training(churn)
churn.test <- testing(churn)
```

Ames, Iowa housing data.

```
set.seed(123)

ames <- AmesHousing::make_ames()
ames.h2o <- as.h2o(ames)

ames.split <- initial_split(ames, prop =.7, strata = "Sale_Price")

ames.train <- training(ames.split)
ames.test <- testing(ames.split)
```

Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The most common type of decision tree is a *c*lassification *a*nd *r*egresion *t*ree (CART).

Partitioning

CART uses *binary recursive partitioning*, where the objective at each node is to find the "best" feature (x_i) to partition the remaining data into one of two regions (R_1, R_2) such that the overall error between the actual response (y_i) and the predicted constant (c_i) is minimized.

For regression, the objective is to minimize the total SSE:

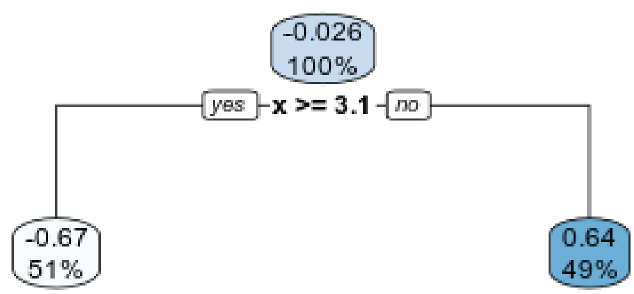
$SSE = \sum_{i \in R_1} (y_i - c_1)^2 + \sum_{i \in R_2} (y_i - c_2)^2$

For classification problems, the partitioning is usually made to maximize the reduction in cross-enthropy or the Gini index.

For example, say we have data generated from a simple sin function with Gaussian noise: $Y_i \overset{i.i.d}{\sim} N(sin(X_i, \sigma^2)$

Data & model:

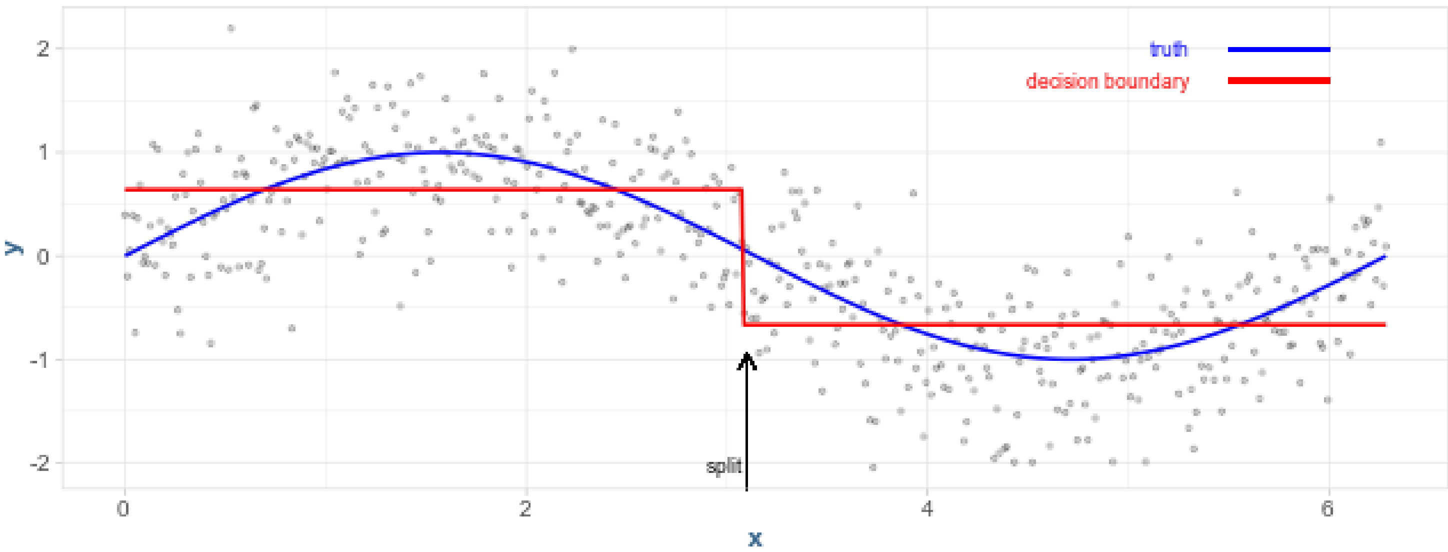
```r
set.seed(1112)

df <- tibble::tibble(
   x = seq(from = 0, to = 2 * pi, length = 500),
   y = sin(x) + rnorm(length(x), sd = .5),
   truth = sin(x)
)

# run decision stump model
ctrl <- list(cp = 0, minbucket = 5, maxdepth = 1)
fit <- rpart(y ~ x, data = df, control = ctrl)

# plot tree
rpart.plot(fit)
```



Decision Boundry:

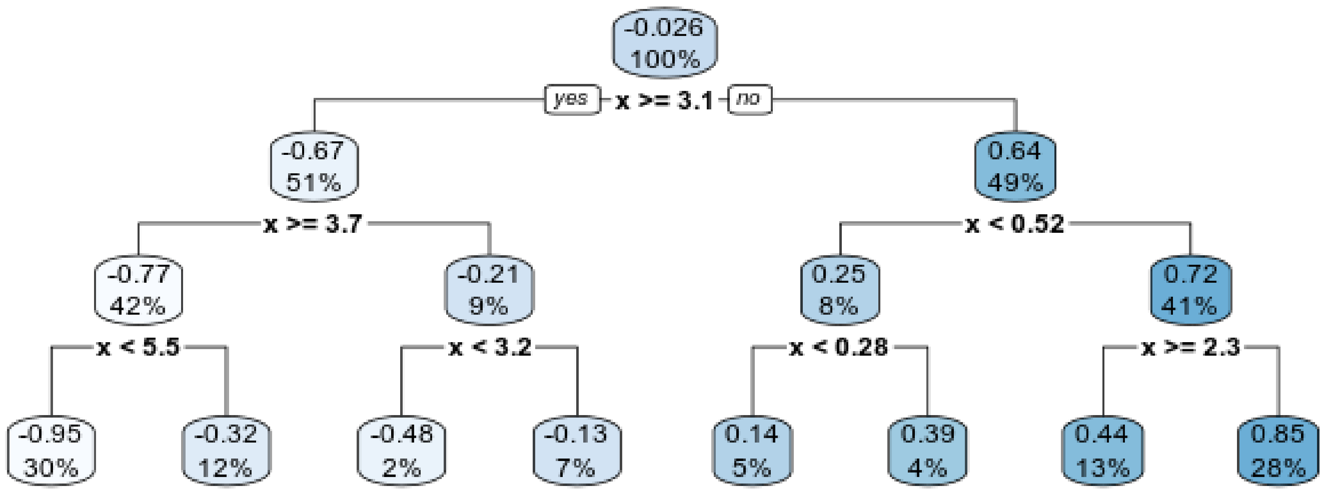
```r
df %>%
   mutate(pred = predict(fit, df)) %>%
   ggplot(aes(x, y)) +
      geom_point(alpha = .2, size = 1) +
      geom_line(aes(x, y = truth), color = "blue", size = .75) +
      geom_line(aes(y = pred), color = "red", size = .75) +
      geom_segment(x = 3.1, xend = 3.1, y = -Inf, yend = -.95,
                   arrow = arrow(length = unit(0.25, "cm")), size = .25) +
      annotate("text", x = 3.1, y = -Inf, label = "split", hjust = 1.2, vjust = -1, size = 3) +
      geom_segment(x = 5.5, xend = 6, y = 2, yend = 2, size = .75, color = "blue") +
      geom_segment(x = 5.5, xend = 6, y = 1.7, yend = 1.7, size = .75, color = "red") +
      annotate("text", x = 5.3, y = 2, label = "truth", hjust = 1, size = 3, color = "blue") +
```

```
    annotate("text", x = 5.3, y = 1.7, label = "decision boundary", hjust = 1, size = 3, colo
```



Depth 3 decision tree:

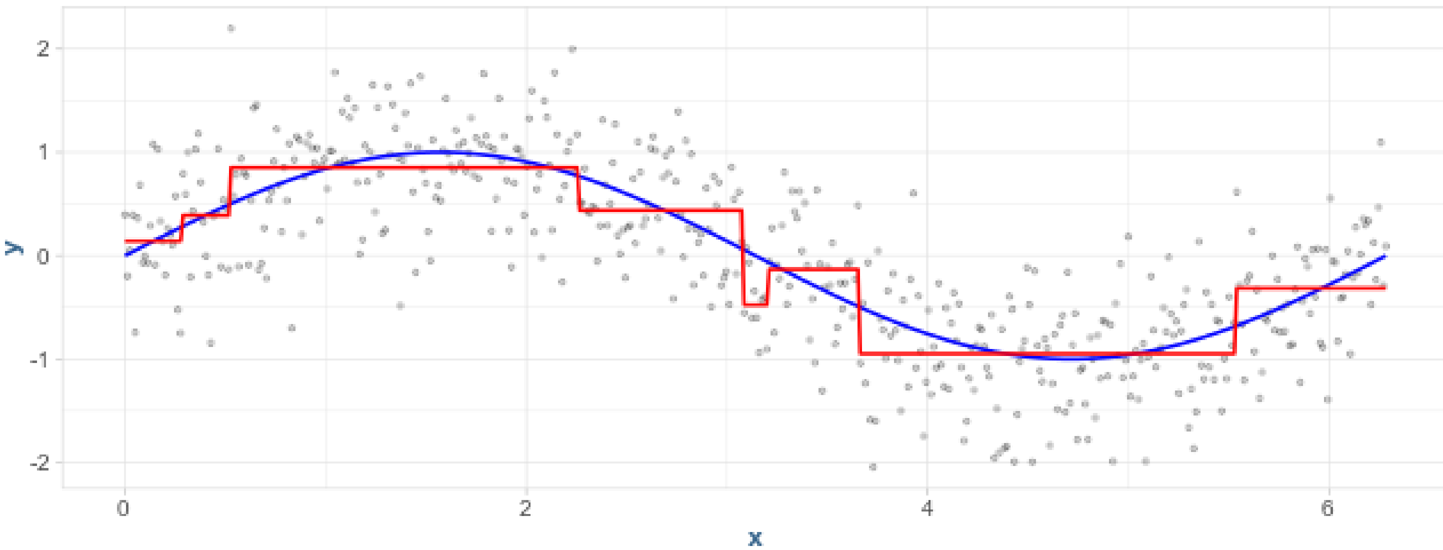
```
# fit depth 3 decision tree
ctrl <- list(cp = 0, minbucket = 5, maxdepth = 3)
fit <- rpart(y ~ x, data = df, control = ctrl)
rpart.plot(fit)
```



Decision Boundry:

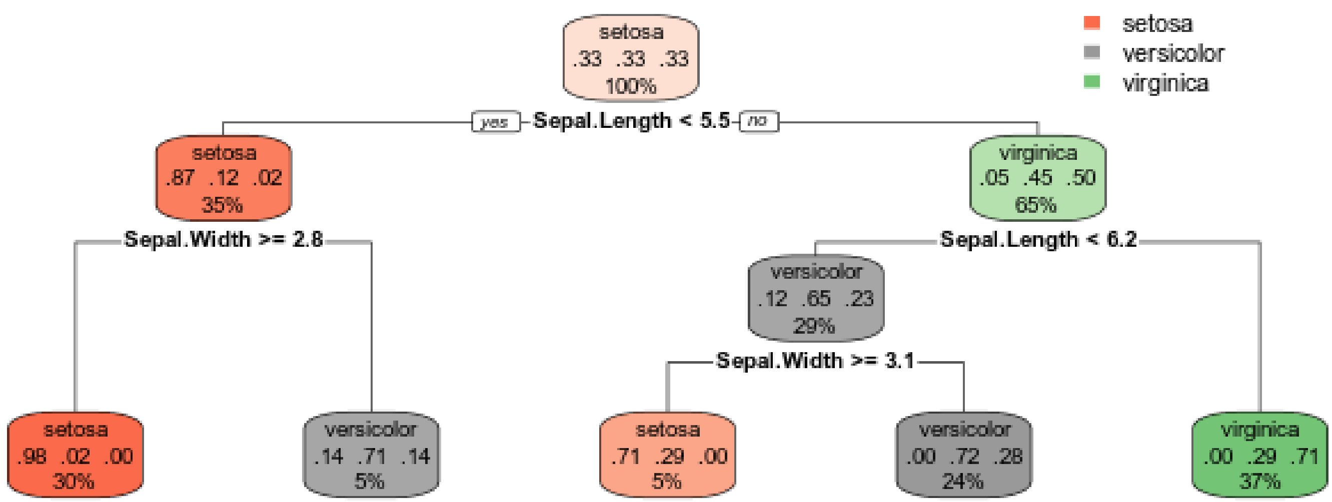
```
# plot decision boundary
df %>%
  mutate(pred = predict(fit, df)) %>%
  ggplot(aes(x, y)) +
```

```
  geom_point(alpha = .2, size = 1) +
  geom_line(aes(x, y = truth), color = "blue", size = .75) +
  geom_line(aes(y = pred), color = "red", size = .75)
```



IRIS dataset:

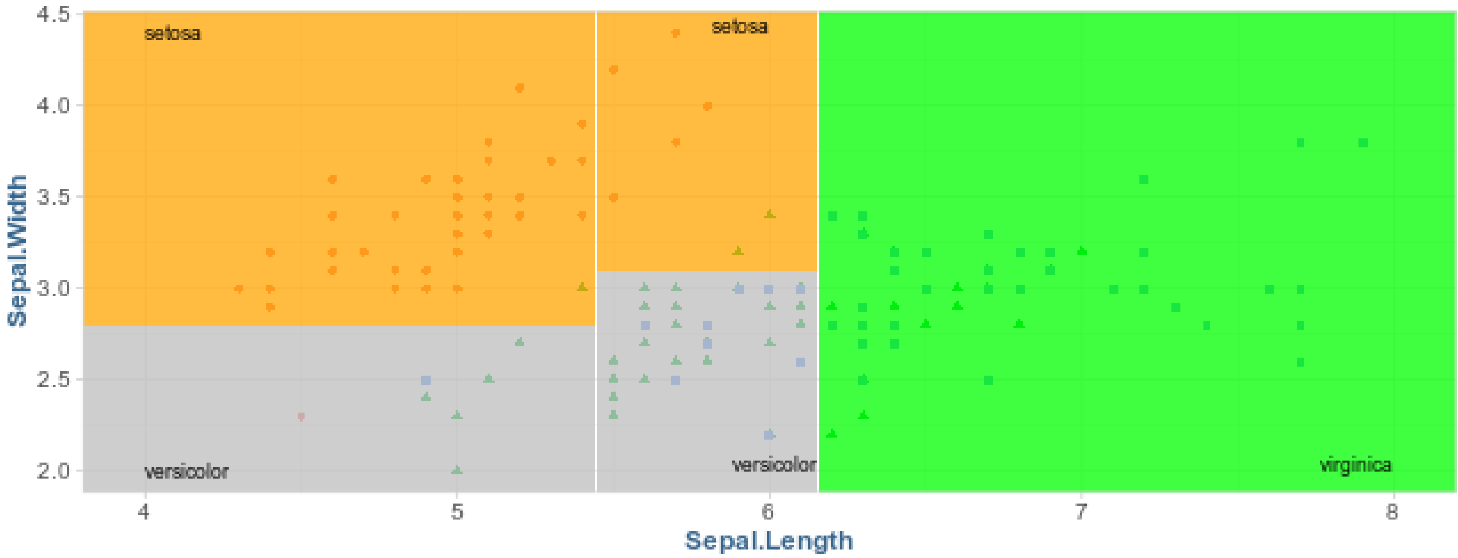
```
# decision tree
iris_fit <- rpart(Species ~ Sepal.Length + Sepal.Width, data = iris)
rpart.plot(iris_fit)
```




```
# decision boundary
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species, shape = Species)) +
  geom_point(show.legend = FALSE) +
  annotate("rect", xmin = -Inf, xmax = 5.44, ymin = 2.8, ymax = Inf, alpha = .75, fill = "orang
```

```
  annotate("text", x = 4.0, y = 4.4, label = "setosa", hjust = 0, size = 3) +
  annotate("rect", xmin = -Inf, xmax = 5.44, ymin = 2.79, ymax = -Inf, alpha = .75, fill = "gre
  annotate("text", x = 4.0, y = 2, label = "versicolor", hjust = 0, size = 3) +
  annotate("rect", xmin = 5.45, xmax = 6.15, ymin = 3.1, ymax = Inf, alpha = .75, fill = "orang
  annotate("text", x = 6, y = 4.4, label = "setosa", hjust = 1, vjust = 0, size = 3) +
  annotate("rect", xmin = 5.45, xmax = 6.15, ymin = 3.09, ymax = -Inf, alpha = .75, fill = "gre
  annotate("text", x = 6.15, y = 2, label = "versicolor", hjust = 1, vjust = 0, fill = "grey",
  annotate("rect", xmin = 6.16, xmax = Inf, ymin = -Inf, ymax = Inf, alpha = .75, fill = "green
  annotate("text", x = 8, y = 2, label = "virginica", hjust = 1, vjust = 0, fill = "green", siz
```

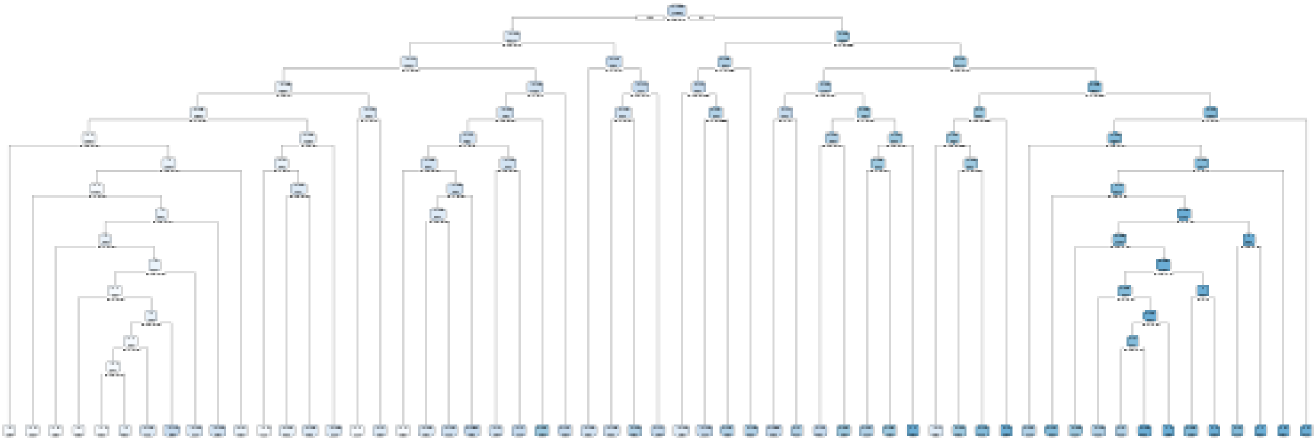
```
Warning: Ignoring unknown parameters: fill

Warning: Ignoring unknown parameters: fill
```



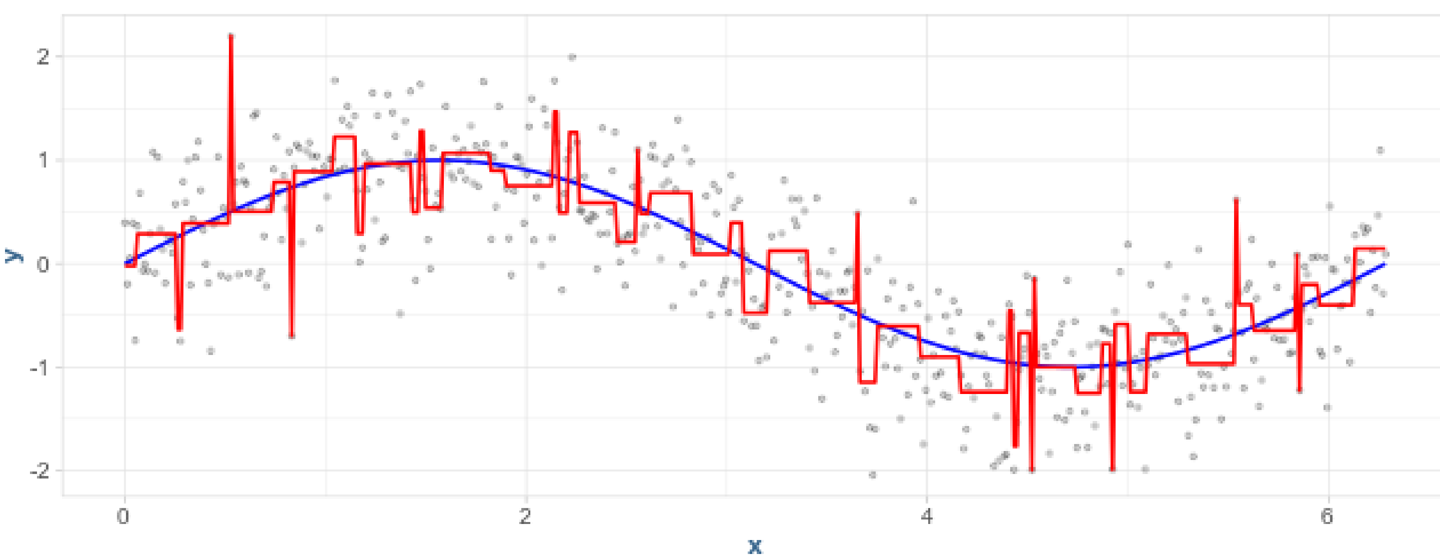
How deep?

If we keep increasing the depth of the tree, we will eventually overfit the training data.

```
ctrl <- list(cp = 0, minbucket = 1, maxdepth = 50)
fit <- rpart(y ~ x, data = df, control = ctrl)
rpart.plot(fit)
```



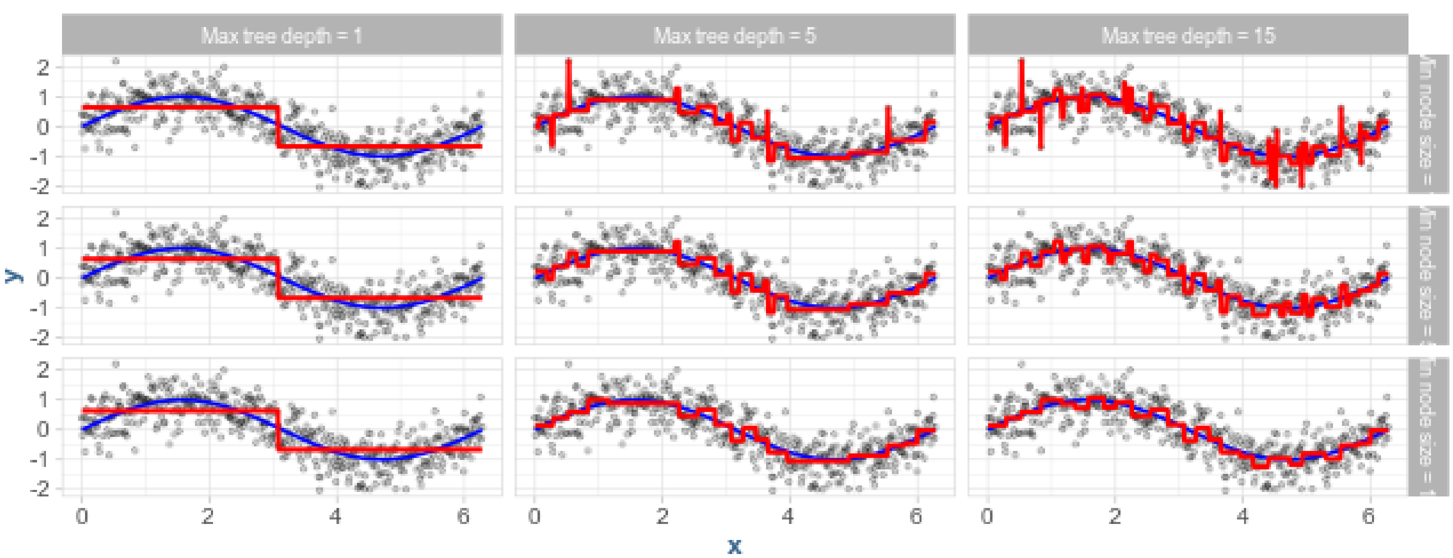
```r
df %>%
  mutate(pred = predict(fit, df)) %>%
  ggplot(aes(x, y)) +
  geom_point(alpha = .2, size = 1) +
  geom_line(aes(x, y = truth), color = "blue", size = 0.75) +
  geom_line(aes(y = pred), color = "red", size = 0.75)
```



There are two basic strategies for finding the optimal depth of the tree, early stopping and pruning:

```r
hyper.grid <- expand.grid(
   maxdepth = c(1, 5, 15),
   minbucket = c(1, 5, 15)
)
```

```
results <- data.frame(NULL)

for(i in seq_len(nrow(hyper.grid))) {
   ctrl <- list(cp = 0, maxdepth = hyper.grid$maxdepth[i], minbucket = hyper.grid$minbucket[i])
   fit <- rpart(y ~ x, data = df, control = ctrl)

   predictions <- mutate(
      df,
      minbucket = factor(paste("Min node size =", hyper.grid$minbucket[i]), ordered = T),
      maxdepth = factor(paste("Max tree depth =", hyper.grid$maxdepth[i]), ordered = T)
   )

   predictions$pred <- predict(fit, df)
   results <- rbind(results, predictions)
}

ggplot(results, aes(x, y)) +
   geom_point(alpha = .2, size = 1) +
   geom_line(aes(x, y = truth), color = "blue", size = .75) +
   geom_line(aes(y = pred), color = "red", size = 1) +
   facet_grid(minbucket ~ maxdepth)
```



Pruning

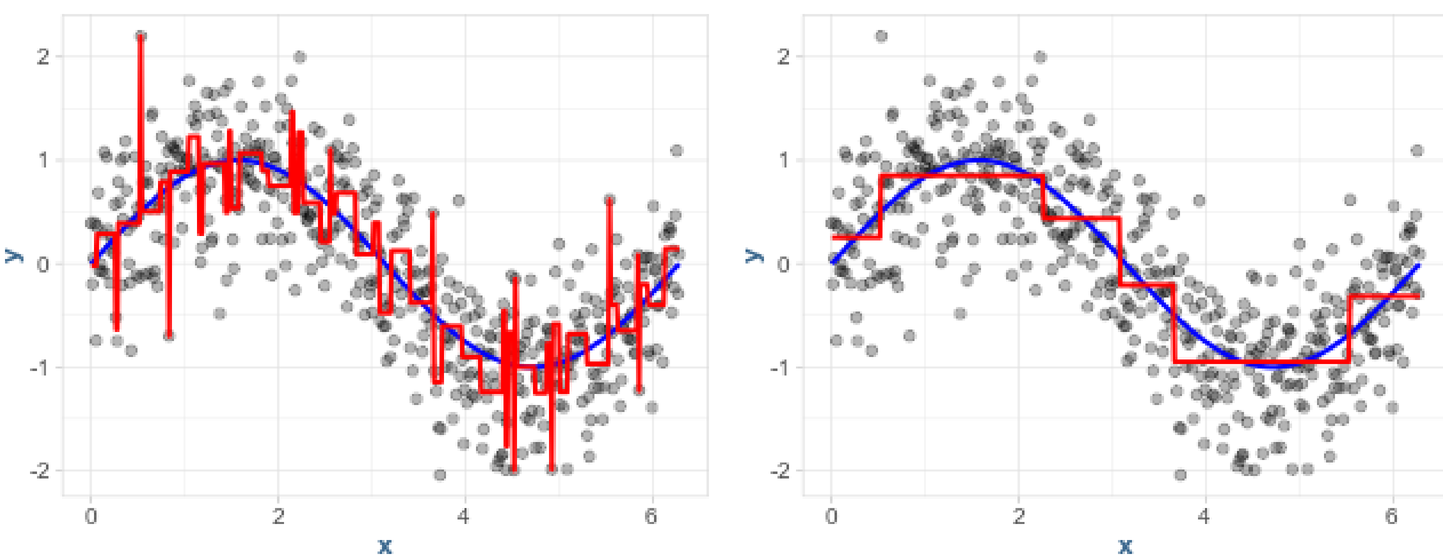
Alternative to specifying a max depth, build the most complicated tree and then prune it back for generalizability.

We find the optimal subtree by using a *cost complexity parameter* (α) that penalizes our objective function:

$minimize\{SST + \alpha|T|\}$

(Similar to Lasso regression)

```
ctrl <- list(cp = 0, minbucket = 1, maxdepth = 50)
fit <- rpart(y ~ x, data = df, control = ctrl)

p1 <- df %>%
  mutate(pred = predict(fit, df)) %>%
  ggplot(aes(x, y)) +
  geom_point(alpha = .3, size = 2) +
  geom_line(aes(x, y = truth), color = "blue", size = 1) +
  geom_line(aes(y = pred), color = "red", size = 1)

fit2 <- rpart(y ~ x, data = df)

p2 <- df %>%
  mutate(pred2 = predict(fit2, df)) %>%
  ggplot(aes(x, y)) +
  geom_point(alpha = .3, size = 2) +
  geom_line(aes(x, y = truth), color = "blue", size = 1) +
  geom_line(aes(y = pred2), color = "red", size = 1)

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



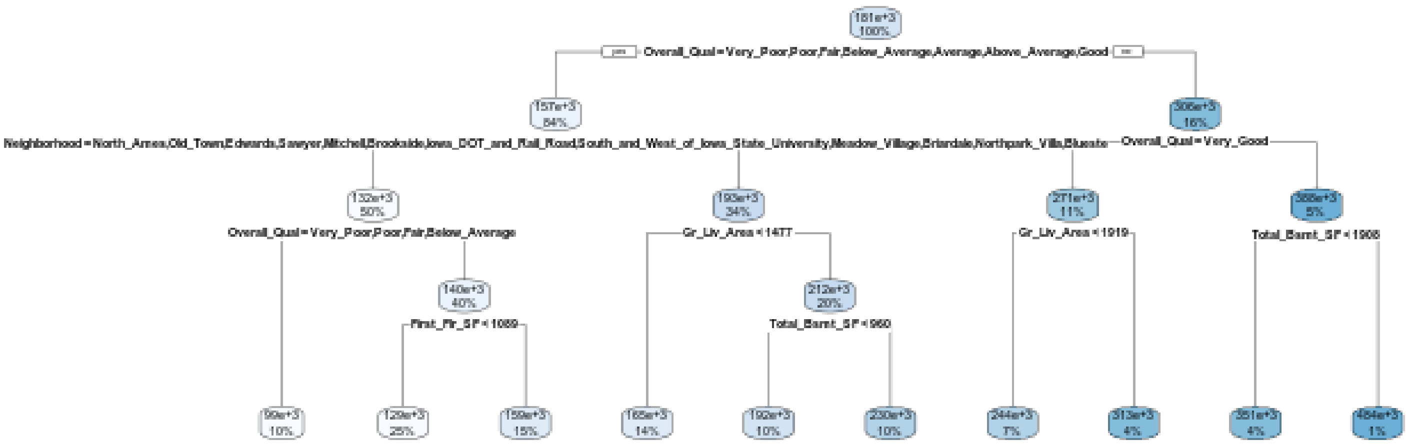
Example: AMES Housing Data

```
ames.dt1 <- rpart(
   formula = Sale_Price ~ .,
   data = ames.train,
```

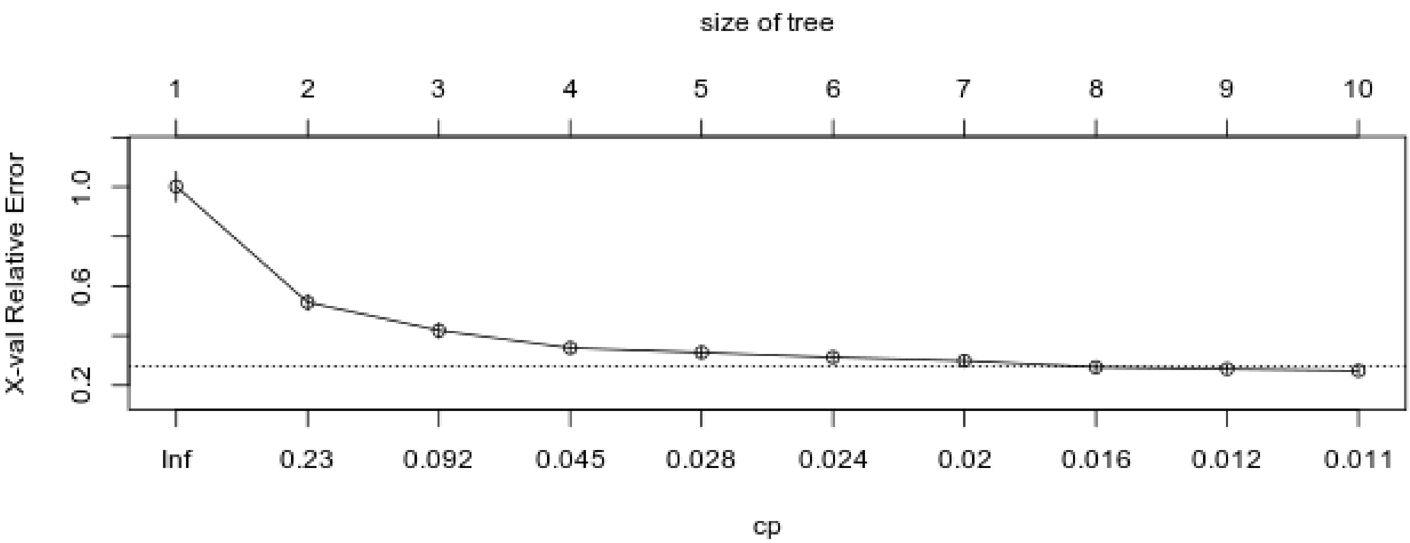
```
  method = "anova"
)

ames.dt1
```

```
n= 2053

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

 1) root 2053 13217940000000 180996.30
   2) Overall_Qual=Very_Poor,Poor,Fair,Below_Average,Average,Above_Average,Good 1722  410788800
     4) Neighborhood=North_Ames,Old_Town,Edwards,Sawyer,Mitchell,Brookside,Iowa_DOT_and_Rail_Ro
       8) Overall_Qual=Very_Poor,Poor,Fair,Below_Average 199   179295400000  98856.51 *
       9) Overall_Qual=Average,Above_Average,Good 823   876239900000 140409.00
        18) First_Flr_SF< 1089 517   290531200000 129244.00 *
        19) First_Flr_SF>=1089 306   412375700000 159272.60 *
     5) Neighborhood=College_Creek,Somerset,Northridge_Heights,Gilbert,Northwest_Ames,Sawyer_We
      10) Gr_Liv_Area< 1477 287   250826800000 165395.90 *
      11) Gr_Liv_Area>=1477 413   630227700000 212054.00
        22) Total_Bsmt_SF< 959.5 199   139087700000 192493.10 *
        23) Total_Bsmt_SF>=959.5 214   344191200000 230243.70 *
   3) Overall_Qual=Very_Good,Excellent,Very_Excellent 331  2936700000000 306070.70
     6) Overall_Qual=Very_Good 231   946974600000 270626.10
      12) Gr_Liv_Area< 1919 142   334978300000 244016.60 *
      13) Gr_Liv_Area>=1919 89   351030800000 313081.60 *
     7) Overall_Qual=Excellent,Very_Excellent 100  1029126000000 387948.00
      14) Total_Bsmt_SF< 1907.5 72   314985900000 350532.40 *
      15) Total_Bsmt_SF>=1907.5 28   354159900000 484159.40 *
```

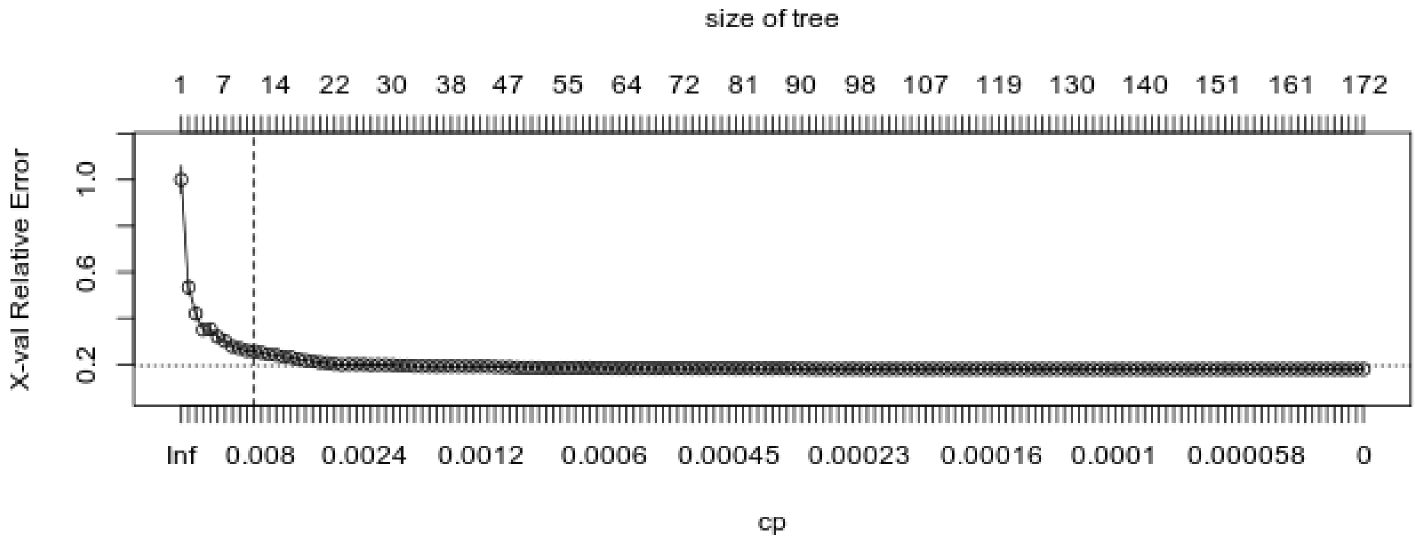
```
rpart.plot(ames.dt1)
```




```r
plotcp(ames.dt1)
```




```r
ames.dt2 <- rpart(
    formula = Sale_Price ~ .,
    data    = ames.train,
    method  = "anova",
    control = list(cp = 0, xval = 10)
)

plotcp(ames.dt2)
abline(v = 11, lty = "dashed")
```




```r
ames.dt1$cptable
```

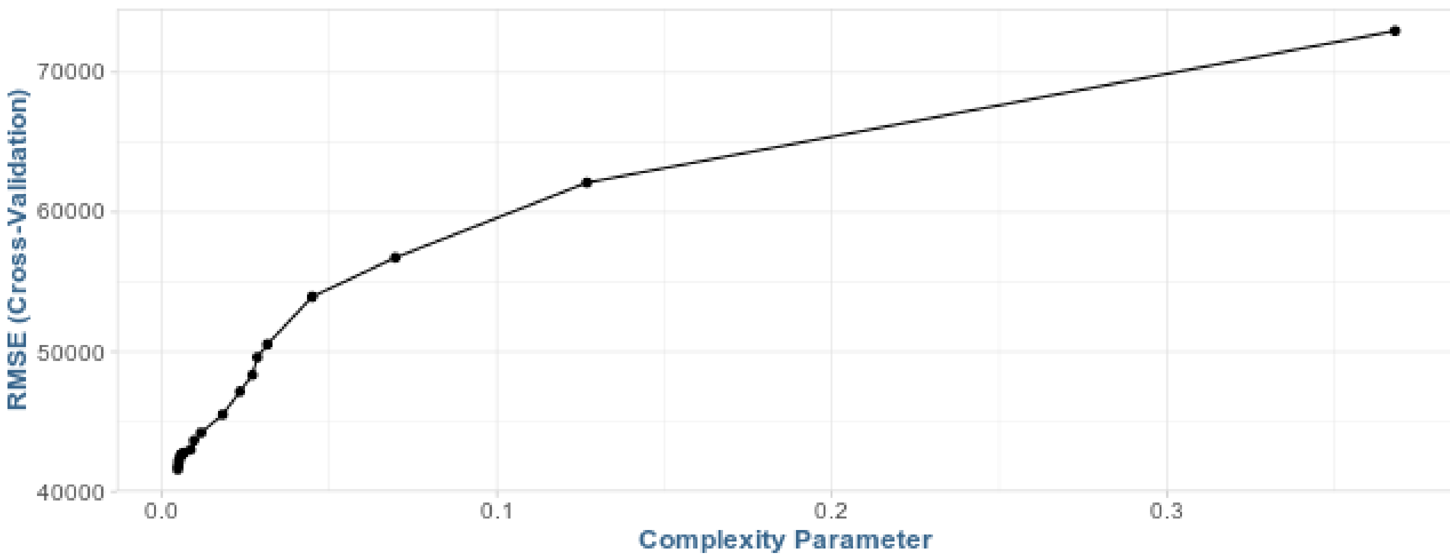
```
           CP nsplit rel error    xerror       xstd
1  0.46704344      0 1.0000000 1.0015334 0.06051267
2  0.11544770      1 0.5329566 0.5343697 0.03079312
3  0.07267387      2 0.4175089 0.4209603 0.03007122
4  0.02788834      3 0.3448350 0.3502963 0.02145751
5  0.02723422      4 0.3169466 0.3319341 0.02225037
6  0.02093301      5 0.2897124 0.3125117 0.02150290
7  0.01974328      6 0.2687794 0.2986956 0.02139660
8  0.01311346      7 0.2490361 0.2726862 0.01738257
9  0.01111737      8 0.2359227 0.2654669 0.01725615
10 0.01000000      9 0.2248053 0.2584346 0.01721996
```

Cross-validated parameter search:

```r
ames.dt3 <- train(
   Sale_Price ~ .,
   data = ames.train,
   method = "rpart",
   trControl = trainControl(method = "cv", number = 10),
   tuneLength = 20
)
```

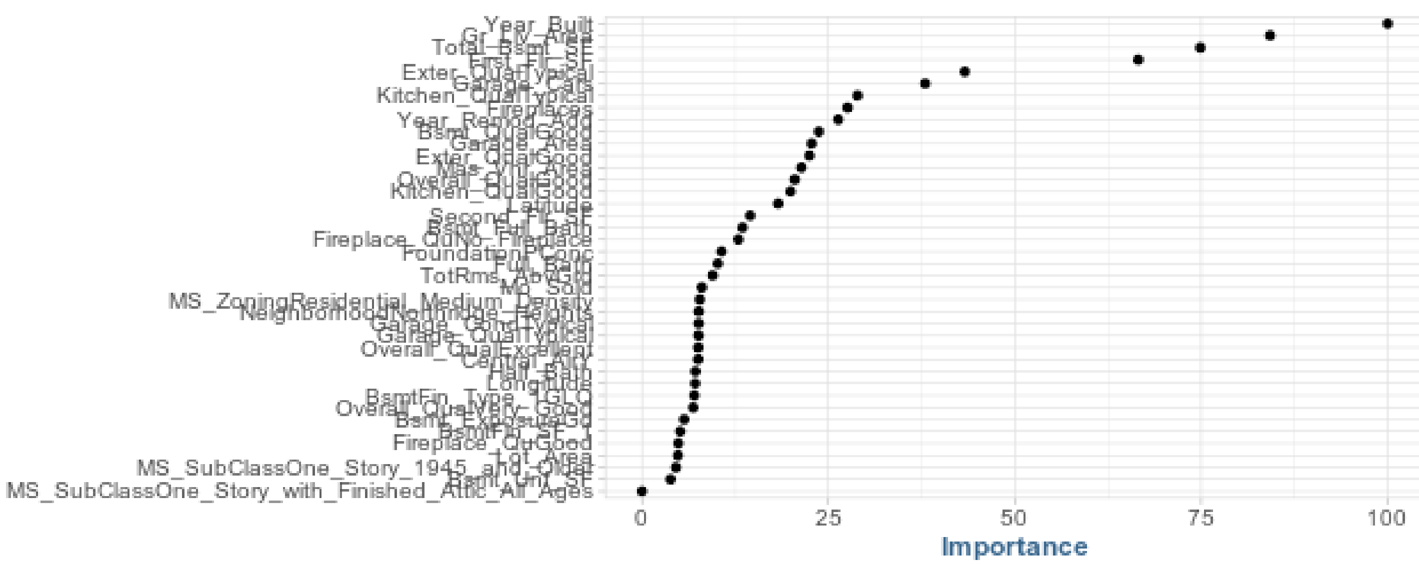
```
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
There were missing values in resampled performance measures.
```

```r
ggplot(ames.dt3)
```



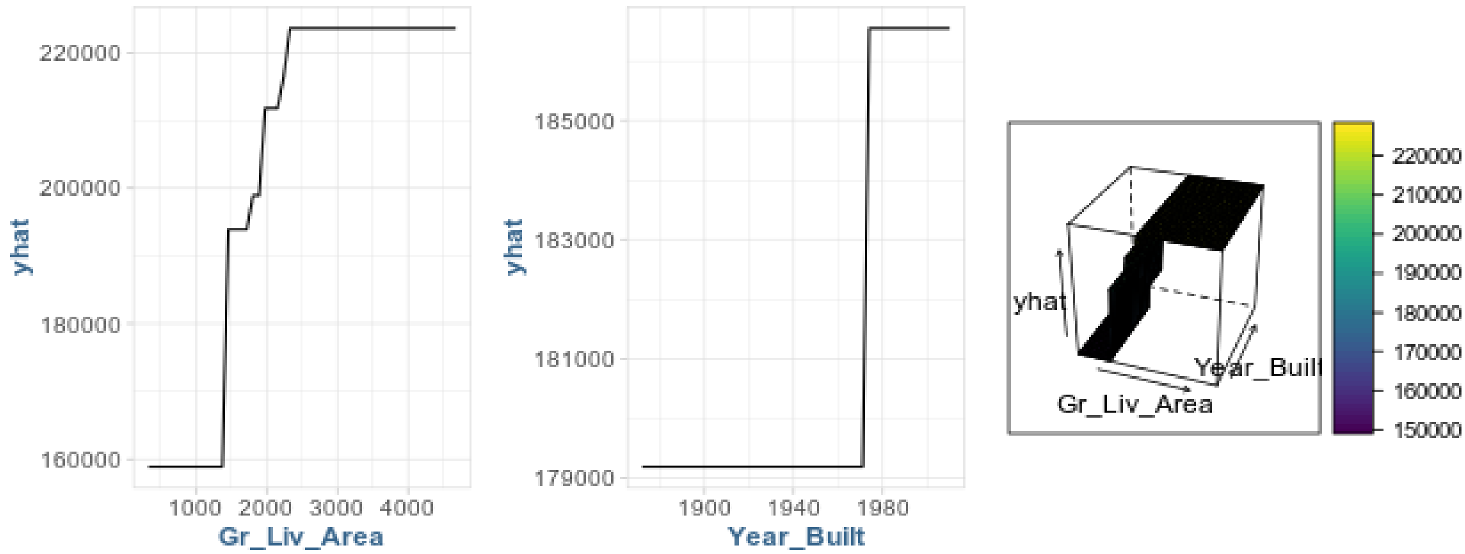
Feature Interpretation

```r
vip(ames.dt3, num_features = 40, geom = "point")
```




```r
# Construct partial dependence plots
p1 <- pdp::partial(ames.dt3, pred.var = "Gr_Liv_Area") %>% autoplot()
p2 <- pdp::partial(ames.dt3, pred.var = "Year_Built") %>% autoplot()
p3 <- pdp::partial(ames.dt3, pred.var = c("Gr_Liv_Area", "Year_Built")) %>%
  plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
              colorkey = TRUE, screen = list(z = -20, x = -60))

# Display plots side by side
```

```r
gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
```




```r
# clean up
rm(list = ls())
```