# Bagging

## 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)
```

## Bagging Overview

Bagging (Bootstrap AGGregatING) is a procedure that uses bootstrapping techniques to create an ensemble of predictions to improve the accuracy of regression and classification methods.

## How Bagging Works

Bagging is a fairly straightforward algorithm that makes *b* bootstrap copies of the original training data, to which the (regression or classification) algorithm is applied (referred to as the “Base Learner”), and then the predictions are averaged together from the individual base learners.

For each record, X, we want to predict:

$\hat{f_{bag}}$ is the bagged prediction, and $f_1\hat{(X)}, f_2\hat{(X)}, \ldots, f_b\hat{(X)}$ are the predictions from the individual base learners.

So that,

$f(\hat{bag}) = f_1\hat{(X)} + f_2\hat{(X)} + \ldots + f_b\hat{(X)}$

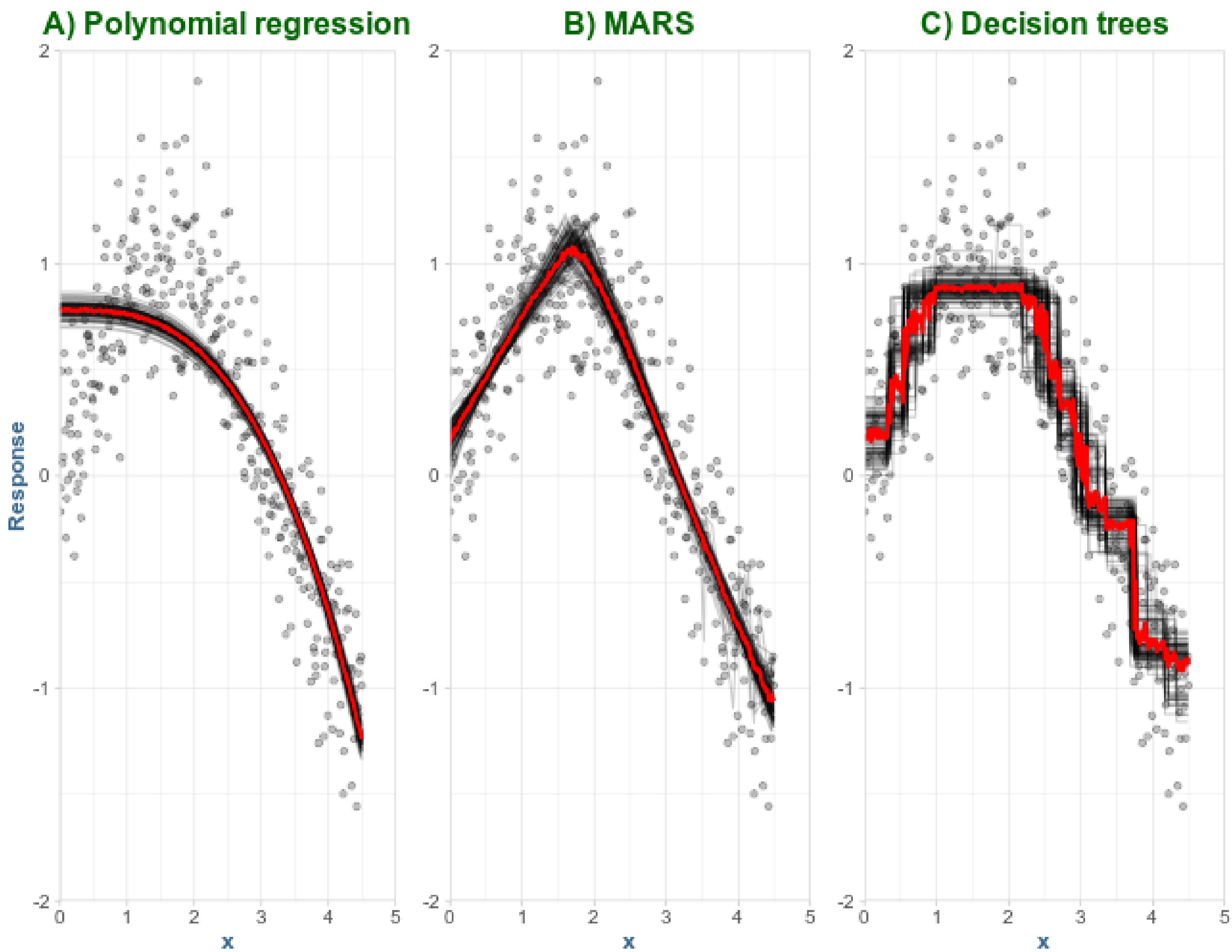
Bagging effectively reduces the variance of an individual base learning through the aggrigation process. So, this process works especially well for high variance base learners (Decision Tress, KNN (small k)).

This is an example of the “Wisdom of the Crowd” approach to learners.

Example of “Bagging”:

```
# Simulate some nonlinear monotonic data
set.seed(123)  # for reproducibility

x <- seq(from = 0, to = 2 * pi, length = 500)
y <- sin(x) + rnorm(length(x), sd = 0.3)
df <- data.frame(x, y) %>%
  filter(x < 4.5)

# bootstrapped polynomial model fit
bootstrap_n <- 100
bootstrap_results <- NULL
for(i in seq_len(bootstrap_n)) {
  # reproducible sampled data frames
  set.seed(i)
  index <- sample(seq_len(nrow(df)), nrow(df), replace = TRUE)
  df_sim <- df[index, ]

  # fit model and add predictions to results data frame
  fit <- lm(y ~ I(x^3), data = df_sim)
  df_sim$predictions <- predict(fit, df_sim)
  df_sim$model <- paste0("model", i)
  df_sim$ob <- index
  bootstrap_results <- rbind(bootstrap_results, df_sim)
}

p1 <- ggplot(bootstrap_results, aes(x, predictions)) +
  geom_point(data = df, aes(x, y), alpha = .25) +
  geom_line(aes(group = model), show.legend = FALSE, size = .5, alpha = .2) +
  stat_summary(fun.y = "mean", colour = "red", size = 1, geom = "line") +
  scale_y_continuous("Response", limits = c(-2, 2), expand = c(0, 0)) +
  scale_x_continuous(limits = c(0, 5), expand = c(0, 0)) +
  ggtitle("A) Polynomial regression")

# bootstrapped MARS model fit
bootstrap_n <- 100
bootstrap_results <- NULL
for(i in seq_len(bootstrap_n)) {
  # reproducible sampled data frames
  set.seed(i)
  index <- sample(seq_len(nrow(df)), nrow(df), replace = TRUE)
  df_sim <- df[index, ]
```

```r
  # fit model and add predictions to results data frame
  fit <- earth::earth(y ~ x, data = df_sim)
  df_sim$predictions <- predict(fit, df_sim)
  df_sim$model <- paste0("model", i)
  df_sim$ob <- index
  bootstrap_results <- rbind(bootstrap_results, df_sim)
}

p2 <- ggplot(bootstrap_results, aes(x, predictions)) +
  geom_point(data = df, aes(x, y), alpha = .25) +
  geom_line(aes(group = model), show.legend = FALSE, size = .5, alpha = .2) +
  stat_summary(fun.y = "mean", colour = "red", size = 1, geom = "line") +
  scale_y_continuous(NULL, limits = c(-2, 2), expand = c(0, 0)) +
  scale_x_continuous(limits = c(0, 5), expand = c(0, 0)) +
  ggtitle("B) MARS")

# bootstrapped decision trees fit
bootstrap_n <- 100
bootstrap_results <- NULL
for(i in seq_len(bootstrap_n)) {
  # reproducible sampled data frames
  set.seed(i)
  index <- sample(seq_len(nrow(df)), nrow(df), replace = TRUE)
  df_sim <- df[index, ]

  # fit model and add predictions to results data frame
  fit <- rpart::rpart(y ~ x, data = df_sim)
  df_sim$predictions <- predict(fit, df_sim)
  df_sim$model <- paste0("model", i)
  df_sim$ob <- index
  bootstrap_results <- rbind(bootstrap_results, df_sim)
}

p3 <- ggplot(bootstrap_results, aes(x, predictions)) +
  geom_point(data = df, aes(x, y), alpha = .25) +
  geom_line(aes(group = model), show.legend = FALSE, size = .5, alpha = .2) +
  stat_summary(fun.y = "mean", colour = "red", size = 1, geom = "line") +
  scale_y_continuous(NULL, limits = c(-2, 2), expand = c(0, 0)) +
  scale_x_continuous(limits = c(0, 5), expand = c(0, 0)) +
  ggtitle("C) Decision trees")

gridExtra::grid.arrange(p1, p2, p3, nrow = 1)
```




Optimal performance is typically found using 50-500 trees.

Interestingly, bagging decision trees does not tend to overfit, however, using a CV approach with bagging does begin to become computationally burdensome.

The out-of-bag (OOB) is an internal estimate of the predictive performance. Think of this as a “free” CV statistic.

### Implementation

Here we will use a bag of unprunded (we don’t prune the individual trees because the bagging process will reduce the variance, and therefore have the largest impact) decision trees on the ames data set.

```
set.seed(123)

# train bagged model
```

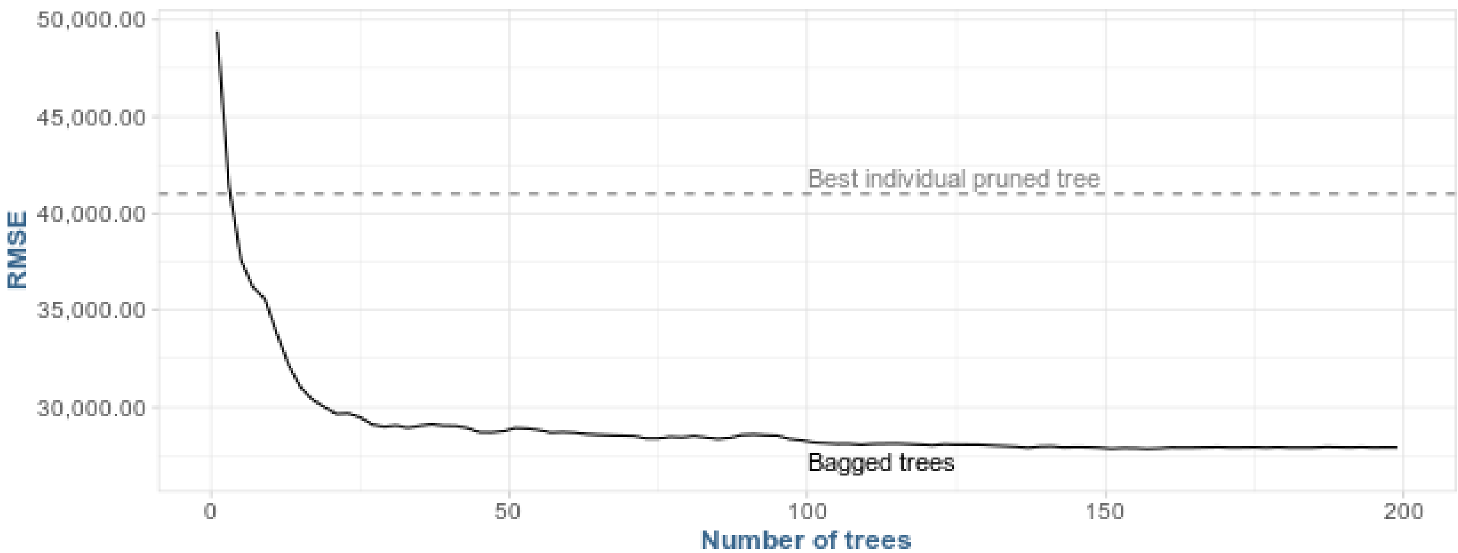
```
ames.bag1 <- bagging(
  formula = Sale_Price ~ .,
  data = ames.train,
  nbagg = 100,
  coob = T,
  control = rpart.control(minsplit = 2, cp = 0)
)

ames.bag1
```

```
Bagging regression trees with 100 bootstrap replications

Call: bagging.data.frame(formula = Sale_Price ~ ., data = ames.train,
    nbagg = 100, coob = T, control = rpart.control(minsplit = 2,
        cp = 0))

Out-of-bag estimate of root mean squared error:  27767.13
```

Error curve for bagging 1-200 deep, unpruned decision trees:

```
n.tree <- seq(1, 200, by = 2)

rmse <- vector(mode = "numeric", length = length(n.tree))

for(i in seq_along(n.tree))
{
  set.seed(123)

  # perform bagged model
  model <- ranger::ranger(
    formula = Sale_Price ~ .,
    data = ames.train,
    num.trees = n.tree[i],
    mtry = ncol(ames.train) - 1,
    min.node.size = 1
  )

  rmse[i] <- sqrt(model$prediction.error)
}

bagging.errors <- data.table(n.tree, rmse)

ggplot(bagging.errors, aes(n.tree, rmse)) +
  geom_line() +
```

```
  geom_hline(yintercept = 41019, lty = "dashed", color = "grey50") +
  annotate("text", x = 100, y = 41385, label = "Best individual pruned tree", vjust = 0, hjust
  annotate("text", x = 100, y = 26750, label = "Bagged trees", vjust = 0, hjust = 0) +
  scale_y_continuous(labels = comma) +
  ylab("RMSE") +
  xlab("Number of trees")
```




```
ames.bag2 <- train(
  Sale_Price ~ .,
  data = ames.train,
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 200,
  control = rpart.control(minsplit = 2, cp = 0)
)

ames.bag2
```

```
Bagged CART

2053 samples
  80 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1847, 1849, 1847, 1847, 1848, 1847, ...
Resampling results:

  RMSE      Rsquared   MAE
```

```
  27460.99  0.8862784  16755.65
```

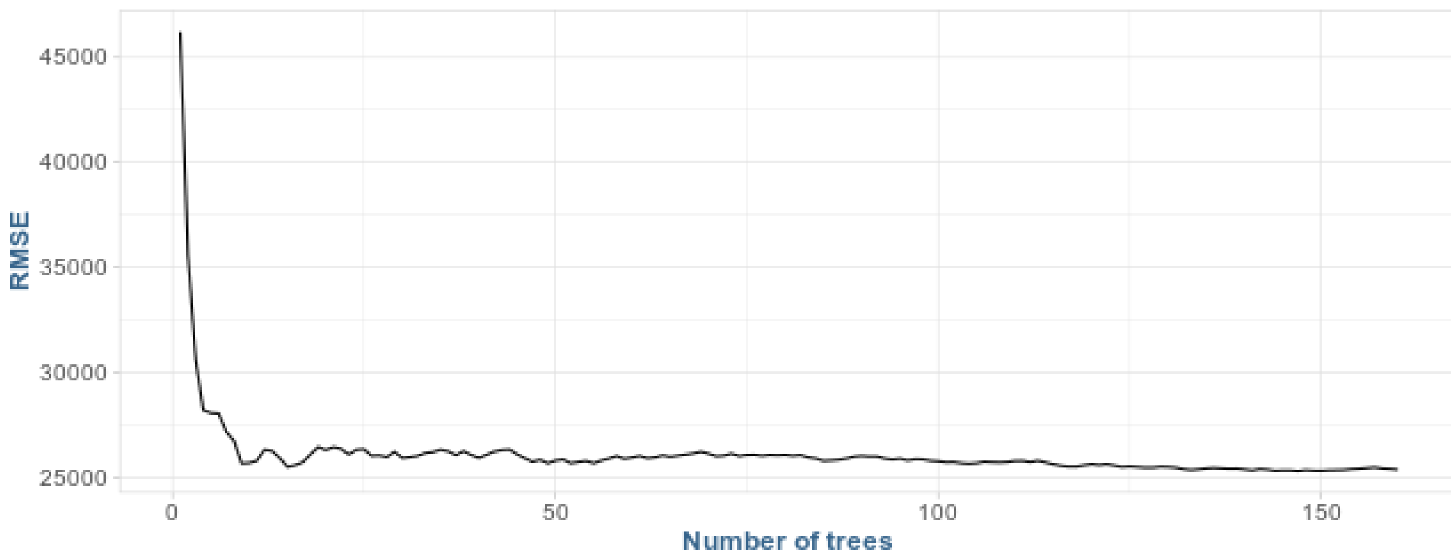
Since each tree is trained on independent samples of the data, we can easily parallelize this process.

```r
cl <- makeCluster(8)

registerDoParallel(cl)

# fit trees in parallel and compute predictions on the test set

predictions <- foreach(
  icount(160),
  .packages = "rpart",
  .combine = cbind
) %dopar% {
    # bootstrap copy of training data
    index <- sample(nrow(ames.train), replace = T)
    ames.train.boot <- ames.train[index, ]

    # fit tree to bootstrap copy
    bagged.tree <- rpart(
      Sale_Price ~ .,
      control = rpart.control(minsplit = 2, cp = 0),
      data = ames.train.boot
    )

    predict(bagged.tree, newdata = ames.test)
}

predictions[1:5, 1:7]
```

```
  result.1 result.2 result.3 result.4 result.5 result.6 result.7
1   375000   206900   165000   243000   206900   226500   167000
2   173000   183000   196500   235000   185000   193000   173000
3   189000   213750   201000   187500   180000   183600   130000
4   174000   162500   173000   165500   156820   167900   173000
5   278000   226500   301000   187500   239500   251000   130000
```

```r
predictions %>%
  as.data.frame() %>%
  mutate(
    observation = 1:n(),
    actual = ames.test$Sale_Price) %>%
  tidyr::gather(tree, predicted, -c(observation, actual)) %>%
  group_by(observation) %>%
  mutate(tree = stringr::str_extract(tree, '\\d+') %>% as.numeric()) %>%
```

```
  ungroup() %>%
  arrange(observation, tree) %>%
  group_by(observation) %>%
  mutate(avg_prediction = cummean(predicted)) %>%
  group_by(tree) %>%
  summarize(RMSE = RMSE(avg_prediction, actual)) %>%
  ggplot(aes(tree, RMSE)) +
  geom_line() +
  xlab('Number of trees')
```



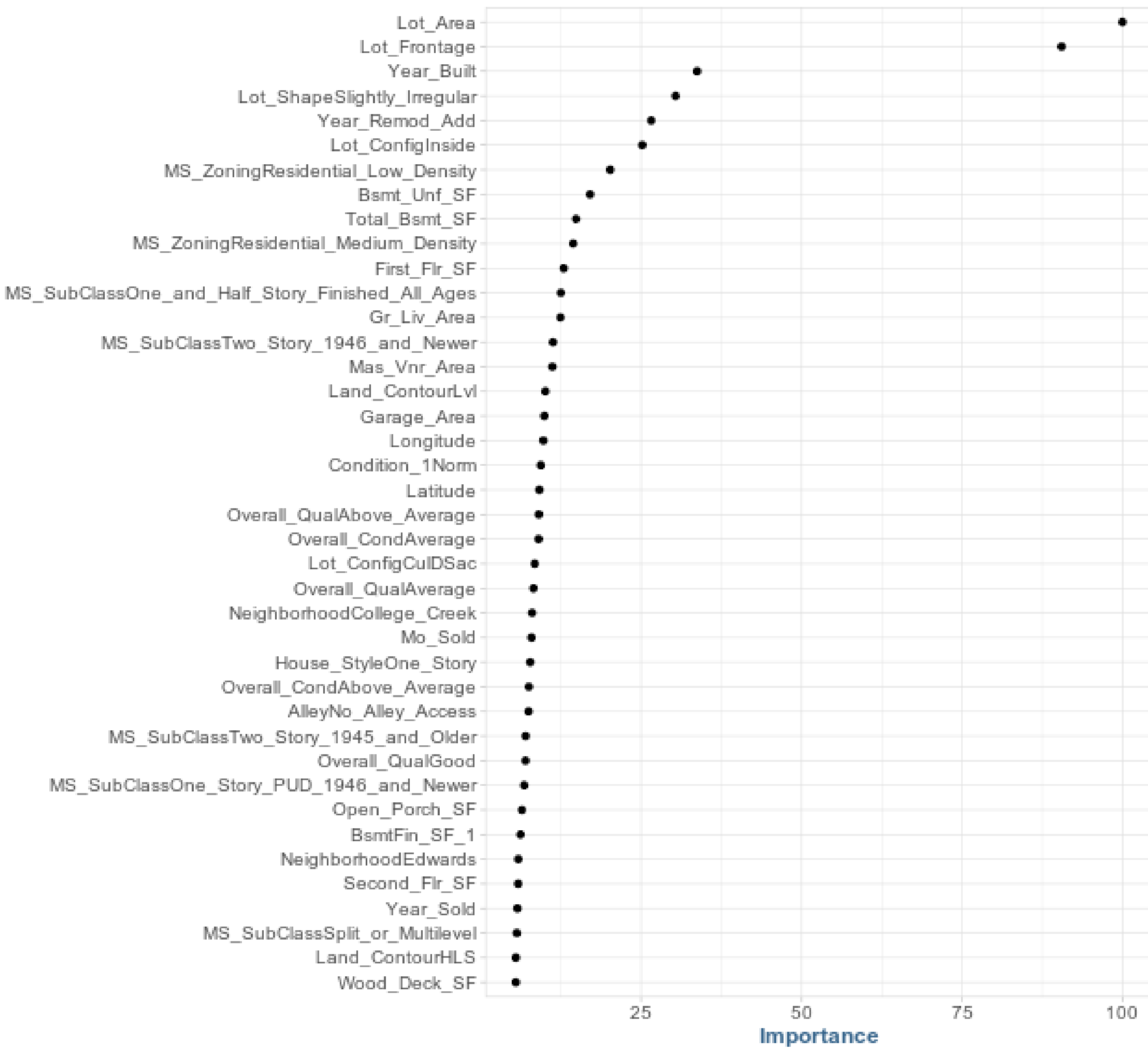

```
stopCluster(cl)
```

### Feature Interpretation

Unfortunatly, due to the many trees involved feature interpretation becomes tricky. Typically, the bagging process will use many features, at lower levels of importance.

```
vip::vip(ames.bag2, num_features = 40, bar = FALSE)
```

```
Warning in vip.default(ames.bag2, num_features = 40, bar = FALSE): The `bar`
argument has been deprecated in favor of the new `geom` argument. It will be
removed in version 0.3.0.
```




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