

STA 235H - Prediction: Bagging, Random Forests, and Boosting

Fall 2021

McCombs School of Business, UT Austin

Announcements

Homework 6 due on Thursday

- **Grades for Homework 5** have been posted: Check the returned submissions!
 - Bi-modal distribution of grades
 - Work as a group!
- **Final Exam** is cumulative: Due on Wednesday Dec 8th
 - It should take you 3 hrs at most to complete.
 - It will be posted Monday Dec 6th (8:00am)
 - No extensions

Announcements (Cont.)

Re-grading only until Friday Dec
3rd

What we have seen...



- **Decision trees:**
 - Classification and Regression Trees
 - When to split? Complexity parameter
 - Advantages and disadvantages.

What we'll cover today

- **Ensemble methods:**

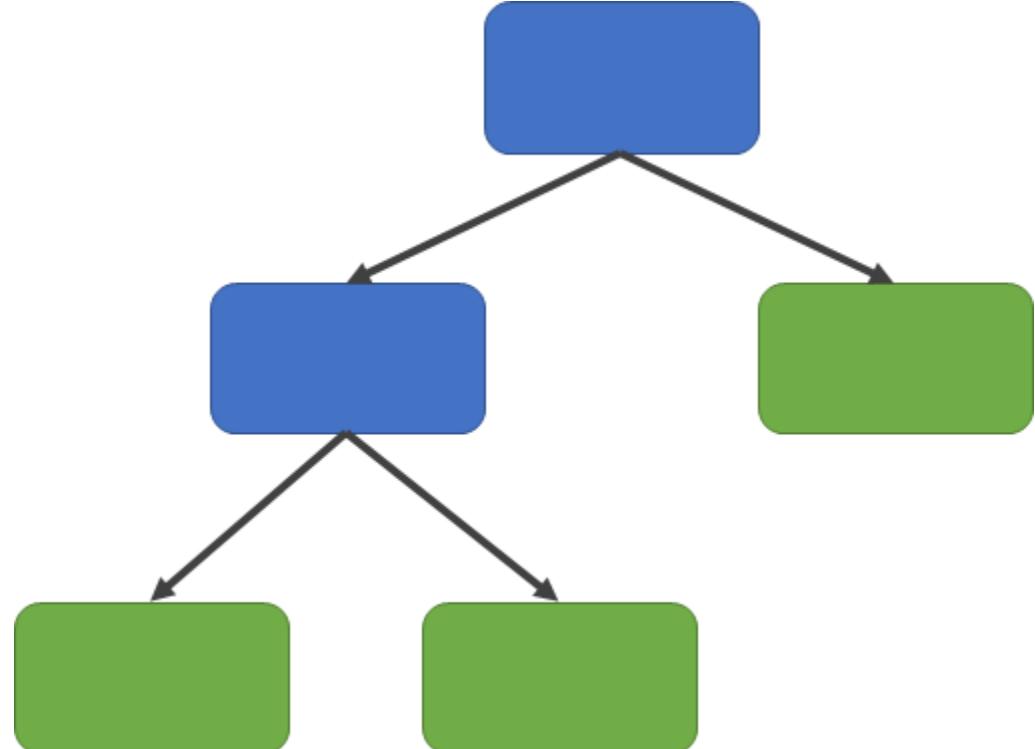
- Bagging (e.g. tree bagging)
- Random Forests
- Boosting



Quick recap on trees

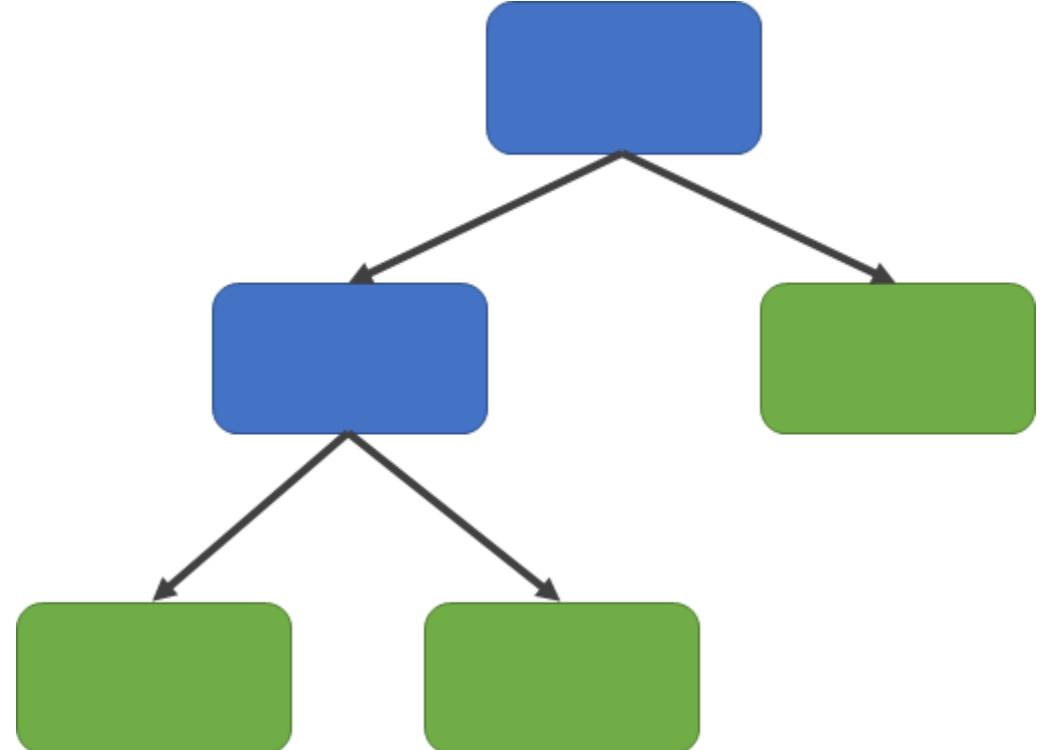
Quick refresher on decision trees

- A decision tree is a structure that **works like a flowchart**
- You start at the **root node**, make your way down the branches through the **(internal) nodes**, and get to the **leaves (terminal nodes)**.
 - At the leaves is where prediction happens!



To split or not to split

- In general, we will only increase the size of our tree (additional split) **if we gain some additional information for prediction**
- We will actually grow **full trees** and then **prune** them.
- How do we measure that information gain?
 - **Classification:** Impurity measure (like Gini Index).
 - **Regression:** Decrease in RSS.



In term of bias, which one is better: A deep tree or a shallow tree?

Let's look at an example: Car seat prices

```
library(ISLR)
data(Carseats)

head(Carseats)

##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50      138     73          11       276    120      Bad    42        17
## 2 11.22      111     48          16       260     83      Good    65        10
## 3 10.06      113     35          10       269     80    Medium    59        12
## 4  7.40      117    100          4       466     97    Medium    55        14
## 5  4.15      141     64          3       340    128      Bad    38        13
## 6 10.81      124    113          13       501     72      Bad    78        16
##   Urban US
## 1  Yes Yes
## 2  Yes Yes
## 3  Yes Yes
## 4  Yes Yes
## 5  Yes  No
## 6  No  Yes
```

Do you wanna build a... tree?

```
library(caret)
library(rattle)
library(rsample)
library(modelr)

set.seed(100)

split <- initial_split(Carseats, prop = 0.7, strata = "Sales")

carseats.train <- training(split)
carseats.test <- testing(split)

tuneGrid <- expand.grid(cp = seq(0, 0.01, 0.0001))

mcv <- train(Sales ~., data = carseats.train, method = "rpart",
  trControl = trainControl("cv", number = 10), tuneGrid = tuneGrid)
```

Do you wanna build a... tree?

```
library(caret)
library(rattle)
library(rsample)
library(modelr)

set.seed(100)

split <- initial_split(Carseats, prop = 0.7, strata = "Sales")

carseats.train <- training(split)
carseats.test <- testing(split)

tuneGrid <- expand.grid(cp = seq(0, 0.01, 0.0001))

mcv <- train(Sales ~., data = carseats.train, method = "rpart",
  trControl = trainControl("cv", number = 10), tuneGrid = tuneGrid)
```

Do you wanna build a... tree?

```
library(caret)
library(rattle)
library(rsample)
library(modelr)

set.seed(100)

split <- initial_split(Carseats, prop = 0.7, strata = "Sales")

carseats.train <- training(split)
carseats.test <- testing(split)

tuneGrid <- expand.grid(cp = seq(0, 0.01, 0.0001))

mcv <- train(Sales ~., data = carseats.train, method = "rpart",
  trControl = trainControl("cv", number = 10), tuneGrid = tuneGrid)
```

Do you wanna build a... tree?

```
library(caret)
library(rattle)
library(rsample)
library(modelr)

set.seed(100)

split <- initial_split(Carseats, prop = 0.7, strata = "Sales")

carseats.train <- training(split)
carseats.test <- testing(split)

tuneGrid <- expand.grid(cp = seq(0, 0.01, 0.0001))

mcv <- train(Sales ~., data = carseats.train, method = "rpart",
  trControl = trainControl("cv", number = 10), tuneGrid = tuneGrid)
```

Do you wanna build a... tree?

```
library(caret)
library(rattle)
library(rsample)
library(modelr)

set.seed(100)

split <- initial_split(Carseats, prop = 0.7, strata = "Sales")

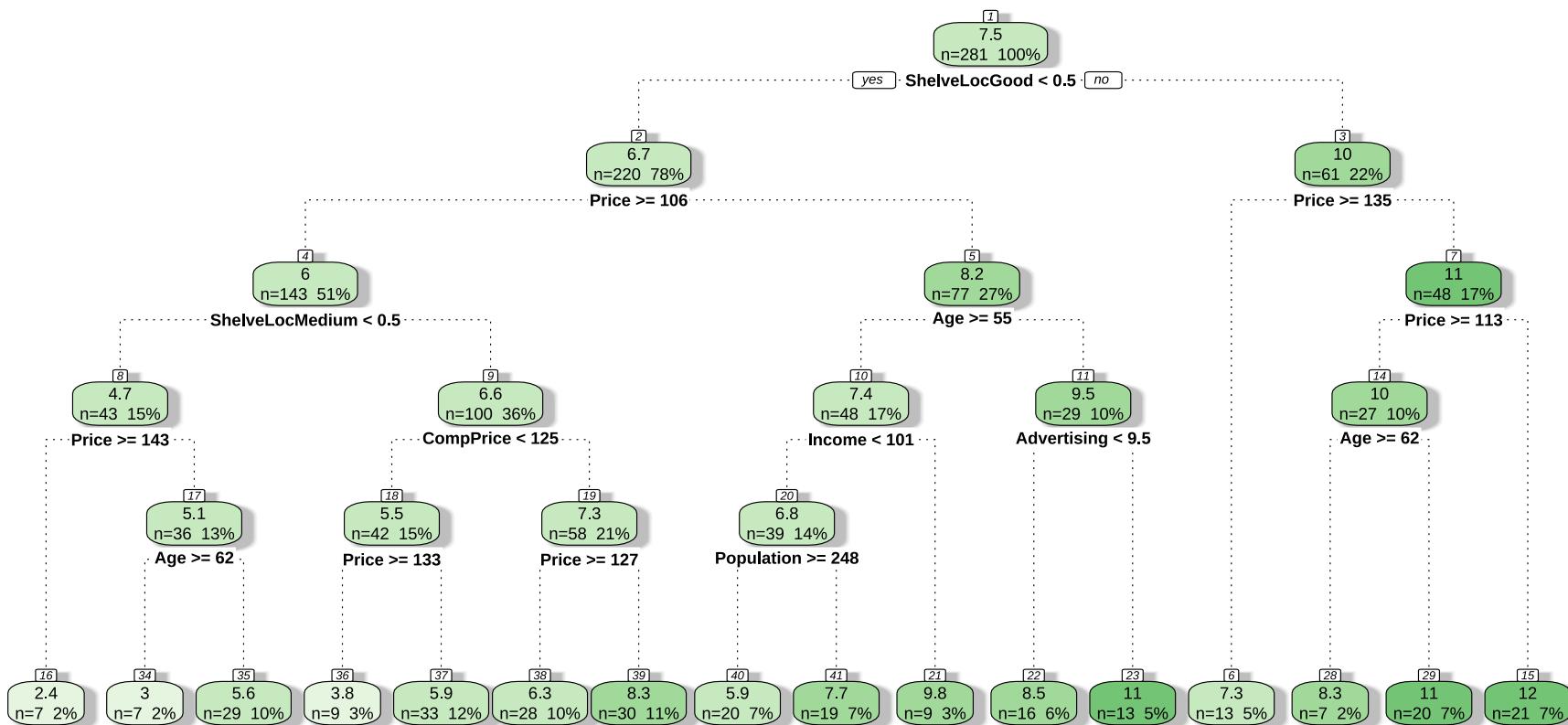
carseats.train <- training(split)
carseats.test <- testing(split)

tuneGrid <- expand.grid(cp = seq(0, 0.01, 0.0001))

mcv <- train(Sales ~., data = carseats.train, method = "rpart",
  trControl = trainControl("cv", number = 10), tuneGrid = tuneGrid)
```

Do you wanna build a... tree?

```
fancyRpartPlot(mcv$finalModel, caption="")
```



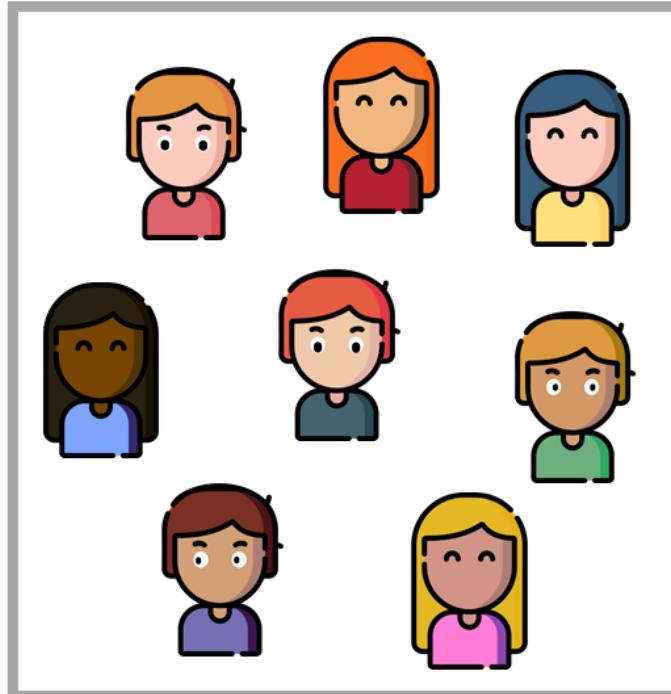
Seems a pretty complex tree... can
we improve it?

Bagging

Introduction to Bagging

- **Bagging (Bootstrap Aggregation)**: Meant to reduce variance.
- Remember **bootstrap sampling**?

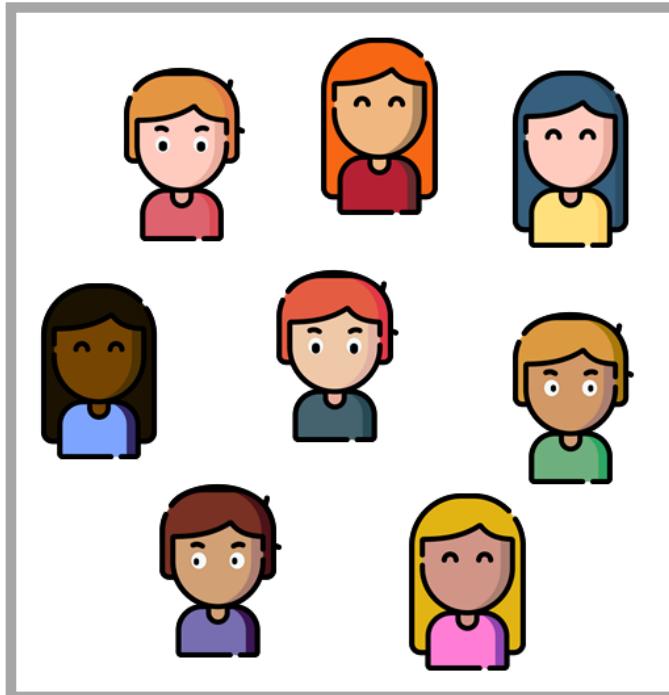
Sample



Introduction to Bagging

- **Bagging (Bootstrap Aggregation)**: Meant to reduce variance.
- Remember **bootstrap sampling**?

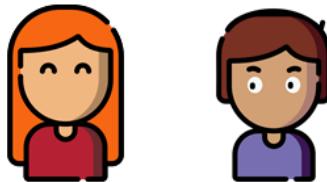
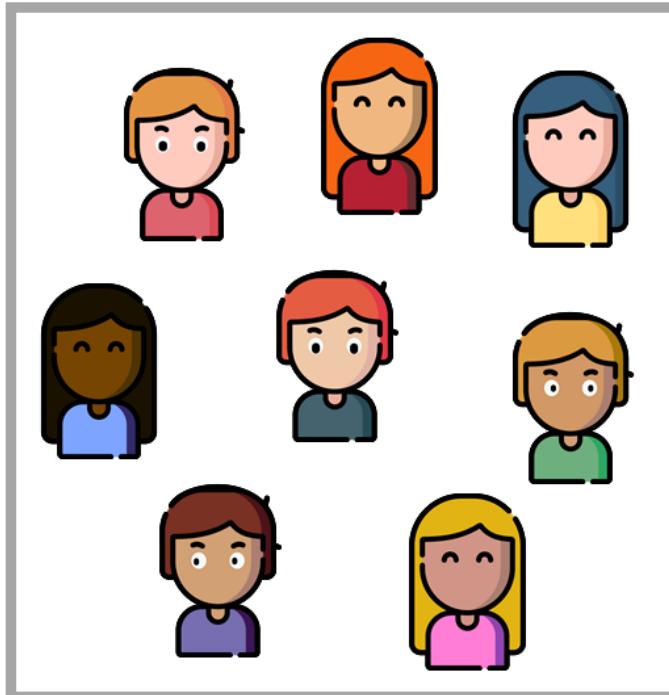
Sample



Introduction to Bagging

- **Bagging (Bootstrap Aggregation)**: Meant to reduce variance.
- Remember **bootstrap sampling**?

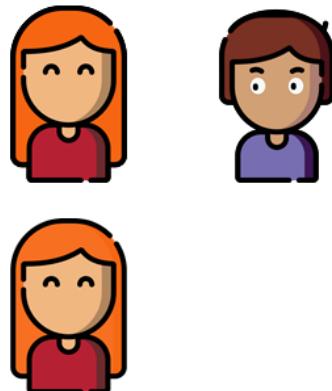
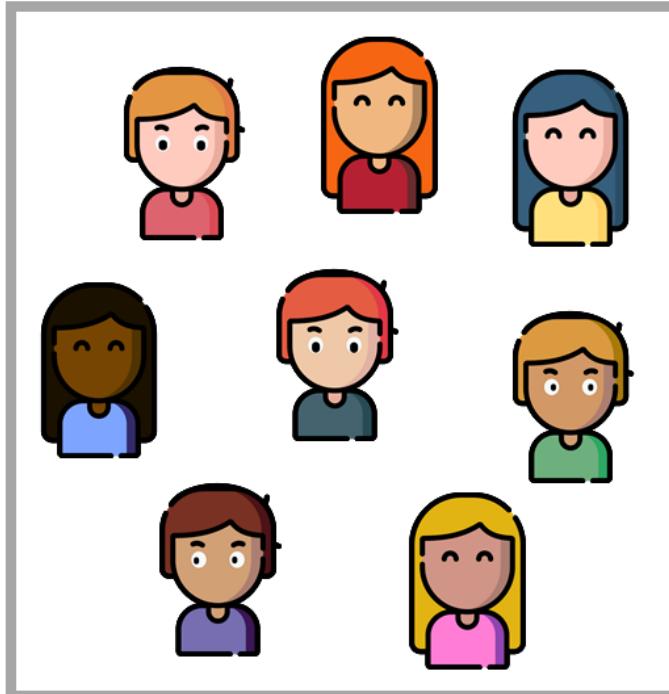
Sample



Introduction to Bagging

- **Bagging (Bootstrap Aggregation)**: Meant to reduce variance.
- Remember **bootstrap sampling**?

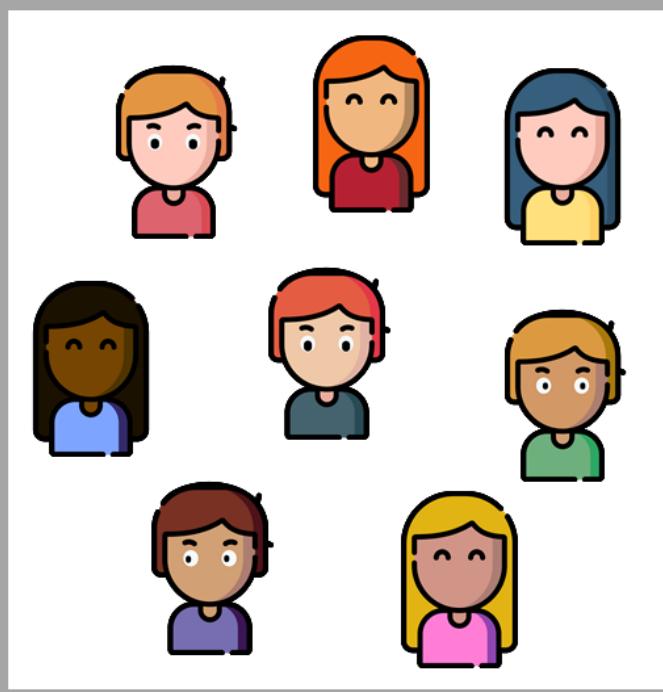
Sample



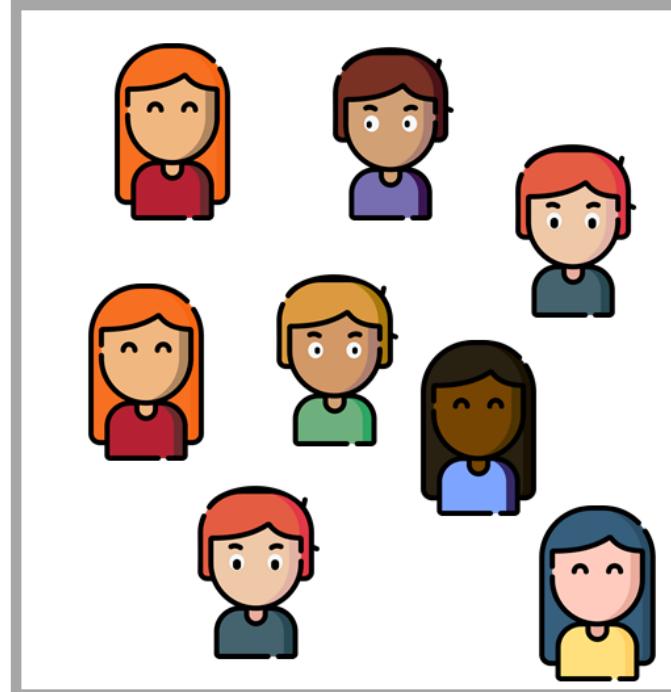
Introduction to Bagging

- **Bagging (Bootstrap Aggregation)**: Meant to reduce variance.
- Remember **bootstrap sampling**?

Sample

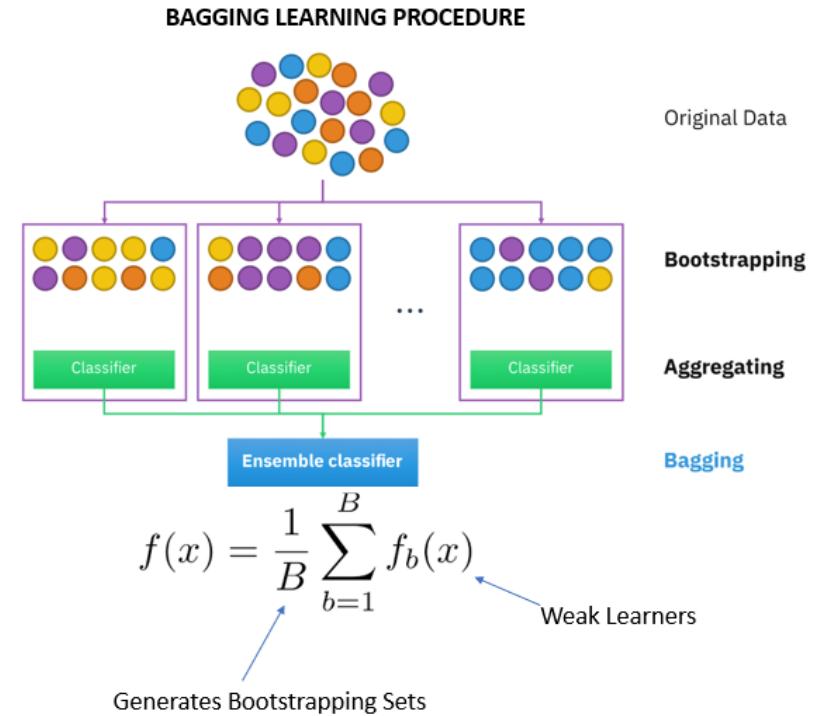


Bootstrapped Sample



Bagging and Decision Trees

1. Bootstrap your training sample B times
2. For each sample b , build a full-grown tree (no pruning).
3. Predict your outcomes!
 - a) Regression: Average the outcomes
 - b) Classification: Majority vote



Source: Singhal (2020)

But... how does this reduce variance?

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)$$

- If $Var(\hat{f}^b(x)) = \sigma^2 \quad \forall b$, then:

$$Var(\hat{f}_{bag}(x)) = Var\left(\frac{1}{B} \sum_{b=1}^B \hat{f}^b(x)\right) = \frac{B}{B^2} \sigma^2 = \frac{\sigma^2}{B}$$

Let's go back to our example

```
library(ipred)
library(rpart)

set.seed(100)

# train bagged model
b1 <- bagging(Sales ~ ., data = carseats.train,
                nbagg = 100, coob = TRUE,
                control = rpart.control(cp = 0))

b1
```

Let's go back to our example

```
library(ipred)
library(rpart)

set.seed(100)

# train bagged model
b1 <- bagging(Sales ~ ., data = carseats.train,
                nbagg = 100, coob = TRUE,
                control = rpart.control(cp = 0))

b1
```

Let's go back to our example

```
library(ipred)
library(rpart)

set.seed(100)

# train bagged model
b1 <- bagging(Sales ~ ., data = carseats.train,
                nbagg = 100, coob = TRUE,
                control = rpart.control(cp = 0))

b1
```

Why do we set $cp=0$?

Let's go back to our example

```
library(ipred)
library(rpart)

set.seed(100)

# train bagged model
b1 <- bagging(Sales ~ ., data = carseats.train,
                nbagg = 100, coob = TRUE,
                control = rpart.control(cp = 0))

b1
```



```
## 
## Bagging regression trees with 100 bootstrap replications
##
## Call: bagging.data.frame(formula = Sales ~ ., data = carseats.train,
##     nbagg = 100, coob = TRUE, control = rpart.control(cp = 0))
##
## Out-of-bag estimate of root mean squared error:  1.7034
```

What is OOB RMSE?

- **OOB = Out of Bag**: It represents the RMSE for units that *were not used for building certain trees*.
- Example:
 - Bag 1 has units $\{1, 2, 2\}$, Bag 2 units $\{1, 2, 3\}$, and Bag 3 $\{2, 3, 3\}$
 - OOB RMSE would be estimating using RMSE for **unit 1 in Bag 3** and **unit 3 in Bag 1**.

No need for cross-validation!

What is the main objective of
bagging?

How does it compare to the best pruned decision tree?

Let's see!

Best DT vs Bagging

```
set.seed(100)

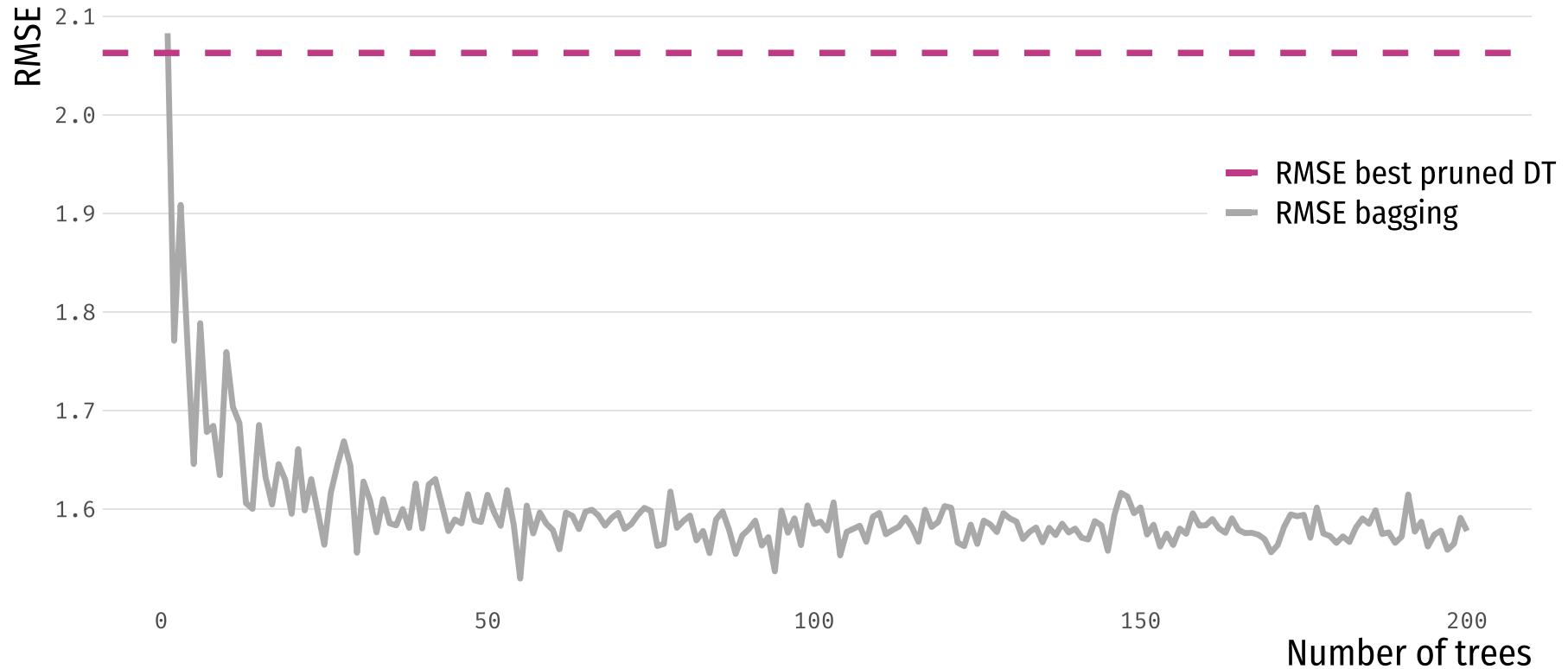
tuneGrid <- expand.grid(cp = seq(0.002, 0.004, 0.00001))

mcv <- train(Sales ~ ., data = carseats.train,
              method = "rpart", trControl = trainControl("cv", number = 10),
              tuneGrid = tuneGrid)

rmse(mcv, carseats.test)

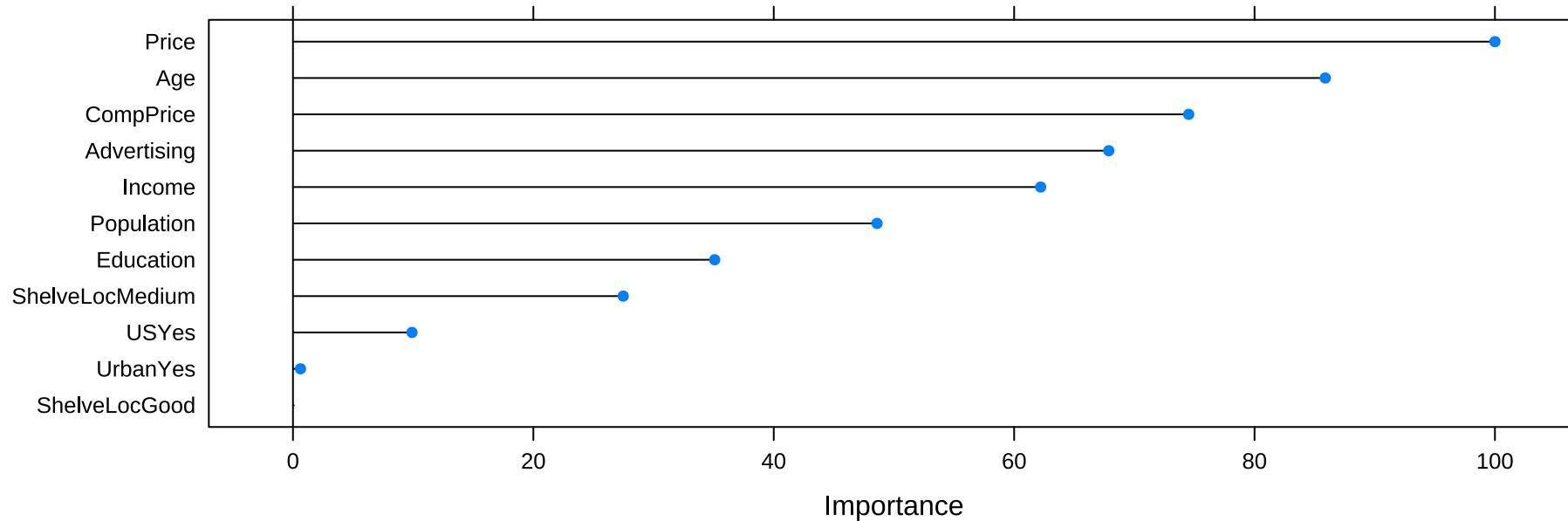
## [1] 2.082905
```

Best DT vs Bagging



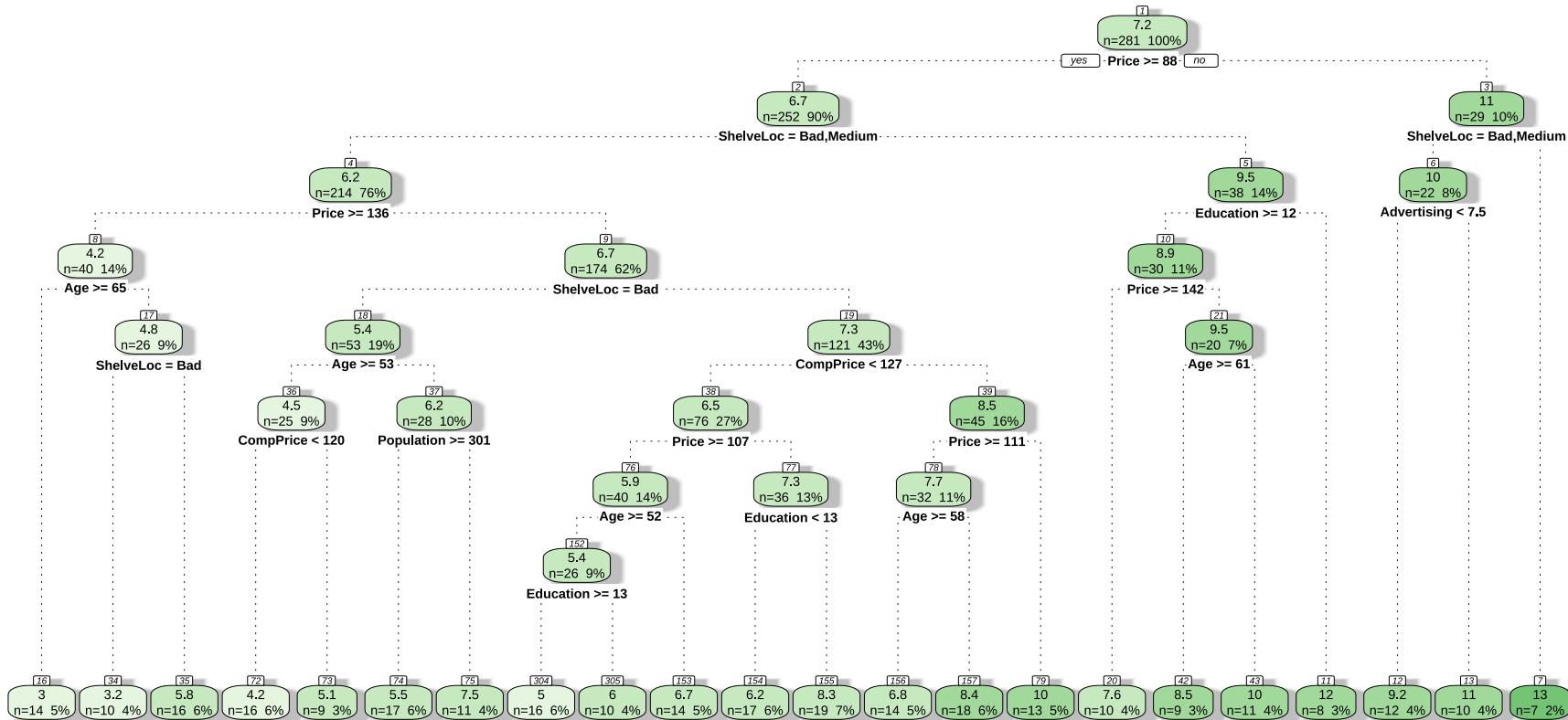
Interpretability?

```
bt <- train(Sales ~ ., data = carseats.train, method = "treebag", trControl = trainControl("cv",  
    nbagg = 100, control = rpart.control(cp = 0))  
  
plot(varImp(bt, scale = TRUE))
```



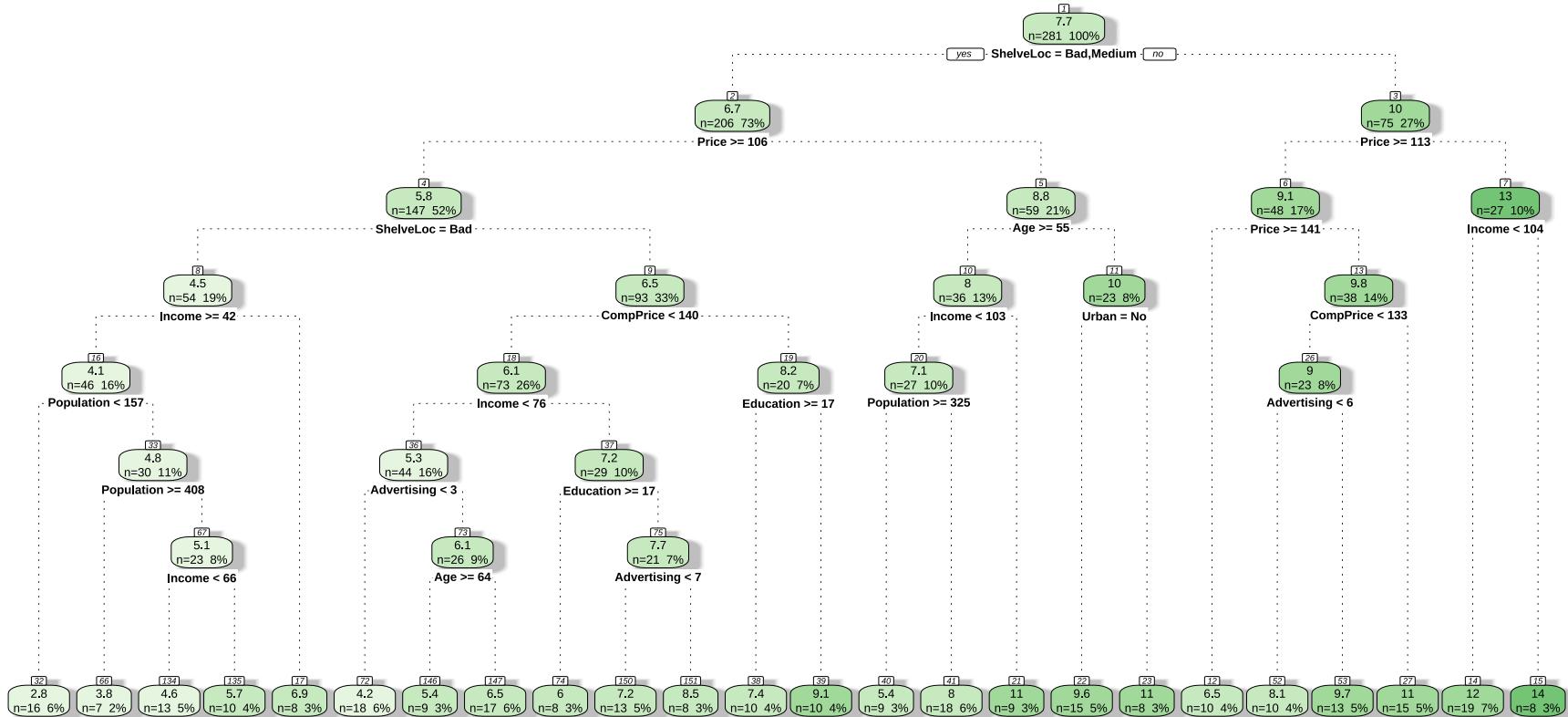
Disadvantages of bagged trees? (Tree 2)

```
fancyRpartPlot(b1$mtrees[[2]]$btree, caption="") #There are 100 trees!
```



Disadvantages of bagged trees? (Tree 3)

```
fancyRpartPlot(b1$mtrees[[3]]$btree, caption="") #There are 100 trees!
```



We can do better...

Random forests

Bringing trees together

- Random Forests uses both the concepts of **decision trees** and **bagging**, but also **de-correlates** the trees.

Bootstrap: Vary n dimension

De-correlation: Vary p dimension

- For each bagged tree, **choose m out of p regressors**.

Basic algorithm

1. Given a training data set
2. Select number of trees to build (n_{trees})
3. for $i = 1$ to n_{trees} do
4. | Generate a bootstrap sample of the original data
5. | Grow a regression/classification tree to the bootstrapped data
6. | for each split do
7. | | Select m_{try} variables at random from all p variables
8. | | Pick the best variable/split-point among the m_{try}
9. | | Split the node into two child nodes
10. | end
11. | Use typical tree model stopping criteria to determine when a
| tree is complete (but do not prune)
12. end
13. Output ensemble of trees

Source: Boehmke & Greenwell (2020)

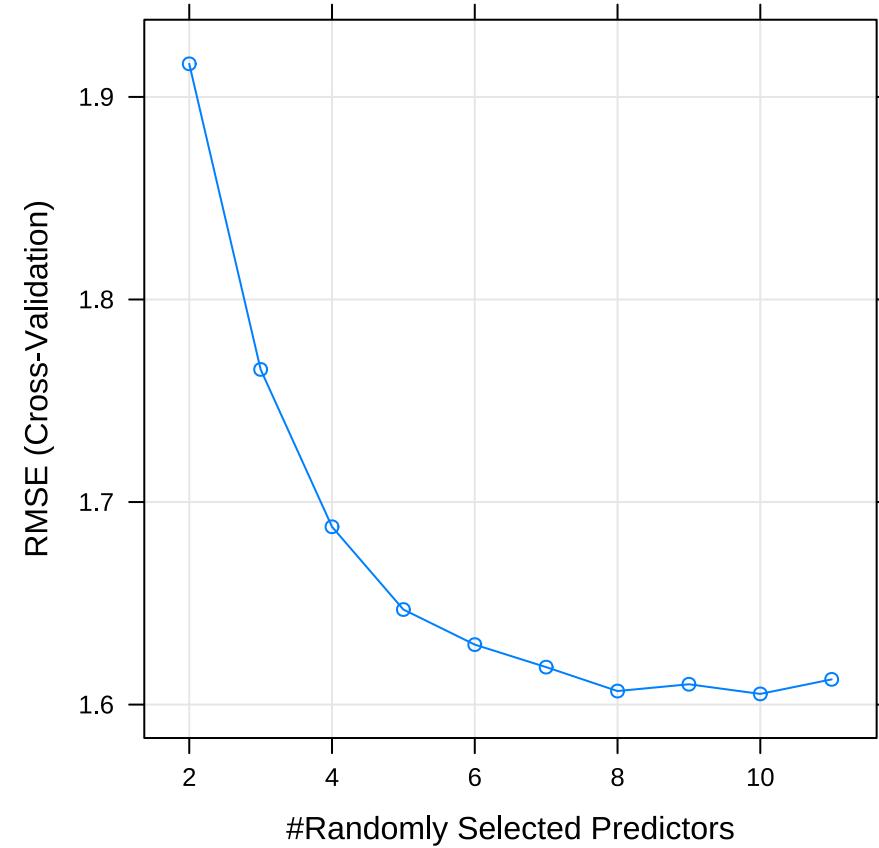
Back to our example!

```
set.seed(100)

tuneGrid <- expand.grid(
  mtry = 2:11,
  splitrule = "variance",
  min.node.size = 5
)

rfcv <- train(Sales ~ ., data = carseats.train
               method = "ranger",
               trControl = trainControl("cv", nrun = 5,
               importance = "permutation",
               tuneGrid = tuneGrid)

plot(rfcv)
```



Our dataset has 11 variables (1 outcome + 10 predictors)... why do the predictors go from 2 to 11?

We need to look at the `model.matrix`

```
head(model.matrix(Sales ~ ., data = carseats.train))
```

```
##   (Intercept) CompPrice Income Advertising Population Price ShelveLocGood
## 1           1      138      73          11       276     120            0
## 2           1      111      48          16       260      83            1
## 4           1      117     100          4       466      97            0
## 5           1      141      64          3       340     128            0
## 7           1      115     105          0       45     108            0
## 11          1      121      78          9       150     100            0
##   ShelveLocMedium Age Education UrbanYes USYes
## 1             0    42        17        1     1
## 2             0    65        10        1     1
## 4             1    55        14        1     1
## 5             0    38        13        1     0
## 7             1    71        15        1     0
## 11            0    26        10        0     1
```

Back to our example! (Runs faster: 30s vs 11s)

```
library(doParallel)
cl <- makePSOCKcluster(7)
registerDoParallel(cl)

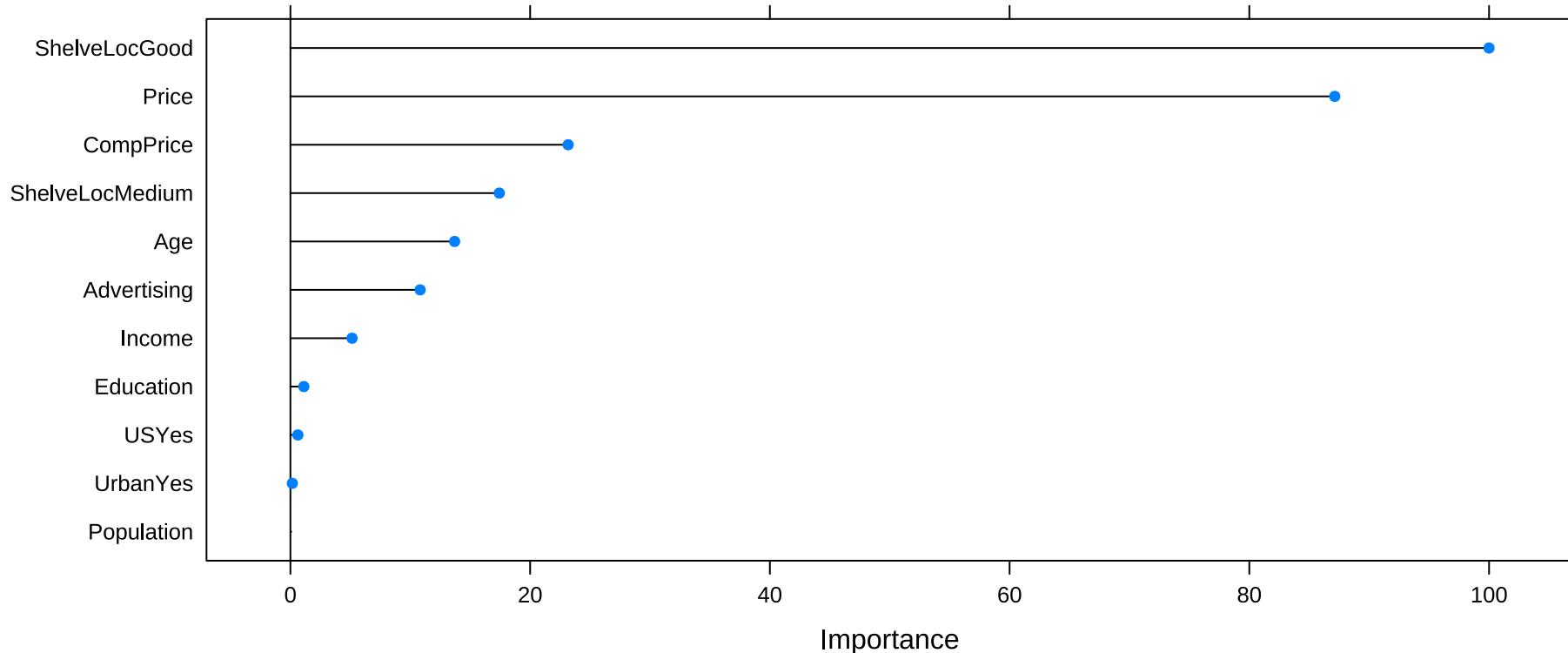
set.seed(100)

rfcv_fast <- train(Sales ~ ., data = carseats.train,
                     method = "ranger",
                     trControl = trainControl("cv", number = 10,
                                              allowParallel = TRUE),
                     tuneGrid = tuneGrid)

stopCluster(cl)
registerDoSEQ()
```

Covariance importance?

```
plot(varImp(rfcv))
```



In a Random Forest, a higher
number of trees will yield an...
**underfitted model? overfitted
model? doesn't affect?**

Let's compare our models:

```
# Pruned tree  
rmse(mcv, carseats.test)
```

```
## [1] 2.082905
```

```
# Bagged trees  
rmse(bt, carseats.test)
```

```
## [1] 1.604022
```

```
# Random Forest  
rmse(rfcv, carseats.test)
```

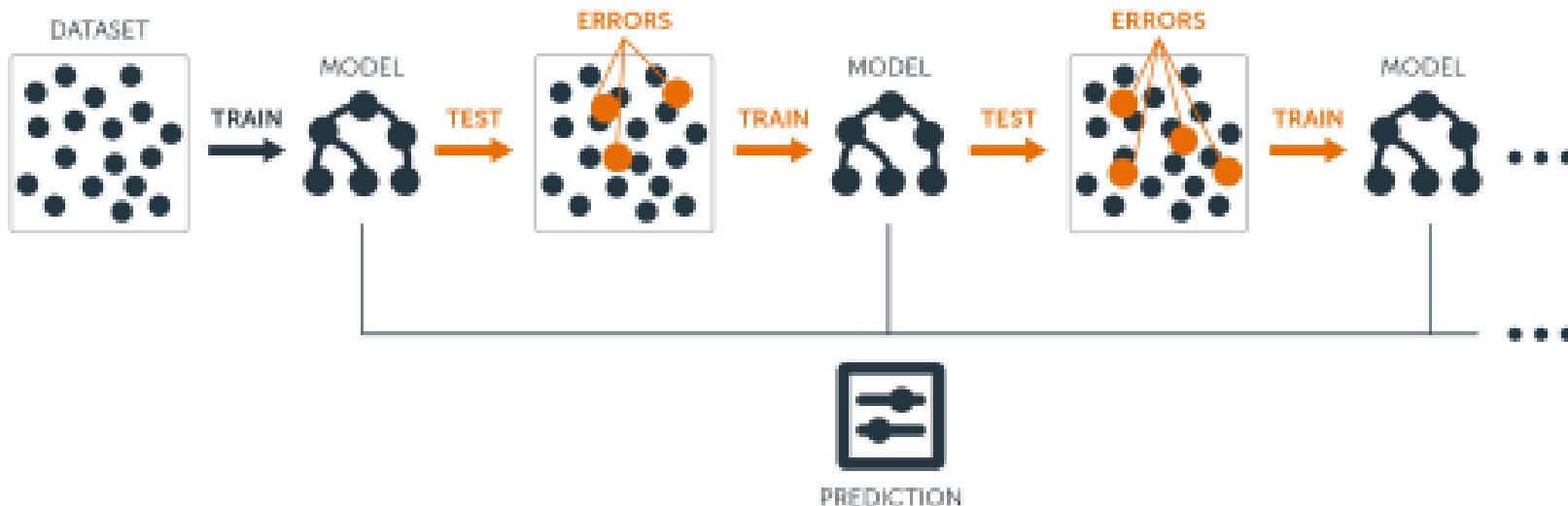
```
## [1] 1.484919
```

Can we do better than this?

Boosting!

What is boosting?

- Similar to bagging, but now **trees grow sequentially**.
- Slowly learning!
- More effective on models with **high bias** and **low variance**



Tuning parameters for boosting

- **Number of trees:** We need to select the B number of trees we will fit. We can get this through cross-validation.
- **Shrinkage parameter:** λ determines how fast the boosting will learn. Typical numbers range are 0.001 to 0.01. If your algorithm is learning too slow (low λ), you're going to need a lot of trees!
- **Number of splits:** Number of splits d controls the complexity of your trees. We usually work with low-complexity trees ($d=1$)

A tree with just a root and two leaves is called a stomp. Are these high or low-bias trees?

In boosting, a higher number of trees will yield an... underfitted model? overfitted model? does't affect?

Boosting in R

- We have seen **gradient boosting** so far.
- There are other types of boosting, like **adaptive boosting** (you saw it in the video!)
 - For classification problems

```
modelLookup("ada")
```

```
##   model parameter          label forReg forClass probModel
## 1  ada     iter      #Trees FALSE    TRUE    TRUE
## 2  ada   maxdepth Max Tree Depth FALSE    TRUE    TRUE
## 3  ada       nu Learning Rate FALSE    TRUE    TRUE
```

```
modelLookup("gbm")
```

```
##   model      parameter          label forReg forClass probModel
## 1  gbm        n.trees  # Boosting Iterations  TRUE    TRUE    TRUE
## 2  gbm interaction.depth  Max Tree Depth  TRUE    TRUE    TRUE
## 3  gbm        shrinkage  Shrinkage  TRUE    TRUE    TRUE
## 4  gbm    n.minobsinnode Min. Terminal Node Size  TRUE    TRUE    TRUE
```

Gradient Boosting in R

```
set.seed(100)

gbm <- train(Sales ~ ., data = carseats.train,
               method = "gbm",
               trControl = trainControl("cv", number = 10),
               tuneLength = 20)
```

Gradient Boosting in R

```
# Final Model information  
gbm$finalModel
```

```
## A gradient boosted model with gaussian loss function.  
## 350 iterations were performed.  
## There were 11 predictors of which 11 had non-zero influence.
```

```
# Best Tuning parameters?  
gbm$bestTune
```

```
## n.trees interaction.depth shrinkage n.minobsinnode  
## 7      350                  1      0.1          10
```

Let's do a comparison!

```
# Pruned tree  
rmse(mcv, carseats.test)
```

```
## [1] 2.082905
```

```
# Bagged trees  
rmse(bt, carseats.test)
```

```
## [1] 1.604022
```

```
# Random Forest  
rmse(rfcv, carseats.test)
```

```
## [1] 1.484919
```

```
# Gradient Boosting  
rmse(gbm, carseats.test)
```

```
## [1] 1.201098
```



What is the main objective of
boosting?

Main takeaway points

- There's a lot we can do to **improve our prediction models!**
- Decision trees by itself are not great...
 - ... but they are awesome for building other stuff like **random forests**.
- **Bagging** and **boosting** can be used with other learners, not only DT!

There are a lot of other methods out there and ways to combine them! (e.g. stacking)



Happy Final Class!

Thank you for all your participation

References

- Boehmke, B. & B. Greenwell. (2020). "Hands-on Machine Learning with R"
- James, G. et al. (2021). "Introduction to Statistical Learning with Applications in R". *Springer. Chapter 8.*
- Singhal, G. (2020). "Ensemble methods in Machine Learning: Bagging vs. Boosting"