# Decision Trees, Random Forests, and Gradient Boosting

Before starting this assignment, make sure to go through the R lab in chapter 8 of Introduction to Statistical Learning. Random forests and gradient boosting are often the default choice when building predictive models; it is important that you understand how they work and how to fit them.

## Problem 1 (ISLR 8.8)

This problem will use the Carseats data in the ISLR package. We will predict sales using the other variables.

### Problem 1

The code below splits the data into a training and test set

```
set.seed(1988)
N = nrow(Carseats)
train_prop = 0.8
train_index = sample(1:N, size = floor(N * train_prop), replace = FALSE)
train_dat = Carseats[train_index, ]
test_dat = Carseats[-train_index, ]
```

### Problem 2

Fit a regression tree to the dataset. Report the RMSE on your test set.

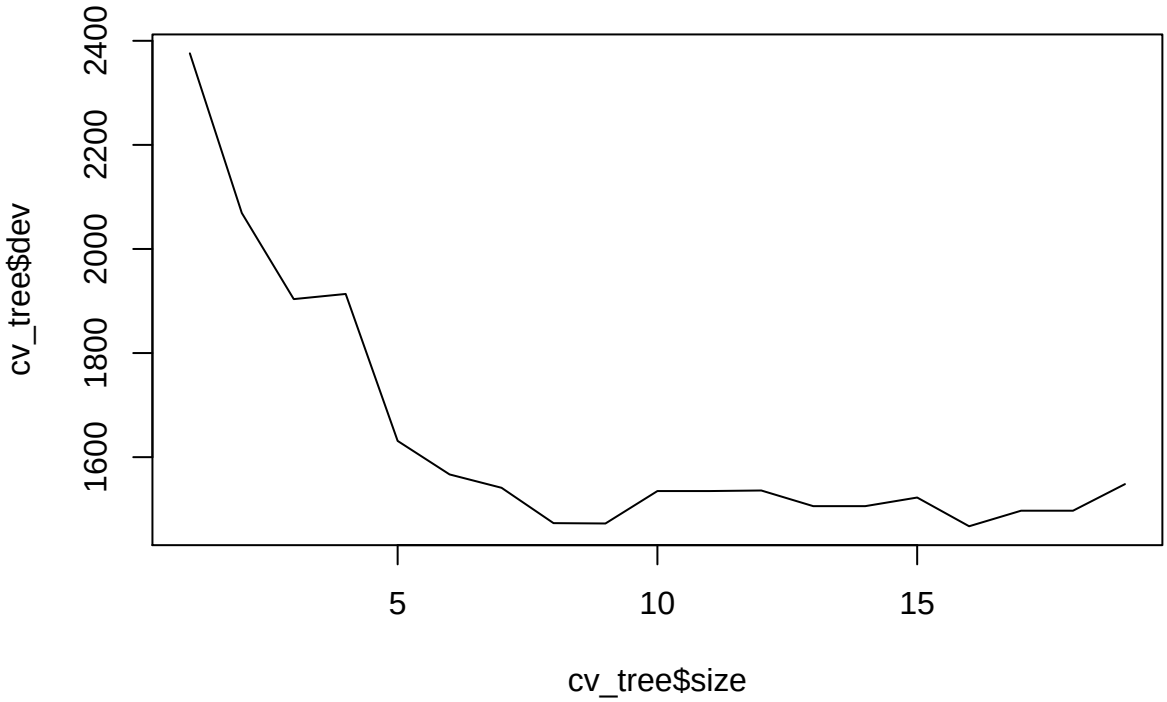
```
full_tree = tree(Sales ~ ., data = train_dat)
test_preds = predict(full_tree, test_dat)
test_RMSE = sqrt(mean((test_dat$Sales - test_preds)^2))
print(test_RMSE)
```

```
## [1] 2.170596
```

### Problem 3

Prune the tree using cross-validation. Plot the deviance by tree size. How many folds does the CV function use by default?

```
cv_tree = cv.tree(full_tree)
plot(cv_tree$size, cv_tree$dev, type = "l")
```




The function using 10 folds by default

## Problem 4

The plot should show the deviance flattening out around a tree size of 8 and attaining its minimum at a size of 16. Prune the tree to a size of 8 and 16 and report test RMSE.

```
min_dev_num = 8
pruned_tree = prune.tree(full_tree, best = min_dev_num)
test_preds = predict(pruned_tree, test_dat)
test_RMSE = sqrt(mean((test_dat$Sales - test_preds)^2))
print(test_RMSE)
```

```
## [1] 2.137022
```

```
min_dev_num = 16
pruned_tree = prune.tree(full_tree, best = min_dev_num)
test_preds = predict(pruned_tree, test_dat)
test_RMSE = sqrt(mean((test_dat$Sales - test_preds)^2))
print(test_RMSE)
```

```
## [1] 2.082202
```

## Problem 5

Which pruning size would you prefer? Justify your answer.

(Opinion) Would prefer a size of 8. smaller tree means more interpretable model, test error is not meaninfully different

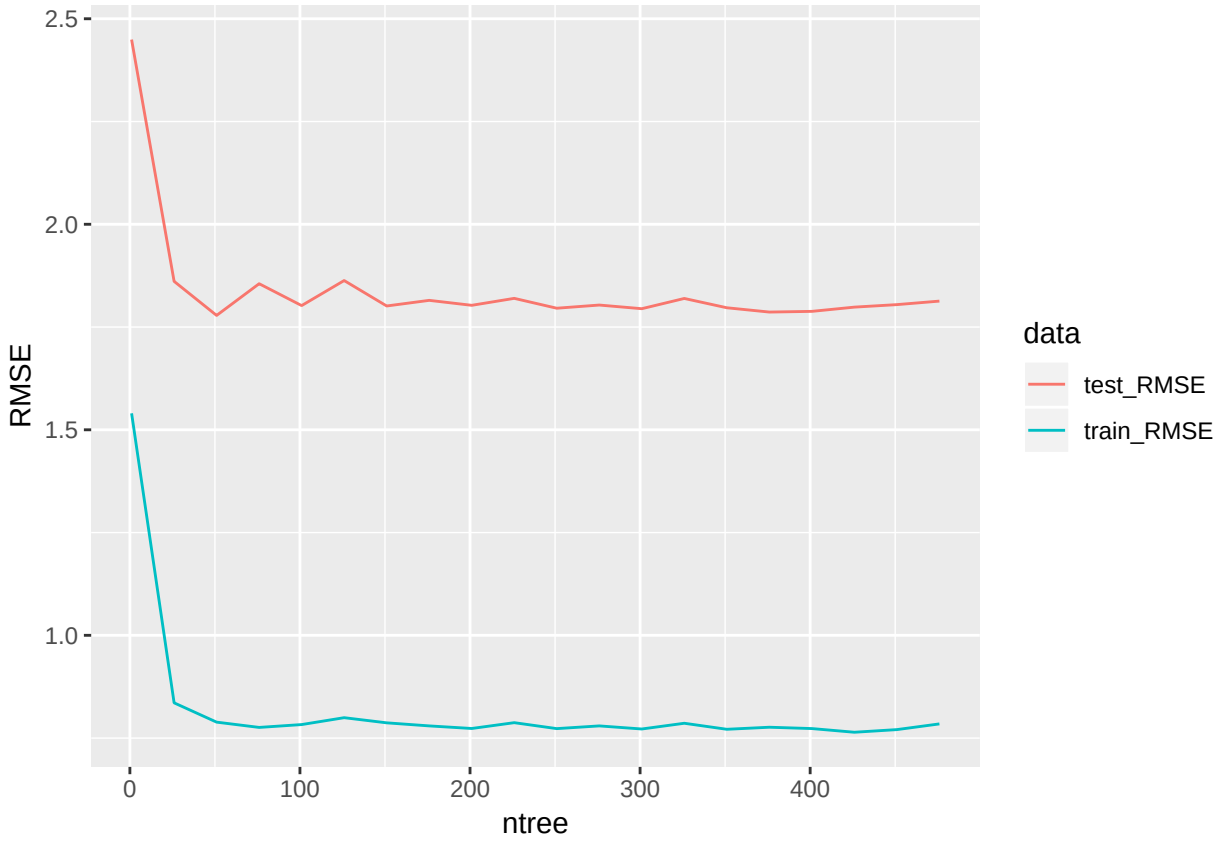
## Problem 6

Fit a random forest to the training data with the ntree range given below. Make a plot of training and test RMSE by tree size. Report the minimum of test RMSEs. Is the improvement meaningful over decision trees?

```
ntree = seq(1, 500, by = 25)
```

```
test_RMSE = rep(NA, length(ntree))
train_RMSE = rep(NA, length(ntree))
for(i in 1:length(ntree))
{
  this_rf = randomForest(Sales ~ ., data = train_dat, ntree = ntree[i])
  test_preds = predict(this_rf, test_dat)
  test_RMSE[i] = sqrt(mean((test_dat$Sales - test_preds)^2))
  train_preds = predict(this_rf, train_dat)
  train_RMSE[i] = sqrt(mean((train_dat$Sales - train_preds)^2))
}

tibble(ntree, train_RMSE, test_RMSE) %>%
  gather(key = data, value = RMSE, -ntree) %>%
  ggplot(aes(x = ntree, y = RMSE, color = data)) +
  geom_line()
```




```
print(min(test_RMSE))
```

```
## [1] 1.778305
```

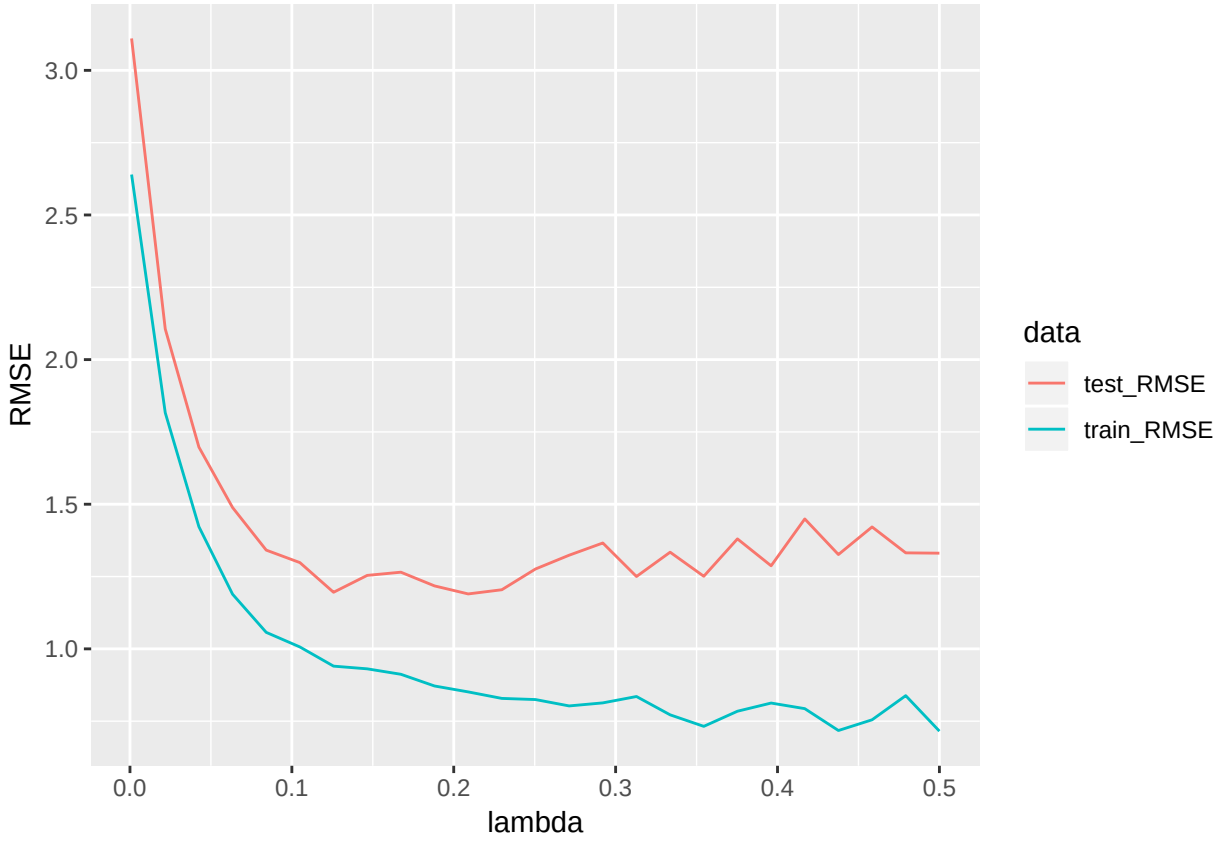
## Problem 7

Perform gradient boosting for the range of learning rates given below. Fix the number of trees at 100 and interaction depth at 2. Report the minimum of test RMSE.

```
lambda = seq(0.001, 0.5, length = 25)
```

```
test_RMSE = rep(NA, length(lambda))
train_RMSE = rep(NA, length(lambda))
for(i in 1:length(lambda))
{
  this_gb = gbm(Sales ~ ., data = train_dat, distribution = "gaussian",
                n.trees = 100, interaction.depth = 2, shrinkage = lambda[i])
  test_preds = predict(this_gb, test_dat, n.trees = 100)
  test_RMSE[i] = sqrt(mean((test_dat$Sales - test_preds)^2))
  train_preds = predict(this_gb, train_dat, n.trees = 100)
  train_RMSE[i] = sqrt(mean((train_dat$Sales - train_preds)^2))
}

tibble(lambda, train_RMSE, test_RMSE) %>%
  gather(key = data, value = RMSE, -lambda) %>%
  ggplot(aes(x = lambda, y = RMSE, color = data)) +
  geom_line()
```




```
min(test_RMSE) %>% print()
```

```
## [1] 1.189887
```

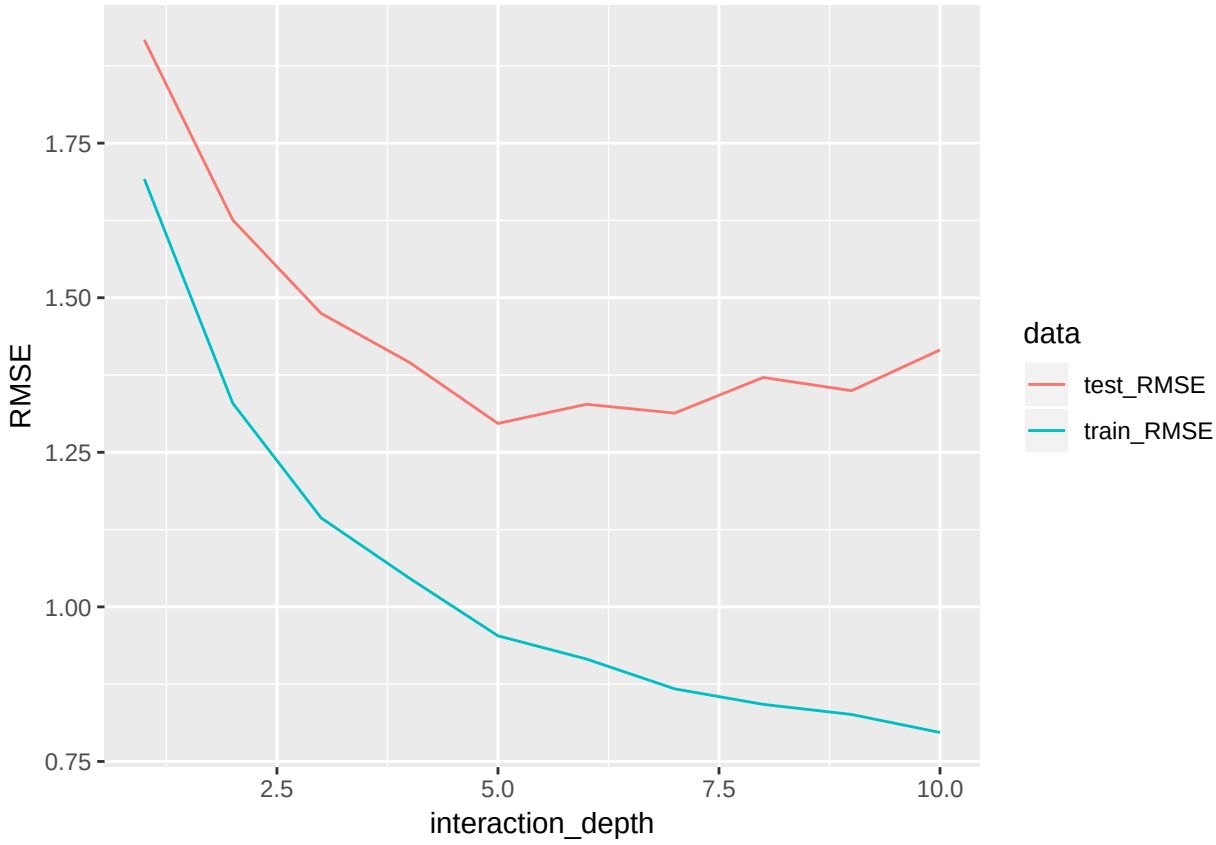
## Problem 8

Do the same analysis above, but with learning rate fixed at 0.01, ntree fixed at 500, and varying interaction depth in the range below.

```
interaction_depth = 1:10
```

```
test_RMSE = rep(NA, length(interaction_depth))
train_RMSE = rep(NA, length(interaction_depth))
for(i in 1:length(interaction_depth))
{
  this_gb = gbm(Sales ~ ., data = train_dat, distribution = "gaussian",
                n.trees = 100, interaction.depth = interaction_depth[i],
                shrinkage = 0.05)
  test_preds = predict(this_gb, test_dat, n.trees = 100)
  test_RMSE[i] = sqrt(mean((test_dat$Sales - test_preds)^2))
  train_preds = predict(this_gb, train_dat, n.trees = 100)
  train_RMSE[i] = sqrt(mean((train_dat$Sales - train_preds)^2))
}

tibble(interaction_depth, train_RMSE, test_RMSE) %>%
  gather(key = data, value = RMSE, -interaction_depth) %>%
  ggplot(aes(x = interaction_depth, y = RMSE, color = data)) +
  geom_line()
```




```
print(min(test_RMSE))
```

```
## [1] 1.296619
```

## Problem 9

Look at the gbm documentation. List three parameters in the model that can be tuned using cross-validation. Also state whether increasing them (holding all else constant) would make the model more or less flexible.

- n.trees (more flexible)
- shrinkage (more flexible)
- bag.fraction (more flexible)
- interaction.depth (more flexible)

## Problem 10

Re-run the analysis with a different seed (you don't have to present the results). Do you get different test RMSEs? What can we change in our process above to stabilize the estimate of test error?

- K-fold crossvalidation
- Monte-Carlo crossvalidat
- Do more iterations of what we did above and average them out.