Lab 8: Tree-based Models

We will use the `Carseats` data set in the `ISLR` package to predict `high_sales` for carseats at 400 different stores.

```r
library(ISLR) ## data package
library(tidyverse) ## data manipulation
library(knitr) ## tables

## reproducible
set.seed(445)

## data
str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 .
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

0.1 Data Preparation

1. Make a copy of the `Carseats` data frame called `df`.
2. Create a variable called `high_sales` in `df` that takes the value "high" if `Sales` > 8 and "low" otherwise.
3. Convert your `high_sales` column to be a factor.
4. Remove the `Sales` column from `df`.

0.2 Decision Trees

The `tree` package is used to contruct classification and regression trees. We will construct a classification tree to predict

```
library(tree) ## tree package
```

1. Using the `tree` function, fit a large classification tree to predict `high_sales` using every variable in `df`. [**Hint:** The syntax is very similar to `lm`]

2. Inspect your tree using `summary`. How many terminal nodes do you have? Whatt is the training error rate?

   [**Note:** The "deviance" reported is given by $-2\sum_m \sum_k n_{mk} \log \hat{p}_{mk}$ where n_{mk} is the number of observations in the mth terminal node that belongs to the kth class. A small deviance indicates a good fit to the training data.]

3. Use the `plot` function to visualize your tree. What is the most important indicator of high sales?

   [**Hint:** Adding the following line after you plot the tree will add labels.
   `text(tree.fit, pretty = 0)`]

4. Split your observations into a training and a test set with 200 records each. Estimate the test error rate of your tree. [**Hint:** using `type = "class"` in your `predict` function will get you the actual class predictions.]

5. Produce a confusion matrix for your test set.

6. Use the `cv.tree` function to perform cross-validation to determine the optimal level of tree complexity. Using `FUN = prune.misclass` indicates that we want to use the classification error rate (instead of deviance) to guide the CV and pruning process. Which α (corresponds to `k` in the output) should we choose?

7. Use the function `prune.misclass` to prune your tree to the chosen complexity.

8. Repeat 4-5 using your pruned tree. Which performs better?

0.3 Bagging & Random Forests

We will use the `randomForest` package to perform bagging and random forests. Recall that bagging is simply a special case of random forests with $m = p$.

```r
library(randomForest) # random forests & bagging
```

1. Perform bagging on your training `df` to predict `high_sales`. Specify `importance = TRUE` to also obtain information on the importance of each predictor.

2. Make a plot of the importance values for each predictor. What is the predictor with the highes importance?

3. Estimate the test error rate using your bagged tree model.

4. Repeat 1-3 using a random forest with $m = \sqrt{p}$.

5. Compare the OOB confusion matrix to your test confusion matrix. [**Hint:** The `confusion` element of the model output is OOB.]

0.4 Boosting

To perform boosting we will use the `gbm` function in the `gbm` package.

```r
library(gbm) ## boosting package
```

1. Fit a boosted tree ensemble to your training `df` predicting `high_sales` with $B = 5,000$ trees, shrinkage parameter of $\lambda = 0.1$, and an interaction depth of $d = 2$. We sure to include `distribution = "bernoulli"` to indicate a classification problem.

2. Estimate the test error rate using your boosted tree model and compare to all previously fit models.