Homework 8

Drew Dahlquist

4/21/2022

1.

```
bike = read.csv("bike.csv")
bike = bike[,-1] # remove the index column

# some variables are transformed to factors.
bike$season = as.factor(bike$season)
bike$workingday = as.factor(bike$workingday)
bike$weathersit = as.factor(bike$weathersit)

# check the data structure
str(bike)
```

```
## 'data.frame':    731 obs. of  9 variables:
##  $ dteday    : chr  "2011-01-01" "2011-01-02" "2011-01-03" "2011-01-04" ...
##  $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...
##  $ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...
##  $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
##  $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
##  $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
##  $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
##  $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

```
set.seed(1)
train = sample(1:nrow(bike), 0.7*nrow(bike))
bike.test = bike[-train, ]
```

(a)

The tree has 9 terminal nodes, and a residual mean deviance of 1496000.

```
library(tree)

tree.fit = tree(cnt ~ . -dteday, bike, subset=train)

summary(tree.fit)
```

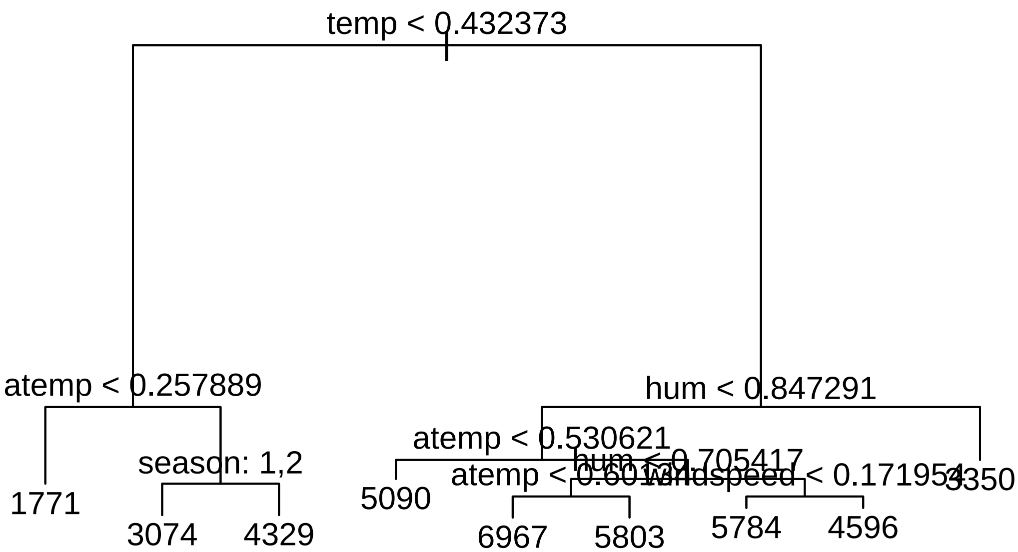
```
##
## Regression tree:
## tree(formula = cnt ~ . - dteday, data = bike, subset = train)
## Variables actually used in tree construction:
## [1] "temp"      "atemp"     "season"    "hum"       "windspeed"
## Number of terminal nodes:  9
## Residual mean deviance:  1496000 = 751100000 / 502
## Distribution of residuals:
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -3328.00   -847.90    -49.31      0.00    979.40   3386.00
```

(b)

The `temp` is the most informative predictor in this model, followed by `atemp` if `temp` < 0.43 or `hum` if `temp` $>$ 0.43, and so on.

```
plot(tree.fit)
text(tree.fit, pretty=0)
```



(c)

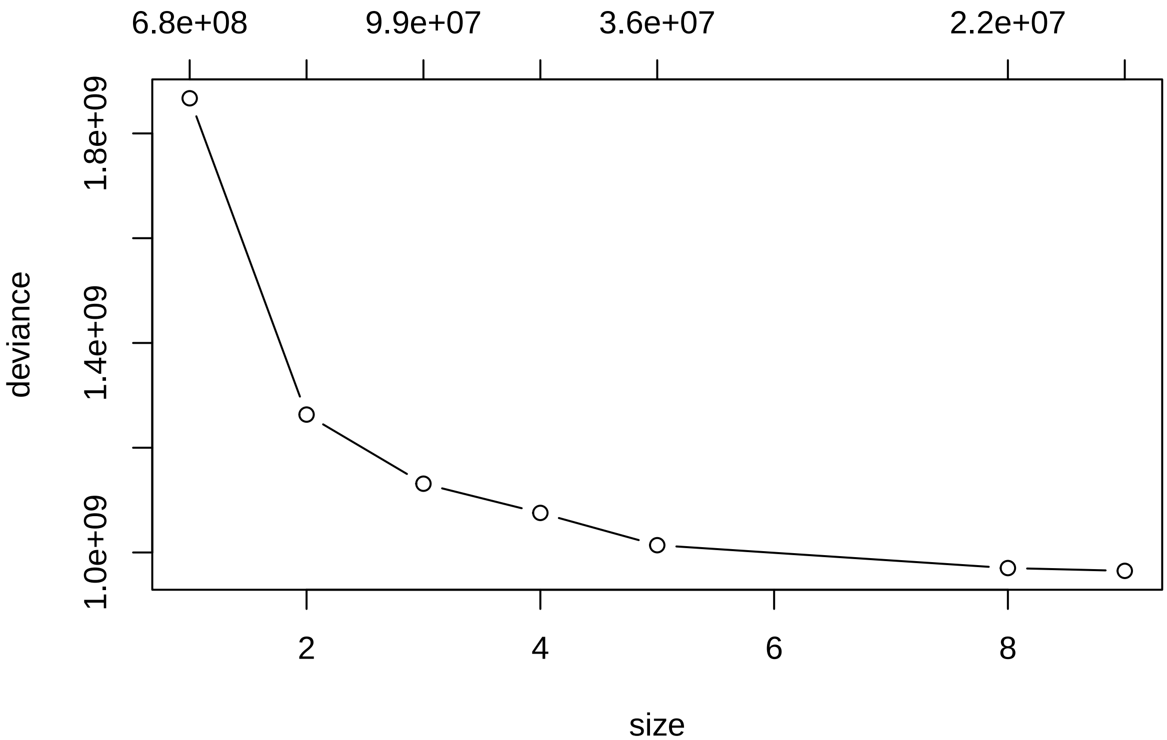
To predict a new value, we start at the root node and evaluate the condition, then go down the tree left or right. We recursively repeat this process until we reach a leaf node, which corresponds to our prediction.

`temp`: $0.426667 < 0.432373$ `atemp`: $0.426737 > 0.257889$ `season`: 3 != 1,2 Predict: 4329

(d)

In this case, pruning the tree isn't necessary.

```
plot(cv.tree(tree.fit), type="b")
```



(e)

```r
yhat = predict(tree.fit, newdata = bike.test)
mean((yhat-bike.test$"cnt")^2)
```

```
## [1] 1779380
```

The test MSE is 1.7793798×10^6.

(f)

```r
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
set.seed(1)

bag.bike=randomForest(cnt~.-dteday,data=bike,subset=train,importance=TRUE)
bag.bike
```

```
##
## Call:
##  randomForest(formula = cnt ~ . - dteday, data = bike, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1504934
##                     % Var explained: 58.63
```

The training MSE is 1504934.

(g)

```
importance(bag.bike)
```

```
##              %IncMSE IncNodePurity
## season     24.994375     231534224
## workingday  4.080443      29370155
## weathersit 16.471384      73157169
## temp       31.071917     507798871
## atemp      28.828857     463419751
## hum        31.874122     235082486
## windspeed  13.501590     183488540
```

(h)

```
yhat.bag = predict(bag.bike,newdata=bike.test)
mean((yhat.bag-bike.test$cnt)^2)
```

```
## [1] 1437450
```

(i)

```
set.seed(1)

rf.bike=randomForest(cnt~.-dteday,data=bike,subset=train,mtry=2,importance=TRUE)
yhat.rf = predict(rf.bike,newdata=bike.test)
mean((yhat.rf-bike.test$cnt)^2)
```

```
## [1] 1437450
```

(j)

For the boosting 4 models, each with a differing number of tree's used, the training MSE decreases initially then begins to increase again. This is telling that the number of trees used in a boosted model controls the bias-variance trade off. Namely, a small value for n.trees has low bias, high variance, whereas a large value for n.trees has a high bias, low variance.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(1)

# 4 boosted models with differing number of trees
boost.bike.1 = gbm(cnt~.-dteday,data=bike[train,],distribution="gaussian",interaction.depth=1,n.trees=5
boost.bike.2 = gbm(cnt~.-dteday,data=bike[train,],distribution="gaussian",interaction.depth=1,n.trees=1
boost.bike.3 = gbm(cnt~.-dteday,data=bike[train,],distribution="gaussian",interaction.depth=1,n.trees=5
boost.bike.4 = gbm(cnt~.-dteday,data=bike[train,],distribution="gaussian",interaction.depth=1,n.trees=1

yhat.boost.1 = predict(boost.bike.1,newdata=bike.test,n.trees=50)
yhat.boost.2 = predict(boost.bike.2,newdata=bike.test,n.trees=100)
yhat.boost.3 = predict(boost.bike.3,newdata=bike.test,n.trees=500)
yhat.boost.4 = predict(boost.bike.4,newdata=bike.test,n.trees=1000)

mean((yhat.boost.1-bike.test$cnt)^2)
```

```
## [1] 1620999
```

```
mean((yhat.boost.2-bike.test$cnt)^2)
```

```
## [1] 1465312
```

```r
mean((yhat.boost.3-bike.test$cnt)^2)
```

```
## [1] 1426833
```

```r
mean((yhat.boost.4-bike.test$cnt)^2)
```

```
## [1] 1485255
```