HW1

Shinjo Hiroki

2023-12-22

1.

(a)

For $q = 1$,

$$
\begin{aligned}
E[\hat{y}^{(1)}_{new} - y_{new}] &= E[-y_{new}] = -\mu \\
Var[\hat{y}^{(1)}_{new} - y_{new}] &= Var[y_{new}] = 1 \\
MSE^{(1)} &= Var[\hat{y}^{(1)}_{new} - y_{new}] + (E[\hat{y}^{(1)}_{new} - y_{new}])^2 \\
&= 1 + \mu^2
\end{aligned}
$$

For $q = 2$,

$$
\begin{aligned}
E[\hat{y}^{(2)}_{new} - y_{new}] &= E[\hat{\mu} - y_{new}] = \frac{1}{n}\sum_{i=1}^{n} E[y_i] - \mu = 0 \\
Var[\hat{y}^{(2)}_{new} - y_{new}] &= Var[\hat{\mu} - y_{new}] = Var(\hat{\mu}) - 2Cov(\hat{\mu}, y_{new}) + Var(y_{new}) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} Var(y_i) - \frac{2}{n}\sum_{i=1}^{n} Cov(y_i, y_{new}) + Var(y_{new}) = \frac{1}{n} + 1 \\
MSE^{(2)} &= Var[\hat{y}^{(2)}_{new} - y_{new}] + (E[\hat{y}^{(2)}_{new} - y_{new}])^2 \\
&= 1 + \frac{1}{n}
\end{aligned}
$$

Bias-Variance tradeoff is expressed as

$$
\begin{aligned}
MSE^{(1)} &= Var(\hat{y}^{(1)}_{new}) - 2Cov(\hat{y}^{(1)}_{new}, y_{new}) + Var(y_{new}) + (E[y_{new} - \hat{y}^{(1)}_{new}])^2 \\
&= Var(\hat{y}^{(1)}_{new}) + (E[y_{new} - \hat{y}^{(1)}_{new}])^2 + Var(y_{new}) \\
&= 0 + \mu^2 + 1 \\
MSE^{(2)} &= Var(\hat{y}^{(2)}_{new}) + (E[y_{new} - \hat{y}^{(2)}_{new}])^2 + Var(y_{new}) \\
&= \frac{1}{n} + 0 + 1
\end{aligned}
$$

The 1st term represents variance,the 2nd represents bias and 3rd represents irreducible error.

The more complex model corresponds to $q = 2$, obtaining an unbiased estimator, but with a variance greater than 0. Conversely, the less complex model for $q = 1$, has a variance of 0, but the bias is $\mu^2 > 0$ if $\mu \neq 0$. This trend captures the essence of bias-variance tradeoff, where an increase in model complexity leads to decrease in bias but an increase in variance.

(b)

If

$$MSE^{(1)} < MSE^{(2)} \iff \mu^2 < \frac{1}{n}$$

holds, forecast 1 is better.

(c)

$$\begin{aligned}
CV^{(1)} &\to_p E[y_i^2] = Var[y_i] + (E[y_i])^2 = 1 + \mu^2 \\
CV^{(2)} &\to_p lim_{n\to\infty} E[(y_i - \hat{\mu}_{(-i)})^2]
\end{aligned}$$

Note

$$\begin{aligned}
E[(y_i - \hat{\mu}_{(-i)})^2] &= E\left[y_i^2\right] - 2E\left[y_i\hat{\mu}_{(-i)}\right] + E\left[\hat{\mu}_{(-i)}^2\right] \\
&= E\left[y_i^2\right] - \frac{2}{n-1}E\left[y_i\sum_{j\neq i} y_j\right] + \frac{1}{(n-1)^2}E\left[\sum_{j\neq i} y_j^2 + \sum_{j\neq i}\sum_{k\neq j\wedge\neq i} y_j y_k\right] \\
&= \left(1 + \frac{1}{n-1}\right)E\left[y_i^2\right] - \frac{2(n-1)}{n-1}\mu^2 + \frac{(n-1)(n-2)}{(n-1)^2}\mu^2 \\
&= \left(1 + \frac{1}{n-1}\right)(1+\mu^2) - \frac{n}{n-1}\mu^2 \\
&= \left(1 + \frac{1}{n-1}\right)(1+\mu^2) - \frac{1}{1-\frac{1}{n}}\mu^2
\end{aligned}$$

Therefore,

$$lim_{n\to\infty}\left[\left(1 + \frac{1}{n-1}\right)(1+\mu^2) - \frac{1}{1-\frac{1}{n}}\mu^2\right] = 1 + \mu^2 - \mu^2 = 1$$

Hence,

$$1 < 1 + \mu^2 \iff CV^{(2)} < CV^{(1)}$$

holds. Then leave-one-out cross-validation chooses the correct forecaset, which is $q = 2$.

2.

```r
pacman::p_load(ISLR,
               tree,
               randomForest,
               MASS,
               gbm)
```

(a)

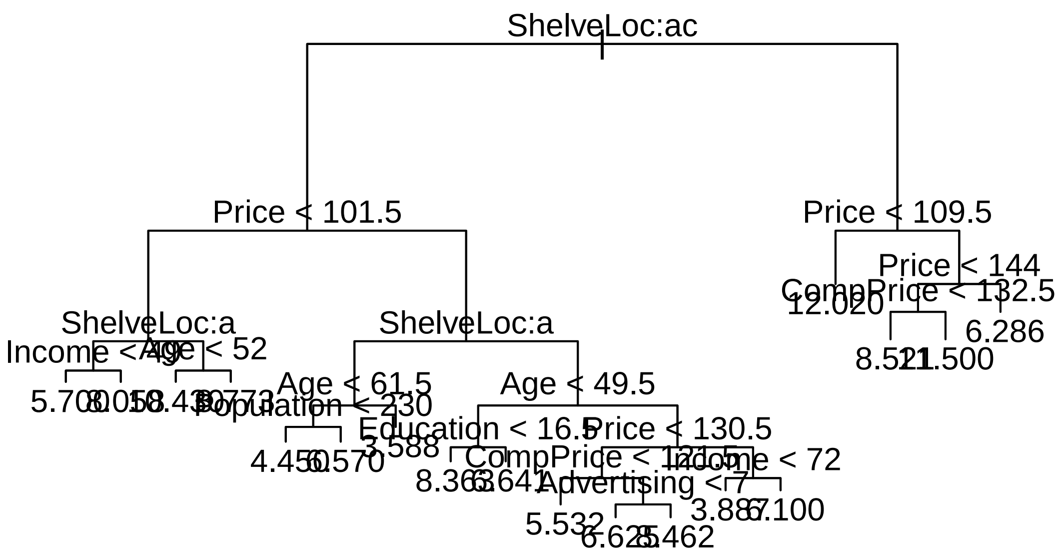
```r
data("Carseats")
set.seed(123)
# Split data set for the tranning and test#
train = sample(1:nrow(Carseats), nrow(Carseats)/2)
```

(b)

```r
# Create a regression tree for housing price prediction using training data #
tree.car = tree(Sales ~ ., data = Carseats, subset = train)

yhat.b = predict(tree.car, newdata = Carseats[-train, ])
car.test = Carseats[-train, "Sales"]

# Plot the regression tree #
plot(tree.car)
text(tree.car)
```




```r
#MSE
mse.b <- mean((yhat.b - car.test)^2);mse.b
```

```
## [1] 4.395357
```

(c)

```r
cv.car = cv.tree(tree.car)
# Plot cross-validation error #
#plot(cv.car$size, cv.car$dev, type='b')
```

```r
# Optimal size selected by cross-validation #
size = cv.car$size[which(cv.car$dev == min(cv.car$dev))];size
```

```
## [1] 15
```

```r
# Learn the regression tree with the selected size using recursive binary splitting #
prune.car = prune.tree(tree.car, best = size)
yhat.c = predict(prune.car, newdata = Carseats[-train, ])

# Mean squared error on the test data #
mse.c<-mean((yhat.c - car.test)^2);mse.c
```

```
## [1] 4.591618
```

```r
mse.b - mse.c
```

```
## [1] -0.1962603
```

Hence, $MSE(b) < MSE(c)$ hold. It implies cross-validation does not decrease the test data MSE.

(d)

```r
##Bagging##
bag.car=randomForest(Sales~.,data=Carseats,subset=train,mtry=10,importance=TRUE)
yhat.bag = predict(bag.car, newdata = Carseats[-train, ])

# Mean squared error on the test data #
mse.d <- mean((yhat.bag - car.test)^2);mse.d
```

```
## [1] 2.706945
```

```r
importance(bag.car)
```

```
##                 %IncMSE IncNodePurity
## CompPrice   20.45893952    163.315084
## Income       5.99352172     88.626184
## Advertising  6.70900949     73.007073
## Population  -1.84004720     53.079505
## Price       46.01586429    395.251820
## ShelveLoc   49.31816789    391.948958
## Age         17.74691675    171.659574
## Education    2.98753578     57.308595
## Urban        0.04864498      7.721022
## US           0.05207339      6.011265
```

(e)

```r
## Prediction using random forest with mtry=p^{1/2} ##
rf.car = randomForest(Sales ~ ., data = Carseats, subset = train, importance = TRUE)
yhat.rf = predict(rf.car, newdata = Carseats[-train, ])
mse.e <- mean((yhat.rf - car.test)^2)
mse.e
```

```
## [1] 3.575026
```

```r
# Compute variable importance #
importance(rf.car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   11.2053520     151.60036
## Income       4.3779492     117.99648
## Advertising  6.1512828      96.48365
## Population  -0.2036319     104.22389
## Price       29.8215710     297.83618
## ShelveLoc   34.5359445     279.70714
## Age         18.8567285     208.55522
## Education    1.5141906      72.05700
## Urban       -1.2098009      15.28180
## US           2.7549016      15.43075
```

(f)

```r
##Gradient Tree Boosting##
boost.car <- gbm(Sales ~ ., data = Carseats[train, ],
                    distribution = "gaussian", n.trees = 5000,
                    interaction.depth = 4)

## Compute test data MSE ##
yhat.boost <- predict(boost.car, newdata = Carseats[-train, ], n.trees = 5000)
mse.f <- mean((yhat.boost - car.test)^2);mse.f
```

```
## [1] 1.97581
```

(g)

```r
mse <- c(mse.b,mse.c,mse.d,mse.e,mse.f)
which(mse == min(mse))
```

```
## [1] 5
```

Hence the boosting acheives the smallest MSE.