HW7

2025-03-11

1 a.

```
# Load necessary libraries
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Function to calculate test error using validation set approach
Validation_Error <- function(split_ratio = 0.5, include_student = FALSE) {
  set.seed(sample(1:10000, 1)) # Random seed for different splits

  # Convert Default variable to binary (1 = Yes, 0 = No)
  Default$default <- ifelse(Default$default == "Yes", 1, 0)

  # Split data into training and validation sets
  n <- nrow(Default)
  train_indices <- sample(1:n, size = floor(split_ratio * n))
  train_data <- Default[train_indices, ]
  test_data <- Default[-train_indices, ]

  # Fit logistic regression model
  if (include_student) {
    model <- glm(default ~ income + balance + student, data = train_data, family = binomial)
  } else {
    model <- glm(default ~ income + balance, data = train_data, family = binomial)
  }

  # Make predictions
  prob_predictions <- predict(model, test_data, type = "response")
  predicted_class <- ifelse(prob_predictions > 0.5, 1, 0)

  # Calculate test error (misclassification rate)
  test_error <- mean(predicted_class != test_data$default)
  return(test_error)
}
```

b.

```r
# Run Validation_Error 100 times and store results
set.seed(123)  # For reproducibility
num_iterations <- 100
test_errors <- numeric(num_iterations)

for (i in 1:num_iterations) {
  test_errors[i] <- Validation_Error(split_ratio = 0.5, include_student = FALSE)
}

# Calculate the average test error
mean_test_error <- mean(test_errors)
cat("Average Test Error (without student variable):", mean_test_error, "\n")
```

```
## Average Test Error (without student variable): 0.02658
```

c.

```r
# Run the function again with student variable included
test_errors_student <- numeric(num_iterations)
for (i in 1:num_iterations) {
  test_errors_student[i] <- Validation_Error(split_ratio = 0.5, include_student = TRUE)
}

# Calculate the average test error with student variable
mean_test_error_student <- mean(test_errors_student)
cat("Average Test Error (with student variable):", mean_test_error_student, "\n")
```

```
## Average Test Error (with student variable): 0.026758
```

```r
# Compare results
if (mean_test_error_student < mean_test_error) {
  cat("Including the student variable reduced the test error.\n")
} else {
  cat("Including the student variable did not significantly reduce the test error.\n")
}
```

```
## Including the student variable did not significantly reduce the test error.
```

Since the test error is nearly identical with and without the student variable (0.02658 vs. 0.026758), including the student variable did not significantly reduce the test error. This suggests that the student status may not be a strong predictor of default when income and balance are already included in the model.

2 a.

```r
# Load necessary libraries
library(ISLR)
library(boot)

# Fit logistic regression model
glm_fit <- glm(default ~ income + balance, data = Default, family = binomial)
```

```
# Display summary to get standard errors
summary(glm_fit)$coefficients[, 2]
```

```
##  (Intercept)       income      balance 
## 4.347564e-01 4.985167e-06 2.273731e-04
```

b.

```
boot.fn <- function(data, index) {
  model <- glm(default ~ income + balance, data = data[index, ], family = binomial)
  return(coef(model))
}
```

c.

```
set.seed(1)
boot_results <- boot(Default, boot.fn, R = 1000)

# Display bootstrap standard errors
boot_results$t0  # Original estimates
```

```
##   (Intercept)        income       balance 
## -1.154047e+01  2.080898e-05  5.647103e-03
```

```
apply(boot_results$t, 2, sd)  # Bootstrap standard errors
```

```
## [1] 4.344722e-01 4.866284e-06 2.298949e-04
```

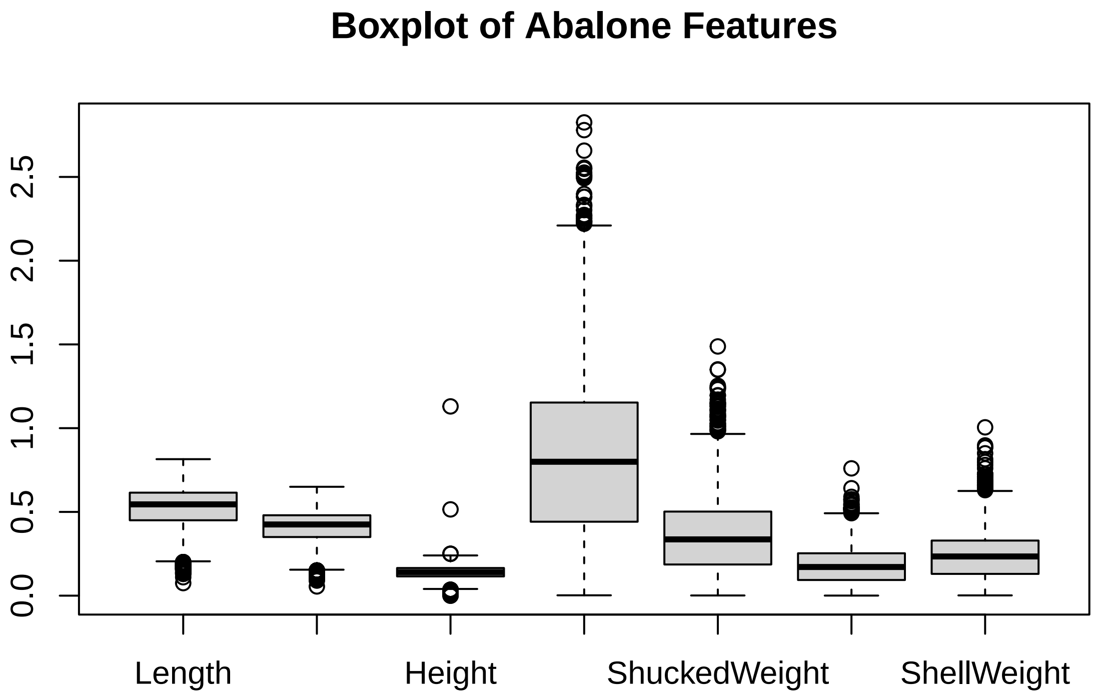
d. Since the results from glm() and bootstrap are nearly identical, there is no significant advantage to using bootstrap here. The standard errors provided by glm() are sufficient.

3 a.

```
# Load necessary library
library(boot)

# Load the Abalone dataset
column_names <- c("Sex", "Length", "Diameter", "Height", "WholeWeight", "ShuckedWeight",
                  "VisceraWeight", "ShellWeight", "Rings")

abalone <- read.csv("abalone.data", header = FALSE, col.names = column_names)

summary(abalone)
```

```
##      Sex                Length         Diameter          Height      
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000  
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150  
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400  
##                     Mean   :0.524   Mean   :0.4079   Mean   :0.1395
##                     3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
##                     Max.   :0.815   Max.   :0.6500   Max.   :1.1300
##   WholeWeight      ShuckedWeight    VisceraWeight     ShellWeight
##  Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015
##  1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300
##  Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340
##  Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388
##  3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290
##  Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050
##      Rings
##  Min.   : 1.000
##  1st Qu.: 8.000
##  Median : 9.000
##  Mean   : 9.934
##  3rd Qu.:11.000
##  Max.   :29.000
```

```
boxplot(abalone[, c("Length", "Diameter", "Height", "WholeWeight",
                    "ShuckedWeight", "VisceraWeight", "ShellWeight")],
        main="Boxplot of Abalone Features")
```



```
# Remove outliers
abalone <- subset(abalone, Height > 0)
```

b.

```
abalone$Age <- abalone$Rings + 1.5
abalone$Rings <- NULL  # Remove Rings column
```

c.

```
set.seed(123)

cv_error <- cv.glm(abalone, glm(Age ~ . -Sex, data = abalone), K = 10)$delta[1]
cat("10-Fold CV Error (Linear Model):", cv_error, "\n")
```

```
## 10-Fold CV Error (Linear Model): 5.078082
```

d.

```
abalone$Length2 <- abalone$Length^2

cv_error_quad <- cv.glm(abalone, glm(Age ~ . -Sex, data = abalone), K = 10)$delta[1]
cat("10-Fold CV Error (Quadratic Model):", cv_error_quad, "\n")
```

```
## 10-Fold CV Error (Quadratic Model): 4.836592
```