Homework 4

Solution 1 Given basis functions:

$$b_1(X) = I(0 \le X \le 1)$$
$$b_2(X) = XI(0 \le X \le 1)$$
$$b_3(X) = I(1 \le X \le 2)$$
$$b_4(X) = XI(1 \le X \le 2)$$

and the model:

$$Y = f(X) + \varepsilon = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \beta_3 b_3(X) + \beta_4 b_4(X) + \varepsilon$$

(a) Figure 1 shows plot with coefficients: $\hat\beta_0 = 1,\ \hat\beta_1 = -1,\ \hat\beta_2 = 1,\ \hat\beta_3 = 1,\ \hat\beta_4 = -2$

On the interval $X \in [0,1]$, the model simplifies to:

$$f(X) = 1 - 1(1) + 1X = X$$

On the interval $X \in [1,2]$, we get:

$$f(X) = 1 + 1(1) - 2X = 2 - 2X$$

For $X = 1$,

$$f(1) = \beta_0 + \beta_1 + \beta_2 + \beta_3 + \beta_4 = 0$$

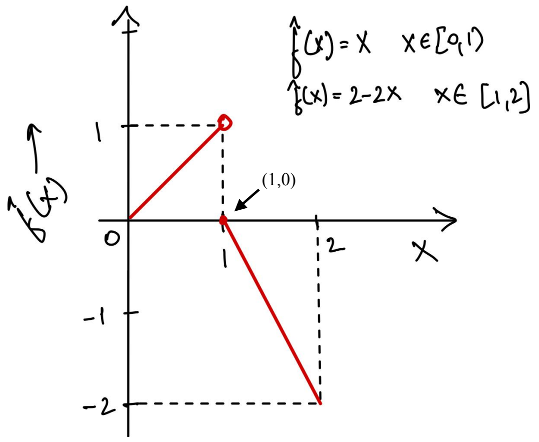

Figure 1: (1a) Estimated piecewise linear function with knot at $X = 1$

(b) There are 5 parameters with no constraints $(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4)$, so the **degrees of freedom is 5**.

(c) Figure 2 shows plot with coefficients: $\hat{\beta}_0 = 1,\ \hat{\beta}_1 = -1,\ \hat{\beta}_2 = 1,\ \hat{\beta}_3 = 2,\ \hat{\beta}_4 = -2$
On the interval $X \in [0, 1]$, the model simplifies to:

$$f(X) = 1 - 1(1) + 1X = X$$

On the interval $X \in [1, 2]$, we get:

$$f(X) = 1 + 2(1) - 2X = 3 - 2X$$

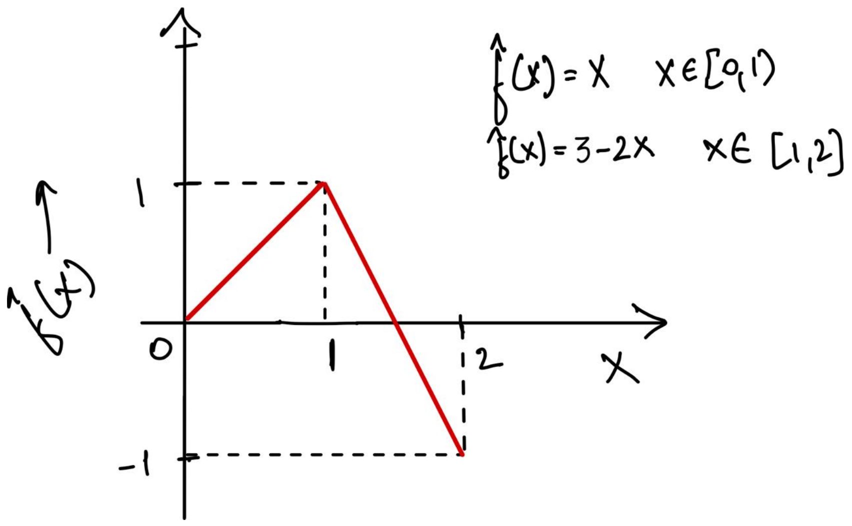

Figure 2: (1c) Estimated piecewise linear function

(d) The function in part (c) is continuous at $X = 1$. In part (a) the function is discontinuous at $X = 1$.

(e) For continuity at $X = 1$ the model values should match from both sides.
Left side : $f(1) = \beta_0 + \beta_1(1) + \beta_2(1)$
Right side : $f(1) = \beta_0 + \beta_3(1) + \beta_4(1)$
Constraint :

$$\beta_0 + \beta_1 + \beta_2 = \beta_0 + \beta_3 + \beta_4$$

$$\beta_1 + \beta_2 = \beta_3 + \beta_4$$

(f) With the constraint, **degrees of freedom would be 4**.

Solution 2 Different regressions to predict Wage using Age.

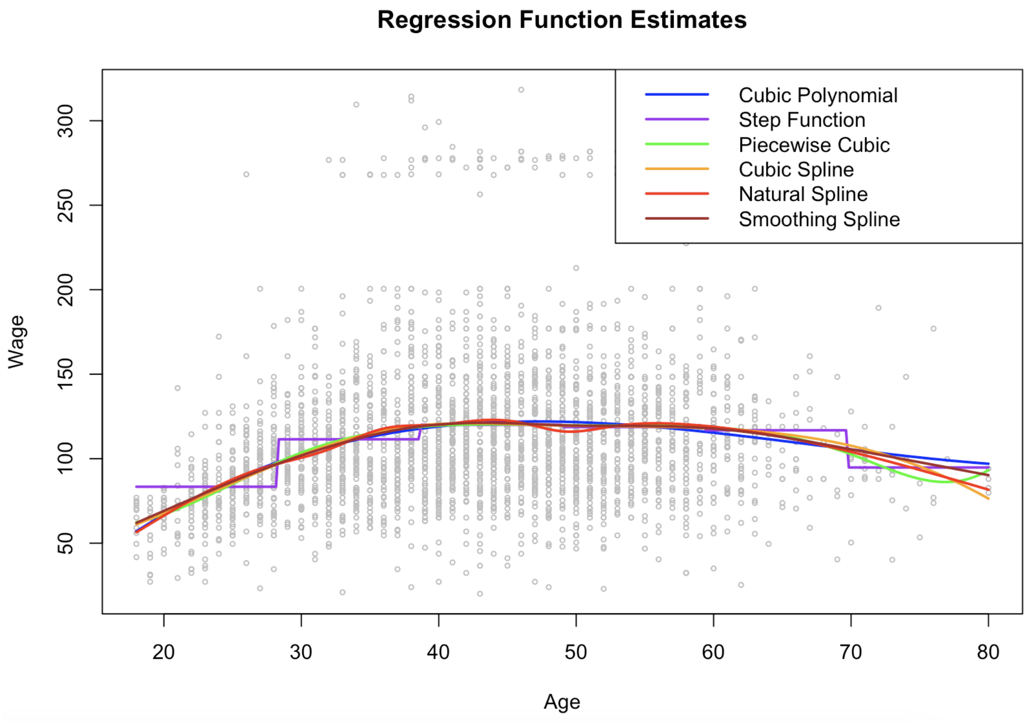

Figure 3: (2) Estimated regression functions

Method	Description	Test MSE
1	Cubic Polynomial	1444.11
2	Step Function	1468.69
3	Piecewise Cubic	1440.70
4	Cubic Spline	1441.59
5	Natural Spline	1442.44
6	Smoothing Spline	1441.14

Table 1: Test MSE for different regression methods

Solution 3

(a) I split the data into a training set (70%) and testing set (30%).

(b) Figure 4 represents the test mean square error v/s degree of freedom plot

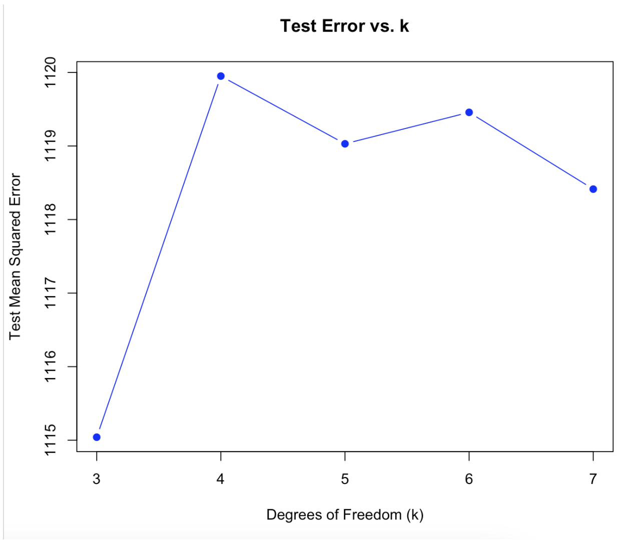

Figure 4: (3b) Test mean square error v/s degree of freedom

(c) From part (b), test error is lowest for $k = 3$. Refitting the model to the entire data.

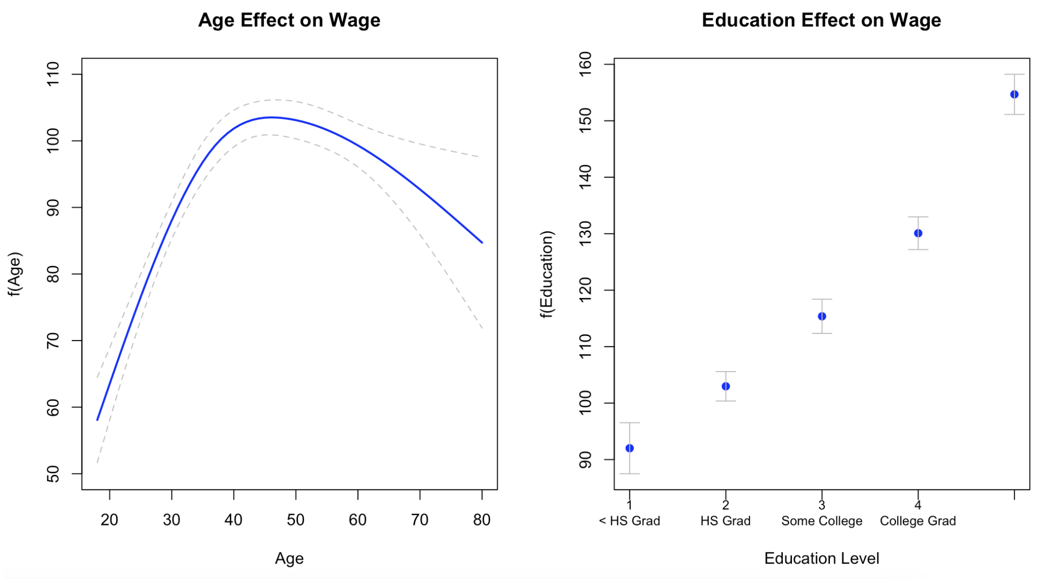

Figure 5: (3c) Estimated mean functions of Age and Education

Solution 4

(a) Figure 6 represents the partition of the feature space.

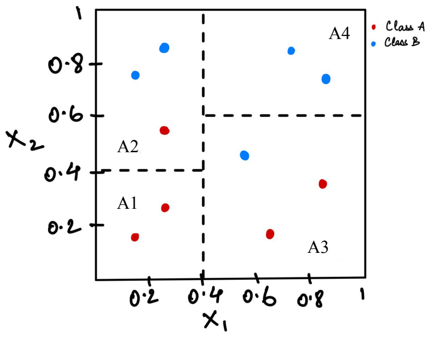

Figure 6: (4a) Partition of feature space

(b) Figure 7 represents the binary tree associated with the partitions in part (a)

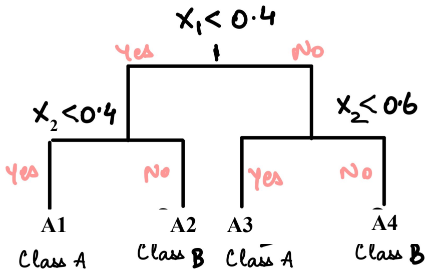

Figure 7: (4b) Binary tree associated with the partitions in part (a)

(c) Figure 8 represents the partition of the feature space. Figure 9 represents the binary tree associated with the partitions.

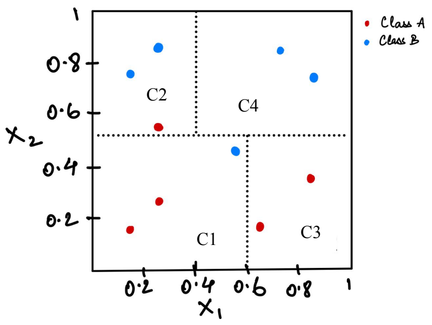

Figure 8: (4c) Partition of feature space

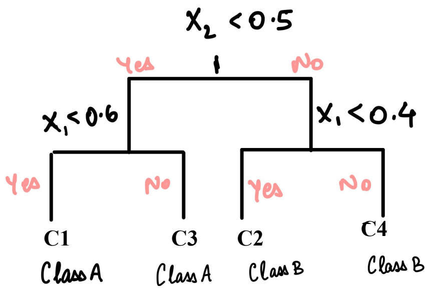

Figure 9: (4c) Binary tree associated with the partitions

(d) Figure 10 represents the partition of the feature space induced by the forest

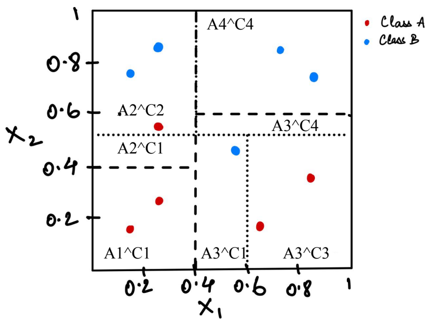

Figure 10: (4d) Forest of partitions from part (a) and (c)

Solution 5

(a) Figure 11 represents train and test errors as a function of tree depth.

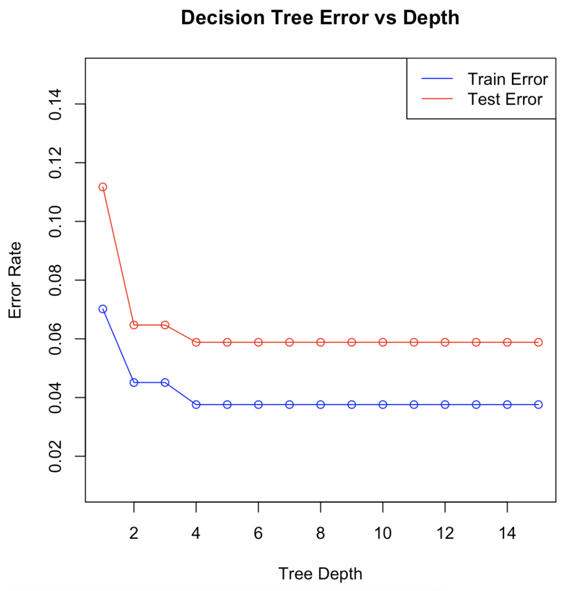

Figure 11: (5a) Train and test errors as a function of tree depth

(b) Figure 12 represents the decision tree with depth 4. It highlights `fractal.dimension_mean` as the most important feature: if it is ≥ 0.15, the tumor is classified as malignant with 96% confidence. For lower values, the tree further splits using `concavity_worst`, `concave.points_se`, and `radius_se` to refine predictions. Most cases with low `fractal.dimension_mean` and small values in these features are classified as benign. The tree is simple and interpretable, relying on a few key features to accurately distinguish between malignant and benign tumors.

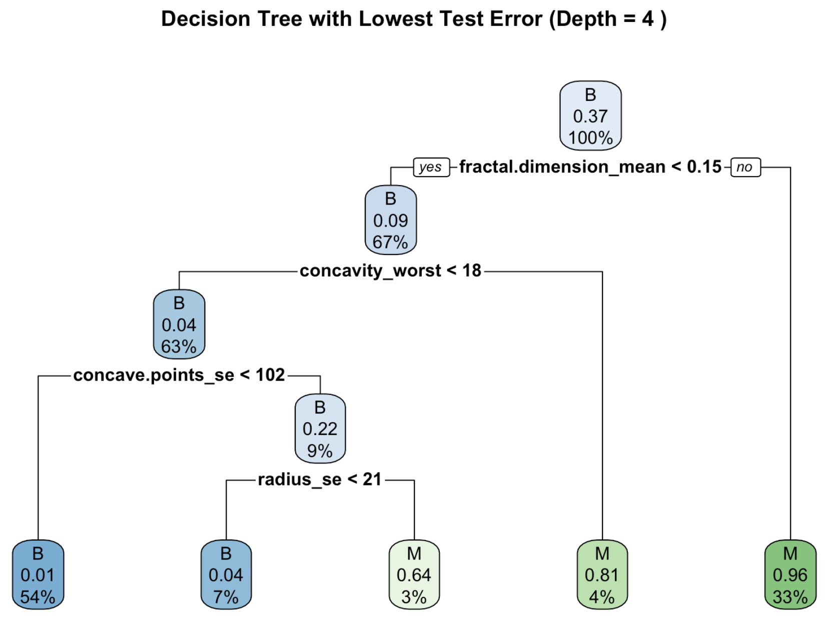

Figure 12: (5b) Decision tree in part (a) with the lowest test error

(c) Figure 13 represents plot of train and test errors as a function of the number of trees across three values of tree depth for random forest model.

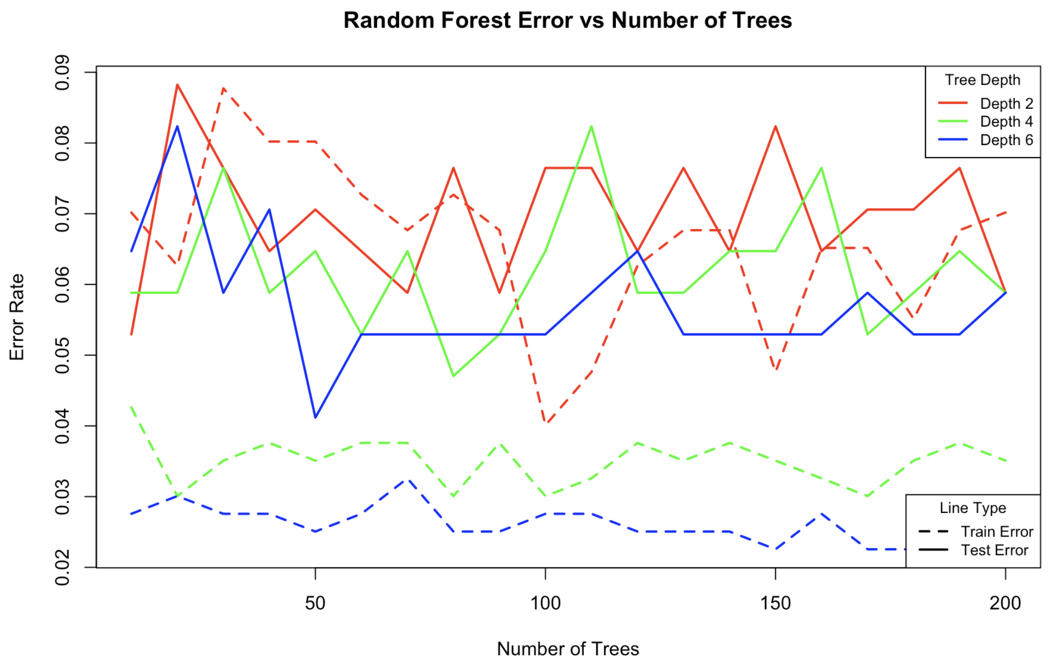

Figure 13: (5c) Train and test errors v/s number of trees, across three values of tree depth

(d) Figure 14 represents train and test errors as a function of the number of iterations, across three values of the learning rate.

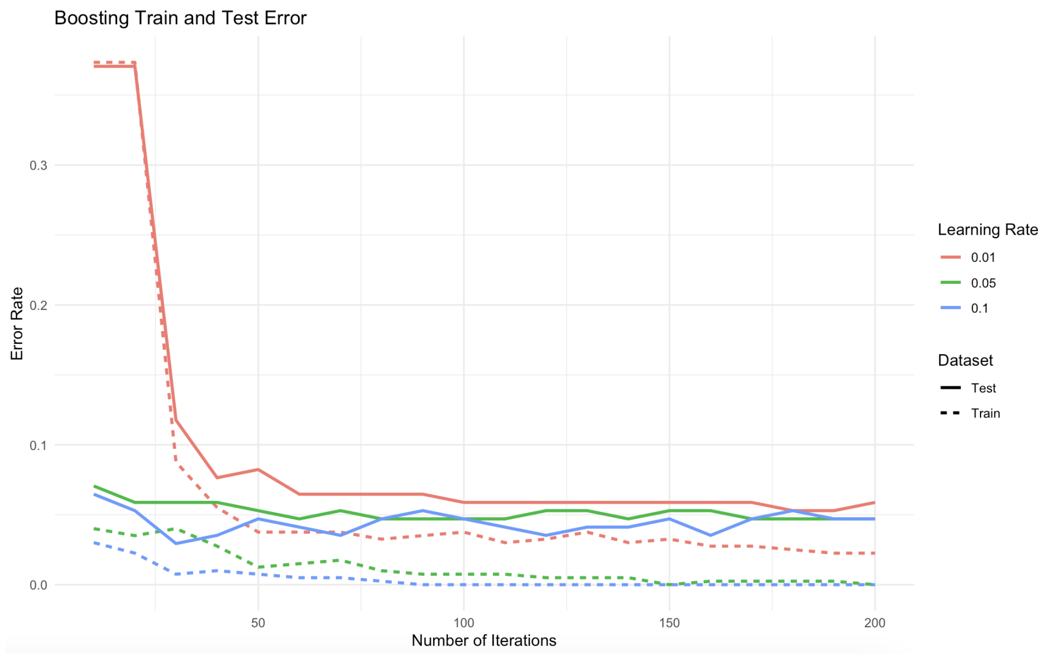

Figure 14: (5d) Train and test errors v/s number of iterations, across three values of the learning rate

APPENDIX

```r
library(ISLR2)
library(splines)
library(gam)

data(Wage)
set.seed(42)

##Question 2
train_index <- sample(1:nrow(Wage), 0.7 * nrow(Wage))
train <- Wage[train_index, ]
test <- Wage[-train_index, ]

age_grid <- seq(min(Wage$age), max(Wage$age), length = 300)

plot(Wage$age, Wage$wage, col = "gray", cex = 0.5, pch = 1,
main = "Regression Function Estimates", xlab = "Age", ylab =
"Wage")

# 1. Cubic Polynomial
fit1 <- lm(wage ~ poly(age, 3), data = train)
pred1 <- predict(fit1, newdata = data.frame(age = age_grid))
lines(age_grid, pred1, col = "blue", lwd = 2)

# 2. Step Function
fit2 <- lm(wage ~ cut(age, 6), data = train)
pred2 <- predict(fit2, newdata = data.frame(age = age_grid))
lines(age_grid, pred2, col = "purple", lwd = 2)

# 3. Piecewise Cubic Polynomial (knots at 30, 50, 70)
fit3 <- lm(wage ~ bs(age, knots = c(30, 50, 70), degree = 3),
data = train)
pred3 <- predict(fit3, newdata = data.frame(age = age_grid))
lines(age_grid, pred3, col = "green", lwd = 2)

# 4. Cubic Spline (df = 5)
fit4 <- lm(wage ~ bs(age, df = 5), data = train)
pred4 <- predict(fit4, newdata = data.frame(age = age_grid))
lines(age_grid, pred4, col = "orange", lwd = 2)

# 5. Natural Cubic Spline (df = 10)
fit5 <- lm(wage ~ ns(age, df = 10), data = train)
pred5 <- predict(fit5, newdata = data.frame(age = age_grid))
lines(age_grid, pred5, col = "red", lwd = 2)

# 6. Smoothing Spline
fit6 <- smooth.spline(train$age, train$wage, cv = TRUE)
pred6 <- predict(fit6, age_grid)$y
lines(age_grid, pred6, col = "brown", lwd = 2)

legend("topright", legend = c("Cubic Polynomial", "Step
Function", "Piecewise Cubic",
"Cubic Spline", "Natural Spline", "Smoothing Spline"),
       col = c("blue", "purple", "green", "orange", "red",
"brown"), lty = 1, lwd = 2)

mse <- function(pred, truth) mean((pred - truth)^2)

mse1 <- mse(predict(fit1, newdata = test), test$wage)
mse2 <- mse(predict(fit2, newdata = test), test$wage)
mse3 <- mse(predict(fit3, newdata = test), test$wage)
mse4 <- mse(predict(fit4, newdata = test), test$wage)
mse5 <- mse(predict(fit5, newdata = test), test$wage)
mse6 <- mse(predict(fit6, test$age)$y, test$wage)

data.frame(
  Method = c("Cubic Polynomial", "Step Function", "Piecewise
Cubic", "Cubic Spline", "Natural Spline", "Smoothing Spline"),
  Test_MSE = round(c(mse1, mse2, mse3, mse4, mse5, mse6), 2)
)

# Question 3

data(Wage)
set.seed(42)

# part a 70/30 split
train_index <- sample(1:nrow(Wage), 0.7 * nrow(Wage))
train <- Wage[train_index, ]
test <- Wage[-train_index, ]

df_values <- 3:7
test_errors <- numeric(length(df_values))

for (i in seq_along(df_values)) {
  k <- df_values[i]
  model <- gam(wage ~ ns(age, df = k) + education, data =
train)
  pred <- predict(model, newdata = test)
  test_errors[i] <- mean((test$wage - pred)^2)
}

# part b
plot(df_values, test_errors, type = "b", pch = 19, col =
"blue",
     xlab = "Degrees of Freedom (k)",
     ylab = "Test Mean Squared Error",
     main = "Test Error vs. k")


# part c
```

```r
best_model <- gam(wage ~ ns(age, df = 3) + education, data =
Wage)
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))
age_seq <- seq(min(Wage$age), max(Wage$age))
pred_data <- data.frame(age = age_seq, education = "2. HS
Grad")
age_pred <- predict(best_model, newdata = pred_data, se.fit =
TRUE)

plot(age_seq, age_pred$fit, type = "l", col = "blue", lwd = 2,
     xlab = "Age", ylab = "f(Age)", main = "Age Effect on
Wage", ylim = c(50, 110))
lines(age_seq, age_pred$fit + 2*age_pred$se.fit, col = "gray",
lty = 2)
lines(age_seq, age_pred$fit - 2*age_pred$se.fit, col = "gray",
lty = 2)

ed_levels <- levels(Wage$education)
pred_data_ed <- data.frame(age = mean(Wage$age), education =
ed_levels)
ed_pred <- predict(best_model, newdata = pred_data_ed, se.fit
= TRUE)

plot(1:length(ed_levels), ed_pred$fit, type = "p", pch = 19,
col = "blue",
     xlab = "Education Level", ylab = "f(Education)", main =
"Education Effect on Wage",
     xaxt = "n", ylim = range(c(ed_pred$fit -
2*ed_pred$se.fit, ed_pred$fit + 2*ed_pred$se.fit)))
axis(1, at = 1:length(ed_levels), labels = gsub("\\. ", "\n",
ed_levels), las = 1, cex.axis = 0.8)
arrows(1:length(ed_levels), ed_pred$fit - 2*ed_pred$se.fit,
       1:length(ed_levels), ed_pred$fit + 2*ed_pred$se.fit,
       angle = 90, code = 3, length = 0.1, col = "gray")

par(mfrow = c(2, 1))


# Question 5

library(rpart)
library(rpart.plot)
library(randomForest)
library(gbm)
library(caret)


data <- read.csv("/Users/ridhijain/Downloads/MSDS courses/
Quarter3/DATA 558/HW4/breast+cancer+wisconsin+diagnostic/
wdbc.data",
 header = FALSE)
colnames(data) <- c("ID", "Diagnosis",
                  paste0(rep(c("radius", "texture",
"perimeter", "area", "smoothness",
                               "compactness", "concavity",
"concave.points", "symmetry", "fractal.dimension"), each = 3),
                         "_", rep(c("mean", "se", "worst"),
times = 10)))

# Drop ID column
data <- data[, -1]
data$Diagnosis <- factor(data$Diagnosis, levels = c("B", "M"))

# part a
set.seed(13)
train_index <- createDataPartition(data$Diagnosis, p = 0.7,
list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

train_errors <- c()
test_errors <- c()
depths <- 1:15

for (d in depths) {
  model <- rpart(Diagnosis ~ ., data = train_data, control =
rpart.control(maxdepth = d))
  train_pred <- predict(model, train_data, type = "class")
  test_pred <- predict(model, test_data, type = "class")
  train_errors <- c(train_errors, mean(train_pred !=
train_data$Diagnosis))
  test_errors <- c(test_errors, mean(test_pred !=
test_data$Diagnosis))
}

plot(depths, train_errors, type = "o", col = "blue", ylim =
c(0.01, 0.15),
     xlab = "Tree Depth", ylab = "Error Rate", main =
"Decision Tree Error vs Depth")
lines(depths, test_errors, type = "o", col = "red")
legend("topright", legend = c("Train Error", "Test Error"),
col = c("blue", "red"), lty = 1)

# part b
best_depth_index <- which.min(test_errors)
best_depth <- depths[best_depth_index]
best_tree_model <- rpart(Diagnosis ~ ., data = train_data,
control = rpart.control(maxdepth = best_depth))

rpart.plot(best_tree_model,
```

```r
           main = paste("Decision Tree with Lowest Test Error
(Depth =", best_depth, ")"),
           extra = 106)

# part c
depth_vals <- c(2, 4, 6)
colors <- c("red", "green", "blue")

plot(NULL, xlim = range(ntree_vals), ylim =
range(c(rf_results$train_error, rf_results$test_error)),
     xlab = "Number of Trees", ylab = "Error Rate",
     main = "Random Forest Error vs Number of Trees")

for (i in seq_along(depth_vals)) {
  d <- depth_vals[i]
  subset_data <- rf_results[rf_results$maxnodes == d, ]
  lines(subset_data$ntree, subset_data$train_error, lty = 2,
lwd = 2, col = colors[i])
  lines(subset_data$ntree, subset_data$test_error, lty = 1,
lwd = 2, col = colors[i])
}

legend("topright",
       legend = c("Depth 2", "Depth 4", "Depth 6"),
       col = colors,
       lty = 1,
       lwd = 2,
       title = "Tree Depth",
       cex = 0.8)

legend("bottomright",
       legend = c("Train Error", "Test Error"),
       col = "black",
       lty = c(2, 1),
       lwd = 2,
       title = "Line Type",
       cex = 0.8)

# part d
nrounds <- seq(10, 200, by = 10)
learning_rates <- c(0.01, 0.05, 0.1)

train_data$y_bin <- ifelse(train_data$Diagnosis == "M", 1, 0)
test_data$y_bin <- ifelse(test_data$Diagnosis == "M", 1, 0)

boost_results <- expand.grid(n.trees = nrounds, shrinkage =
learning_rates)
boost_results$train_error <- NA
boost_results$test_error <- NA
```

```r
for (i in 1:nrow(boost_results)) {
  model <- gbm(y_bin ~ . -Diagnosis - y_bin,
               data = train_data,
               distribution = "bernoulli",
               n.trees = boost_results$n.trees[i],
               interaction.depth = 3,
               shrinkage = boost_results$shrinkage[i],
               verbose = FALSE)
  
  train_prob <- predict(model, train_data, n.trees = 
boost_results$n.trees[i], type = "response")
  test_prob <- predict(model, test_data, n.trees = 
boost_results$n.trees[i], type = "response")
  
  train_pred <- ifelse(train_prob > 0.5, 1, 0)
  test_pred <- ifelse(test_prob > 0.5, 1, 0)
  
  boost_results$train_error[i] <- mean(train_pred != 
train_data$y_bin)
  boost_results$test_error[i] <- mean(test_pred != 
test_data$y_bin)
}

boost_results_long <- rbind(
  data.frame(n.trees = boost_results$n.trees,
             shrinkage = boost_results$shrinkage,
             error = boost_results$train_error,
             type = "Train"),
  data.frame(n.trees = boost_results$n.trees,
             shrinkage = boost_results$shrinkage,
             error = boost_results$test_error,
             type = "Test")
)

ggplot(boost_results_long, aes(x = n.trees, y = error, color = 
as.factor(shrinkage), linetype = type)) +
  geom_line(size = 1) +
  labs(title = "Boosting Train and Test Error",
       x = "Number of Iterations",
       y = "Error Rate",
       color = "Learning Rate",
       linetype = "Dataset") +
  theme_minimal()
```