STAT 4360 (Introduction to Statistical Learning, Spring 2023)
Mini Project 1
Name: Sayema Rahman

---

1. (a) Refer to code

   (b)

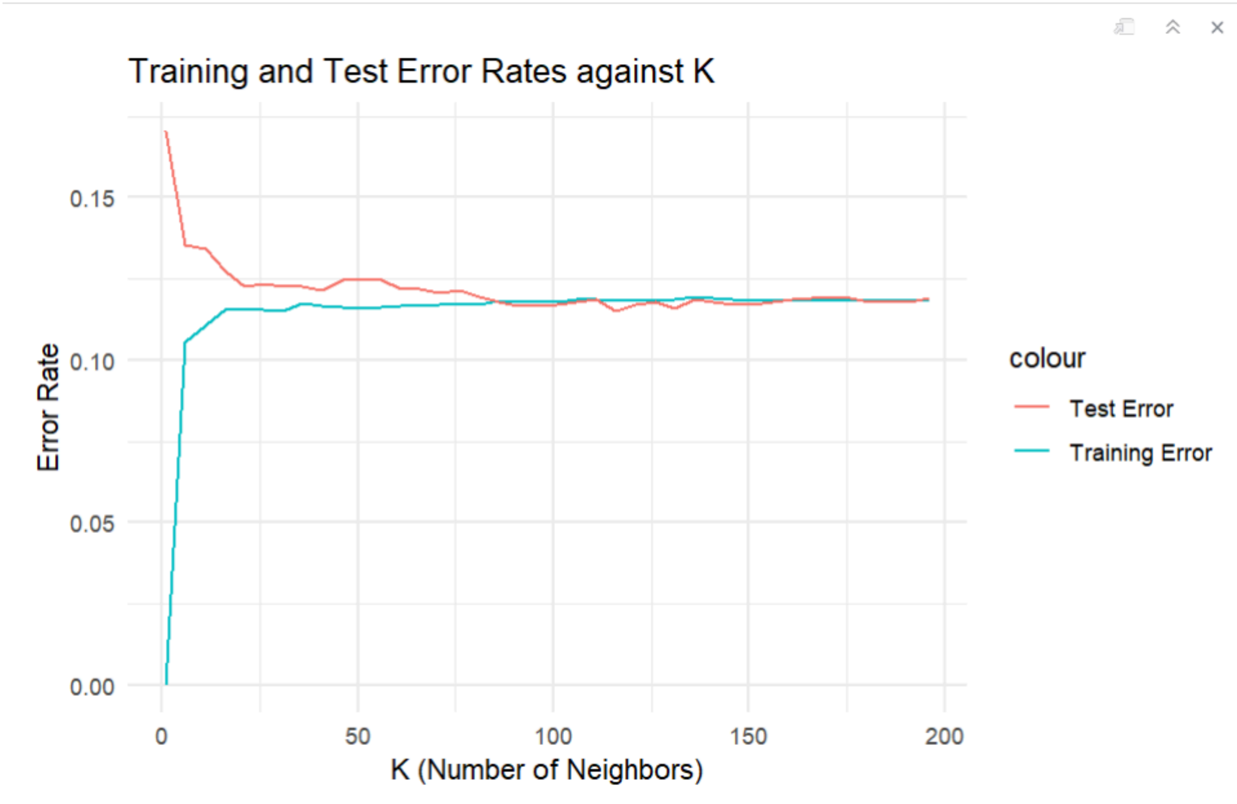
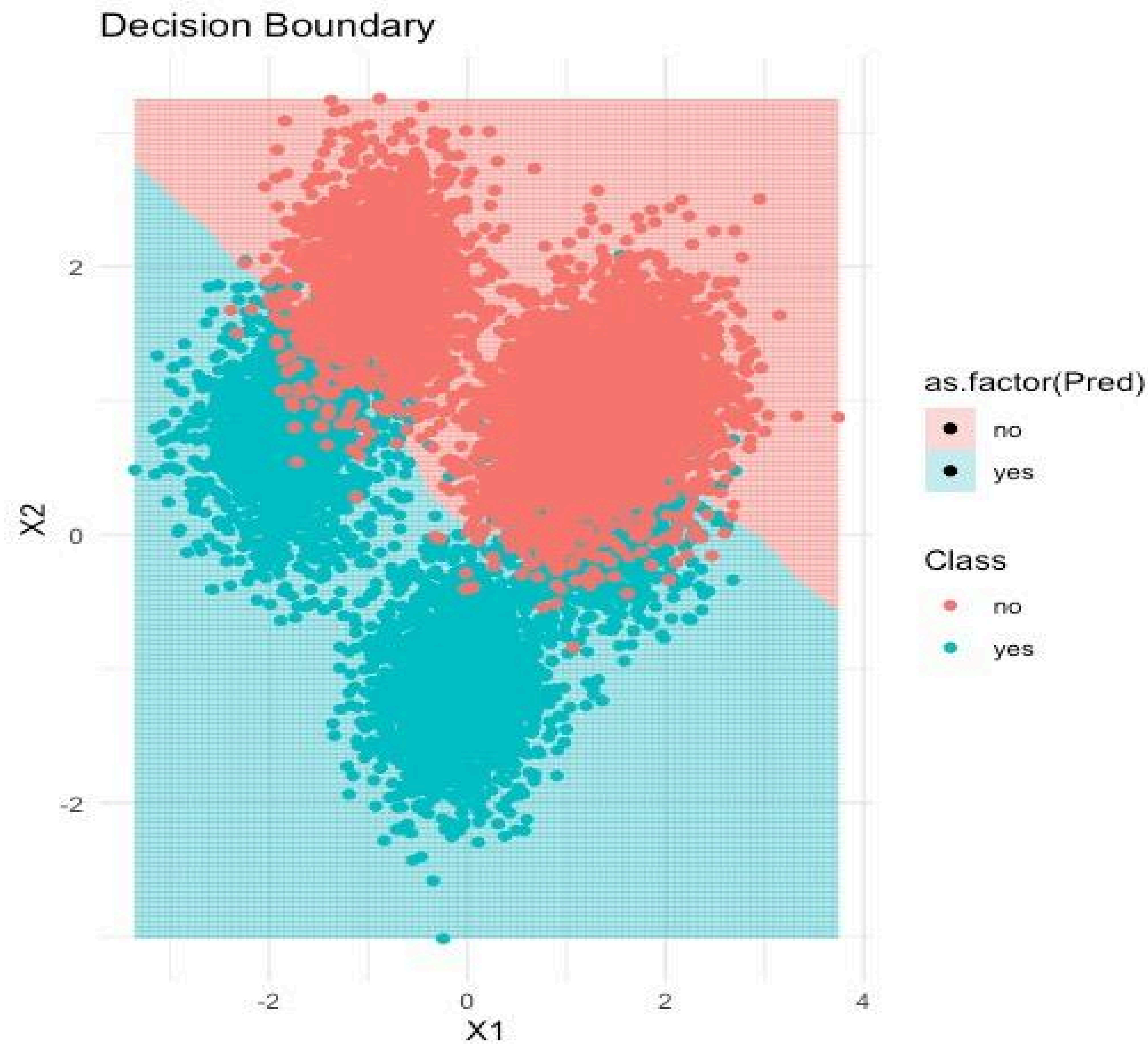


Figure 1: Graph 1

Figure 1 is a scatter plot.

The training error rate went up from 0 to approximately 0.12 and then it continued at a steady rate staying around 0.12 to 0.13. The test error rate starts at 0.19 and goes down to around 0.12 to 0.13 and stays around that range. It is consistent with what I expected
(c) The optimal K value is 131
(d) I fitted the KNN model with the optimal K value given in the code.



(e) I think the decision boundary is sensible. The points near the boundary-cannot be classified as confident as they could be placed in the wrong class. These points are most likely similar in certain features causing them to be so close to each other. The more points that are closer to the decision boundary, the worse the model could be. This could lead to very inaccurate predictions

2. (a)

$$MSE\ \hat{f}(x_0) = E_f\left[(\hat{f}(x_0) - f(x_0))^2\right]$$

$$= E_f\left[(\hat{f}(x_0) - E_f[\hat{f}(x_0)] + E_f[\hat{f}(x_0)] - f(x_0))^2\right]$$

$$= E_f\left[(\hat{f}(x_0) - E_f(\hat{f}(x_0)))^2 + 2(\hat{f}(x_0) - E_f(\hat{f}(x_0)))(E_f(\hat{f}(x_0)) - f(x_0))\right.$$

$$\left. + (E_f(\hat{f}(x_0)) - f(x_0))^2\right]$$

$$= E_f\left[(\hat{f}(x_0) - E_f(\hat{f}(x_0)))^2 + 2(E_f(\hat{f}(x_0)) - f(x_0))E_f(\hat{f}(x_0) - E_f(\hat{f}(x_0)))\right]$$

$$+ (E_f(\hat{f}(x_0)) - \theta)^2$$

$$= E_f\left[(\hat{f}(x_0) - E_f(\hat{f}(x_0)))^2\right] + (E(E_f(\hat{f}(x_0)) - f(x_0))^2)$$

$$= (Bias(\hat{f}(x_0)))^2 + Var(\hat{f}(x_0))$$

(b)

$$var(f(x_0)) = E\left[(f(x_0) - \hat{f}(x_0))^2\right]$$

$$E(\hat{Y}_0 - Y_0)^2 = E\left[(\hat{f}(x_0) + E(f(x_0)) - f(x_0) - e)^2\right]$$

$$= E[(E[f(x_0) - f(x_0)])^2 + (\hat{f}(x_0) - E[f(x_0)])^2 + e^2$$

$$\pm 2(E[f(x_0) - f(x_0)](\hat{f}(x_0) - E[f(x_0)]) + 2e(E[f(x_0) - f(x_0)]))]$$

$$= 0^2 + 0 + \sigma^2 + (Bias[\hat{f}(x_0)])^2 + var[f(x_0)]$$

$$= (Bias[\hat{f}(x_0)])^2 + var[\hat{f}(x_0)] + \sigma^2$$

(c) Based on the idea of the "bias and variance trade off " there is usually a "U" shape for the test MSE when the model's flexibility increases, the square bias decreases and the variance increases. This is usually because model flexibility is when a model can fit many different forms, but involve estimating a greature number of parameters. The change is variance can be strong at certain times that causes it to go back up after when bias dominated it goes down. Although, they can change at different rates, which causes the "U" shape in the Test MSE graph.

Python Code (or R Code)

````
```{r}
# load libraries
library(class)
library(ggplot2)
```
```{r}
# load data
training_data <- read.csv("1-training_data.csv")
testing_data <- read.csv("1-test_data.csv")
```
```{r}
# Get the features and labels
train_features <- training_data[, -ncol(training_data)]
train_labels <- training_data[, ncol(training_data)]
test_features <- testing_data[, -ncol(testing_data)]
test_labels <- testing_data[, ncol(testing_data)]

```
````

Question 1

Part A: Fit KNN with K = 1, 6, 11, . . . , 200.

````
```{r}
k_values <- seq(1, 200, by = 5)

# Initialize the vectors to store error rates
train_error <- numeric(0)
test_error <- numeric(0)

# Fit KNN models for each K value
for (k in k_values) {
 # Train KNN model
 knn_model <- knn(train = train_features, test = train_features, cl = train_labels, k = k)

 # Calculate training error rate
 train_error_rate <- mean(knn_model != train_labels)

 # Store training error rate
 train_error <- c(train_error, train_error_rate)

 # Predict using test data
 knn_pred <- knn(train = train_features, test = test_features, cl = train_labels, k = k)
````
````

```
  # Calculate test error rate
  test_error_rate <- mean(knn_pred != test_labels)

  # Store test error rate
  test_error <- c(test_error, test_error_rate)
}
```
Part B: Plot training and test error rates against K.
```{r}
error_plot <- data.frame(K = k_values, Train_Error = train_error, Test_Error = test_error)
 # making the plot
ggplot(error_plot, aes(x = K)) +
  geom_line(aes(y = Train_Error, color = "Training Error")) +
  geom_line(aes(y = Test_Error, color = "Test Error")) +
  labs(title = "Training and Test Error Rates vs. K",
       x = "K (Number of Neighbors)",
       y = "Error Rate") +
  theme_minimal()
```
Part C: What is the optimal value of K? What are the training and test error rates associated with the optimal K?
```{r}
optimalKvalue <- kvals[which.min(test_error)]
optimalTrainError <- train_error[which.min(test_error)]
optimalTestError <- min(test_error)

cat("Optimal Value of K is", optimalKvalue, "\n")
cat("Training Error with Optimal K is", optimalTrainError, "\n")
cat("Test Error with Optimal K is", optimalTestError, "\n")
```
Part D: Make a plot of the training data that also shows the decision boundary for the optimal K.
```{r}
library(MASS)

optimal_knn_model <- knn(train = train_features, test = test_features, cl = train_labels, k =
optimalKvalue)

# Convert the training data to a data frame for visualization
train_df <- data.frame(X1 = train_features[, 1], X2 = train_features[, 2], Class =
as.factor(train_labels))

# Add predicted labels to the training data
train_df$Pred <- optimal_knn_model
```

```r
# Plot the training data
ggplot(train_df, aes(x = X1, y = X2, color = Class)) +
  geom_point() +
  ggtitle("Training Data with Decision Boundary") +
  theme_minimal()

# Add decision boundary
contour_plot <- function(model, x_range, y_range, h = 0.01) {
  x <- seq(min(x_range), max(x_range), by = h)
  y <- seq(min(y_range), max(y_range), by = h)
  grid <- expand.grid(X1 = x, X2 = y)
  grid$Pred <- knn(train = train_features, test = grid, cl = train_labels, k = model)

  ggplot(grid, aes(x = X1, y = X2, fill = as.factor(Pred))) +
    geom_tile(alpha = 0.3) +
    geom_point(data = train_df, aes(x = X1, y = X2, color = Class)) +
    ggtitle("Decision Boundary") +
    theme_minimal()
}

# Display the plot with the decision boundary
contour_plot(optimalKvalue, range(train_features[, 1]), range(train_features[, 2]))
```