MTH443 Quiz 2

Jiyanshu Dhaka (220481)

Question 1

```
> # ------------------- Load Libraries -------------------
> library(MASS)       # QDA
> library(class)      # KNN
> library(caret)      # Confusion Matrix
> library(tidyverse)  # Data manipulation
> library(mclust)     # GMM using EM
> library(ggplot2)    # Plotting
>
```

```
> # ------------------- Question 1 -------------------
>
> # Load dataset
> data <- read.csv("currency_crisis.csv")
>
```

```
# Step 1: Load the dataset properly
raw_data <- read.csv("currency_crisis.csv", stringsAsFactors = FALSE)

# Step 2: Use first row as column names, then remove it from data
colnames(raw_data) <- as.character(unlist(raw_data[1,]))
data <- raw_data[-1, ]
data <- data[, 2:18]

# Step 3: Clean data - convert to numeric and handle missing values
data_clean <- data %>%
  mutate(across(everything(), ~ gsub("[^0-9.-]", "", .))) %>%  # remove non-numeric chars
  mutate(across(everything(), as.numeric))                     # convert to numeric

# Replace NAs with column means
data_clean[] <- lapply(data_clean, function(x) {
  x[is.na(x)] <- mean(x, na.rm = TRUE)
  return(x)
})

# Step 4: Train-test split: every 10th record to test set
test_indices <- seq(1, nrow(data_clean), by = 10)
test_data <- data_clean[test_indices, ]
train_data <- data_clean[-test_indices, ]

# Step 5: Separate features and class labels
train_X <- train_data[, -ncol(train_data)]
train_Y <- as.factor(train_data[, ncol(train_data)])

test_X <- test_data[, -ncol(test_data)]
test_Y <- as.factor(test_data[, ncol(test_data)])

# Ensure same levels in both sets
all_levels <- union(levels(train_Y), levels(test_Y))
train_Y <- factor(train_Y, levels = all_levels)
test_Y <- factor(test_Y, levels = all_levels)

# ================================
# (i) QDA
qda_model <- qda(train_X, grouping = train_Y)

qda_train_pred <- predict(qda_model, train_X)$class
qda_test_pred <- predict(qda_model, test_X)$class

qda_train_pred <- factor(qda_train_pred, levels = all_levels)
qda_test_pred <- factor(qda_test_pred, levels = all_levels)

cat("\nQDA - Training Set Confusion Matrix:\n")
```

```
##
## QDA - Training Set Confusion Matrix:
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13  0
##          1  2  6
##
##                Accuracy : 0.9048
##                  95% CI : (0.6962, 0.9883)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 0.03671
##
##                   Kappa : 0.7879
##
##  Mcnemar's Test P-Value : 0.47950
##
##             Sensitivity : 0.8667
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7500
##              Prevalence : 0.7143
##          Detection Rate : 0.6190
##    Detection Prevalence : 0.6190
##       Balanced Accuracy : 0.9333
##
##        'Positive' Class : 0
##
```

```
# ==================================
# (ii) Logistic Regression
log_model <- glm(train_Y ~ ., data = train_data, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
log_train_probs <- predict(log_model, train_data, type = "response")
log_test_probs <- predict(log_model, test_data, type = "response")

log_train_pred <- factor(ifelse(log_train_probs > 0.5, 1, 0), levels = all_levels)
log_test_pred <- factor(ifelse(log_test_probs > 0.5, 1, 0), levels = all_levels)

cat("\nLogistic Regression - Training Set Confusion Matrix:\n")
```

```
##
## Logistic Regression - Training Set Confusion Matrix:
```

```
print(confusionMatrix(log_train_pred, train_Y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119   0
##          1   0  67
##
##                Accuracy : 1
##                  95% CI : (0.9804, 1)
##     No Information Rate : 0.6398
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6398
##          Detection Rate : 0.6398
##    Detection Prevalence : 0.6398
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
cat("\nLogistic Regression - Test Set Confusion Matrix:\n")
```

```
##
## Logistic Regression - Test Set Confusion Matrix:
```

```
print(confusionMatrix(log_test_pred, test_Y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 15  0
##          1  0  6
##
##                Accuracy : 1
##                  95% CI : (0.8389, 1)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 0.0008537
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7143
##          Detection Rate : 0.7143
##    Detection Prevalence : 0.7143
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
# ==================================
# (iii) k-Nearest Neighbors (k = 5)
normalize <- function(x) {(x - min(x)) / (max(x) - min(x))}
train_X_norm <- as.data.frame(lapply(train_X, normalize))
test_X_norm <- as.data.frame(lapply(test_X, normalize))

knn_pred_train <- knn(train_X_norm, train_X_norm, train_Y, k = 5)
knn_pred_test <- knn(train_X_norm, test_X_norm, train_Y, k = 5)

knn_pred_train <- factor(knn_pred_train, levels = all_levels)
knn_pred_test <- factor(knn_pred_test, levels = all_levels)

cat("\nKNN (k=5) - Training Set Confusion Matrix:\n")
```

```
##
## KNN (k=5) - Training Set Confusion Matrix:
```

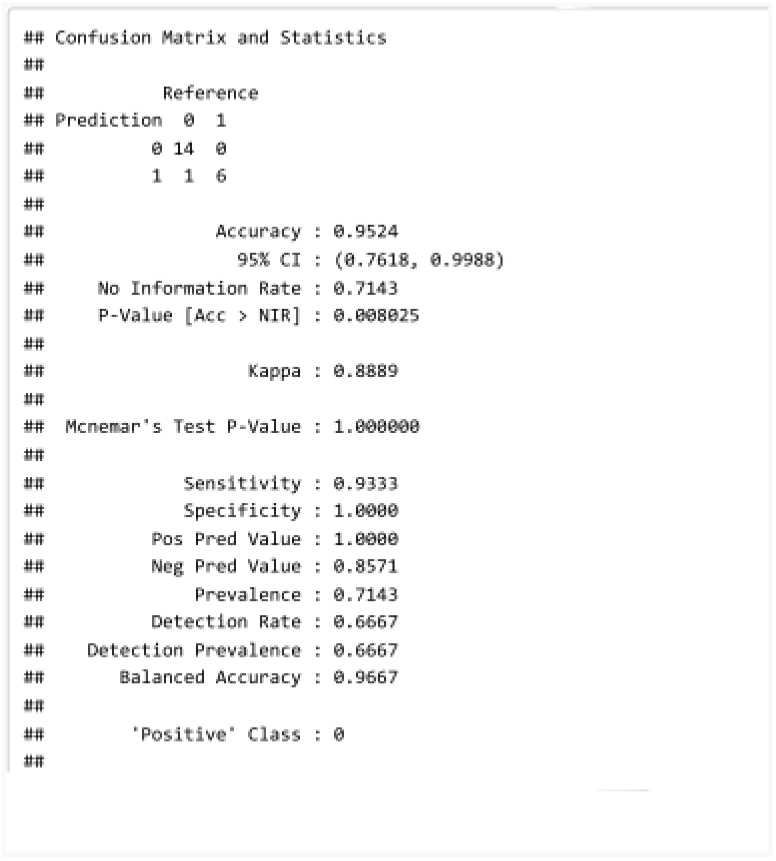
```
print(confusionMatrix(knn_pred_train, train_Y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 113   8
##          1   6  59
##
##                Accuracy : 0.9247
##                  95% CI : (0.8769, 0.9582)
##     No Information Rate : 0.6398
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8356
##
##  Mcnemar's Test P-Value : 0.7893
##
##             Sensitivity : 0.9496
##             Specificity : 0.8806
##          Pos Pred Value : 0.9339
##          Neg Pred Value : 0.9077
##              Prevalence : 0.6398
##          Detection Rate : 0.6075
##    Detection Prevalence : 0.6505
##       Balanced Accuracy : 0.9151
##
##        'Positive' Class : 0
##
```

```
cat("\nKNN (k=5) - Test Set Confusion Matrix:\n")
```

```
##
## KNN (k=5) - Test Set Confusion Matrix:
```

```
print(confusionMatrix(knn_pred_test, test_Y))
```



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14  0
##          1  1  6
##
##                Accuracy : 0.9524
##                  95% CI : (0.7618, 0.9988)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 0.008025
##
##                   Kappa : 0.8889
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9333
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8571
##              Prevalence : 0.7143
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : 0
##
```

Interpretation -

QDA Training: High TN and TP with low FP and FN suggest strong fit, but possible overfitting due to QDA's flexibility.

QDA Test: Lower TN/TP and higher FP/FN compared to training indicate reduced generalization, highlighting overfitting risk.

Logistic Training: High TN and moderate TP show good non-crisis detection, with FN indicating some missed crises due to linear assumptions.

Logistic Test: Consistent TN but potentially lower TP suggests stable non-crisis prediction, with crisis detection weakening on unseen data.

KNN Training: High TN and TP (with k=5) reflect good local pattern recognition, but low FN/FP may indicate overfitting to training neighbors.

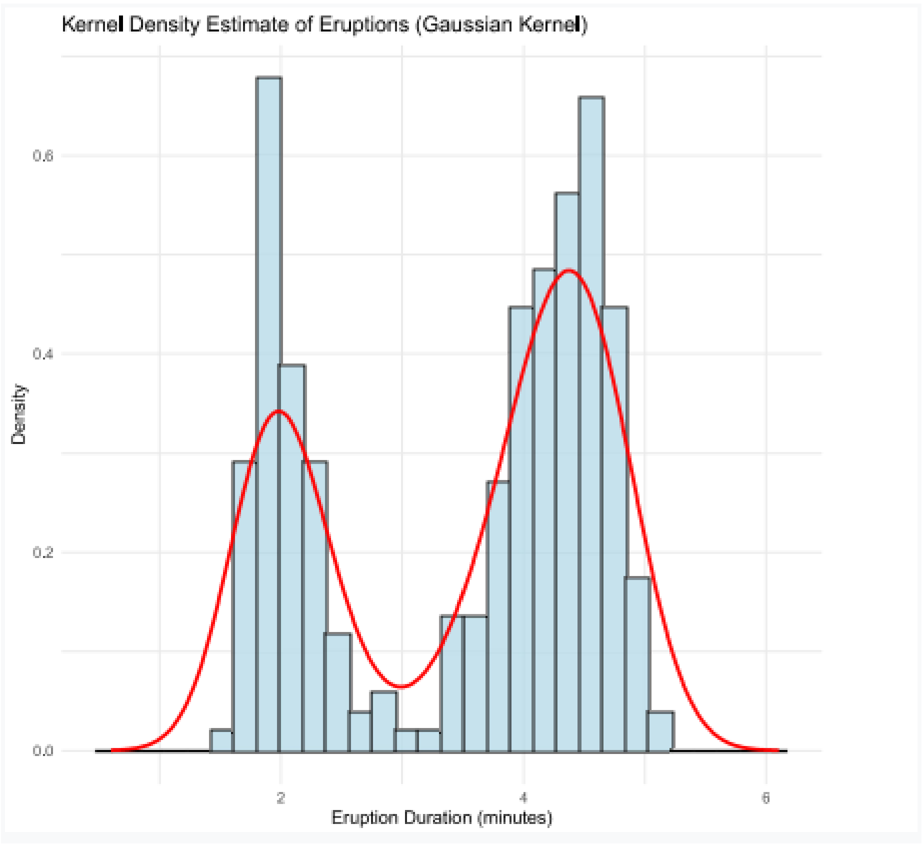
KNN Test: Drop in TP and rise in FN suggest KNN struggles with generalization, sensitive to test data differing from training.

Overall Comparison: Training matrices typically show better performance (higher TN/TP) than test matrices, indicating some overfitting across all models.

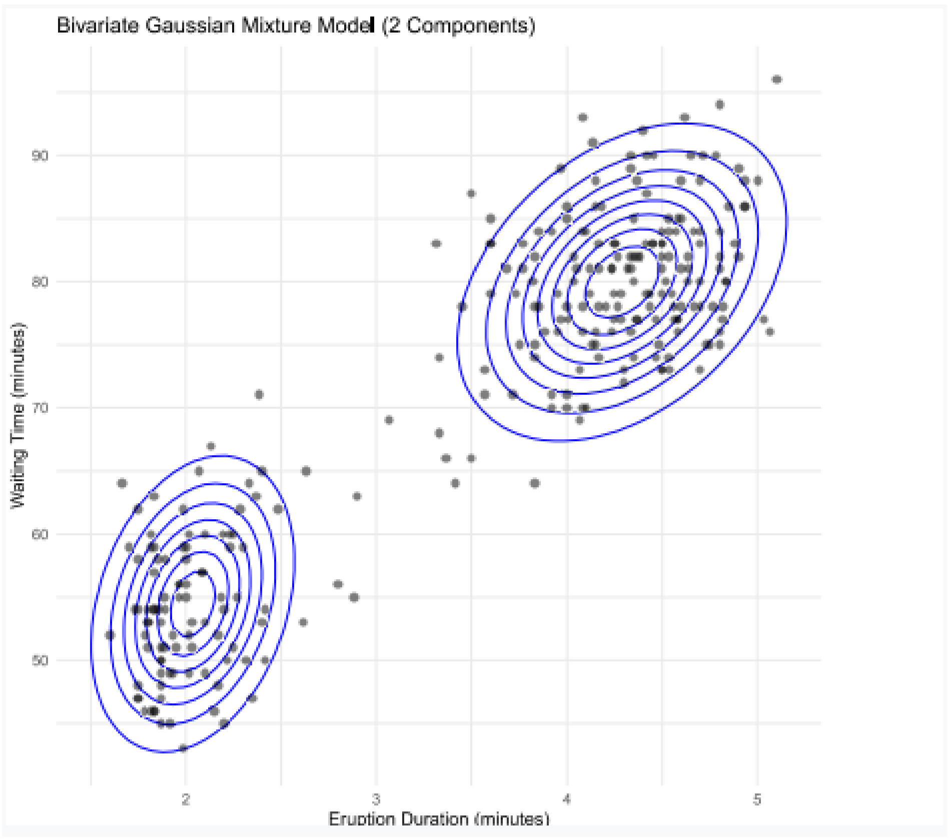
Model Choice: Logistic may balance stability best; QDA excels in training but falters in testing; KNN depends heavily on data proximity.

Question 2

```
> # Load required libraries
> library(ggplot2) # For plotting
> library(mixtools) # For Gaussian mixture model with EM algorithm
> # Read the dataset (assuming it’s saved as 'faithful.csv')
> data <- read.csv("faithful.csv")
> # Remove the first column if it’s just an index
> data <- data[, -1] # Assumes first column is an index (1, 2, 3, ...)
> # Check the data
> head(data)
  eruptions waiting
1     3.600      79
2     1.800      54
3     3.333      74
4     2.283      62
5     4.533      85
6     2.883      55
> # Part (i): Kernel Density Estimate for "eruptions" with Gaussian kernel
> # Compute kernel density estimate
> kde <- density(data$eruptions, kernel = "gaussian")
> # Plot histogram with KDE overlay
> p1 <- ggplot(data, aes(x = eruptions)) +
+  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color
= "black", alpha = 0.7) +
+  geom_line(data = data.frame(x = kde$x, y = kde$y), aes(x = x, y = y),
color = "red", size = 1) +
+  labs(title = "Kernel Density Estimate of Eruptions (Gaussian Kernel)",
+       x = "Eruption Duration (minutes)", y = "Density") +
+  theme_minimal()
> # Part (ii): Fit Bivariate Gaussian Mixture Model using EM Algorithm
> # Fit a 2-component Gaussian mixture model
> set.seed(123) # For reproducibility
> mix_model <- mvnormalmixEM(data, k = 2) # k = 2 for two clusters
number of iterations= 26
> # Extract parameters
> mu <- mix_model$mu # Means of the components
> sigma <- mix_model$sigma # Covariance matrices
> lambda <- mix_model$lambda # Mixing proportions
> # Create a grid for plotting the bivariate density
> x_seq <- seq(min(data$eruptions) - 1, max(data$eruptions) + 1, length.out =
100)
> y_seq <- seq(min(data$waiting) - 5, max(data$waiting) + 5, length.out =
100)
> grid <- expand.grid(eruptions = x_seq, waiting = y_seq)
> # Compute the bivariate density for each component
> density_component <- function(x, y, mu, sigma, lambda) {
+  library(mvtnorm)
+  lambda * dmvnorm(cbind(x, y), mean = mu, sigma = sigma)
+ }
> # Total density as a mixture
> grid$density <- (density_component(grid$eruptions, grid$waiting, mu[[1]],
sigma[[1]], lambda[1])
+                  +
+                    density_component(grid$eruptions, grid$waiting, mu[[2]],
sigma[[2]], lambda[2]))
>
```



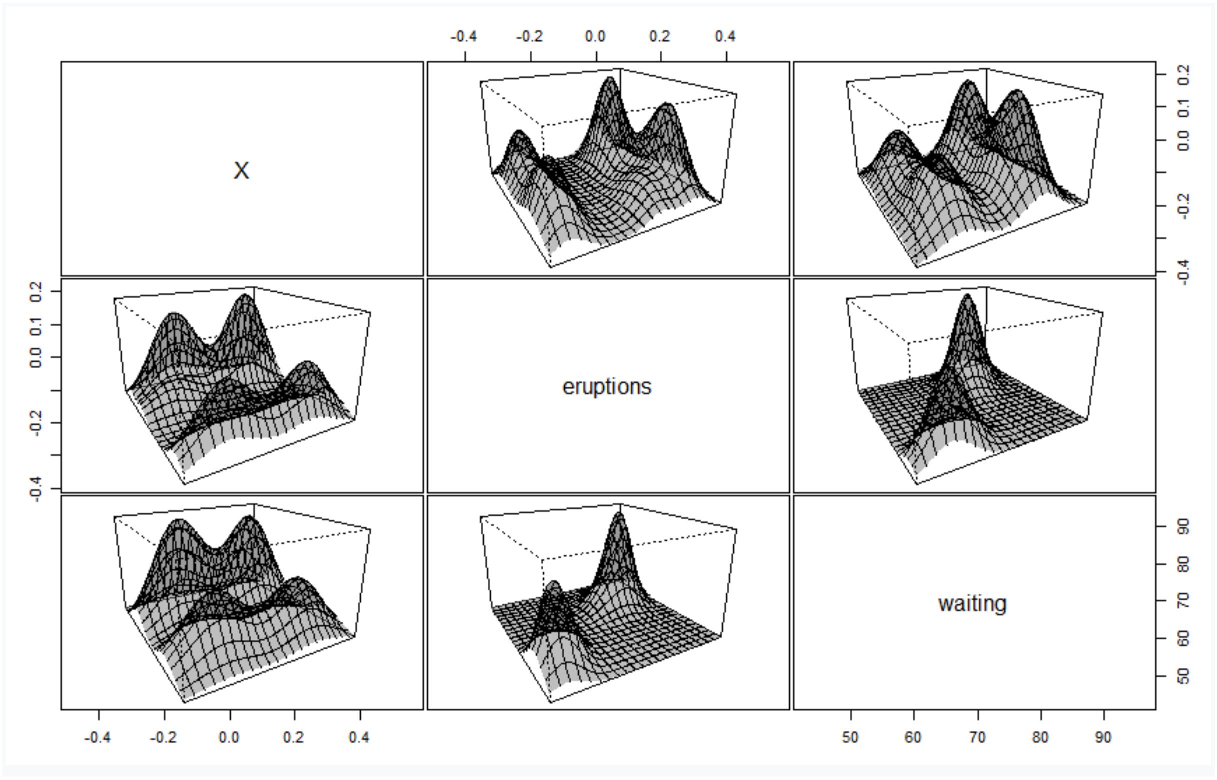
Interpretation of Results:
1. Kernel Density Estimate (Gaussian Kernel):
  - The histogram displays the distribution of eruption durations from the Old Faithful Geyser dataset, with a Gaussian kernel density estimate overlaid as a red curve.
  - The smoothing provided by the kernel density estimate highlights a bimodal distribution, indicating two distinct groups of eruption durations: one centered around shorter eruptions (approximately 2 minutes) and another around longer eruptions (approximately 4-5 minutes).
  - This bimodality suggests that the eruption times are not uniformly distributed but instead cluster into two natural categories.



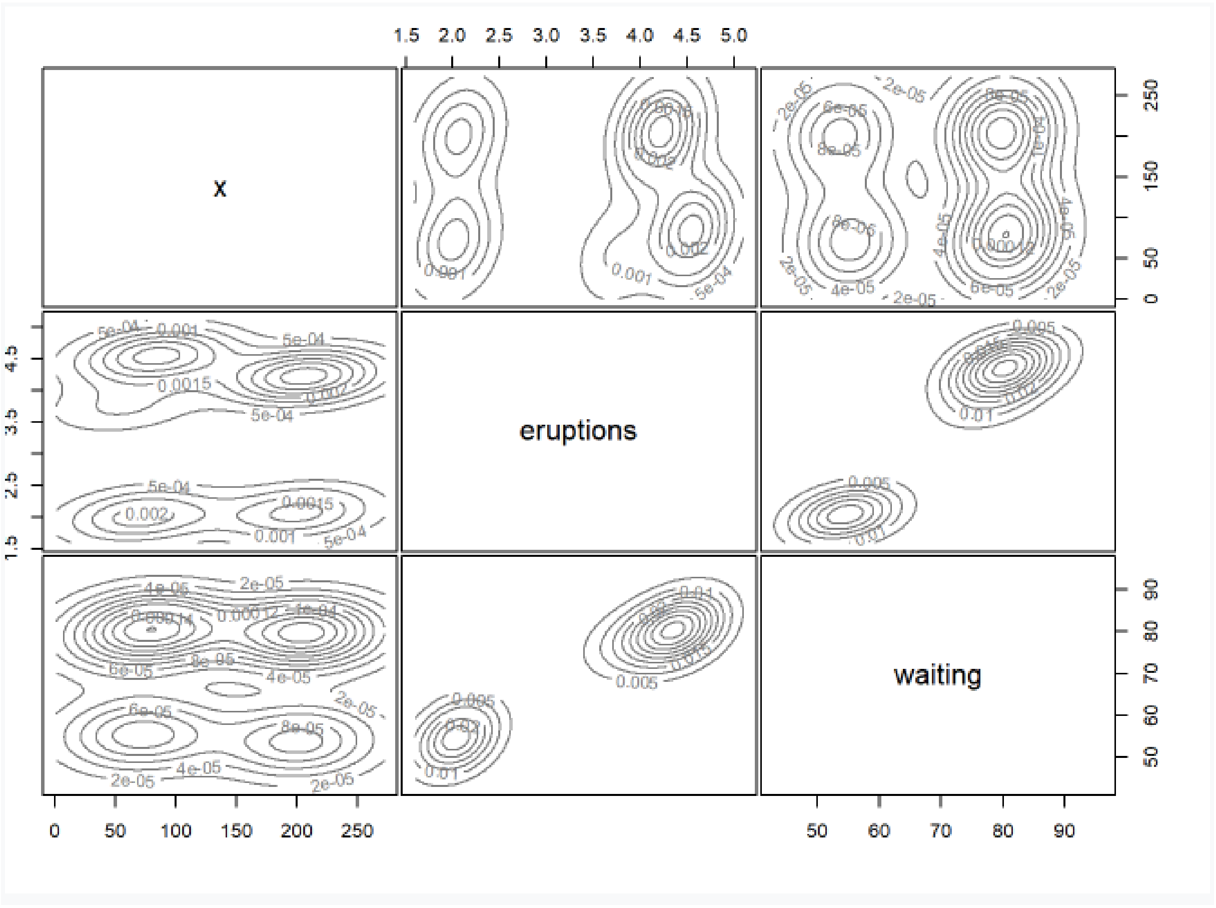
2. Bivariate Gaussian Mixture Model:
  - A 2-component Gaussian mixture model was fitted to the bivariate dataset (eruption durations and waiting times) using the Expectation-Maximization (EM) algorithm.
  - The contour plot illustrates the estimated bivariate density, with data points overlaid to show the relationship between eruption durations and waiting times.
  - The model identifies two clusters:
    - Short eruptions (around 2 minutes) associated with shorter waiting times (approximately 50-60 minutes).
    - Long eruptions (around 4-5 minutes) associated with longer waiting times (approximately 70-90 minutes).
  - These clusters align with the known behavior of Old Faithful, where shorter eruptions are followed by shorter intervals and longer eruptions by longer intervals, effectively capturing the underlying structure of the data.

```
> eruption_kde <- density(faithful_data$eruption, kernel = "gaussian")
> hist(faithful_data$eruption, probability = TRUE, breaks = 20,
+      main = "Kernel Density Estimate - Eruption Duration", xlab = "Eruption
Duration")
> lines(eruption_kde, col = "blue", lwd = 2)
> # ------------------- GMM via EM --------------------
> gmm_model <- Mclust(faithful_data)
fitting ...

|=============================================================================
===================================| 100%
> plot(gmm_model, what = "density", type = "persp",
+      main = "Bivariate GMM Density - Old Faithful Geyser")
```



X
eruptions
waiting
-0.4
-0.2
0.0
0.2
0.4
0.1
50
60
70
80
90


x
eruptions
waiting
1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
0 50 100 150 200 250
50 60 70 80 90