```r
# Load necessary libraries
library(caTools)
library(class)
library(caret)
library(gmodels)
library(ggplot2)

# Set seed for reproducibility
set.seed(123)

# Read data
data <- read.csv("data.csv")

# Convert factors to numeric
data$Sepsis_Result <- factor(data$Sepsis_Result, levels = c(1, 0), labels = c(1, 0))
data$Gender <- factor(data$Gender, levels = c(1, 0), labels = c(1, 0))

# Display the distribution of Sepsis_Result
table(data$Sepsis_Result)

# Split data into train and test sets
split <- sample.split(data, SplitRatio = 0.75)
Train_data <- subset(data, split == "TRUE")
Test_data <- subset(data, split == "FALSE")

# Display structure of train and test data
str(Train_data)
str(Test_data)

# Select features
Train_data1 <- Train_data[1:9]
Test_data1 <- Test_data[1:9]

# Display structure of selected features
str(Train_data1)
str(Test_data1)

# Define target variable
cluster <- Train_data$Sepsis_Result

# Check for missing values
sum(is.na(Train_data1))
sum(is.na(Test_data1))
sum(is.na(cluster))

# KNN model
library(class)

# Data preparation
k_values <- c(1, 3, 5, 7, 15, 19)

# Calculate accuracy for each k value
accuracy_values <- sapply(k_values, function(k) {
  model <- knn(train = Train_data1, test = Test_data1, cl = cluster, k = k)
  1 - mean(model != Test_data$Sepsis_Result)
})

#Accuracy formula
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

# Display accuracy for each k value
accuracy_values

# Calculate confusion matrix
```

```
model1 <- knn(train = Train_data1, test = Test_data1, cl = cluster, k = 5)
ac_table <- table(Test_data$Sepsis_Result, model1)

# Display confusion matrix
ac_table

# Calculate accuracy
accuracy(ac_table)

# Create dataframe for accuracy values
accuracy_data <- data.frame(K = k_values, Accuracy = accuracy_values)

# Display accuracy data
accuracy_data

# Calculate confusion matrix
xtab <- table(model1, Test_data$Sepsis_Result)

# Calculate confusion matrix
cm <- caret::confusionMatrix(xtab, mode = "everything", positive = "1")

# Display confusion matrix
cm

# Plotting accuracy for different K values
ggplot(accuracy_data, aes(x = K, y = Accuracy * 100)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 3) +
  labs(title = "Model Accuracy for Different K Values",
       x = "Number of Neighbors (K)",
       y = "Accuracy") +
  theme_minimal()

# Support Vector Machine (SVM)
library(e1071)
library(caret)

# Prepare data for SVM
svm_train <- Train_data[1:10]
svm_test <- Test_data[1:9]

# Display structure of SVM test and train data
str(svm_test)
str(svm_train)

# Define parameter grid for tuning
param_grid <- expand.grid(C = c(0.1, 1),
                          sigma = c(0.1, 1))  # sigma parameter not used in e1071, but
required for parameter grid

# Set up cross-validation control
ctrl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)

# Train SVM model
model2 <- train(Sepsis_Result ~ ., data = svm_train, method = "svmRadial", trControl =
ctrl, tuneLength = 8, tuneGrid = param_grid)

# Make predictions
predictions1 <- predict(model2, newdata = svm_test)

# Visualize the model
plot(model2)

# Check if the length of predictions matches the length of test data
length(predictions1) == length(Test_data$Sepsis_Result)
```

```
# Display the best tuning parameters
print(model2$bestTune)

# Display SVM model details
print(model2)

# Calculate confusion matrix for SVM
xtab1 <- table(predictions1, Test_data$Sepsis_Result)

# Compute confusion matrix
cm1 <- caret::confusionMatrix(xtab1, mode = "everything", positive = "1")

# Display confusion matrix
cm1

# Decision Tree
library(rpart)
library(rpart.plot)
library(ggplot2)

# Prepare data for Decision Tree
dt_train <- Train_data[1:10]
dt_test <- Test_data[1:9]

# Build Decision Tree model
model3 <- rpart(formula = Sepsis_Result ~ ., data = dt_train, control =
rpart.control(minsplit = 1))

# Make predictions
predictions2 <- predict(model3, newdata = dt_test, type = "class")

# Plot Decision Tree
rpart.plot(model3)

# Compute Decision Tree accuracy
CrossTable(x = Test_data$Sepsis_Result, y = predictions2, prop.chisq = FALSE)

# Compute confusion matrix for Decision Tree
confusion_matrix <- table(predictions2, Test_data$Sepsis_Result)
print(confusion_matrix)
accuracy(confusion_matrix)

# Compute confusion matrix for Decision Tree
xtab2 <- table(predictions2, Test_data$Sepsis_Result)
cm2 <- caret::confusionMatrix(xtab2, mode = "everything", positive = "1")
cm2


# LightGBM
library(lightgbm)

# Prepare data for LightGBM
lgbm_train <- Train_data[1:10]
lgbm_test <- Test_data[1:9]
lgbm_train$Gender <- as.numeric(as.character(lgbm_train$Gender))
lgbm_test$Gender <- as.numeric(as.character(lgbm_test$Gender))
lgbm_train$Sepsis_Result <- as.integer(as.character(lgbm_train$Sepsis_Result))

# Create dataset for LightGBM
train_datalgbm <- lgb.Dataset(data = as.matrix(lgbm_train[, -10]), label =
lgbm_train$Sepsis_Result)

# Define parameters for LightGBM model
params <- list(
```

```
  objective = "binary",  # Binary classification
  metric = "binary_error",  # Error rate as evaluation metric
  num_leaves = 10,
  learning_rate = 0.1,
  num_iterations = 100
)

# Train LightGBM model
model4 <- lgb.train(params, train_datalgbm)

# Make predictions
predictions3 <- predict(model4, as.matrix(lgbm_test))

# Calculate accuracy for LightGBM
predicted_labels <- as.integer(predictions3 > 0.5)
accuracy <- mean(predicted_labels == Test_data$Sepsis_Result)
cat("Accuracy:", accuracy * 100, "\n")

# Compute confusion matrix for LightGBM
bp1 <- factor(predicted_labels, levels = c(0, 1))
ts <- factor(Test_data$Sepsis_Result, levels = c(0, 1))
lgb_cm <- confusionMatrix(bp1, reference = ts)
lgb_precision <- lgb_cm$byClass["Precision"]
lgb_recall <- lgb_cm$byClass["Recall"]
lgb_f1 <- lgb_cm$byClass["F1"]
lgb_accuracy <- lgb_cm$overall["Accuracy"]

# Display LightGBM metrics
print("LightGBM Metrics:")
print(paste("Precision:", lgb_precision))
print(paste("Recall:", lgb_recall))
print(paste("F1 Score:", lgb_f1))
print(paste("Accuracy:", lgb_accuracy))

# Plot LightGBM predictions
plot(predictions3)

# XGBoost
library(xgboost)

# Prepare data for XGBoost
xgboost_train <- Train_data[1:10]
xgboost_test <- Test_data[1:9]

# Display structure of XGBoost train data
str(xgboost_train)

# Convert factors to numeric
xgboost_train$Gender <- as.numeric(as.character(xgboost_train$Gender))
xgboost_test$Gender <- as.numeric(as.character(xgboost_test$Gender))
xgboost_train$Sepsis_Result <- as.numeric(as.character(xgboost_train$Sepsis_Result))

# Create matrix for XGBoost
train_matrix <- xgb.DMatrix(data = as.matrix(xgboost_train[, c("Gender", "Age", "ICULOS",
"HospAdmTime", "HR", "O2Sat", "SBP", "MAP", "DBP")]),
                            label = xgboost_train$Sepsis_Result)

# Define parameters for XGBoost model
params <- list(
  objective = "binary:logistic",  # For binary classification tasks
  max_depth = 3,
  eta = 0.1
)

# Train XGBoost model
```

```
model5 <- xgboost(data = train_matrix, params = params, nrounds = 100)

# Make predictions
predictions4 <- predict(model5, newdata = xgb.DMatrix(data = as.matrix(xgboost_test)))

# Plot XGBoost predictions
plot(predictions4)

# Calculate accuracy for XGBoost
binary_predictions <- ifelse(predictions4 > 0.5, 1, 0)
correct_predictions <- binary_predictions == Test_data$Sepsis_Result
accuracy <- mean(correct_predictions) * 100

# Display XGBoost accuracy
accuracy

# Compute confusion matrix for XGBoost
bp <- factor(binary_predictions, levels = c(0, 1))
ts <- factor(Test_data$Sepsis_Result, levels = c(0, 1))
xgb_cm <- confusionMatrix(bp, reference = ts)
xgb_precision <- xgb_cm$byClass["Precision"]
xgb_recall <- xgb_cm$byClass["Recall"]
xgb_f1 <- xgb_cm$byClass["F1"]
xgb_accuracy <- xgb_cm$overall["Accuracy"]

# Display XGBoost metrics
print("XGBoost Metrics:")
print(paste("Precision:", xgb_precision))
print(paste("Recall:", xgb_recall))
print(paste("F1 Score:", xgb_f1))
print(paste("Accuracy:", xgb_accuracy))

# Random Forest
library(randomForest)

# Prepare data for Random Forest
rf_train <- Train_data[1:10]
rf_test <- Test_data[1:9]

# Display structure of Random Forest test data
str(rf_test)

# Define hyperparameter grid for Random Forest
hyper_grid <- expand.grid(
  ntree = c(100, 200, 300),
  mtry = c(2, 4, 6)
)

# Set up cross-validation control
ctrl <- trainControl(method = "cv", number = 5)

# Train Random Forest model
model6 <- randomForest(Sepsis_Result ~ ., data = rf_train ,method = "rf",
                       trControl = ctrl,
                       tuneGrid = hyper_grid)

# Make predictions
predictions5 <- predict(model6, rf_test)

# Calculate confusion matrix for Random Forest
xtab5 <- table(predictions5, Test_data$Sepsis_Result)

# Compute confusion matrix for Random Forest
cm5 <- caret::confusionMatrix(xtab5, mode = "everything", positive = "1")
```

```
# Display confusion matrix for Random Forest
cm5

# Compute accuracy for Random Forest
confusion_matrix <- table(predictions5, Test_data$Sepsis_Result)
print(confusion_matrix)
accuracy(confusion_matrix)
accuracy <- mean(predictions5 == Test_data$Sepsis_Result)
cat("Accuracy:", accuracy * 100, "\n")

# Plot Random Forest model
plot(model6)

# Ensemble method - Bagging

# Define models
models <- list(model1, model2, model3, model4, model5, model6)

# Define train control
ctrl <- trainControl(method = "boot")

# Split data into training and testing sets
training <- Train_data[1:10]
testing <- Test_data1

# Train bagging ensemble model
fit_bagging <- train(Sepsis_Result ~ ., data = training,
                     models = models,
                     trControl = ctrl)

# Plot accuracy of bagging ensemble model
plot(fit_bagging, main = "Bagging Ensemble Method Final Accuracy")

# Predict on the testing set using the bagged ensemble
predictions_bag <- predict(fit_bagging, testing)
confusion_matrix <- table(predictions_bag, Test_data$Sepsis_Result)

# Compute and print accuracy
cat("Accuracy:", accuracy(confusion_matrix))

# Compute confusion matrix
xtab6 <- table(predictions_bag, Test_data$Sepsis_Result)
cm6 <- caret::confusionMatrix(xtab6, mode = "everything", positive = "1")
cm6

# Ensemble method - Boosting

# Define number of trees and learning rate
n.trees <- 100
lr <- 0.1

# Train boosting ensemble model
fit_boosting <- train(Sepsis_Result ~ ., data = training,
                      models = models,
                      trControl = ctrl,
                      numTrees = n.trees,
                      learningRate = lr)

# Plot accuracy of boosting ensemble model
plot(fit_boosting, main = "Boosting Ensemble Method Final Accuracy")

# Predict on the testing set using the boosting ensemble
predictions_boost <- predict(fit_boosting, testing)
confusion_matrix <- table(predictions_boost, Test_data$Sepsis_Result)
```

```
# Compute and print accuracy
cat("Accuracy:", accuracy(confusion_matrix))

# Compute confusion matrix
xtab7 <- table(predictions_boost, Test_data$Sepsis_Result)
cm7 <- caret::confusionMatrix(xtab7, mode = "everything", positive = "1")
cm7

# XAI - LIME

# Import lime library
library(lime)

# Create LIME explainer
lime_explainer <- lime(rf_train, model6)

#model type defenition
model_type.randomForest <- function(x) {
  return("regression") # Or "classification" depending on your problem
}

# Summary of LIME explainer
class(lime_explainer)
summary(lime_explainer)
plot(lime_explainer$preprocess)
# Explain predictions using LIME
explanation <- explain(rf_test[1:5,], lime_explainer, n_labels = 1, n_features = 10)
plot_features(explanation)

# Compute Lime explanation
lime_explanation <- explain(
  x = rf_test,
  explainer = lime_explainer,
  n_permutations = 5000,
  dist_fun = "gower",
  kernel_width = .75,
  n_features = 10,
  feature_select = "highest_weights",
  labels = "Yes"
)

# Plot feature importance
barplot(feature_importance, main = "Feature Importance", xlab = "Features", ylab =
"Importance")

# Plot feature distribution
plot_features(feature_distribution)

# Plot distribution of feature bins
par(mfrow = c(2, 5))
for (i in seq_along(n_bins)) {
  barplot(n_bins[[i]], main = paste("Feature", i, "Distribution"), xlab = "Value", ylab =
"Density")
}

# Plot bin cuts
par(mfrow = c(2, 5))
for (i in seq_along(bin_cuts)) {
  hist(unlist(bin_cuts[[i]]), main = paste("Bin Cuts for Feature", i), xlab = "Value",
ylab = "Frequency")
}

# Plot feature importance using a bar plot
barplot(feature_weights, names.arg = feature_names, main = "Feature Importance (LIME)",
xlab = "Features", ylab = "Weight")
```

```
# SHAP explanation

# Import necessary libraries
library(explainer)
library(randomForest)

# Create SHAP explainer
shap_explainer <- explain(model6,
                          data = as.data.frame(rf_test),
                          y = Test_data$Sepsis_Result,
                          verbose = FALSE)

# Remove rows with missing values
rf_test <- rf_test[complete.cases(rf_test), ]

# Get feature labels
feature_labels <- names(rf_test)

# Explain predictions using SHAP
explanation_shap <- predict_parts(shap_explainer, type = "shap", new_observation =
rf_test[1:5, ])
plot(explanation_shap)

# Compute SHAP explanations
explanation_shap <- predict_parts(rf_explainer, type = "shap", new_observation = rf_test)
plot(explanation_shap)

# Compute Break Down explanations
explanation_break_down <- predict_parts(rf_explainer, type = "break_down", new_observation
= rf_test)
plot(explanation_break_down)
```