```r
# Step 1: Data Preparation and Exploration

install.packages("mlbench")
install.packages("caret")
install.packages("rpart")
install.packages("rpart.plot")

# Load necessary libraries
library(mlbench)
library(dplyr)
library(ggplot2)

# Load the dataset
data(PimaIndiansDiabetes2)
df <- PimaIndiansDiabetes2

# Examine data statistics
summary(df)
str(df)

# Visualize the data (example: histogram for glucose levels)
ggplot(df, aes(x = glucose)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  theme_minimal() +
  labs(title = "Glucose Level Distribution", x = "Glucose", y = "Frequency")

### Step 2: Data Preprocessing

# Handle missing values (example: remove rows with NA)
df <- na.omit(df)

# Encode categorical variables
df$diabetes <- ifelse(df$diabetes == "pos", 1, 0)

# Split data into training and testing sets
set.seed(123)
train_indices <- sample(1:nrow(df), 0.7 * nrow(df))
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]

### Step 3: Model Building - Decision Tree Classifier


# Load decision tree library
library(rpart)
library(rpart.plot)

# Initialize and train the model
decision_tree <- rpart(diabetes ~ ., data = train_data, method = "class")

# Plot the decision tree
rpart.plot(decision_tree)

# Evaluate model performance
pred <- predict(decision_tree, test_data, type = "class")
confusion_matrix <- table(Predicted = pred, Actual = test_data$diabetes)
print(confusion_matrix)

# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))


### Step 4: Hyperparameter Tuning Using Grid Search and Cross-Validation
```

```r
# Set up grid search for hyperparameters
library(caret)
control <- trainControl(method = "cv", number = 5)
grid <- expand.grid(cp = seq(0.01, 0.1, by = 0.01))

# Train model with cross-validation
tuned_model <- train(diabetes ~ ., data = train_data, method = "rpart",
                     trControl = control, tuneGrid = grid)

# Display best hyperparameters
print(tuned_model$bestTune)

# Retrain the model with best parameters
final_model <- rpart(diabetes ~ ., data = train_data, method = "class",
                     control = rpart.control(cp = tuned_model$bestTune$cp))

# Evaluate optimized model
final_pred <- predict(final_model, test_data, type = "class")
final_conf_matrix <- table(Predicted = final_pred, Actual = test_data$diabetes)
final_accuracy <- sum(diag(final_conf_matrix)) / sum(final_conf_matrix)
print(paste("Optimized Accuracy:", final_accuracy))
```