1. Load the Titanic dataset using read.csv(), explore it with str(), summary(), and head(). Identify numerical/categorical variables and handle missing values by imputing with mean, median, or mode. Convert categorical data into factors, scale numerical features, engineer new features, and save the preprocessed data using write.csv().

```
# Load required libraries
library(dplyr)

# 1. Load the Titanic dataset
titanic <- read.csv("titanic.csv", stringsAsFactors = FALSE)

# 2. Explore the dataset
str(titanic)
summary(titanic)
head(titanic)

# 3. Identify numerical and categorical variables
num_vars <- sapply(titanic, is.numeric)
cat_vars <- sapply(titanic, is.character)

# 4. Handle missing values
# Impute numeric columns with mean or median
titanic$Age[is.na(titanic$Age)] <- median(titanic$Age, na.rm = TRUE)
titanic$Fare[is.na(titanic$Fare)] <- median(titanic$Fare, na.rm = TRUE)

# Impute categorical columns with mode
get_mode <- function(v) {
  uniqv <- na.omit(unique(v))
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

titanic$Embarked[titanic$Embarked == ""] <- get_mode(titanic$Embarked)

# 5. Convert categorical variables into factors
titanic$Sex <- as.factor(titanic$Sex)
titanic$Embarked <- as.factor(titanic$Embarked)
titanic$Pclass <- as.factor(titanic$Pclass)
titanic$Survived <- as.factor(titanic$Survived)

# 6. Scale numerical features
# Example: scale Age and Fare
titanic$Age <- scale(titanic$Age)
titanic$Fare <- scale(titanic$Fare)

# 7. Engineer new features
```

```
# FamilySize = SibSp + Parch + 1 (include self)
titanic$FamilySize <- titanic$SibSp + titanic$Parch + 1

# Title extracted from Name
titanic$Title <- sub(".*, (.*?)\\..*", "\\1", titanic$Name)
titanic$Title <- as.factor(titanic$Title)

# 8. Save the preprocessed data
write.csv(titanic, "titanic_preprocessed.csv", row.names = FALSE)
```

2. Use Esquisse to visualize the mtcars dataset: 1.Create a histogram/density plot for mpg (Miles per Gallon). 2. Compare the number of automatic vs. manual cars using a bar plot of am.3.Show the relationship between cyl (Cylinders) and hp (Horsepower) using a box/scatter plot. 4.Visualize mpg vs. wt (Weight) with color for cyl. 5.Create a bar plot showing car counts by gear (Transmission gears). Export and save the R code.

```
install.packages("esquisse")  # if not already installed

library(esquisse)

esquisse::esquisse(data = mtcars)
```

 Create Visualizations:

- **Histogram/Density of mpg**
- **Bar plot for am (Automatic vs Manual)**
- **Box or Scatter plot: cyl vs hp**
- **Scatter plot: mpg vs wt with color by cyl**
- **Bar plot: count of gear**

 Click “Export” in Esquisse to copy or save the generated R code.

💾 Saving the R Code

To save this code to an R script file:

r

CopyEdit

```
writeLines(c(
  "# Esquisse Exported Code for mtcars Visualizations",
  "",
  "# Load libraries",
```

```
  "library(ggplot2)",
  "library(dplyr)",
  "",
  "# Data preprocessing",
  "mtcars <- mtcars %>%",
  "  mutate(",
  "    am = factor(am, labels = c('Automatic', 'Manual')),\n    cyl = factor(cyl),\n    gear = factor(gear)",
  "  )",
  "",
  "# Plots follow here...",
  "# (Paste in the plotting code above)"
), "mtcars_visualizations.R")
```

3. Load a mtcars dataset, identify numerical and categorical variables, and visualize distributions using box plots, histograms, and violin plots. Use scatter plots with trend lines to analyze relationships. Create facets, bar plots, and heatmaps to explore patterns and correlations. Provide insights on distributions, outliers, and variable relationships.

```
# ==============================
# 1. Load Libraries
# ==============================
install.packages("ggplot2")
install.packages("dplyr")
install.packages("ggcorrplot")
install.packages("reshape2")

library(ggplot2)
library(dplyr)
library(ggcorrplot)
library(reshape2)
```

```r
# ==============================
# 2. Load & Prepare mtcars Dataset
# ==============================
data("mtcars")

# Convert appropriate variables to factors
mtcars <- mtcars %>%
  mutate(
    cyl = factor(cyl),
    gear = factor(gear),
    am = factor(am, labels = c("Automatic", "Manual")),
    vs = factor(vs, labels = c("V-shaped", "Straight")),
    carb = factor(carb)
  )

# Identify numerical and categorical variables
num_vars <- sapply(mtcars, is.numeric)
cat_vars <- sapply(mtcars, is.factor)

# ==============================
# 3. Distribution Visualizations
# ==============================

# Boxplot: MPG by Cylinders
ggplot(mtcars, aes(x = cyl, y = mpg, fill = cyl)) +
  geom_boxplot() +
  labs(title = "MPG by Number of Cylinders", x = "Cylinders", y = "MPG") +
  theme_minimal()

# Histogram: Horsepower
```

```r
ggplot(mtcars, aes(x = hp)) +
  geom_histogram(binwidth = 20, fill = "#69b3a2", color = "white") +
  labs(title = "Distribution of Horsepower", x = "Horsepower", y = "Count") +
  theme_minimal()

# Violin Plot: MPG by Gear
ggplot(mtcars, aes(x = gear, y = mpg, fill = gear)) +
  geom_violin(trim = FALSE) +
  labs(title = "Violin Plot of MPG by Gear", x = "Gear", y = "MPG") +
  theme_minimal()

# ==============================
# 4. Scatter Plots with Trend Lines
# ==============================

# Scatter: MPG vs Weight colored by Transmission
ggplot(mtcars, aes(x = wt, y = mpg, color = am)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "MPG vs Weight by Transmission", x = "Weight (1000 lbs)", y = "MPG") +
  theme_minimal()

# ==============================
# 5. Faceted Plot
# ==============================

# HP vs MPG faceted by Gear
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(color = gear), size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
```

```
  facet_wrap(~ gear) +
  labs(title = "HP vs MPG Faceted by Gears") +
  theme_minimal()

# ================================
# 6. Bar Plot
# ================================

# Car counts by Carburetor
ggplot(mtcars, aes(x = carb, fill = carb)) +
  geom_bar() +
  labs(title = "Car Counts by Carburetor", x = "Carburetor", y = "Count") +
  theme_minimal()

# ================================
# 7. Heatmap: Correlation Matrix
# ================================

# Correlation matrix for numerical vars
cor_matrix <- cor(mtcars[, num_vars])

# Heatmap
ggcorrplot(cor_matrix,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           colors = c("#6D9EC1", "white", "#E46726"),
           title = "Correlation Heatmap")
```

```
# ==============================
# 8. Insights Output
# ==============================
cat("\n--- Insights Summary ---\n")
cat("• MPG tends to be lower for cars with more cylinders (box plot).\n")
cat("• Horsepower shows right-skewed distribution, centered around 100-150 hp (histogram).\n")
cat("• Manual transmission cars show slightly better MPG on average (scatter plot).\n")
cat("• Violin plot shows 4-gear cars tend to be more fuel efficient.\n")
cat("• Scatter plots indicate negative correlation between weight and MPG.\n")
cat("• Strong correlations: wt negatively with mpg, hp positively with wt (heatmap).\n")
cat("• 4-cylinder cars dominate the dataset, and more gears usually mean better efficiency.\n")
```

4. Develop a Shiny app using the iris dataset with three features: (1) Select a numerical and categorical variable to display summary statistics. (2) Choose two numerical variables for a scatter plot, colored by a categorical variable. (3) Generate a box plot for a selected numerical and categorical variable.

```
# Load required packages
library(shiny)
library(ggplot2)
library(dplyr)

# Define UI
ui <- fluidPage(
  titlePanel("Iris Dataset Explorer"),

  sidebarLayout(
    sidebarPanel(
      # Feature 1: Summary Stats
      h4("1. Summary Statistics"),
```

```
    selectInput("num_var_summary", "Select Numerical Variable:",
                choices = names(iris)[sapply(iris, is.numeric)]),
    selectInput("cat_var_summary", "Select Categorical Variable:",
                choices = names(iris)[sapply(iris, is.factor)]),

    hr(),

    # Feature 2: Scatter Plot
    h4("2. Scatter Plot"),
    selectInput("x_var", "X-axis Variable:",
                choices = names(iris)[sapply(iris, is.numeric)], selected = "Sepal.Length"),
    selectInput("y_var", "Y-axis Variable:",
                choices = names(iris)[sapply(iris, is.numeric)], selected = "Petal.Length"),
    selectInput("color_var", "Color By (Categorical):",
                choices = names(iris)[sapply(iris, is.factor)], selected = "Species"),

    hr(),

    # Feature 3: Box Plot
    h4("3. Box Plot"),
    selectInput("num_var_box", "Select Numerical Variable:",
                choices = names(iris)[sapply(iris, is.numeric)], selected = "Petal.Length"),
    selectInput("cat_var_box", "Select Categorical Variable:",
                choices = names(iris)[sapply(iris, is.factor)], selected = "Species")
  ),

  mainPanel(
```

```
      tabsetPanel(
        tabPanel("Summary Statistics",
                 verbatimTextOutput("summary_output")),

        tabPanel("Scatter Plot",
                 plotOutput("scatter_plot")),

        tabPanel("Box Plot",
                 plotOutput("box_plot"))
      )
    )
  )
)

# Define server logic
server <- function(input, output) {

  # Summary Statistics
  output$summary_output <- renderPrint({
    iris %>%
      group_by(.data[[input$cat_var_summary]]) %>%
      summarise(Mean = mean(.data[[input$num_var_summary]], na.rm = TRUE),
                Median = median(.data[[input$num_var_summary]], na.rm = TRUE),
                SD = sd(.data[[input$num_var_summary]], na.rm = TRUE),
                Min = min(.data[[input$num_var_summary]], na.rm = TRUE),
                Max = max(.data[[input$num_var_summary]], na.rm = TRUE),
                .groups = 'drop')
  })
```

```
  # Scatter Plot
  output$scatter_plot <- renderPlot({
    ggplot(iris, aes_string(x = input$x_var, y = input$y_var, color = input$color_var)) +
      geom_point(size = 3, alpha = 0.7) +
      theme_minimal() +
      labs(title = "Scatter Plot", x = input$x_var, y = input$y_var)
  })

  # Box Plot
  output$box_plot <- renderPlot({
    ggplot(iris, aes_string(x = input$cat_var_box, y = input$num_var_box, fill = input$cat_var_box)) +
      geom_boxplot() +
      theme_minimal() +
      labs(title = "Box Plot", x = input$cat_var_box, y = input$num_var_box)
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

5. Analyze outliers in the mtcars dataset using IQR and Z-score methods. Compute probabilities of selecting outlier cars based on mpg, hp, and wt. Explore conditional probabilities, independence, and expected outliers. Examine distribution shapes, skewness, correlations, and compare different outlier detection methods to assess consistency and insights.

```
install.packages("dplyr")
install.packages("ggplot2")
install.packages("e1071")
```

```
install.packages("gridExtra")

library(dplyr)
library(ggplot2)
library(e1071)
library(gridExtra)

data("mtcars")
df <- mtcars

iqr_outliers <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR_val <- Q3 - Q1
  lower <- Q1 - 1.5 * IQR_val
  upper <- Q3 + 1.5 * IQR_val
  return(x < lower | x > upper)
}

zscore_outliers <- function(x, threshold = 2) {
  z <- (x - mean(x)) / sd(x)
  return(abs(z) > threshold)
}

df <- df %>%
  mutate(outlier_mpg_iqr = iqr_outliers(mpg),
         outlier_hp_iqr = iqr_outliers(hp),
         outlier_wt_iqr = iqr_outliers(wt),
         outlier_mpg_z = zscore_outliers(mpg),
         outlier_hp_z = zscore_outliers(hp),
```

```
        outlier_wt_z = zscore_outliers(wt))

p1 <- ggplot(df, aes(x = mpg)) +
  geom_boxplot(fill = "lightblue") +
  ggtitle("MPG - Boxplot (IQR Method)")

p2 <- ggplot(df, aes(x = hp)) +
  geom_boxplot(fill = "lightgreen") +
  ggtitle("HP - Boxplot (IQR Method)")

p3 <- ggplot(df, aes(x = wt)) +
  geom_boxplot(fill = "lightcoral") +
  ggtitle("WT - Boxplot (IQR Method)")

grid.arrange(p1, p2, p3, ncol = 3)

n <- nrow(df)
p_mpg_outlier <- sum(df$outlier_mpg_iqr) / n
p_hp_outlier <- sum(df$outlier_hp_iqr) / n
p_wt_outlier <- sum(df$outlier_wt_iqr) / n

cat("P(MPG Outlier) =", p_mpg_outlier, "\n")
cat("P(HP Outlier)  =", p_hp_outlier, "\n")
cat("P(WT Outlier)  =", p_wt_outlier, "\n")

p_joint <- sum(df$outlier_mpg_iqr & df$outlier_hp_iqr) / n
p_hp_given_mpg <- p_joint / p_mpg_outlier

cat("P(MPG & HP Outlier) =", p_joint, "\n")
cat("P(HP | MPG Outlier) =", p_hp_given_mpg, "\n")
```

```
cat("P(MPG)*P(HP) =", p_mpg_outlier * p_hp_outlier, "\n")


cat("Skewness:\n")
cat("MPG:", skewness(df$mpg), "\n")
cat("HP:", skewness(df$hp), "\n")
cat("WT:", skewness(df$wt), "\n")


cor_matrix <- cor(df[, sapply(df, is.numeric)])
print(round(cor_matrix, 2))
heatmap(cor_matrix, main = "Correlation Matrix", col = colorRampPalette(c("skyblue", "white", "tomato"))(20))


compare_outliers <- function(var) {
  iqr_ids <- which(df[[paste0("outlier_", var, "_iqr")]])
  z_ids <- which(df[[paste0("outlier_", var, "_z")]])
  both <- intersect(iqr_ids, z_ids)
  cat(sprintf("Outliers in %s:\n", var))
  cat("  - IQR method:", length(iqr_ids), "\n")
  cat("  - Z-score method:", length(z_ids), "\n")
  cat("  - Both methods:", length(both), "\n\n")
}


compare_outliers("mpg")
compare_outliers("hp")
compare_outliers("wt")
```

6. Perform correlation analysis on the mtcars dataset to evaluate relationships between mpg and other features using Pearson and Spearman methods. Identify key influencing variables and visualize insights using heatmaps, scatter plots, box plots, bar plots, and clustering dendrograms, ensuring a comprehensive understanding of mpg dependencies and trends.

```
install.packages("corrplot")
install.packages("ggplot2")
install.packages("pheatmap")
install.packages("dendextend")
install.packages("cluster")
install.packages("factoextra")

library(corrplot)
library(ggplot2)
library(pheatmap)
library(dendextend)
library(cluster)
library(factoextra)

data("mtcars")
df <- mtcars

# Pearson correlation
cor_pearson <- cor(df, method = "pearson")

# Spearman correlation
cor_spearman <- cor(df, method = "spearman")

# Heatmaps
corrplot(cor_pearson, method = "color", title = "Pearson Correlation", mar = c(0,0,2,0), addCoef.col = "black")
corrplot(cor_spearman, method = "color", title = "Spearman Correlation", mar = c(0,0,2,0), addCoef.col = "black")

# Sorted correlations with mpg
```

```
mpg_corr_pearson <- sort(cor_pearson["mpg", ], decreasing = TRUE)
mpg_corr_spearman <- sort(cor_spearman["mpg", ], decreasing = TRUE)
print(mpg_corr_pearson)
print(mpg_corr_spearman)

# Scatter plots for top correlations
top_vars <- names(sort(abs(cor_pearson["mpg", -1]), decreasing = TRUE))[1:3]
for (var in top_vars) {
  p <- ggplot(df, aes_string(x = var, y = "mpg")) +
    geom_point(color = "steelblue", size = 3) +
    geom_smooth(method = "lm", se = FALSE, color = "red") +
    theme_minimal() + ggtitle(paste("MPG vs", var))
  print(p)
}

# Box plots for categorical features converted to factors
df$cyl <- as.factor(df$cyl)
df$gear <- as.factor(df$gear)
df$am <- as.factor(df$am)
df$carb <- as.factor(df$carb)
df$vs <- as.factor(df$vs)

ggplot(df, aes(x = cyl, y = mpg, fill = cyl)) +
  geom_boxplot() +
  ggtitle("MPG by Cylinder") +
  theme_minimal()

ggplot(df, aes(x = gear, y = mpg, fill = gear)) +
  geom_boxplot() +
  ggtitle("MPG by Gear") +
```

```
  theme_minimal()

ggplot(df, aes(x = am, y = mpg, fill = am)) +
  geom_boxplot() +
  ggtitle("MPG by Transmission (0 = Auto, 1 = Manual)") +
  theme_minimal()

# Bar plot: average mpg by carb
df %>%
  group_by(carb) %>%
  summarise(avg_mpg = mean(mpg)) %>%
  ggplot(aes(x = carb, y = avg_mpg, fill = carb)) +
  geom_bar(stat = "identity") +
  ggtitle("Average MPG by Carb") +
  theme_minimal()

# Clustering dendrogram
dist_mat <- dist(scale(df))
hc <- hclust(dist_mat)
dend <- as.dendrogram(hc)
plot(color_branches(dend, k = 4), main = "Hierarchical Clustering of Cars")
```

7. Analyze sales trends over time using scatter plots and regression models. Fit linear and quadratic regression, compare their R-squared and RMSE values, and interpret results. Identify the best-fitting model and predict sales for the next 6 months (Months: 61-66) to assess future trends and decision-making insights.

```
# Sample Sales Data (Monthly)
set.seed(123)
months <- 1:60
sales <- 100 + 2 * months + rnorm(60, mean = 0, sd = 10)
df <- data.frame(month = months, sales = sales)
```

```
# Scatter Plot
library(ggplot2)
ggplot(df, aes(x = month, y = sales)) +
  geom_point(color = "blue") +
  ggtitle("Monthly Sales Scatter Plot") +
  theme_minimal()

# Linear Regression
model_linear <- lm(sales ~ month, data = df)
summary(model_linear)

# Quadratic Regression
model_quad <- lm(sales ~ month + I(month^2), data = df)
summary(model_quad)

# R-squared and RMSE Comparison
r2_linear <- summary(model_linear)$r.squared
r2_quad <- summary(model_quad)$r.squared

rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}
rmse_linear <- rmse(df$sales, predict(model_linear, df))
rmse_quad <- rmse(df$sales, predict(model_quad, df))

cat("Linear R2:", r2_linear, " | RMSE:", rmse_linear, "\n")
cat("Quadratic R2:", r2_quad, " | RMSE:", rmse_quad, "\n")

# Plot with Regression Lines
```

```
df$pred_linear <- predict(model_linear, df)
df$pred_quad <- predict(model_quad, df)

ggplot(df, aes(x = month, y = sales)) +
  geom_point(color = "black") +
  geom_line(aes(y = pred_linear), color = "blue", size = 1) +
  geom_line(aes(y = pred_quad), color = "red", size = 1, linetype = "dashed") +
  ggtitle("Sales with Linear (blue) & Quadratic (red) Regression") +
  theme_minimal()

# Predict Next 6 Months (Months 61–66)
future_months <- data.frame(month = 61:66)
future_months$pred_linear <- predict(model_linear, future_months)
future_months$pred_quad <- predict(model_quad, future_months)

print(future_months)
```

8. A medical researcher is studying the relationship between various blood test parameters, such as glucose levels, cholesterol, and blood pressure, to identify potential risk factors for diabetes. Compute the Pearson and Spearman correlation coefficients for a given dataset and analyze the relationships between variables. Explain how correlation matrices assist in feature selection and why highly correlated features might lead to redundancy in predictive models. Using linear algebra, describe how the correlation matrix can be decomposed using eigenvalues and eigenvectors to understand the data structure.

```
# Sample Dataset (replace with your actual data)
set.seed(123)
glucose <- rnorm(100, mean = 100, sd = 20)
cholesterol <- 0.7 * glucose + rnorm(100, mean = 180, sd = 30)
blood_pressure_systolic <- rnorm(100, mean = 120, sd = 15)
blood_pressure_diastolic <- 0.6 * blood_pressure_systolic + rnorm(100, mean = 80, sd = 10)
age <- sample(20:70, 100, replace = TRUE)
```

```
data <- data.frame(glucose, cholesterol, blood_pressure_systolic, blood_pressure_diastolic, age)

# 1. Compute Pearson Correlation Matrix
pearson_corr_matrix <- cor(data, method = "pearson")
print("Pearson Correlation Matrix:")
print(pearson_corr_matrix)

# 2. Compute Spearman Correlation Matrix
spearman_corr_matrix <- cor(data, method = "spearman")
print("\nSpearman Correlation Matrix:")
print(spearman_corr_matrix)

# 3. Analyze Relationships (example interpretations)
cat("\nAnalysis of Relationships (Example):\n")
if (pearson_corr_matrix["glucose", "cholesterol"] > 0.5) {
  cat("Glucose and cholesterol show a moderate positive linear relationship.\n")
} else {
  cat("Glucose and cholesterol do not show a strong linear relationship.\n")
}

if (spearman_corr_matrix["blood_pressure_systolic", "blood_pressure_diastolic"] > 0.7) {
  cat("Systolic and diastolic blood pressure show a strong positive monotonic relationship.\n")
} else {
  cat("Systolic and diastolic blood pressure do not show a strong monotonic relationship.\n")
}

# 4. Feature Selection Considerations (based on correlation)
cat("\nFeature Selection Considerations:\n")
highly_correlated_pairs <- which(abs(pearson_corr_matrix) > 0.8 & row(pearson_corr_matrix) < col(pearson_corr_matrix), arr.ind = TRUE)
if (nrow(highly_correlated_pairs) > 0) {
  cat("Highly correlated pairs (Pearson > 0.8):\n")
  for (i in 1:nrow(highly_correlated_pairs)) {
    var1 <- rownames(pearson_corr_matrix)[highly_correlated_pairs[i, 1]]
    var2 <- colnames(pearson_corr_matrix)[highly_correlated_pairs[i, 2]]
    cat(paste("-", var1, "and", var2, "\n"))
```

```
   cat("Consider removing one of these variables to avoid redundancy in predictive models.\n")
  }
} else {
  cat("No highly correlated pairs (Pearson > 0.8) found.\n")
}

# 5. Eigenvalue Decomposition of the Correlation Matrix (using Pearson)
eigen_decomposition <- eigen(pearson_corr_matrix)
eigenvalues <- eigen_decomposition$values
eigenvectors <- eigen_decomposition$vectors

print("\nEigenvalues of the Pearson Correlation Matrix:")
print(eigenvalues)

print("\nEigenvectors of the Pearson Correlation Matrix:")
print(eigenvectors)

# 6. Understanding Data Structure (example interpretation)
cat("\nUnderstanding Data Structure (Example):\n")
cat("The eigenvalues represent the variance explained by each principal component.\n")
cat("The eigenvectors represent the direction of these principal components in the original variable space.\n")
cat(paste("The first principal component explains", round(eigenvalues[1] / sum(eigenvalues) * 100, 2), "% of the total variance.\n"))
cat("By examining the loadings (elements of the eigenvectors), you can see which original variables contribute most to each principal component.\n")
```

9. Analyze the Heart Disease dataset by exploring its structure, target variable, and missing values. Visualize correlations, split data, and train a Logistic Regression model. Evaluate using accuracy, precision, recall, and F1-score. Apply PCA for dimensionality reduction and assess if performance improves for better heart disease prediction.

```
# Install necessary packages

install.packages("caTools")

install.packages("caret")

install.packages("e1071")

install.packages("pROC")

install.packages("ggplot2")
```

```
install.packages("corrplot")
install.packages("FactoMineR")

library(caTools)
library(caret)
library(e1071)
library(pROC)
library(ggplot2)
library(corrplot)
library(FactoMineR)

# Load the Heart Disease dataset (UCI dataset or any available source)
# For example, using the 'UCI Heart Disease' dataset in the `mlbench` package
# Alternatively, load your dataset using read.csv() if not pre-installed

# Download and load the dataset from a CSV (example)
df <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart.dat", sep = ";", header = FALSE)

# Rename columns based on dataset description
colnames(df) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal", "target")

# Check the structure and summary
str(df)
summary(df)

# Check for missing values
sum(is.na(df))

# Visualize correlations using a correlation matrix
```

```
cor_matrix <- cor(df[, -14])  # Exclude target variable for correlation matrix
corrplot(cor_matrix, method = "color", title = "Correlation Matrix", mar = c(0, 0, 2, 0))

# Split data into training and testing sets (70% training, 30% testing)
set.seed(123)
split <- sample.split(df$target, SplitRatio = 0.7)
train_data <- subset(df, split == TRUE)
test_data <- subset(df, split == FALSE)

# Logistic Regression Model (Without PCA)
model_logistic <- glm(target ~ ., data = train_data, family = binomial)
summary(model_logistic)

# Predictions on test set
pred_logistic <- predict(model_logistic, test_data, type = "response")
pred_class <- ifelse(pred_logistic > 0.5, 1, 0)

# Evaluation Metrics: Accuracy, Precision, Recall, F1-Score
conf_matrix <- confusionMatrix(as.factor(pred_class), as.factor(test_data$target))
cat("Accuracy: ", conf_matrix$overall["Accuracy"], "\n")
cat("Precision: ", conf_matrix$byClass["Pos Pred Value"], "\n")
cat("Recall: ", conf_matrix$byClass["Sensitivity"], "\n")
cat("F1-Score: ", conf_matrix$byClass["F1"], "\n")

# Apply PCA for dimensionality reduction
pca <- prcomp(train_data[, -14], scale = TRUE)  # Exclude target variable for PCA
summary(pca)
```

```
# Variance explained by each component
cat("Variance Explained by Each Component: \n")
print(summary(pca))

# Visualize PCA components
pca_data <- data.frame(pca$x)
ggplot(pca_data, aes(PC1, PC2)) +
  geom_point(aes(color = train_data$target)) +
  ggtitle("PCA - First Two Components") +
  theme_minimal()

# Create a new dataset with reduced dimensions (using the first two principal components)
train_data_pca <- data.frame(target = train_data$target, pca$x[, 1:2])
test_data_pca <- predict(pca, test_data[, -14])[, 1:2]

# Logistic Regression Model with PCA-transformed data
model_logistic_pca <- glm(target ~ PC1 + PC2, data = train_data_pca, family = binomial)
summary(model_logistic_pca)

# Predictions with PCA
pred_logistic_pca <- predict(model_logistic_pca, test_data_pca, type = "response")
pred_class_pca <- ifelse(pred_logistic_pca > 0.5, 1, 0)

# Evaluation Metrics for PCA Model
conf_matrix_pca <- confusionMatrix(as.factor(pred_class_pca), as.factor(test_data$target))
cat("Accuracy (PCA): ", conf_matrix_pca$overall["Accuracy"], "\n")
cat("Precision (PCA): ", conf_matrix_pca$byClass["Pos Pred Value"], "\n")
cat("Recall (PCA): ", conf_matrix_pca$byClass["Sensitivity"], "\n")
```

```
cat("F1-Score (PCA): ", conf_matrix_pca$byClass["F1"], "\n")
```

10. A real estate company aims to predict house prices based on factors such as square footage, number of bedrooms, and location. Formulate the problem as a multiple linear regression equation and express it in matrix form. Using the normal equation, calculate the regression coefficients to fit the model. Discuss how multicollinearity among predictor variables affects the reliability of the regression model and explain how Principal Component Regression (PCR) can be used to address multicollinearity issues while improving predictive accuracy.

```
# Simulating a real estate dataset

set.seed(123)

n <- 100

square_footage <- rnorm(n, mean = 2000, sd = 500)

bedrooms <- rnorm(n, mean = 3, sd = 1)

location <- factor(sample(c('A', 'B', 'C'), n, replace = TRUE))


# Encode location as a dummy variable

location_dummies <- model.matrix(~location - 1)


# House prices (dependent variable) based on these predictors

price <- 50000 + 150 * square_footage + 10000 * bedrooms + ifelse(location == 'A', 10000,
ifelse(location == 'B', 20000, 30000)) + rnorm(n, mean = 0, sd = 50000)


# Creating the design matrix X

X <- cbind(1, square_footage, bedrooms, location_dummies)


# Convert the response variable into a column vector Y

Y <- price


# Normal Equation: beta = (X^T X)^-1 X^T Y

beta_hat <- solve(t(X) %*% X) %*% t(X) %*% Y


# Display the estimated regression coefficients
```

```
print(beta_hat)
```