**1.Explore ANY DATASET 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().**

```
# Install necessary packages (only once)
install.packages("readxl")
install.packages("caret")

# Load libraries
library(readxl)
library(caret)

# Read Excel file (make sure the file is in your working directory)
data <- read_excel("Food_Delivery_Times.csv")

# Explore the dataset
str(data)
summary(data)
head(data)

# Handle missing values
Mode <- function(x) {
  ux <- na.omit(unique(x))
  ux[which.max(tabulate(match(x, ux)))]
}

for (col in names(data)) {
  if (is.numeric(data[[col]])) {
    data[[col]][is.na(data[[col]])] <- median(data[[col]], na.rm = TRUE)
  } else {
    data[[col]][is.na(data[[col]])] <- Mode(data[[col]])
    data[[col]] <- as.factor(data[[col]])
  }
}

# Encode categorical variables
dummies <- dummyVars(" ~ .", data = data)
data_encoded <- data.frame(predict(dummies, newdata = data))

# Scale numeric values
data_scaled <- as.data.frame(scale(data_encoded))

# Save to CSV
write.csv(data_scaled, "preprocessed_data.csv", row.names = FALSE)
```

**OUTPUT:**

```
tibble [1,000 × 9] (S3: tbl_df/tbl/data.frame)
 $ Order_ID              : num [1:1000] 522 738 741 661 412 679 627 514 860 137 ...
 $ Distance_km           : num [1:1000] 7.93 16.42 9.52 7.44 19.03 ...
 $ Weather               : chr [1:1000] "Windy" "Clear" "Foggy" "Rainy" ...
 $ Traffic_Level         : chr [1:1000] "Low" "Medium" "Low" "Medium" ...
 $ Time_of_Day           : chr [1:1000] "Afternoon" "Evening" "Night" "Afternoon" ...
 $ Vehicle_Type          : chr [1:1000] "Scooter" "Bike" "Scooter" "Scooter" ...
 $ Preparation_Time_min  : num [1:1000] 12 20 28 5 16 8 12 5 20 29 ...
 $ Courier_Experience_yrs: num [1:1000] 1 2 1 1 5 9 1 6 6 1 ...
 $ Delivery_Time_min     : num [1:1000] 43 84 59 37 68 57 49 46 35 73 ...
    Order_ID       Distance_km       Weather          Traffic_Level
 Min.   :   1.0   Min.   : 0.590   Length:1000        Length:1000
 1st Qu.: 250.8   1st Qu.: 5.105   Class :character   Class :character
 Median : 500.5   Median :10.190   Mode  :character   Mode  :character
 Mean   : 500.5   Mean   :10.060
 3rd Qu.: 750.2   3rd Qu.:15.018
 Max.   :1000.0   Max.   :19.990

 Time_of_Day        Vehicle_Type       Preparation_Time_min
 Length:1000        Length:1000        Min.   : 5.00
 Class :character   Class :character   1st Qu.:11.00
 Mode  :character   Mode  :character   Median :17.00
                                       Mean   :16.98
                                       3rd Qu.:23.00
                                       Max.   :29.00

 Courier_Experience_yrs Delivery_Time_min
 Min.   :0.000          Min.   :  8.00
 1st Qu.:2.000          1st Qu.: 41.00
 Median :5.000          Median : 55.50
 Mean   :4.579          Mean   : 56.73
 3rd Qu.:7.000          3rd Qu.: 71.00
 Max.   :9.000          Max.   :153.00
 NA's   :30
```

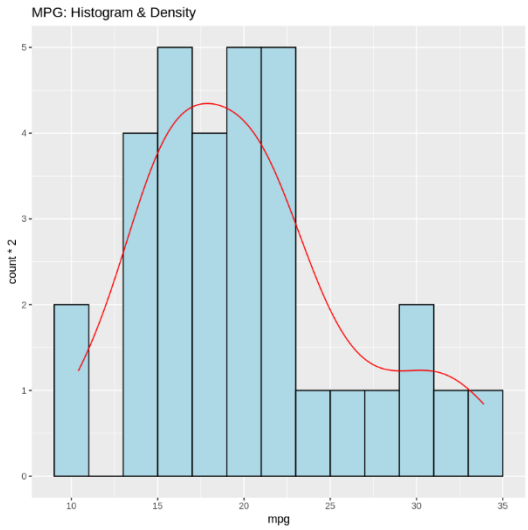
| Order_ID | Distance_km | Weather | Traffic_Level | Time_of_Day | Vehicle_Type | Preparation_Time_min | Courier_Experience_yrs | Delivery_Time_min |
|---|---|---|---|---|---|---|---|---|
| <dbl> | <dbl> | <chr> | <chr> | <chr> | <chr> | <dbl> | <dbl> | <dbl> |
| 522 | 7.93 | Windy | Low | Afternoon | Scooter | 12 | 1 | 43 |
| 738 | 16.42 | Clear | Medium | Evening | Bike | 20 | 2 | 84 |
| 741 | 9.52 | Foggy | Low | Night | Scooter | 28 | 1 | 59 |
| 661 | 7.44 | Rainy | Medium | Afternoon | Scooter | 5 | 1 | 37 |
| 412 | 19.03 | Clear | Low | Morning | Bike | 16 | 5 | 68 |
| 679 | 19.40 | Clear | Low | Evening | Scooter | 8 | 9 | 57 |

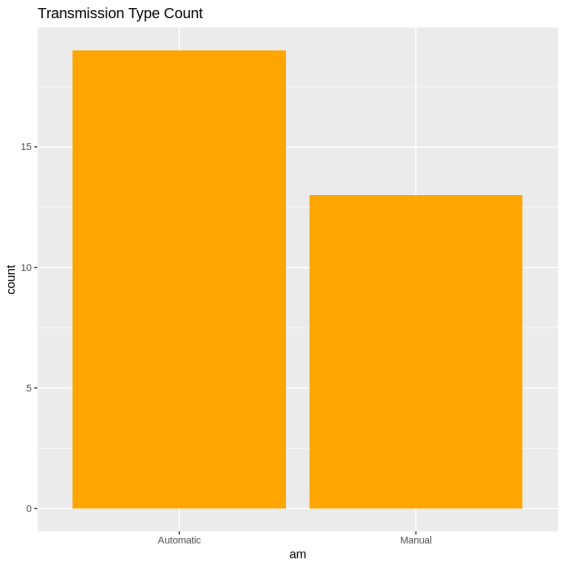
**2. .A.Create a histogram/density plot for mpg (Miles per Gallon). 2. Compare the number of automatic vs. manual cars using a bar plot of am.3.Show the relationship between cyl (Cylinders) and hp (Horsepower) using a box/scatter plot. 4.Visualize mpg vs. wt (Weight) with color for cyl. 5.Create a bar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

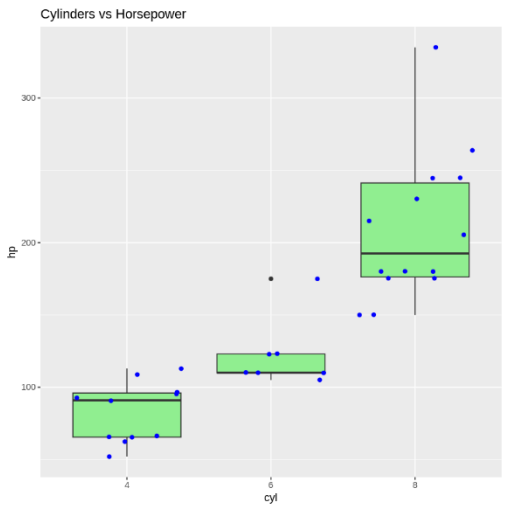
```
library(ggplot2)


# Convert to factors

mtcars$am <- factor(mtcars$am, labels = c("Automatic", "Manual"))

mtcars$cyl <- factor(mtcars$cyl)
```

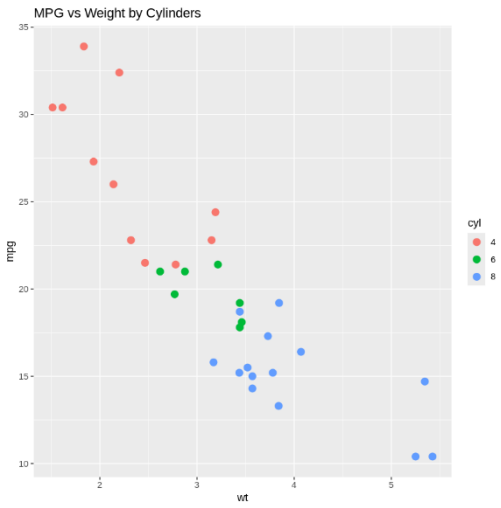
```
mtcars$gear <- factor(mtcars$gear)

# 1. Histogram & density for mpg
ggplot(mtcars, aes(mpg)) +
  geom_histogram(binwidth = 2, fill = "lightblue", color = "black") +
  geom_density(aes(y = ..count.. * 2), color = "red") +
  ggtitle("MPG: Histogram & Density")

# 2. Bar plot: Automatic vs Manual
ggplot(mtcars, aes(am)) +
  geom_bar(fill = "orange") +
  ggtitle("Transmission Type Count")

# 3. Box + scatter: Cylinders vs Horsepower
ggplot(mtcars, aes(cyl, hp)) +
  geom_boxplot(fill = "lightgreen") +
  geom_jitter(color = "blue") +
  ggtitle("Cylinders vs Horsepower")

# 4. Scatter: MPG vs Weight, color by Cylinders
ggplot(mtcars, aes(wt, mpg, color = cyl)) +
  geom_point(size = 3) +
  ggtitle("MPG vs Weight by Cylinders")

# 5. Bar plot: Count by Gear
ggplot(mtcars, aes(gear)) +
  geom_bar(fill = "purple") +
  ggtitle("Car Count by Gears")
```

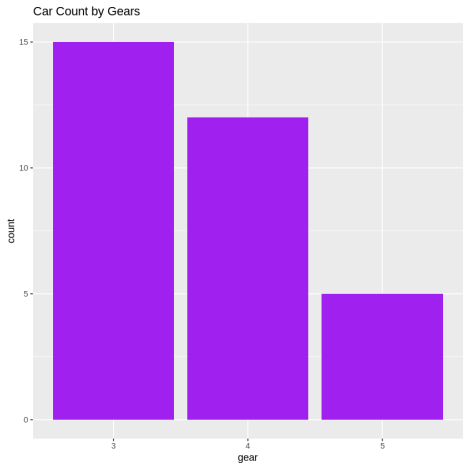
# OUTPUT:


MPG: Histogram & Density
count * 2
mpg


Transmission Type Count
count
Automatic
Manual
am


Cylinders vs Horsepower
hp
cyl


MPG vs Weight by Cylinders
mpg
wt
cyl
4
6
8

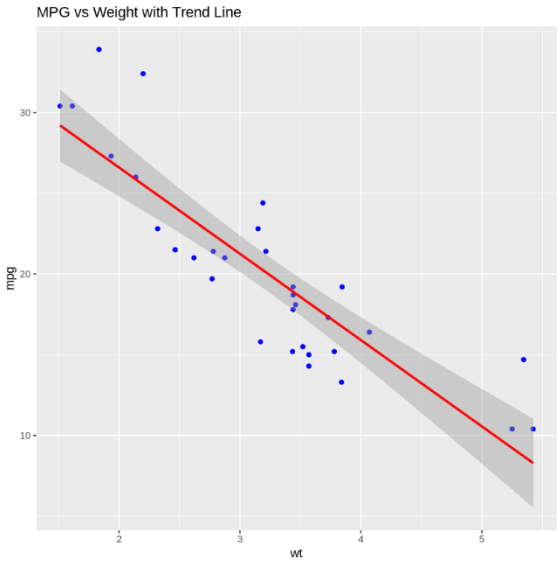

Car Count by Gears
count
gear

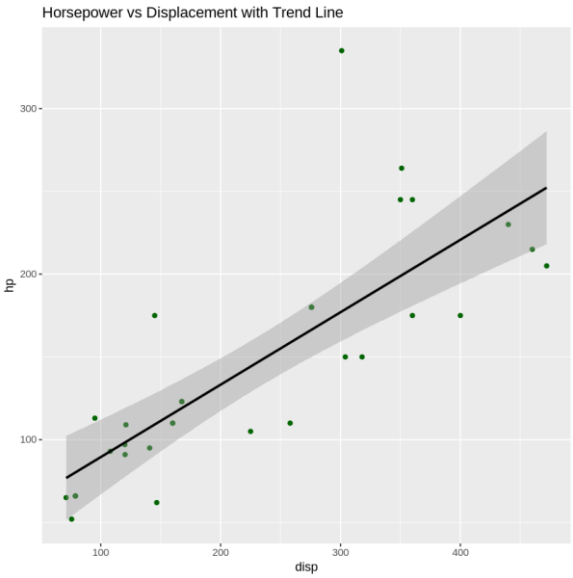
**3. Use scatter plots with trend lines to analyze relationships.**

```
library(ggplot2)

# Use mtcars dataset
data("mtcars")

# Convert relevant columns to factors
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$am <- factor(mtcars$am, labels = c("Automatic", "Manual"))

# 1. MPG vs Weight with trend line
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  ggtitle("MPG vs Weight with Trend Line")

# 2. HP vs Displacement with trend line
ggplot(mtcars, aes(x = disp, y = hp)) +
  geom_point(color = "darkgreen") +
  geom_smooth(method = "lm", se = TRUE, color = "black") +
  ggtitle("Horsepower vs Displacement with Trend Line")

# 3. MPG vs Horsepower, colored by Transmission
ggplot(mtcars, aes(x = hp, y = mpg, color = am)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = FALSE) +
  ggtitle("MPG vs Horsepower by Transmission Type")
```

**OUTPUT:**

**`geom_smooth()` using formula = 'y ~ x'**

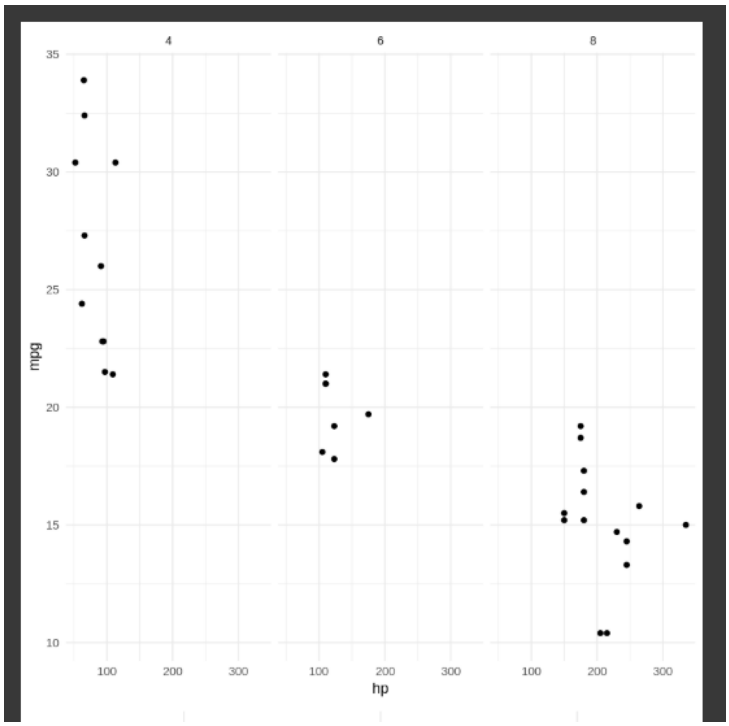


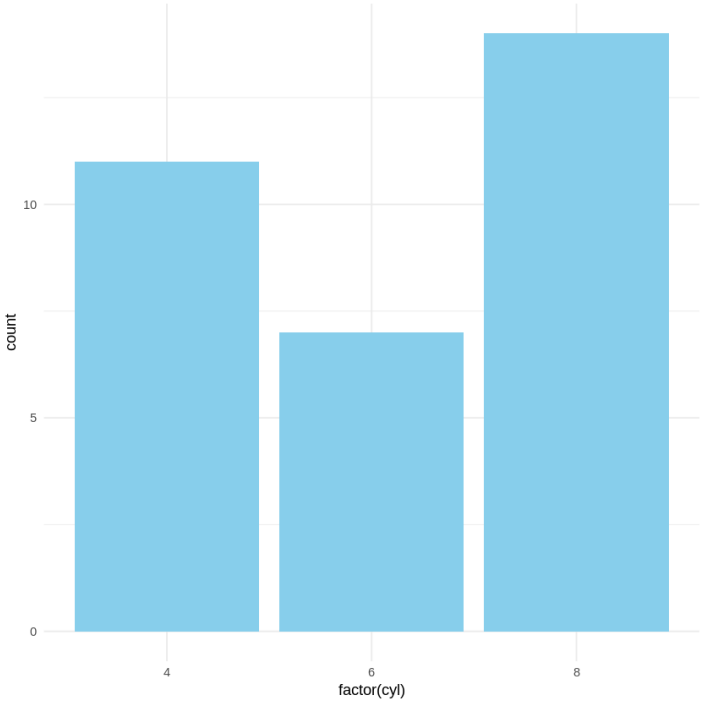



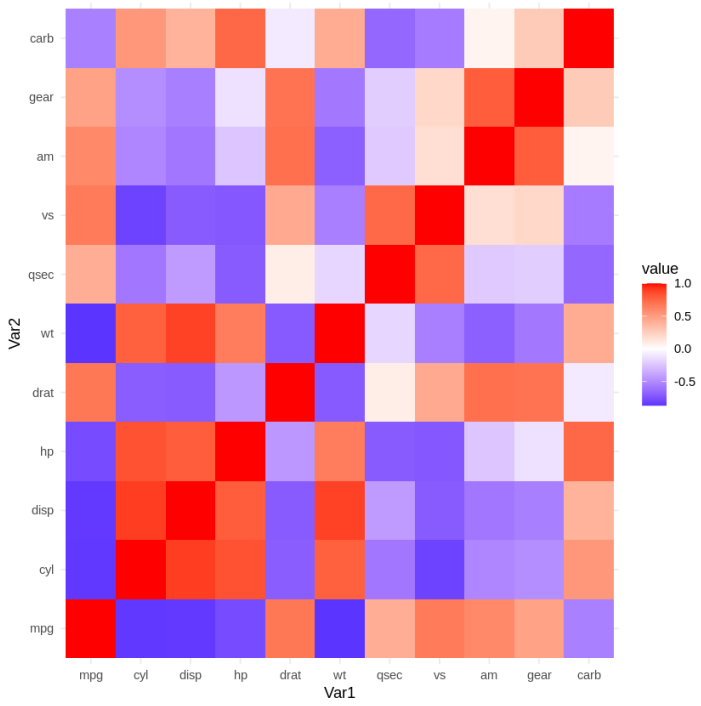

**4. Create facets, bar plots, and heatmaps to explore patterns and correlations. Provide insights on distributions, outliers, and variable relationships.**

```
install.packages(c("ggplot2", "reshape2"))
library(ggplot2)
library(reshape2)

# Sample data
data(mtcars)

# 1. Facet Plot (mpg vs hp by cyl)
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  facet_wrap(~ cyl) +
  theme_minimal()

# 2. Bar Plot (Count of cars by cyl)
ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar(fill = "skyblue") +
  theme_minimal()

# 3. Heatmap (Correlation matrix)
cor_matrix <- cor(mtcars)
cor_melted <- melt(cor_matrix)
ggplot(cor_melted, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", midpoint = 0) +
  theme_minimal()
```

**OUTPUT:**


4
6
8
35
30
25
20
15
10
mpg
100
200
300
hp


10
5
0
count
4
6
8
factor(cyl)




**5.Create shiny app with following features**

**(1) to display summary statistics. (2) Choose two numerical variables for a scatter plot. (3) Generate a box plot**

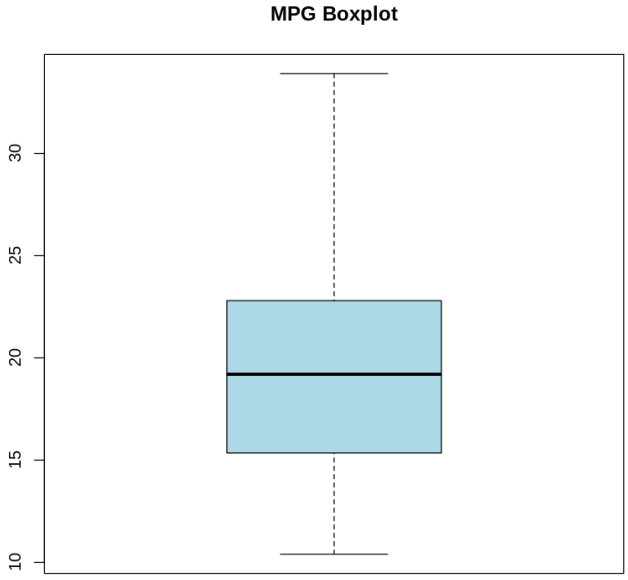
**6. Explore conditional probabilities, independence, and expected outliers. Examine distribution shapes, skewness, correlations, and compare different outlier detection methods to assess consistency and insights.**

```
# Install required package

if (!require(e1071)) install.packages("e1071", dependencies = TRUE)


# Load libraries

library(e1071)

library(ggplot2)


# Dataset

data(mtcars)


# 1. Conditional Probability: P(gear=4 | cyl=4)
```

```
p <- sum(mtcars$cyl == 4 & mtcars$gear == 4) / sum(mtcars$cyl == 4)
cat("P(gear=4 | cyl=4):", round(p, 2), "\n")

# 2. Independence Check
pa <- mean(mtcars$gear == 4)
pb <- mean(mtcars$cyl == 4)
pab <- mean(mtcars$gear == 4 & mtcars$cyl == 4)
cat("Independence? P(A)*P(B):", round(pa * pb, 3), "vs P(A∩B):", round(pab, 3), "\n")

# 3. Skewness
cat("Skewness of mpg:", round(skewness(mtcars$mpg), 2), "\n")

# 4. Correlation
cat("Correlation (mpg vs hp):", round(cor(mtcars$mpg, mtcars$hp), 2), "\n")

# 5. Boxplot for outliers
boxplot(mtcars$mpg, main = "MPG Boxplot", col = "lightblue")
```

**OUTPUT:**

```
P(gear=4 | cyl=4): 0.73
Independence? P(A)*P(B): 0.129 vs P(A∩B): 0.25
Skewness of mpg: 0.61
Correlation (mpg vs hp): -0.78
```



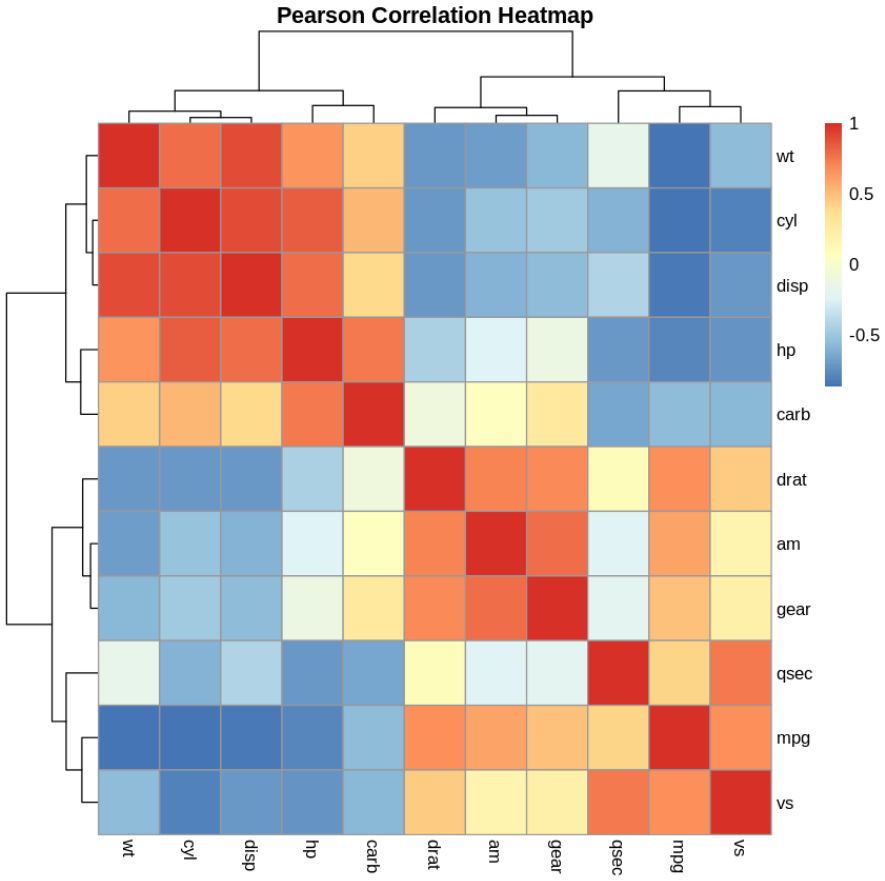

**7. Using Pearson and Spearman methods. Identify key influencing variables and visualize insights using heatmaps, scatter plots, box plots, bar plots, and clustering dendrograms**

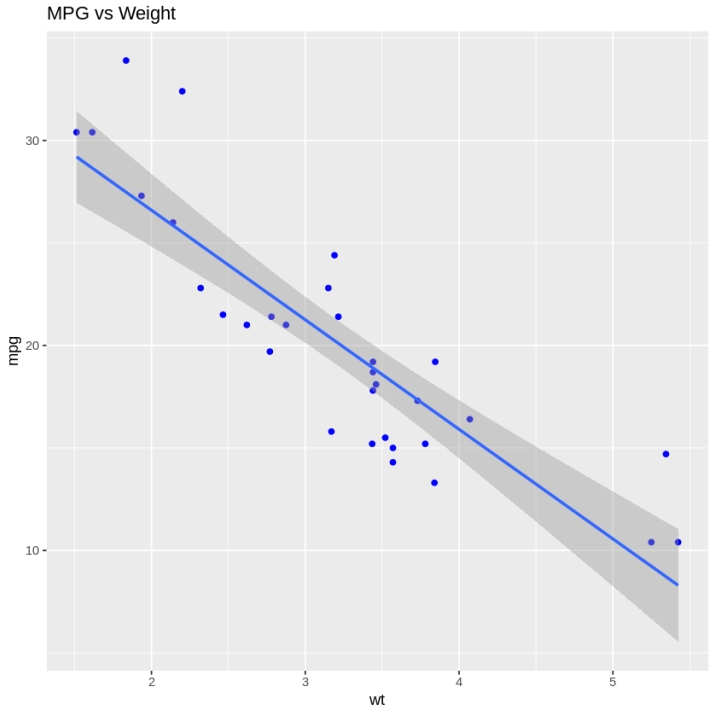
```
# Install and load required packages
if (!require(pheatmap)) install.packages("pheatmap", dependencies=TRUE)
library(ggplot2)
library(pheatmap)

data(mtcars)

# 1. Pearson & Spearman correlations
pear_cor <- cor(mtcars, method = "pearson")
spear_cor <- cor(mtcars, method = "spearman")
print("Top Pearson correlations with mpg:")
print(sort(pear_cor["mpg",], decreasing = TRUE))
print("Top Spearman correlations with mpg:")
print(sort(spear_cor["mpg",], decreasing = TRUE))
```

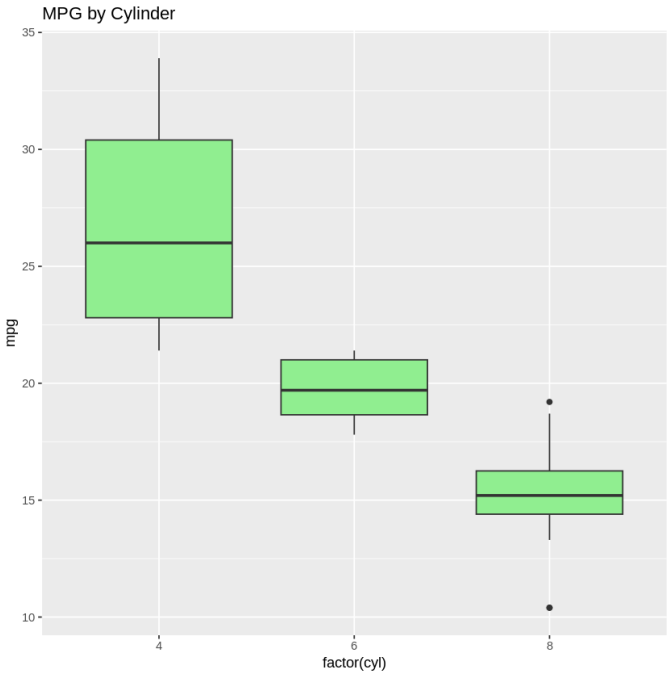
```
# 2. Heatmap
pheatmap(pear_cor, main = "Pearson Correlation Heatmap")

# 3. Scatter plot: mpg vs wt (strongest negative corr)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm") +
  ggtitle("MPG vs Weight")

# 4. Box plot: mpg by cylinder
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot(fill = "lightgreen") +
  ggtitle("MPG by Cylinder")

# 5. Bar plot: mean mpg by gear
gear_mean <- aggregate(mpg ~ gear, data = mtcars, mean)
ggplot(gear_mean, aes(x = factor(gear), y = mpg)) +
  geom_bar(stat = "identity", fill = "orange") +
  ggtitle("Mean MPG by Gear")

# 6. Clustering Dendrogram
d <- dist(scale(mtcars))        # scale and compute distance
hc <- hclust(d)
plot(hc, main = "Clustering Dendrogram", xlab = "", sub = "")
```

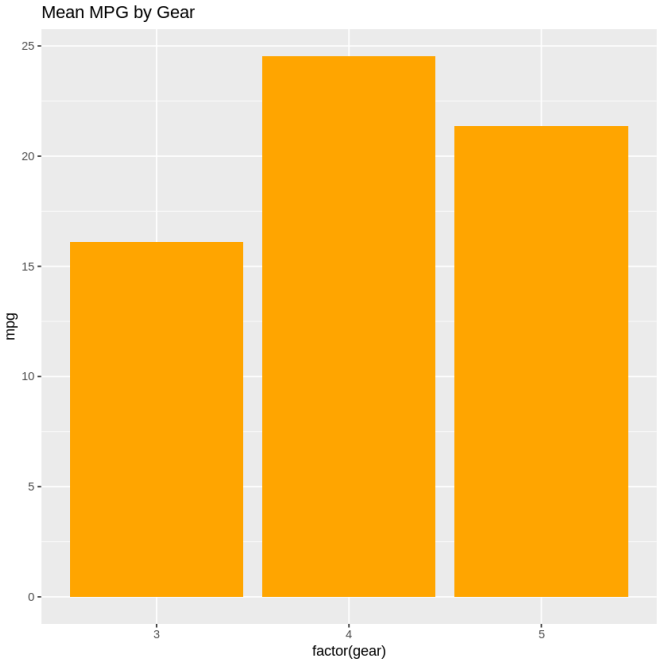
**OUTPUT:**

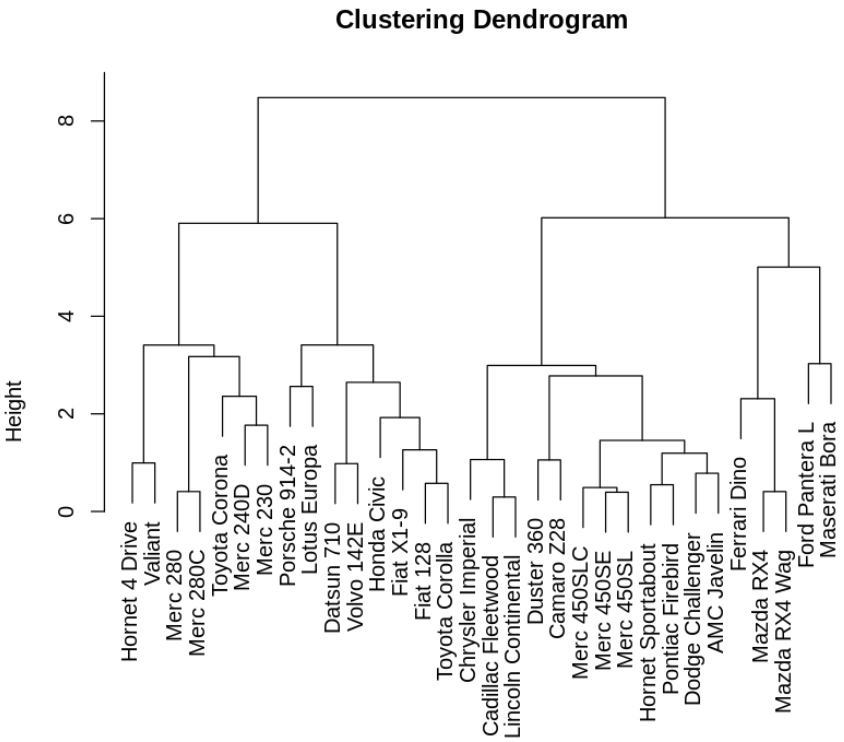
```
[1] "Top Pearson correlations with mpg:"
      mpg       drat         vs         am       gear       qsec       carb
 1.0000000  0.6811719  0.6640389  0.5998324  0.4802848  0.4186840 -0.5509251
       hp       disp        cyl         wt
-0.7761684 -0.8475514 -0.8521620 -0.8676594
[1] "Top Spearman correlations with mpg:"
      mpg         vs       drat         am       gear       qsec       carb
 1.0000000  0.7065968  0.6514555  0.5620057  0.5427816  0.4669358 -0.6574976
       wt         hp       disp        cyl
-0.8864220 -0.8946646 -0.9088824 -0.9108013
`geom_smooth()` using formula = 'y ~ x'
```




MPG vs Weight
mpg
wt
30
20
10
2
3
4
5


MPG by Cylinder
mpg
factor(cyl)
35
30
25
20
15
10
4
6
8


Mean MPG by Gear
mpg
25
20
15
10
5
0
3
4
5
factor(gear)

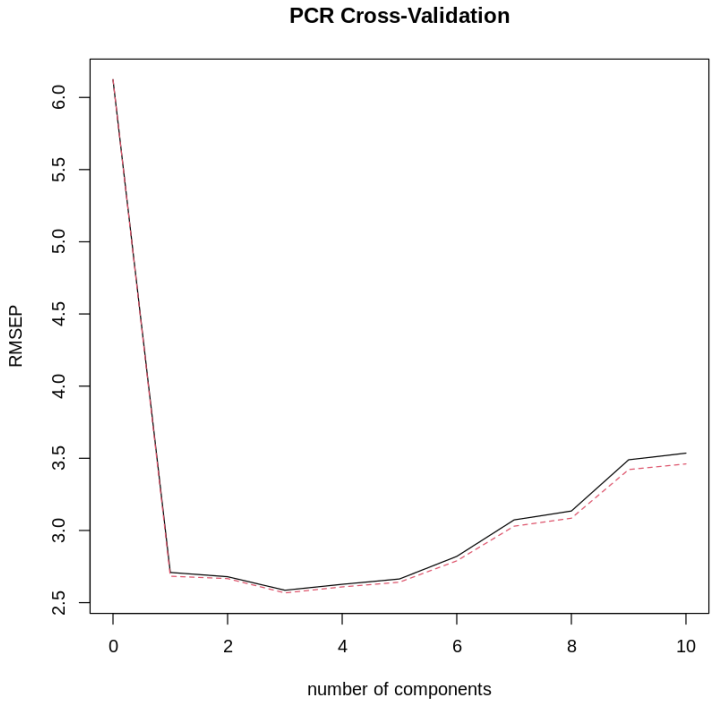

Clustering Dendrogram
Height
0
2
4
6
8
Hornet 4 Drive
Valiant
Merc 280
Merc 280C
Toyota Corona
Merc 240D
Merc 230
Porsche 914-2
Lotus Europa
Datsun 710
Volvo 142E
Honda Civic
Fiat X1-9
Fiat 128
Toyota Corolla
Chrysler Imperial
Cadillac Fleetwood
Lincoln Continental
Duster 360
Camaro Z28
Merc 450SLC
Merc 450SE
Merc 450SL
Hornet Sportabout
Pontiac Firebird
Dodge Challenger
AMC Javelin
Ferrari Dino
Mazda RX4
Mazda RX4 Wag
Ford Pantera L
Maserati Bora

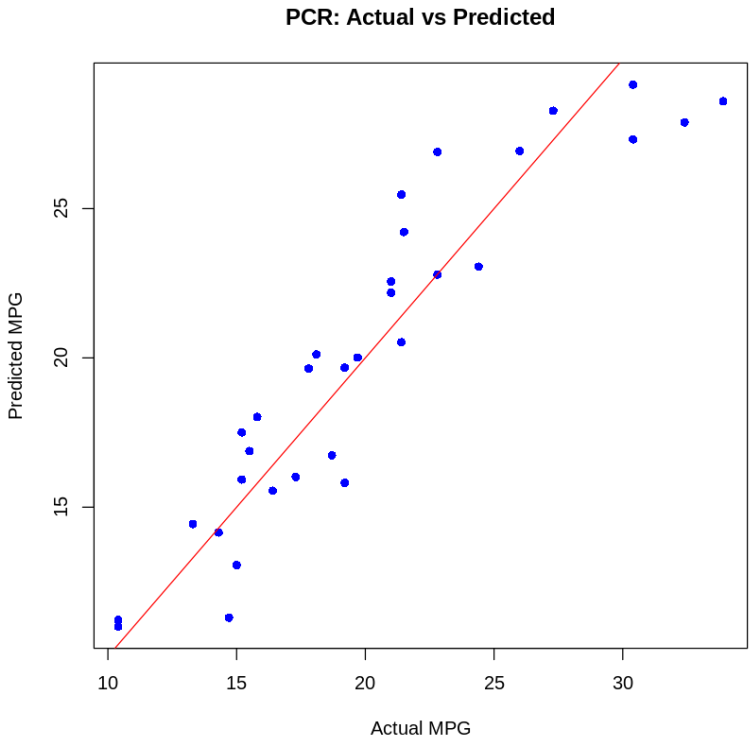
**8. explain how Principal Component Regression (PCR) can be used to address multicollinearity issues while improving predictive accuracy.**

```
# Install required package
if (!require(pls)) install.packages("pls", dependencies = TRUE)

# Load the library
library(pls)

# Load and prepare data
data(mtcars)
X <- scale(mtcars[, -1])  # All predictors, scaled
Y <- mtcars$mpg           # Response variable

# Perform PCR with cross-validation
pcr_model <- pcr(Y ~ X, validation = "CV")

# Summary of PCR model
summary(pcr_model)

# Plot RMSE vs number of components (shows optimal component count)
validationplot(pcr_model, val.type = "RMSEP", main = "PCR Cross-Validation")

# Predict using optimal number of components (e.g., 3)
predicted <- predict(pcr_model, ncomp = 3)

# Compare predicted vs actual
plot(Y, predicted, col = "blue", pch = 16,
     xlab = "Actual MPG", ylab = "Predicted MPG",
     main = "PCR: Actual vs Predicted")
abline(a = 0, b = 1, col = "red")
```

**OUTPUT:**

```
Data:   X dimension: 32 10
        Y dimension: 32 1
Fit method: svdpc
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           6.123    2.709    2.680    2.586    2.627    2.664    2.821
adjCV        6.123    2.683    2.666    2.567    2.609    2.642    2.789
       7 comps  8 comps  9 comps  10 comps
CV       3.073    3.135    3.490     3.536
adjCV    3.030    3.085    3.422     3.461

TRAINING: % variance explained
   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X    57.60    84.10    90.07    92.77    94.99    97.09    98.42    99.23
Y    82.53    82.63    85.40    85.41    85.47    85.56    85.58    85.85
   9 comps  10 comps
X    99.76     100.0
Y    86.09      86.9
```






**9. Fit linear and quadratic regression, compare their R-squared and RMSE values, and interpret results. Identify the best-fitting model and predict sales**

```
# Install and load required packages
if (!require(Metrics)) install.packages("Metrics", dependencies = TRUE)
library(Metrics)

# Simulate example data
set.seed(123)
Advertising <- seq(1, 100, by = 2)
Sales <- 5 + 0.8 * Advertising - 0.01 * Advertising^2 + rnorm(length(Advertising), sd = 5)
data <- data.frame(Advertising, Sales)

# Linear regression model
linear_model <- lm(Sales ~ Advertising, data = data)

# Quadratic regression model
quad_model <- lm(Sales ~ Advertising + I(Advertising^2), data = data)
```

```
# Predictions
predict_linear <- predict(linear_model)
predict_quad <- predict(quad_model)

# R-squared
r2_linear <- summary(linear_model)$r.squared
r2_quad <- summary(quad_model)$r.squared

# RMSE
rmse_linear <- rmse(data$Sales, predict_linear)
rmse_quad <- rmse(data$Sales, predict_quad)

# Compare models
cat("Linear Model: R² =", round(r2_linear, 3), ", RMSE =", round(rmse_linear, 3), "\n")
cat("Quadratic Model: R² =", round(r2_quad, 3), ", RMSE =", round(rmse_quad, 3), "\n")

# Predict sales for new advertising budgets
new_data <- data.frame(Advertising = c(30, 60, 90))
predicted_sales <- predict(quad_model, newdata = new_data)
print(data.frame(new_data, Predicted_Sales = round(predicted_sales, 2)))
```

**OUTPUT:**

```
Linear Model: R² = 0.332 , RMSE = 8.333
Quadratic Model: R² = 0.8 , RMSE = 4.558
  Advertising Predicted_Sales
1          30           19.97
2          60           16.68
3          90           -3.48
```

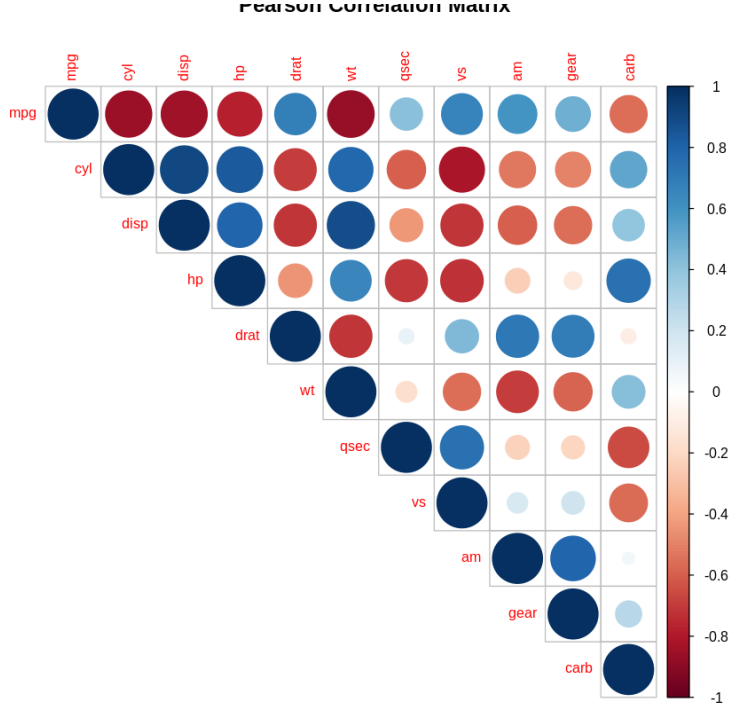
**10. Compute the Pearson and Spearman correlation coefficients for a given dataset and analyze the relationships between variables. Explain how correlation matrices assist in feature selection and why highly correlated features might lead to redundancy in predictive models.**

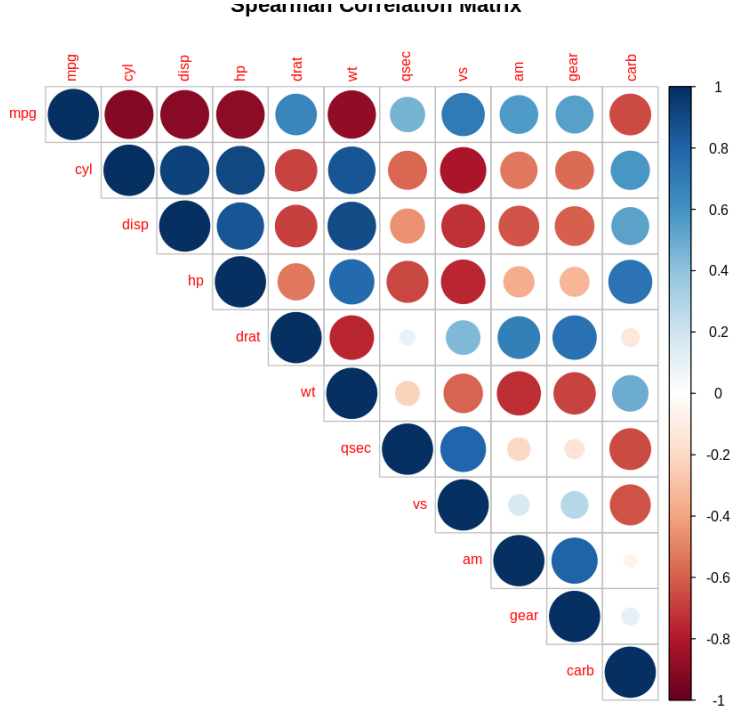
```
# Load required package
if (!require(corrplot)) install.packages("corrplot", dependencies = TRUE)
library(corrplot)

# Example dataset: mtcars
data(mtcars)

# Pearson correlation matrix
pearson_corr <- cor(mtcars, method = "pearson")
cat("Pearson Correlation Matrix:\n")
print(pearson_corr)

# Spearman correlation matrix
spearman_corr <- cor(mtcars, method = "spearman")
cat("\nSpearman Correlation Matrix:\n")
print(spearman_corr)

# Plot the Pearson correlation matrix
corrplot(pearson_corr, method = "circle", type = "upper", tl.cex = 0.8, main = "Pearson Correlation Matrix")

# Plot the Spearman correlation matrix
corrplot(spearman_corr, method = "circle", type = "upper", tl.cex = 0.8, main = "Spearman Correlation Matrix")
```

**OUTPUT:**

```
Pearson Correlation Matrix:
            mpg        cyl       disp         hp        drat         wt
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
            qsec         vs          am       gear        carb
mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

```
Spearman Correlation Matrix:
            mpg        cyl       disp         hp        drat         wt
mpg   1.0000000 -0.9108013 -0.9088824 -0.8946646  0.65145546 -0.8864220
cyl  -0.9108013  1.0000000  0.9276516  0.9017909 -0.67888119  0.8577282
disp -0.9088824  0.9276516  1.0000000  0.8510426 -0.68359210  0.8977064
hp   -0.8946646  0.9017909  0.8510426  1.0000000 -0.52012499  0.7746767
drat  0.6514555 -0.6788812 -0.6835921 -0.5201250  1.00000000 -0.7503904
wt   -0.8864220  0.8577282  0.8977064  0.7746767 -0.75039041  1.0000000
qsec  0.4669358 -0.5723509 -0.4597818 -0.6666060  0.09186863 -0.2254012
vs    0.7065968 -0.8137890 -0.7236643 -0.7515934  0.44745745 -0.5870162
am    0.5620057 -0.5220712 -0.6240677 -0.3623276  0.68657079 -0.7377126
gear  0.5427816 -0.5643105 -0.5944703 -0.3314016  0.74481617 -0.6761284
carb -0.6574976  0.5800680  0.5397781  0.7333794 -0.12522294  0.4998120
            qsec         vs          am       gear        carb
mpg   0.46693575  0.7065968  0.56200569  0.5427816 -0.65749764
cyl  -0.57235095 -0.8137890 -0.52207118 -0.5643105  0.58006798
disp -0.45978176 -0.7236643 -0.62406767 -0.5944703  0.53977806
hp   -0.66660602 -0.7515934 -0.36232756 -0.3314016  0.73337937
drat  0.09186863  0.4474575  0.68657079  0.7448162 -0.12522294
wt   -0.22540120 -0.5870162 -0.73771259 -0.6761284  0.49981205
qsec  1.00000000  0.7915715 -0.20333211 -0.1481997 -0.65871814
vs    0.79157148  1.0000000  0.16834512  0.2826617 -0.63369482
am   -0.20333211  0.1683451  1.00000000  0.8076880 -0.06436525
gear -0.14819967  0.2826617  0.80768800  1.0000000  0.11488698
carb -0.65871814 -0.6336948 -0.06436525  0.1148870  1.00000000
```




**11.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**

# Install and load necessary libraries

if (!require(caret)) install.packages("caret")

```
if (!require(caTools)) install.packages("caTools")
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(MASS)) install.packages("MASS")
if (!require(reshape2)) install.packages("reshape2")  # For melt function
library(caret)
library(caTools)
library(ggplot2)
library(MASS)
library(reshape2)  # For melt function

# Load and preprocess data (iris dataset)
data(iris)
iris$Species <- ifelse(iris$Species == "setosa", 1, 0)  # Binary classification

# Visualize correlations
cor_matrix <- cor(iris[, 1:4])  # Calculate correlation matrix
cor_matrix_melted <- melt(cor_matrix)  # Melt correlation matrix for heatmap visualization
ggplot(cor_matrix_melted, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  labs(title = "Correlation Matrix Heatmap") +
  theme_minimal()

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

# Train Logistic Regression model
model <- glm(Species ~ ., data = train_data, family = "binomial")
```

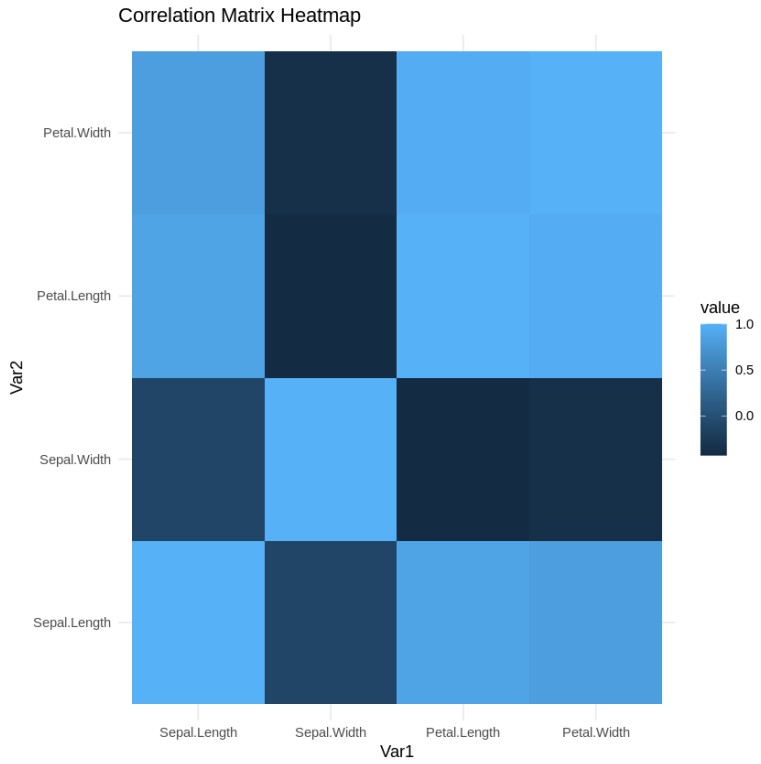
```
pred <- predict(model, test_data, type = "response")
pred_class <- ifelse(pred > 0.5, 1, 0)

# Evaluate model performance
conf_matrix <- confusionMatrix(as.factor(pred_class), as.factor(test_data$Species))
print(conf_matrix)

# Apply PCA for dimensionality reduction
pca <- prcomp(train_data[, 1:4], center = TRUE, scale. = TRUE)
train_pca <- data.frame(pca$x)
train_pca$Species <- train_data$Species
test_pca <- predict(pca, test_data[, 1:4])

# Train Logistic Regression on PCA data
model_pca <- glm(Species ~ ., data = train_pca, family = "binomial")
pred_pca <- predict(model_pca, data.frame(test_pca), type = "response")
pred_class_pca <- ifelse(pred_pca > 0.5, 1, 0)

# Evaluate PCA model performance
conf_matrix_pca <- confusionMatrix(as.factor(pred_class_pca), as.factor(test_data$Species))
print(conf_matrix_pca)
```

**OUTPUT:**

```
[1] "Correlation Matrix:"
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

```
          Reference
Prediction  0  1
         0 30  0
         1  0 15

               Accuracy : 1
                 95% CI : (0.9213, 1)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 1.191e-08

                  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 : 0.6667
         Detection Rate : 0.6667
   Detection Prevalence : 0.6667
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```


```
          Reference
Prediction  0  1
         0 30  0
         1  0 15

               Accuracy : 1
                 95% CI : (0.9213, 1)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 1.191e-08

                  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 : 0.6667
         Detection Rate : 0.6667
   Detection Prevalence : 0.6667
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```


**12. 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.**

```
# Install required packages
if (!require(pls)) install.packages("pls")
library(pls)

# --- 1. Simulate data with multicollinearity ---
set.seed(123)
X1 <- rnorm(50)
X2 <- X1 * 0.95 + rnorm(50, 0, 0.1)  # Highly correlated with X1
Y <- 4 + 3 * X1 - 2 * X2 + rnorm(50)

# --- 2. Matrix formulation of Multiple Linear Regression ---
X <- cbind(1, X1, X2)    # Add intercept column
y <- as.matrix(Y)

# Normal Equation: β̂ = (X'X)^(-1) X'y
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
colnames(beta_hat) <- "Coefficients"
rownames(beta_hat) <- c("Intercept", "X1", "X2")
cat("Coefficients using Normal Equation:\n")
print(beta_hat)

# --- 3. Check multicollinearity ---
cat("\nCorrelation between X1 and X2:\n")
print(cor(X1, X2))  # Close to 1 means high multicollinearity

# --- 4. Apply Principal Component Regression (PCR) ---
df <- data.frame(Y, X1, X2)
pcr_model <- pcr(Y ~ X1 + X2, data = df, scale = TRUE, validation = "CV")

cat("\nSummary of PCR model:\n")
```

summary(pcr_model)

# Show coefficients from PCR using top 1 principal component

cat("\nPCR Coefficients (1 component):\n")

print(coef(pcr_model, ncomp = 1))

**OUTPUT:**

```
Coefficients using Normal Equation:
          Coefficients
Intercept     3.770066
X1            4.636428
X2           -3.696081

Correlation between X1 and X2:
[1] 0.9947113

Summary of PCR model:
Data:   X dimension: 50 2
        Y dimension: 50 1
Fit method: svdpc
Number of components considered: 2

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps
CV            1.49    1.074    1.022
adjCV         1.49    1.072    1.019

TRAINING: % variance explained
   1 comps  2 comps
X    99.74   100.00
Y    49.23    56.15

PCR Coefficients (1 component):
, , 1 comps

           Y
X1 0.5182636
X2 0.5182636
```

**13 .Explore conditional probabilities, independence, and expected outliers. Examine distribution shapes, skewness, correlations**

# Install required packages

if (!require(e1071)) install.packages("e1071")

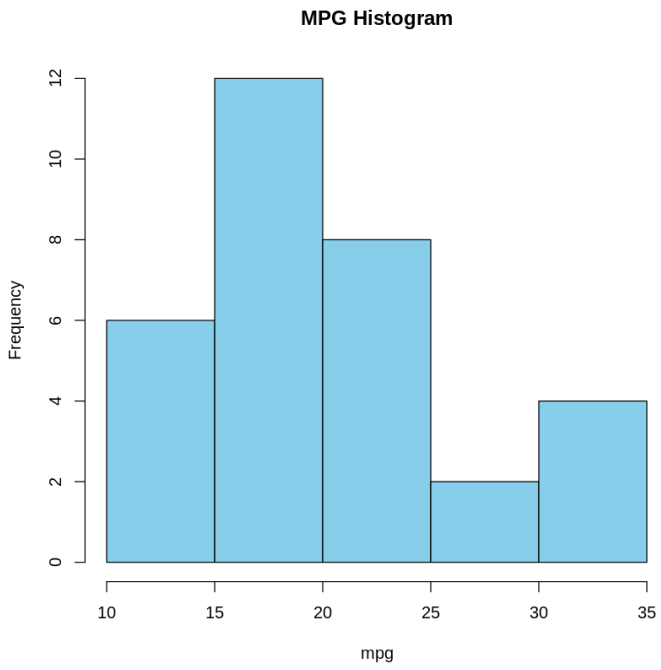
if (!require(corrplot)) install.packages("corrplot")

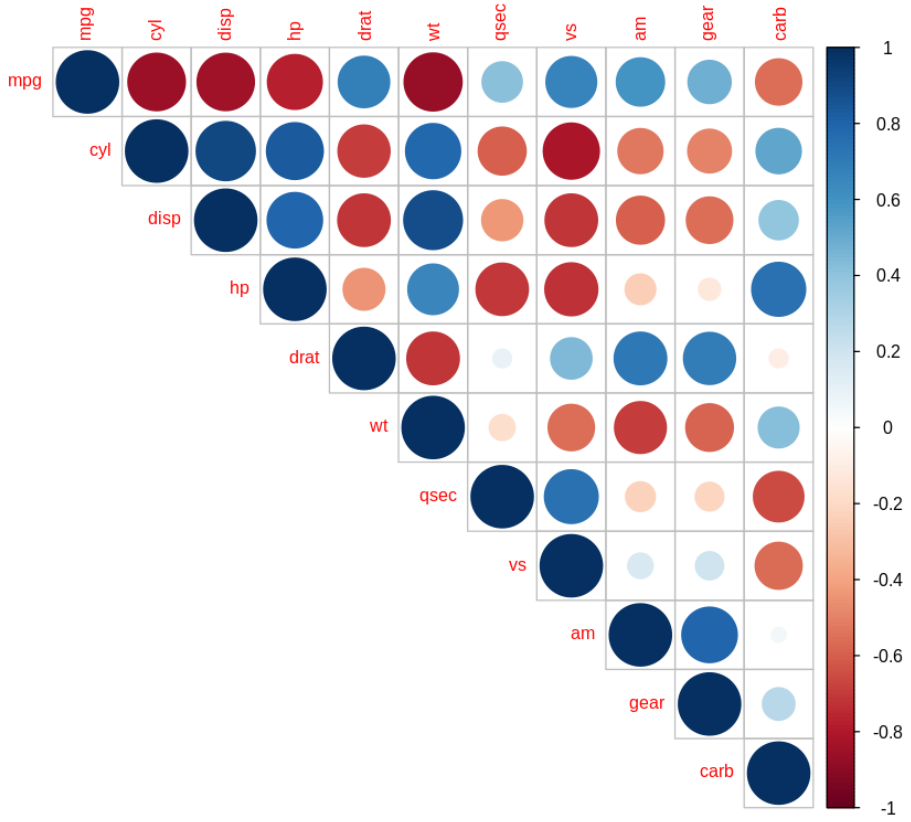
library(e1071)

```
library(corrplot)

# Load dataset
data(mtcars)

# 1. Conditional Probability: P(am = 1 | cyl = 4)
cond_prob <- sum(mtcars$cyl == 4 & mtcars$am == 1) / sum(mtcars$cyl == 4)
cat("P(am = 1 | cyl = 4):", cond_prob, "\n")

# 2. Independence check: Table of cyl vs am
independence_table <- table(mtcars$cyl, mtcars$am)
cat("\nContingency Table (cyl vs am):\n")
print(independence_table)

# 3. Outlier detection using IQR for mpg
Q1 <- quantile(mtcars$mpg, 0.25)
Q3 <- quantile(mtcars$mpg, 0.75)
IQR_val <- Q3 - Q1
outliers <- mtcars$mpg[mtcars$mpg < (Q1 - 1.5 * IQR_val) | mtcars$mpg > (Q3 + 1.5 *
IQR_val)]
cat("\nOutliers in mpg:\n")
print(outliers)

# 4. Distribution shape and skewness
hist(mtcars$mpg, col = "skyblue", main = "MPG Histogram", xlab = "mpg")
mpg_skew <- skewness(mtcars$mpg)
cat("\nSkewness of mpg:", mpg_skew, "\n")

# 5. Correlation matrix
corr_matrix <- cor(mtcars)
corrplot(corr_matrix, method = "circle", type = "upper", tl.cex = 0.8)
```

**OUTPUT:**

```
P(am = 1 | cyl = 4): 0.7272727

Contingency Table (cyl vs am):

    0  1
4   3  8
6   4  3
8  12  2

Outliers in mpg:
[1] 33.9

Skewness of mpg: 0.610655
```






**14. calculate the regression coefficients to fit the model. Discuss how multicollinearity among predictor variables affects the reliability**

```
if (!require(car)) install.packages("car")   # For VIF calculation
library(car)

# Load the mtcars dataset
data(mtcars)

# Fit a linear regression model using 'mpg' as the dependent variable and other variables as predictors
model <- lm(mpg ~ ., data = mtcars)

# Display the regression coefficients
print("Regression Coefficients:")
summary(model)
```

```
# Check for multicollinearity using Variance Inflation Factor (VIF)
vif_values <- vif(model)
print("Variance Inflation Factor (VIF):")
print(vif_values)
```

**OUTPUT:**

```
[1] "Regression Coefficients:"

Call:
lm(formula = mpg ~ ., data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4506 -1.6044 -0.1196  1.2193  4.6271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.30337   18.71788   0.657   0.5181
cyl         -0.11144    1.04502  -0.107   0.9161
disp         0.01334    0.01786   0.747   0.4635
hp          -0.02148    0.02177  -0.987   0.3350
drat         0.78711    1.63537   0.481   0.6353
wt          -3.71530    1.89441  -1.961   0.0633 .
qsec         0.82104    0.73084   1.123   0.2739
vs           0.31776    2.10451   0.151   0.8814
am           2.52023    2.05665   1.225   0.2340
gear         0.65541    1.49326   0.439   0.6652
carb        -0.19942    0.82875  -0.241   0.8122
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.65 on 21 degrees of freedom
Multiple R-squared:  0.869,     Adjusted R-squared:  0.8066
F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07
[1] "Variance Inflation Factor (VIF):"
      cyl      disp        hp      drat        wt      qsec        vs        am
15.373833 21.620241  9.832037  3.374620 15.164887  7.527958  4.965873  4.648487
     gear      carb
 5.357452  7.908747
```

**15. Show the relationship between cyl (Cylinders) and hp (Horsepower) using a box/scatter plot.**

```
# Load the dataset
data(mtcars)

# Box Plot: Horsepower grouped by Cylinders
```

```
boxplot(hp ~ factor(cyl), data = mtcars,
        main = "Horsepower by Number of Cylinders",
        xlab = "Cylinders", ylab = "Horsepower",
        col = "skyblue")

# Scatter Plot: Cylinders vs Horsepower
plot(mtcars$cyl, mtcars$hp,
     main = "Scatter Plot: Cylinders vs Horsepower",
     xlab = "Cylinders", ylab = "Horsepower",
     pch = 19, col = "tomato")
```

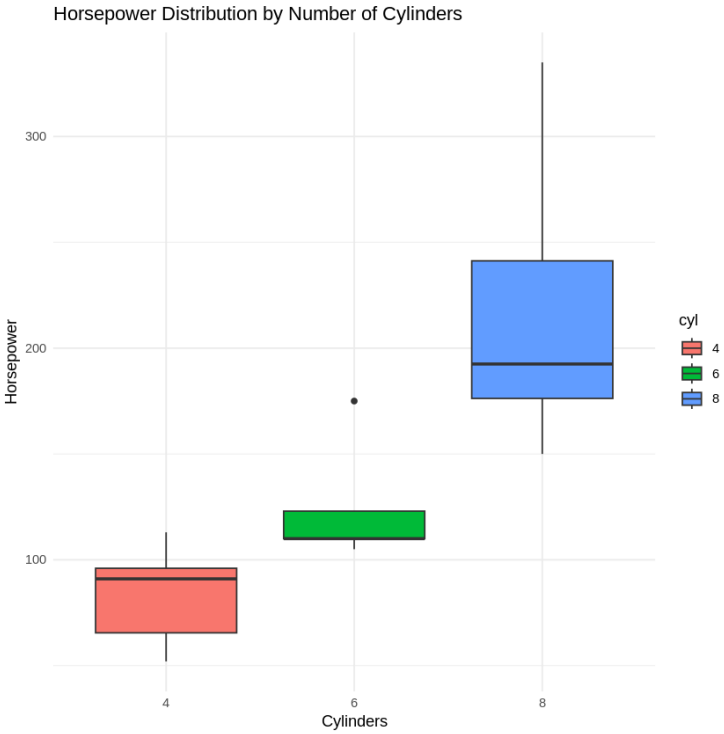
**(OR)**

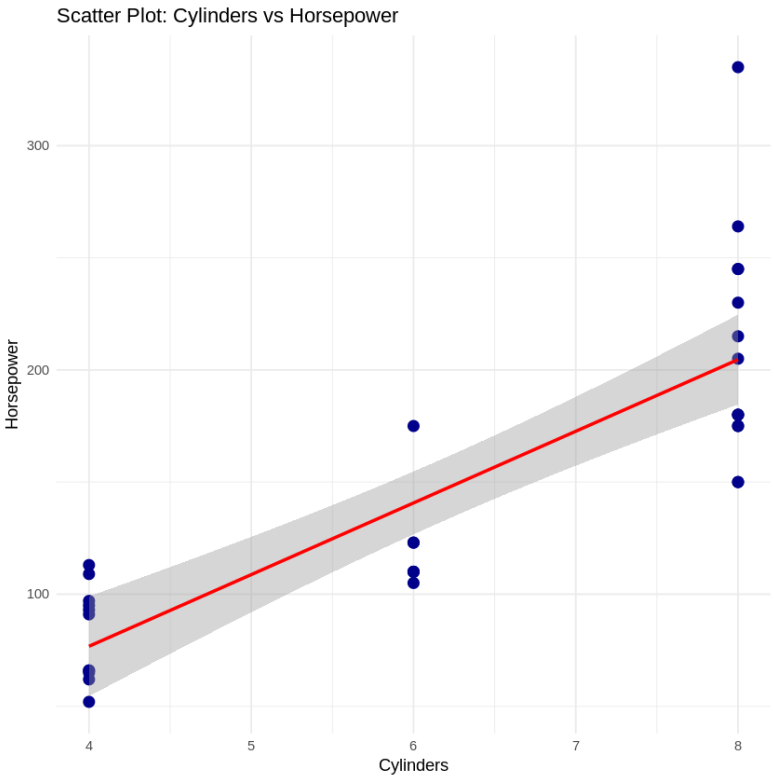
```
# Install and load necessary package
if (!require(ggplot2)) install.packages("ggplot2")
library(ggplot2)

# Load dataset
data(mtcars)

# Convert cyl to a factor for grouping
mtcars$cyl <- as.factor(mtcars$cyl)

# Summary statistics
cat("Summary of Horsepower by Cylinders:\n")
print(aggregate(hp ~ cyl, data = mtcars, summary))

# Correlation between hp and numeric cyl
correlation <- cor(mtcars$hp, as.numeric(as.character(mtcars$cyl)))
cat("\nPearson Correlation between hp and cyl:", correlation, "\n")
```

```
# Box Plot: Horsepower by Cylinders
ggplot(mtcars, aes(x = cyl, y = hp, fill = cyl)) +
  geom_boxplot() +
  labs(title = "Horsepower Distribution by Number of Cylinders",
       x = "Cylinders", y = "Horsepower") +
  theme_minimal()

# Scatter Plot with Linear Regression Line
ggplot(mtcars, aes(x = as.numeric(as.character(cyl)), y = hp)) +
  geom_point(color = "darkblue", size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  labs(title = "Scatter Plot: Cylinders vs Horsepower",
       x = "Cylinders", y = "Horsepower") +
  theme_minimal()
```

**OUTPUT:**

```
Summary of Horsepower by Cylinders:
  cyl   hp.Min. hp.1st Qu. hp.Median   hp.Mean hp.3rd Qu.   hp.Max.
1   4  52.00000   65.50000  91.00000  82.63636   96.00000 113.00000
2   6 105.00000  110.00000 110.00000 122.28571  123.00000 175.00000
3   8 150.00000  176.25000 192.50000 209.21429  241.25000 335.00000

Pearson Correlation between hp and cyl: 0.8324475
`geom_smooth()` using formula = 'y ~ x'
```


Horsepower Distribution by Number of Cylinders
300
200
100
Horsepower
4
6
8
Cylinders
cyl
4
6
8


Scatter Plot: Cylinders vs Horsepower
300
200
100
Horsepower
4
5
6
7
8
Cylinders

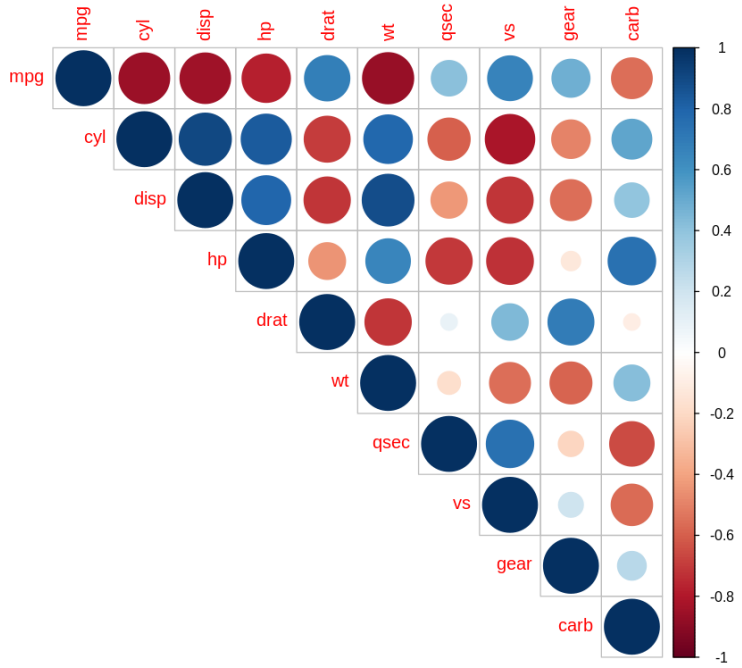
**16. Visualize correlations, split data, and train a Logistic Regression model. Evaluate using accuracy, precision, recall, and F1-score.**

```
# Install required packages
if (!require(caret)) install.packages("caret", dependencies = TRUE)
if (!require(corrplot)) install.packages("corrplot")
if (!require(e1071)) install.packages("e1071")  # Required for confusionMatrix

library(caret)
library(corrplot)
library(e1071)

# Load and prepare data
data(mtcars)
df <- mtcars
df$am <- factor(df$am)  # Target variable: automatic (0) vs manual (1)

# Correlation matrix (numeric only)
corr_matrix <- cor(df[, sapply(df, is.numeric)])
corrplot(corr_matrix, method = "circle", type = "upper")

# Data splitting
set.seed(123)
train_index <- createDataPartition(df$am, p = 0.7, list = FALSE)
train_data <- df[train_index, ]
test_data  <- df[-train_index, ]

# Train logistic regression
model <- glm(am ~ mpg + hp + wt, data = train_data, family = "binomial")
summary(model)
```

```
# Predictions
pred_probs <- predict(model, test_data, type = "response")
pred_class <- ifelse(pred_probs > 0.5, "1", "0")
pred_class <- factor(pred_class, levels = c("0", "1"))

# Evaluation
conf_mat <- confusionMatrix(pred_class, test_data$am, positive = "1")
print(conf_mat)
```

**OUTPUT:**

```
    Null deviance: 32.6013  on 23  degrees of freedom
Residual deviance:  8.5918  on 20  degrees of freedom
AIC: 16.592

Number of Fisher Scoring iterations: 9
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 5 0
         1 0 3

               Accuracy : 1
                 95% CI : (0.6306, 1)
    No Information Rate : 0.625
    P-Value [Acc > NIR] : 0.02328

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.375
         Detection Rate : 0.375
   Detection Prevalence : 0.375
      Balanced Accuracy : 1.000

       'Positive' Class : 1
```




**17.Identify numerical/categorical variables and handle missing values by imputing with mean, median, or mode. Convert categorical data into factors.**

```
# Install required packages
if (!require(caret)) install.packages("caret", dependencies = TRUE)
if (!require(corrplot)) install.packages("corrplot")
if (!require(e1071)) install.packages("e1071")  # Required for confusionMatrix

library(caret)
library(corrplot)
library(e1071)

# Load and prepare data
data(mtcars)
df <- mtcars
df$am <- factor(df$am)  # Target variable: automatic (0) vs manual (1)

# Correlation matrix (numeric only)
corr_matrix <- cor(df[, sapply(df, is.numeric)])
```

```
corrplot(corr_matrix, method = "circle", type = "upper")

# Data splitting
set.seed(123)
train_index <- createDataPartition(df$am, p = 0.7, list = FALSE)
train_data <- df[train_index, ]
test_data  <- df[-train_index, ]

# Train logistic regression
model <- glm(am ~ mpg + hp + wt, data = train_data, family = "binomial")
summary(model)

# Predictions
pred_probs <- predict(model, test_data, type = "response")
pred_class <- ifelse(pred_probs > 0.5, "1", "0")
pred_class <- factor(pred_class, levels = c("0", "1"))

# Evaluation
conf_mat <- confusionMatrix(pred_class, test_data$am, positive = "1")
print(conf_mat)
```

**OUTPUT:**

```
Numerical Variables:
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"

Categorical Variables:
character(0)

Cleaned Data with Imputed Values:
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4          20.06129   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag      21.00000   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710         22.80000   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive     21.40000   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout  18.70000   8 360.0 123 3.15 3.440 17.02  0  0    3    2
Valiant            18.10000   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360         14.30000   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D          24.40000   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230           22.80000   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280           19.20000   6 167.6 123 3.92    NA 18.30  1  0    4    4
Merc 280C          17.80000   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE         16.40000   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL         17.30000   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC        15.20000   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood 10.40000   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.40000  8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial  14.70000   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128           32.40000   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic        30.40000   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla     33.90000   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona      21.50000   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger   15.50000   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin        15.20000   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28         13.30000   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird   19.20000   8 400.0 175 3.08 3.845 17.05  0  0    3    2
```

```
 Fiat X1-9          27.30000   4  79.0  66 4.08 1.935 18.90  1  1    4    1
 Porsche 914-2      26.00000   4 120.3  91 4.43 2.140 16.70  0  1    5    2
 Lotus Europa       30.40000   4  95.1 113 3.77 1.513 16.90  1  1    5    2
 Ford Pantera L     15.80000   8 351.0 264 4.22 3.170 14.50  0  1    5    4
 Ferrari Dino       19.70000   6 145.0 175 3.62 2.770 15.50  0  1    5    6
 Maserati Bora      15.00000   8 301.0 335 3.54 3.570 14.60  0  1    5    8
 Volvo 142E         21.40000   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

**18 .Visualize mpg vs. wt (Weight) with color for cyl. 5.Create a bar plot showing car counts by gear (Transmission gears).**

1)# Install and load required packages

if (!require(ggplot2)) install.packages("ggplot2")

library(ggplot2)

# Load the mtcars dataset

data(mtcars)

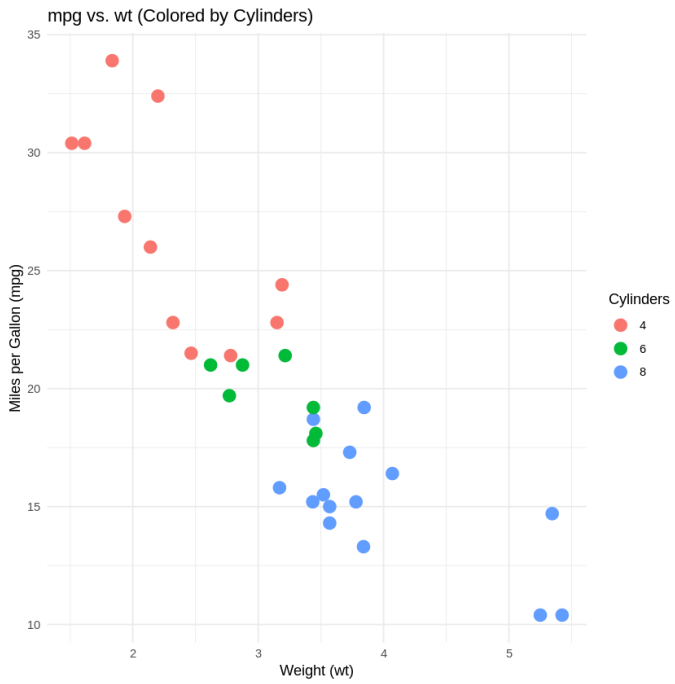
```
# Visualize mpg vs wt with color for cyl (Cylinders)
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point(size = 4) +
  labs(title = "mpg vs. wt (Colored by Cylinders)", x = "Weight (wt)", y = "Miles per Gallon (mpg)", color = "Cylinders") +
  theme_minimal()

# Bar plot showing the count of cars by gear (Transmission gears)
ggplot(mtcars, aes(x = factor(gear))) +
  geom_bar(fill = "steelblue") +
  labs(title = "Car Count by Gear (Transmission)", x = "Number of Gears", y = "Count of Cars") +
  theme_minimal()
```

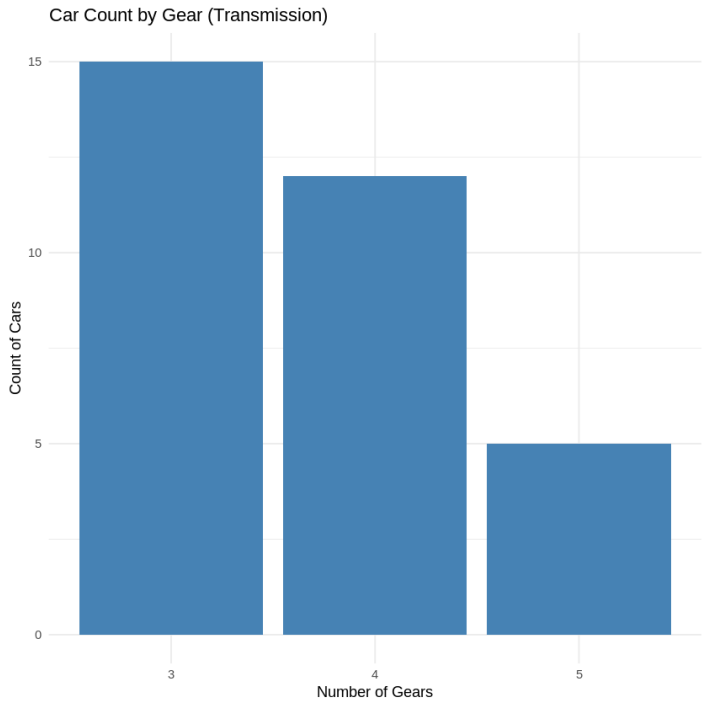
**(OR)**

```
# Install and load required packages
if (!require(ggplot2)) install.packages("ggplot2")
library(ggplot2)

# Load the mtcars dataset
data(mtcars)

# Convert 'gear' and 'cyl' to factors for better grouping in plots
mtcars$gear <- factor(mtcars$gear)
mtcars$cyl <- factor(mtcars$cyl)

# 1. Visualize mpg vs wt with color for cyl (Cylinders)
mpg_wt_plot <- ggplot(mtcars, aes(x = wt, y = mpg, color = cyl)) +
  geom_point(size = 4, alpha = 0.7) + # Scatter plot with transparency for better visibility
  scale_color_manual(values = c("red", "green", "blue")) + # Custom colors for cyl
  labs(title = "Miles per Gallon vs. Weight (Colored by Cylinders)",
```

```
      x = "Weight (wt)",
      y = "Miles per Gallon (mpg)",
      color = "Cylinders") +
  theme_minimal(base_size = 15) +  # Minimal theme with custom font size
  theme(legend.position = "top")  # Position the legend at the top

# Display the plot
print(mpg_wt_plot)

# 2. Bar plot showing the count of cars by gear (Transmission gears)
gear_bar_plot <- ggplot(mtcars, aes(x = gear, fill = gear)) +
  geom_bar(stat = "count", width = 0.7, show.legend = FALSE) + # Bar plot with custom width and no legend
  labs(title = "Car Count by Gear (Transmission)",
      x = "Number of Gears",
      y = "Count of Cars") +
  scale_fill_manual(values = c("orange", "purple", "cyan")) + # Custom colors for gears
  theme_minimal(base_size = 15) +  # Minimal theme with custom font size
  theme(legend.position = "none")  # No legend displayed

# Display the plot
print(gear_bar_plot)
```

**OUTPUT:**




**19. handle missing values by imputing with mean, median, or mode.**

# Load required packages

```
if (!require(dplyr)) install.packages("dplyr")
library(dplyr)

# Sample dataset (with some missing values)
data(mtcars)
mtcars_with_na <- mtcars
mtcars_with_na[1, "mpg"] <- NA
mtcars_with_na[5, "hp"] <- NA

# Impute missing numerical variables
mtcars_with_na$mpg[is.na(mtcars_with_na$mpg)] <- mean(mtcars_with_na$mpg, na.rm = TRUE)  # Mean for mpg
mtcars_with_na$hp[is.na(mtcars_with_na$hp)] <- median(mtcars_with_na$hp, na.rm = TRUE)  # Median for hp

# Impute missing categorical variables (mode)
get_mode <- function(v) { uniqv <- unique(v); uniqv[which.max(tabulate(match(v, uniqv)))] }
mtcars_with_na$cyl[is.na(mtcars_with_na$cyl)] <- get_mode(mtcars_with_na$cyl)

# View the cleaned dataset
print(mtcars_with_na)
```

**OUTPUT:**

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 20.06129 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.00000 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.80000 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.40000 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.70000 | 8 | 360.0 | 123 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.10000 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.30000 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.40000 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.80000 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.20000 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.80000 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.40000 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |
| Merc 450SL | 17.30000 | 8 | 275.8 | 180 | 3.07 | 3.730 | 17.60 | 0 | 0 | 3 | 3 |
| Merc 450SLC | 15.20000 | 8 | 275.8 | 180 | 3.07 | 3.780 | 18.00 | 0 | 0 | 3 | 3 |
| Cadillac Fleetwood | 10.40000 | 8 | 472.0 | 205 | 2.93 | 5.250 | 17.98 | 0 | 0 | 3 | 4 |
| Lincoln Continental | 10.40000 | 8 | 460.0 | 215 | 3.00 | 5.424 | 17.82 | 0 | 0 | 3 | 4 |
| Chrysler Imperial | 14.70000 | 8 | 440.0 | 230 | 3.23 | 5.345 | 17.42 | 0 | 0 | 3 | 4 |
| Fiat 128 | 32.40000 | 4 | 78.7 | 66 | 4.08 | 2.200 | 19.47 | 1 | 1 | 4 | 1 |
| Honda Civic | 30.40000 | 4 | 75.7 | 52 | 4.93 | 1.615 | 18.52 | 1 | 1 | 4 | 2 |
| Toyota Corolla | 33.90000 | 4 | 71.1 | 65 | 4.22 | 1.835 | 19.90 | 1 | 1 | 4 | 1 |
| Toyota Corona | 21.50000 | 4 | 120.1 | 97 | 3.70 | 2.465 | 20.01 | 1 | 0 | 3 | 1 |
| Dodge Challenger | 15.50000 | 8 | 318.0 | 150 | 2.76 | 3.520 | 16.87 | 0 | 0 | 3 | 2 |
| AMC Javelin | 15.20000 | 8 | 304.0 | 150 | 3.15 | 3.435 | 17.30 | 0 | 0 | 3 | 2 |
| Camaro Z28 | 13.30000 | 8 | 350.0 | 245 | 3.73 | 3.840 | 15.41 | 0 | 0 | 3 | 4 |
| Pontiac Firebird | 19.20000 | 8 | 400.0 | 175 | 3.08 | 3.845 | 17.05 | 0 | 0 | 3 | 2 |
| Fiat X1-9 | 27.30000 | 4 | 79.0 | 66 | 4.08 | 1.935 | 18.90 | 1 | 1 | 4 | 1 |
| Porsche 914-2 | 26.00000 | 4 | 120.3 | 91 | 4.43 | 2.140 | 16.70 | 0 | 1 | 5 | 2 |
| Lotus Europa | 30.40000 | 4 | 95.1 | 113 | 3.77 | 1.513 | 16.90 | 1 | 1 | 5 | 2 |
| Ford Pantera L | 15.80000 | 8 | 351.0 | 264 | 4.22 | 3.170 | 14.50 | 0 | 1 | 5 | 4 |
| Ferrari Dino | 19.70000 | 6 | 145.0 | 175 | 3.62 | 2.770 | 15.50 | 0 | 1 | 5 | 6 |
| Maserati Bora | 15.00000 | 8 | 301.0 | 335 | 3.54 | 3.570 | 14.60 | 0 | 1 | 5 | 8 |
| Volvo 142E | 21.40000 | 4 | 121.0 | 109 | 4.11 | 2.780 | 18.60 | 1 | 1 | 4 | 2 |

**20. Apply PCA for dimensionality reduction and assess if performance improves for better**

```
# Load necessary packages
if (!require(dplyr)) install.packages("dplyr")
if (!require(caret)) install.packages("caret")
if (!require(stats)) install.packages("stats")
library(dplyr)
library(caret)
library(stats)
```

```
# Load the mtcars dataset
data(mtcars)

# Standardize the dataset (important for PCA)
mtcars_scaled <- scale(mtcars)

# Apply PCA
pca <- prcomp(mtcars_scaled, center = TRUE, scale. = TRUE)

# Summary of PCA (Variance explained by each component)
summary(pca)

# Create a data frame with PCA-transformed data
pca_data <- data.frame(pca$x)

# Add the response variable 'mpg' back to the PCA data (for regression)
pca_data$mpg <- mtcars$mpg

# Train a linear regression model using original data (before PCA)
set.seed(123)
model_original <- train(mpg ~ ., data = mtcars, method = "lm")
print("Original Model Performance:")
print(model_original$results)

# Train a linear regression model using PCA-transformed data
set.seed(123)
model_pca <- train(mpg ~ ., data = pca_data, method = "lm")
print("PCA Model Performance:")
print(model_pca$results)
```

**OUTPUT:**

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6    PC7
Standard deviation     2.5707 1.6280 0.79196 0.51923 0.47271 0.46000 0.3678
Proportion of Variance 0.6008 0.2409 0.05702 0.02451 0.02031 0.01924 0.0123
Cumulative Proportion  0.6008 0.8417 0.89873 0.92324 0.94356 0.96279 0.9751
                           PC8    PC9    PC10   PC11
Standard deviation     0.35057 0.2776 0.22811 0.1485
Proportion of Variance 0.01117 0.0070 0.00473 0.0020
Cumulative Proportion  0.98626 0.9933 0.99800 1.0000
[1] "Original Model Performance:"
  intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
1      TRUE 4.419971 0.5635225 3.545899 1.313434  0.2141735 0.9752435
[1] "PCA Model Performance:"
  intercept         RMSE Rsquared          MAE      RMSESD RsquaredSD
1      TRUE 7.461506e-15        1 5.986891e-15 4.56624e-15          0
         MAESD
1 3.744624e-15
```