|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1.CLUSTERING** | **customer\_data$Cluster <-** | **veg\_count <-** | **5. SCATTER PLOT FOR** | **c("Bread", "Milk",** | **sorted\_marks <- sort(marks)** | **print(clustering\_bins)** | **hist(marks, breaks =** |
| **ANALYSIS** | **as.factor(kmeans\_result$clust** | **sum(people\_data$Vegetarian** | **MOBILE PHONE** | **"Yogurt"),** | **equal\_freq\_bins <-** | **pdf("plots.pdf", width = 8,** | **length(marks), main =** |

library(ggplot2) customer\_data <- data.frame( CustomerID = 1:5,

Gender = c("Male", "Male",

"Female", "Female", "Female"),

Age = c(19, 21, 20, 23, 31),

AnnualIncome = c(15, 15, 16,

16, 17),

SpendingScore = c(39, 81, 6,

77, 40)

**)**

print(customer\_data)

data\_for\_clustering <- customer\_data[, c("AnnualIncome", "SpendingScore")]

n <- nrow(data\_for\_clustering)

k <- min(2, n) set.seed(123) kmeans\_result <-

kmeans(data\_for\_clustering, centers = k, nstart = 25)

AnnualIncome, y = SpendingScore, color = Cluster)) +

geom\_point(size = 4) +

scale\_color\_manual(values = c("red", "blue")) +

labs(title = "Customer Segments Based on Annual Income and Spending Score",

x = "Annual Income (k$)",

y = "Spending Score (1- 100)") +

theme\_minimal()

# 4.PERSON TOTAL COUNT

people\_data <- data.frame( Person = c("Gopu", "Babu",

"Baby", "Gopal", "Krishna",

"Jai", "Dev", "Malini",

"Hema", "Anu"),

Vegetarian = c("yes", "yes", "yes", "no", "yes", "no",

"no", "yes", "yes", "yes"))

sum(people\_data$Vegetarian

== "no")

cat("Number of Vegetarians:", veg\_count, "\n")

cat("Number of Non- Vegetarians:", non\_veg\_count, "\n")

if (veg\_count > non\_veg\_count) {

cat("The number of vegetarians is greater.\n")

} else if (veg\_count < non\_veg\_count) {

cat("The number of non- vegetarians is greater.\n")

} else {

cat("The number of vegetarians and non- vegetarians is equal.\n")

**}**

y <- c(12, 5, 13, 19, 31, 7, 153,

72, 275, 110)

plot(x, y, main = "Scatter Plot of Mobile Phones Sold vs Money",

xlab = "Number of Mobile Phones Sold", ylab = "Money",

pch = 19, col = "blue")

abline(lm(y ~ x), col = "red", lwd = 2)

# 6. TRANSACTION FOR PURCHASED

if(!require(arules)) { install.packages("arules") library(arules)

**}**

transactions\_list <- list( c("Bread", "Cheese",

"Egg", "Juice"),

c("Bread", "Cheese", "Juice"),

c("Cheese", "Juice", "Milk"))

transactions <- as(transactions\_list, "transactions")

inspect(transactions)

rules <- apriori(transactions, parameter = list(supp = 0.5, conf = 0.75))

inspect(rules)

# 8. HISTOGRAM FOR STUDENT MARKS

marks <- c(55, 60, 71, 63, 55,

65, 50, 55, 58, 59, 61, 63, 65,

67, 71, 72, 75)

if(!require(cluster)) { install.packages("cluster") library(cluster)

**}**

num\_bins <- 3

bin\_size <- ceiling(length(marks) / num\_bins)

ks) / bin\_size))

cat("\nEqual-Frequency Partitioning Bins:\n")

print(equal\_freq\_bins) range\_marks <- range(marks)

bin\_width <- ceiling((range\_marks[2] - range\_marks[1]) / num\_bins)

equal\_width\_bins <- cut(marks, breaks = seq(range\_marks[1], range\_marks[2] + bin\_width, by = bin\_width), include.lowest = TRUE)

cat("\nEqual-Width Partitioning Bins:\n")

print(table(equal\_width\_bins)

**)**

set.seed(123) # For reproducibility

kmeans\_result <- kmeans(marks, centers = num\_bins, nstart = 20)

clustering\_bins <- kmeans\_result$cluster

cat("\nClustering Results:\n")

par(mfrow = c(3, 1))

hist(marks, breaks = length(marks), main = "Histogram: Equal- Frequency Partitioning",

xlab = "Marks", col = "lightblue", border = "black")

abline(v = unlist(lapply(equal\_freq\_bins, max)), col = "red", lwd = 2, lty = 2)

hist(marks, breaks = seq(range\_marks[1], range\_marks[2] + bin\_width, by = bin\_width),

main = "Histogram: Equal-Width Partitioning", xlab = "Marks", col = "lightgreen", border = "black")

abline(v = seq(range\_marks[1], range\_marks[2] + bin\_width, by = bin\_width), col = "blue", lwd = 2, lty = 2)

**,**

"lightpink", border = "black")

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **er)** | **== "yes")** | **x <- c(4, 1, 5, 7, 10, 2, 50, 25,** | **c("Bread", "Juice",** | **split(sorted\_marks,** | **height = 12) # Adjust the size** | **"Histogram: Clustering"** |
| **ggplot(customer\_data, aes(x =** | **non\_veg\_count <-** | **90, 36)** | **"Milk"),** | **ceiling(seq\_along(sorted\_mar** | **as needed** | **xlab = "Marks", col =** |

abline(v = tapply(marks, clustering\_bins, mean), col = "purple", lwd = 2, lty = 2)

par(mfrow = c(1, 1)) dev.off()

plot(1:10, main = "Test Plot")

# 11.MIN-MAX,Z-SCORE

strike\_rates <- c(100, 70, 60,

90, 90)

min\_value <- 0

max\_value <- 1

data\_min <- min(strike\_rates)

data\_max <- max(strike\_rates)

min\_max\_normalized <- (strike\_rates - data\_min) / (data\_max - data\_min) \* (max\_value - min\_value) + min\_value

cat("Min-Max Normalization:\n")

print(min\_max\_normalized)

mean\_value <- mean(strike\_rates)

std\_dev <- sd(strike\_rates)

z\_score\_normalized <- (strike\_rates - mean\_value) / std\_dev

cat("\nZ-Score Normalization:\n")

print(z\_score\_normalized)

mad <- mean(abs(strike\_rates

- mean\_value))

mad

cat("\nZ-Score Normalization using MAD:\n")

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **z\_score\_mad\_normalized <-** | **cat("AvgSpeed:",** | **cov\_matrix <-** | **sorted\_prices <- sort(prices)** | **abline(v = bin\_means, col =** | **boxplot(class\_A, class\_B,** | **print(z\_score\_normalized)** |
| **(strike\_rates - mean\_value) /** | **std\_dev\_avg\_speed, "\n")** | **cov(preferences)** | **equal\_freq\_bins <-** | **"red", lwd = 2, lty = 2)** | **names = c("Class A",** |  |

print(z\_score\_mad\_normalize d)

j <- ceiling(log10(max(abs(strike\_ rates)))) # Calculate j for decimal scaling

decimal\_scaled <- strike\_rates

/ (10^j)

cat("\nDecimal Scaling Normalization:\n")

print(decimal\_scaled)

# 12.SD &VARIANCE OF AVG SPEED

AvgSpeed <- c(78, 81, 82, 74,

83, 82, 77, 80)

TotalTime <- c(39, 37, 36, 42,

35, 36, 40, 70, 38, 46)

std\_dev\_avg\_speed <- sd(AvgSpeed)

std\_dev\_total\_time <- sd(TotalTime)

cat("Standard Deviation:\n")

cat("TotalTime:", std\_dev\_total\_time, "\n")

variance\_avg\_speed <- var(AvgSpeed)

variance\_total\_time <- var(TotalTime)

cat("\nVariance:\n")

cat("AvgSpeed:", variance\_avg\_speed, "\n")

cat("TotalTime:", variance\_total\_time, "\n")

# COVARIENCE & CORRELATION

photograph\_A <- c(18, 2, 20)

photograph\_B <- c(22, 28, 10)

photograph\_C <- c(20, 40, 40)

preferences <- data.frame(A = photograph\_A, B = photograph\_B, C = photograph\_C)

cov\_BC <- cov(preferences$B, preferences$C)

cat("Sample Covariance between B and C:", cov\_BC, "\n")

cat("Sample Covariance Matrix:\n")

print(cov\_matrix)

cor\_BC <- cor(preferences$B, preferences$C)

cat("Sample Correlation between B and C:", cor\_BC, "\n")

cor\_matrix <- cor(preferences)

cat("Sample Correlation Matrix:\n")

print(cor\_matrix)

# HISTOGRAM FOR FD

prices <- c(1, 1, 5, 5, 5, 5, 5, 8,

8, 10, 10, 10, 10, 12, 14, 14,

14, 15, 15, 15, 15, 15,

15, 18, 18, 18, 18, 18, 18, 18,

18, 20, 20, 20, 20, 20,

20, 20, 21, 21, 21, 21, 25,

25, 25, 25, 25, 28, 28,

30, 30, 30)

num\_bins <- 3

bin\_size <- ceiling(length(prices) / num\_bins)

split(sorted\_prices, ceiling(seq\_along(sorted\_pric es) / bin\_size))

cat("Equal-Frequency Partitioning Bins:\n")

print(equal\_freq\_bins)

bin\_means <- sapply(equal\_freq\_bins, mean)

bin\_boundaries <- lapply(equal\_freq\_bins, range)

cat("\nBin Means:\n") print(bin\_means) cat("\nBin Boundaries:\n") print(bin\_boundaries)

breaks <- seq(min(prices) - 1, max(prices) + 1, by = 5)

hist(prices, breaks = breaks,

main = "Histogram of Prices",

xlab = "Price",

ylab = "Frequency", col = "lightblue", border = "black")

# MEAN,MEDIAN &BOX PLOT

class\_A <- c(76, 35, 47, 64, 95,

66, 89, 36, 84)

class\_B <- c(51, 56, 84, 60, 59,

70, 63, 66, 50)

mean\_A <- mean(class\_A) median\_A <- median(class\_A) range\_A <- range(class\_A) mean\_B <- mean(class\_B) median\_B <- median(class\_B) range\_B <- range(class\_B) cat("Class A:\n") cat("Mean:", mean\_A, "\n")

cat("Median:", median\_A, "\n")

cat("Range:", range\_A[1],

"to", range\_A[2], "\n") cat("\nClass B:\n") cat("Mean:", mean\_B, "\n")

cat("Median:", median\_B, "\n")

cat("Range:", range\_B[1],

"to", range\_B[2], "\n")

"Class B"),

main = "Boxplot of Class A and Class B Exam Scores",

ylab = "Scores",

col = c("lightblue", "lightgreen"))

grid()

# MIN,MAX & Z- SCORE

data <- c(200, 300, 400, 600,

1000)

min\_val <- min(data) max\_val <- max(data)

min\_max\_normalized <- (data

- min\_val) / (max\_val - min\_val)

cat("Min-Max Normalized Values:\n")

print(min\_max\_normalized) mean\_val <- mean(data) sd\_val <- sd(data)

z\_score\_normalized <- (data - mean\_val) / sd\_val

cat("\nZ-Score Normalized Values:\n")

# MPG &QSEC

data("AirPassengers") data("mtcars") hist(AirPassengers,

breaks = seq(100, 700, by = 150),

main = "Histogram of AirPassengers",

xlab = "Number of Passengers",

col = "lightblue", xlim = c(100, 700), border = "black")

plot(mtcars$mpg, type = "l", col = "blue", lwd = 2, ylim = c(10, 30),

ylab = "Values", xlab = "Index", main = "Line Chart of mpg and qsec")

lines(mtcars$qsec, col = "red", lwd = 2)

legend("topright", legend = c("mpg", "qsec"), col =

c("blue", "red"), lty = 1, lwd

= 2)

# MORTALITY

set.seed(123)

hardness <- seq(0, 100, by = 10)

mortality <- 5 + 0.4 \* hardness + rnorm(length(hardness), mean = 0, sd = 2) # Adding noise

water <- data.frame(hardness, mortality)

str(water)

plot(water$hardness, water$mortality,

main = "Mortality vs Hardness",

xlab = "Hardness", ylab = "Mortality", col = "blue",

pch = 19)

linear\_model <- lm(mortality

~ hardness, data = water) summary(linear\_model) hardness\_value <- 88

predicted\_mortality <- predict(linear\_model, newdata = data.frame(hardness = hardness\_value))

cat("Predicted Mortality for Hardness =", hardness\_value, "is:", predicted\_mortality, "\n")

# MPG,CYL,MTCARS

library(ggplot2) data(mtcars)

boxplot(mpg ~ as.factor(cyl), data = mtcars,

main = "Boxplot of MPG by Number of Cylinders",

xlab = "Number of Cylinders",

ylab = "Miles per Gallon (MPG)",

col = "lightblue", border = "black")

grid()

# TENNIS BOX PLOT

scores <- c(20, 22, 23, 24, 25,

24, 26, 27, 30, 31, 40, 42, 43,

45, 50)

boxplot(scores,

main = "Boxplot of Tennis Players' Scores",

ylab = "Scores", col = "lightgreen", border = "black")

grid()

outliers <- boxplot.stats(scores)$out

points(rep(1, length(outliers)), outliers, col = "red", pch = 19)

# SCATTER PLOT &BAR CHART

library(ggplot2) library(dplyr) set.seed(123)

n <- 100

age <- sample(20:80, n, replace = TRUE)

blood\_pressure <- rnorm(n, mean = 70 + (age - 20) \* 0.5,

sd = 10)

diabetes\_data <- data.frame(Age = age, BloodPressure = blood\_pressure)

ggplot(diabetes\_data, aes(x = Age, y = BloodPressure)) +

geom\_point(color = "blue", alpha = 0.6) +

labs(title = "Blood Pressure vs Age",

x = "Age",

y = "Blood Pressure") +

theme\_minimal()

diabetes\_data <- diabetes\_data %>%

mutate(AgeGroup = case\_when(

Age < 30 ~ "Under 30",

Age >= 30 & Age < 40 ~ "30-39",

Age >= 40 & Age < 50 ~ "40-49",

Age >= 50 & Age < 60 ~ "50-59",

Age >= 60 & Age < 70 ~ "60-69",

Age >= 70 ~ "70 and above"

**))**

age\_bp\_summary <- diabetes\_data %>%

group\_by(AgeGroup) %>% summarise(AverageBloodPres sure = mean(BloodPressure, na.rm = TRUE))

ggplot(age\_bp\_summary, aes(x = AgeGroup, y = AverageBloodPressure, fill = AgeGroup)) +

geom\_bar(stat = "identity")

**+**

labs(title = "Average Blood Pressure by Age Group",

x = "Age Group",

y = "Average Blood Pressure") +

theme\_minimal() +

theme(legend.position = "none") # Hide the legend

library(ggplot2)

data("water", package = "datasets")

str(water)

ggplot(water, aes(x = hardness, y = mortality)) +

geom\_point(color = "blue")

**+**

labs(title = "Scatter Plot of Mortality vs Hardness",

x = "Hardness (mg/L)", y = "Mortality Rate") +

theme\_minimal()

model <- lm(mortality ~ hardness, data = water)

summary(model)

new\_data <- data.frame(hardness = 88)

predicted\_mortality <- predict(model, newdata = new\_data)

cat("Predicted mortality for hardness = 88:", predicted\_mortality, "\n")