K MEAN CLUSTERING

1st output

```
# Install required packages if not already installed
if (!require("ggplot2")) install.packages("ggplot2")
# Load required library
library(ggplot2)
# Read the dataset
grade_input <- read.csv("C:\\Users\\lenovo\\Downloads\\grades_km_input.csv")
# Select relevant columns for clustering
kmdata <- as.matrix(grade_input[, c("English", "Math", "Science")])
# Elbow Method to determine optimal number of clusters
wss <- sapply(1:15, function(k) sum(kmeans(kmdata, k, nstart = 25)$withinss))
# Plot Elbow Method
plot(1:15, wss, type = "b", xlab = "Number of Clusters", ylab = "Within Sum of Squares (WSS)",
    main = "Elbow Method for Optimal Clusters")
```

2nd output

```
# Install required packages if not already installed
if (!require("ggplot2"))
install.packages("ggplot2")
if (!require("gridExtra"))
install.packages("gridExtra")
# Load required libraries
library(ggplot2)
library(gridExtra)
# Read the dataset
grade_input <-
read.csv("C:\\Users\\lenovo\\Downloads\\grades_km_input.csv")
# Select relevant columns for clustering
kmdata <- as.matrix(grade_input[, c("English", "Math", "Science")])
# K-means clustering with 3 clusters
km <- kmeans(kmdata, 3, nstart = 25)
# Prepare dataframe for plotting
df <- as.data.frame(kmdata)
df$cluster <- factor(km$cluster)
# Ensure centers have correct column names
centers <- as.data.frame(km$centers)
colnames(centers) <- c("English", "Math", "Science")
# Scatter Plot 1: English vs Math
g1 <- ggplot(df, aes(English, Math, color = cluster)) +
  geom_point() +
  geom_point(data = centers, aes(English, Math), color = "black", size = 5, shape = 8) +
  labs(title = "English vs Math")
# Scatter Plot 2: English vs Science
g2 <- ggplot(df, aes(English, Science, color = cluster)) +
  geom_point() +
  geom_point(data = centers, aes(English, Science), color = "black", size = 5, shape = 8) +
  labs(title = "English vs Science")
# Scatter Plot 3: Math vs Science
g3 <- ggplot(df, aes(Math, Science, color = cluster)) +
  geom_point() +
  geom_point(data = centers, aes(Math, Science), color = "black", size = 5, shape = 8) +
  labs(title = "Math vs Science")
# Arrange all plots vertically
grid.arrange(g1, g2, g3, ncol = 1, top = "High School Student Cluster Analysis")
```

APRIORI

```
# Install required packages
install.packages("arules")
install.packages("arulesViz")
install.packages("RColorBrewer")
# Load libraries
library(arules)
library(arulesViz)
library(RColorBrewer)
# Load the Groceries dataset
data("Groceries")
# Summary and class of the dataset
summary(Groceries)
class(Groceries)
# Generate association rules
rules <- apriori(Groceries, parameter =
list(supp = 0.02, conf = 0.2))
# Summary and inspection of rules
summary(rules)
inspect(rules[1:10])
# Plot top 20 item frequencies (relative)
itemFrequencyPlot(Groceries,
            topN = 20,
            col = brewer.pal(8, 'Pastel2'),
            main = 'Relative Item Frequency
Plot',
            type = "relative",
            ylab = "Item Frequency (Relative)")
# Generate frequent itemsets of length 2
itemsets_2 <- apriori(Groceries,
                parameter = list(minlen = 2,
maxlen = 2, support = 0.02, target = "frequent
 itemsets"))
summary(itemsets_2)
inspect(itemsets_2[1:10])
# Generate frequent itemsets of length 3
itemsets_3 <- apriori(Groceries,
                parameter = list(minlen = 3,
maxlen = 3, support = 0.02, target = "frequent
 itemsets"))
summary(itemsets_3)
inspect(itemsets_3)
```

LINEAR REGRESSION

```
# Define data vectors years_of_exp = c(7, 5, 1,
3) salary_in_lakhs = c(21, 13, 6, 8) # Create a
data frame employee.data =
data.frame(years_of_exp, salary_in_lakhs) #
View the dataset print(employee.data) # Build
the linear regression model model =
lm(salary_in_lakhs ~ years_of_exp, data =
employee.data) # Show model summary
summary(model) # Plot the data with
regression line plot(salary_in_lakhs ~
years_of_exp, data = employee.data, main =
"Salary vs Years of Experience", xlab = "Years
of Experience", ylab = "Salary (in Lakhs)", col =
"blue", pch = 16) # Add regression line
abline(model, col = "red", lwd = 2)
```

LOGISTIC REGRESSION – NOT VISUAL

```
# Install and load packages
install.packages("ISLR")
install.packages("InformationValue")
install.packages("caret") library(ISLR)
library(InformationValue) library(caret) # Load
data data <- ISLR::Default print(head(data))
summary(data) nrow(data) # Split data into
train and test set.seed(1) sample <-
sample(c(TRUE, FALSE), nrow(data), replace =
TRUE, prob = c(0.7, 0.3)) train <- data[sample,
] test <- data[!sample, ] nrow(train) nrow(test)
# Fit logistic regression model model <-
glm(default ~ student + balance + income,
family = "binomial", data = train)
summary(model) # Predict probabilities on
test set predicted_probs <- predict(model,
test, type = "response") # Convert
probabilities to class labels using 0.5 cutoff
predicted_classes <- ifelse(predicted_probs >
0.5, "Yes", "No") # Confusion matrix cm<-
confusionMatrix(factor(predicted_classes),
factor(test$default), positive = "Yes")
print(cm)
```

DECISION TREE

```
# Training set visualization
dev.new()
plot(set[,-3],
    main = "Decision Tree Classification
(Training Set)",
    xlab = "Age", ylab = "Estimated Salary",
    xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid),
length(X1), length(X2)), add = TRUE)
points(grid_set, pch = ".", col = ifelse(y_grid ==
1, "springgreen3", "tomato"))
points(set, pch = 21, bg = ifelse(set[, 3] == 1,
"green4", "red3"))

# Test set visualization
dev.new()
plot(set[,-3],
    main = "Decision Tree Classification (Test
Set)",
    xlab = "Age", ylab = "Estimated Salary",
    xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid),
length(X1), length(X2)), add = TRUE)
points(grid_set, pch = ".", col = ifelse(y_grid ==
1, "springgreen3", "tomato"))
points(set, pch = 21, bg = ifelse(set[, 3] == 1,
"green4", "red3"))

# Decision Tree Plot
dev.new()
rpart.plot(classifier, type = 3, extra = 101,
fallen.leaves = TRUE,
         main = "Decision Tree Visualization")
```

NAÏVE BAYES CLASSIFICATION

```
dataset <-
read.csv("C:\\Users\\lenovo\\Downloads\\soc
ialnetworking.csv")
dataset <- dataset[3:5] # Keep Age,
EstimatedSalary, Purchased
dataset$Purchased <-
factor(dataset$Purchased, levels = c(0, 1))
install.packages("caTools")
library(caTools)
set.seed(123)
split <- sample.split(dataset$Purchased,
SplitRatio = 0.75)
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)
training_set[, 1:2] <- scale(training_set[, 1:2])
test_set[, 1:2] <- scale(test_set[, 1:2])
install.packages("e1071")
library(e1071)
classifier <- naiveBayes(x = training_set[, 1:2],
y = training_set$Purchased)
y_pred <- predict(classifier, newdata =
test_set[, 1:2])
cm <- table(Actual = test_set$Purchased,
Predicted = y_pred)
print("Confusion Matrix:")
print(cm)
set <- training_set
X1 <- seq(min(set[, 1])- 1, max(set[, 1]) + 1, by
= 0.01)
X2 <- seq(min(set[, 2])- 1, max(set[, 2]) + 1, by
= 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c("Age",
"EstimatedSalary")
y_grid <- predict(classifier, newdata =
grid_set)
dev.new() # Open new window
plot(set[,-3],
    main = "Naive Bayes (Training Set)",
    xlab = "Age", ylab = "Estimated Salary",
    xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid),
length(X1), length(X2)), add = TRUE)
points(grid_set, pch = ".", col = ifelse(y_grid ==
1, "springgreen3", "tomato"))
points(set, pch = 21, bg = ifelse(set[, 3] == 1,
"green4", "red3"))
set <- test_set
X1 <- seq(min(set[, 1])- 1, max(set[, 1]) + 1, by
= 0.01)
X2 <- seq(min(set[, 2])- 1, max(set[, 2]) + 1, by
= 0.01)
```

```
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c("Age",
"EstimatedSalary")
y_grid <- predict(classifier, newdata =
grid_set)
dev.new() # Open new window
plot(set[,-3],
    main = "Naive Bayes (Test Set)",
    xlab = "Age", ylab = "Estimated Salary",
    xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid),
length(X1), length(X2)), add = TRUE)
points(grid_set, pch = ".", col = ifelse(y_grid ==
1, "springgreen3", "tomato"))
points(set, pch = 21, bg = ifelse(set[, 3] == 1,
"green4", "red3"))
```

TEXT ANALYSIS

```
# Read dataset
dataset_original <-
read.delim("C:\\Users\\lenovo\\Downloads\\
Restaurant_Reviews.tsv",
                    quote = '', stringsAsFactors =
FALSE)
# Install and load text mining packages
install.packages('tm')
install.packages('SnowballC')
library(tm)
library(SnowballC)
# Text preprocessing
corpus <-
VCorpus(VectorSource(dataset_original$Revie
w))
corpus <- tm_map(corpus,
content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus,
removePunctuation)
corpus <- tm_map(corpus, removeWords,
stopwords())
corpus <- tm_map(corpus, stemDocument)
corpus <- tm_map(corpus, stripWhitespace)
# Create Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)
dtm <- removeSparseTerms(dtm, 0.999)
# Convert to data frame
dataset <- as.data.frame(as.matrix(dtm))
dataset$Liked <- dataset_original$Liked
# Encode target as factor
dataset$Liked <- factor(dataset$Liked, levels =
c(0, 1))
# Install and load caTools
install.packages('caTools')
library(caTools)
# Split dataset
set.seed(123)
split <- sample.split(dataset$Liked, SplitRatio
= 0.8)
training_set <- subset(dataset, split == TRUE)
test_set <- subset(dataset, split == FALSE)
# Install and load randomForest
install.packages('randomForest')
library(randomForest)
# Train Random Forest
classifier <- randomForest(x = training_set[,-
ncol(training_set)],
                    y = training_set$Liked,
                    ntree = 10)
# Predict
y_pred <- predict(classifier, newdata =
test_set[,-ncol(test_set)])
# Confusion Matrix
cm <- table(Actual = test_set$Liked, Predicted
= y_pred)
print("Confusion Matrix:")
print(cm)
```