***#Remove duplicates***

# Create a data frame

df <- data.frame(

ID = c(1, 2, 1, 2),

Name = c("Gopal Sharma", "Ram Pandey", "Gopal Sharma", "Ram Pandey"),

Product = c("Laptop", "Tablet", "Laptop", "Tablet"),

Quantity = c(1, 2, 1, 2),

Price = c(900, 300, 900, 300))

df\_cleaned <- df[!duplicated(df), ]

# Display the cleaned data frame

print(df\_cleaned)

**#Standardization Format**

# Create initial dataframe

df <- data.frame(

TransactionID = c("T001", "T002", "T003"),

Date = c("01/08/2023", "2023-08-01", "August 1, 2023"),

PaymentMethod = c("Credit Card", "credit card", "CC"),

stringsAsFactors = FALSE)

***# Standardize the Date column***

df$Date <- as.Date(df$Date, tryFormats = c("%d/%m/%Y", "%Y-%m-%d", "%B %d, %Y"))

# Standardize the PaymentMethod column

df$PaymentMethod <- tolower(df$PaymentMethod)

df$PaymentMethod[df$PaymentMethod %in% c("credit card", "cc")] <- "credit card"

# View the standardized data frame

print(df)

***#Missing values Treatment***

# Create initial data frame which comprises missing values

df <- data.frame(StudentID = c("S001", "S002", "S003"),

Name = c("Deepti Gupta ", " Arati Tripathy ", " Badri Sehgal "),

Age = c(17, NA, 18),

Score = c(88, 92, NA),

stringsAsFactors = FALSE)

**# Impute missing Age with mean age**

mean\_age <- mean(df$Age, na.rm = TRUE)

df$Age[is.na(df$Age)] <- mean\_age

# Impute missing Score with mean score

mean\_score <- mean(df$Score, na.rm = TRUE)

df$Score[is.na(df$Score)] <- mean\_score

# View the data frame after imputing missing values

print(df)

***#Treatment of Outliers***

# Create the sales data frame

sales\_data <- data.frame(Day = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"),Sales = c(1200, 1150, 1180, 10000, 1220))

Identifying the Outlier:

# Calculate basic statistics

mean\_sales <- mean(sales\_data$Sales)

sd\_sales <- sd(sales\_data$Sales)

# Identify outlier (values beyond mean ± 2\*SD)

sales\_data$Outlier <- abs(sales\_data$Sales - mean\_sales) > 2 \* sd\_sales

# View the data with outlier flagged

print(sales\_data)

***#Merging and Splitting Columns***

# Employee Details Data Frame

employee\_details <- data.frame(EmployeeID = c("E001", "E002"),FirstName = c("Rajesh", "Anjali"),

LastName = c("Kumar", "Singh"))

# Department Info Data Frame

department\_info <- data.frame(EmployeeID = c("E001", "E002"),Department = c("Engineering", "Marketing"))

# Merging the Data Frames on 'EmployeeID'

merged\_data <- merge(employee\_details, department\_info, by = "EmployeeID")

# Displaying the Merged Data

print(merged\_data)

***#Splitting Columns***

#For splitting, the tidyr package streamlines the process.

#R Code for Splitting

# Install 'tidyr' if not already installed

# install.packages("tidyr")

library(tidyr)

# Full Address Data Frame

addresses <- data.frame(FullAddress = c("123 MG Road, Bengaluru, Karnataka 560001","456 Park Street, Kolkata, West Bengal 700016"))

# Splitting the 'FullAddress' Column into Separate Columns

addresses\_split <- addresses %>% separate(FullAddress, into = c("Street", "City", "State", "PIN\_Code"), sep = ",\\s|\\s(?=\\d)")

# Displaying the Split Data

print(addresses\_split)

***#Bias Variance Trade Off***

# Load the mtcars dataset

data(mtcars)

# Set seed for reproducibility

set.seed(123)

# Split data into training and testing sets (70% train, 30% test)

train\_indices <- sample(1:nrow(mtcars), size = 0.7 \* nrow(mtcars))

train\_data <- mtcars[train\_indices, ]

test\_data <- mtcars[-train\_indices, ]

# Define degrees of polynomial to test

degrees <- 1:5 # Testing degrees from 1 to 5

# Initialize a vector to store testing RMSE for each degree

test\_rmse <- numeric(length(degrees))

# Loop over polynomial degrees

for (i in degrees) {

# Fit polynomial regression model

model <- lm(mpg ~ poly(wt, i), data = train\_data)

# Predict on test data

predictions <- predict(model, newdata = test\_data)

# Calculate Root Mean Squared Error (RMSE) on test data

rmse <- sqrt(mean((test\_data$mpg - predictions)^2))

test\_rmse[i] <- rmse

}

*# Identify the degree with the minimum test RMSE*

optimal\_degree <- which.min(test\_rmse)

cat("Optimal polynomial degree:", optimal\_degree, "\n")

# Plot Test RMSE vs. Polynomial Degree

plot(degrees, test\_rmse, type = "b", pch = 19, col = "blue",

xlab = "Polynomial Degree", ylab = "Test RMSE",

main = "Test RMSE vs. Polynomial Degree")

points(optimal\_degree, test\_rmse[optimal\_degree], col = "red", pch = 19)

***#Diagonostocs Plot***

data("mtcars")

names(mtcars)

model <- lm(mpg ~ cyl + wt,data=mtcars)

#Diagonostic Plot

plot(model,pch=18,cex=2)

***#Apply Multiple Regression***

library(psych)

setwd('D:/BPB\_Publication\_BOOK')

Reg\_DM<- read.csv (paste ("HR\_comma\_sep.csv", sep=""))

describe(Reg\_DM)

***#linear regression on number of projects done***

mlm <- lm (number\_project ~ satisfaction\_level+average\_montly\_hours+Work\_accident+promotion\_last\_5years+sales+salary, data = Reg\_DM)

summary(mlm)

***#Another Example of Multiple Regression***

# Load the dataset

Reg\_DM <- read.csv("HR\_comma\_sep.csv")

# Fit the regression model

simple\_lm <- lm(satisfaction\_level ~ number\_project, data = Reg\_DM)

# Plot the data points

plot(Reg\_DM$number\_project, Reg\_DM$satisfaction\_level,

xlab = "Number of Projects",

ylab = "Satisfaction Level",

main = "Satisfaction Level vs. Number of Projects")

# Add the regression line

abline(simple\_lm)

***#The following decision tree is built using IRIS dataset in R. Party package is a popular library in R for #constructing and visualizing a decision tree***

install.packages("party")

library(party)

names(iris)

str(iris)

set.seed(12345)

outr <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))

train.data <- iris[outr == 1, ]

test.data <- iris[outr == 2, ]

tree\_build <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width

ctree <- ctree(tree\_build, data = train.data)

# checking of the prediction

table(predict(ctree), train.data$Species)

plot(ctree)

plot (ctree, type = "simple")

testPred <- predict(ctree, newdata = test.data)

table(testPred, test.data$Species)

# logistic regression model using R. ISLR package is installed, and the Default dataset is #used in the following case study

# ISLR package in R is a collection of datasets and functions

install.packages("ISLR")

names(ISLR::Default)

data\_set<-ISLR::Default

summary(data\_set)

nrow(data\_set)

model\_regress<-glm(default~balance+student+income, family="binomial", data=data\_set)

***#k-Nearest Neighbors (k-NN) Classifier***

# Load necessary library

library(class)

# Use the built-in iris dataset

data(iris)

# Split the dataset into training and testing sets

set.seed(123)

index <- sample(1:nrow(iris), 0.7 \* nrow(iris))

train\_data <- iris[index, ]

test\_data <- iris[-index, ]

# Apply k-NN with k = 3

pred\_knn <- knn(train = train\_data[, -5], test = test\_data[, -5], cl = train\_data$Species, k = 3)

# Evaluate the model with a confusion matrix

confusion\_matrix <- table(Predicted = pred\_knn, Actual = test\_data$Species)

print(confusion\_matrix)

***#Support Vector Machine (SVM) Classifier***

# Load necessary library

library(e1071)

# Use the built-in iris dataset

data(iris)

# Split the dataset into training and testing sets

set.seed(456)

index <- sample(1:nrow(iris), 0.7 \* nrow(iris))

train\_data <- iris[index, ]

test\_data <- iris[-index, ]

# Build the SVM model

model\_svm <- svm(Species ~ ., data = train\_data, kernel = "linear")

# Predict on the test set

pred\_svm <- predict(model\_svm, test\_data)

# Evaluate the model with a confusion matrix

confusion\_matrix <- table(Predicted = pred\_svm, Actual = test\_data$Species)

print(confusion\_matrix)