SET-1

1. a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables

b. Predict weight for Time=10 and Diet=1 c. Find the error in model for same

chicken\_data <- data.frame(

Time = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),

Diet = c(1, 2, 1, 2, 1, 2, 1, 2, 1, 2),

Weight = c(5, 7, 6, 8, 7, 9, 8, 10, 9, 11)

)

print(chicken\_data)

model <- lm(Weight ~ Time + Diet, data = chicken\_data)

summary(model)

new\_data <- data.frame(Time = 10, Diet = 1)

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

print(predicted\_weight)

predicted\_weights <- predict(model)

rmse <- sqrt(mean((chicken\_data$Weight - predicted\_weights)^2))

print(paste("Root Mean Squared Error:", rmse))

2. (i)Get the Summary Statistics of air quality dataset (ii)Melt airquality data set and display as a long – format data? (iii) Melt airquality data and specify month and day to be “ID variables”? (iv)Cast the molten airquality data set with respect to month and date features (v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

library(reshape2)

summary(airquality)

melted\_data <- melt(airquality)

print(melted\_data)

melted\_data <- melt(airquality, id.vars = c("Month", "Day"))

print(melted\_data)

casted\_data <- dcast(melted\_data, Month + Day ~ variable)

print(casted\_data)

monthly\_avg <- dcast(melted\_data, Month ~ variable, mean)

print(monthly\_avg)

3. Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model.

set.seed(123)

sample\_indices <- sample(1:nrow(iris), size = 0.8 \* nrow(iris), replace = FALSE)

train\_data <- iris[sample\_indices, ]

test\_data <- iris[-sample\_indices, ]

library(glmnet)

model <- glm(Species ~ Petal.Width + Petal.Length, data = train\_data, family = "binomial")

test\_probabilities <- predict(model, newdata = test\_data, type = "response")

predicted\_class <- ifelse(test\_probabilities > 0.5, "virginica", "not\_virginica")

conf\_matrix <- table(test\_data$Species, predicted\_class)

print(conf\_matrix)

4. Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array.

col1 <- c(1, 2, 3)

col2 <- c(4, 5, 6)

col3 <- c(7, 8, 9)

col4 <- c(10, 11, 12)

my\_array <- array(data = c(col1, col2, col3, col4), dim = c(3, 4, 2))

print(my\_array)

SET-02

1.Create below data frame exam\_data = data.frame( name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'), score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19), attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1), qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes') )

exam\_data <- data.frame(

name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'),

score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19),

attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1),

qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes')

)

print(exam\_data)

2.Write a R program to create a list of elements using vectors, matrices and a function. Print itcontent of the list.

vector1 <- c(1, 2, 3, 4, 5)

vector2 <- c('a', 'b', 'c', 'd', 'e')

matrix1 <- matrix(1:9, nrow = 3, ncol = 3)

matrix2 <- matrix(letters[1:9], nrow = 3, ncol = 3)

my\_function <- function(x) { return(x^2) }

my\_list <- list(

"vector1" = vector1,

"vector2" = vector2,

"matrix1" = matrix1,

"matrix2" = matrix2,

"my\_function" = my\_function

)

print(my\_list)

3. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area. (i)

a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape c. Print the states with the max & min crime rates for murder

(ii) a. Find the correlation among the features b. Print the states which have assault arrests more than median of the country c. Print the states are in the bottom 25% of murder

(iii). a. Create a histogram and density plot of murder arrests by US stat b. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanized by state. Then enrich the chart by adding assault arrest rates (by coloring the points from blue (low) to red (high)). c. Draw a bar graph to show the murder rate for each of the 50 states

data("USArrests")

summary(USArrests)

features <- names(USArrests)

feature\_types <- sapply(USArrests, class)

print("Number of Features and their Types:")

print(data.frame(Features = features, Type = feature\_types))

print("Number of Records for Each Feature:")

print(sapply(USArrests, length))

print("Statistical Features of the Data:")

print(sapply(USArrests, function(x) c(mean(x), median(x), min(x), max(x))))

largest\_rape\_state <- rownames(USArrests)[which.max(USArrests$Rape)]

print("State with the Largest Total Number of Rape:")

print(largest\_rape\_state)

max\_murder\_state <- rownames(USArrests)[which.max(USArrests$Murder)]

min\_murder\_state <- rownames(USArrests)[which.min(USArrests$Murder)]

print("State with the Max Crime Rate for Murder:")

print(max\_murder\_state)

print("State with the Min Crime Rate for Murder:")

print(min\_murder\_state)

correlation\_matrix <- cor(USArrests)

print("Correlation Matrix:")

print(correlation\_matrix)

median\_assault <- median(USArrests$Assault)

states\_above\_median\_assault <- rownames(USArrests)[USArrests$Assault > median\_assault]

print("States with Assault Arrests More Than Median of the Country:")

print(states\_above\_median\_assault)

bottom\_25\_murder <- rownames(USArrests)[USArrests$Murder < quantile(USArrests$Murder, 0.25)]

print("States in the Bottom 25% of Murder:")

print(bottom\_25\_murder)

hist(USArrests$Murder, main = "Histogram of Murder Arrests", xlab = "Murder Arrests")

plot(density(USArrests$Murder), main = "Density Plot of Murder Arrests")

plot(USArrests$UrbanPop, USArrests$Murder, xlab = "Proportion Urban Population", ylab = "Murder Arrests", main = "Murder Arrests vs. Urban Population", col = "blue")

points(USArrests$UrbanPop, USArrests$Murder, col = heat.colors(n = length(USArrests$Assault))[rank(USArrests$Assault)], pch = 19)

barplot(USArrests$Murder, names.arg = rownames(USArrests), las = 2, main = "Murder Rate by State", xlab = "State", ylab = "Murder Rate")

4. Write a R program to read the .csv file and display the content

# Read the CSV file

data <- read.csv("your\_file.csv")

# Display the content of the CSV file

print(data)

SET -03

1. Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type

numeric\_vector <- c(1, 2, 3, 4, 5)

character\_vector <- c("apple", "banana", "orange", "grape", "kiwi")

logical\_vector <- c(TRUE, FALSE, TRUE, FALSE, TRUE)

print("Numeric Vector:")

print(numeric\_vector)

print("Character Vector:")

print(character\_vector)

print("Logical Vector:")

print(logical\_vector)

print("Type of Numeric Vector:")

print(class(numeric\_vector))

print("Type of Character Vector:")

print(class(character\_vector))

print("Type of Logical Vector:")

print(class(logical\_vector))

3. Write a R program to create a factor corresponding to height of women data set , which inbuild in R, contains height and weights for a sample of women.

data(women)

breaks <- c(50, 55, 60, 65, 70, 75)

height\_factor <- cut(women$height, breaks = breaks, labels = c("Short", "Below Average", "Average", "Above Average", "Tall"))

print(height\_factor)

4. Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns

matrix\_5x4 <- matrix(c(1:20), nrow = 5, ncol = 4, byrow = TRUE, dimnames = list(c("Row1", "Row2", "Row3", "Row4", "Row5"), c("Col1", "Col2", "Col3", "Col4")))

matrix\_3x3 <- matrix(c(1:9), nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2", "Col3")))

matrix\_2x2 <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = FALSE, dimnames = list(c("Row1", "Row2"), c("Col1", "Col2")))

print("Matrix 5x4:")

print(matrix\_5x4)

print("Matrix 3x3:")

print(matrix\_3x3)

print("Matrix 2x2:")

print(matrix\_2x2)

SET-04

1. Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension

values <- c(1, 2, 3, 4, 5, 6, 7, 8)

dimensions <- c(2, 2, 2)

dimnames\_list <- list(

c("row1", "row2"),

c("col1", "col2"),

c("depth1", "depth2")

)

my\_array <- array(values, dim = dimensions, dimnames = dimnames\_list)

print(my\_array)

1. Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column. remove the variables 'Solar.R' and 'Wind' and display the data frame

data(airquality)

if(is.data.frame(airquality)) {

print("airquality is a data frame.")

} else {

print("airquality is not a data frame.")

}

ordered\_airquality <- airquality[order(airquality$Month, airquality$Day), ]

print("Ordered Data Frame:")

print(ordered\_airquality)

modified\_airquality <- subset(ordered\_airquality, select = -c(Solar.R, Wind))

print("Modified Data Frame:")

print(modified\_airquality)

1. Load dataset named ChickWeight, ( i). Order the data frame, in ascending order by feature name “weight” grouped by feature “diet” and Extract the last 6 records from order data frame. (ii).a Perform melting function based on “Chick", "Time", "Diet" features as ID variables b. Perform cast function to display the mean value of weight grouped by Dit

data(ChickWeight)

ordered\_chick\_weight <- ChickWeight[order(ChickWeight$weight, ChickWeight$diet), ]

last\_6\_records <- tail(ordered\_chick\_weight, 6)

print("(i) Last 6 Records from Ordered Data Frame:")

print(last\_6\_records)

melted\_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

print("(ii) Melted Data:")

print(melted\_data)

casted\_data <- dcast(melted\_data, Diet ~ variable, mean)

print("(ii) Casted Data (Mean Value of Weight Grouped by Diet):")

print(casted\_data)

data(ChickWeight)

ordered\_chick\_weight <- ChickWeight[order(ChickWeight$weight, ChickWeight$diet), ]

last\_6\_records <- tail(ordered\_chick\_weight, 6)

print("(i) Last 6 Records from Ordered Data Frame:")

print(last\_6\_records)

melted\_data <- reshape2::melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))

print("(ii) a. Melted Data:")

print(melted\_data)

casted\_mean\_data <- reshape2::dcast(melted\_data, Diet ~ variable, mean)

print("(ii) b. Casted Data (Mean Value of Weight Grouped by Diet):")

print(casted\_mean\_data)

casted\_mode\_data <- reshape2::dcast(melted\_data, Diet ~ variable, function(x) {

table(x)[which.max(table(x))]

})

print("(ii) c. Casted Data (Mode of Weight Grouped by Diet):")

print(casted\_mode\_data)

1. Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model

data(iris)

set.seed(123)

train\_indices <- sample(1:nrow(iris), 0.8 \* nrow(iris))

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

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

library(nnet)

logistic\_model <- multinom(Species ~ Petal.Width + Petal.Length, data = train\_data)

predicted\_probabilities <- predict(logistic\_model, newdata = test\_data, type = "probs")

predicted\_classes <- colnames(predicted\_probabilities)[apply(predicted\_probabilities, 1, which.max)]

confusion\_matrix <- table(test\_data$Species, predicted\_classes)

print("Confusion Matrix:")

print(confusion\_matrix)

SET-05

1.User

Consider the following data present. Create this file using Windows Notepad. Save the file as input.csv using the save As All files(.) option in Notepad.

i. Use appropriate R commands to read the input.csv file.

ii. Analyze the CSV File and compute the following.

a. Get the maximum salary

b. Get the details of the person with max salary

c. Get all the people working in IT department

d. Get the persons in IT department whose salary is greater than 600

e. Get the people who joined on or after 2014

# Read the CSV file

data <- read.csv("input.csv")

# Display the data

print(data)

# Get the maximum salary

max\_salary <- max(data$Salary)

print(paste("Maximum salary:", max\_salary))

# Get the details of the person with max salary

person\_max\_salary <- data[data$Salary == max\_salary, ]

print("Details of the person with max salary:")

print(person\_max\_salary)

# Get all the people working in IT department

it\_department <- data[data$Department == "IT", ]

print("People working in IT department:")

print(it\_department)

# Get the persons in IT department whose salary is greater than 600

it\_high\_salary <- it\_department[it\_department$Salary > 600, ]

print("People in IT department with salary greater than 600:")

print(it\_high\_salary)

# Get the people who joined on or after 2014

joined\_after\_2014 <- data[data$JoiningYear >= 2014, ]

print("People who joined on or after 2014:")

print(joined\_after\_2014)

2.Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution)

set.seed(123)

sample\_letters <- sample(LETTERS, 20)

sample\_factor <- as.factor(sample\_letters)

five\_levels <- levels(sample\_factor)[1:5]

print("Five levels of the factor:")

print(five\_levels)

3.Write a R program to create a vector which contains 10 random integer values between - 50 and +50.

set.seed(123)

random\_integers <- sample(-50:50, 10, replace = TRUE)

print(random\_integers)

4.Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

fizz\_buzz <- function(num) {

if (num %% 3 == 0 && num %% 5 == 0) {

print("FizzBuzz")

} else if (num %% 3 == 0) {

print("Fizz")

} else if (num %% 5 == 0) {

print("Buzz")

} else {

print(num)

}

}

for (i in 1:100) {

fizz\_buzz(i)

}

SET-06

1. . Write a R program to find the factors of a given number

find\_factors <- function(num) {

factors <- numeric(0)

for (i in 1:num) {

if (num %% i == 0) {

factors <- c(factors, i)

}

}

return(factors)

}

number <- 24

result <- find\_factors(number)

print(paste("Factors of", number, "are:", result))

1. Write a R program to create a list of random numbers in normal distribution and count occurrences of each value.

set.seed(123)

random\_numbers <- rnorm(100, mean = 0, sd = 1) # Generate 100 random numbers with mean 0 and standard deviation 1

count\_occurrences <- table(round(random\_numbers, digits = 2))

print(count\_occurrences)

3. For this exercise, use the (built-in) dataset Titanic. a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class b. Modify the above plot based on gender of people who survived c. Draw histogram plot to show distribution of feature “Age”\

data(Titanic)

titanic\_df <- as.data.frame(Titanic)

barplot(table(titanic\_df$Class, titanic\_df$Survived), beside = TRUE, col = c("red", "green"),

main = "Survival of passengers based on Passenger Class",

xlab = "Passenger Class", ylab = "Count", legend = c("Died", "Survived"))

barplot(table(titanic\_df$Class, titanic\_df$Survived, titanic\_df$Gender), beside = TRUE, col = c("red", "green"),

main = "Survival of passengers based on Passenger Class and Gender",

xlab = "Passenger Class", ylab = "Count", legend = c("Male", "Female"), args.legend = list(x = "topright"))

hist(titanic\_df$Age, breaks = seq(0, 100, by = 5), col = "skyblue", border = "black",

main = "Distribution of Age of Titanic Passengers",

xlab = "Age", ylab = "Frequency")

4. Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third array

array1 <- matrix(1:4, nrow = 2, ncol = 2)

array2 <- matrix(5:8, nrow = 2, ncol = 2)

array3 <- matrix(9:12, nrow = 2, ncol = 2)

combined\_array <- rbind(array1[1,], array2[1,], array3[1,])

print(combined\_array)

SET-07

1. Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

# Given vectors

vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vector2 <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)

matrix1 <- matrix(vector1, nrow = 3, ncol = 3, byrow = TRUE)

matrix2 <- matrix(vector2, nrow = 3, ncol = 3, byrow = TRUE)

array <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

print("Second row of the second matrix of the array:")

print(array[2, , 2])

print("Element in the 3rd row and 3rd column of the 1st matrix:")

print(array[3, 3, 1])

2. Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array

vector1 <- c(1, 2, 3)

vector2 <- c(4, 5, 6)

vector3 <- c(7, 8, 9)

vector4 <- c(10, 11, 12)

vector5 <- c(13, 14, 15)

vector6 <- c(16, 17, 18)

matrix1 <- matrix(c(vector1, vector2, vector3), nrow = 3, byrow = TRUE)

matrix2 <- matrix(c(vector4, vector5, vector6), nrow = 3, byrow = TRUE)

array <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

print(array)

3. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months: • Ozone: mean ozone concentration (ppb), • Solar.R: solar radiation (Langley), • Wind: average wind speed (mph), • Temp: maximum daily temperature in degrees Fahrenheit, • Month: numeric month (May=5, June=6, and so on), • Day: numeric day of the month (1- 31). i. Compute the mean temperature(don’t use build in function) ii. Extract the first five rows from airquality. iii. Extract all columns from airquality except Temp and Wind iv. Which was the coldest day during the period? v. How many days was the wind speed greater than 17 mph?

data(airquality)

mean\_temp <- sum(airquality$Temp) / length(airquality$Temp)

print(paste("Mean temperature:", mean\_temp))

first\_five\_rows <- airquality[1:5, ]

print("First five rows from airquality:")

print(first\_five\_rows)

airquality\_subset <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

print("Subset of airquality without Temp and Wind columns:")

print(airquality\_subset)

coldest\_day <- airquality[which.min(airquality$Temp), "Day"]

print(paste("Coldest day during the period:", coldest\_day))

days\_greater\_than\_17mph <- sum(airquality$Wind > 17, na.rm = TRUE)

print(paste("Number of days with wind speed greater than 17 mph:", days\_greater\_than\_17mph))

4. Write a R program to sort a given data frame by name and score.

data <- data.frame(

name = c("John", "Alice", "Bob", "Emily"),

score = c(85, 92, 78, 90)

)

cat("Original Data Frame:\n")

print(data)

sorted\_data <- data[order(data$name, data$score),]

cat("\nSorted Data Frame by Name and Score:\n")

print(sorted\_data)

SET-08

1. Write a R program to draw an empty plot and an empty plot specify the axes limits of the graphic

plot.new()

title(main = "Empty Plot")

plot(NA, xlim = c(0, 10), ylim = c(0, 10), type = "n")

title(main = "Empty Plot with Specified Axes Limits")

2. Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension.

values <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

dims <- c(3, 3, 1) # Specify the dimensions as a vector

arr <- array(values, dim = dims, dimnames = list(

c("Row1", "Row2", "Row3"),

c("Col1", "Col2", "Col3"),

c("Table1")

))

print(arr)

3. (i) Get the Summary Statistics of air quality dataset (iii) Melt airquality data set and display as a long – format data? (iv) Melt airquality data and specify month and day to be “ID variables”? (v) Cast the molten airquality data set with respect to month and date features (vi) Use cast function appropriately

data(airquality)

summary\_stats <- summary(airquality)

print("(i) Summary Statistics of air quality dataset:")

print(summary\_stats)

library(reshape2)

melted\_data <- melt(airquality)

print("(iii) Melted airquality dataset (long-format data):")

print(head(melted\_data))

melted\_data\_id <- melt(airquality, id.vars = c("Month", "Day"))

print("(iv) Melted airquality dataset with Month and Day as ID variables:")

print(head(melted\_data\_id))

casted\_data <- dcast(melted\_data\_id, Month ~ Day)

print("(v) Casted airquality dataset with respect to month and date features:")

print(head(casted\_data))

mean\_ozone\_by\_month <- dcast(melted\_data, Month ~ variable, mean)

print("(vi) Mean ozone concentration by month:")

print(mean\_ozone\_by\_month)

. . a. Create a data frame based on below table b .C reate a regression model for that data frame table to show the amount of sales(Sales) based on the how much the company spends (Spends) in advertising c. Predict the Sales if Spend=13500

SET-09

1. Write a R program to create a factor corresponding to height of women data set , which inbuild in R, contains height and weights for a sample of women.

data(women)

height\_factor <- cut(women$height, breaks = c(50, 60, 70), labels = c("Short", "Medium", "Tall"))

print(height\_factor)

2. Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.

data("LETTERS")

set.seed(123) # Set seed for reproducibility

sample\_letters <- sample(LETTERS, 100, replace = TRUE)

sample\_factor <- factor(sample\_letters)

five\_levels <- levels(sample\_factor)[1:5]

print("Five levels of the factor:")

print(five\_levels)

3. Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

vector2 <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)

matrix1 <- matrix(vector1, nrow = 3, ncol = 3, byrow = TRUE)

matrix2 <- matrix(vector2, nrow = 3, ncol = 3, byrow = TRUE)

array <- array(c(matrix1, matrix2), dim = c(3, 3, 2))

print("Second row of the second matrix of the array:")

print(array[2, , 2])

print("Element in the 3rd row and 3rd column of the 1st matrix:")

print(array[3, 3, 1])

4. For this exercise, use the (built-in) dataset Titanic. a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class b. Modify the above plot based on gender of people who survived c. Draw histogram plot to show distribution of feature “Age

ALREADY WRITEN’

# Load the Titanic dataset

data(Titanic)

# Convert the dataset to a data frame

titanic\_df <- as.data.frame(Titanic)

# Load necessary libraries

library(ggplot2)

# a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class

ggplot(titanic\_df, aes(x = Class, fill = Survived)) +

geom\_bar(position = "dodge") +

labs(title = "Survival of passengers based on Passenger Class",

x = "Passenger Class", y = "Count") +

scale\_fill\_manual(values = c("died" = "red", "survived" = "green")) +

theme\_minimal()

# b. Modify the above plot based on gender of people who survived

ggplot(subset(titanic\_df, Survived == "survived"), aes(x = Class, fill = Gender)) +

geom\_bar(position = "dodge") +

labs(title = "Survival of passengers based on Passenger Class and Gender",

x = "Passenger Class", y = "Count") +

scale\_fill\_manual(values = c("male" = "blue", "female" = "pink")) +

theme\_minimal()

# c. Draw histogram plot to show distribution of feature “Age”

ggplot(titanic\_df, aes(x = Age)) +

geom\_histogram(binwidth = 5, fill = "skyblue", color = "black") +

labs(title = "Distribution of Age of Titanic Passengers",

x = "Age", y = "Frequency") +

theme\_minimal()

SET-10

1. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area. (i) a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape c. Print the states with the max & min crime rates for murder (ii). a. Find the correlation among the features b. Print the states which have assault arrests more than median of the country c. Print the states are in the bottom 25% of murder (iii). a. Create a histogram and density plot of murder arrests by US stat b. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanised by state. Then enrich the chart by adding assault arrest rates (by colouring the points from blue (low) to red (high)). c. Draw a bar graph to show the murder rate for each of the 50 states .

# Load the USArrests dataset

data("USArrests")

# (i) a. Explore the summary of Dataset

# Get summary of the dataset

summary\_data <- summary(USArrests)

cat("(i) Summary of Dataset:\n")

print(summary\_data)

# (i) b. Print the state which saw the largest total number of rape

largest\_rape\_state <- names(which.max(USArrests$Rape))

cat("\n(i) b. State with largest total number of rape:", largest\_rape\_state, "\n")

# (i) c. Print the states with the max & min crime rates for murder

max\_murder\_state <- names(which.max(USArrests$Murder))

min\_murder\_state <- names(which.min(USArrests$Murder))

cat("\n(i) c. State with max crime rate for murder:", max\_murder\_state, "\n")

cat("(i) c. State with min crime rate for murder:", min\_murder\_state, "\n")

# (ii) a. Find the correlation among the features

correlation\_matrix <- cor(USArrests)

cat("\n(ii) a. Correlation among the features:\n")

print(correlation\_matrix)

# (ii) b. Print the states which have assault arrests more than median of the country

median\_assault <- median(USArrests$Assault)

states\_above\_median\_assault <- names(USArrests$Assault[USArrests$Assault > median\_assault])

cat("\n(ii) b. States with assault arrests more than median of the country:\n")

print(states\_above\_median\_assault)

# (ii) c. Print the states that are in the bottom 25% of murder

bottom\_25\_percent\_murder <- names(USArrests$Murder[USArrests$Murder < quantile(USArrests$Murder, 0.25)])

cat("\n(ii) c. States in the bottom 25% of murder:\n")

print(bottom\_25\_percent\_murder)

# (iii) a. Create a histogram and density plot of murder arrests by US state

hist(USArrests$Murder, main = "Histogram of Murder Arrests by US State", xlab = "Murder Arrests")

# Density plot

plot(density(USArrests$Murder), main = "Density Plot of Murder Arrests by US State", xlab = "Murder Arrests")

# (iii) b. Create the plot showing the relationship between murder arrest rate and proportion of the population that is urbanised by state

urban\_population <- USArrests$UrbanPop

murder\_arrests <- USArrests$Murder

assault\_arrests <- USArrests$Assault

# Plotting

plot(urban\_population, murder\_arrests, col = heat.colors(length(urban\_population)),

main = "Relationship between Murder Arrest Rate and Urban Population",

xlab = "Proportion of Population Urbanized", ylab = "Murder Arrests")

points(urban\_population, assault\_arrests, col = heat.colors(length(urban\_population)), pch = 2)

# (iii) c. Draw a bar graph to show the murder rate for each of the 50 states

barplot(USArrests$Murder, names.arg = row.names(USArrests), main = "Murder Rate for Each US State",

xlab = "State", ylab = "Murder Rate")

2. Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model

# Load the iris dataset

data(iris)

# Set seed for reproducibility

set.seed(123)

# Randomly sample the dataset into training and testing sets

train\_indices <- sample(1:nrow(iris), 0.8 \* nrow(iris)) # 80% for training

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

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

# Fit logistic regression model with training data

model <- glm(Species ~ Petal.Length + Petal.Width, data = train\_data, family = "binomial")

# Predict probabilities using test data

test\_probabilities <- predict(model, test\_data, type = "response")

# Convert predicted probabilities to class predictions

test\_predictions <- ifelse(test\_probabilities > 0.5, "versicolor", "not versicolor")

# Create confusion matrix

confusion\_matrix <- table(test\_predictions, test\_data$Species)

print("Confusion Matrix:")

print(confusion\_matrix)

3. Write a R program to save the information of a data frame in a file and display the information of the file.

# Create a sample data frame

df <- data.frame(

ID = c(1, 2, 3, 4, 5),

Name = c("Alice", "Bob", "Charlie", "David", "Eva"),

Age = c(25, 30, 35, 40, 45)

)

# Save the data frame to a CSV file

write.csv(df, "data\_frame\_info.csv", row.names = FALSE)

# Display the information of the file

file\_info <- file.info("data\_frame\_info.csv")

print("Information of the saved file:")

print(file\_info)

4. Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column. remove the variables 'Solar.R' and 'Wind' and display the data frame

# Load the airquality dataset

data(airquality)

# Check if airquality is a data frame

is\_data\_frame <- is.data.frame(airquality)

print("Is airquality a data frame?")

print(is\_data\_frame)

# Order the entire data frame by the first and second column

ordered\_airquality <- airquality[order(airquality$Month, airquality$Day), ]

# Remove the variables 'Solar.R' and 'Wind'

cleaned\_airquality <- subset(ordered\_airquality, select = -c(Solar.R, Wind))

# Display the cleaned data frame

print("Cleaned data frame:")

print(cleaned\_airquality)