**DAY-3 ASSESSMENT**

**ITA0443-STATISTICS WITH R PROGRAMMING**

**I.REVANTH REDDY**

**191911268**

**1. (i) Write a function in R programming to print generate Fibonacci sequence using  Recursion in R.**

fibonacci <- function(n) {

if (n <= 0) {

return(0)

}

if (n == 1) {

return(1)

}

return (fibonacci(n - 1) + fibonacci(n - 2))

}

**(ii) Find sum of natural numbers up-to 10, without formula using loop statement.**

sum <- 0

for (i in 1:10) {

sum <- sum + i

}

print(sum)

**(iii) create a vector 1:10 and Find a square of each number and store that in a**

**separate list.**

numbers <- 1:10

squared\_numbers <- numeric(length(numbers))

for (i in 1:length(numbers)) {

squared\_numbers[i] <- numbers[i]^2

}

squared\_numbers

**2.    (motor trend car road test) comprises fuel consumption, performance and  10 aspects of automobile design for 32 automobiles. It comes pre-installed  with  package in R.**

**(i)Find the dimension of the dataset**

mtcars <- mtcars

dim(mtcars)

**(ii)Give the statistical summary of the features.**

summary(mtcars)

**(iii)Print the categorical features in Dataset**

cat\_cols <- sapply(mtcars, is.factor)

cat\_cols

**(iv)Find the average weight(wt) grouped by Engine shape(vs)**

library(dplyr)

mtcars %>%

group\_by(vs) %>%

summarise(mean\_wt = mean(wt))

**(v)Find the largest and smallest value of the variable weight with respect to Engine shape**

mtcars %>%

group\_by(vs) %>%

summarise(min\_wt = min(wt), max\_wt = max(wt))

**3.Use ggplot package to plot below EDA questions label the plot accordingly**

**(i)Create weight(wt) vs displacement(disp) scatter plot factor by  Engine Shape(vs)**

ggplot(mtcars, aes(x = wt, y = disp, color = factor(vs))) +

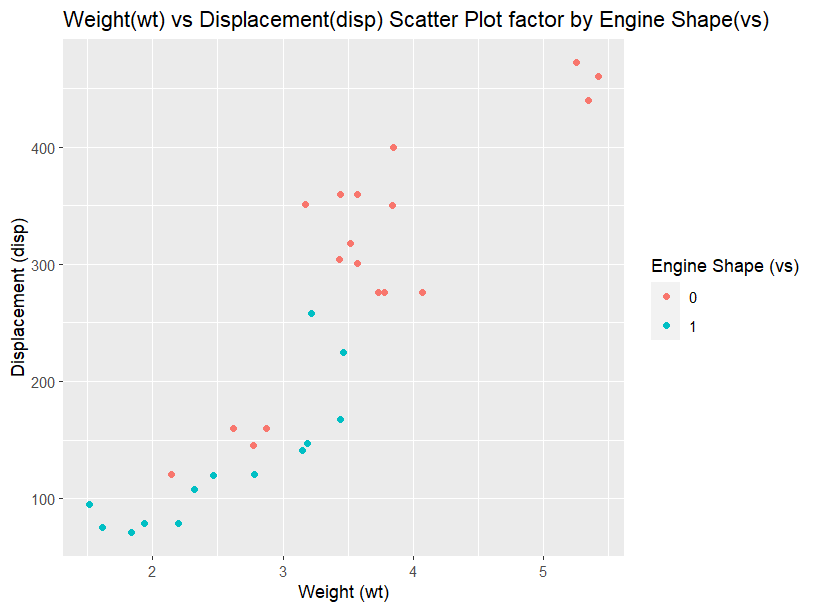
geom\_point() +

ggtitle("Weight(wt) vs Displacement(disp) Scatter Plot factor by Engine Shape(vs)") +

xlab("Weight (wt)") +

ylab("Displacement (disp)") +

scale\_color\_discrete(name = "Engine Shape (vs)")



**(ii) Create horsepower(hp) vs mileage (mgp) scatter plot factor by  Engine Shape(vs)**

ggplot(mtcars, aes(x = hp, y = mpg, color = factor(vs))) +

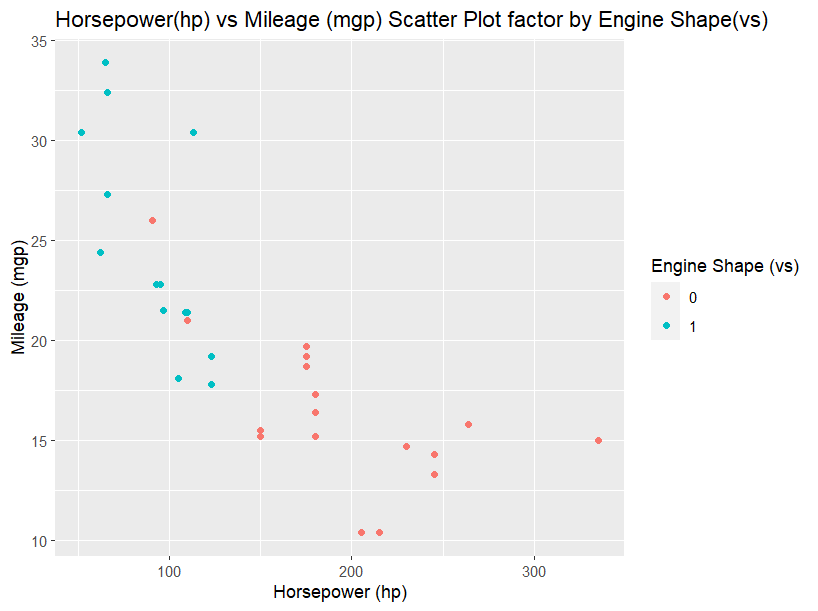
geom\_point() +

ggtitle("Horsepower(hp) vs Mileage (mgp) Scatter Plot factor by Engine Shape(vs)") +

xlab("Horsepower (hp)") +

ylab("Mileage (mgp)") +

scale\_color\_discrete(name = "Engine Shape (vs)")



**(iv)In above plot , Separate columns according to cylinders(cyl) size**

ggplot(mtcars, aes(x = hp, y = mpg, color = factor(cyl))) +

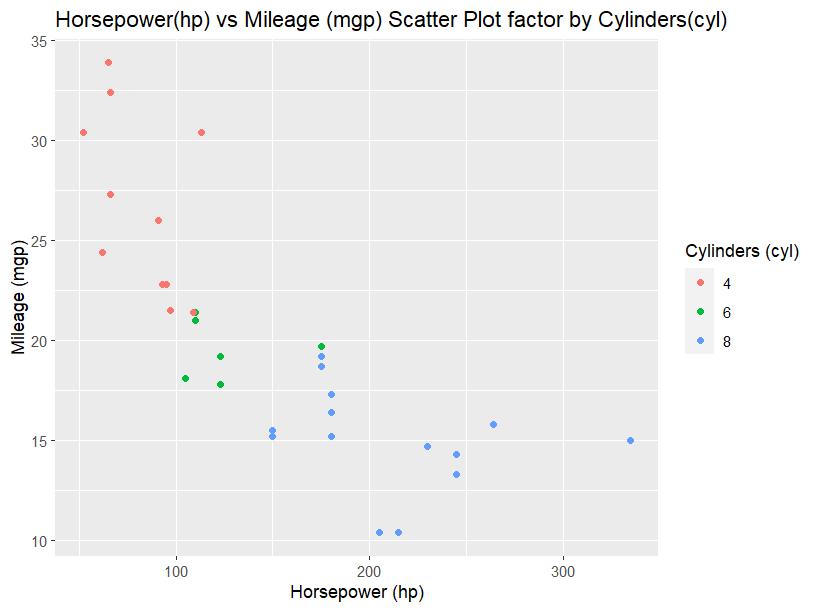
geom\_point() +

ggtitle("Horsepower(hp) vs Mileage (mgp) Scatter Plot factor by Cylinders(cyl)") +

xlab("Horsepower (hp)") +

ylab("Mileage (mgp)") +

scale\_color\_discrete(name = "Cylinders (cyl)")



**(v) Create histogram plot for horsepower (hp) with bin-width size of 5**

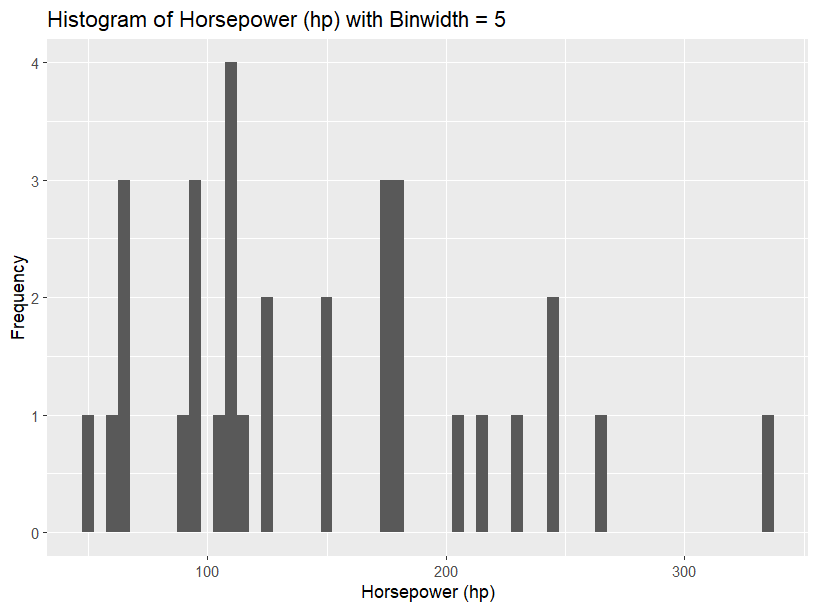
ggplot(mtcars, aes(x = hp)) +

geom\_histogram(binwidth = 5) +

ggtitle("Histogram of Horsepower (hp) with Binwidth = 5") +

xlab("Horsepower (hp)") +

ylab("Frequency")



**4. Performing Logistic regression on dataset to predict the cars Engine shape(vs) .**

**(i)Do the EDA analysis and find the features which is impact the Engine shape and use this for model.**

library(ggplot2)

# Plotting the relationship between engine shape (vs) and weight (wt)

ggplot(mtcars, aes(x = wt, y = vs, color = vs)) +

geom\_point() +

labs(title = "Weight vs Engine Shape",

x = "Weight",

y = "Engine Shape")

# Plotting the relationship between engine shape (vs) and horsepower (hp)

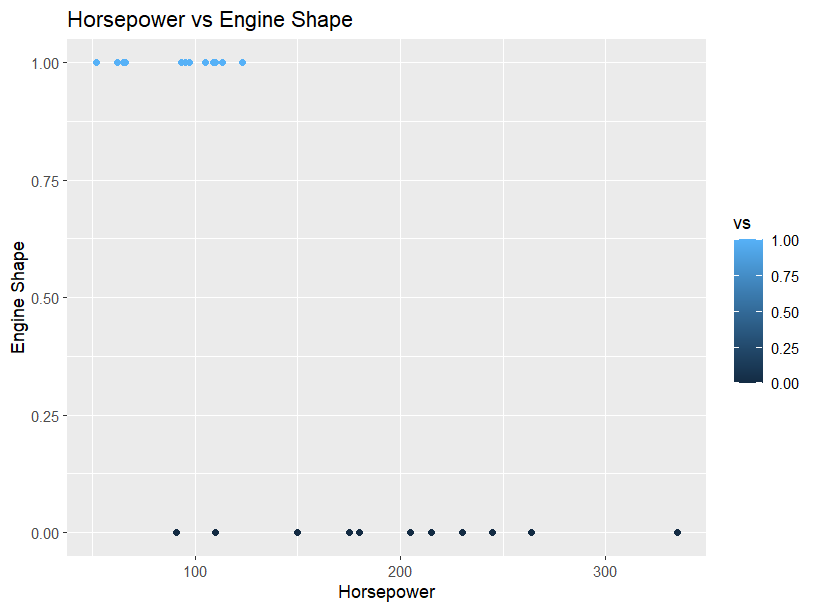
ggplot(mtcars, aes(x = hp, y = vs, color = vs)) +

geom\_point() +

labs(title = "Horsepower vs Engine Shape",

x = "Horsepower",

y = "Engine Shape")



**(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create logistic model**

# Splitting the data into train and test datasets

set.seed(123)

trainIndex <- sample(1:nrow(mtcars), 0.8 \* nrow(mtcars))

trainData <- mtcars[trainIndex, ]

testData <- mtcars[-trainIndex, ]

# Fitting a logistic regression model using weight (wt) and horsepower (hp) as predictor variables

model <- glm(vs ~ wt + hp, data = trainData, family = binomial)

# Predicting engine shape on the test dataset

predictions <- predict(model, testData, type = "response")

# Converting predictions to binary values (1 or 0) based on a threshold value of 0.5

predictionsBinary <- ifelse(predictions >= 0.5, 1, 0)

**(iii)Create the Confusion matrix among prediction and test data.**

confusionMatrix <- table(testData$vs, predictionsBinary)

confusionMatrix

**OUTPUT:**

predictionsBinary

0 1

0 4 1

1 0 2

**5. (I) Write R Program to create 15 x15 matrix filled with random numbers between -10 to 10, numbers can repeat. set random seed value to 328**

# Set random seed value to 328

set.seed(328)

# Create a 15 x 15 matrix filled with random numbers

matrix <- matrix(runif(225, min = -10, max = 10), nrow = 15, ncol = 15)

**(ii)Write R Program to display Lower Diagonal and upper Diagonal matrix**

# Display lower diagonal of the matrix

lower\_diagonal <- diag(matrix, k = -1)

print(lower\_diagonal)

# Display upper diagonal of the matrix

upper\_diagonal <- diag(matrix, k = 1)

print(upper\_diagonal)

**(iii)Write R Program to count 0’Sin the matrix and check the matrix is sparse matrix or not**

# Count the number of zeros in the matrix

zero\_count <- length(which(matrix == 0))

# Check if the matrix is sparse

if (zero\_count / length(matrix) >= 0.9) {

print("The matrix is sparse")

} else {

print("The matrix is not sparse")

}

**(iv) Write R code to remove outliers. Here the outliers are negative numbers. replace the negative values with positive values**

matrix[matrix < 0] <- abs(matrix[matrix < 0])

**(v) Find the mean median and mode of the values corresponding to column**

column\_means <- colMeans(matrix)

column\_medians <- apply(matrix, 2, median)

column\_modes <- sapply(1:ncol(matrix), function(i) {

u <- unique(matrix[, i])

u[which.max(tabulate(match(matrix[, i], u)))]

})

**(vi)Find the mean median and mode of the values corresponding to row**

# Find the mean, median and mode of the values corresponding to a row

row\_means <- rowMeans(matrix)

row\_medians <- apply(matrix, 1, median)

row\_modes <- sapply(1:nrow(matrix), function(i) {

u <- unique(matrix[i, ])

u[which.max(tabulate(match(matrix[i, ], u)))]

})