

**DEPARTMENT OFARTIFICIALINTELLIGENCE AND DATA SCIENCE LAB MANUAL**

**AD23431-STATISTICALANALYSISANDCOMPUTING**

**(REGULATION2023)**

**RAJALAKSHMIENGINEERINGCOLLEGE**

**Thandalam,Chennai-602015**

Name: Allwin Joseph A Register No: 231801008

Year/ Branch/ Section: 2nd / AI&DS / FA Semester: IV

AcademicYear:2024-2025

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | Date | Title | Page No |
| 1. | 05/2/25 | ImplementSimpleProgramsinR | 3 |
| 2. | 19/2/25 | PerformDataPreprocessinginR | 9 |
| 3. | 05/3/25 | PerformStatisticalAnalysisforaGivenDataset | 12 |
| 4. | 26/3/25 | ImplementDecisionTreeAlgorithminR | 17 |
| 5. | 02/4/25 | ImplementK-NearestNeighborAlgorithminR | 21 |
| 6. | 16/4/25 | ImplementNaiveBayesianClassifierinR | 26 |
| 7. | 16/4/25 | ImplementLinearRegressioninR | 30 |
| 8. | 23/4/25 | ImplementK-meansClusteringAlgorithminR | 33 |

**IMPLEMENTSIMPLEPROGRAMSINR**

**EXPNO:1**

# Aim:

ToImplementSimpleProgramsusingR.

**Algorithm:**

1. **BasicArithmeticOperations**
   1. **FindingAreaofCircle**
      * Input:Read radiusr.
      * Process:Calculatetheareausingtheformula:

Area=π×r2\text{Area}= \pi\timesr^2Area=π×r2

* + - Output:Printthecalculatedarea.

# ControlStructures(if-else,forloop)

* 1. **CheckWhethertheGivenYearisLeaporNot**
     + Input:Reada yearly.
     + Process:
       - Iflyisdivisibleby400,it’saleapyear.
       - Else,ifdivisibleby100(butnotby400),it’snotaleapyear.
       - Else,ifdivisibleby4,it’saleapyear.
       - Otherwise,it’snotaleapyear.
     + Output:Printwhethertheyearisaleapyearornot.

# ReverseaGivenNumber

* + - Input:Read anumbernum.
    - Process:
      * Initializerev=0.
      * Whilenum>0:
        + Extractlastdigit:ld= num%10.
        + Updaterev=rev\*10+ld.
        + Removelastdigit:num= num//10.
    - Output:Printthereversednumber.

# FindingPrimeNumbersfortheGivenRange

* + - Input:Readthenumbern(upperlimit).
    - Process:
      * Foreachnumberifrom1ton,checkifit'sprime:
        + Ifdivisiblebyanynumberfrom2to√i,it’snotprime.
        + Ifnodivisorsfound,itisprime.
    - Output:Printallprimenumbersfrom1ton.

# FunctionsandRecursiveFunctions

* 1. **Print theFibonacciSequenceusing Functions(Iterative)**
     + Input:Readn(numberoftermsinthesequence).
     + Process:
       - Initializefirsttwoterms:a= 0,b=1.
       - Printaandb.
       - Loop(n-2)times:
         * Calculatenexttermc=a+b.
         * Updatea=b, b =c.
       - Printthesequenceofnterms.

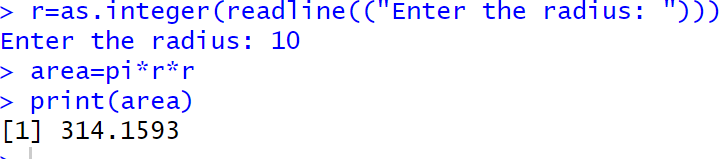
# PrinttheFibonacciSequenceusing RecursiveFunctions

* + - Input:Readn(numberoftermsinthesequence).
    - Process:
      * Definearecursivefunctionfibo(n):
        + Ifn==0,return0(basecase).
        + Ifn==1,return1(basecase).
        + Else,returnfibo(n-1)+fibo(n-2).
      * Callfibo(i)foreachifrom0ton-1andprintthesequence.

# Programs:

1. **BasicArithmeticOperations**
   1. **Finding Area of Circle** r=as.integer(readline(("Enter the radius: "))) area=pi\*r\*r

print(area)

**Output:**

1. **ControlStructure(if-else,forloop)**
   1. **To Check Whether the Given Year is Leap or Not** ly=as.integer(readline(("Enter a Number: "))) if(ly%%400==0){

print("LeapYear")

}else if(ly%%100==0){ print("Not a Leap Year")

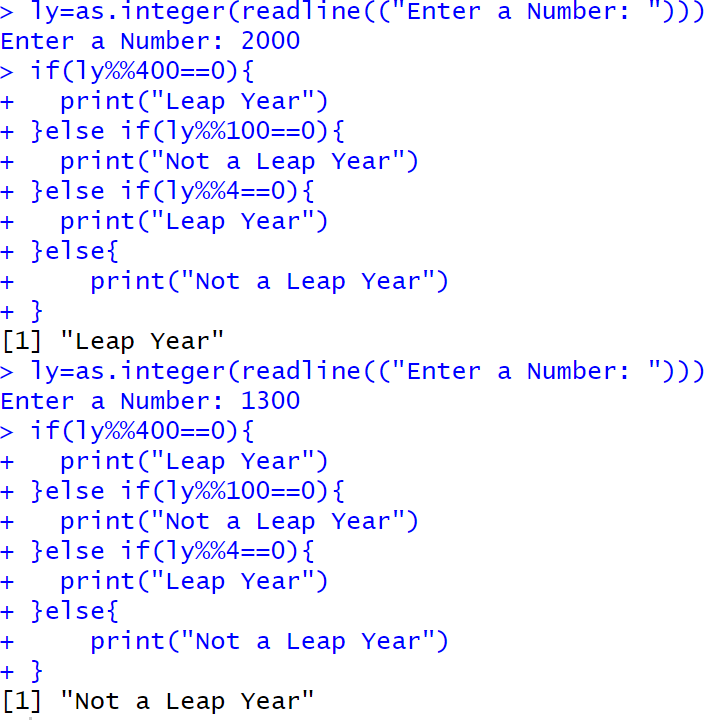
}else if(ly%%4==0){ print("Leap Year")

}else{

print("NotaLeapYear")

}

**Output:**

****

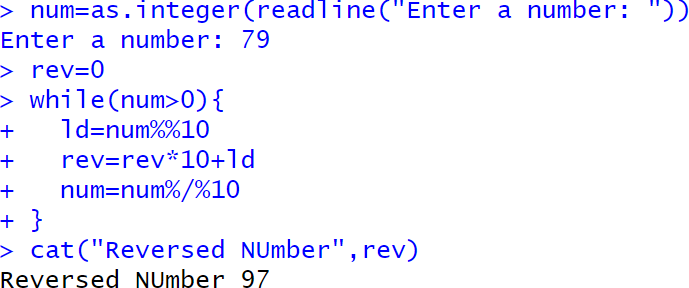
* 1. **Reverse a Given Number** num=as.integer(readline("Enter a number: "))rev=0

while(num>0){ ld=num%%10 rev=rev\*10+ld num=num%/%10

}

cat("ReversedNUmber",rev)

**Output:**

****

* 1. **FindingPrimeNumbersfortheGivenRange**

prime<-function(n){ if(n<=1){

return (FALSE)} for(iin2:sqrt(n)){

if(n%%i==0){ return (FALSE)

}

}

return(TRUE)

}

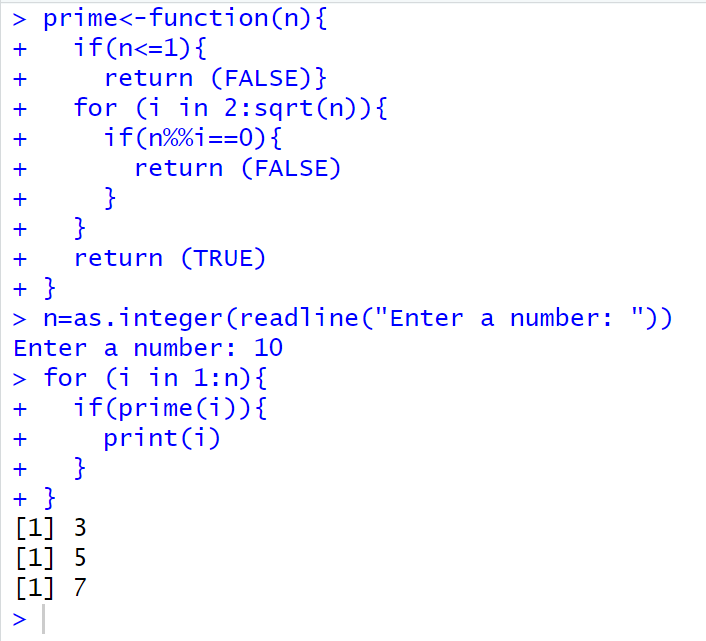
n=as.integer(readline("Enter a number: ")) for (i in 1:n){

if(prime(i)){ print(i)

}

}

**Output:**

****

1. **FunctionsandRecursiveFunctions**
   1. **Print the Fibonacci Sequence using Functions** fibonacci\_iterative<-function(n){fib\_series <- numeric(n)

fib\_series[1]<-0

if(n>1)fib\_series[2]<-1

for(iin3:n){

fib\_series[i]<-fib\_series[i-1]+fib\_series[i-2]

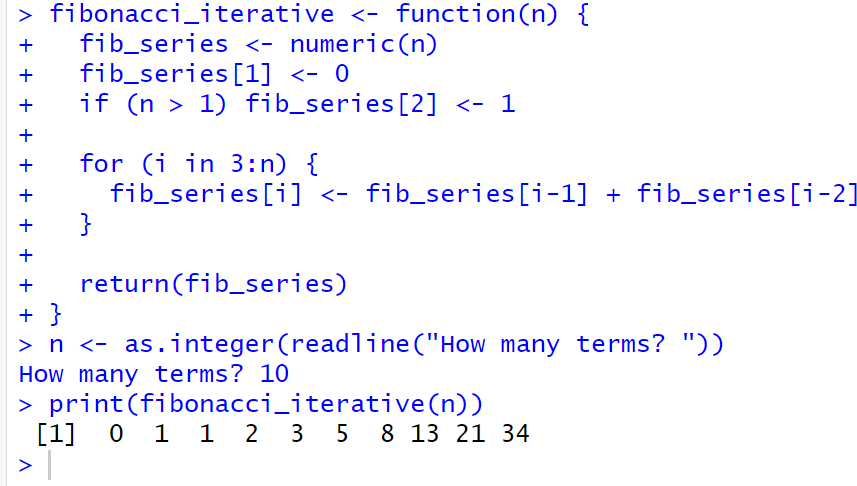
}

return(fib\_series)

}

n <- as.integer(readline("How many terms? "))print(fibonacci\_iterative(n))

**Output:**

****

* 1. **PrinttheFibonacciSequenceusingRecursiveFunctions**

fibonacci\_recursive <- function(n) {if (n == 1) {

return(0)

} else if (n == 2) { return(1)

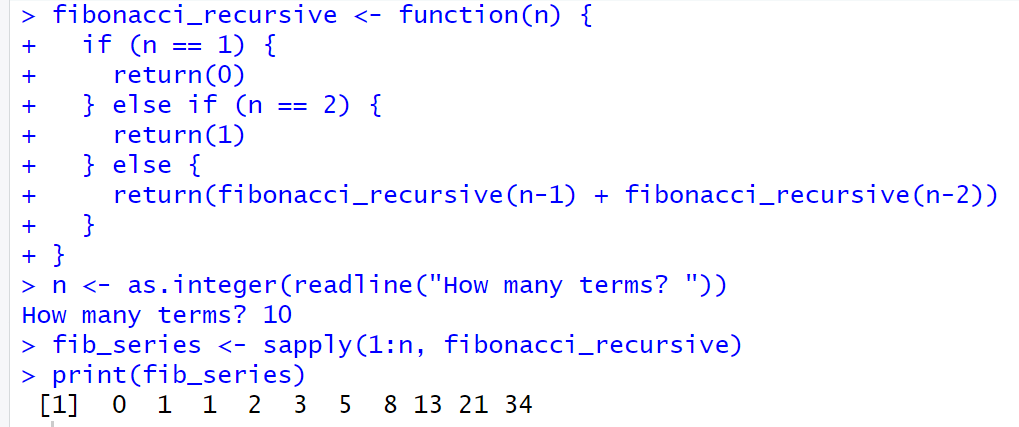
}else{

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

}

}

n <- as.integer(readline("How many terms? "))fib\_series <- sapply(1:n, fibonacci\_recursive) print(fib\_series)

**Output:**

# Result:

TheSimpleProgramusingRisSuccessfullyImplemented.

**PERFORMDATAPREPROCESSINGINR**

**EXPNO:2**

# Aim:

ToPerformPreprocessingofdatausingR.

# Algorithm:

1. **LoadingData/Cleaningthe Data:**
   * Createemp\_df2withcolumns:emp\_id,age,dept,salary,experience.

# Storing/Uploading Datato ExcelSheet:

* + Create a workbook wb, add a worksheet "Employee Data Preprocessing", and save emp\_df2 to emp\_df2.xlsx.

# CleaningtheData:

* + Replacemissingageandsalarywiththeirrespectivemeanvalues.
  + Convertdepttonumeric.

# ScalingtheData:

* + Scale the age, salary, and experience columns using z-score and update emp\_df2.

# SplittingtheDataintoTrainandTest:

* + Setseed,splitdatainto80%trainand20%test(dataTrain,dataTest).

# CorrelationMatrix:

* + Compute the correlation matrix for the scaled features (age, salary, experience) to examine relationships between them.

# Programs:

library(openxlsx)

emp\_df2<-data.frame( emp\_id=1:10,

age=c(25,30,35,NA,55,65,NA,25,85,78),dept=c("AI&DS","IT","AI&ML","CSE","PHY","FT","BIOTECH","CSBS","CIVIL","MECH"),salary=c(50000,85100,52802,144510,552410,520000,445100,5552410,524160,NA),

experience=c(2,5,8,14,4,6,3,2,4,5)

)

wb<-createWorkbook()addWorksheet(wb,"Employee Data Preprocessing")

writeData(wb,"EmployeeDataPreprocessing",emp\_df2)

saveWorkbook(wb,"C:\\Users\\karthick.S\\OneDrive\\Documents\\231801079-4\\SAC\\emp\_df2.xlsx",overwrite = TRUE)

emp\_df2$age[is.na(emp\_df2$age)]<-floor(mean(emp\_df2$age,na.rm= TRUE))emp\_df2$salary[is.na(emp\_df2$salary)]<-floor(mean(emp\_df2$salary,na.rm=TRUE))emp\_df2$dept<-as.numeric(as.factor(emp\_df2$dept))

emp\_df\_scaled<-scale(emp\_df2[,c("age","salary","experience")])emp\_df2<-data.frame(emp\_df2[,c("emp\_id","dept")],emp\_df\_scaled)

correlation\_matrix <- cor(emp\_df2[, c("age", "salary", "experience")])print("Correlation Matrix:")

print(correlation\_matrix)

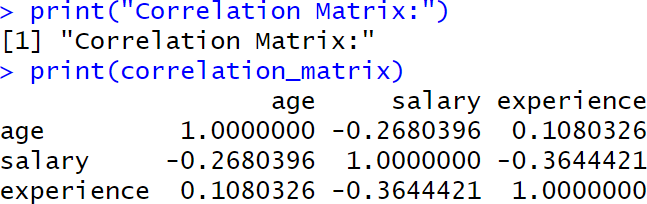
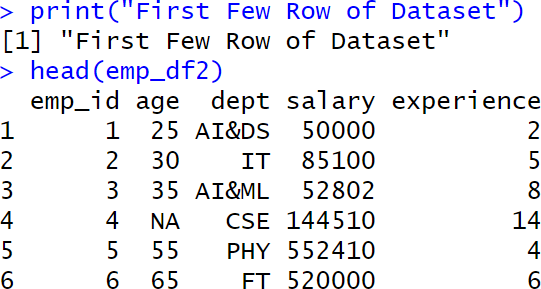
set.seed(42)

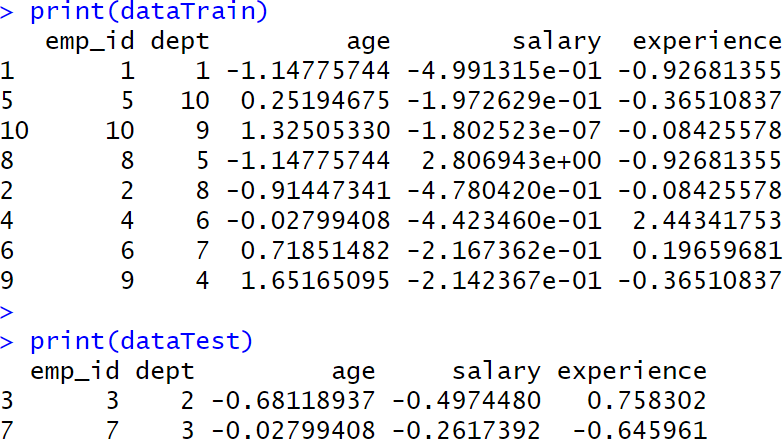
trainIndex<-sample(1:nrow(emp\_df2),0.8\*nrow(emp\_df2))dataTrain<-emp\_df2[trainIndex,]

dataTest<-emp\_df2[-trainIndex,]

print(dataTrain) print(dataTest)

# Output:

****



**Result:**

Thus,Preprocessingdataiscleaned,transformedandformatteddatasetreadyforanalysisor modelling.

**PERFORM STATISTICAL ANALYSIS FOR A GIVEN DATASET**

**EXPNO:3**

# Aim:

ToPerformStatisticalAnalysisforGivenDataset.

# Algorithm:

1. **LoadingLibraries:**
   * Loadthenecessarylibraries:dplyr,summarytools,psych.

# LoadingData:

* + CreateadatasetdatawithcolumnsAgeandSalary.

# StatisticalAnalysis:

* + Mean:CalculatethemeanofAge.
  + Median:CalculatethemedianofAge.
  + Mode:CalculatethemodeofAgeusingthetablefunction.
  + Variance:CalculatethevarianceofAge.
  + StandardDeviation:CalculatethestandarddeviationofAge.
  + Correlation:CalculatethecorrelationbetweenAgeandSalary.

# DescriptiveStatistics:

* + Usethesummary()functiontogeneratesummarystatisticsforthedataset.

# QuantileAnalysis:

* + CalculatethequantilesforbothAgeandSalary.

# InterquartileRange(IQR):

* + CalculatetheIQRforbothAgeandSalary.

# HypothesisTesting(T-Test):

* + Performaone-samplet-testonSalarywithahypothesizedmeanof70,000.

# Visualization:

* + Boxplot:CreateaboxplotforAgeandSalarytovisualizetheirdistributions.

# DetailedDescriptiveStatistics:

* + Usedescribe()fromthepsychpackagetogetdetailedstatisticsforAgeandSalary.
  + Usedescr()fromthesummarytoolspackagefordetaileddescriptivestatistics.

# Program:

library(dplyr) library(summarytools) library(psych)

data<-data.frame(Age=c(25,30,28,35,40,45,50,32,38,42),

Salary=c(50000,60000,55000,75000,80000,85000,90000,65000,78000,

82000))

cat("Dataset:\n")

print(data)

mean\_age <- mean(data$Age) median\_age <- median(data$Age)

mode\_age<-as.numeric(names(sort(table(data$Age),decreasing=TRUE))[1])

var\_age<-var(data$Age) sd\_age <- sd(data$Age)

corr<-cor(data$Age,data$Salary)

cat("\nStatistical Analysis Results:\n") print(mean\_age)

print(median\_age) print(mode\_age) print(var\_age) print(sd\_age) print(corr)

data\_summary <- summary(data) print(data\_summary)

quantile\_age <- quantile(data$Age) quantile\_salary <- quantile(data$Salary)

IQR\_age <- IQR(data$Age) IQR\_salary<-IQR(data$Salary)

cat("Quantile Age", quantile\_age) cat("\nQuantile Salary", quantile\_salary)

cat("\nIQR Age", IQR\_age) cat("\nIQRSalary",IQR\_salary)

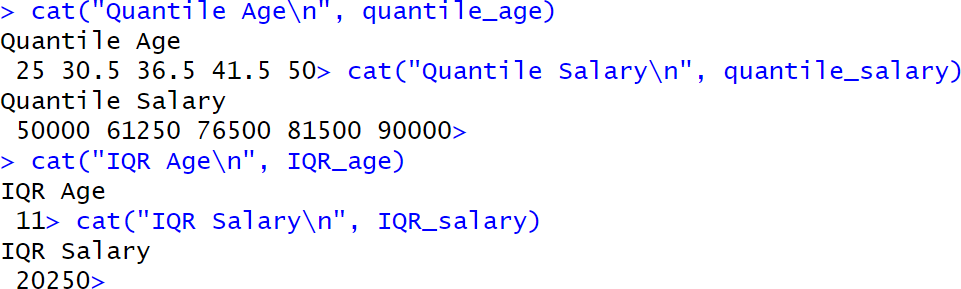
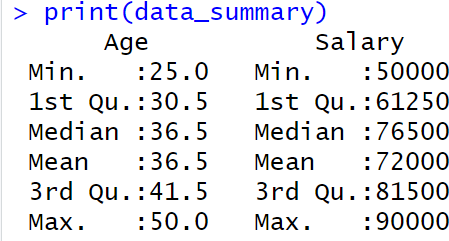
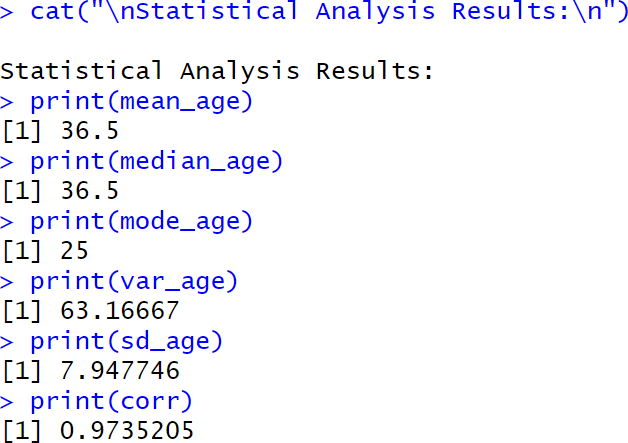
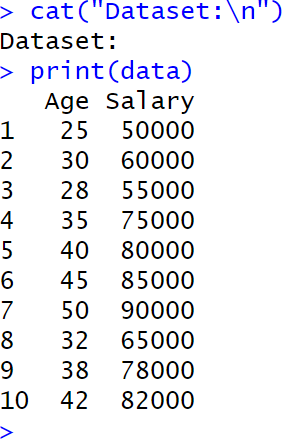
t\_test\_result <- t.test(data$Salary, mu = 70000) print(t\_test\_result)

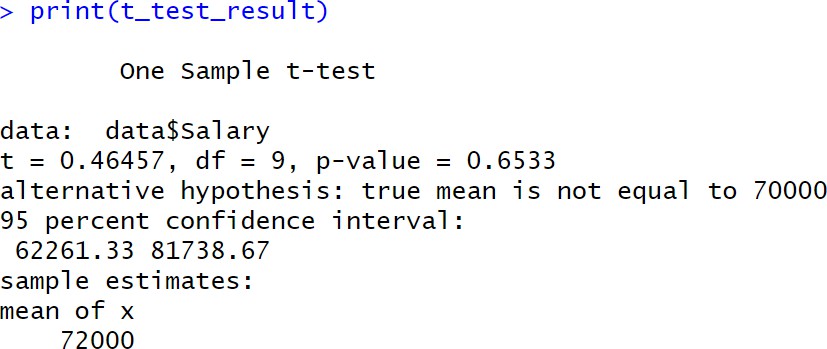
boxplot(data$Age, main = "Boxplot of Age", ylab = "Age", col = "lightblue") boxplot(data$Salary, main = "Boxplot of Salary", ylab = "Salary", col = "lightgreen")

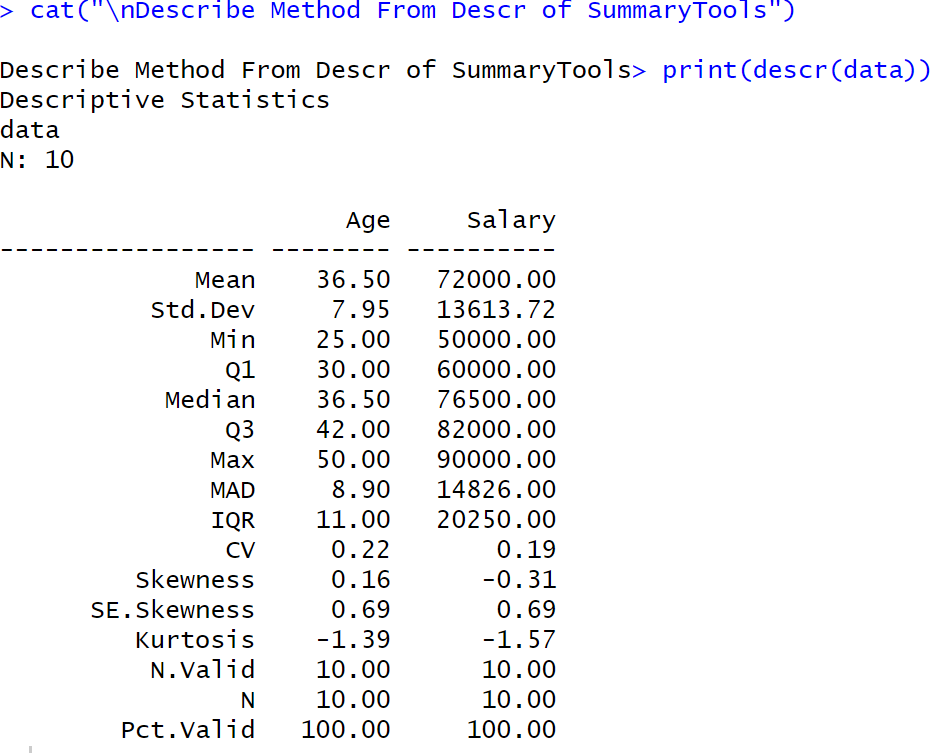
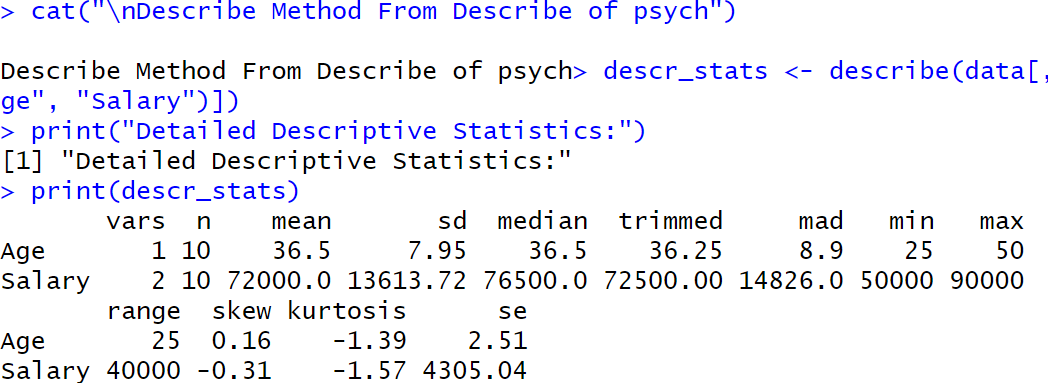
cat("\nDescribe Method From Describe of psych") descr\_stats <- describe(data[, c("Age", "Salary")]) print("Detailed Descriptive Statistics:") print(descr\_stats)

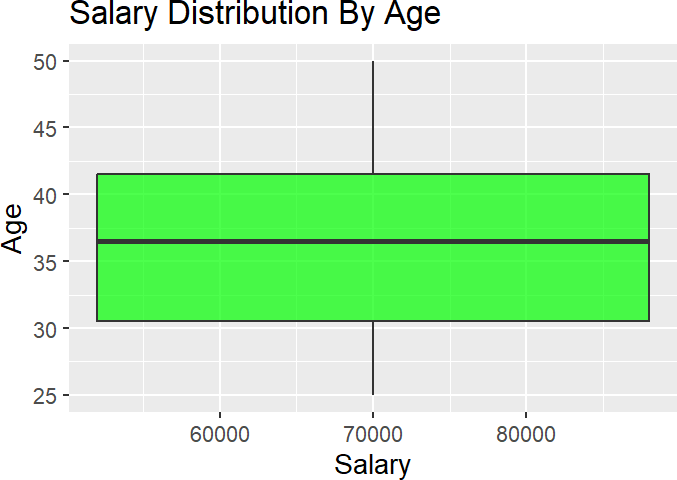
cat("\nDescribe Method From Descr of SummaryTools") print(descr(data))

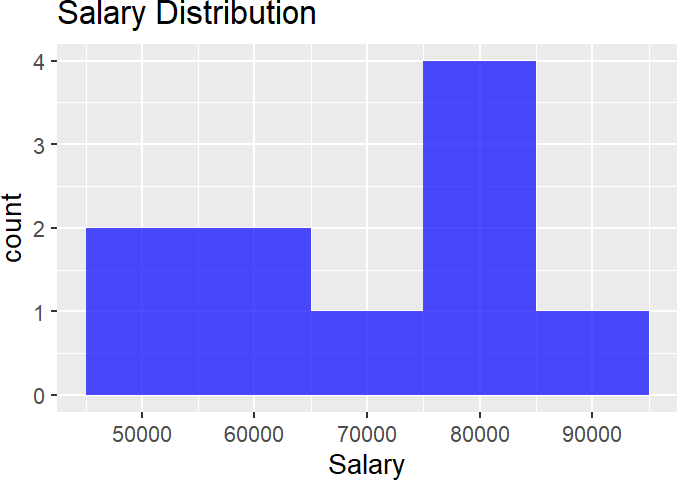
# Output:

****



****



****

**Result:**

Thus,StatisticalAnalysisforaGivenDatasetusingisAnalysedandScaled.

**IMPLEMENTDECISIONTREEALGORITHMINR**

**EXPNO:4**

# Aim:

ImplementaDecisionTreeClassificationontheGivenDataset.

# Procedure:

1. **LoadRequiredLibraries**
   * Loadthenecessarylibraries:
     + rpartforbuildingdecisiontreemodels.
     + rpart.plotforvisualizingdecisiontrees.
     + caretfordatasplittingandmodelevaluation.

# Code:

library(rpart) library(rpart.plot) library(caret)

# Loadthe Dataset

* + LoadtheIrisdataset(built-ininR).
  + Displaythefirstfewrowstounderstandthedatastructure.

# Code:

data("iris")

print("First FewRows of Dataset") head(iris)

# SplittheDataintoTrainingand TestingSets

* + Setaseedforreproducibility.
  + UsecreateDataPartitiontosplitthedatainto:
    - 80%trainingset
    - 20%testingset

# Code:

set.seed(123)

train\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE) train\_data <- iris[train\_index, ]

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

# TrainaDecisionTreeModel

* + Buildadecisiontreeclassifierusingrpart,predictingSpeciesbasedonthefeatures.

# Code:

tree\_model <- rpart(Species ~ ., data = train\_data, method = "class") print(tree\_model)

# VisualizetheDecisionTree

* + Plotthetraineddecisiontreeusingrpart.plotwithenhancedformatting.

# Code:

rpart.plot(tree\_model,

main ="Decision Tree for Iris Dataset", type = 3,

extra = 101, under=TRUE, tweak = 1.2,

box.palette="RdBu")

# MakePredictionsonTestData

* + Usethetrainedmodeltopredictthespeciesonthetestdataset.

# Code:

pred<-predict(tree\_model,test\_data,type="class")

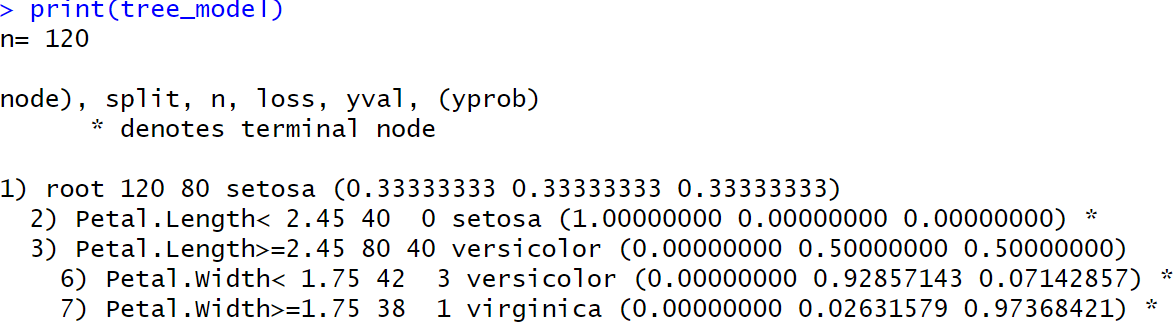
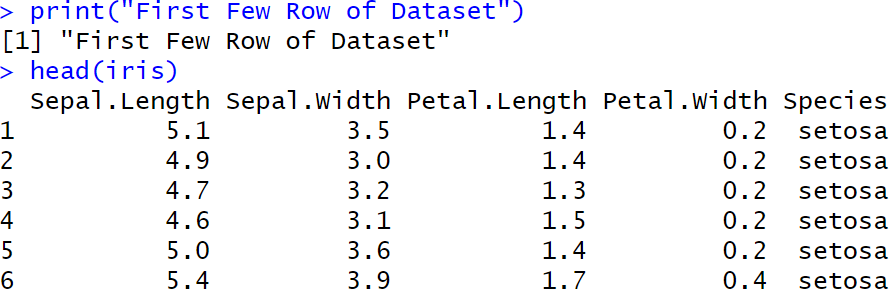
# EvaluateModelPerformance

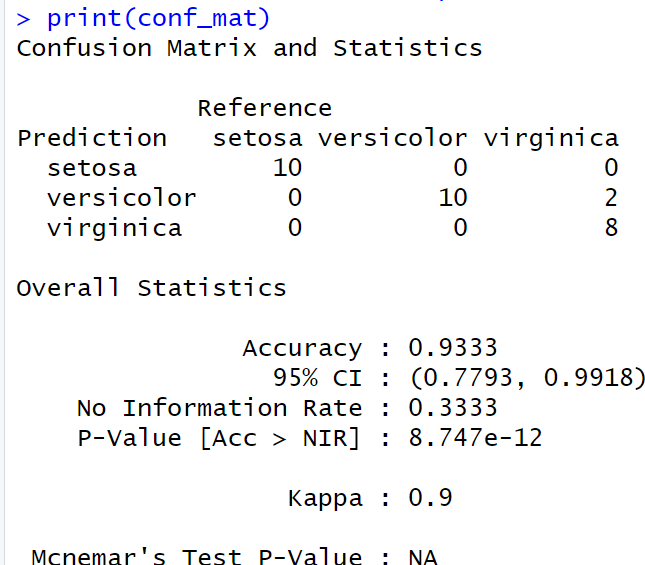
* + Createaconfusionmatrixtocomparepredictedvsactuallabels.
  + Printevaluationmetricslikeaccuracy,sensitivity,specificity,etc.

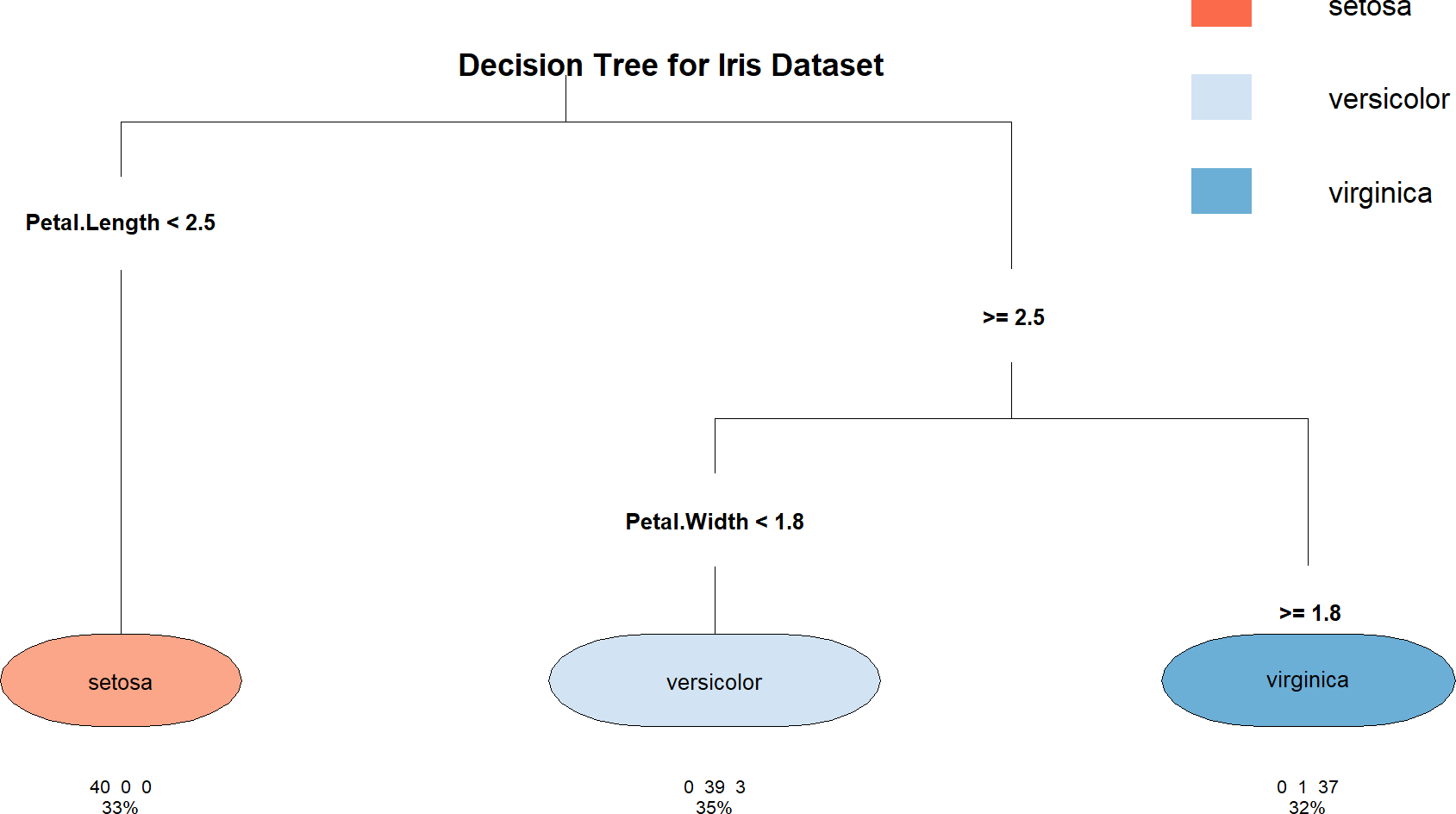
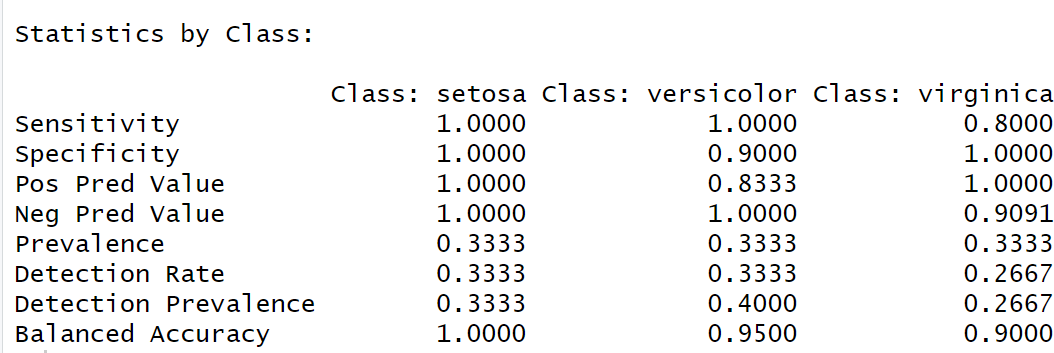
# Code:

conf\_mat <- confusionMatrix(pred, test\_data$Species) print(conf\_mat)

# Output:

****



****

**Result:**

TheDecisionTreeisImplementedSuccessfully.

**IMPLEMENT K-NEAREST NEIGHBOR ALGORITHM IN R**

**EXPNO:5**

# Aim:

ImplementaKNNClassificationontheGivenDataset.

# Procedure:

1. **LoadRequiredLibraries**
   * Loadthenecessarylibraries:
     + classforKNNmodel.
     + ggplot2forplotting.
     + GGallyforadvancedplots(pairwiseplots).
     + caretfordatapartitioningandevaluation.

# Code:

library(class) library(ggplot2) library(GGally) library(caret)

# Loadthe Dataset

* + Load theIrisdataset.
  + Displaythefirstfewrowstounderstandthestructure.

# Code:

data("iris")

print("First FewRows of Dataset") head(iris)

# DefineaNormalizeFunction

* + Create a custom function to normalize (scale between 0 and 1) the numerical feature columns.

# Code:

normalize<-function(x){

return((x-min(x))/(max(x)-min(x)))

}

# NormalizetheFeatureColumns

* + Applythenormalizationfunctiontothefirstfourfeaturecolumns.
  + AddbacktheSpeciescolumnseparately.

**Code:**

iris\_norm <- as.data.frame(lapply(iris[1:4], normalize))iris\_norm$Species <- iris$Species

# SplittheDataintoTrainingand TestingSets

* + Setarandomseedforreproducibility.
  + UsecreateDataPartitiontosplit:
    - 80%fortraining
    - 20%fortesting

# Code:

set.seed(123)

train\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE) train\_data <- iris\_norm[train\_index, ]

test\_data<-iris\_norm[-train\_index,]

# ExtractTrainingandTestLabels

* + Separatethelabels(Species)fromthefeaturedataforbothtrainandtestsets.

# Code:

train\_labels <- train\_data$Species test\_labels <- test\_data$Species

# TraintheKNNModel

* + TraintheK-NearestNeighborsmodelusing:
    - Normalizedfeaturecolumns
    - k =5 neighbors.

# Code:

knn\_model<-knn(train=train\_data[,1:4],test=test\_data[,1:4],cl=train\_labels,k

=5)

print(knn\_model)

# VisualizetheData

* + Createvisualizationstounderstandfeaturedistributions:
    - ScatterplotofSepalLengthvsSepalWidth.
    - Pairwiseplots(allfeaturecombinations).

# Code:

ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) + geom\_point() +

labs(title ="Scatter Plot of Sepal Dimensions", x ="Sepal Length", y ="Sepal Width") +

theme\_minimal()

ggpairs(iris, aes(color = Species)) + theme\_minimal()

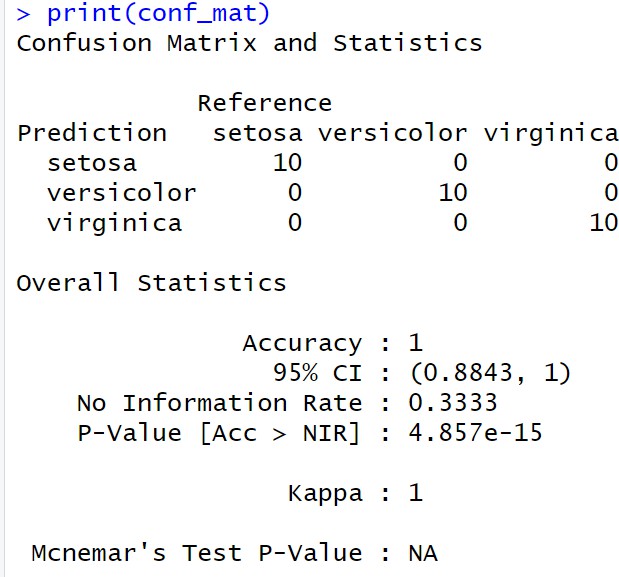
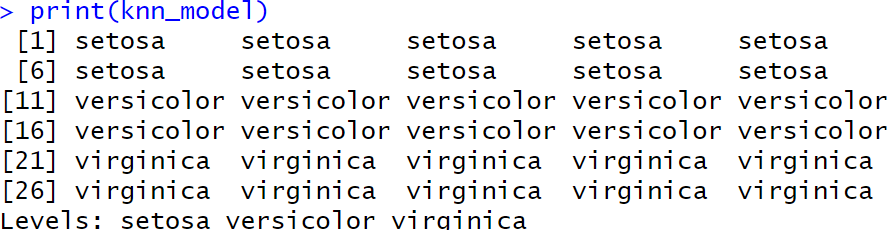
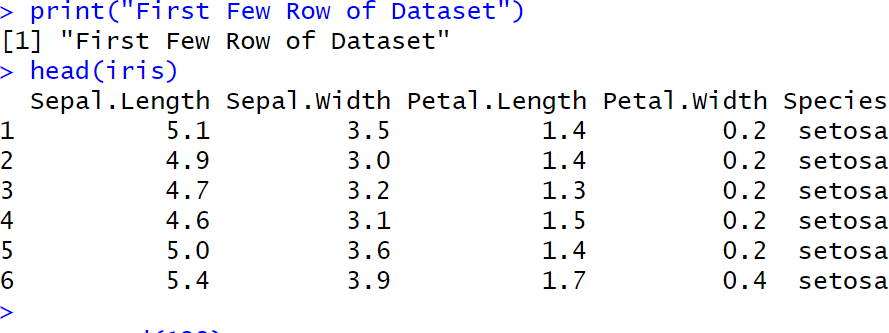
# EvaluateModelPerformance

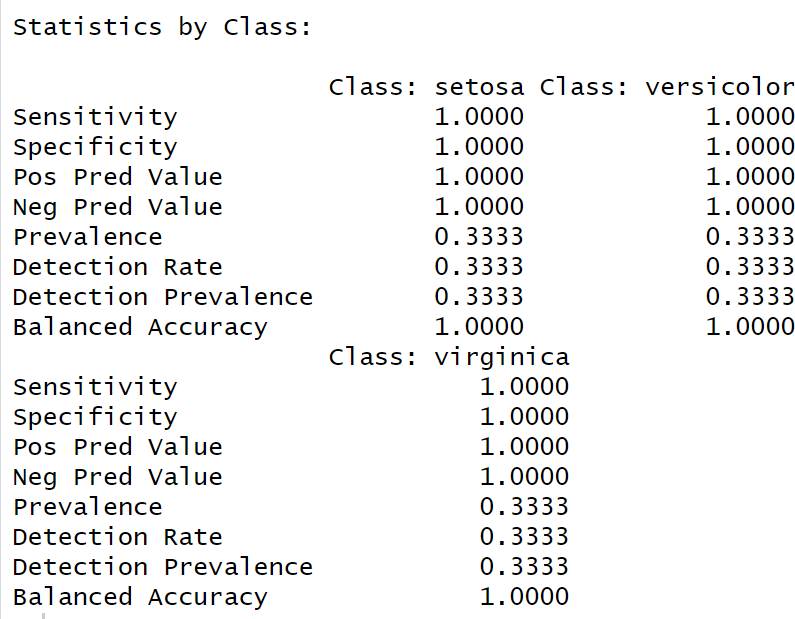
* + Generateaconfusionmatrixcomparingpredictionsandtruelabels.
  + Printclassificationresultsincludingaccuracy,sensitivity,andspecificity.

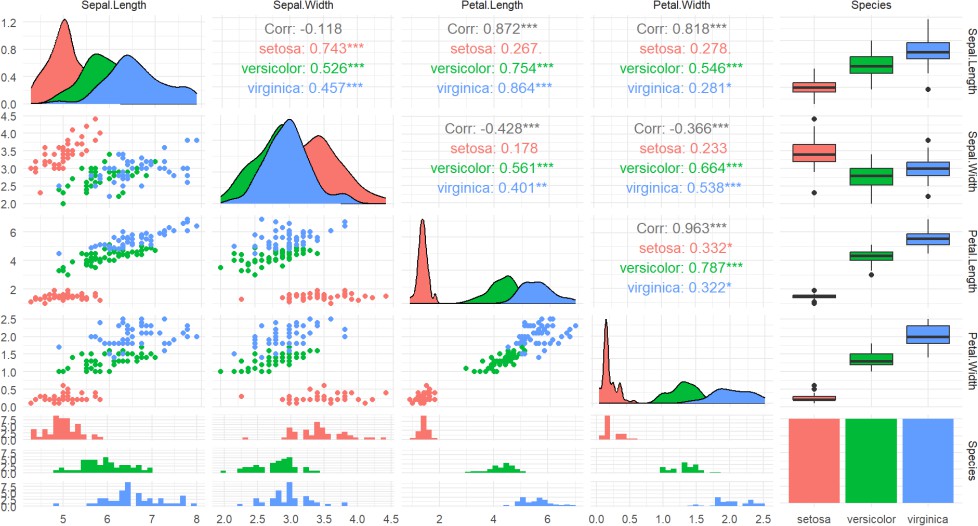
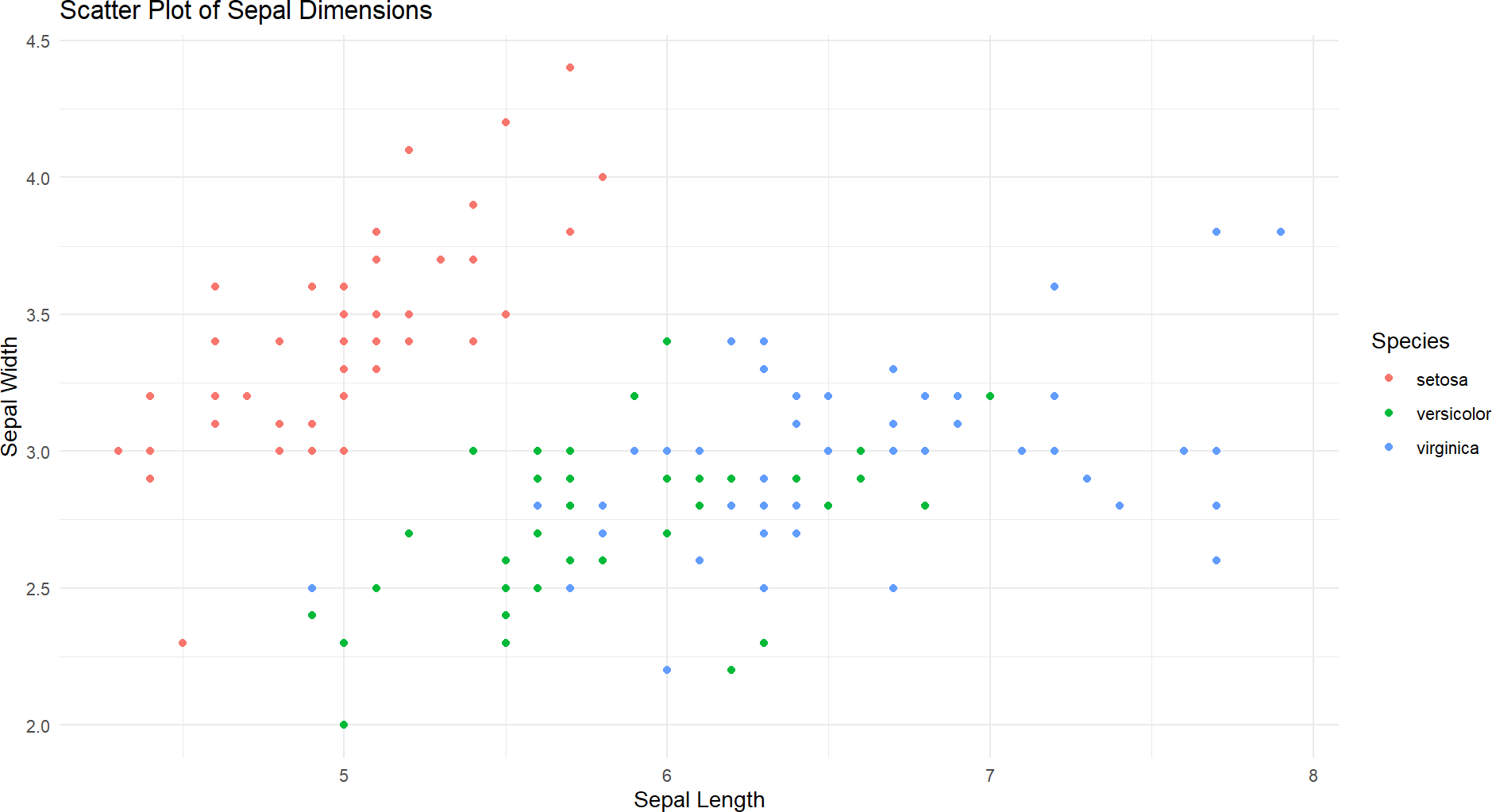
# Code:

conf\_mat <- confusionMatrix(knn\_model, test\_labels) print(conf\_mat)

# Output:

****



****

**Result:**

TheKNNClassificationisSuccessfullyImplemented.

**IMPLEMENTNAIVEBAYESIANCLASSIFIERINR**

**EXPNO:6**

# Aim:

ImplementaNaïveBayesClassificationontheGivenDataset.

# Procedure:

1. **LoadRequiredLibraries**
   * Loadthenecessarylibraries:
     + e1071fortheNaiveBayesmodel.
     + ggplot2forvisualization.
     + caretfordatapartitioningandevaluation.

# Code:

library(e1071) library(ggplot2) library(caret)

# Loadthe Dataset

* + Load theIrisdataset.
  + Displaythefirstfewrowsforaquickoverview.

# Code:

data("iris")

print("First FewRows of Dataset") head(iris)

# SplittheDataintoTrainingand TestingSets

* + Setarandomseedtoensurereproducibility.
  + Splitthedatainto:
    - 80%fortraining
    - 20%fortesting

# Code:

set.seed(123)

train\_index <- createDataPartition(iris$Species, p = 0.8, list = FALSE) train\_data <- iris[train\_index, ]

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

# ExtractTrainingandTestLabels

* + AssigntheSpeciescolumnasthelabelsfortrainingandtesting.

# Code:

train\_labels <- train\_data$Species test\_labels <- test\_data$Species

# Trainthe Naive BayesModel

* + TraintheNaiveBayesclassifierusingthetrainingdata.

# Code:

nb\_model <- naiveBayes(Species ~ ., data =train\_data) print(nb\_model)

# VisualizetheData

* + CreateascatterplotofSepalLengthvsSepalWidthcoloredbyspecies.

# Code:

ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) + geom\_point() +

labs(title ="Scatter Plot of Sepal Dimensions", x ="Sepal Length", y ="Sepal Width") +

theme\_minimal()

# MakePredictionsontheTestData

* + Predictthespeciesforthetestdatasetusingthetrainedmodel.

# Code:

pred<-predict(nb\_model,test\_data)

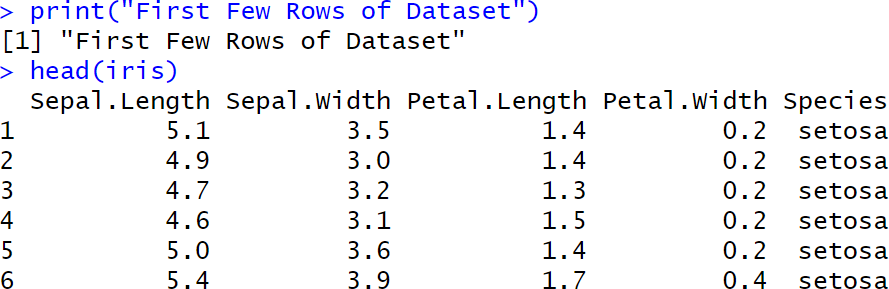
# EvaluateModelPerformance

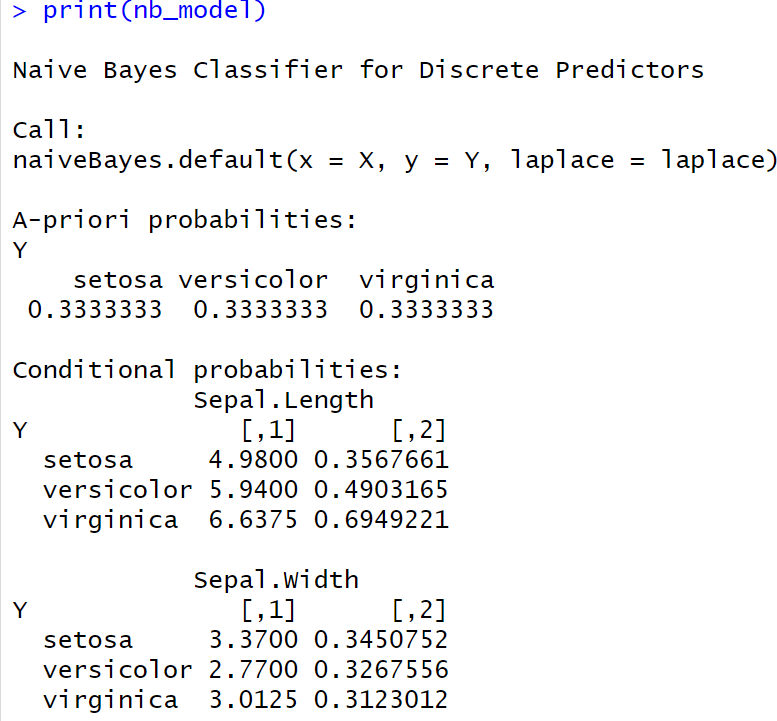
* + Generateaconfusionmatrixtocomparethepredictedlabelsandtruelabels.
  + Printevaluationmetricslikeaccuracy,sensitivity,andspecificity.

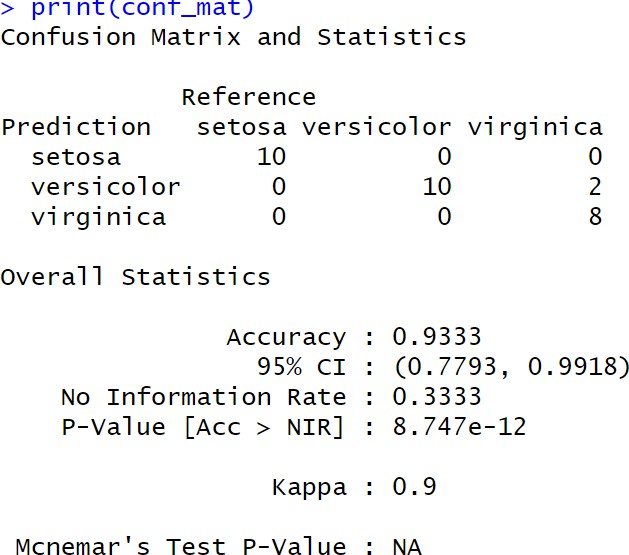
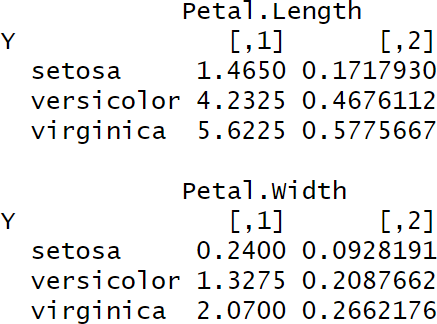
# Code:

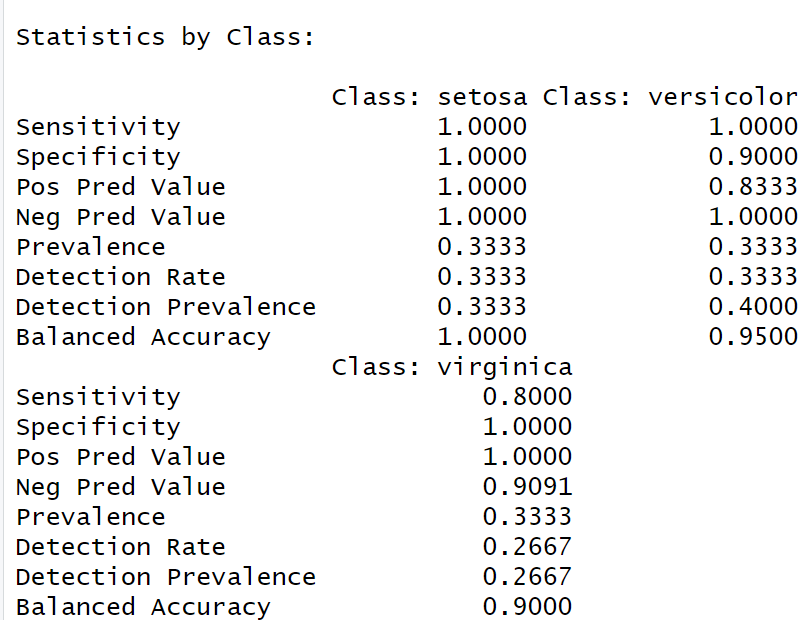
conf\_mat <- confusionMatrix(pred, test\_labels) print(conf\_mat)

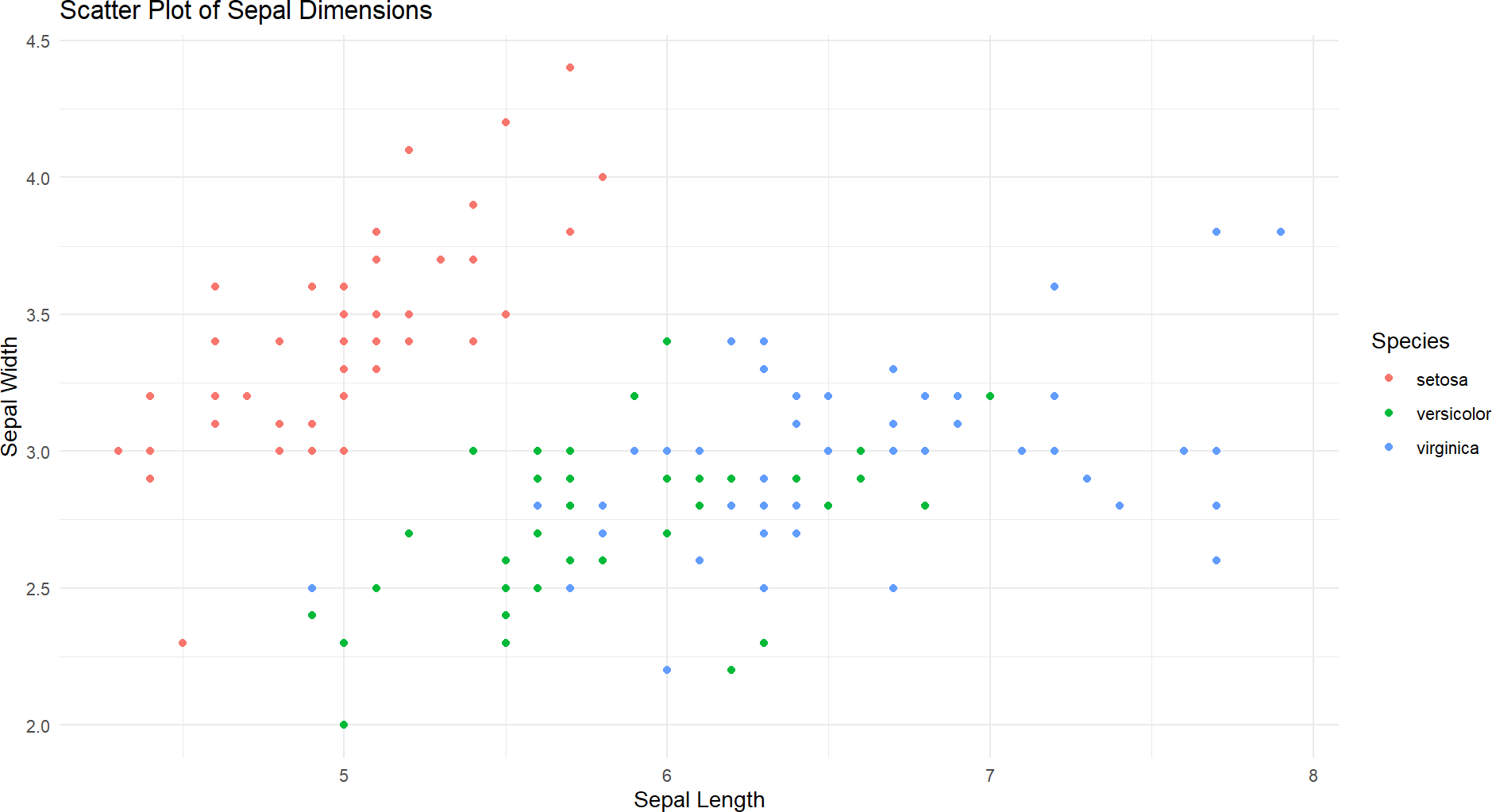
# Output:

****



****



****

**Result:**

TheNaïveBayesClassificationisSuccessfullyImplemented.

**IMPLEMENTLINEARREGRESSIONINR**

**EXPNO:7**

# Aim:

**EXPNO:1 Title**

ImplementaLinearRegressionontheGivenDataset.

# Procedure:

1. **LoadRequiredLibraries**
   * Loadthenecessarylibraries:
     + ggplot2forvisualization.
     + caretforsplittingthedataandevaluatingthemodel.

# Code:

library(ggplot2) library(caret)

# Loadthe Dataset

* + LoadtheHeadbraindatasetfromaCSVfile.
  + Displaythefirstfewrowstoinspectthedata.

# Code:

df <- read.csv("C:/Users/karthick.S/OneDrive/Documents/231801079- 4/SAC/headbrain.csv")

print("First FewRows of Dataset") head(df)

# SplittheDataintoTrainingand TestingSets

* + Setarandomseedforreproducibility.
  + Splitthedatainto:
    - 70%fortraining
    - 30%fortesting

# Code:

set.seed(123)

index <- createDataPartition(df$Brain.Weight.grams., p = 0.7, list = FALSE) train <- df[index, ]

test<-df[-index,]

# TraintheLinearRegression Model

* + Train a linear regression model to predict Brain.Weight.grams. based on Head.Size.cm.3..

# Code:

print("LinearRegressionModel")

model <- lm(Brain.Weight.grams. ~ Head.Size.cm.3., data = train) print(model)

# MakePredictionsontheTestData

* + Usethetrainedmodeltopredictbrainweightvaluesforthetestdataset.

# Code:

pred<-predict(model,newdata=test)

# EvaluateModelPerformance

* + UsepostResampletocalculateevaluationmetrics:
    - RMSE(RootMean Squared Error)
    - R-squared(CoefficientofDetermination)
    - MAE (MeanAbsoluteError)

# Code:

evaluation <- postResample(pred, test$Brain.Weight.grams.) cat("RMSE:", evaluation["RMSE"], "\n")

cat("R-squared:",evaluation["Rsquared"],"\n")

cat("MAE:",evaluation["MAE"],"\n")

# VisualizetheData

* + Plotthescatterpointsoftheoriginaldata.
  + Overlaytheregressionlinebasedonthemodel’spredictions.

# Code:

x\_vals <- seq(min(df$Head.Size.cm.3.) - 100, max(df$Head.Size.cm.3.) + 100, length.out = 1000)

pred\_line <- data.frame(Head.Size.cm.3. = x\_vals) pred\_line$Brain.Weight.grams. <- predict(model, newdata = pred\_line)

plot(df$Head.Size.cm.3., df$Brain.Weight.grams., col = "green", pch = 19,

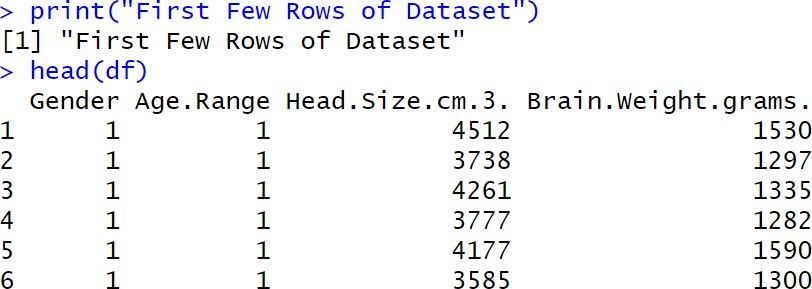
xlab="HeadSize(cm³)",

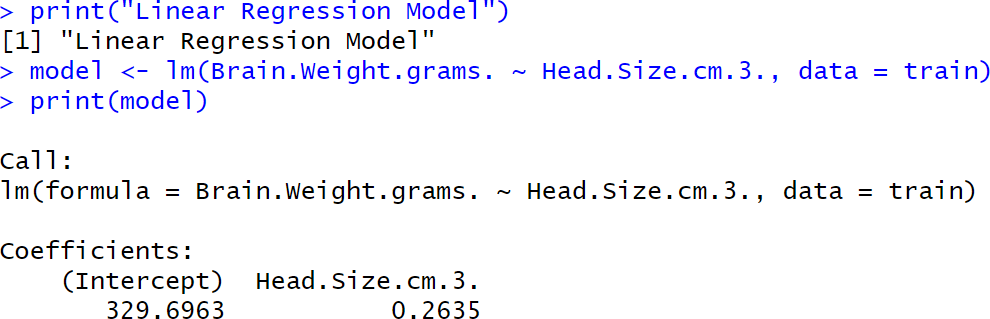
ylab= "BrainWeight(grams)",

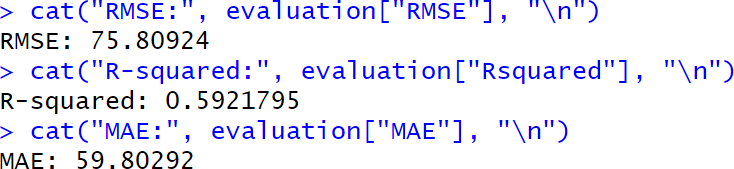
main= "HeadSizevsBrainWeightwithRegressionLine")

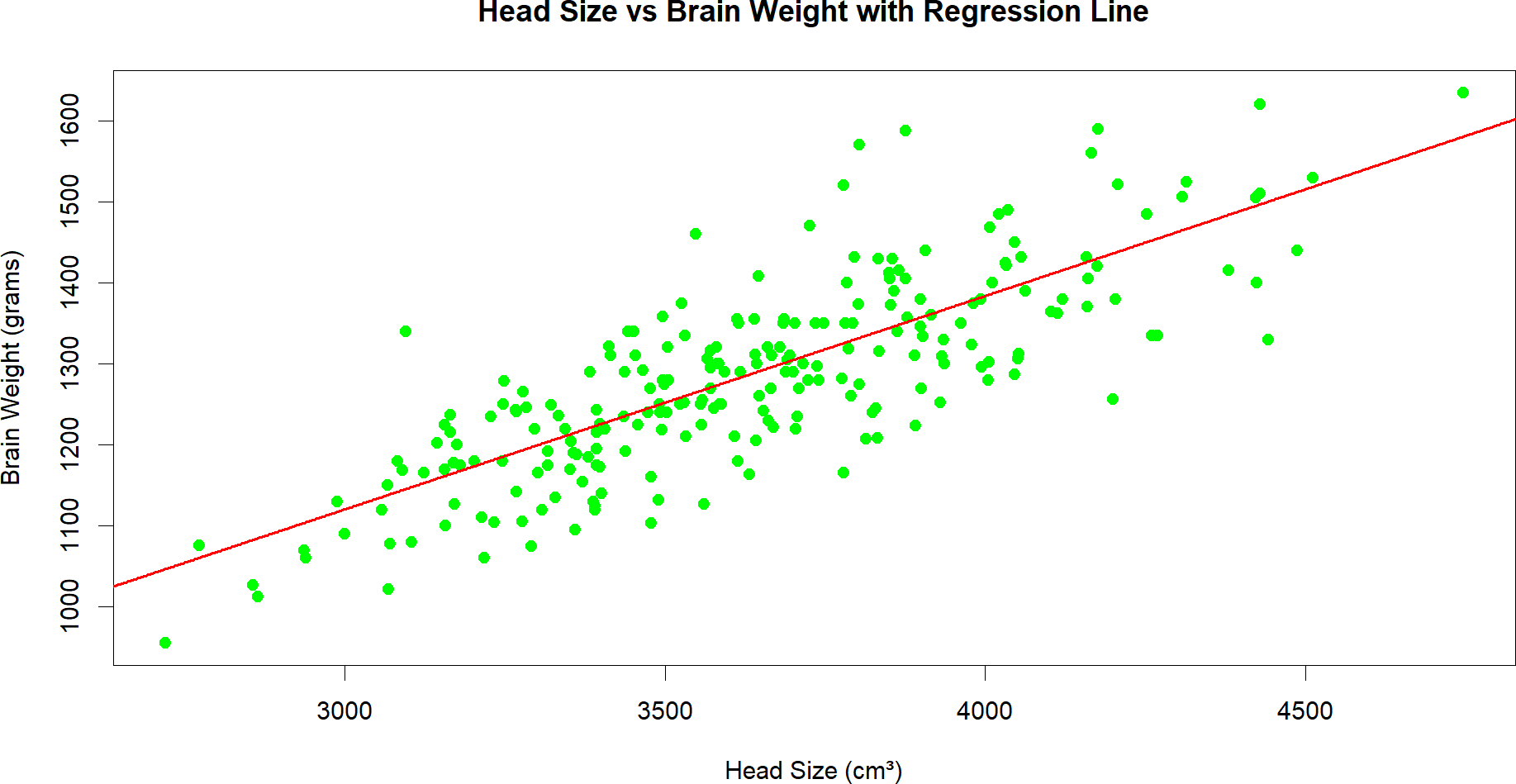
lines(pred\_line$Head.Size.cm.3., pred\_line$Brain.Weight.grams., col = "red", lwd = 2)

# Output:

****



****

****

**Result:**

TheLinearRegressionisSuccessfullyImplemented.

**IMPLEMENT K-MEANS CLUSTERING ALGORITHM IN R**

**EXPNO:8**

# Aim:

ImplementaKmeansClusteringontheGivenDataset.

**Procedure:**

**ProcedureforPerformingandEvaluatingK-meansClusteringinR**

1. **LoadRequiredLibraries**
   * Loadthenecessarylibraries:
     + ggplot2forplotting.
     + clusterforsilhouetteanalysis.
     + factoextraforeasyvisualizationofclustering.

# Code:

library(ggplot2) library(cluster) library(factoextra)

# Loadthe Dataset

* + Load theIrisdataset.
  + RemovetheSpeciescolumntofocusonlyonthenumericfeaturesforclustering.

# Code:

data(iris)

iris\_data <- iris[, -5] head(iris\_data)

# Determine theOptimalNumberofClustersUsingElbowMethod

* + Use the Within-Cluster Sum of Squares (WSS) method to decide howmany clusters are appropriate.

# Code:

fviz\_nbclust(iris\_data, kmeans, method = "wss") + ggtitle("Elbow Method for Optimal K")

# ApplyK-meansClusteringwith3Clusters

* + Setarandomseedforreproducibility.
  + ApplyK-meansclusteringspecifying3clusters(sinceIrishas3species).

# Code:

set.seed(123)

kmeans\_model<-kmeans(iris\_data,centers=3,nstart=25)

# PrintClusterCentersandClusterAssignments

* + Viewthecenterpointsoftheclustersandhowthedatapointswereassigned.

# Code:

print(kmeans\_model$centers) print(kmeans\_model$cluster)

# VisualizetheClusters

* + Visualizetheclusteringresultusingascatterplotwithconvexhullsaroundclusters.

# Code:

fviz\_cluster(kmeans\_model, data = iris\_data, geom = "point", ellipse.type = "convex") +

ggtitle("K-meansClusteringonIrisDataset")

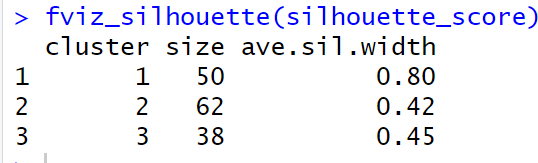
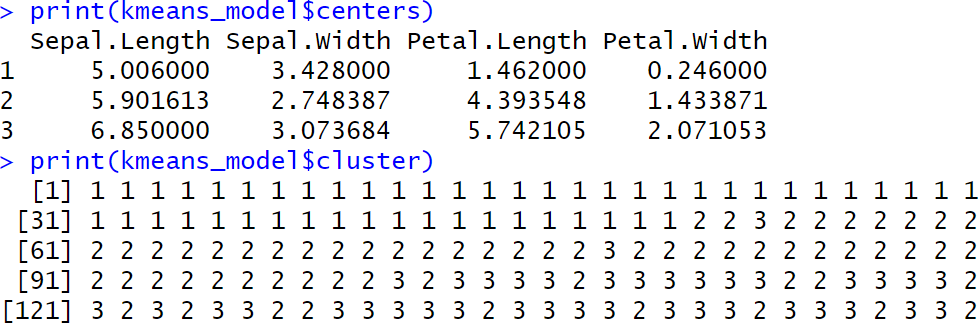
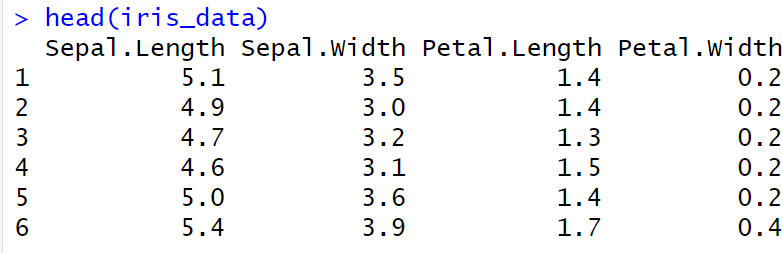
# EvaluatetheClustering(SilhouetteAnalysis)

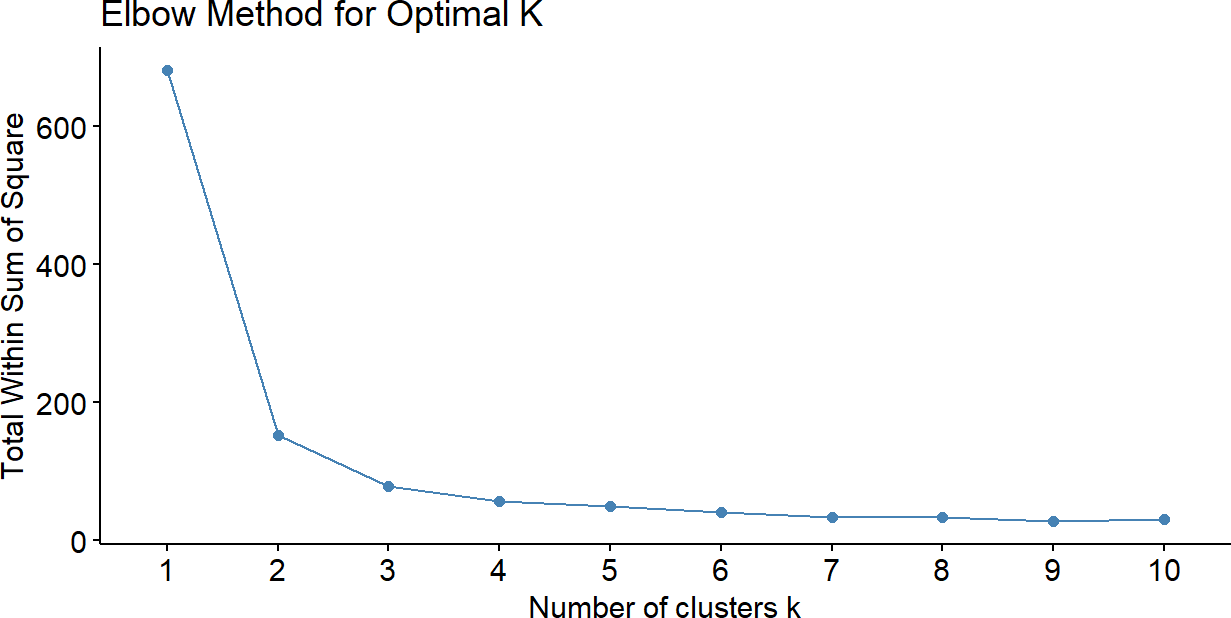
* + Performsilhouetteanalysistoassessthequalityoftheclustering.

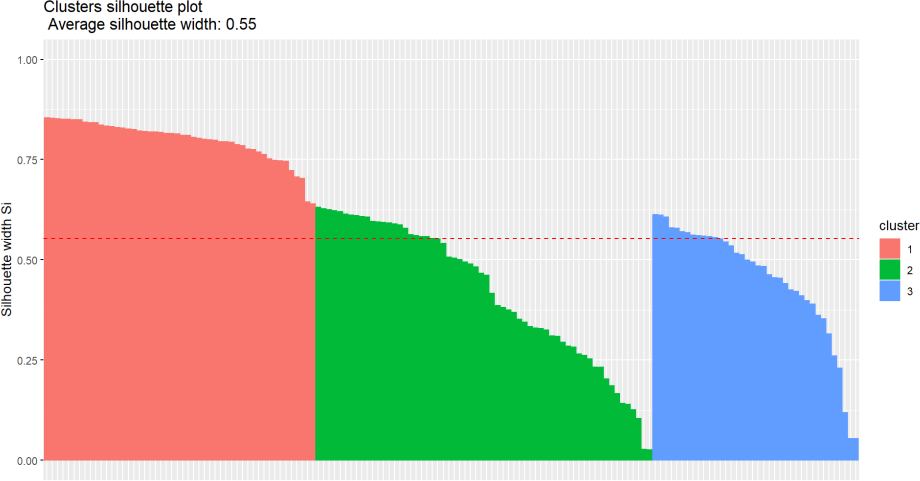
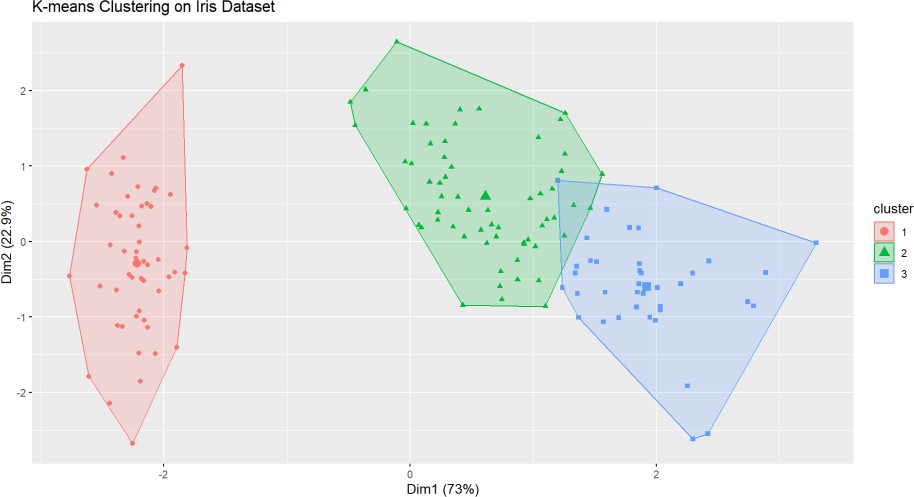
# Code:

silhouette\_score <- silhouette(kmeans\_model$cluster, dist(iris\_data)) fviz\_silhouette(silhouette\_score)

# Output:

****



****

**Result:**

TheKmeansisSuccessfullyImplemented.