Institute of Technology & Management

GIDA Gorakhpur

DATA ANALYTICS LAB

Subject Code: KIT-651

Prepared by:

PANKAJ KUMAR GOND

1812013022

IT 3rd YEAR

Submitted to:

Mr. RAHUL CHAKRAVORTY

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

Data Analytics Lab (KIT-651)

INDEX

Sr. No.	OBJECTS	DATE	GRADE
1	**Program to print the Fibonacci Series.**		
2	**Write a program to find Mean, Median and Mode.**		
3	**Write a program in R for Linear Regression of Salary.**		
4	**Write a program in R for logistic regression.**		
5	**Program in R to perform matrix addition, subtraction, multiplication, division**		
6	**Program in R for dimensionality reduction using PCA.**		
7	**To perform K-Means clustering operation and visualization of iris dataset.**		
8	**Program to perform Apriori Algorithm in R.**		
9	**To perform KNN classifier in R.**		
10	**Program to perform Time series analysis in R.**		

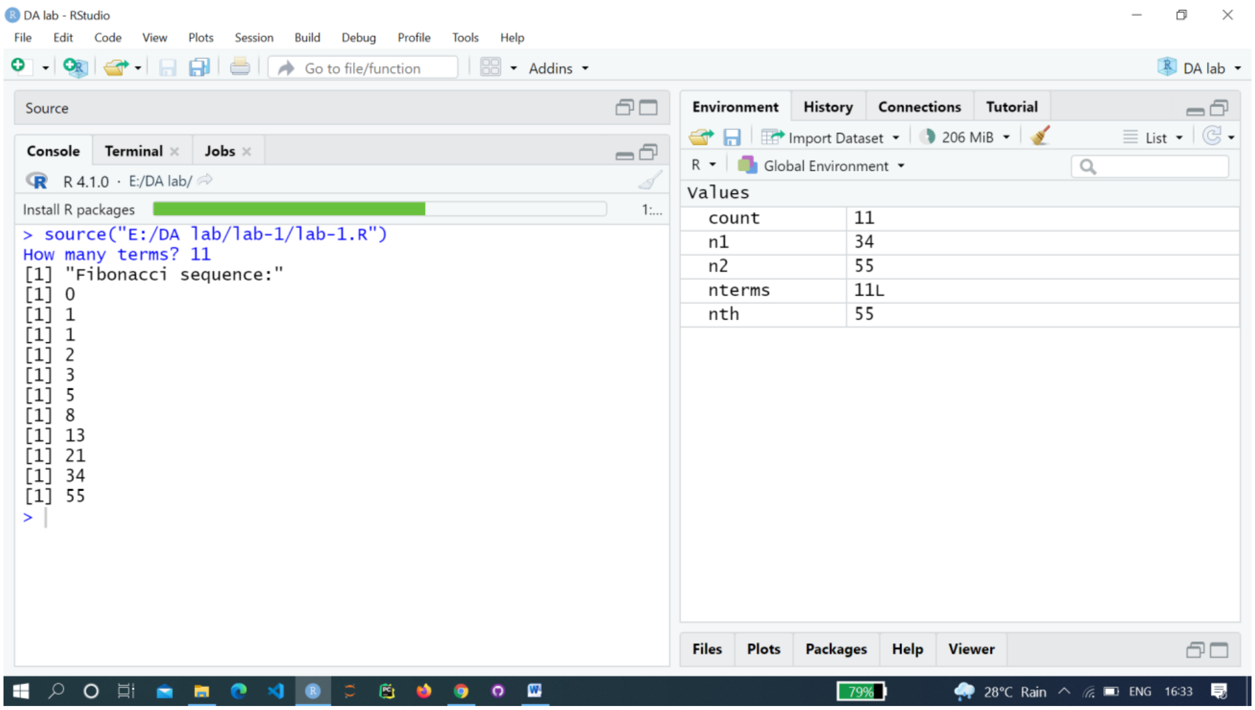
Program 1

Object: Program to print the Fibonacci Series

```
# take input from the user

nterms = as.integer(readline(prompt="How many terms? "))

# first two terms

n1 = 0

n2 = 1

count = 2

# check if the number of terms is valid

if(nterms<= 0) {

print("Plese enter a positive integer")

} else {

if(nterms == 1) {

print("Fibonacci sequence:")

print(n1)

} else {

print("Fibonacci sequence:")

print(n1)

print(n2)

while(count <nterms) {

nth = n1 + n2

print(nth)

# update values

n1 = n2

n2 = nth

count = count + 1

}

}
```

}

Output:


DA lab - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
DA lab
Source
Console Terminal Jobs
R 4.1.0 · E:/DA lab/
Install R packages
> source("E:/DA lab/lab-1/lab-1.R")
How many terms? 11
[1] "Fibonacci sequence:"
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
[1] 55
>
Environment History Connections Tutorial
Import Dataset
206 MiB
List
Global Environment
Values
count 11
n1 34
n2 55
nterms 11L
nth 55
Files Plots Packages Help Viewer
79%
28°C Rain
ENG
16:33

Program 2

Object: Write a program to find Mean, Median and Mode.

Mean:

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
# Find mean.
result.mean<-  mean(x)
print(result.mean)
# Find mean dropping NA values.
result.mean<-  mean(x,na.rm = TRUE)
print(result.mean)
```

Median:

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Find the median.
median.result<- median(x)
print(median.result)
```

Mode:

```
# Create the function.
getmode<- function(v) {
uniqv<- unique(v)
uniqv[which.max(tabulate(match(v, uniqv)))]
}
# Create the vector with numbers.
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
# Calculate the mode using the user function.
result<- getmode(v)
print(result)
```

Create the vector with characters.

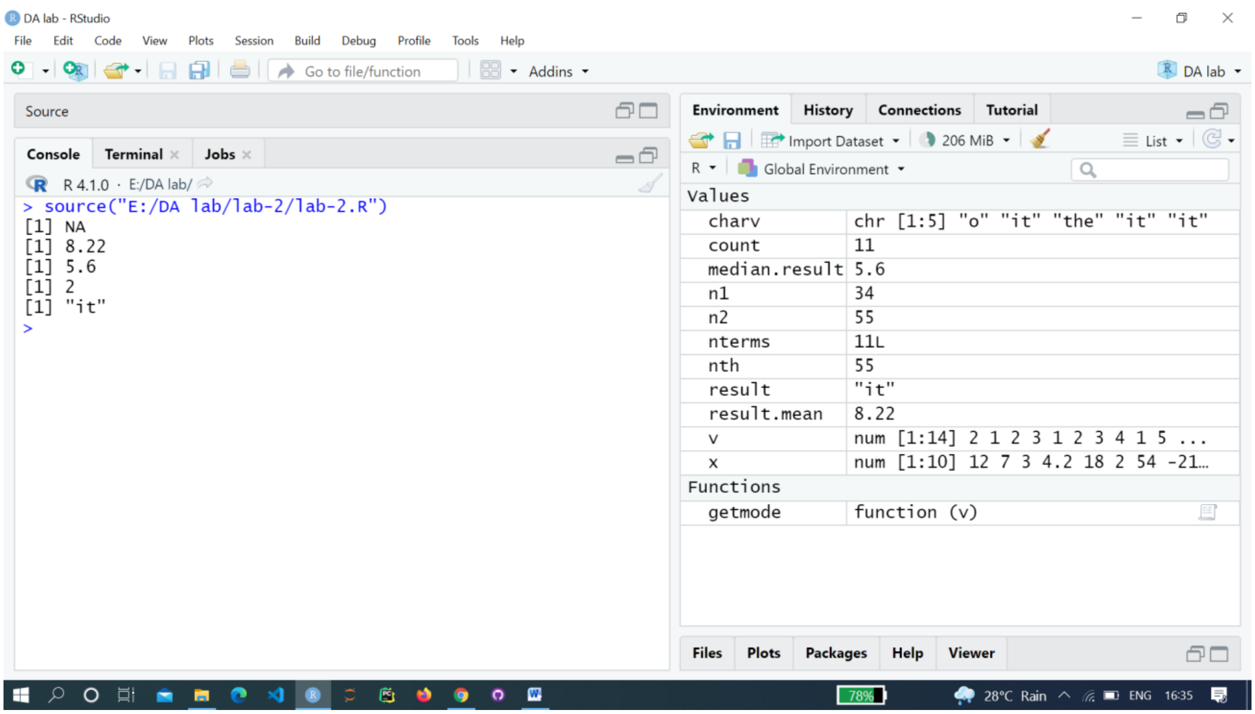
charv<- c("o","it","the","it","it")

Calculate the mode using the user function.

result<- getmode(charv)

print(result)

Output:



Program 3

Object: Write a program in R for Linear Regression of Salary

Source: Salary.csv

Years experienced	Salary
1.1	39343.00
1.3	46205.00
1.5	37731.00
2.0	43525.00
2.2	39891.00
2.9	56642.00
3.0	60150.00
3.2	54445.00
3.2	64445.00
3.7	57189.00

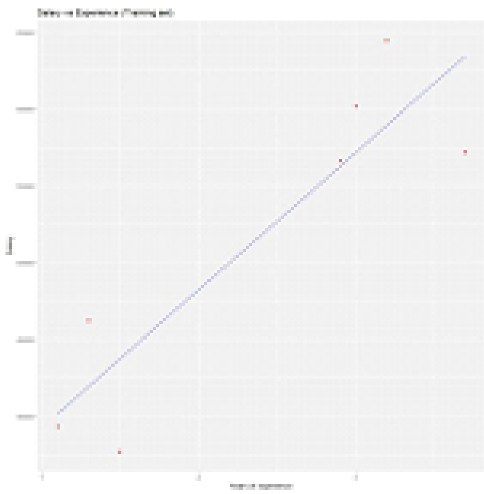
```
# Simple Linear Regression
# Importing the dataset
dataset = read.csv('salary.csv')
# Splitting the dataset into the
# Training set and Test set
install.packages('caTools')
library(caTools)
split = sample.split(dataset$Salary, SplitRatio = 0.7)
trainingset = subset(dataset, split == TRUE)
testset = subset(dataset, split == FALSE)
# Fitting Simple Linear Regression to the Training set
lm.r= lm(formula = Salary ~ YearsExperience, data = trainingset)
```

```
coef(lm.r)

# Predicting the Test set results

ypred = predict(lm.r, newdata = testset)

install.packages("ggplot2")

library(ggplot2)

# Visualising the Training set results

ggplot() + geom_point(aes(x = trainingset$YearsExperience,

                                        y = trainingset$Salary), colour = 'red') +

geom_line(aes(x = trainingset$YearsExperience,

y = predict(lm.r, newdata = trainingset)), colour = 'blue') +

ggtitle('Salary vs Experience (Training set)') +

xlab('Years of experience') +

ylab('Salary')

# Visualising the Test set results

ggplot() +

geom_point(aes(x = testset$YearsExperience, y = testset$Salary),

                              colour = 'red') +

geom_line(aes(x = trainingset$YearsExperience,

                              y = predict(lm.r, newdata = trainingset)),

                              colour = 'blue') +

ggtitle('Salary vs Experience (Test set)') +

xlab('Years of experience') +

ylab('Salary')
```

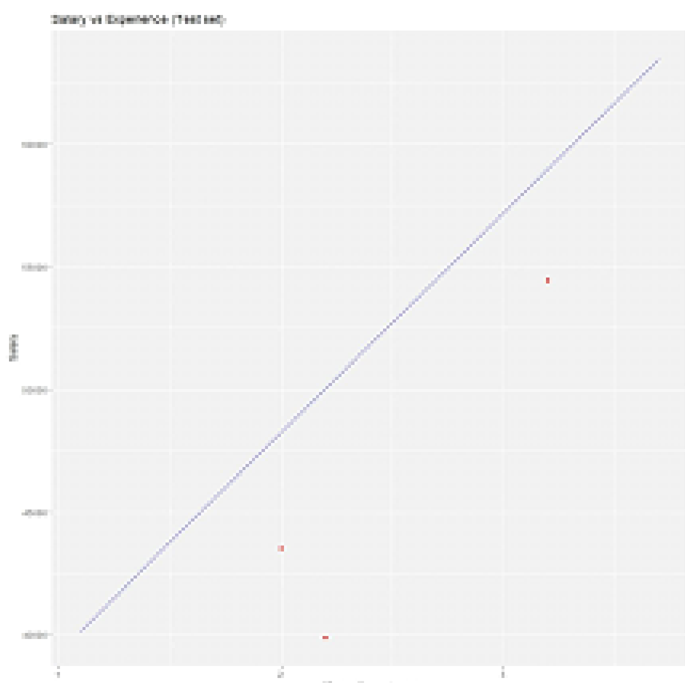
Output:

Intercept YearsExperience
24558.39 10639.23

Visualising the Training set results:



Visualising the Testing set results:



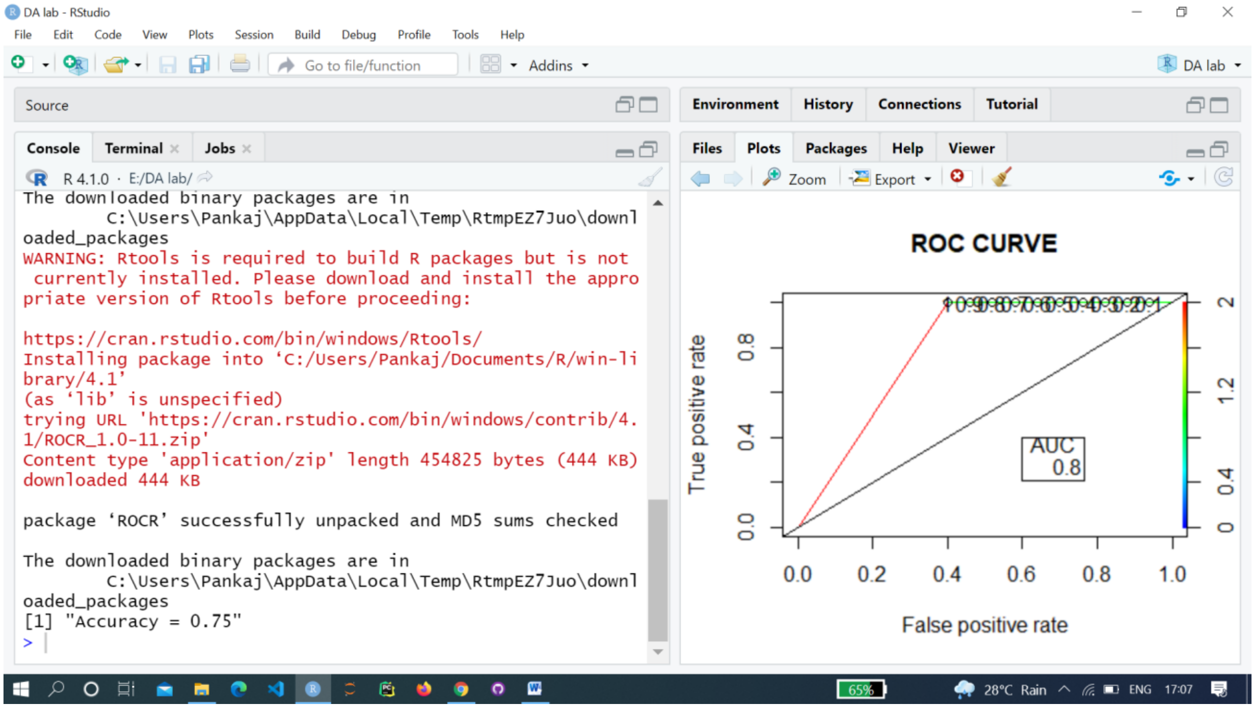
Program 4

Object: Write a program in R for logistic regression

```
# Installing the package
install.packages("dplyr")
# Loading package
library(dplyr)
# Summary of dataset in package
summary(mtcars)
# Installing the package
install.packages("caTools") # For Logistic regression
install.packages("ROCR")          # For ROC curve to evaluate model
# Loading package
library(caTools)
library(ROCR)
# Splitting dataset
split<- sample.split(mtcars, SplitRatio = 0.8)
split
train_reg<- subset(mtcars, split == "TRUE")
test_reg<- subset(mtcars, split == "FALSE")

# Training model
logistic_model<- glm(vs ~ wt + disp,data = train_reg,family = "binomial")
logistic_model

# Summary
summary(logistic_model)
# Predict test data based on model
```

```
predict_reg<- predict(logistic_model,test_reg, type = "response")
predict_reg
# Changing probabilities
predict_reg<- ifelse(predict_reg>0.5, 1, 0)
# Evaluating model accuracy
# using confusion matrix
table(test_reg$vs, predict_reg)
missing_classerr<- mean(predict_reg != test_reg$vs)
print(paste('Accuracy =', 1 - missing_classerr))
# ROC-AUC Curve
ROCPred<- prediction(predict_reg, test_reg$vs)
ROCPer<- performance(ROCPred, measure = "tpr",x.measure = "fpr")
auc<- performance(ROCPred, measure = "auc")
auc<- auc@y.values[[1]]
auc

# Plotting curve
plot(ROCPer)
plot(ROCPer, colorize = TRUE,print.cutoffs.at = seq(0.1, by = 0.1),main = "ROC CURVE")
abline(a = 0, b = 1)
auc<- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```

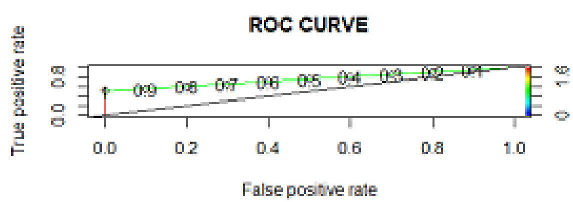
Output:



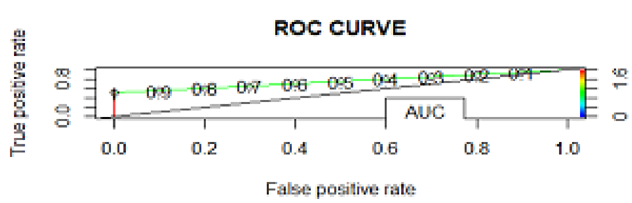
Evaluating model accuracy using confusion matrix:

```
   predict_reg
    0 1
  0 3 0
  1 3 3
```

ROC curve:



- **ROC-AUC Curve:**



AUC is 0.7333, so the more AUC is, the better the model performs.

Program 5

Object: Program in R to perform matrix addition, subtraction, multiplication, and division.

```
# Create two 2x3 matrixes.
m1 = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
print("Matrix-1:")
print(m1)
m2 = matrix(c(0, 1, 2, 3, 0, 2), nrow = 2)
print("Matrix-2:")
print(m2)

result = m1 + m2
print("Result of addition")
print(result)

result = m1 - m2
print("Result of subtraction")
print(result)

result = m1 * m2
print("Result of multiplication")
print(result)

result = m1 / m2
print("Result of division:")
print(result)
```

Output:

```
Console   Terminal ×   Jobs ×
R 4.1.0 · E:/DA lab/
> source("E:/DA lab/lab-5/lab-5.R")
[1] "Matrix-1:"
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[1] "Matrix-2:"
     [,1] [,2] [,3]
[1,]    0    2    0
[2,]    1    3    2
[1] "Result of addition"
     [,1] [,2] [,3]
[1,]    1    5    5
[2,]    3    7    8
[1] "Result of subtraction"
     [,1] [,2] [,3]
[1,]    1    1    5
[2,]    1    1    4
[1] "Result of multiplication"
     [,1] [,2] [,3]
[1,]    0    6    0
[2,]    2   12   12

[1] "Result of division:"
     [,1]     [,2] [,3]
[1,]  Inf 1.500000  Inf
[2,]    2 1.333333    3
>
```

Program 6

Object: Program in R for dimensionality reduction using PCA

dataset = read.csv('Beer_dataset.csv')

	ABV	Brewing.Company	Beer.Name	Ratings	Cellar_Temperatire_Min	Cellar_Temperatire_Max	Serving_Temperatire_Min	Serving_Temperatire_Max	Score
1	7.30	7646	151675	8	40	45	45	50	3.40
2	5.40	8307	129948	546	40	45	45	50	3.46
3	5.00	4588	67651	0	35	40	40	45	0.00
4	7.50	8032	70755	24	40	45	45	50	3.93
5	2.90	5370	161045	27	40	45	45	50	4.27
6	7.30	1728	4306	118	40	45	45	50	3.80
7	6.50	7969	88431	4	35	40	40	45	3.68
8	5.40	9967	2043	2	35	40	40	45	3.67
9	9.20	2709	75690	0	45	50	50	55	0.00
10	5.00	2023	6749	4	40	45	45	50	3.41
11	8.50	8672	9607	16	45	50	50	55	4.31
12	4.50	9478	70425	2	40	45	45	50	3.46
13	4.90	8567	6927	0	35	40	40	45	0.00
14	5.20	5258	113288	9	35	40	40	45	3.55
15	4.00	8450	78037	0	40	45	45	50	0.00

```
library(caTools)

set.seed(123)

split = sample.split(dataset$Score, SplitRatio = 0.80)

training_set = subset(dataset, split == TRUE)

test_set = subset(dataset, split == FALSE)

# Multiple Linear Regressor

mlr = lm(formula = Score ~ . , data = training_set )

print(mlr)

predy = predict(mlr, newdata = test_set)

actuals_and_preds<- data.frame(cbind(actuals=test_set$Score, predicteds = predy))

min_max_accuracy<- mean(apply(actuals_and_preds, 1, min) / apply(actuals_and_preds, 1, max))print(min_max_accuracy)
```

Output:

0.7055808

```
install.packages("caret") # Execute Once
library(caret)
install.packages("e1071") # Execute Once
library(e1071)
pca = preProcess(training_set[-9], method = 'pca', pcaComp = 2)
training_set.pca = predict(pca, training_set)
test_set.pca = predict(pca, test_set)
training_set.pca = training_set.pca[c(2,3,1)] test_set.pca = test_set.pca[c(2,3,1)]
```

training_set.pca:

PC1	PC2	Score
-0.433930559	-1.0792608635	3.40
-0.194680704	-1.0223053679	3.46
2.816675591	0.6762499363	0.00
-0.468215109	0.0395622722	3.93
0.249060407	-0.9414476455	4.27
-0.442408527	2.1129572950	3.80
2.580929450	-0.1870407489	3.68
2.748755535	0.6903255651	3.67
-0.074098229	1.9770570919	3.41
-3.515699158	0.8043516533	4.31
-0.001653801	-0.2959694644	3.46

test_set.pca:

PC1	PC2	Score
-3.616556523	0.921330860	0.00
0.077143304	-0.238512854	0.00
-0.295401230	1.110283477	3.60
-0.156313985	-0.689357507	2.83
-0.616626456	0.369873191	4.00
-0.112304524	-1.769146986	3.69
-3.591286379	-0.365072885	3.86
-0.312866781	-1.630388360	3.92
2.967341843	0.451319298	0.00
-0.415121371	-0.648549846	3.96
-0.126410206	0.518091211	0.00
-0.309686106	-0.388796461	3.00

```
mlr_pca = lm(formula = Score ~ . , data = training_set.pca)print(mlr_pca)

predy_pca = predict(mlr_pca, newdata = test_set.pca)

actuals_and_preds_pca<- data.frame(cbind(actuals= test_set.pca$Score, predicteds = predy_pca))

min_max_accuracy_pca<- mean(apply(actuals_and_preds_pca, 1, min) / apply(actuals_and_preds_pca, 1, max))print(min_max_accuracy_pca)
```

Output:

0.7301324

Program 7

Object: To perform K-Means clustering operation and visualization of iris dataset

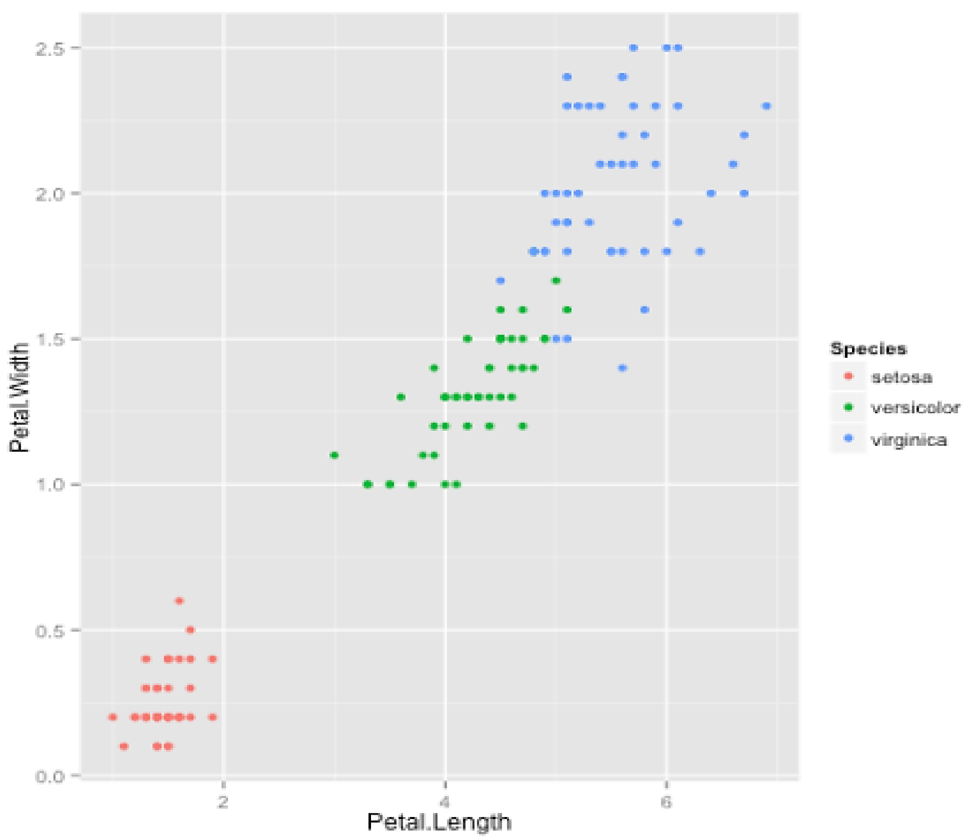
library(datasets)

head(iris)

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()



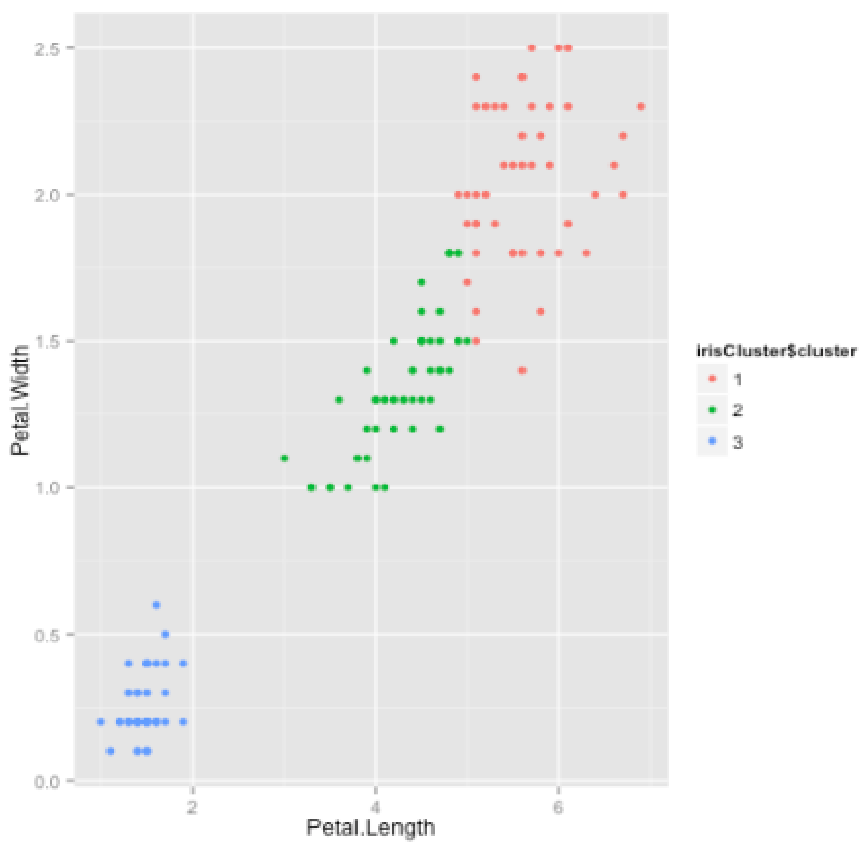
set.seed(20)

```
irisCluster<- kmeans(iris[, 3:4], 3, nstart = 20)

irisCluster

K-means clustering with 3 clusters of sizes 46, 54, 50

Cluster means:

Petal.LengthPetal.Width
1     5.626087    2.047826
2     4.292593    1.359259
3     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [69] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
[103] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1
[137] 1 1 2 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 15.16348 14.22741  2.02200
 (between_SS / total_SS=  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

table(irisCluster$cluster, iris$Species)

setosaversicolorvirginica
  1     0         2        44
```

2	0	48	6
3	50	0	0

```
irisCluster$cluster<- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```

Graph:



Program 8

Object: Program to perform Apriori Algorithm In R

Load required library

library(arules)

library(arulesViz)

library(RColorBrewer)

Import the dataset

data("Groceries")

Applying apriori() function

rules<- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))

Applying inspect() function

inspect(rules[1:10])

Applying itemFrequencyPlot() function

arules::itemFrequencyPlot(Groceries, topN = 20,

col = brewer.pal(8, 'Pastel2'),

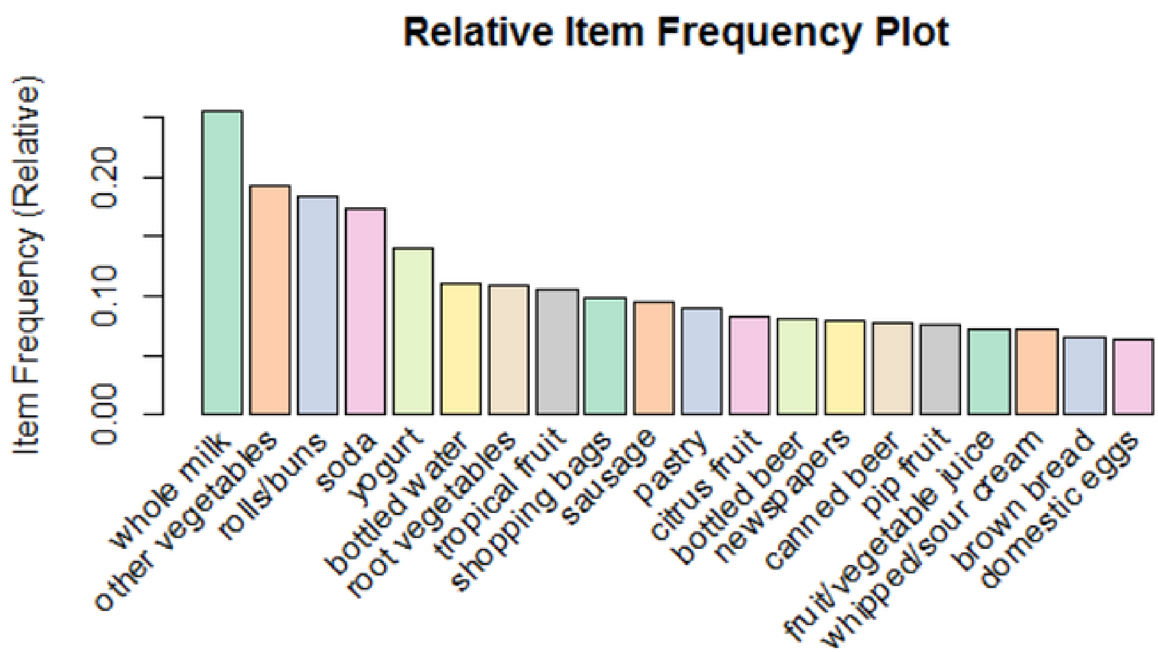
main = 'Relative Item Frequency Plot',

type = "relative",

ylab = "Item Frequency (Relative)")

```
     lhs                  rhs                support    confidence lift     count
[1]  {}                => {whole milk}       0.25551601 0.2555160  1.000000 2513
[2]  {hard cheese}     => {whole milk}       0.01006609 0.4107884  1.607682   99
[3]  {butter milk}     => {other vegetables} 0.01037112 0.3709091  1.916916  102
[4]  {butter milk}     => {whole milk}       0.01159126 0.4145455  1.622385  114
[5]  {ham}             => {whole milk}       0.01148958 0.4414062  1.727509  113
[6]  {sliced cheese}   => {whole milk}       0.01077783 0.4398340  1.721356  106
[7]  {oil}             => {whole milk}       0.01128622 0.4021739  1.573968  111
[8]  {onions}          => {other vegetables} 0.01423488 0.4590164  2.372268  140
[9]  {onions}          => {whole milk}       0.01209964 0.3901639  1.526965  119
[10] {berries}         => {yogurt}           0.01057448 0.3180428  2.279848  104
```

Graph:



Relative Item Frequency Plot
Item Frequency (Relative)
0.00
0.10
0.20
whole milk
other vegetables
rolls/buns
soda
yogurt
bottled water
root vegetables
tropical fruit
shopping bags
sausage
pastry
citrus fruit
bottled beer
newspapers
canned beer
pip fruit
fruit/vegetable juice
whipped/sour cream
brown bread
domestic eggs

Program 9

Object: To perform KNN classifier in R

```
# Loading data
data(iris)
# Structure
str(iris)
# Installing Packages
install.packages("e1071")
install.packages("caTools")
install.packages("class")

# Loading package
library(e1071)
library(caTools)
library(class)

# Loading data
data(iris)
head(iris)

# Splitting data into train
# and test data
split<- sample.split(iris, SplitRatio = 0.7)
train_cl<- subset(iris, split == "TRUE")
test_cl<- subset(iris, split == "FALSE")

# Feature Scaling
```

```
train_scale<- scale(train_cl[, 1:4])
test_scale<- scale(test_cl[, 1:4])

# Fitting KNN Model
# to training dataset
classifier_knn<- knn(train = train_scale,
                     test = test_scale,
                     cl = train_cl$Species,
                     k = 1)
classifier_knn

# Confusiin Matrix
cm <- table(test_cl$Species, classifier_knn)
cm

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 3
classifier_knn<- knn(train = train_scale,
                     test = test_scale,
                     cl = train_cl$Species,
                     k = 3)
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 5
```

```
classifier_knn<- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 5)
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 7
classifier_knn<- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 7)
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))
# K = 15
classifier_knn<- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 15)
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError))

# K = 19
classifier_knn<- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$Species,
                      k = 19)
misClassError<- mean(classifier_knn != test_cl$Species)
print(paste('Accuracy =', 1-misClassError)
```

OUTPUT:

- **Model classifier_knn(k=1):**

```
> classifier_knn
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
[15] setosa     setosa     setosa     setosa     setosa     setosa     versicolor
[22] versicolor versicolor virginica  versicolor versicolor versicolor versicolor
[29] virginica  versicolor versicolor versicolor versicolor versicolor virginica 
[36] versicolor versicolor versicolor versicolor versicolor virginica  virginica 
[43] virginica  versicolor virginica  virginica  virginica  virginica  virginica 
[50] virginica  virginica  versicolor virginica  virginica  virginica  virginica 
[57] virginica  virginica  virginica  versicolor
Levels: setosa versicolor virginica
```

The KNN model is fitted with a train, test, and k value. Also, the Classifier Species feature is fitted in the model.

- **Confusion Matrix:**

```
> cm
            classifier_knn
             setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         17         3
  virginica       0          3        17
```

So, 20 Setosa are correctly classified as Setosa. Out of 20 Versicolor, 17 Versicolor are correctly classified as Versicolor and 3 are classified as virginica. Out of 20 virginica, 17 virginica are correctly classified as virginica and 3 are classified as Versicolor.

- **Model Evaluation:**
  (k=1)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.9"
```

The model achieved 90% accuracy with k is 1.

(K=3)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.883333333333333"
```

The model achieved 88.33% accuracy with k is 3 which is lower than when k was 1.

(K=5)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.916666666666667"
```

The model achieved 91.66% accuracy with k is 5 which is more than when k was 1 and 3.

(K=7)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.933333333333333"
```

The model achieved 93.33% accuracy with k is 7 which is more than when k was 1, 3, and 5.

(K=15)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.95"
```

The model achieved 95% accuracy with k is 15 which is more than when k was 1, 3, 5, and 7.

(K=19)

```
> misClassError <- mean(classifier_knn != test_cl$Species)
> print(paste('Accuracy=', 1-misClassError))
[1] "Accuracy= 0.95"
```

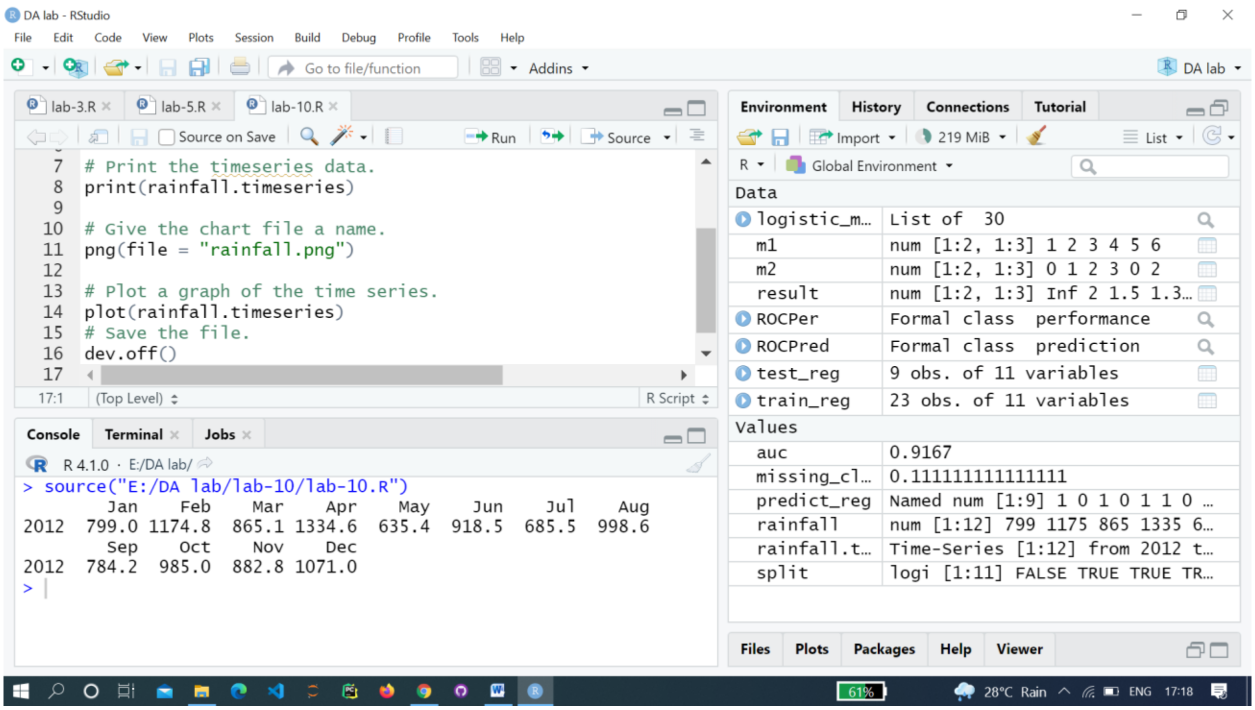
The model achieved 95% accuracy with k is 19 which is more than when k was 1, 3, 5, and 7. Its same accuracy when k was 15 which means now increasing k values doesn't affect the accuracy.

Program 10

Object: Program to perform Time series analysis in R

```
# Get the data points in form of a R vector.
rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object.
rainfall.timeseries<- ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data.
print(rainfall.timeseries)

# Give the chart file a name.
png(file = "rainfall.png")

# Plot a graph of the time series.
plot(rainfall.timeseries)
# Save the file.
dev.off()
```

Output:



Graph: