SRI MUTHUKUMARAN INSTITUTE OF TECHNOLOGY

CHIKKARAYAPURAM(NEAR MANGADU), CHENNAI-69

DEPARTMENT OF COMPUTER APPLICATIONS

ACADEMIC YEAR 2023– 2024/ ODD SEMESTER

CLASS & SEM: II MCA/ III SEM

MC4311- MACHINE LEARNING LABORATORY MANUAL
(R-2021 -2 YEARS)

INDEX

EX.NO.1(i)	**Download, Install And Explore The Features Of R For Data Analytics.**
DATE:04.09.23	

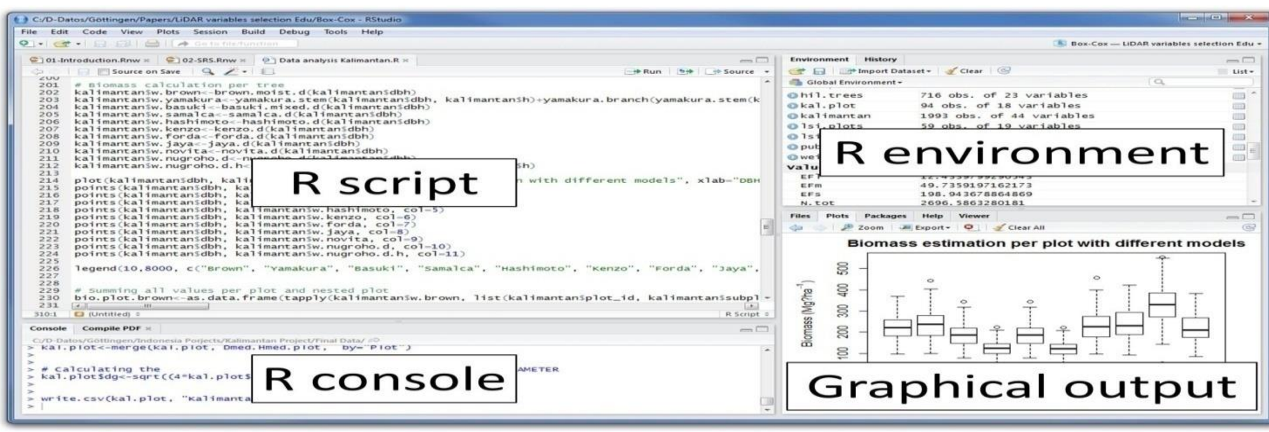
AIM:-To Download, install and explore the features of R for data analytics.

PROCEDURE:-

1. Download R in Windows :

Follow the steps below for installing R Studio:

1. Go to https://www.rstudio.com/products/rstudio/download/
2. In 'Installers for Supported Platforms' section, choose and click the R Studio installer based on your operating system. The download should begin as soon as you click.
3. Click Next..Next..Finish.
4. Download Complete.
5. To Start R Studio, click on its desktop icon or use 'search windows' to access the program. It looks like this:



2. Installation of R Packages

In R, most data handling tasks can be performed in 2 ways: Using R packages and R base functions. To install a package, simply type:

install.packages("package name")

As a first time user, a pop might appear to select your CRAN mirror (country server), choose accordingly and press OK.

Note: You can type this either in console directly and press 'Enter' or in R script and click 'Run'.

3. (i). R AS Calculator Application

a.Using without R objects on console

```
> 2587+2149
[1] 4736
> 287954-135479
[1] 152475
> 257*52
[1] 13364
> 257/21
[1] 12.2381
```

Using with R objects on console:

```
> A=1000
> B=2000
> C=A+B
> C
[1] 3000
> D=A - B
> D
[1] -1000
> E=A * B
> E
[1] 2e+06
>F=A/B
> F
[1] 0.5
```

b. Using mathematical functions on console

```
> a=100
> a=100
> class(a)
[1] "numeric"
> b=500
> c=a-b
> class(b)
[1] "numeric"
> sum<-a-b
> [1] FALSE
> sum
[1] -400
```

c. Write an R script, to create R objects for calculator application and save in a specified location in disk.

> getwd()

[1] "E:/JOSI SMIT21/DS LAB 21/DS RPrg"

>write.csv(a,'diabetes.csv')

> write.csv(a,'file:///E:/JOSI SMIT21/DS LAB 21/DS RPrg/diabetes.csv')

(ii). Descriptive Statistics In R

a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars& cars datasets.

> mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1

Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1

	gear	carb
Mazda RX4	4	4
Mazda RX4 Wag	4	4
Datsun 710	4	1
Hornet 4 Drive	3	1
Hornet Sportabout	3	2
Valiant	3	1
Duster 360	3	4
Merc 240D	4	2
Merc 230	4	2
Merc 280	4	4
Merc 280C	4	4
Merc 450SE	3	3
Merc 450SL	3	3
Merc 450SLC	3	3
Cadillac Fleetwood	3	4
Lincoln Continental	3	4
Chrysler Imperial	3	4
Fiat 128	4	1
Honda Civic	4	2
Toyota Corolla	4	1
Toyota Corona	3	1
Dodge Challenger	3	2
AMC Javelin	3	2
Camaro Z28	3	4
Pontiac Firebird	3	2
Fiat X1-9	4	1
Porsche 914-2	5	2
Lotus Europa	5	2
Ford Pantera L	5	4
Ferrari Dino	5	6
Maserati Bora	5	8
Volvo 142E	4	2

```
> str(mtcars)
'data.frame':	32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
>quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

```
> summary(cars)
     speed           dist
 Min.   : 4.0   Min.   :  2.00
 1st Qu.:12.0   1st Qu.: 26.00
 Median :15.0   Median : 36.00
 Mean   :15.4   Mean   : 42.98
 3rd Qu.:19.0   3rd Qu.: 56.00
 Max.   :25.0   Max.   :120.00
> class(cars)
[1] "data.frame"
```

b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

```
>aggregate(. ~ Species, data = iris, mean)
     Species Sepal.Length Sepal.Width Petal.Length
1     setosa        5.006       3.428        1.462
2 versicolor        5.936       2.770        4.260
3  virginica        6.588       2.974        5.552
  Petal.Width
1       0.246
2       1.326
3       2.026
```

> subset(iris,iris$Sepal.Length==5.0)

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width
5             5         3.6          1.4         0.2
8             5         3.4          1.5         0.2
26            5         3.0          1.6         0.2
27            5         3.4          1.6         0.4
36            5         3.2          1.2         0.2
41            5         3.5          1.3         0.3
44            5         3.5          1.6         0.6
50            5         3.3          1.4         0.2
61            5         2.0          3.5         1.0
94            5         2.3          3.3         1.0
      Species
5      setosa
8      setosa
26     setosa
27     setosa
36     setosa
41     setosa
44     setosa
50     setosa
61 versicolor
94 versicolor
```

Result: Thus RStudio & R is downloaded, installed and explore the features of R for calculator applications and descriptive statistics for data analytics.

EX.NO.1(ii)	Reading And Writing Different Types Of Datasets Using R
DATE:04.09.23	

Aim:- To read and write different types of Dataset using R

a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.

Source Code:-

#Create the following Student DataSet using Microsoft Excel and save it as .c sv file

student.csv

```
  RegNo    Name    Class  MARKS
 2126162001 Priya   IIMCA    70
 2126162002 Kala    IIMCA    78
 2126162003 Lotus   IIMCA    90
 2126162004 Jasmine IIMCA    65
 2126162005 Fathima IIMCA    73
```

 # a. Read data from Student.csv

```
>library(utils)
> data<- read.csv("Student.csv")
> data
```

Output:-

```
       RegNo    Name  Class MARKS
1 2126162001   Priya II MCA    70
2 2126162002    Kala II MCA    78
3 2126162003   Lotus II MCA    90
4 2126162004 Jasmine II MCA    65
5 2126162005 Fathima II MCA    73
```

```
> library(readr)
> data<- read.csv("Student.csv")
> print(is.data.frame(data))
[1] TRUE
> print(ncol(data))
[1] 4
> print(nrow(data))
[1] 5
```

 # Create Employee DataSet

Employee.csv

Id	name	salary	start_date	dept
10001	Mark	30000	12/10/2021	IT
10002	RICK	20000	12/10/2021	HR
10003	Michael	60000	20/11/2021	FINANCE
10004	Gary	70000	20/11/2021	OPERATION
10005	Jasmine	50000	21/12/2021	IT

> # Create a data frame.
> library(readr)
> data<- read_csv("Employee.csv")
Rows: 5 Columns: 5
-- Column specification --------------------------------------
Delimiter: ","
chr (3): name, start_date, dept
dbl (2): Id, salary

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message
.
> # Get the max salary from data frame.
> sal<- max(data$salary)
> # Get the person detail having max salary.
> retval<- subset(data, salary == max(salary))
> retval
A tibble: 1 x 5
    Id name salary start_date dept
  <dbl> <chr> <dbl> <chr>   <chr>
1 10004 Gary   70000 20/11/2021 OPERATION

> #Get all the people working in IT department
> # Create a data frame.
> library(readr)
> data<- read_csv("Employee.csv")
Rows: 5 Columns: 5
-- Column specification --------------------------------------
Delimiter: ","
chr (3): name, start_date, dept

dbl (2): Id, salary
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message
.

```
> retval<- subset( data, dept == "IT")
> retval
# A tibble: 2 x 5
     Id name    salary start_date dept
  <dbl> <chr>    <dbl> <chr>      <chr>
1 10001 Mark     30000 12/10/2021 IT
2 10005 Jasmine  50000 21/12/2021 IT
```

b. Reading Excel data sheet in R.

#Create Emp.xlsx file using Microsoft Excel

Id	name	salary	start_date	dept
10001	Mark	30000	21/12/2021	IT
10002	RICK	20000	21/12/2021	HR
10003	Michael	60000	20/11/2021	FINANCE
10004	Gary	70000	20/11/2021	OPERATION
10005	Jasmine	50000	21/12/2021	IT

Source:-

```
install.packages("xlsx")
# Loading
 library("readxl")
# xlsx files
my_data <- read_excel("Emp.xlsx")
my_data
```

OUTPUT:-

```
# A tibble: 5 x 5
     Id name    salary start_date dept
  <dbl> <chr>    <dbl> <chr>      <chr>
1 10001 Mark     30000 44540      IT
2 10002 RICK     20000 44540      HR
3 10003 Michael  60000 20/11/2021 FINANCE
4 10004 Gary     70000 20/11/2021 OPERATION
5 10005 Jasmine  50000 21/12/2021 IT
```

c. Reading XML dataset in R.

Create Input.xml file using notepad

```
<?xml version="1.0"?>
<Address>
  <No>  1</No>
  <Name> Rick </Name>
  <DNO>458</DNO>
  <Street>SMIT STREET</Street>
  <City> Chennai </City>
  <State>Tamil Nadu</State>
  <Pincode> 600001</Pincode>
</Address>
```

Source code:-

```
 install.packages("XML")
> library("XML")
> library("methods")
> result<- xmlParse(file = "input.xml")
> result
```

Output:-

```
<?xml version="1.0"?>
<Address>
  <No>  1</No>
  <Name> Rick </Name>
  <DNO>458</DNO>
  <Street>SMIT STREET</Street>
  <City> Chennai </City>
  <State>Tamil Nadu</State>
  <Pincode> 600001</Pincode>
</Address>
```

Result:- Thus read and write different types of Dataset using R is executed successfully.

EX.NO.1 (iii)	Structure Data In Machine Learning Using R
DATE:04.09.23	

Aim: To demonstrate how do you structure data in Machine Learning

The most essential data structures used in R include:

1.Vectors:- A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type e.g homogeneous data structures. Vectors are one-dimensional data structures.

Program Code:-

```
# R program to illustrate Vector
# Vectors(ordered collection of same data type)
X = c(1, 3, 5, 7, 8)
 # Printing those elements in console
print(X)
```

Output:-

```
[1] 1 3 5 7 8
```

2.Lists:- A list is a generic object consisting of an ordered collection of objects. Lists are heterogeneous data structures. These are also one-dimensional data structures. A list can be a list of vectors, list of matrices, a list of characters and a list of functions and so on.

Program Code:-

```
# R program to illustrate a List
# The first attributes is a numeric vector
# containing the employee IDs which is
# created using the 'c' command here
empId = c(1, 2, 3, 4)

# The second attribute is the employee name
# which is created using this line of code here
# which is the character vector
empName = c("Debi", "Sandeep", "Subham", "Shiba")
```

The third attribute is the number of employees
which is a single numeric variable.
numberOfEmp = 4
 # We can combine all these three different
data types into a list
containing the details of employees
which can be done using a list command
empList = list(empId, empName, numberOfEmp)
 print(empList)

Output:
[[1]]
[1] 1 2 3 4

[[2]]
[1] "Debi"   "Sandeep" "Subham" "Shiba"

[[3]]
[1] 4

3.Dataframes:- Dataframes are generic data objects of R which are used to store the tabular data. Dataframes are the foremost popular data objects in R programming because we are comfortable in seeing the data within the tabular form. They are two-dimensional, heterogeneous data structures. These are lists of vectors of equal lengths.

Data frames have the following constraints placed upon them:

- A data-frame must have column names and every row should have a unique name.
- Each column must have the identical number of items.
- Each item in a single column must be of the same data type.
- Different columns may have different data types.

To create a data frame we use the data.frame() function.

Program Code:-

```
# R program - dataframe
# A vector which is a character vector
Name = c("Amiya", "Raj", "Asish")
 # A vector which is a character vector
Language = c("R", "Python", "Java")
 # A vector which is a numeric vector
Age = c(22, 25, 45)

# To create dataframe use data.frame command
# and then pass each of the vectors
# we have created as arguments
# to the function data.frame()
df = data.frame(Name, Language, Age)
print(df)
```

Output:

```
Name       Language   Age
1 Amiya     R          22
2  Raj      Python     25
3  Asish    Java       45
```

4. Matrices:-A matrix is a rectangular arrangement of numbers in rows and columns. In a matrix, as we know rows are the ones that run horizontally and columns are the ones that run vertically. Matrices are two-dimensional, homogeneous data structures.

Program Code:-

```
# R program -matrix
A = matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE  )
print(A)
```

OUTPUT:

```
 [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

5. Arrays:- Arrays are the R data objects which store the data in more than two dimensions. Arrays are n-dimensional data structures. For example, if we create an array of dimensions (2, 3, 3) then it creates 3 rectangular matrices each with 2 rows and 3 columns. They are homogeneous data structures.

Program Code:-

```
# R program to illustrate an array
A = array(c(1, 2, 3, 4, 5, 6, 7, 8),   dim = c(2, 2, 2))
print(A)
```

Output:

```
, , 1  [,1] [,2]
[1,]    1    3
[2,]    2    4
, , 2   [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Result:- Thus Structure data in Machine Learning using R is executed successfully

EX.NO.1(iv)	PERFORM UNIVARIATE ANALYSIS FOR PIMA INDIANS DIABETES DATA SET USING R
DATE:11.09.23	

Aim: To Use the Diabetes data set from UCI and Pima Indians Diabetes data set for performing Univariate Analysis such as Frequency, Mean, Median, Mode, Variance, Standard Deviation using R and also compare the results of the above analysis for the two data sets.

SOURCE Code:-Univar2A.R

diabetSet=scan()
1: 6 148 72 35 0 33.6 50 1 1 85 66 29 0 26.6 31 0 8 183 64 0 0 23.3 32 1
25:
Read 24 items

 # Mean of DiabetSet

> mean(diabetSet)
[1] 37.3125

 # Median of DiabetSet

> median(diabetSet)
[1] 27.8

#Mode of DiabetSet

> mode(diabetSet)
[1] "numeric"

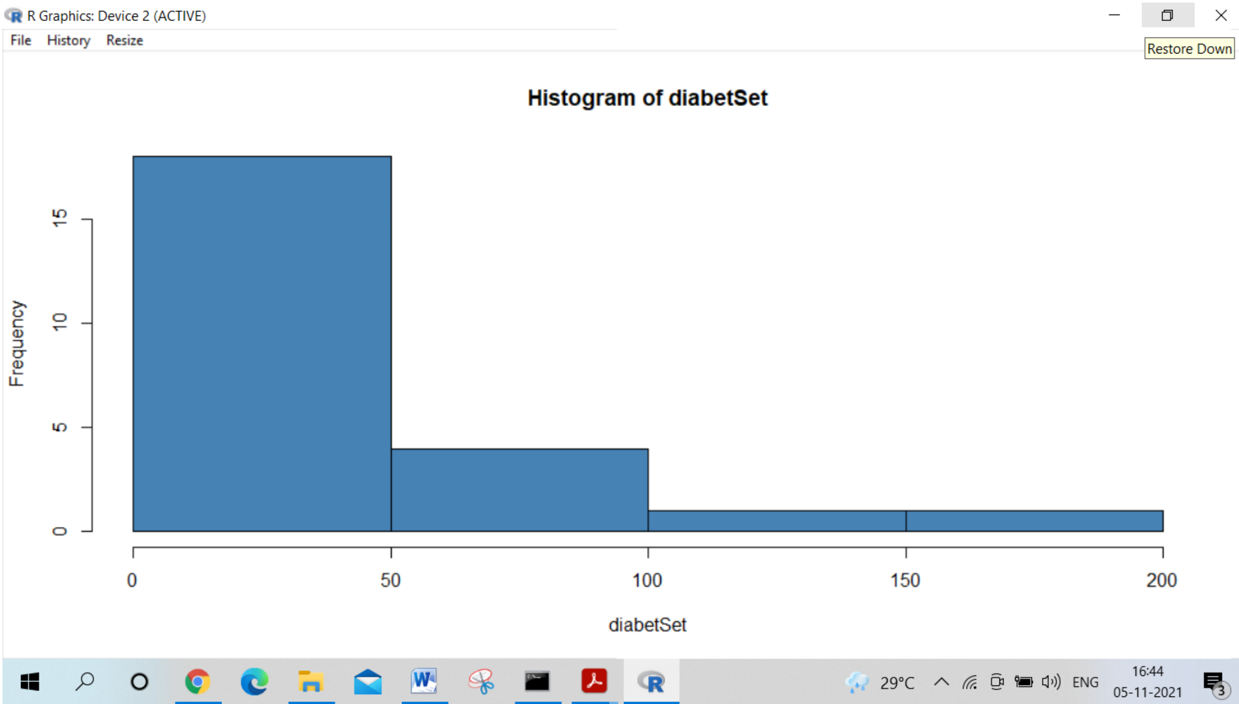
#Variance of DiabetSet

var(diabetSet)
[1] 2255.394

#Standard Deviation of DiabetSet

> sd(diabetSet)
[1] 47.49099

diabetSet<-read.table("D:\Josi(2017-20)\JOSI MCALAb-2017-21\Data Science Lab/diabets.csv",header=TRUE,sep=",")
diabetScore<-diabetSet$BMI
hist(diabetScore,col='steelblue')



diabetSet.csv

6	148	72	35	0	33.6	50	1
1	85	66	29	0	26.6	31	0
8	183	64	0	0	23.3	32	1

Result:-Thus the Univariate Analysis for PIMA diabetes dataset using R is implemented and executed successfully.

EX.NO.1(v)	Perform Bivariate Analysis For Pima Indians Diabetes Data Set Using R
DATE:11.09.23	

Aim: To Use the Diabetes data set from UCI and Pima Indians Diabetes data set for performing Bivariate Analysis such as Linear and logistic regression modeling using R and also compare the results of the above analysis for the two data sets.

Program:-

EXI-2BBIVARDIA.R

```
library(tidyverse)
library(ggplot2)
library(readr)
library(scales)
library(dplyr)
library(reshape2)
library(readxl)
library(corrplot)
# loading data (.csv)

data <- read_csv(/diabetes.csv")
diabetes<- as_tibble(data)
diabetes #display data as tibble
```

output:-

```
Rows: 3 Columns: 8
-- Column specification --------------------------------------
Delimiter: ","
dbl (8): Pregnancies, Glucose, BloodPressure, SkinThicknes...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message
.
tibble [3 x 8] (S3: tbl_df/tbl/data.frame)
```

```
 $ Pregnancies  : num [1:3] 6 1 8
 $ Glucose      : num [1:3] 148 85 183
 $ BloodPressure: num [1:3] 72 66 64
 $ SkinThickness: num [1:3] 35 29 0
 $ Insulin      : num [1:3] 0 0 0
 $ BMI          : num [1:3] 33.6 26.6 23.3
 $ Age          : num [1:3] 50 31 32
 $ Outcome      : num [1:3] 1 0 1
```

> summary(diabetes)

Output:-

```
  Pregnancies    Glucose        BloodPressure
 Min.   :1.0    Min.   : 85.0   Min.   :64.00
 1st Qu.:3.5    1st Qu.:116.5   1st Qu.:65.00
 Median :6.0    Median :148.0   Median :66.00
 Mean   :5.0    Mean   :138.7   Mean   :67.33
 3rd Qu.:7.0    3rd Qu.:165.5   3rd Qu.:69.00
 Max.   :8.0    Max.   :183.0   Max.   :72.00

 SkinThickness     Insulin      BMI              Age
 Min.   : 0.00   Min.   :0   Min.   :23.30   Min.   :31.00
 1st Qu.:14.50   1st Qu.:0   1st Qu.:24.95   1st Qu.:31.50
 Median :29.00   Median :0   Median :26.60   Median :32.00
 Mean   :21.33   Mean   :0   Mean   :27.83   Mean   :37.67
 3rd Qu.:32.00   3rd Qu.:0   3rd Qu.:30.10   3rd Qu.:41.00
 Max.   :35.00   Max.   :0   Max.   :33.60   Max.   :50.00
 Outcome
 Min.   :0.0000
 1st Qu.:0.5000
 Median :1.0000
 Mean   :0.6667
 3rd Qu.:1.0000
 Max.   :1.0000
```

>head(diabetes)

Output:-

```
# A tibble: 3 x 8
  Pregnancies Glucose BloodPressure  SkinThickness Insulin
        <dbl>     <dbl>     <dbl>              <dbl>    <dbl>
1          6       148       72                 35        0
2          1        85       66                 29        0
3          8       183       64                  0        0
# ... with 3 more variables: BMI <dbl>, Age <dbl>,
```

>dim(diabetes) # to find the dimensions of the dataset(Diabetes)

>names(diabetes) # Name of each variable in the dataset

```
[1] "Pregnancies"   "Glucose"       "BloodPressure"
[4] "SkinThickness" "Insulin"       "BMI"
[7] "Age"           "Outcome"
```

>str(diabetes)

output:-

```
tibble [3 x 8] (S3: tbl_df/tbl/data.frame)
 $ Pregnancies  : num [1:3] 6 1 8
 $ Glucose      : num [1:3] 148 85 183
 $ BloodPressure: num [1:3] 72 66 64
 $ SkinThickness: num [1:3] 35 29 0
 $ Insulin      : num [1:3] 0 0 0
 $ BMI          : num [1:3] 33.6 26.6 23.3
 $ Age          : num [1:3] 50 31 32
 $ Outcome      : num [1:3] 1 0 1
```

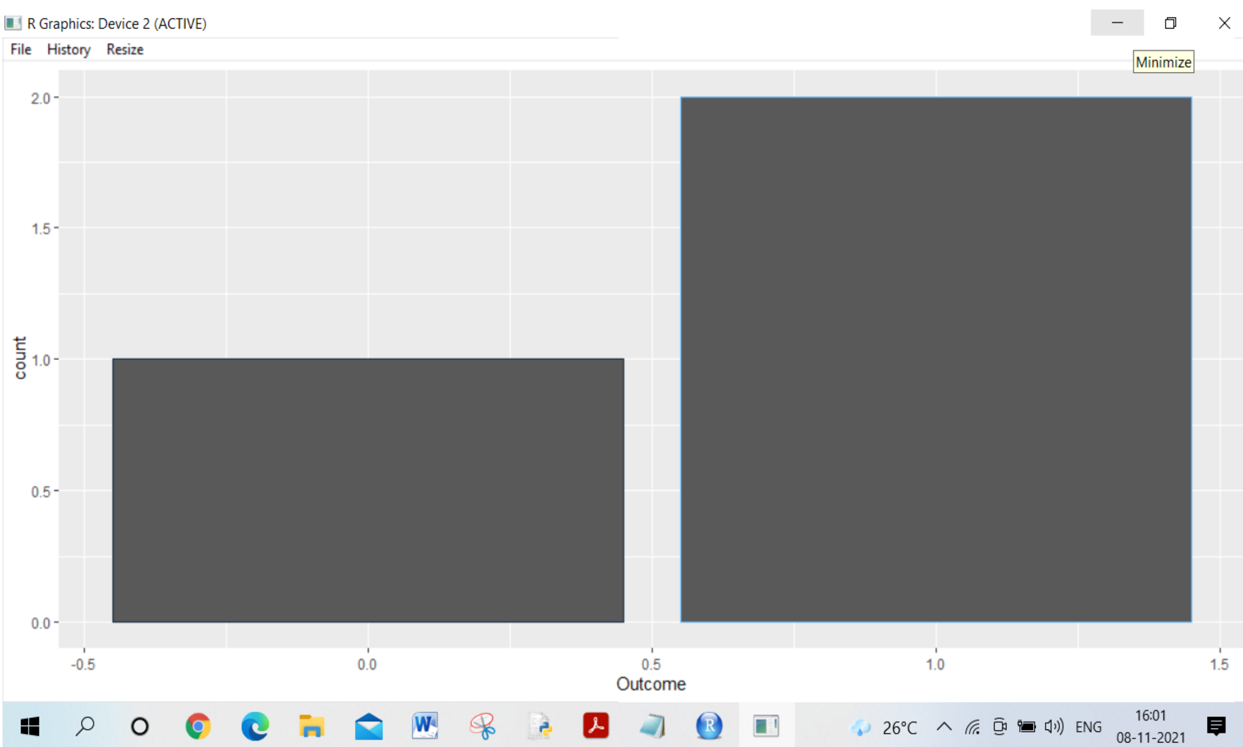
>sapply(diabetes, typeof) # individual feature's datatype in the dataset

output:-

```
  Pregnancies      Glucose BloodPressure  SkinThickness
     "double"     "double"      "double"       "double"
      Insulin          BMI           Age        Outcome
     "double"     "double"      "double"       "double"
```

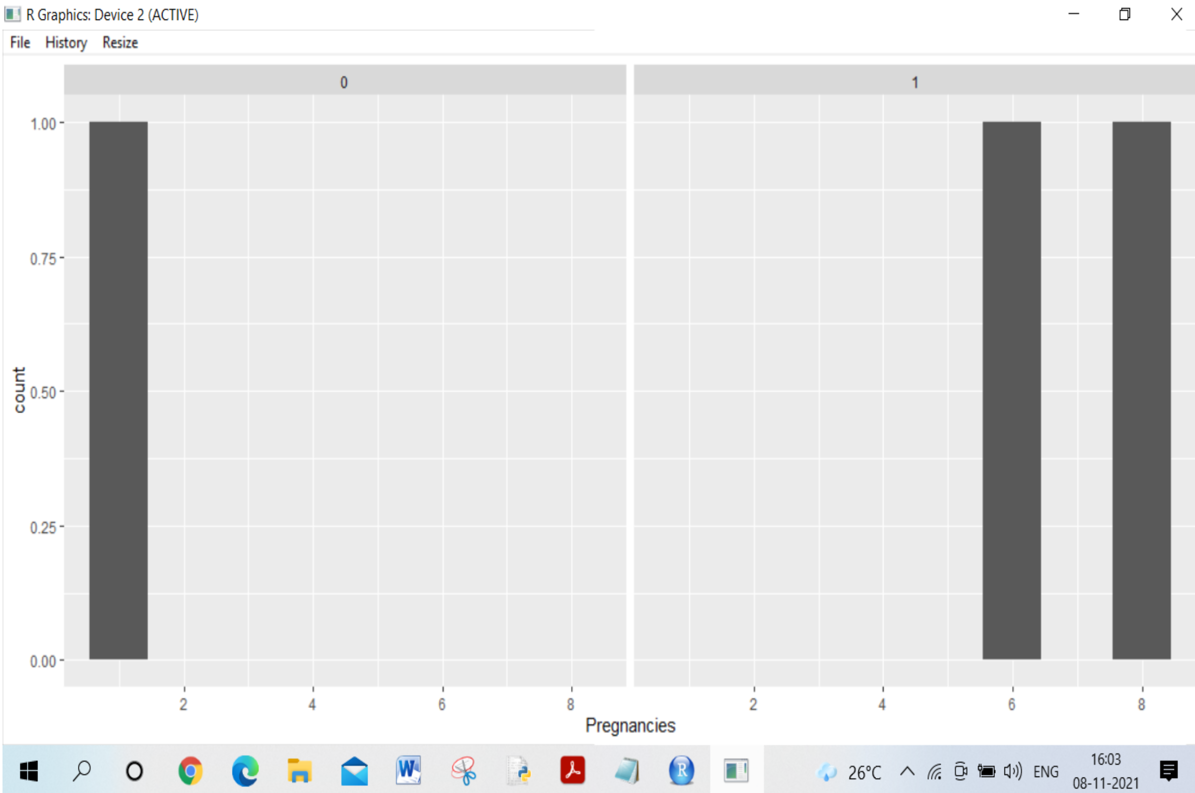
>table(diabetes$Outcome)
output:-
0 1
1 2

>`#` bar chart displaying target variable "Outcome"
>`#` variable, 268 of 768 people have diabetes and 500 are normal
>g <- ggplot(diabetes, aes(Outcome))
>g + geom_bar(aes(group=Outcome, color=Outcome)) + theme(legend.position = "none")



>#Bivariate test
>`#` # ggplot (Pregnancies Outcome)
>g <- ggplot(diabetes, aes(Pregnancies))
>g + geom_bar(aes(group=Outcome)) + facet_wrap(~Outcome) + theme(legend.position = "none")

Output:-

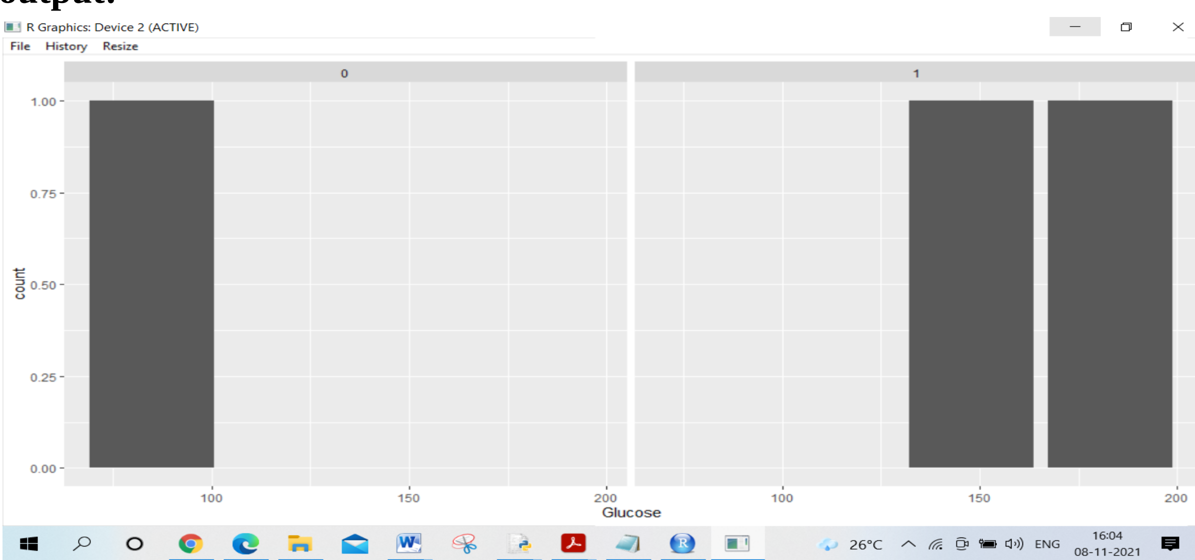


># # ggplot (Glucose and Outcome)

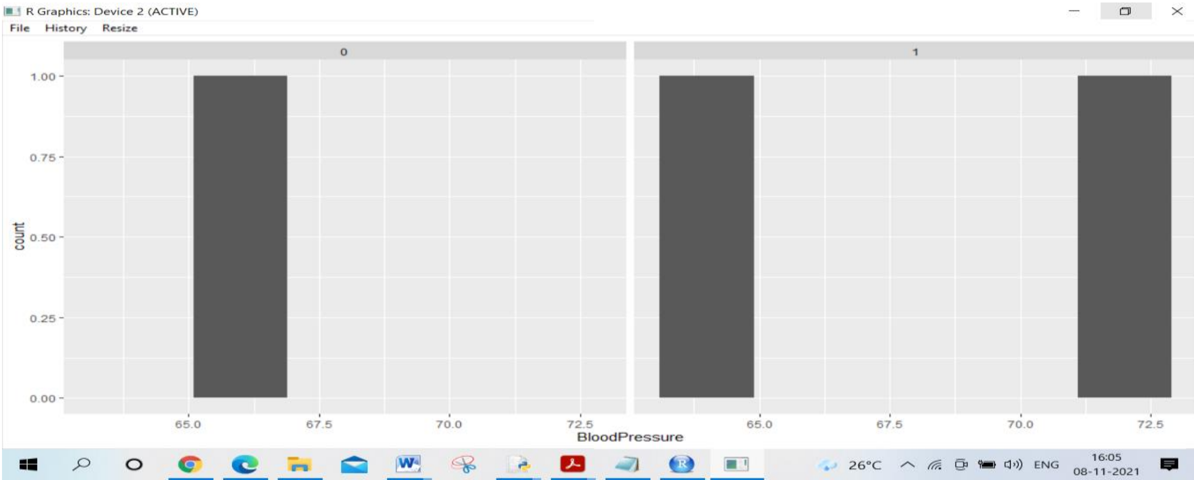
>g <- ggplot(diabetes, aes(Glucose))

>g + geom_bar(aes(group=Outcome)) + facet_wrap(~Outcome) + theme(legend.position = "none")

output:-



># ggplot (BloodPressure and Outcome)
>g <- ggplot(diabetes, aes(BloodPressure))
>g + geom_bar(aes(group=Outcome)) + facet_wrap(~Outcome) + theme(legend.position = "none")
output:-



># ggplot (SkinThickness and Outcome)
>g <- ggplot(diabetes, aes(SkinThickness))
>g + geom_bar(aes(group=Outcome)) + facet_wrap(~Outcome) + theme(legend.position = "none")
Output:-

Result:- Thus the Bivariate Analysis such as Linear and logistic regression modeling using R is implemented and executed successfully.

EX.NO.1(vi)	**Perform Multiple Regression Analysis Compare The Results Of The Above Analysis For The Two Data Sets Using R**
DATE:11.09.23	

Aim: To use the Diabetes data set from UCI and Pima Indians Diabetes data set for performing Multiple Regression Analysis using R and also compare the results of the above analysis for the two data sets.

PROBLEM DEFINATION:

REGRESSION MODEL:

Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS)

SOURCE CODE:

```
>input <- mtcars[,c("mpg","disp","hp","wt")]

> print(head(input))

                  mpg disp hp    wt

Mazda RX4         21.0  160 110 2.620

Mazda RX4 Wag     21.0  160 110 2.875

Datsun 710        22.8  108  93 2.320

Hornet 4 Drive    21.4  258 110 3.215

Hornet Sportabout 18.7  360  175 3.440

Valiant           18.1  225 105 3.460

>> # Create the relationship model.

> model <- lm(mpg~disp+hp+wt, data = input)

>> # Show the model.

> print(model)
```

Output:-

Call:

lm(formula = mpg ~ disp + hp + wt, data = input)

Coefficients:

```
(Intercept)         disp          hp          wt
  37.105505    -0.000937    -0.031157    -3.800891
```

> # Get the Intercept and coefficients as vector elements.

>cat("# # # # The Coefficient Values # # # ","\n")

> a <- coef(model)[1]

> print(a)

Output:-

```
(Intercept)
   37.10551
```

>Xdisp<- coef(model)[2]

>Xhp<- coef(model)[3]

>Xwt<- coef(model)[4]

> x1 = 221

> x2 = 102

> x3 = 2.91

>print(Xdisp)

disp

-0.0009370091

>print(Xhp)

Output:-

hp

-0.03115655

>print(Xwt)

Output:-

wt

-3.800891

> #Create Equation for Regression Model

> Y = a+Xdisp * x1+Xhp * x2+Xwt * x3

> print(Y)

Output:-

(Intercept)

22.65987

Result:-Thus the Multiple Regression Analysis using R is implemented and executed successfully.

EX.NO.1(vii)	Perform Multiple Regression Analysis Compare The Results Of The Above Analysis For The Two Data Sets Using Python
DATE:18.09.23	

Aim: To perform multiple linear regression for a fictitious economy, where the index_price is the dependent variable, and the 2 independent/input variables are:

- interest_rate
- unemployment_rate

SOURCE CODE

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],
        'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
        'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],
        'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],
        'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]
        }
df = pd.DataFrame(data)
print(df)
print("Index Price vs Interst rate:\n")
plt.scatter(df['interest_rate'], df['index_price'], color='red')
plt.title('Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

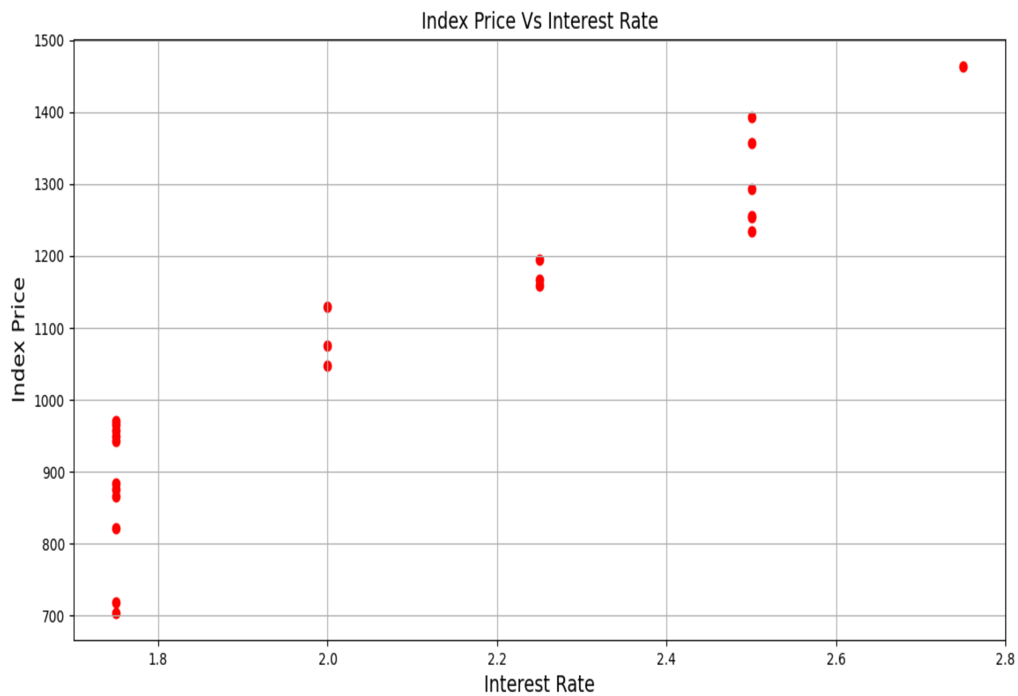
```
print("Unemployement Rate and index Price:\n")
plt.scatter(df['unemployment_rate'], df['index_price'], color='green')
plt.title('Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

Output:-

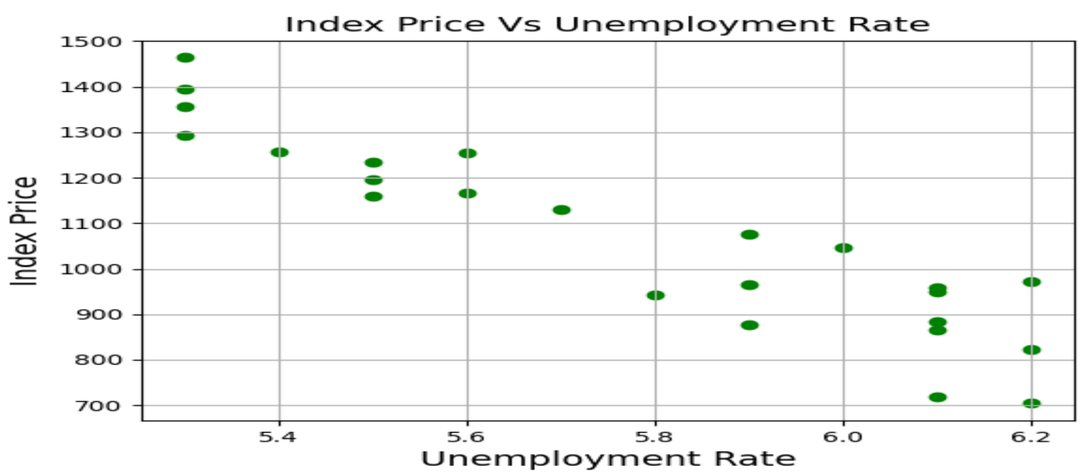
```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
================ RESTART: E:/JosiSMit 2023/MLLAB 23/MultiRegr.py ==

    year  month  interest_rate  unemployment_rate  index_price
0   2017     12           2.75                5.3         1464
1   2017     11           2.50                5.3         1394
2   2017     10           2.50                5.3         1357
3   2017      9           2.50                5.3         1293
4   2017      8           2.50                5.4         1256
5   2017      7           2.50                5.6         1254
6   2017      6           2.50                5.5         1234
7   2017      5           2.25                5.5         1195
8   2017      4           2.25                5.5         1159
9   2017      3           2.25                5.6         1167
10  2017      2           2.00                5.7         1130
11  2017      1           2.00                5.9         1075
12  2016     12           2.00                6.0         1047
13  2016     11           1.75                5.9          965
14  2016     10           1.75                5.8          943
15  2016      9           1.75                6.1          958
16  2016      8           1.75                6.2          971
17  2016      7           1.75                6.1          949
18  2016      6           1.75                6.1          884
19  2016      5           1.75                6.1          866
```

20	2016	4	1.75	5.9	876
21	2016	3	1.75	6.2	822
22	2016	2	1.75	6.2	704
23	2016	1	1.75	6.1	719

Index Price vs Interst rate:



index Price and Unemployement Rate :



Result:-Thus the Multiple Regression Analysis using Python is implemented and executed successfully.

EX.NO. 2	Implement Data Preprocessing Techniques On Real Time Dataset
DATE:18.09.23	

Aim:-To implement data preprocessing techniques on real time dataset

First create dataset data.csv using Microsoft Excel

data.csv

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	NA	Yes
France	35	58000	Yes
Spain	NA	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Procedure & Coding :-

Data preprocessing is the initial phase of Machine Learning where data is prepared for machine learning models.

Step 1: Importing the dataset

```
Dataset = read_csv('data.csv')
view(Dataset)
```

Output:-

	Country	Age	Salary	Purchased
1	France	44	72000	No
2	Spain	27	48000	Yes
3	Germany	30	54000	No
4	Spain	38	61000	No
5	Germany	40	NA	Yes
6	France	35	58000	Yes
7	Spain	NA	52000	No
8	France	48	79000	Yes
9	Germany	50	83000	No
10	France	37	67000	Yes

#Handling the missing data

Step 2:- Replace the missing Age data with the average of the feature in which the data is missing:

```
Dataset$Age = ifelse(is.na(Dataset$Age),
                     ave(Dataset$Age, FUN = function (x)mean(x, na.rm = TRUE)),
                        Dataset$Age)
```

Output:-

	Country	Age	Salary	Purchased
1	France	44.00000	72000	No
2	Spain	27.00000	48000	Yes
3	Germany	30.00000	54000	No
4	Spain	38.00000	61000	No
5	Germany	40.00000	NA	Yes
6	France	35.00000	58000	Yes
7	Spain	38.77778	52000	No
8	France	48.00000	79000	Yes
9	Germany	50.00000	83000	No
10	France	37.00000	67000	Yes

```
Dataset$Salary = ifelse(is.na(Dataset$Salary),
                        ave(Dataset$Salary, FUN = function (x)mean(x, na.rm =
     TRUE)),  Dataset$Salary)
```

Output:-

	Country	Age	Salary	Purchased
1	France	44.00000	72000.00	No
2	Spain	27.00000	48000.00	Yes
3	Germany	30.00000	54000.00	No
4	Spain	38.00000	61000.00	No
5	Germany	40.00000	63777.78	Yes
6	France	35.00000	58000.00	Yes
7	Spain	38.77778	52000.00	No
8	France	48.00000	79000.00	Yes
9	Germany	50.00000	83000.00	No
10	France	37.00000	67000.00	Yes

Step 3: Encoding categorical data

```
Dataset$Country = factor(Dataset$Country,
                         levels = c('France','Spain','Germany'),
                         labels = c(1.0, 2.0 , 3.0 ))
```

Output:-

	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	No
2	2	27.00000	48000.00	Yes
3	3	30.00000	54000.00	No
4	2	38.00000	61000.00	No
5	3	40.00000	63777.78	Yes
6	1	35.00000	58000.00	Yes
7	2	38.77778	52000.00	No
8	1	48.00000	79000.00	Yes
9	3	50.00000	83000.00	No
10	1	37.00000	67000.00	Yes

#the purchased column.

Dataset$Purchased = factor(Dataset$Purchased,
levels = c('No', 'Yes'),
labels = c(0, 1))
Dataset$Purchased[is.na(Dataset$Purchased)] <- 0
as.factor(Dataset$Purchased)

Output:-

	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	0
2	2	27.00000	48000.00	1
3	3	30.00000	54000.00	0
4	2	38.00000	61000.00	0
5	3	40.00000	63777.78	1
6	1	35.00000	58000.00	1
7	2	38.77778	52000.00	0
8	1	48.00000	79000.00	1
9	3	50.00000	83000.00	0
10	1	37.00000	67000.00	1

Step 4: Splitting the dataset into the training and test set

#Using our dataset, let's split it into the training and test sets.
#To begin with, we first load the required library.
library(caTools)# required library for data splition
set.seed(123)
split = sample.split(Dataset$Purchased, SplitRatio = 0.8)

returns true if observation goes to the Training set and false if observation goes #to the test set.

#Creating the training set and test set separately

training_set = subset(Dataset, split == TRUE)

test_set = subset(Dataset, split == FALSE)

training_set

test_set

Output:

Training Set

```
> training_set
   Country      Age   Salary Purchased
1        1 44.00000 72000.00         0
2        2 27.00000 48000.00         1
3        3 30.00000 54000.00         0
4        2 38.00000 61000.00         0
5        3 40.00000 63777.78         1
7        2 38.77778 52000.00         0
8        1 48.00000 79000.00         1
10       1 37.00000 67000.00         1
```

Test Set:-

```
> test_set
  Country Age Salary Purchased
6       1  35  58000         1
9       3  50  83000         0
>
```

returns true if observation goes to the Training set and false if observation goes to the test set.

Step 5:- Feature scale

training_set[, 2:3] = scale(training_set[, 2:3])

test_set[, 2:3] = scale(test_set[, 2:3])

training_set

test_set

Output:-

Training Set:-

```
> training_set
   Country           Age      Salary Purchased
1        1  0.90101716  0.9392746         0
2        2 -1.58847494 -1.3371160         1
3        3 -1.14915281 -0.7680183         0
4        2  0.02237289 -0.1040711         0
5        3  0.31525431  0.1594000         1
7        2  0.13627122 -0.9577176         0
8        1  1.48678000  1.6032218         1
10       1 -0.12406783  0.4650265         1
```

Test Set:-

```
> test_set
  Country        Age     Salary Purchased
6       1 -0.7071068 -0.7071068         1
9       3  0.7071068  0.7071068         0
>
```

Result:- Thus the program is implemented for data preprocessing techniques on real time dataset is executed successfully.

EX.NO.3.	Implement Boruta Feature Subset Selection Techniques Using R
DATE:25.09.23	

AIM:- to implement Boruta Feature subset selection techniques using R

Program:-

```
# Load Packages and prepare dataset
library(TH.data)
library(caret)
data("GlaucomaM", package = "TH.data")
trainData <- GlaucomaM
head(trainData)
```

Output:-

Glaucoma Dataset

	ag	at	as	an	ai	eag	eat	eas	ean	eai	···	tmt	tms	tmn	tmi	mr	rnf	mdic	emd	mv	Class
2	2.220	0.354	0.580	0.686	0.601	1.267	0.336	0.346	0.255	0.331	···	-0.018	-0.230	-0.510	-0.158	0.841	0.410	0.137	0.239	0.035	normal
43	2.681	0.475	0.672	0.868	0.667	2.053	0.440	0.520	0.639	0.454	···	-0.014	-0.165	-0.317	-0.192	0.924	0.256	0.252	0.329	0.022	normal
25	1.979	0.343	0.508	0.624	0.504	1.200	0.299	0.396	0.259	0.246	···	-0.097	-0.235	-0.337	-0.020	0.795	0.378	0.152	0.250	0.029	normal
65	1.747	0.269	0.476	0.525	0.476	0.612	0.147	0.017	0.044	0.405	···	-0.035	-0.449	-0.217	-0.091	0.746	0.200	0.027	0.078	0.023	normal
70	2.990	0.599	0.686	1.039	0.667	2.513	0.543	0.607	0.871	0.492	···	-0.105	0.084	-0.012	-0.054	0.977	0.193	0.297	0.354	0.034	normal
16	2.917	0.483	0.763	0.901	0.770	2.200	0.462	0.637	0.504	0.597	···	0.087	0.018	-0.094	-0.051	0.965	0.339	0.333	0.442	0.028	normal

```
# install.packages('Boruta')
library(Boruta)
# Perform Boruta search
boruta_output <- Boruta(Class ~ ., data=na.omit(trainData), doTrace=0)
names(boruta_output)
```

OUTPUT:-

1. 'finalDecision'
2. 'ImpHistory'
3. 'pValue'
4. 'maxRuns'
5. 'light'
6. 'mcAdj'
7. 'timeTaken'
8. 'roughfixed'
9. 'call'
10. 'impSource'

Get significant variables including tentatives
boruta_signif <- getSelectedAttributes(boruta_output, withTentative = TRUE)
print(boruta_signif)

Output:-

```
 [1] "as"   "ean"  "abrg" "abrs" "abrn" "abri" "hic"  "mhcg" "mhcn" "mhci"
[11] "phcg" "phcn" "phci" "hvc"  "vbss" "vbsn" "vbsi" "vasg" "vass" "vasi"
[21] "vbrg" "vbrs" "vbrn" "vbri" "varg" "vart" "vars" "varn" "vari" "mdn"
[31] "tmg"  "tmt"  "tms"  "tmn"  "tmi"  "rnf"  "mdic" "emd"
```

Do a tentative rough fix
roughFixMod <- TentativeRoughFix(boruta_output)
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)

Output:-

```
 [1] "abrg" "abrs" "abrn" "abri" "hic"  "mhcg" "mhcn" "mhci" "phcg" "phcn"
[11] "phci" "hvc"  "vbsn" "vbsi" "vasg" "vbrg" "vbrs" "vbrn" "vbri" "varg"
[21] "vart" "vars" "varn" "vari" "tmg"  "tms"  "tmi"  "rnf"  "mdic" "emd"
```

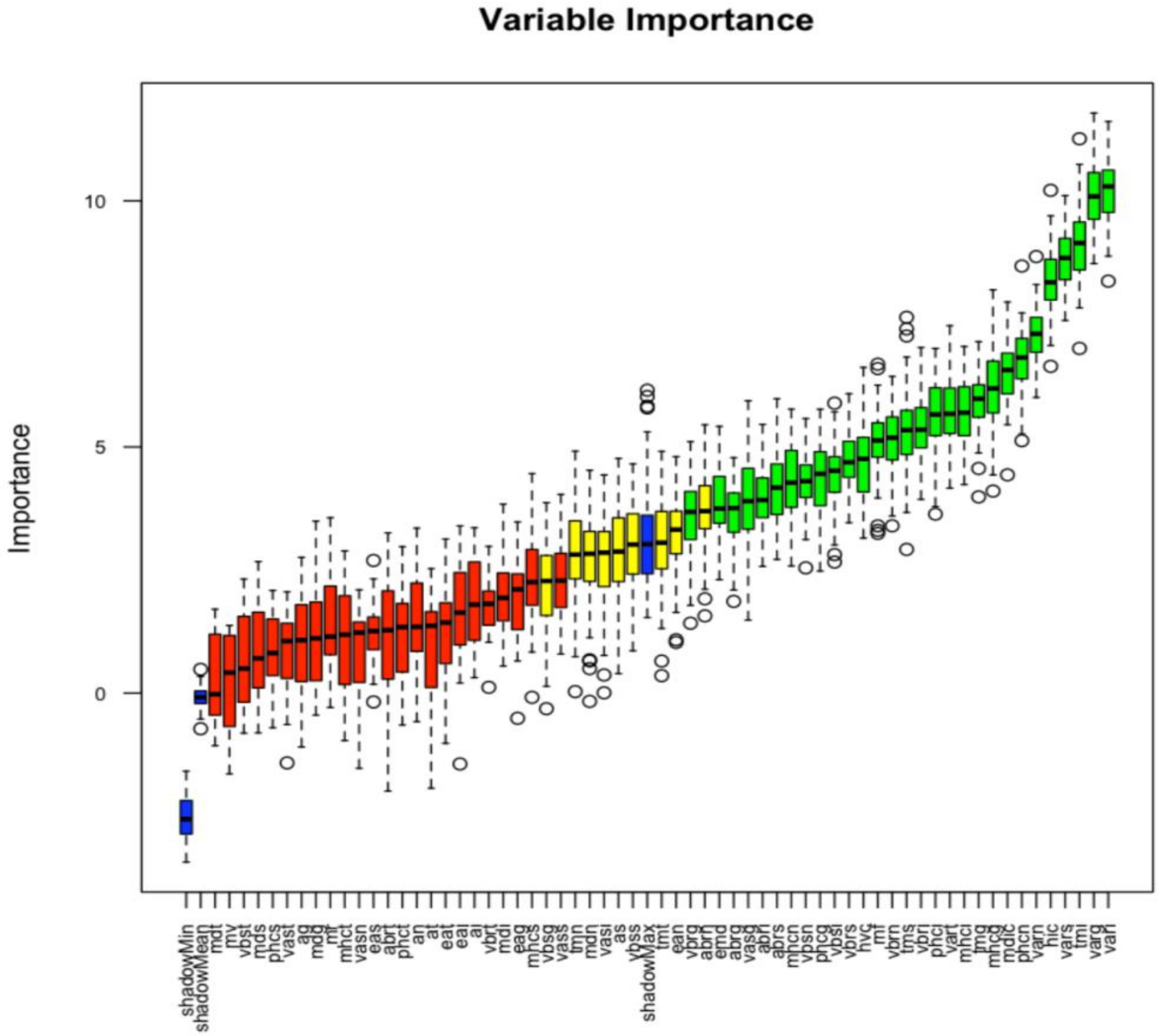
 # Variable Importance Scores
imps <- attStats(roughFixMod)
imps2 = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]
head(imps2[order(-imps2$meanImp),]) # descending sort

Output:-

	Mean Imp	decision
varg	10.279747	Confirmed
vari	10.245936	Confirmed
tmi	9.067300	Confirmed
vars	8.690654	Confirmed
hic	8.324252	Confirmed
varn	7.327045	Confirmed

 # Plot variable importance
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")

Output:-



Result: Thus the feature subset selection techniques in R is implemented successfully.

EX.NO.4.	MEASURE THE PERFORMANCE OF A MACHINE LEARNING MODEL
DATE:25.09.23	

Aim:- To demonstrate how will you measure the performance of a machine learning model

Program:- Modelperf.py

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import log_loss
X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]
results = confusion_matrix(X_actual, Y_predic)
print ('Confusion Matrix :')
print(results)
print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))
print ('Classification Report : ')
print (classification_report(X_actual, Y_predic))
print('AUC-ROC:',roc_auc_score(X_actual, Y_predic))
print('LOGLOSS Value is',log_loss(X_actual, Y_predic))
```

Output:-

```
Confusion Matrix :
[[3 3]
 [1 3]]
Accuracy Score is 0.6
Classification Report :
              precision    recall  f1-score   support

           0       0.75      0.50      0.60         6
           1       0.50      0.75      0.60         4
    accuracy                           0.60        10
   macro avg       0.62      0.62      0.60        10
weighted avg       0.65      0.60      0.60        10
AUC-ROC: 0.625
LOGLOSS Value is 13.815750437193334
```

Result:- Thus measure the performance of a machine learning model is executed successfully.

EX.NO. 5.	Implement The Naïve Bayesian Classifier For Sample Training Data Set Stored As A tennisdata.csv File.
DATE:09.10.23	

Aim:

To implement the naïve Bayesian classifier for a sample training data set stored as a naivedata.CSV file. Compute the accuracy of the classifier, considering few test data sets.

tennisdata.csv

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	FALSE	Yes
Sunny	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Mild	High	TRUE	No

PROGRAM :-NaiveBay.py

```
# import necessary libarities
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

# load data from CSV
data = pd.read_csv('tennisdata.csv')
print("THe first 5 values of data is :\n",data.head())
# obtain Train data and Train output
```

```
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
# Convert then in numbers
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)

classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

Output:-

```
THe first 5 values of data is :
   Outlook Temperature Humidity  Windy PlayTennis
0    Sunny         Hot     High  False         No
1    Sunny         Hot     High   True         No
```

2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes

The First 5 values of train data is

	Outlook	Temperature	Humidity	Windy
0	Sunny	Hot	High	False
1	Sunny	Hot	High	True
2	Overcast	Hot	High	False
3	Rainy	Mild	High	False
4	Rainy	Cool	Normal	False

The first 5 values of Train output is
0 No
1 No
2 Yes
3 Yes
4 Yes
Name: PlayTennis, dtype: object

Now the Train data is :

	Outlook	Temperature	Humidity	Windy
0	2	1	0	0
1	2	1	0	1
2	0	1	0	0
3	1	2	0	0
4	1	0	1	0

Now the Train output is
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Accuracy is: 0.3333333333333333

Result:-Thus the naïve Bayesian classifier for a sample training data set stored as tennisdata.csv file is implemented and executed successfully.

EX.NO.6	Construct A Bayesian Network For Medical Data
DATE:16.10.23	

AIM:- To construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set using Python.

#create the dataset such as data7_names.csv and data7_heart.csv using Microsoft excel.

data7_names.csv

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope
ca thal heartdisease

data7_heart.csv

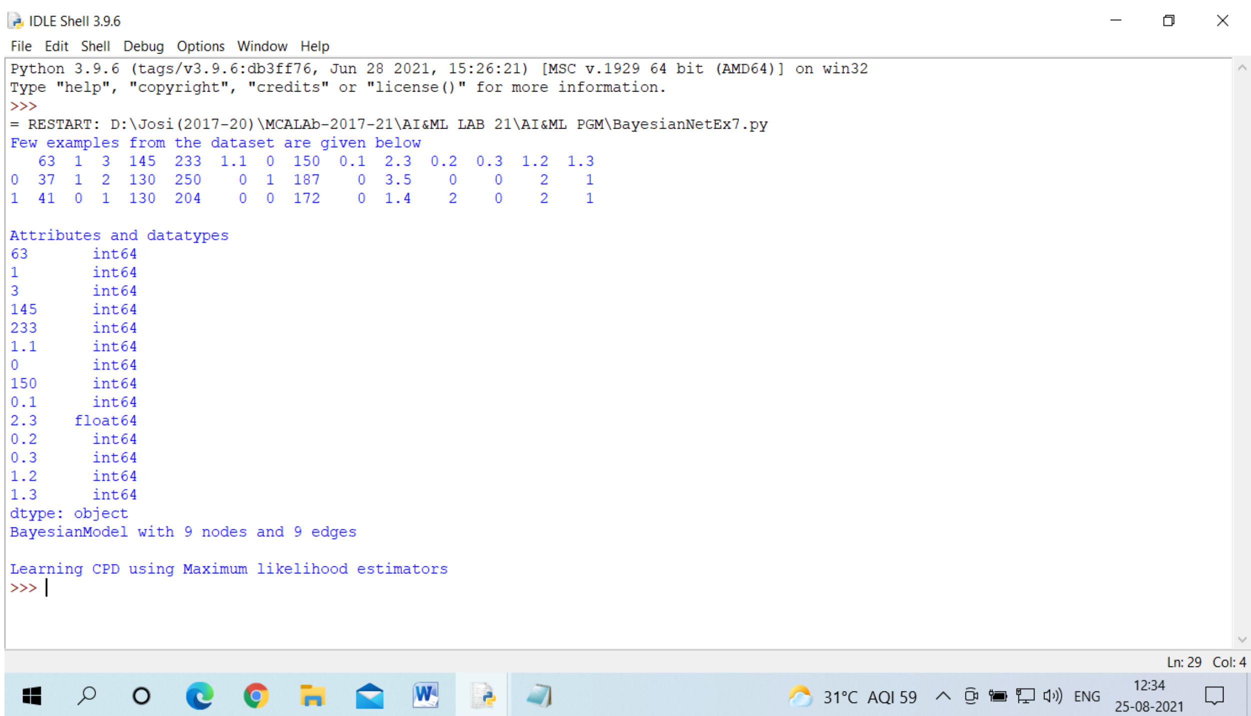
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1

PROGRAM:-BayesMed.py

```
import  numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
#Read the attributes
lines=list(csv.reader(open('data7_names.csv','r')))
attributes=lines[0]
#Read Cleveland Heart disease data
heartDisease = pd.read_csv('data7_heart.csv')
heartDisease = heartDisease.replace('?',np.nan)
#Display data
print('Few examples from the dataset are given below')
print(heartDisease.head())
print ('\nAttributes and datatypes')
print(heartDisease.dtypes)
```

```
#Model Bayesian Network model
BayesianModel([('age','trestbps'),('age','fbs'),('sex','trestbps'),('exang','trestbps'),('trestbps','heartdisease'),('fbs','heartdisease'),('heartdisease','restecg'),('heartdisease','thalach'),('heartdisease','chol')])
print(model)
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)
print('\n Inferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)
print('\n 1. Probability of HeartDisease given Age=28')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence={'age':28})
print(q['heartdisease'])
print('\n 2. Probability of HeartDisease given cholesterol=100')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence={'chol':100})
print(q['heartdisease'])
```

OUTPUT:-



RESULT: Thus a Bayesian Network considering medical data is constructed and executed successfully.

EX.NO.7	**Apply EM Algorithm To Cluster A Set Of Data Stored In A .csv File Using Python**
DATE:23.10.23	

AIM:- To Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering using Python.

PROGRAM:- EMEx7.py

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np
#import matplotlib inline
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
#colormap = np.array(['red', 'lime', 'black'])
# K Means Cluster
model = KMeans(n_clusters=3)
model.fit(X)
# This is what KMeans thought
model.labels_
# View the results
# Set the size of the plot
plt.figure(figsize=(14,7))
# Create a colormap
colormap = np.array(['red', 'lime', 'black'])
# Plot the Original Classifications
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
```

```
# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
# View the results
# Set the size of the plot
plt.figure(figsize=(14,7))
# Create a colormap
#print('The accuracy score : ',sm.accuracy_score(y, model.labels_))
#sm.confusion_matrix(y, model.labels_)
predY = np.choose(model.labels_, [0, 1, 2]).astype(np.int64)
print (predY)
#colormap = np.array(['red', 'lime', 'black'])
# Plot Orginal
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')

# Plot Predicted with corrected values
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length,X.Petal_Width, c=colormap[predY], s=40)
plt.title('K Mean Classification')
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean: ',sm.confusion_matrix(y, model.labels_))
fromsklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
#xs.sample(5)
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)
y_cluster_gmm = gmm.predict(xs)
```

#y_cluster_gmm

plt.subplot(2, 2, 3)

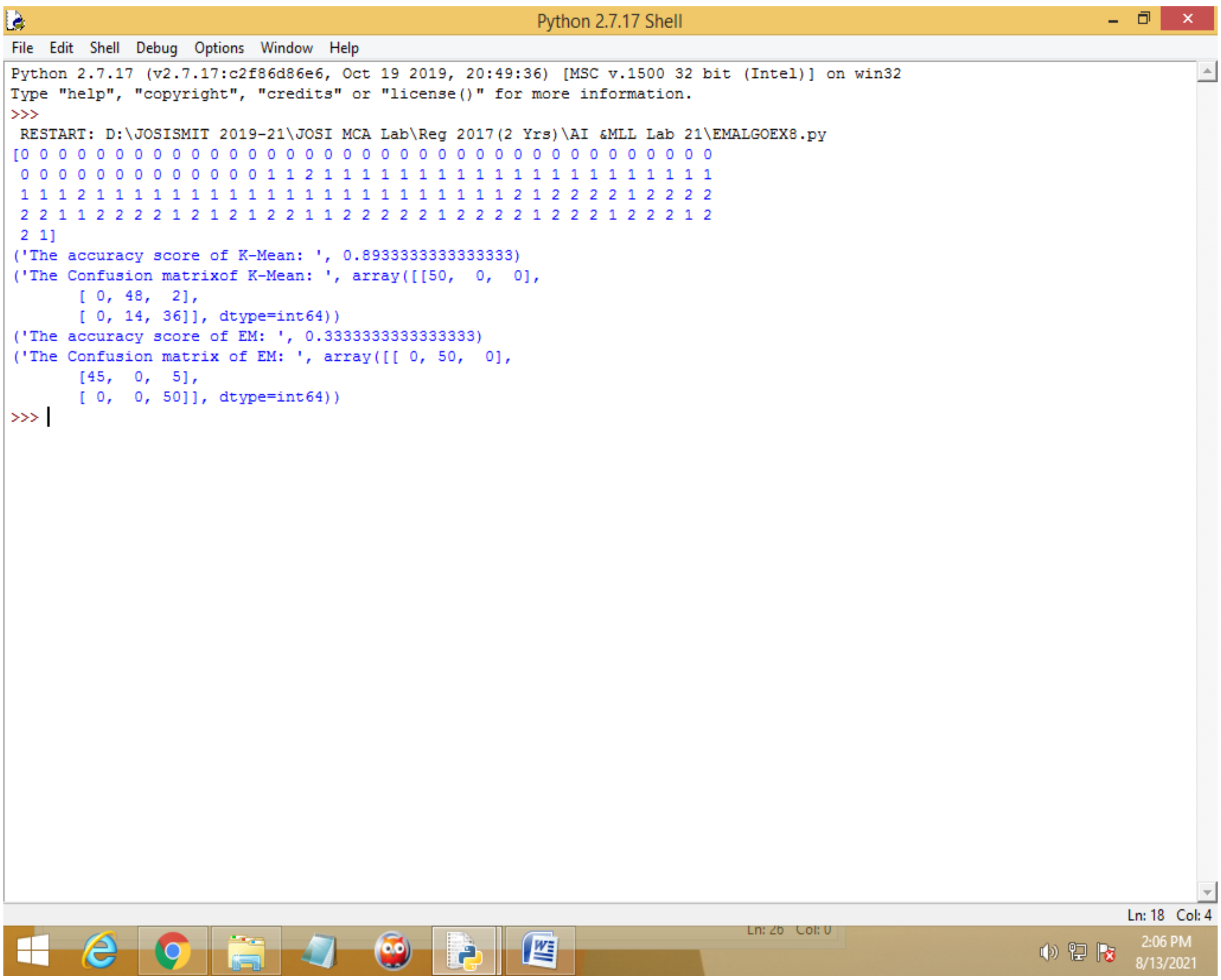
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_cluster_gmm], s=40)

plt.title('GMM Classification')

print('The accuracy score of EM: ',sm.accuracy_score(y, y_cluster_gmm))

print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_cluster_gmm))

OUTPUT:-



Result:- Thus Apply EM algorithm using k-Means algorithm is implemented and executed successfully.

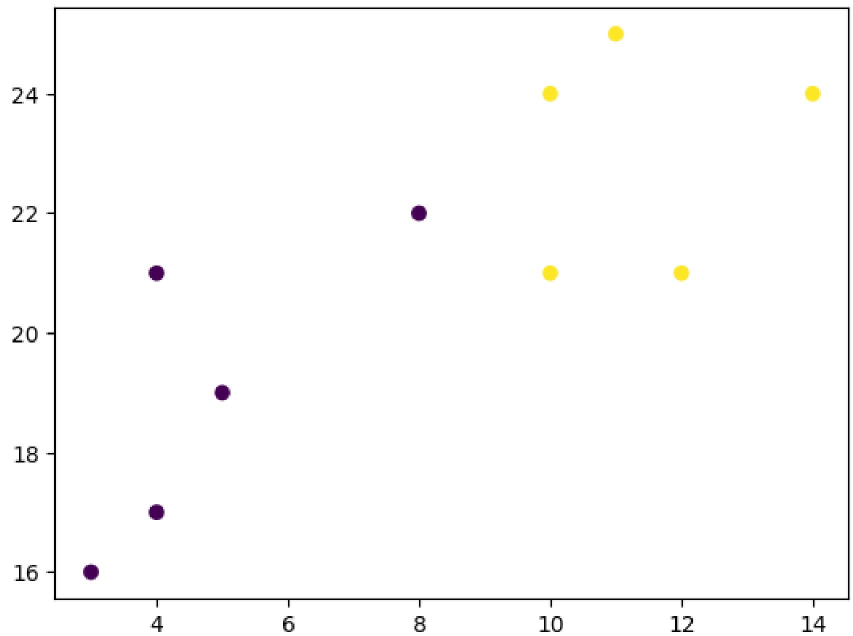
EX.NO.8.(i)	**Implement K-Nearest Neighbour Algorithm using Python**
DATE:30.10.23	

Aim:
To write a program to implement k-Nearest Neighbour algorithm for predictions using Python

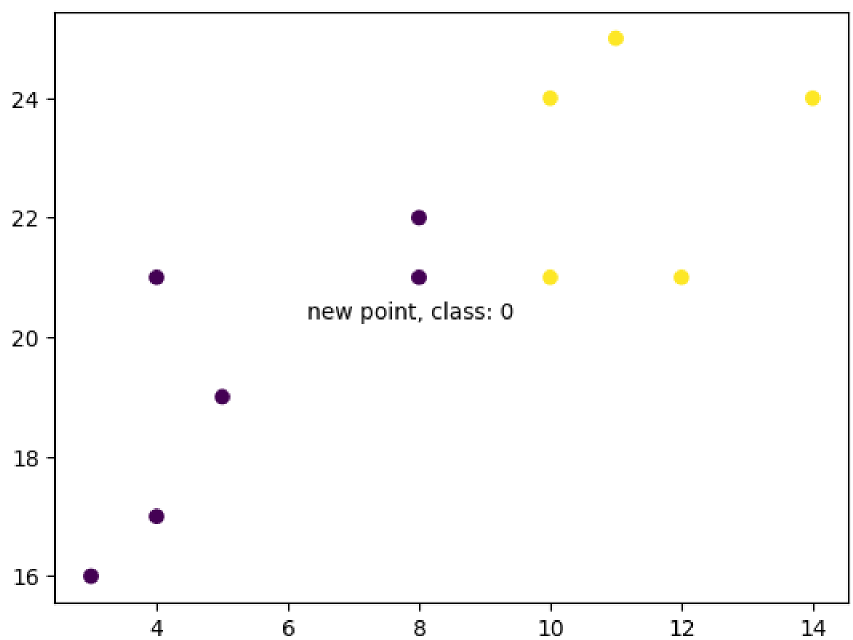
PROGRAM :-KNearNei.py

```
import  matplotlib.pyplot as plt
x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
plt.scatter(x, y, c=classes)
plt.show()
from sklearn.neighbors import KNeighborsClassifier
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(data, classes)
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(data, classes)
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

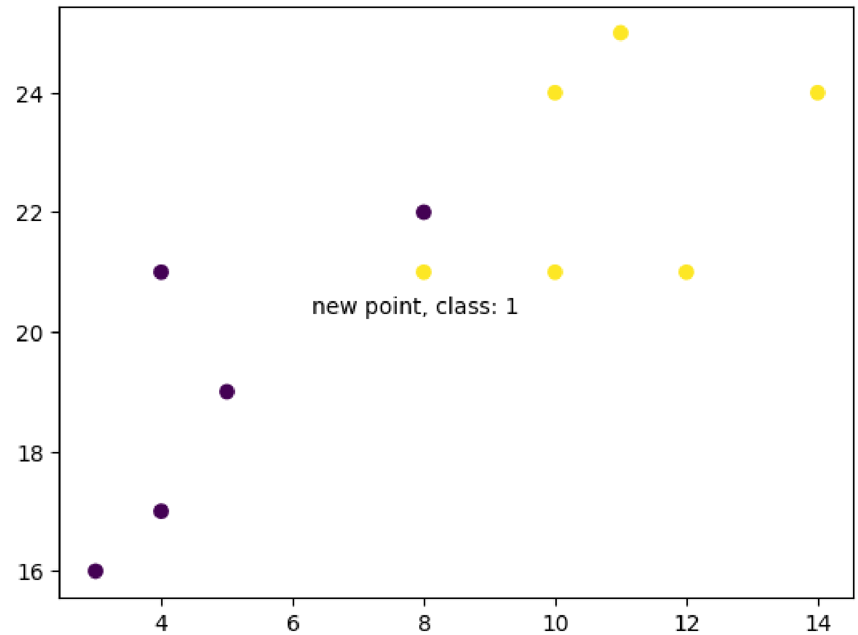
Output:-



24
22
20
18
16
4
6
8
10
12
14



24
22
20
18
16
new point, class: 0
4
6
8
10
12
14



Result:- Thus k-Nearest Neighbour algorithm to classify the iris data set is implemented and executed successfully.

EX.NO.8.(ii)	**Implement K-Nearest Neighbour Algorithm To Classify The iris Data Set using Python**
DATE:30.10.23	

Aim:-To write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions using Python

#create iris.csv using Microsoft excel

iris.csv

sepal_length	sepal_width	petal_length	petal_width	Variety
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa
5.1	3.7	1.5	0.4	Iris-setosa
4.6	3.6	1	0.2	Iris-setosa
5.1	3.3	1.7	0.5	Iris-setosa
4.8	3.4	1.9	0.2	Iris-setosa
5	3	1.6	0.2	Iris-setosa
5	3.4	1.6	0.4	Iris-setosa
5.2	3.5	1.5	0.2	Iris-setosa
5.2	3.4	1.4	0.2	Iris-setosa
4.7	3.2	1.6	0.2	Iris-setosa

4.8	3.1	1.6	0.2	Iris-setosa
5.4	3.4	1.5	0.4	Iris-setosa
5.2	4.1	1.5	0.1	Iris-setosa
5.5	4.2	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5	3.2	1.2	0.2	Iris-setosa
5.5	3.5	1.3	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5	3.3	1.4	0.2	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5	2	3.5	1	Iris-versicolor
5.9	3	4.2	1.5	Iris-versicolor
6	2.2	4	1	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor
6.7	3.1	4.4	1.4	Iris-versicolor
5.6	3	4.5	1.5	Iris-versicolor
5.8	2.7	4.1	1	Iris-versicolor
6.2	2.2	4.5	1.5	Iris-versicolor
5.6	2.5	3.9	1.1	Iris-versicolor

5.9	3.2	4.8	1.8	Iris-versicolor
6.1	2.8	4	1.3	Iris-versicolor
6.3	2.5	4.9	1.5	Iris-versicolor
6.1	2.8	4.7	1.2	Iris-versicolor
6.4	2.9	4.3	1.3	Iris-versicolor
6.6	3	4.4	1.4	Iris-versicolor
6.8	2.8	4.8	1.4	Iris-versicolor
6.7	3	5	1.7	Iris-versicolor
6	2.9	4.5	1.5	Iris-versicolor
5.7	2.6	3.5	1	Iris-versicolor
5.5	2.4	3.8	1.1	Iris-versicolor
5.5	2.4	3.7	1	Iris-versicolor
5.8	2.7	3.9	1.2	Iris-versicolor
6	2.7	5.1	1.6	Iris-versicolor
5.4	3	4.5	1.5	Iris-versicolor
6	3.4	4.5	1.6	Iris-versicolor
6.7	3.1	4.7	1.5	Iris-versicolor
6.3	2.3	4.4	1.3	Iris-versicolor
5.6	3	4.1	1.3	Iris-versicolor
5.5	2.5	4	1.3	Iris-versicolor
5.5	2.6	4.4	1.2	Iris-versicolor
6.1	3	4.6	1.4	Iris-versicolor
5.8	2.6	4	1.2	Iris-versicolor
5	2.3	3.3	1	Iris-versicolor
5.6	2.7	4.2	1.3	Iris-versicolor
5.7	3	4.2	1.2	Iris-versicolor
5.7	2.9	4.2	1.3	Iris-versicolor
6.2	2.9	4.3	1.3	Iris-versicolor
5.1	2.5	3	1.1	Iris-versicolor
5.7	2.8	4.1	1.3	Iris-versicolor
6.3	3.3	6	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica
6.5	3	5.8	2.2	Iris-virginica
7.6	3	6.6	2.1	Iris-virginica
4.9	2.5	4.5	1.7	Iris-virginica
7.3	2.9	6.3	1.8	Iris-virginica
6.7	2.5	5.8	1.8	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica

6.5	3.2	5.1	2	Iris-virginica
6.4	2.7	5.3	1.9	Iris-virginica
6.8	3	5.5	2.1	Iris-virginica
5.7	2.5	5	2	Iris-virginica
5.8	2.8	5.1	2.4	Iris-virginica
6.4	3.2	5.3	2.3	Iris-virginica
6.5	3	5.5	1.8	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	2.6	6.9	2.3	Iris-virginica
6	2.2	5	1.5	Iris-virginica
6.9	3.2	5.7	2.3	Iris-virginica
5.6	2.8	4.9	2	Iris-virginica
7.7	2.8	6.7	2	Iris-virginica
6.3	2.7	4.9	1.8	Iris-virginica
6.7	3.3	5.7	2.1	Iris-virginica
7.2	3.2	6	1.8	Iris-virginica
6.2	2.8	4.8	1.8	Iris-virginica
6.1	3	4.9	1.8	Iris-virginica
6.4	2.8	5.6	2.1	Iris-virginica
7.2	3	5.8	1.6	Iris-virginica
7.4	2.8	6.1	1.9	Iris-virginica
7.9	3.8	6.4	2	Iris-virginica
6.4	2.8	5.6	2.2	Iris-virginica
6.3	2.8	5.1	1.5	Iris-virginica
6.1	2.6	5.6	1.4	Iris-virginica
7.7	3	6.1	2.3	Iris-virginica
6.3	3.4	5.6	2.4	Iris-virginica
6.4	3.1	5.5	1.8	Iris-virginica
6	3	4.8	1.8	Iris-virginica
6.9	3.1	5.4	2.1	Iris-virginica
6.7	3.1	5.6	2.4	Iris-virginica
6.9	3.1	5.1	2.3	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
6.8	3.2	5.9	2.3	Iris-virginica
6.7	3.3	5.7	2.5	Iris-virginica
6.7	3	5.2	2.3	Iris-virginica
6.3	2.5	5	1.9	Iris-virginica
6.5	3	5.2	2	Iris-virginica
6.2	3.4	5.4	2.3	Iris-virginica
5.9	3	5.1	1.8	Iris-virginica

Program:-

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Read dataset to pandas dataframe
dataset = pd.read_csv("iris.csv", names=names)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
print(X.head())
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.10)
classifier = KNeighborsClassifier(n_neighbors=5).fit(Xtrain, ytrain)
ypred = classifier.predict(Xtest)
i = 0
print ("\n----------------------------------------------------------------------")
print ('%-25s %-25s %-25s' % ('Original Label', 'Predicted Label',
'Correct/Wrong'))
print ("----------------------------------------------------------------------")
for label in ytest:
    print ('%-25s %-25s' % (label, ypred[i]), end="")
    if (label == ypred[i]):
        print (' %-25s' % ('Correct'))
    else:
        print (' %-25s' % ('Wrong'))
    i = i + 1
print ("----------------------------------------------------------------------")
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print ("----------------------------------------------------------------------")
print("\nClassification Report:\n",metrics.classification_report(ytest, ypred))
print ("----------------------------------------------------------------------")
print('Accuracy of the classifer is %0.2f' % metrics.accuracy_score(ytest,ypred))
print ("----------------------------------------------------------------------")
```

Output:-

```
sepal-length  sepal-width  petal-length  petal-width
0          5.1          3.5          1.4          0.2
1          4.9          3.0          1.4          0.2
2          4.7          3.2          1.3          0.2
3          4.6          3.1          1.5          0.2
4          5.0          3.6          1.4          0.2
-------------------------------------------------------------------------
Original Label          Predicted Label          Correct/Wrong
-------------------------------------------------------------------------
Iris-virginica          Iris-virginica           Correct
Iris-setosa             Iris-setosa              Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-virginica          Iris-virginica           Correct
Iris-setosa             Iris-setosa              Correct
Iris-setosa             Iris-setosa              Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-setosa             Iris-setosa              Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct

-------------------------------------------------------------------------
Confusion Matrix:
 [[4 0 0]
 [0 6 0]
 [0 0 5]]
-------------------------------------------------------------------------

Classification Report:
                 precision    recall  f1-score   support
    Iris-setosa     1.00      1.00      1.00         4
Iris-versicolor     1.00      1.00      1.00         6
 Iris-virginica     1.00      1.00      1.00         5
       accuracy                         1.00        15
```

```
   macro avg   1.00      1.00      1.00        15
weighted avg 1.00      1.00      1.00        15
```

Accuracy of the classifer is 1.00

Result:- Thus k-Nearest Neighbour algorithm to classify the iris data set is implemented and executed successfully.

EX.NO.9.(i)	**Apply The Technique Of Pruning For A Noisy Data Monk2 Data, And Derive The Decision Tree From This Data Using Python.**
DATE:06.11.23	

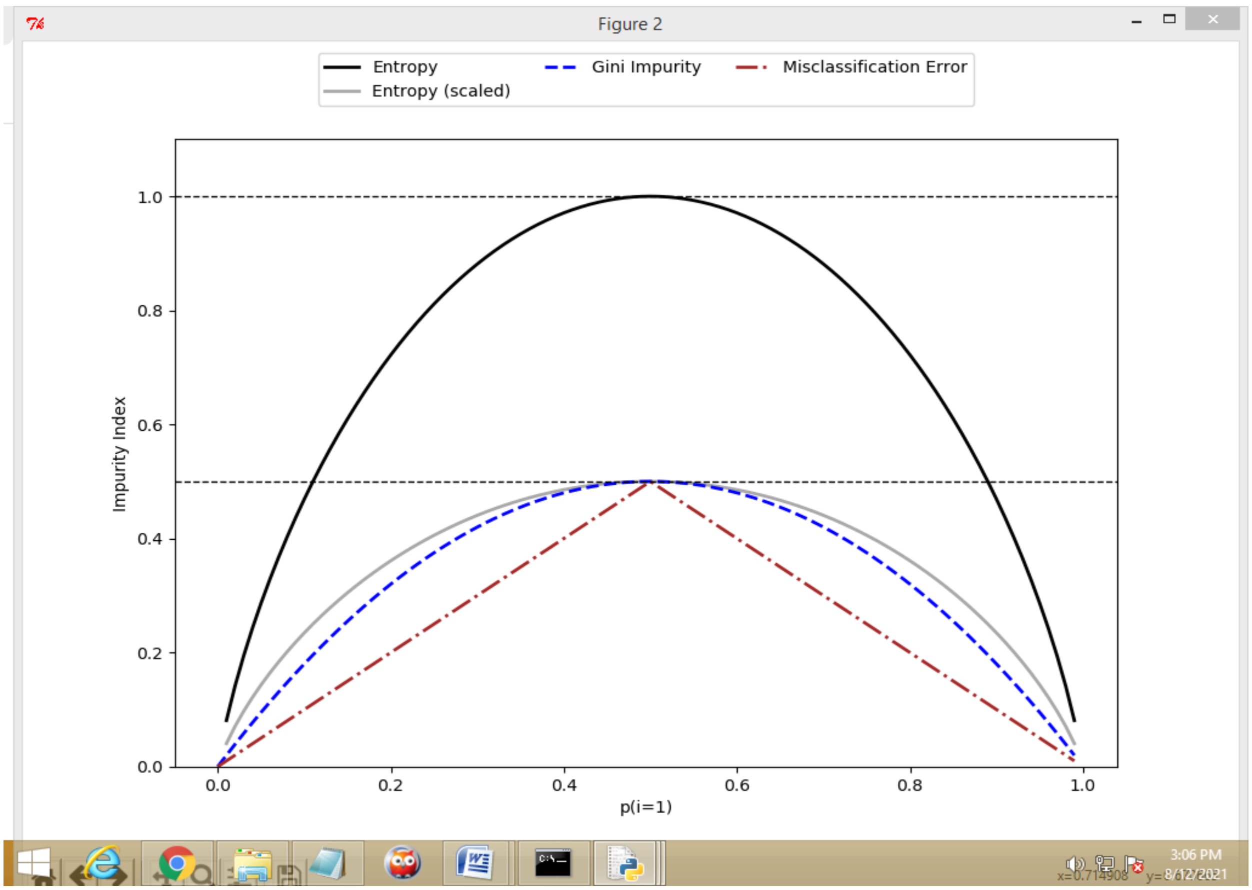
Aim :- To apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned tree

Program:DeciTree1.py

```
import  matplotlib.pyplot as plt
import  numpy as np
#-----Calculating Gini Index
def gini(p):
   return (p)*(1 - (p)) + (1 - p)*(1 - (1-p))
#-----Calculating Entropy
def entropy(p):
   return - p*np.log2(p) - (1 - p)*np.log2((1 - p))
#-----Calculating Classification Error
def classification_error(p):
   return 1 - np.max([p, 1 - p])
#----Creating a Numpy Array of probability values from 0 to 1, with an #increment
of 0.01
x = np.arange(0.0, 1.0, 0.01)
#---Obtaining Entropy for different values of p
ent = [entropy(p) if p != 0 else None for p in x]
#---Obtaining scaled entropy
sc_ent = [e*0.5 if e else None for e in ent]
#--Classification Error
err = [classification_error(i) for i in x]
#--Plotting
fig = plt.figure();
plt.figure(figsize=(10,8));
ax = plt.subplot(111);
for i, lab, ls, c, in zip([ent, sc_ent, gini(x), err], ['Entropy', 'Entropy (scaled)','Gini
Impurity',   'Misclassification Error'],['-', '-', '--', '-.'], ['black', 'darkgray','blue',
'brown', 'cyan']):
```

```
    line = ax.plot(x, i, label=lab,
    linestyle=ls, lw=2, color=c)
ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=3, fancybox=True,
shadow=False)
ax.axhline(y=0.5, linewidth=1, color='k', linestyle='--')
ax.axhline(y=1.0, linewidth=1, color='k', linestyle='--')
plt.ylim([0, 1.1])
plt.xlabel('p(i=1)')
plt.ylabel('Impurity Index')
plt.show()
```

OUTPUT:



RESULT: Thus the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data is implemented and executed successfully.

EX.NO.9.(ii)	**Demonstrate The Working Of The Decision Tree Based ID3 Algorithm. Use An Appropriate Data Set For Building The Decision Tree And Apply This Knowledge To Classify A New Sample.**
Date:06.11.23	

Aim:- To write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

3dataset.csv

outlook	**temperature**	**humidity**	**wind**	**Answer**
sunny	hot	high	weak	No
sunny	hot	high	strong	No
overcast	hot	high	weak	Yes
rain	mild	high	weak	Yes
rain	cool	normal	weak	Yes
rain	cool	normal	strong	No
overcast	cool	normal	strong	Yes
sunny	mild	high	weak	No
sunny	cool	normal	weak	Yes
rain	mild	normal	weak	Yes
sunny	mild	normal	strong	Yes
overcast	mild	high	strong	Yes
overcast	hot	normal	weak	Yes
rain	mild	high	strong	No

Program:DeciTree2.py

```
import pandas as pd
import math
import numpy as np
data = pd.read_csv('/content/drive/My Drive//3-dataset.csv')
features = [feat for feat in data]
features.remove("answer")

#Create a class named Node with four members children, value, isLeaf and
#pred.
class Node:
    def __init__(self):
```

```
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""
```

#Define a function called entropy to find the entropy oof the dataset.

```
def entropy(examples):
    pos = 0.0
    neg = 0.0
    for _, row in examples.iterrows():
        if row["answer"] == "yes":
            pos += 1
        else:
            neg += 1
    if pos == 0.0 or neg == 0.0:
        return 0.0
    else:
        p = pos / (pos + neg)
        n = neg / (pos + neg)
        return -(p * math.log(p, 2) + n * math.log(n, 2))
```

#Define a function named info_gain to find the gain of the attribute

```
def info_gain(examples, attr):
    uniq = np.unique(examples[attr])
    #print ("\n",uniq)
    gain = entropy(examples)
    #print ("\n",gain)
    for u in uniq:
        subdata = examples[examples[attr] == u]
        #print ("\n",subdata)
        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
        #print ("\n",gain)
    return gain
def ID3(examples, attrs):
```

```
    root = Node()
    max_gain = 0
    max_feat = ""
    for feature in attrs:
        #print ("\n",examples)
        gain = info_gain(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat
    #print ("\nMax feature attr",max_feat)
    uniq = np.unique(examples[max_feat])
    #print ("\n",uniq)
    for u in uniq:
        #print ("\n",u)
        subdata = examples[examples[max_feat] == u]
        #print ("\n",subdata)
        if entropy(subdata) == 0.0:
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["answer"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)
    return root
```

#Define a function named printTree to draw the decision tree

```
def printTree(root: Node, depth=0):
    for i in range(depth):
```

```
        print("\t", end="")
    print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    print()
    for child in root.children:
        printTree(child, depth + 1)
```

#Define a function named classify to classify the new example

```
def classify(root: Node, new):
    for child in root.children:
        if child.value == new[root.value]:
            if child.isLeaf:
                print ("Predicted Label for new example", new," is:", child.pred)
                exit
            else:
                classify (child.children[0], new)
```

#Finally, call the ID3, printTree and classify functions

```
root = ID3(data, features)
print("Decision Tree is:")
printTree(root)
print ("------------------")
new = {"outlook":"sunny", "temperature":"hot", "humidity":"normal",
"wind":"strong"}
classify (root, new)
```

Output:-

```
Decision Tree is:
outlook
    overcast ->  ['yes']
    rain
        wind
            strong ->  ['no']

            weak ->  ['yes']
    sunny
        humidity
            high ->  ['no']

            normal ->  ['yes']
------------------
Predicted Label for new example {'outlook': 'sunny', 'temperature': 'hot', 'humidity': 'normal', 'wind': 'strong'}  is: ['yes']
```

Result:-

Thus the program of the decision tree based ID3 algorithm is used an appropriate data set for building the decision tree and apply this knowledge to classify a new sample is executed successfully.

EX.NO.10.	Build An Artificial Neural Network By Implementing The Backpropagation Algorithm And Test The Same Using Appropriate Data Sets.
DATE:13.11.23	

Aim:

To build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

PROGRAM :-Backprop.py

```
import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)

y = np.array(([92], [86], [89]), dtype=float)

X = X/np.amax(X,axis=0) # maximum of X array longitudinally

y = y/100

#Sigmoid Function

def  sigmoid (x):

return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function

def  derivatives_sigmoid(x):

return x * (1 - x)

#Variable initialization

epoch=7000 #Setting training iterations

lr=0.1 #Setting learning rate

inputlayer_neurons = 2 #number of features in data set

hiddenlayer_neurons = 3 #number of hidden layers neurons

output_neurons = 1 #number of neurons at output layer

#weight and bias initialization
```

```
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
#draws a random range of numbers uniformly of dim x*y
for  i in range(epoch):
	#Forward Propogation
	hinp1=np.dot(X,wh)
	hinp=hinp1 + bh
	hlayer_act = sigmoid(hinp)
	outinp1=np.dot(hlayer_act,wout)
	outinp= outinp1+ bout
	output = sigmoid(outinp)
	#Backpropagation
	EO = y-output
	outgrad = derivatives_sigmoid(output)
	d_output = EO* outgrad
	EH = d_output.dot(wout.T)
	hiddengrad = derivatives_sigmoid(hlayer_act)#how much hidden layer wts
	contributed to error
	d_hiddenlayer = EH * hiddengrad
	wout += hlayer_act.T.dot(d_output) *lr# dotproduct of nextlayererror and
	currentlayerop
```

```
        # bout += np.sum(d_output, axis=0,keepdims=True) *lr
        wh += X.T.dot(d_hiddenlayer) *lr
        #bh += np.sum(d_hiddenlayer, axis=0,keepdims=True) *lr
        print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

OUTPUT

Input:

```
[[ 0.66666667 1. ]
[ 0.33333333 0.55555556]
[ 1. 0.66666667]]
Actual Output:
[[ 0.92]
[ 0.86]
[ 0.89]]
Predicted Output:
[[ 0.89559591]
[ 0.88142069]
[ 0.8928407 ]]
```

Result:- Thus to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

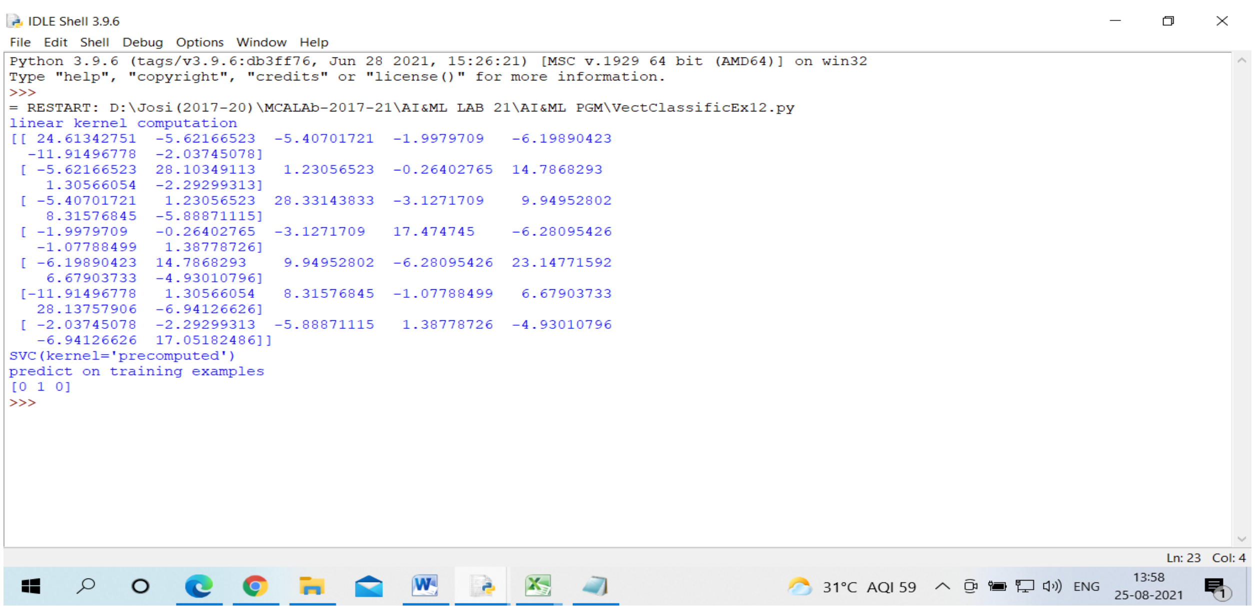
EX.NO.11	**Implement Support Vector Classification For Linear Kernel.**
DATE:20.11.23	

AIM: To implement Support Vector Classification for linear kernel.

PROGRAM:- SVCLK.py

```
import numpy as np
from sklearn.datasets import  make_classification
from sklearn.model_selection  import  train_test_split
from sklearn import  svm
X, y = make_classification(n_samples=10, random_state=0)
X_train ,X_test , y_train, y_test = train_test_split(X, y, random_state=0)
clf = svm.SVC(kernel='precomputed')
# linear kernel computation
gram_train = np.dot(X_train, X_train.T)
print("Linear kernel computation")
print(gram_train)
print(clf.fit(gram_train, y_train))
# predict on training examples
gram_test = np.dot(X_test, X_train.T)
print("Predict on Training examples")
print(clf.predict(gram_test))
```

OUTPUT:



RESULT: Thus the Support Vector Classification for linear kernel has been executed successfully.

EX.NO.12	Implement Logistic Regression To Classify The Problems Such As Spam Detection
DATE:27.11.23	

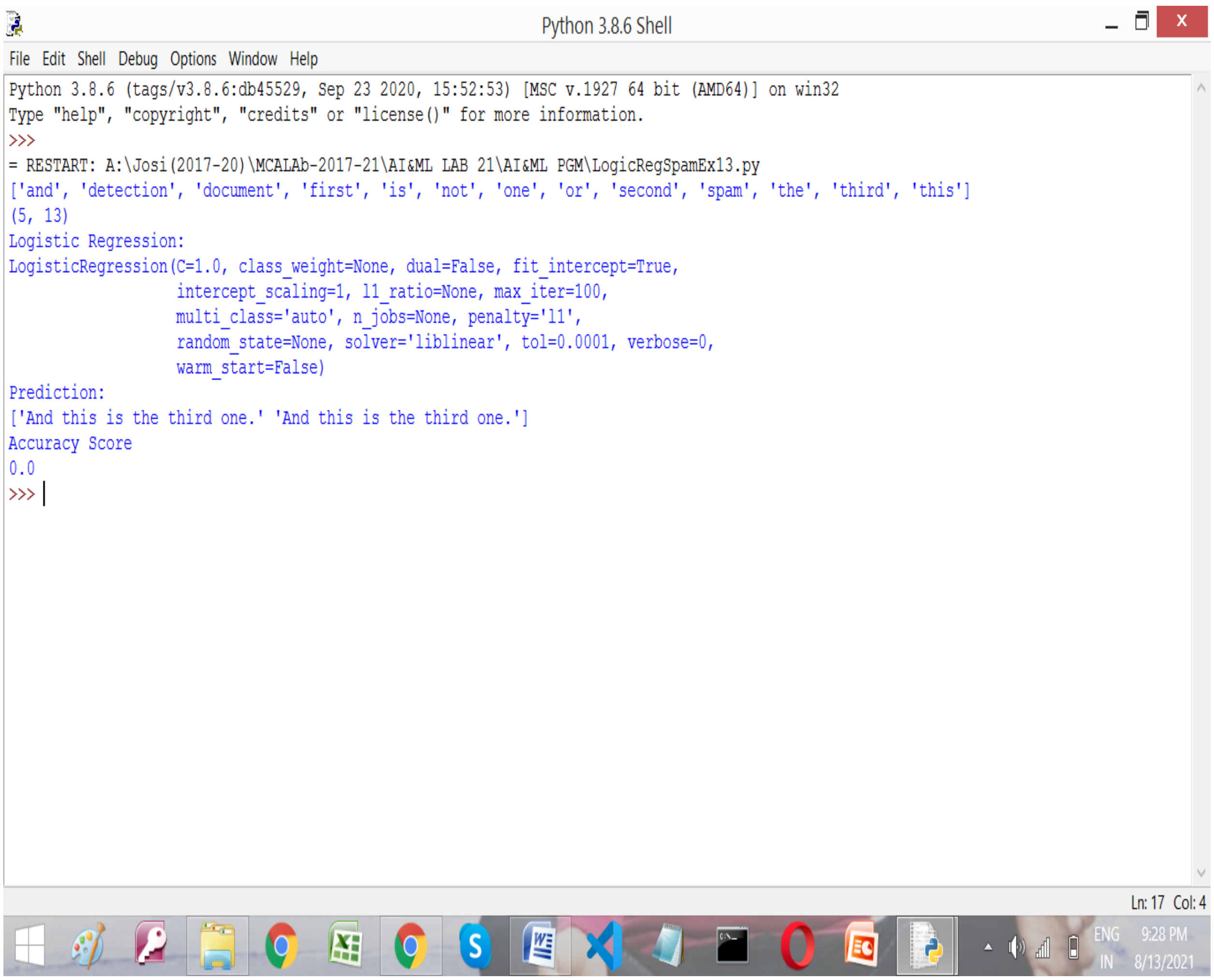
AIM:-To implement Logistic Regression to classify the problems such as spam detection.

PROGRAM: LogiReg.py

```
from sklearn.feature_extraction.text  import  TfidfVectorizer
from sklearn.model_selection  import  train_test_split
import csv
data = ['This is the first document.',
        'This document is the second document.',
        'And this is the third one.',
        'Is this the first document?',]
#pre-processing on message text, like removing-
# punctuation and stop words.
def text_preprocess(text):
        text = text.translate(str.maketrans('', '', string.punctuation))
        text = [word for word in text.split() if word.lower() not in
        stopwords.words('english')]
        return " ".join(text)
#vectorize the data
vectorizer = TfidfVectorizer()
message_mat = vectorizer.fit_transform(data)
print(message_mat)
print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
print(message_mat.shape)
#vector matrix can be used create train/test split.
message_train, message_test, spam_nospam_train,spam_nospam_test =
train_test_split(message_mat, data,test_size=0.3, random_state=20)
# Choose logistic regression model
from sklearn.linear_model  import  LogisticRegression
from sklearn.metrics import  accuracy_score
Spam_model = LogisticRegression(solver='liblinear', penalty='l1')
print("Logistic Regression:")
```

```
print(Spam_model.fit(message_train, spam_nospam_train))
print("Prediction:")
pred = Spam_model.predict(message_test)
print(pred)
print('Accuracy Score')
print(accuracy_score(spam_nospam_test,pred))
```

OUTPUT:



```
Python 3.8.6 Shell
File Edit Shell Debug Options Window Help
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: A:\Josi(2017-20)\MCALAb-2017-21\AI&ML LAB 21\AI&ML PGM\LogicRegSpamEx13.py
['and', 'detection', 'document', 'first', 'is', 'not', 'one', 'or', 'second', 'spam', 'the', 'third', 'this']
(5, 13)
Logistic Regression:
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l1',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
Prediction:
['And this is the third one.' 'And this is the third one.']
Accuracy Score
0.0
>>> |
Ln: 17  Col: 4
```

RESULT: Thus the Logistic Regression to classify the problems such as spam detection is implemented and executed successfully.