Data Science Lab - Manual

Prepared By : Vasudevan T V

List of Programs (Python and R)

1 - Get the Largest Number from a List

Write a python program to get the largest number from a list.

```
Program 1 - Version 1
---------------------
list=[10,20,30]
print("List:", list)
print("The largest element is:", max(list))

Program 1 - Version 2
---------------------
list1=[]
while True :
  print("Enter an integer to the list:")
  element = int(input())
  list1.append(element)
  print("Is it end of list?(y/n)")
  flag = input()
  if flag == "y" :
    break
print("The List =", list1)
print("Maximum Value in this list:", max(list1))
```

2 - Remove Duplicates from a List

Write a python program to remove duplicates from a list.

```
Program 2 - Version 1
---------------------
list1=[20,20,10,30,30]
print("Duplicated List:", list1)
list2=[]
for i in list1:
    if i not in list2:
        list2.append(i)
print("Deduplicated List:", list2)

Program 2 - Version 2
---------------------
list1 = [9,9,5,6,3,4,1,9,2,4,2]
```

```
print("Original List:", list1)
list2 = list(set(list1))
print("Duplicates Removed List : ", list2)
```

3 - Conversion of Tuple to Dictionary

Write a python program to convert a tuple to a dictionary.

```
Program 3
---------
tuple = (('Rajesh', 75),('Vasudevan', 80), ('Thomas', 85))
print("Tuple=",tuple)
dictionary = dict(tuple)
print("Dictionary=",dictionary)
```

4 - Merging Dictionaries

Write a python program to merge two python dictionaries.
Hint - Use update() method or ** operator or | operator

```
Program 4 - Version 1
---------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)

# Merging dict2 into dict1
dict1.update(dict2)

print(dict1)

Program 4 - Version 2
---------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)
```

```
# Merging dictionaries
merged_dictionary = {**dict1, **dict2}

print("Merged Dictionary=", merged_dictionary)

Program 4 - Version 3
---------------------
dict1 = {'a': 1, 'b': 2}
print("Dictionary 1=",dict1)
dict2 = {'c': 3, 'd': 4}
print("Dictionary 2=",dict2)

# Merging dictionaries
merged_dictionary = dict1|dict2

print("Merged Dictionary=", merged_dictionary)
```

5 - Checking Common Member in Lists

Write a python program that takes two lists and returns true if they have at least one common member.

```
Program 5
---------
list1=[10,20,30]
# list is converted to set to make the elements unique
# and to perform set operations
set1=set(list1)
list2=[40,50,60]
set2=set(list2)
result=set1 & set2
# If there is no common element, then result = set()
if (result== set()):
    print("False")
else :
    print("True")
```

6 - Find Earlier Date

Write a python program to determine which one is the earlier date from the two given dates.

```
Program 6
---------
date1="2025-01-26"
date2="2025-08-15"
if date1<date2 :
    print("Earlier date is", date1)
elif date1>date2 :
    print("Earlier date is", date2)
else:
    print("Two dates are equal")
```

7 - Subtract 5 Days from Current Date

Write a python program to subtract 5 days from current date.

```
Program 7
---------
import datetime
today=datetime.date.today()
no_of_days=datetime.timedelta(days=5)
print(today-no_of_days)
```

8 - File Copy Operation

Write a Python program to open a file and copy the contents to another file.

```
Program 8
---------
source_file=open("file1.txt","r")
contents=source_file.read()
source_file.close()
destination_file=open("file2.txt","w")
destination_file.write(contents)
destination_file.close()
```

9 - Capitalise Each Word in a File

Write a python program to capitalise each word in a file.
Hint - Use str.upper() method

```
Program 9
---------
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.upper()
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

10 - Search and Replace in a File

Write a Python program to search for a word in a file and replace it with another word.
Hint - Use str.replace("old text","new text") method

```
Program 10
----------
file=open("file1.txt","r")
contents=file.read()
new_contents=contents.replace("Python","C")
file.close()
file=open("file1.txt","w")
file.write(new_contents)
file.close()
```

11 - Count Number of Lines in a File

Write a python program to count number of lines in a file.

```
Program 11
----------
count = 0
file = open("file1.txt", "r")
for line in file:
 count = count + 1
print("Number of Lines:", count)
```

12 - Retrieve Lines Having Two Consecutive 1's

Write a python program to retrieve lines having two consecutive 1's.

```
Program 12
----------
lines_with_consecutive_ones = []
file = open("file1.txt", "r")
for line in file:
    if '11' in line:  # Check for two consecutive '1's
        lines_with_consecutive_ones.append(line.strip()) # remove leading
                                                         # and trailing
                                                         # white space
print(lines_with_consecutive_ones)
print("Lines containing two consecutive '1's:")
for line in lines_with_consecutive_ones:
    print(line)
```

13 - Matrix Operations Using Vectorisation

Write python programs to perform the following matrix operations using vectorisation. (Use matrices of order 3 X 3)

1. addition
2. subtraction
3. multiplication
4. scalar multiplication
5. transpose

```
Program 13
----------
import numpy
matrix1=numpy.matrix([[1,2,3],[1,1,1],[1,1,1]])
matrix2=numpy.matrix([[1,0,0],[0,1,0],[0,0,1]])
print("Matrix 1=\n",matrix1)
print("Matrix 2=\n",matrix2)
```

```
matrix3 = numpy.add(matrix1,matrix2)
matrix4 = numpy.subtract(matrix1,matrix2)
matrix5 = numpy.matmul(matrix1,matrix2)
matrix6 = numpy.transpose(matrix1)
print("Matrix 1 + Matrix 2=\n", matrix3)
print("Matrix 1 - Matrix 2=\n", matrix4)
print("Matrix 1 * Matrix 2=\n", matrix5)
print("2 * Matrix 1=\n", 2*matrix1)
print("Transpose of Matrix 1=\n", matrix6)
```

14 - Creation of Matrices for 2D Geometric Transformations

Write a python program to create the following matrices for various 2D geometric transformations.

1. translation matrix
2. rotation matrix
3. scaling matrix

```
Program 14 - Translation Matrix
-------------------------------
import numpy
def translationMatrix(tx=0, ty=0):
       return numpy.matrix([[1,0,tx],
                            [0,1,ty],
                            [0,0,1]])
matrix=translationMatrix(1,1)
print(matrix)
Program 14 - Rotation Matrix
----------------------------
import numpy
def rotationMatrix(degree):
       theta = numpy.radians(degree)
       c,s=numpy.cos(theta),numpy.sin(theta)
       return numpy.matrix([[c,-s,0],
                            [s, c,0],
                            [0,0,1]])
matrix=rotationMatrix(30)
print(matrix)
```

```
Program 14 - Scaling Matrix
---------------------------
import numpy
def scalingMatrix(sx=0, sy=0):
       return numpy.matrix([[sx,0,0],
                            [0,sy,0],
                            [0,0,1]])
matrix=scalingMatrix(2,2)
print(matrix)
```

15 - Creation of Matrices for 3D Geometric Transformations

Write a python program to create the following matrices for various 3D geometric transformations.

1. translation matrix
2. rotation matrix
3. scaling matrix

```
Program 15 - Translation Matrix
-------------------------------
import numpy
def translationMatrix(tx=0, ty=0, tz=0):
       return numpy.matrix([[1,0,0,tx],
                            [0,1,0,ty],
                            [0,0,1,tz],
                            [0,0,0,1 ]])
matrix=translationMatrix(1,1,1)
print(matrix)
```

```
Program 15 - Rotation Matrix about x axis
-----------------------------------------
# Defines 3d rotation matrix for rotating about x axis
import numpy
def rotationMatrix(degree):
       theta = numpy.radians(degree)
       c,s=numpy.cos(theta),numpy.sin(theta)
       return numpy.matrix([[1,0,0,0],
                            [0,c,-s,0],
                            [0,s,c,0],
```

```
                              [0,0,0,1]])
matrix=rotationMatrix(30)
print(matrix)
```

```
Program 15 - Rotation Matrix about y axis
-----------------------------------------
# Defines 3d rotation matrix for rotating about y axis
import numpy
def rotationMatrix(degree):
       theta = numpy.radians(degree)
       c,s=numpy.cos(theta),numpy.sin(theta)
       return numpy.matrix([[c,0,s,0],
                            [0,1,0,0],
                            [-s,0,c,0],
                            [0,0,0,1]])
matrix=rotationMatrix(30)
print(matrix)
```

```
Program 15 - Rotation Matrix about z axis
-----------------------------------------
# Defines 3d rotation matrix for rotating about z axis
import numpy
def rotationMatrix(degree):
       theta = numpy.radians(degree)
       c,s=numpy.cos(theta),numpy.sin(theta)
       return numpy.matrix([[c,-s,0,0],
                            [s,c,0,0],
                            [0,0,1,0],
                            [0,0,0,1]])
matrix=rotationMatrix(30)
print(matrix)
```

```
Program 15 - Scaling Matrix
---------------------------
import numpy
def scalingMatrix(sx=0, sy=0, sz=0):
       return numpy.matrix([[sx,0,0,0],
                            [0,sy,0,0],
                            [0,0,sz,0],
                            [0,0,0,1]])
matrix=scalingMatrix(2,2,2)
print(matrix)
```

16 - Singular Value Decomposition of a Matrix

Write a python program to perform SVD (Singular Value Decomposition) of a square matrix of order 3. Reconstruct the original matrix from the components.

```
Program 16
----------
# Imports matrix, matmul and diag functions only
from numpy import matrix
from numpy import matmul
from numpy import diag
# Imports svd fn from linalg(linear algebra) submodule of scipy module
from scipy.linalg import svd
# define a matrix
A = matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# Singular-value decomposition
# A is decomposed into 3 matrices U, a diagonal matrix and V
# Here S contains only the diagonal elements of the diagonal matrix
U, S, V = svd(A)
print(U)
print(S)
print(V)
# create diagonal matrix from diagonal elements
Sigma = diag(S)
print(Sigma)
# reconstruct matrix
B = matmul(U,matmul(Sigma,V))
print(B)
```

17 - Plot a Histogram

The marks obtained by students in a class are given below.

22,87,5,43,56,73,55,54,11,20,51,5,79,31,27

Write a python program to draw a histogram of these marks for intervals 0 - 10, 10 - 20,..., 90 - 100.

```
Program 17
----------
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# imports array() from numpy package
from numpy import array
# subplots() specify the number of plots in the figure
# first argument is number of rows
# second argument is number of columns
# This function returns a tuple containing figure and axes objects
# These objects are assigned to fig and ax
# They are needed for changing figure level and axes level attributes
fig,ax = pyplot.subplots(1,1)
a = array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
# Draws a histogram, first argument is the array of numbers,
# second argument bins are intervals of values
ax.hist(a, bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
ax.set_title("histogram of result")
ax.set_xticks([0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
ax.set_xlabel('marks')
ax.set_ylabel('no. of students')
# Shows the plot
pyplot.show()
```

18 - Plot a Histogram Using Iris Data Set

Write a python program to draw a histogram of sepal length in the iris data set (given).

```
Program 18
----------
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# pandas is used for data analysis
import pandas
# reads the csv file into a data frame
# A data frame is a table with rows and columns
df = pandas.read_csv('iris.csv')
# subplots() specify the number of plots in the figure
# first argument is number of rows
```

```
# second argument is number of columns
# This function returns a tuple containing figure and axes objects
# These objects are assigned to fig and ax
# They are needed for changing figure level and axes level attributes
fig,ax = pyplot.subplots(1,1)
# plots the histogram of petal length attribute
# By default bins = 10
df['petal.length'].plot(kind='hist', edgecolor="black", bins=49)
ax.set_title("Histogram")
ax.set_xticks([1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,6.5,7.0])
ax.set_yticks([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19])
ax.set_xlabel('Petal Length')
# shows the histogram
pyplot.show()
```

19 - Plot a Scatterplot

Write a python program to draw a scatterplot that shows the relationship between rollnos and marks of students (given below) in a class.
rollnos = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
marks = [22,87,5,43,56,73,55,54,11,20,51,5,79,31,27]

```
Program 19
----------
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
rollnos = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
marks = [22,87,5,43,56,73,55,54,11,20,51,5,79,31,27]
# subplots() specify the number of plots in the figure
# first argument is number of rows
# second argument is number of columns
# This function returns a tuple containing figure and axes objects
# These objects are assigned to fig and ax
# They are needed for changing figure level and axes level attributes
fig,ax = pyplot.subplots(1,1)
# Draws a scatterplot, first argument is x axis values,
# second argument is y axis values
ax.scatter(rollnos, marks)
ax.set_title("Scatter Plot")
ax.set_xticks([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])
```

```
ax.set_yticks([25,50,75,100])
ax.set_xlabel('rollnos')
ax.set_ylabel('marks')
# Shows the plot
pyplot.show()
```

20 - Plot a Scatter plot Using Iris Data Set

Write a python program to draw a scatter plot that shows the relationship between sepal length and sepal width in the iris data set (given).

```
Program 20
----------
# imports pyplot, a module used in the package matplotlib to plot various
# figures
from matplotlib import pyplot
# pandas is used for data analysis
import pandas
# reads the csv file into a data frame
df = pandas.read_csv('iris.csv')
# subplots() specify the number of plots in the figure
# first argument is number of rows
# second argument is number of columns
# This function returns a tuple containing figure and axes objects
# These objects are assigned to fig and ax
# They are needed for changing figure level and axes level attributes
fig, ax = pyplot.subplots(1,1)
# Creates a dictionary of colour values of each species
colors = {'Setosa':'red', 'Versicolor':'green', 'Virginica':'blue'}
# Groups the data based on species values
grouped = df.groupby('species')
# group represents the grouped data frame
# draws the scatter plot for each group
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='petal.length', y='petal.width',
               label=key, color=colors[key])
ax.set_title("Scatter Plot")
ax.set_xlabel('Petal Length')
ax.set_ylabel('Petal Width')
pyplot.show()
```

21 - Predict the Food Type Using k-NN algorithm

Given a data set of 15 food items having 4 features - ingredient, sweetness, crunchiness and food type. Write a R program to predict the food type of tomato using k-NN algorithm.
data set - food.csv

```
Program 21
----------
# Read the csv file into a data frame
food=read.csv("food.csv")
# Creates a data frame of food item tomato
tomato=data.frame(ingredient="tomato",sweetness=6,crunchiness=4)
# Create a data frame of second and third columns of food
food1=food[,2:3]
# Create a data frame of second and third columns of tomato
tomato1=tomato[,2:3]
# Inputs the value of k from user while using RGui
k <- as.numeric(readline(prompt = "Enter the value of k:"))
# Inputs the value of k from user while using Rscript
cat("Enter the value of k:")
k=scan("stdin", integer(), n=1)
library(class)
pred=knn(food1,tomato1,food$FoodType,k)
cat("Predicted Food Type of Tomato:\n")
print(pred)
```

22 - Classify a Student Using k-NN algorithm

As per a survey conducted in an institution, students are classified based on the two attributes of academic excellence(X) and other activities(Y). Write a R program to identify the classification of a student with X = 5 and Y = 7 using k-NN algorithm(choose k as 3).
data set - survey.csv

```
Program 22
----------
```

```
# Loads class packge containing knn()
library(class)
# Read the csv file into a data frame
survey=read.csv("survey.csv")
# Creates a data frame of a student, X(Academic Excellence),
# Y(Other Activities)
student=data.frame(X=5,Y=7)
# Create a data frame of first and second columns of survey
survey1=survey[,1:2]
pred=knn(survey1,student,survey$Z,k=3)
cat("Predicted Classification of Student:\n")
print(pred)
```

23 - Prediction of Cancer Using k-NN algorithm

Given a data set containing 569 examples of breast cancer biopsies, each having 32 features. Write a R program to predict the result of cancer biopsies using k-NN algorithm and analyse the same.
data set - wisc_bc_data.csv

```
Program 23
----------
# Loads class packge containing knn()
library(class)
# Loads gmodels packge containing CrossTable()
library(gmodels)
# Read the csv file into a data frame
wbcd = read.csv("wisc_bc_data.csv")
# Define normalize fn for performing min max normalisation
# This will transform the values of all features to a range between 0 and 1
normalize <- function(x)
{
 return ((x - min(x)) / (max(x) - min(x)))
}
# Apply this function to our data frame
wbcd_n = as.data.frame(lapply(wbcd[3:31], normalize))
# Training Data
wbcd_train = wbcd_n[1:469, ]
# Test data
wbcd_test = wbcd_n[470:569, ]
# Training Labels
```

```
wbcd_train_labels = wbcd[1:469, 2]
# Test Labels
wbcd_test_labels = wbcd[470:569, 2]
# Prediction of Breast Cancer using knn()
wbcd_test_pred = knn(wbcd_train, wbcd_test, wbcd_train_labels,21)
# Print the prediction results
print(wbcd_test_pred)
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq=FALSE)
```

24 - Prediction of Flu Using Naive Bayes Algorithm

Given the following data on a certain set of patients seen by a doctor. Can the doctor conclude that a person having chills, fever, mild headache and without running nose has flu? Write a R program that uses Naive Bayes classification.
data set - symptoms.csv

```
Program 24
----------
# Loads e1071 package containing naiveBayes
library(e1071)

# Read the csv file into a data frame
symptoms = read.csv("symptoms.csv")

#Training Data
symptoms_train = symptoms[,1:4]
#Test Data
symptoms_test = data.frame(Chills="Y",RunningNose="N",Headache="mild",
Fever="Y")

#Naive Bayes Classification
classifier_cl <- naiveBayes(symptoms_train,symptoms$HasFlu)
# This will print classical and conditional probabilities
print(classifier_cl)

# Predicting on test data
symptoms_test_pred <- predict(classifier_cl, symptoms_test)
cat("Prediction of Flu:\n")
print(symptoms_test_pred)
```

25 - Prediction of Stealing Using Naive Bayes Algorithm

Write a R program using Naive Bayes classification to determine whether a red domestic SUV car is a stolen car or not. Use the following data set.
data set - cars.csv

```
Program 25
----------
# Loads e1071 package containing naiveBayes
library(e1071)

# Read the csv file into a data frame
cars = read.csv("cars.csv")
#Training Data
cars_train = cars[,2:4]
cars_train_labels = cars[,5]
#Test Data
cars_test = data.frame(Colour="red",Type="SUV",Origin="domestic")

#Naive Bayes Classification
classifier_cl <- naiveBayes(cars_train,cars_train_labels)
# This will print classical and conditional probabilities
print(classifier_cl)

# Predicting on test data
cars_test_pred <- predict(classifier_cl, cars_test)
cat("Prediction of Stolen Status:\n")
print(cars_test_pred)
```

26 - Prediction of Iris Species Using Naive Bayes Algorithm

Write a R program to predict the species of the iris data set(given) using Naive Bayes Classification.

```
Program 26
```

```
----------
# Loads e1071 package containing naiveBayes
library(e1071)
# Loads caTools package containing sample.split()
library(caTools)
# Loads gmodels packge containing CrossTable()
library(gmodels)

# Read the csv file into a data frame
iris = read.csv("iris.csv")

# Splitting data into train
# and test data
# set.seed() is used for generating the same sample in every execution
# We specify a seed number
set.seed(100)

split <- sample.split(iris$species, SplitRatio = 0.7)
iris1 <- subset(iris, split == "TRUE")
iris2 <- subset(iris, split == "FALSE")

iris_train = iris1[,1:4]
iris_test = iris2[,1:4]

iris_train_labels = iris1[,5]
iris_test_labels = iris2[,5]

classifier_cl <- naiveBayes(iris_train,iris_train_labels )
print(classifier_cl)

# Predicting on test data'
iris_test_pred <- predict(classifier_cl, iris_test)
print(iris_test_pred)

# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(iris_test_labels, iris_test_pred, prop.chisq=FALSE)
```

27 - Prediction of Cheating Using C5.0 Decision Tree Algorithm

Write a R program to determine whether a person cheats using C5.0 Decision Tree Algo-

rithm and evaluate its performance. Given a data set of 10 items, each having 5 features.
data set - person.csv

```
Program 27
----------
# Loads C50 package containg C5.0()
library(C50)
# Loads gmodels packge containing CrossTable()
library(gmodels)
# Read the csv file into a data frame
person <- read.csv("person.csv")
# Training Data, 5th column Cheats? is omitted
person_train <- person[1:8,-5 ]
# Test Data, 5th column Cheats? is omitted
person_test <- person[9:10,-5 ]
# Training Labels, containing values of 5th column Cheats?
person_train_labels = person[1:8, 5]
# Test Labels, containing values of 5th column Cheats?
person_test_labels = person[9:10, 5]
# C5.0() returns a C5.0 model object and stores it in
# person_model
# person_train is a data frame containing training data
# person_train_labels is converted into a factor containing
# categorical values
person_model<-C5.0(person_train,as.factor(person_train_labels))
# Prints basic data about the decision tree
print(person_model)
# Shows the decision tree and some other information
print(summary(person_model))
# Predicting on test data
person_pred <- predict(person_model, person_test)
# Prints predicted values
print(person_pred)
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(person_test_labels, person_pred, prop.chisq=FALSE )
```

28 - Prediction of Cricket Play Using C5.0 Decision Tree Algorithm

Write a R program to determine whether play of cricket is possible depending on the weather condition using C5.0 Decision Tree Algorithm and evaluate its performance. Given a data set of 14 items, each having 5 features.
data set - cricket.csv

```
Program 28
----------
# Loads C50 package containg C5.0()
library(C50)
# Loads gmodels packge containing CrossTable()
library(gmodels)
# Read the csv file into a data frame
cricket <- read.csv("cricket.csv")
# Training Data, 5th column Play Cricket is omitted
cricket_train <- cricket[1:9,-5 ]
# Test Data, 5th column Play Cricket is omitted
cricket_test <- cricket[10:14,-5 ]
# Training Labels, containing values of 5th column Play Cricket
cricket_train_labels = cricket[1:9, 5]
# Test Labels, containing values of 5th column Play Cricket
cricket_test_labels = cricket[10:14, 5]
# C5.0() returns a C5.0 model object and stores it in
# cricket_model
# cricket_train is a data frame containing training data
# cricket_train_labels is converted into a factor containing
# categorical values
cricket_model <- C5.0(cricket_train, as.factor(cricket_train_labels))
# Prints basic data about the decision tree
print(cricket_model)
# Shows the decision tree and some other information
print(summary(cricket_model))
# Predicting on test data
cricket_pred <- predict(cricket_model, cricket_test)
# Print prediction
print(cricket_pred)
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(cricket_test_labels, cricket_pred, prop.chisq=FALSE )
```

29 - Identification of Risky Bank Loans Using C5.0 Decision Tree Algorithm

Write a R program to identify risky bank loans using C5.0 Decision Tree Algorithm and evaluate its performance. Given a data set of 1000 customers, each having 17 features. *data set - credit.csv*

```
Program 29
----------
# Loads C50 package containg C5.0()
library(C50)
# Loads gmodels packge containing CrossTable()
library(gmodels)
# Read the csv file into a data frame
credit <- read.csv("credit.csv")
# Training Data, 17th column default is omitted
credit_train <- credit[1:900,-17 ]
#Test Data, 17th column default is omitted
credit_test <- credit[901:1000,-17 ]
# Training Labels, containing values of 17th column default
credit_train_labels = credit[1:900, 17]
# Test Labels, containing values of 17th column default
credit_test_labels = credit[901:1000, 17]
# C5.0() returns a C5.0 model object and stores it in credit_model
# credit_train is a data frame containing training data
# credit_train_labels is converted into a factor containing
# categorical values
credit_model <- C5.0(credit_train, as.factor(credit_train_labels))
# Prints basic data about the decision tree
print(credit_model)
# Shows the decision tree and some other information
print(summary(credit_model))
# Predicting on test data
credit_pred <- predict(credit_model, credit_test)
# Print prediction
print(credit_pred)
# Analysis of Prediction
# prop.chisq=FALSE will remove unnecessary chi square values
CrossTable(credit_test_labels, credit_pred, prop.chisq=FALSE )
```

30 - Prediction of Stock Index Price Using Multiple Linear Regression Technique

Write a R program to predict the Stock Index Price (the dependent variable) of a fictitious economy based on two independent variables Interest Rate and Unemployment Rate using

Multiple Linear Regression Technique and evaluate its performance. Given a data set of 24 items.
data set - economy.csv

```
Program 30
----------
# Read the csv file into a data frame
economy <- read.csv("economy.csv")
# Training Data
economy_train <- economy[1:18,]
# Test Data
economy_test <- economy[19:24,]
# lm() returns a multiple linear regression model object
# the dependent variable stock_index_price goes to the left of the tilde
# the independent variables go to the right, separated by + sign
# data specifies the data frame in which these variables can be found
# lm() is contained is stats package, which is loaded by default
economy_model <- lm(Stock_Index_Price ~ Interest_Rate + Unemployment_Rate,
data = economy_train)
# Prints estimated regression coefficients
print(economy_model)
# Evaluate Model Performance
print(summary(economy_model))
# Predicting on test data
economy_pred <- predict(economy_model, economy_test)
# Print prediction
print(economy_pred)
```

31 - Prediction of Heart Disease Using Multiple Linear Regression Technique

Write a R program to predict the percentage of heart disease (the dependent variable) based on two independent variables, percentage of people biking to town and percentage of people smoking using Multiple Linear Regression Technique and evaluate its performance. Given a data set of 498 items.
data set - heart.csv

```
Program 31
----------
# Read the csv file into a data frame
```

```
heart <- read.csv("heart.csv")
# Training Data
heart_train <- heart[1:400,]
# Test Data
heart_test <- heart[401:498,]
# lm() returns a multiple linear regression model object
# the dependent variable heart.disease goes to the left of the tilde
# the independent variables go to the right, separated by + sign
# data specifies the data frame in which these variables can be found
# lm() is contained is stats package, which is loaded by default
heart_model <- lm(heart.disease ~  biking + smoking, data = heart_train)
# Prints estimated regression coefficients
print(heart_model)
# Evaluate Model Performance
print(summary(heart_model))
# Predicting on test data
heart_pred <- predict(heart_model, heart_test)
# Print prediction
print(heart_pred)
```

32 - Prediction of Medical Expenses Using Multiple Linear Regression Technique

Write a R program to predict medical expenses (the dependent variable) based on the independent variables, age, sex, bmi, children, smoker and region using Multiple Linear Regression Technique and evaluate its performance. Given a data set of 1338 items.
data set - insurance.csv

```
Program 32
----------
# Our model's dependent variable is expenses , which measures the medical
# costs each person charged to the insurance plan for the year
# Read the csv file into a data frame
insurance <- read.csv("insurance.csv")
# Training Data
insurance_train <- insurance[1:1000,]
# Test Data
insurance_test <- insurance[1001:1338,]
# lm() returns a multiple linear regression model object
# the dependent variable expenses goes to the left of the tilde
# the independent variables go to the right, separated by + sign
# data specifies the data frame in which these variables can be found
# lm() is contained in stats package, which is loaded by default
```

```
insurance_model <- lm(expenses ~ age + sex + bmi + children + smoker +
region, data = insurance_train)
# Prints estimated regression coefficients
print(insurance_model)
# Evaluate Model Performance
print(summary(insurance_model))
# Predicting on test data
insurance_pred <- predict(insurance_model, insurance_test)
# Print prediction
print(insurance_pred)
```

33 - Partition of Iris Data Set Using k-means Clustering Algorithm

Write a R program to partition the iris data set(given) into different clusters using k-means clustering algorithm. Choose k as 3.Check clustering result against species class label.

```
Program 33
----------
# set.seed() is used for generating the same sample in every execution
# We specify a seed number
set.seed(100)

# Reads the iris data set into the iris data frame
iris <- read.csv("iris.csv")

# Make a copy of iris data
iris2 <- iris

# Remove the species class label
iris2$species <- NULL

# Clustering with kmeans is performed by kmeans()
# kmeans() is contained within stats package which is loaded by default
# The kmeans() function requires a data frame containing only numeric data
# and a parameter specifying the desired number of clusters
# This function will return a cluster object that stores cluster
# information
# The cluster information includes cluster sizes, cluster means,
# vector of cluster assignments etc.
iris_clusters <- kmeans(iris2, 3)
print(iris_clusters)
```

```
# Check clustering result against species class label
# iris_clusters$cluster is a vector of cluster assignments from the
# kmeans()
# table() performs a tabulation of categorical variable and gives its
# frequency as output
print(table(iris$species, iris_clusters$cluster))
```

34 - Partition of Breast Cancer Data Set Using k-means Clustering Algorithm

Given a data set containing 569 examples of breast cancer biopsies, each having 32 features.Write a R program to partition this data set into different clusters using k-means clustering algorithm. Choose k as 2. Check clustering result against diagnosis class label.
data set - wisc_bc_data.csv

```
Program 34
----------
# set.seed() is used for generating the same sample in every execution
# We specify a seed number
set.seed(100)

# Reads the data set into a data frame
wisc_bc_data <- read.csv("wisc_bc_data.csv")

# Make a copy of data
wisc_bc_data2 <- wisc_bc_data

# Remove the id, diagnosis class labels
wisc_bc_data2$id <- NULL
wisc_bc_data2$diagnosis <- NULL

# Clustering with kmeans is performed by kmeans()
# kmeans() is contained within stats package which is loaded by default
# The kmeans() function requires a data frame containing only numeric data
# and a parameter specifying the desired number of clusters
# This function will return a cluster object that stores cluster
# information
# The cluster information includes cluster sizes, cluster means, vector of
# cluster assignments etc.
wisc_bc_data_clusters <- kmeans(wisc_bc_data2, 2)
print(wisc_bc_data_clusters)
```

```
# Check clustering result against diagnosis class label
# wisc_bc_data_clusters$cluster is a vector of cluster assignments from the
# kmeans()
# table() performs a tabulation of categorical variable and gives its
# frequency as output
print(table(wisc_bc_data$diagnosis, wisc_bc_data_clusters$cluster))
```