DEPARTMENT OF COMPUTER APPLICATIONS

MCA 251 – DATA MINING AND BIG DATA LAB

II/II MCA - (I SEMESTER)

LAB MANUAL

R.V.R. & J.C.COLLEGE OF ENGINEERING

(AUTONOMOUS)

(ACCREDITED BY NAAC WITH 'A' GRADE)

(Approved by A.I.C.T.E.)

(Affiliated to Acharya Nagarjuna University)

Chandramoulipuram :: Chowdavaram

GUNTUR – 522019 :: ANDHRA PRADESH

PROFORMA FOR LAB BASED COURSE DESCRIPTION

Course Code	: CA251
Course Title	: DATA MINING AND BIG DATA LAB
Year & Semester	: II/II MCA (I semester)
Periods/Week	: 06 – LAB
Regulation	: R20
Nature of the Course	: Core
Name of the Instructors	: Dr. M.Sridhar
Designation	: Associate Professor
E – Mail	: mandapati12@gmail.com

COURSE CONTENT

CA251: DATA MINING AND BIG DATA LAB

Lectures	: ---	Sessional Marks	: 40
Practical's	: 6 Periods/Week	Sem. End Exam Marks	: 60
Sem. End Exam Duration	: 3 hours	Credits	: 02

Prerequisite:

Course Objectives:

At the end of the course the students will understand

- To learn the algorithms used for various types of Data Mining Problems.
- To discover interesting patterns, analyze supervised and unsupervised models and estimate the accuracy of the algorithms.
- Exercise the data mining techniques with varied input values for different parameters.
- To understand setting up of Hadoop Cluster
- To solve problems using Map Reduce Technique
- To solve Big Data problem

Course Outcomes:

At the end of the course the students will able to

- Ability to understand the various kinds of tools.
- Demonstrate the classification, clustering and etc. in large data sets.
- Ability to add mining algorithms as a component to the exiting tools.
- Ability to apply mining techniques for realistic data.
- Set up multi-node Hadoop Clusters
- Apply Map Reduce algorithms for various algorithms
- Design a new algorithm that uses Map Reduce to apply on Unstructured and structured data.

LAB CYCLE 1(Data Mining)

1. Create the following NumPy arrays:
   a) A 1-D array called *zeros* having 10 elements and all the elements are set to zero.
   b) A 1-D array called *vowels* having the elements 'a', 'e', 'i', 'o' and 'u'.
   c) A 2-D array called ones having 2 rows and 5 columns and all the elements are set to 1 and *dtype* as int.
   d) Use nested Python lists to create a 2-D array called *myarray1* having 3 rows and 3 columns and store the following data:
      i. 2.7, -2, -19
      ii. 0, 3.4, 99.9
      iii. 10.6, 0, 13
   e) A 2-D array called *myarray2* using *arange()* having 3 rows and 5 columns with start value = 4, step size 4 and *dtype* as float.

Using the arrays created in the above, write NumPy commands for the following:

a) Find the dimensions, shape, size, data type of the items and itemsize of arrays *zeros, vowels, ones, myarray1 and myarray2*.
b) Reshape the array ones to have all the 10 elements in a single row.
c) Display the 2nd and 3rd element of the array *vowels*.
d) Display all elements in the 2nd and 3rd row of the array *myarray1.*
e) Display the elements in the 1st and 2nd column of the array *myarray1*.
f) Display the elements in the 1st column of the 2nd and 3rd row of the array *myarray1*.
g) Reverse the array of *vowels.*
h) Divide all elements of array ones by 3.
i) Add the arrays *myarray1* and *myarray2*.
j) Subtract *myarray1* from *myarray2* and store the result in a new array.
k) Multiply *myarray1* and *myarray2* element wise.
l) Do the matrix multiplication of *myarray1* and *myarray2* and store the result in a new array *myarray3*.
m) Divide *myarray1* by *myarray2*.
n) Find the cube of all elements of *myarray1* and divide the resulting array by 2.
o) Find the square root of all elements of myarray2 and divide the resulting array by 2. The result should be rounded to two places of decimals.
p) Find the transpose of ones and myarray2.
q) Sort the array vowels in reverse.
r) Sort the array myarray1 such that it brings the lowest value of the column in the first row and so on.
s) Use NumPy. split() to split the array myarray2 into 5 arrays column wise. Store your resulting arrays in *myarray2A, myarray2B, myarray2C, myarray2D* and *myarray2E*. Print the arrays *myarray2A, myarray2B, myarray2C, myarray2D* and *myarray2E*.
t) Split the array *zeros* at array index 2, 5, 7, 8 and store the resulting arrays in *zerosA, zerosB, zerosC* and *zerosD* and print them.
u) Concatenate the arrays *myarray2A*, *myarray2B* and *myarray2C* into an array having 3 rows and 3 columns.

v) Create a 2-D array called *myarray4* using *arange()* having 14 rows and 3 columns with start value = -1, step size 0.25 having. Split this array row wise into 3 equal parts and print the result.

w) Using the *myarray4* created in the above questions, write commands for the following:

i. Find the sum of all elements.
ii. Find the sum of all elements row wise.
iii. Find the sum of all elements column wise.
iv. Find the max of all elements.
v. Find the min of all elements in each row.
vi. Find the mean of all elements in each row.
vii. Find the standard deviation column wise.

2. Write a Python program to do the following operations(Using NumPy Library)
   a) Create multi-dimensional arrays and find its shape and dimension
   b) Create a matrix full of zeros and ones
   c) Reshape and flatten data in the array
   d) Append data vertically and horizontally
   e) Apply indexing and slicing on array
   f) Use statistical functions on array - Min, Max, Mean, Median and Standard Deviation.
3. Write a Python program to do the following operations(Using NumPy Library)
   a) Dot and matrix product of two arrays
   b) Compute the Eigen values of a matrix
   c) Solve a linear matrix equation such as $3 * x^0 + x^1 = 9$ and $x^0 + 2 * x^1 = 8$.
   d) Compute the multiplicative inverse of a matrix
   e) Compute the rank of a matrix
   f) Compute the determinant of an array.
4. Suppose that the data for analysis includes the attribute *age*. The *age* values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

   Write a Python program for the following using the above data:

   a) Calculate measures of central tendency.
   b) To find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data.
   c) Give the five-number summary of the data.
   d) Show a boxplot of the data.
5. Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

   a) Calculate the mean, median, and standard deviation of age and %fat.
   b) Draw the boxplots for age and %fat.
   c) Draw a scatter plot and a q-q plot based on these two variables.
6. Write a Python program to perform the following using the three fundamental Pandas data structures: the Series, DataFrame, and Index.

a) Series as generalized NumPy array
b) Series as specialized dictionary
c) Constructing Series objects
d) DataFrame as a generalized NumPy array
e) DataFrame as specialized dictionary
f) Constructing DataFrame objects:
    i. From a single Series object.
    ii. From a list of dicts.
    iii. From a dictionary of Series objects.
    iv. From a two-dimensional NumPy array.
    v. From a NumPy structured array.
g) Index as immutable array.
h) Index as ordered set.
i) Data Selection in Series:
    i. Series as dictionary
    ii. Series as one-dimensional array
    iii. Indexers: loc, iloc, and ix
j) Data Selection in DataFrame
    i. DataFrame as a dictionary
    ii. DataFrame as two-dimensional array

7. Write a Python program to perform the following:
    a) Input as CSV File
    b) Reading a CSV File
    c) Reading Specific Rows
    d) Reading Specific Columns
    e) Reading Specific Columns and Rows
    f) Reading Specific Columns for a Range of Rows
    g) Identify the missing data
    h) Identify the outlier data
    i) Replace with mean or mode
    j) Remove Blank Rows
    k) Data Categories
    l) Data types
    m) Analyze the data
    n) Visualize the data
    o) Find correlation among all attributes
8. Write a python program to perform transformation of data using Discretization (Binning) and Normalization (MinMaxScaler or MaxAbsScaler) on given dataset.
9. Write a program to implement three frequent itemset mining algorithms:
    a) Apriori
    b) FP-growth
10. Write a program to implement Decision tree algorithm.
11. Write a program to implement Naïve Bayesian Classification.
12. Write a program to implement k-means clustering algorithm.

LAB CYCLE 2 (Hadoop)

1. Study and configure hadoop for big data. Use web based tools to monitor your Hadoop setup.
2. Implement the following file management tasks in Hadoop:

- Adding files and directories
- Retrieving files
- Deleting files

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

3. Run a basic Word Count Map Reduce program to understand MapReduce Paradigm.
4. Implement Matrix Multiplication with Hadoop MapReduce

Web References

1. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_data_preprocessing_analysis_visualization.htm
2. https://cloudxlab.com/blog/numpy-pandas-introduction/
3. nptel.ac.in/courses/106106093/35
4. https://www.cse.iitb.ac.in/infolab/Data/Talks/krithi-talk-impact.pp
5. https://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop-hcatalog-hive-and-pig/https://developer.ibm.com/hadoop/docs/getting-started/tutorials/overview-tutorial/overview-lab-1-getting-started-hadoop-biginsights-2/

Lab Cycle : I
Exp.No : 1

Aim:
Create the following NumPy arrays:

a) A 1-D array called *zeros* having 10 elements and all the elements are set to zero.
b) A 1-D array called *vowels* having the elements 'a', 'e', 'i', 'o' and 'u'.
c) A 2-D array called ones having 2 rows and 5 columns and all the elements are set to 1 and *dtype* as int.
d) Use nested Python lists to create a 2-D array called *myarray1* having 3 rows and 3 columns and store the following data:
    i. 2.7, -2, -19
    ii. 0, 3.4, 99.9
    iii. 10.6, 0, 13
e) A 2-D array called *myarray2* using *arange()* having 3 rows and 5 columns with start value = 4, step size 4 and *dtype* as float.

Using the arrays created in the above, write NumPy commands for the following:

a) Find the dimensions, shape, size, data type of the items and itemsize of arrays *zeros, vowels, ones, myarray1 and myarray2*.
b) Reshape the array ones to have all the 10 elements in a single row.
c) Display the 2nd and 3rd element of the array *vowels*.
d) Display all elements in the 2nd and 3rd row of the array *myarray1.*
e) Display the elements in the 1st and 2nd column of the array *myarray1*.
f) Display the elements in the 1st column of the 2nd and 3rd row of the array *myarray1*.
g) Reverse the array of *vowels.*
h) Divide all elements of array ones by 3.
i) Add the arrays *myarray1* and *myarray2*.
j) Subtract *myarray1* from *myarray2* and store the result in a new array.
k) Multiply *myarray1* and *myarray2* element wise.
l) Do the matrix multiplication of *myarray1* and *myarray2* and store the result in a new array *myarray3*.
m) Divide *myarray1* by *myarray2*.
n) Find the cube of all elements of *myarray1* and divide the resulting array by 2.
o) Find the square root of all elements of myarray2 and divide the resulting array by 2. The result should be rounded to two places of decimals.
p) Find the transpose of ones and myarray2.
q) Sort the array vowels in reverse.
r) Sort the array myarray1 such that it brings the lowest value of the column in the first row and so on.
s) Use NumPy. split() to split the array myarray2 into 5 arrays column wise. Store your resulting arrays in *myarray2A, myarray2B, myarray2C, myarray2D* and *myarray2E*. Print the arrays *myarray2A, myarray2B, myarray2C, myarray2D* and *myarray2E*.
t) Split the array *zeros* at array index 2, 5, 7, 8 and store the resulting arrays in *zerosA, zerosB, zerosC* and *zerosD* and print them.

u) Concatenate the arrays *myarray2A*, *myarray2B* and *myarray2C* into an array having 3 rows and 3 columns.

v) Create a 2-D array called *myarray4* using *arange()* having 14 rows and 3 columns with start value = -1, step size 0.25 having. Split this array row wise into 3 equal parts and print the result.

w) Using the *myarray4* created in the above questions, write commands for the following:

i. Find the sum of all elements.
ii. Find the sum of all elements row wise.
iii. Find the sum of all elements column wise.
iv. Find the max of all elements.
v. Find the min of all elements in each row.
vi. Find the mean of all elements in each row.
vii. Find the standard deviation column wise.

Ans:

Create the following NumPy arrays:

```
import numpy as np

# a) A 1-D array called zeros having 10 elements and all the elements are
set to zero.
zeros = np.zeros(10,dtype=int)
zeros

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

# b) A 1-D array called vowels having the elements 'a', 'e', 'i', 'o' and
'u'. # # b. A 1-D array called vowels having the elements 'a', 'e', 'i', 'o'
and 'u'.
vowels = np.array(['a','e','i','o','u'])
vowels

array(['a', 'e', 'i', 'o', 'u'], dtype='<U1')

# c) A 2-D array called ones having 2 rows and 5 columns and all the
elements are set to 1 and  dtype as int.
ones = np.ones((2,5),dtype=int)
ones

array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])

# d) Use nested Python lists to create a 2-D array called myarray1 having 3
rows and 3 columns and store the following data: i. 2.7, -2, -19, ii. 0,
3.4, 99.9, iii. 10.6, 0, 13
myarray1 = np.array([[2.7, -2, -19],[0, 3.4, 99.9],[10.6, 0, 13]])
myarray1

array([[  2.7,  -2. , -19. ],
       [  0. ,   3.4,  99.9],
       [ 10.6,   0. ,  13. ]])
```

```
# e) A 2-D array called myarray2 using arange() having 3 rows and 5 columns
with start value = 4,step size 4 and dtype as float.
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray2
```

```
array([[ 4.,  8., 12.],
       [16., 20., 24.],
       [28., 32., 36.]])
```

Using the arrays created in the above, write NumPy commands for the following:

```
# a) Find the dimensions, shape, size, data type of the items and itemsize
of arrays zeros, vowels,ones, myarray1 and myarray2.
zeros.ndim
```

```
1
```

```
zeros.shape
```

```
(10,)
```

```
zeros.dtype
```

```
dtype('int32')
```

```
zeros.itemsize
```

```
4
```

```
vowels.ndim
```

```
1
```

```
vowels.shape
```

```
(5,)
```

```
vowels.size
```

```
5
```

```
vowels.dtype
```

```
dtype('<U1')
```

```
vowels.itemsize
```

```
4
```

```
ones.ndim
```

```
2
```

```
ones.shape
```

```
(2, 5)
ones.size
10
ones.dtype
dtype('int32')
ones.itemsize
4
myarray1.ndim
2
myarray1.shape
(3, 3)
myarray1.size
9
myarray1.dtype
dtype('float64')
myarray1.itemsize
8
myarray2.ndim
2
myarray2.shape
(3, 3)
myarray2.size
9
myarray2.dtype
dtype('float64')
myarray2.itemsize
8
```

```
# b) Reshape the array ones to have all the 10 elements in a single row.
ones = np.ones((2,5),dtype=int)
ones.reshape(1,10)

array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])

# c) Display the 2nd and 3rd element of the array vowels.
print(vowels[1],vowels[2])
e i

# d) Display all elements in the 2nd and 3rd row of the array myarray1.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray1[1:3]

array([[ 0. ,  3.4, 99.9],
       [10.6,  0. , 13. ]])

# e) Display the elements in the 1st and 2nd column of the array myarray1.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray1[0:3,0:2]

array([[ 2.7, -2. ],
       [ 0. ,  3.4],
       [10.6,  0. ]])

# f) Display the elements in the 1st column of the 2nd and 3rd row of the
array myarray1.
myarray1 = np.array([[2.7,-2,19],[0,3.4,99.9],[10.6,0,13]])
myarray1[1:3,0:3]

array([[ 0. ,  3.4, 99.9],
       [10.6,  0. , 13. ]])

# g) Reverse the array of vowels.
vowels = np.array(['a','e','i','o','u'])
vowels[::-1]

array(['u', 'o', 'i', 'e', 'a'], dtype='<U1')

# h) Divide all elements of array ones by 3.
ones = np.ones((2,5),dtype=int)
ones/3

array([[0.33333333, 0.33333333, 0.33333333, 0.33333333, 0.33333333],
       [0.33333333, 0.33333333, 0.33333333, 0.33333333, 0.33333333]])

# i) Add the arrays myarray1 and myarray2.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray1 + myarray2

array([[  6.7,   6. ,  -7. ],
       [ 16. ,  23.4, 123.9],
       [ 38.6,  32. ,  49. ]])
```

```
# j) Subtract myarray1 from myarray2 and store the result in a new array
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
newarray=myarray2-myarray1
newarray

array([[  1.3,  10. ,  31. ],
       [ 16. ,  16.6, -75.9],
       [ 17.4,  32. ,  23. ]])

# k) Multiply myarray1 and myarray2 element wise.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray2 * myarray1

array([[  10.8,  -16. , -228. ],
       [   0. ,   68. , 2397.6],
       [ 296.8,    0. ,  468. ]])

# l) Do the matrix multiplication of myarray1 and myarray2 and store the
result in a new array myarray3.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray3 = myarray2 @ myarray1
myarray3

array([[ 138. ,   19.2,  879.2],
       [ 297.6,   36. , 2006. ],
       [ 457.2,   52.8, 3132.8]])

# m) Divide myarray1 by myarray2.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray1 / myarray2

array([[ 0.675     , -0.25      , -1.58333333],
       [ 0.        ,  0.17      ,  4.1625    ],
       [ 0.37857143,  0.        ,  0.36111111]])

# n) Find the cube of all elements of myarray1 and divide the resulting
array by 2.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarraycube = myarray1 ** 3
myarraycube/2

array([[ 9.841500e+00, -4.000000e+00, -3.429500e+03],
       [ 0.000000e+00,  1.965200e+01,  4.985015e+05],
       [ 5.955080e+02,  0.000000e+00,  1.098500e+03]])
```

```
# o) Find the square root of all elements of myarray2 and divide the
resulting array by 2. The result should be rounded to two places of
decimals.
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
arraysqrt = np.sqrt(myarray2)
newarraysqrt = arraysqrt/2
np.around(newarraysqrt,2)

array([[1.  , 1.41, 1.73],
       [2.  , 2.24, 2.45],
       [2.65, 2.83, 3.  ]])

# p) Find the transpose of ones and myarray2.
ones = np.ones((2,5),dtype=int)
ones.transpose()
array([[1, 1],
       [1, 1],
       [1, 1],
       [1, 1],
       [1, 1]])
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray2.transpose()
array([[ 4., 16., 28.],
       [ 8., 20., 32.],
       [12., 24., 36.]])

# q) Sort the array vowels in reverse.
vowels = np.array(['a','e','i','o','u'])
vowels[ : : -1]

array(['u', 'o', 'i', 'e', 'a'], dtype='<U1')

# r) Sort the array myarray1 such that it brings the lowest value of the
column in the first row and so on.
myarray1 = np.array([[2.7,-2,-19],[0,3.4,99.9],[10.6,0,13]])
myarray1.sort()
myarray1

array([[-19. ,  -2. ,   2.7],
       [  0. ,   3.4,  99.9],
       [  0. ,  10.6,  13. ]])

# s) Use NumPy. split() to split the array myarray2 into 5 arrays column
wise. Store your resulting arrays in myarray2A, myarray2B, myarray2C,
myarray2D and myarray2E. Print the arrays myarray2A, myarray2B, myarray2C,
myarray2D and myarray2E
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray2A, myarray2B, myarray2C = np.split(myarray2,[1,3])
myarray2A,myarray2B,myarray2C

(array([[ 4.,  8., 12.]]),
 array([[16., 20., 24.],
        [28., 32., 36.]]),
 array([], shape=(0, 3), dtype=float64))
```

```
# t) Split the array zeros at array index 2, 5, 7, 8 and store the resulting
arrays in zerosA, zerosB, zerosC and zerosD and print them.
zeros = np.zeros(10,dtype=int)
zerosnew = np.split(zeros,[2,5,7,8])
zerosA = zerosnew[0]
zerosB = zerosnew[1]
zerosC = zerosnew[2]
zerosD = zerosnew[3]

# u) Concatenate the arrays myarray2A, myarray2B and myarray2C into an array
having 3 rows and 3 columns
myarray2 = np.arange(4,45,4,dtype=float)
myarray2.resize(3,3)
myarray2A, myarray2B, myarray2C = np.split(myarray2,[1,3])
np.concatenate((myarray2A, myarray2B, myarray2C),axis=0)

array([[ 4.,  8., 12.],
       [16., 20., 24.],
       [28., 32., 36.]])

# v) Create a 2-D array called myarray4 using arange() having 14 rows and 3
columns with start value = -1, step size 0.25 having. Split this array row
wise into 3 equal parts and print the result.

myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
np.split(myarray4,3,axis=1)

[array([[-1.  ],
        [-0.25],
        [ 0.5 ],
        [ 1.25],
        [ 2.  ],
        [ 2.75],
        [ 3.5 ],
        [ 4.25],
        [ 5.  ],
        [ 5.75],
        [ 6.5 ],
        [ 7.25],
        [ 8.  ],
        [ 8.75]]),
 array([[-0.75],
        [ 0.  ],
        [ 0.75],
        [ 1.5 ],
        [ 2.25],
        [ 3.  ],
        [ 3.75],
        [ 4.5 ],
        [ 5.25],
        [ 6.  ],
        [ 6.75],
        [ 7.5 ],
        [ 8.25],
        [ 9.  ]]),
```

```
 array([[-0.5 ],
        [ 0.25],
        [ 1.  ],
        [ 1.75],
        [ 2.5 ],
        [ 3.25],
        [ 4.  ],
        [ 4.75],
        [ 5.5 ],
        [ 6.25],
        [ 7.  ],
        [ 7.75],
        [ 8.5 ],
        [ 9.25]])]
```

w) Using the myarray4 created in the above questions, write commands for the following:

```
# i. Find the sum of all elements.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.sum()
```

```
173.25
```

```
# ii. Find the sum of all elements row wise.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.sum(axis=1)
```

```
array([-2.25,  0.  ,  2.25,  4.5 ,  6.75,  9.  , 11.25, 13.5 , 15.75,
       18.  , 20.25, 22.5 , 24.75, 27.  ])
```

```
# iii. Find the sum of all elements column wise.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.sum(axis=0)
```

```
array([54.25, 57.75, 61.25])
```

```
# iv. Find the max of all elements.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.max()
```

```
9.25
```

```
# v. Find the min of all elements in each row.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.min(axis=1)
```

```
array([-1.  , -0.25,  0.5 ,  1.25,  2.  ,  2.75,  3.5 ,  4.25,  5.  ,
        5.75,  6.5 ,  7.25,  8.  ,  8.75])
```

```
# vi. Find the mean of all elements in each row.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.mean(axis=1)
```

```
array([-0.75,  0.  ,  0.75,  1.5 ,  2.25,  3.  ,  3.75,  4.5 ,  5.25,
        6.  ,  6.75,  7.5 ,  8.25,  9.  ])
```

```
# vii. Find the standard deviation column wise.
myarray4 = np.arange(-1,45,0.25,dtype=float)
myarray4.resize(14,3)
myarray4.std(axis=0)
```

```
array([3.02334666, 3.02334666, 3.02334666])
```

Lab Cycle : I
Exp.No : 2

Aim: Write a Python program to do the following operations(Using NumPy Library)

a) Create multi-dimensional arrays and find its shape and dimension
b) Create a matrix full of zeros and ones
c) Reshape and flatten data in the array
d) Append data vertically and horizontally
e) Apply indexing and slicing on array
f) Use statistical functions on array - Min, Max, Mean, Median and Standard Deviation.

PROCEDURE:
1. Create: Open a new file in Python shell, write a program and save the program with .py extension.
2. Execute: Go to Run -> Run module (F5)

PROGRAM LOGIC:

a) Create multi-dimensional arrays and find its shape and dimension

```
Import numpy as np
#creation of multi-dimensional
array a=np.array([[1,2,3],[2,3,4],[3,4,5]])

#shape
b=a.shape
print("shape:",a.shape)

#dimension
c=a.ndim
print("dimensions:",a.ndim)
```

b) Create a matrix full of zeros and ones

```
#matrix full of zeros
z=np.zeros((2,2))
print("zeros:",z)

#matrix full of ones
o=np.ones((2,2))
print("ones:",o)
```

c) Reshape and flatten data in the array

```
#matrix reshape
a=np.array([[1,2,3,4],[2,3,4,5],[3,4,5,6],[4,5,6,7]])
b=a.reshape(4,2,2)
print("reshape:",b)

#matrix flatten
c=a.flatten()
print("flatten:",c)
```

d) Append data vertically and horizontally

```
#Appending data vertically
x=np.array([[10,20],[80,90]])
y=np.array([[30,40],[60,70]])
v=np.vstack((x,y))
print("vertically:",v)
#Appending data horizontally
h=np.hstack((x,y))
print("horizontally:",h)
```

e) Apply indexing and slicing on array

```
#indexing
a=np.array([[1,2,3,4],[2,3,4,5],[3,4,5,6],[4,5,6,7]])
temp = a[[0, 1, 2, 3], [1, 1, 1, 1]]
print("indexing",temp)

#slicing
i=a[:4,::2]
print("slicing",i
```

f) Use statistical functions on array - Min, Max, Mean, Median and Standard Deviation

```
#min for finding minimum of an array
a=np.array([[1,3,-1,4],[3,-2,1,4]])
b=a.min() print("minimum:",b)

#max for finding maximum of an array
C=a.max()
Print("maximum",c)

#mean
a=np.array([1,2,3,4,5])
d=a.mean()
print("mean:",d)

#median
e=np.median(a)
print("median:",e)

#standard deviation
f=a.std()
print("standard deviation",f)
```

INPUT/OUTPUT:

a) shape: (3, 3)
dimensions: 2
zeros:
[[0. 0.]
[0. 0.]]
ones:
[[1. 1.]

[1. 1.]]

b) reshape:
[[[1 2]
[3 4]]
[[2 3]
[4 5]]
[[3 4]
[5 6]]
[[4 5]
[6 7]]]
flatten: [1 2 3 4 2 3 4 5 3 4 5 6 4 5 6 7]

c) vertically:
[[10 20]
[80 90]
[30 40]
[60 70]]

horizontally:
[[10 20 30 40]
[80 90 60 70]]

d) indexing
[2 3 4 5]
slicing
[[1 3]
[2 4]
[3 5]
[4 6]]

e) minimum: -2
maximum: 4
mean: 3
median: 3 s
tandard deviation: 1.4142135623730951

Lab Cycle : I
Exp.No : 3

Aim: Write a Python program to do the following operations(Using NumPy Library)

a) Dot and matrix product of two arrays
b) Compute the Eigen values of a matrix
c) Solve a linear matrix equation such as $3 * x^0 + x^1 = 9$ and $x^0 + 2 * x^1 = 8$.
d) Compute the multiplicative inverse of a matrix
e) Compute the rank of a matrix
f) Compute the determinant of an array.

PROCEDURE:

1. Create: Open a new file in Python shell, write a program and save the program with .py extension.
2. Execute: Go to Run -> Run module (F5)

PROGRAM LOGIC:

a) Dot and matrix product of two arrays

```
#dot product of two arrays
Import numpy as np
a=np.array([1,2,3])
b=np.array([2,3,4])
print("dot product of one dimension is:", np.dot(a,b))

#matrix elements multiplication
a=np.array([[1,2],[3,4]])
b=np.array([[1,2],[3,4]])
print("element multiplication of matrix;", np.multiply(a,b))

#matrix multiplication
print("matrix multiplication", np.matmul(a,b))
```

b) Compute the Eigen values of a matrix

```
#eigen values of a matrix
Import numpy as np
a=np.array([[1,2],[3,4]])
eigvalues,eigvectors=np.linalg.eig(a)
print("eigen value:",eigvalues,"eigen vector:",eigvectors)
```

c) Solve a linear matrix equation such as $3 * x^0 + x^1 = 9, x^0 + 2 * x^1 = 8$

```
#linear matric equation
Import numpy as np
a=np.array([[3,1],[1,2]])
b=np.array([[9],[8]])
a_inv=np.linalg.inv(a)
e=np.matmul(a_inv,b)
print("linear equation:",e)
```

d) Compute the multiplicative inverse of a matrix

```
#multiplicative inverse
import numpy as np
a=np.array([[3,1],[1,2]])
a_inv=np.linalg.inv(a)
print("a inverse:",a_inv)
```

e) Compute the rank of a matrix

```
#matric rank
a=np.array([[3,1],[1,2]])
b=np.linalg.matrix_rank(a)
print("rank:",b)
```

f) Compute the determinant of an array

```
a=np.array([[3,1],[1,2]])
b=np.linalg.det(a)
print("determinant:",b)
```

INPUT/OUTPUT:

a)

```
dot product of one dimension is: 20
element multiplication of matrix;
        [[ 1 4]
       [ 9 16]]
        matrix multiplication
        [[ 7 10]
       [15 22]]
```

b) eigen value:

```
[-0.37228132 5.37228132]
```

eigen vector:

```
[[-0.82456484 -0.41597356]
 [0.56576746 -0.90937671]]
```

c) linear equation:

```
[[ 3.6 -1.8]
 [-1.6 4.8]]
```

d) a inverse:

```
[[ 0.4 -0.2]
 [-0.2 0.6]]
```

e)

```
rank: 2
```

f)

```
determinant: 5.000000000000001
```

Lab Cycle : I
Exp.No : 4

Aim: Suppose that the data for analysis includes the attribute *age*. The *age* values for the data tuples are (in increasing order) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

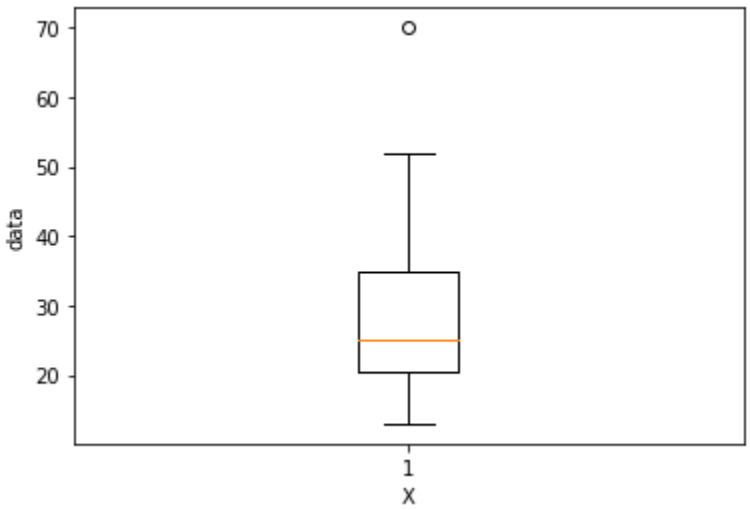
Write a Python program for the following using the above data:

a) Calculate measures of central tendency.
b) To find (roughly) the first quartile (Q1) and the third quartile (Q3) of the data.
c) Give the five-number summary of the data.
d) Show a boxplot of the data.

Solution:

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
x=np.array(([ 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30,
33, 33, 35, 35,
35, 35, 36, 40, 45, 46, 52, 70]))
print(x)
d=pd.Series(x)
print(d)
[13 15 16 16 19 20 20 21 22 22 25 25 25 25 30 33 33 35 35 35 35 36 40 45
 46 52 70]
0     13
1     15
2     16
3     16
4     19
5     20
6     20
7     21
8     22
9     22
10    25
11    25
12    25
13    25
14    30
15    33
16    33
17    35
18    35
19    35
20    35
21    36
22    40
23    45
24    46
25    52
26    70
```

```
dtype: int32

#Calculate measures of central tendency
mean_ = statistics.mean(x)
print("mean:",mean_)
mean: 29

median_ = statistics.median(x)
print("median:",median_)
median: 25

mode_ = statistics.multimode(x)
print("mode:",mode_)
mode: [25, 35]

#To find (roughly) the first quartile (Q1) and the third quartile (Q3) of
the data.
quartile=statistics.quantiles(x, n=4, method='inclusive')
print(quartile)
print ("Q1=",quartile[0])
print ("Q3=",quartile[2])
[20.5, 25.0, 35.0]
Q1= 20.5
Q3= 35.0

#Give the five-number summary of the data
result=d.describe()
print(result)
#print("count=",result['count'])
#print("mean=",result['mean'])
#print("standard deviation=",result['std'])
#print("minimum;",result['min'])
#print("maximum",result['max'])
#print("Q1=",result['25%'])
#print("Q2=",result['50%'])
#print("Q3=",result['75%'])
count    27.000000
mean     29.962963
std      12.942124
min      13.000000
25%      20.500000
50%      25.000000
75%      35.000000
max      70.000000
dtype: float64

#Show a boxplot of the data
import matplotlib.pyplot as plt
fig, d = plt.subplots()
plt.xlabel('X')
plt.ylabel('data')
d.boxplot((x))
plt.show()
```


70
60
50
40
30
20
data
1
X

Lab Cycle : I
Exp.No : 5

Aim: Suppose that a hospital tested the age and body fat data for 18 randomly selected adults with the following results:

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

a) Calculate the mean, median, and standard deviation of age and %fat.
b) Draw the boxplots for age and %fat.
c) Draw a scatter plot and a q-q plot based on these two variables.

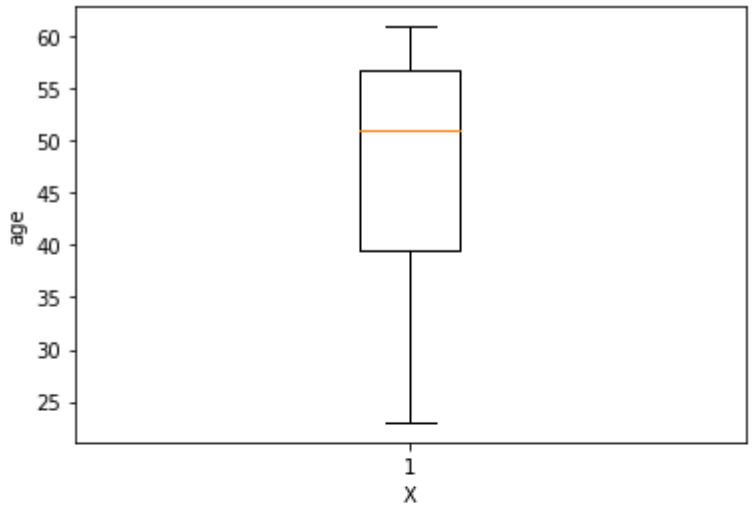
Solution:

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
age=np.array(([23,23,27,27,39,41,47,49,50,52,54,54,56,57,58,58,60,61]))
print("age",age)
fat=np.array(([9.5,26.5,7.8,17.8,31.4,25.9,27.4,27.2,31.2,34.6,42.5,28.8,33.
4,30.2,34.1,32.9,41.2,35.7]))
print("fat",fat)
d=pd.Series(age)
print(d)
f=pd.Series(fat)
print(f)
age [23 23 27 27 39 41 47 49 50 52 54 54 56 57 58 58 60 61]
fat [ 9.5 26.5  7.8 17.8 31.4 25.9 27.4 27.2 31.2 34.6 42.5 28.8 33.4 30.2
 34.1 32.9 41.2 35.7]
0     23
1     23
2     27
3     27
4     39
5     41
6     47
7     49
8     50
9     52
10    54
11    54
12    56
13    57
14    58
15    58
16    60
17    61
dtype: int32
0      9.5
```

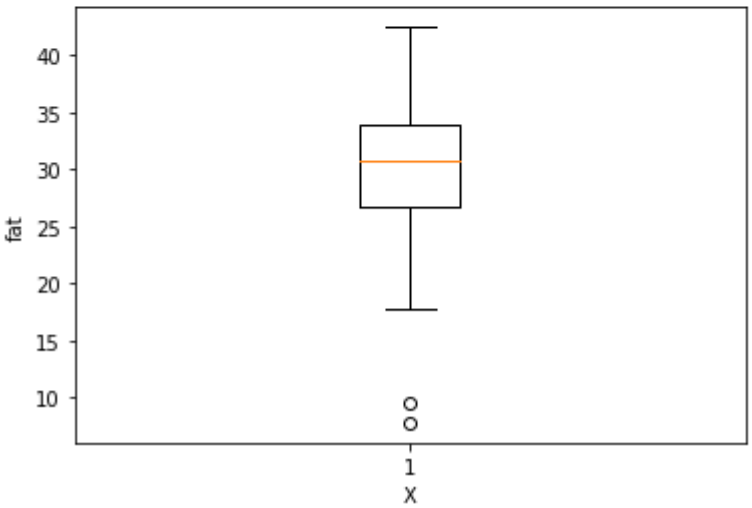
```
1     26.5
2      7.8
3     17.8
4     31.4
5     25.9
6     27.4
7     27.2
8     31.2
9     34.6
10    42.5
11    28.8
12    33.4
13    30.2
14    34.1
15    32.9
16    41.2
17    35.7
dtype: float64

#Calculate the mean of age and %fat.
mean_ = statistics.mean(age)
print(" age mean:",mean_)
mean_ = statistics.mean(fat)
print(" fat mean:",mean_)
 age mean: 46
 fat mean: 28.783333333333335

#Calculate the median of age and %fat.
median_ = statistics.median(age)
print(" age median:",median_)
median_ = statistics.median(fat)
print("fat median:",median_)
 age median: 51.0
fat median: 30.7

#Calculate the standard deviation of age and %fat.
std=np.std(age)
print(" age standard deviation=",std)
std=np.std(fat)
print(" fat standard deviation=",std)
 age standard deviation= 12.846193652519204
 fat standard deviation= 8.993655170915401

#Draw the boxplots for age
import matplotlib.pyplot as plt
fig, d = plt.subplots()
d.boxplot((age))
print(" boxplots for age")
plt.xlabel('X')
plt.ylabel('age')
plt.show()

#Draw the boxplots for %fat
import matplotlib.pyplot as plt
fig, f = plt.subplots()
f.boxplot((fat))
print(" boxplots for fat")
```

```
plt.xlabel('X')
plt.ylabel('fat')
plt.show()
```

boxplots for age

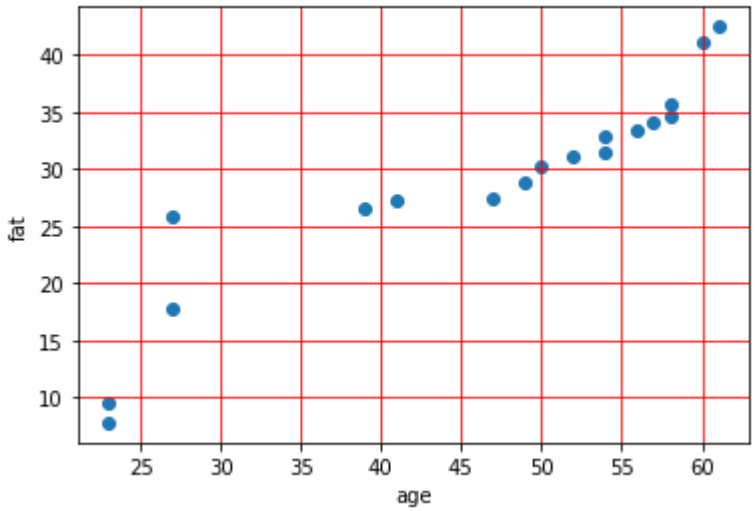


boxplots for fat



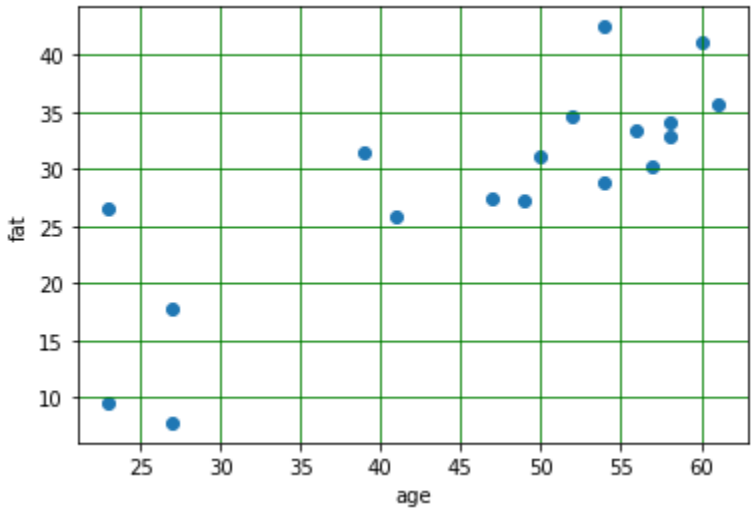

```
#Draw  a q-q plot based on these two variables.
import numpy as np
import statsmodels.api as sm
import pylab as py
import matplotlib.pyplot as plt
plt.figure()
plt.scatter(np.sort(age), np.sort(fat))
print("q-q plot")
plt.xlabel('age')
plt.ylabel('fat')
plt.grid(color= 'red')
plt.show()
```

q-q plot




```
##Draw a scatter plot based on these two variables.
import matplotlib.pyplot as plt

plt.scatter(age, fat)
print("scatter plot")
plt.xlabel('age')
plt.ylabel('fat')
plt.grid(color= 'green')
plt.show()
```

scatter plot



Lab Cycle : I
Exp.No : 6

Aim: Write a Python program to perform the following using the three fundamental Pandas data structures: the Series, DataFrame, and Index.

a) Series as generalized NumPy array
b) Series as specialized dictionary
c) Constructing Series objects
d) DataFrame as a generalized NumPy array
e) DataFrame as specialized dictionary
f) Constructing DataFrame objects:
    i. From a single Series object.
    ii. From a list of dicts.
    iii. From a dictionary of Series objects.
    iv. From a two-dimensional NumPy array.
    v. From a NumPy structured array.
g) Index as immutable array.
h) Index as ordered set.
i) Data Selection in Series:
    vi. Series as dictionary
    vii. Series as one-dimensional array
    viii. Indexers: loc, iloc, and ix
j) Data Selection in DataFrame
    ix. DataFrame as a dictionary
    x. DataFrame as two-dimensional array

Solution:

```
import pandas as pd

#a) Series as generalized NumPy array

data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a', 'b', 'c', 'd'])
data
```

```
a    0.25
b    0.50
c    0.75
d    1.00
dtype: float64
```

```
#b) Series as specialized dictionary
population_dict = {'California': 38332521,
                   'Texas': 26448193,
                   'New York': 19651127,
                   'Florida': 19552860,
                   'Illinois': 12882135}
population = pd.Series(population_dict)
```

```
population

California    38332521
Texas         26448193
New York      19651127
Florida       19552860
Illinois      12882135
dtype: int64

#c) Constructing Series objects
import numpy as np
import pandas as pd
Series=pd.Series(data=[2,4,6,8,10,12,14])
print("index:",Series.index)
print("shape:",Series.shape)
print("dtype:",Series.dtype)
print("size:",Series.size)
print("empty:",Series.empty)
print("hasnans:",Series.hasnans)
print("nbytes:",Series.nbytes)
print("ndim:",Series.ndim)
index: RangeIndex(start=0, stop=7, step=1)
shape: (7,)
dtype: int64
size: 7
empty: False
hasnans: False
nbytes: 56
ndim: 1

#d) DataFrame as a generalized NumPy array
area_dict = {'California': 423967, 'Texas': 695662, 'New York': 141297,
             'Florida': 170312, 'Illinois': 149995}
area = pd.Series(area_dict)
area

California    423967
Texas         695662
New York      141297
Florida       170312
Illinois      149995
dtype: int64

states = pd.DataFrame({'population': population,
                       'area': area})
states
```

	population	area
California	38332521	423967
Texas	26448193	695662
New York	19651127	141297
Florida	19552860	170312

	population	area
Illinois	12882135	149995

```
#e) DataFrame as specialized dictionary

states['area']
```

```
California    423967
Texas         695662
New York      141297
Florida       170312
Illinois      149995
Name: area, dtype: int64
```

```
#f) Constructing DataFrame objects:
#i. From a single Series object.
pd.DataFrame(population, columns=['population'])
```

	population
California	38332521
Texas	26448193
New York	19651127
Florida	19552860
Illinois	12882135

```
#ii. From a list of dicts.
data = [{'a': i, 'b': 2 * i}
        for i in range(3)]
pd.DataFrame(data)
```

	a	b
0	0	0
1	1	2
2	2	4

```
#iii. From a dictionary of Series objects
pd.DataFrame({'population': population,
              'area': area})
```

	population	area
California	38332521	423967
Texas	26448193	695662
New York	19651127	141297

	population	area
Florida	19552860	170312
Illinois	12882135	149995

```
#iv. From a two-dimensional NumPy array.
pd.DataFrame(np.random.rand(3, 2),
             columns=['foo', 'bar'],
             index=['a', 'b', 'c'])
```

	foo	bar
a	0.983789	0.317771
b	0.691952	0.490791
c	0.851536	0.421588

```
#v. From a NumPy structured array.
A = np.zeros(3, dtype=[('A', 'i8'), ('B', 'f8')])
print(A)
pd.DataFrame(A)
[(0, 0.) (0, 0.) (0, 0.)]
```

	A	B
0	0	0.0
1	0	0.0
2	0	0.0

```
ind = pd.Index([2, 3, 5, 7, 11])

#g) Index as immutable array

print(ind[1])
print(ind[::2])
ind[1] = 0
3
Int64Index([2, 5, 11], dtype='int64')

#h) Index as ordered set.
indA = pd.Index([1, 3, 5, 7, 9])
indB = pd.Index([2, 3, 5, 7, 11])
print("intersection:",indA & indB)
print(" union:",indA | indB)
print("symmetric difference",indA ^ indB  )

intersection: Int64Index([3, 5, 7], dtype='int64')
union: Int64Index([1, 2, 3, 5, 7, 9, 11], dtype='int64')
symmetric difference Int64Index([1, 2, 9, 11], dtype='int64')

#i) Data Selection in Series:
```

```
import pandas as pd

#i. Series as dictionary
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a', 'b', 'c', 'd'])
data
```

```
a    0.25
b    0.50
c    0.75
d    1.00
dtype: float64
```

```
#ii. Series as one-dimensional array
# slicing by explicit index
data['a':'c']
```

```
a    0.25
b    0.50
c    0.75
dtype: float64
```

```
#iii. Indexers: loc, iloc, and ix
data = pd.Series(['a', 'b', 'c'], index=[1, 3, 5])
data
```

```
1    a
3    b
5    c
dtype: object
```

```
# explicit index when indexing
print(data[1])
# implicit index when slicing
data[1:3]
a
```

```
3    b
5    c
dtype: object
```

```
#Loc
print(data.loc[1])
data.loc[1:3]
a
```

```
1    a
3    b
dtype: object
```

```
#iloc
print(data.iloc[1])
data.iloc[1:3]
b
```

```
3    b
```

```
5    c
dtype: object
```

```
#j) Data Selection in DataFrame
        #i. DataFrame as a dictionary
area = pd.Series({'California': 423967, 'Texas': 695662,
                  'New York': 141297, 'Florida': 170312,
                  'Illinois': 149995})
pop = pd.Series({'California': 38332521, 'Texas': 26448193,
                 'New York': 19651127, 'Florida': 19552860,
                 'Illinois': 12882135})
data = pd.DataFrame({'area':area, 'pop':pop})
data
```

	area	pop
California	423967	38332521
Texas	695662	26448193
New York	141297	19651127
Florida	170312	19552860
Illinois	149995	12882135

```
#ii. DataFrame as two-dimensional array
import numpy as np
import pandas as pd

df = pd.DataFrame({'color': ['red', 'blue', 'black'] * 2,
                   'vehicle': ['car', 'truck'] * 3,
                   'value': np.arange(1,7)})
df
```

	color	vehicle	value
0	red	car	1
1	blue	truck	2
2	black	car	3
3	red	truck	4
4	blue	car	5
5	black	truck	6

Lab Cycle : I
Exp.No : 7

Aim: Write a Python program to perform the following:

a) Input as CSV File
b) Reading a CSV File
c) Reading Specific Rows
d) Reading Specific Columns
e) Reading Specific Columns and Rows
f) Reading Specific Columns for a Range of Rows
g) Identify the missing data
h) Identify the outlier data
i) Replace with mean or mode
j) Remove Blank Rows
k) Data Categories
l) Data types
m) Analyze the data
n) Visualize the data
o) Find correlation among all attributes

Solution:

```
#a) Input as csv File
import pandas as pd
import numpy as np
import csv
frame = pd.DataFrame([['booker12',9012,'Rachel','Booker'],
                      ['grey07',2070,'Laura','Grey'],
                      ['johnson81',4081,'Craig','Johnson'],
                      ['jenkins46',9346,'Mary','Jenkins'],
                      ['smith79',5079,'Jamie','Smith']],
                      index=[1,2,3,4,5],
                      columns=[
'Username','Identifier','Firstname','Lastname'])
frame
```

	Username	Identifier	Firstname	Lastname
1	booker12	9012	Rachel	Booker
2	grey07	2070	Laura	Grey
3	johnson81	4081	Craig	Johnson
4	jenkins46	9346	Mary	Jenkins
5	smith79	5079	Jamie	Smith

```
#Reading a CSV File
import pandas as pd

df = pd.read_csv('Z:\data.csv')

print(df.to_string())
   Username   Identifier First name Last name
0   booker12        9012     Rachel    Booker
1     grey07        2070      Laura      Grey
2  johnson81        4081      Craig   Johnson
3  jenkins46        9346       Mary   Jenkins
4    smith79        5079      Jamie     Smith

#Reading Specific Rows
import csv

with open('Z:\data.csv') as csv_file:
    csv_reader = csv.reader(csv_file)
    rows = list(csv_reader)

    print(rows[3])
    print(rows[2])

['johnson81', '4081', 'Craig', 'Johnson']
['grey07', '2070', 'Laura', 'Grey']

#Reading Specific Cols
import csv

with open('Z:\data.csv') as csv_file:
    csv_reader = csv.reader(csv_file)
    cols = list(csv_reader)

    print(cols[1])
    print(cols[2])
['booker12', '9012', 'Rachel', 'Booker']
['grey07', '2070', 'Laura', 'Grey']

#Reading Specific Rows and cols
import csv

with open('Z:\data.csv') as csv_file:
    csv_reader = csv.reader(csv_file)
    rows = list(csv_reader)

    print(rows[3])
    print(rows[2])
    print(cols[1])
    print(cols[2])

['johnson81', '4081', 'Craig', 'Johnson']
['grey07', '2070', 'Laura', 'Grey']

#Reading Specific Columns for a Range of Rows
import pandas as pd
data = pd.read_csv('Z:\data.csv')
print (data.loc[2:6])
```

```
   Username   Identifier First name Last name
2  johnson81        4081      Craig   Johnson
3  jenkins46        9346       Mary   Jenkins
4    smith79        5079      Jamie     Smith

#Identify the missing data
import pandas as pd

df = pd.read_csv('Z:\data.csv')
df.isnull()
```

	Username; Identifier;First name;Last name
0	False
1	False
2	False
3	False
4	False

```
#h) Identify the outlier data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading the data
df = pd.read_csv('Z:\data.csv')
print(df.shape)
print(df.info())
Matplotlib is building the font cache; this may take a moment.
(5, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Username    5 non-null      object
 1   Identifier  5 non-null      int64
 2   First name  5 non-null      object
 3   Last name   5 non-null      object
dtypes: int64(1), object(3)
memory usage: 288.0+ bytes
None

df.describe()
```

	Identifier
count	5.000000
mean	5917.600000

	Identifier
std	3170.552523
min	2070.000000
25%	4081.000000
50%	5079.000000
75%	9012.000000
max	9346.000000

```
#Replace with mean or mode
import csv
df = pd.read_csv('Z:\data.csv')

df.dropna(inplace = True)

print(df.to_string())
   Username   Identifier First name Last name
0   booker12        9012     Rachel    Booker
1     grey07        2070      Laura      Grey
2  johnson81        4081      Craig   Johnson
3  jenkins46        9346       Mary   Jenkins
4    smith79        5079      Jamie     Smith

#j) Remove Blank Rows
import pandas as pd

df = pd.read_csv('Z:\data.csv')

new_df = df.dropna()

print(new_df.to_string())
   Username   Identifier First name Last name
0   booker12        9012     Rachel    Booker
1     grey07        2070      Laura      Grey
2  johnson81        4081      Craig   Johnson
3  jenkins46        9346       Mary   Jenkins
4    smith79        5079      Jamie     Smith

# Data Categories

#Data types
import csv
df = pd.read_csv('Z:\data.csv')
df.dtypes

Username      object
Identifier     int64
First name    object
Last name     object
dtype: object
```

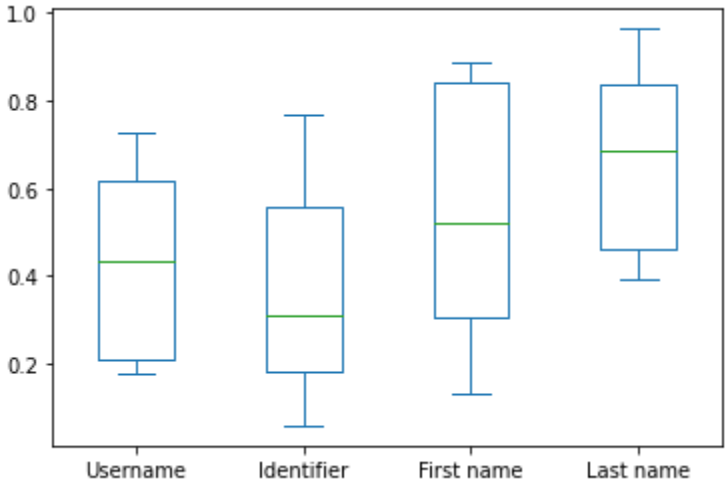
```
#Analyze the data
df.head()
```

	Username	Identifier	First name	Last name
0	booker12	9012	Rachel	Booker
1	grey07	2070	Laura	Grey
2	johnson81	4081	Craig	Johnson
3	jenkins46	9346	Mary	Jenkins
4	smith79	5079	Jamie	Smith

```
#Find correlation among all attributes
import csv
df = pd.read_csv('Z:\data.csv')
print(df.corr())
```

```
                Identifier
Identifier         1.0
```

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(6, 4), columns=['Username',
'Identifier','First name','Last name'])
df.plot.box()
```



Lab Cycle : I
Exp.No : 8

Aim: Write a python program to perform transformation of data using Discretization (Binning) and Normalization (MinMaxScaler or MaxAbsScaler) on given dataset.

Solution:

```
import numpy as np
import math
from sklearn.datasets import load_iris
from sklearn import datasets, linear_model, metrics

# load iris data set
dataset = load_iris()
a = dataset.data
b = np.zeros(150)

# take 1st column among 4 column of data set
for i in range (150):
    b[i]=a[i,1]

b=np.sort(b) #sort the array

# create bins
bin1=np.zeros((30,5))
bin2=np.zeros((30,5))
bin3=np.zeros((30,5))

# Bin mean
for i in range (0,150,5):
    k=int(i/5)
    mean=(b[i] + b[i+1] + b[i+2] + b[i+3] + b[i+4])/5
    for j in range(5):
        bin1[k,j]=mean
print("Bin Mean: \n",bin1)

# Bin boundaries
for i in range (0,150,5):
    k=int(i/5)
    for j in range (5):
        if (b[i+j]-b[i]) < (b[i+4]-b[i+j]):
            bin2[k,j]=b[i]
        else:
            bin2[k,j]=b[i+4]
print("Bin Boundaries: \n",bin2)

# Bin median
for i in range (0,150,5):
    k=int(i/5)
    for j in range (5):
        bin3[k,j]=b[i+2]
print("Bin Median: \n",bin3)
```

```
Bin Mean:
 [[2.18 2.18 2.18 2.18 2.18]
```

```
 [2.34 2.34 2.34 2.34 2.34]
 [2.48 2.48 2.48 2.48 2.48]
 [2.52 2.52 2.52 2.52 2.52]
 [2.62 2.62 2.62 2.62 2.62]
 [2.7  2.7  2.7  2.7  2.7 ]
 [2.74 2.74 2.74 2.74 2.74]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.86 2.86 2.86 2.86 2.86]
 [2.9  2.9  2.9  2.9  2.9 ]
 [2.96 2.96 2.96 2.96 2.96]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.04 3.04 3.04 3.04 3.04]
 [3.1  3.1  3.1  3.1  3.1 ]
 [3.12 3.12 3.12 3.12 3.12]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.26 3.26 3.26 3.26 3.26]
 [3.34 3.34 3.34 3.34 3.34]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.5  3.5  3.5  3.5  3.5 ]
 [3.58 3.58 3.58 3.58 3.58]
 [3.74 3.74 3.74 3.74 3.74]
 [3.82 3.82 3.82 3.82 3.82]
 [4.12 4.12 4.12 4.12 4.12]]
Bin Boundaries:
 [[2.  2.3 2.3 2.3 2.3]
 [2.3 2.3 2.3 2.4 2.4]
 [2.4 2.5 2.5 2.5 2.5]
 [2.5 2.5 2.5 2.5 2.6]
 [2.6 2.6 2.6 2.6 2.7]
 [2.7 2.7 2.7 2.7 2.7]
 [2.7 2.7 2.7 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.9 2.9 2.9]
 [2.9 2.9 2.9 2.9 2.9]
 [2.9 2.9 3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.1 3.1]
 [3.1 3.1 3.1 3.1 3.1]
 [3.1 3.1 3.1 3.1 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.3 3.3 3.3]
 [3.3 3.3 3.3 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.5 3.5 3.5 3.5 3.5]
 [3.5 3.6 3.6 3.6 3.6]
```

```
 [3.7 3.7 3.7 3.8 3.8]
 [3.8 3.8 3.8 3.8 3.9]
 [3.9 3.9 3.9 4.4 4.4]]
Bin Median:
 [[2.2 2.2 2.2 2.2 2.2]
 [2.3 2.3 2.3 2.3 2.3]
 [2.5 2.5 2.5 2.5 2.5]
 [2.5 2.5 2.5 2.5 2.5]
 [2.6 2.6 2.6 2.6 2.6]
 [2.7 2.7 2.7 2.7 2.7]
 [2.7 2.7 2.7 2.7 2.7]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.9 2.9 2.9 2.9 2.9]
 [2.9 2.9 2.9 2.9 2.9]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.1 3.1 3.1 3.1 3.1]
 [3.1 3.1 3.1 3.1 3.1]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.3 3.3 3.3 3.3 3.3]
 [3.3 3.3 3.3 3.3 3.3]
 [3.4 3.4 3.4 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.5 3.5 3.5 3.5 3.5]
 [3.6 3.6 3.6 3.6 3.6]
 [3.7 3.7 3.7 3.7 3.7]
 [3.8 3.8 3.8 3.8 3.8]
 [4.1 4.1 4.1 4.1 4.1]]
```

Perform transformation of data using normalization (MinMaxScaler or MaxAbsScaler) on given dataset.

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = MinMaxScaler()
print(scaler.fit(data))
MinMaxScaler()
print("data:\n",scaler.data_max_)
print("Transformed data:\n",scaler.transform(data))
MinMaxScaler()
data:
 [ 1. 18.]
Transformed data:
 [[0.   0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.   1.  ]]
```

Lab Cycle : I
Exp.No : 9

Aim: Write a program to implement three frequent itemset mining using Apriori algorithms.

Solution:

```
data = [
        ['T100',['I1','I2','I5']],
        ['T200',['I2','I4']],
        ['T300',['I2','I3']],
        ['T400',['I1','I2','I4']],
        ['T500',['I1','I3']],
        ['T600',['I2','I3']],
        ['T700',['I1','I3']],
        ['T800',['I1','I2','I3','I5']],
        ['T900',['I1','I2','I3']]
        ]

init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)

sp = 0.4
s = int(sp*len(init))
s

from collections import Counter

c = Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = l
pos = 1
for count in range (2,1000):
    nc = set()
    temp = list(l)
    for i in range(0,len(temp)):
```

```
            for j in range(i+1,len(temp)):
                t = temp[i].union(temp[j])
                if(len(t) == count):
                    nc.add(temp[i].union(temp[j]))
        nc = list(nc)
        c = Counter()
        for i in nc:
            c[i] = 0
            for q in data:
                temp = set(q[1])
                if(i.issubset(temp)):
                    c[i]+=1
        print("C"+str(count)+":")
        for i in c:
            print(str(list(i))+": "+str(c[i]))
        print()
        l = Counter()
        for i in c:
            if(c[i] >= s):
                l[i]+=c[i]
        print("L"+str(count)+":")
        for i in l:
            print(str(list(i))+": "+str(l[i]))
        print()
        if(len(l) == 0):
            break
        pl = l
        pos = count
    print("Result: ")
    print("L"+str(pos)+":")
    for i in pl:
        print(str(list(i))+": "+str(pl[i]))
    print()

    from itertools import combinations
    for l in pl:
        c = [frozenset(q) for q in combinations(l,len(l)-1)]
        mmax = 0
        for a in c:
            b = l-a
            ab = l
            sab = 0
            sa = 0
            sb = 0
            for q in data:
                temp = set(q[1])
                if(a.issubset(temp)):
                    sa+=1
                if(b.issubset(temp)):
                    sb+=1
                if(ab.issubset(temp)):
                    sab+=1
            temp = sab/sa*100
            if(temp > mmax):
                mmax = temp
            temp = sab/sb*100
            if(temp > mmax):
```

```
            mmax = temp
        print(str(list(a))+" -> "+str(list(b))+" = "+str(sab/sa*100)+"%")
        print(str(list(b))+" -> "+str(list(a))+" = "+str(sab/sb*100)+"%")
    curr = 1
    print("choosing:", end=' ')
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
        temp = sab/sb*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
    print()
['I1', 'I2', 'I3', 'I4', 'I5']
C1:
['I1']: 6
['I2']: 7
['I3']: 6
['I4']: 2
['I5']: 2

L1:
['I1']: 6
['I2']: 7
['I3']: 6
['I4']: 2
['I5']: 2

C2:
['I2', 'I4']: 2
['I1', 'I4']: 1
['I1', 'I5']: 2
['I5', 'I2']: 2
['I3', 'I4']: 0
['I5', 'I3']: 1
['I5', 'I4']: 0
['I3', 'I2']: 4
['I1', 'I2']: 4
['I1', 'I3']: 4

L2:
['I2', 'I4']: 2
```

```
['I1', 'I5']: 2
['I5', 'I2']: 2
['I3', 'I2']: 4
['I1', 'I2']: 4
['I1', 'I3']: 4

C3:
['I5', 'I2', 'I4']: 0
['I1', 'I3', 'I2']: 2
['I1', 'I5', 'I2']: 2
['I3', 'I2', 'I4']: 0
['I1', 'I3', 'I5']: 1
['I5', 'I3', 'I2']: 1
['I1', 'I2', 'I4']: 1

L3:
['I1', 'I3', 'I2']: 2
['I1', 'I5', 'I2']: 2

C4:
['I5', 'I3', 'I2', 'I1']: 1

L4:

Result:
L3:
['I1', 'I3', 'I2']: 2
['I1', 'I5', 'I2']: 2

['I1', 'I3'] -> ['I2'] = 50.0%
['I2'] -> ['I1', 'I3'] = 28.57142857142857%
['I1', 'I2'] -> ['I3'] = 50.0%
['I3'] -> ['I1', 'I2'] = 33.33333333333333%
['I3', 'I2'] -> ['I1'] = 50.0%
['I1'] -> ['I3', 'I2'] = 33.33333333333333%
choosing: 1 3 5
['I1', 'I5'] -> ['I2'] = 100.0%
['I2'] -> ['I1', 'I5'] = 28.57142857142857%
['I1', 'I2'] -> ['I5'] = 50.0%
['I5'] -> ['I1', 'I2'] = 100.0%
['I5', 'I2'] -> ['I1'] = 100.0%
['I1'] -> ['I5', 'I2'] = 33.33333333333333%
choosing: 1 4 5
```

Lab Cycle : I
Exp.No : 10

Aim: Write a program to implement Decision tree algorithm.

Solution:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder#for train test splitting
from sklearn.model_selection import train_test_split#for decision tree
object
from sklearn.tree import DecisionTreeClassifier#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix#for
visualizing tree
from sklearn.tree import plot_tree

#reading the data
df = sns.load_dataset('iris')
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
#getting information of dataset
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df.shape
```

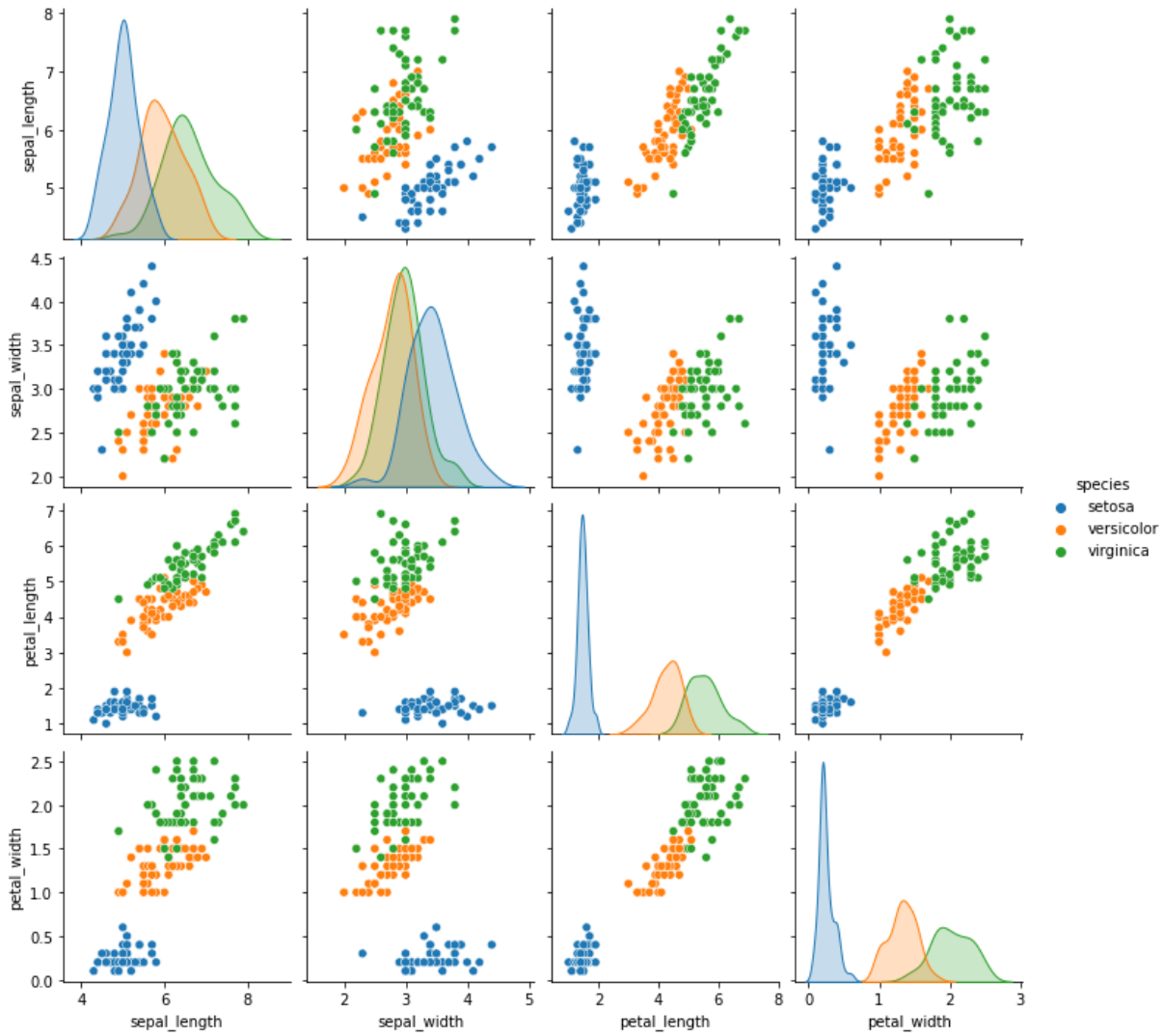
```
(150, 5)
```

```
df.isnull().any()
```

```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species         False
dtype: bool
```

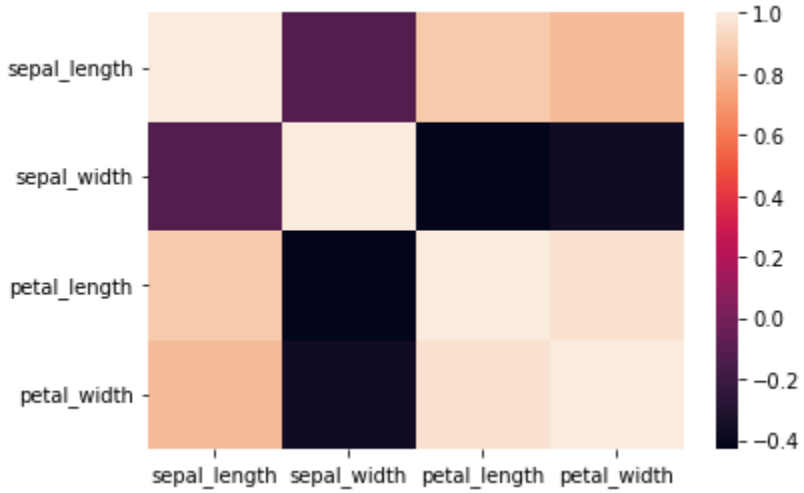
```
# let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue = 'species')
```

```
<seaborn.axisgrid.PairGrid at 0x1966ff22a60>
```



```
# correlation matrix
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



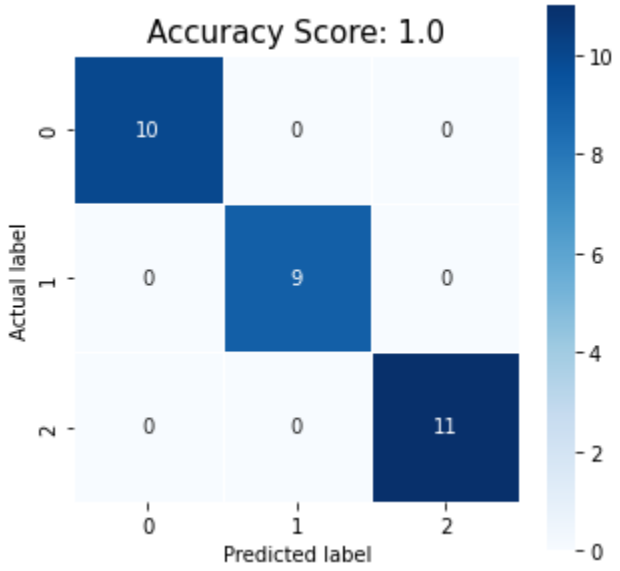

```
target = df['species']
df1 = df.copy()
df1 = df1.drop('species', axis =1)

# Defining the attributes
X = df1

target
```

```
0          setosa
1          setosa
2          setosa
3          setosa
4          setosa
          ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

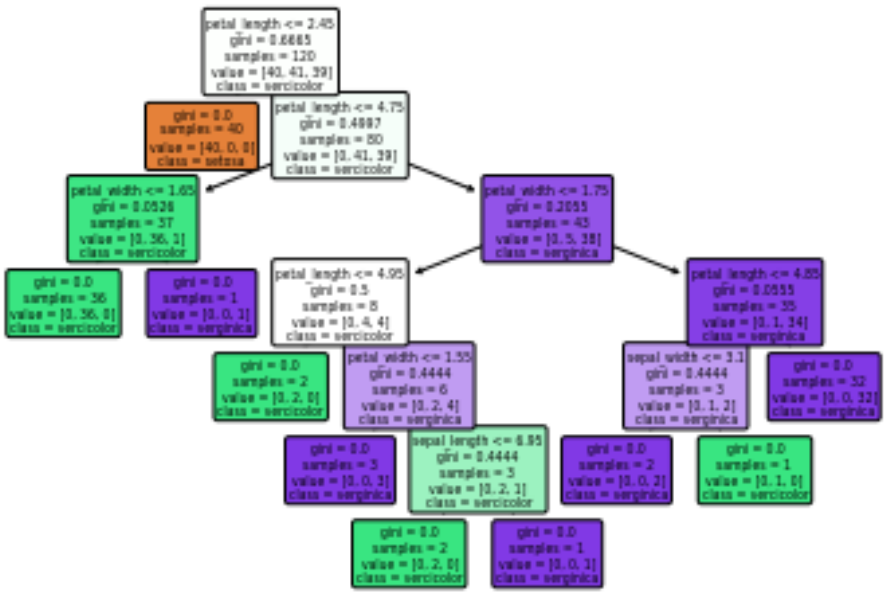
```
#label encoding
le = LabelEncoder()
target = le.fit_transform(target)
target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
y = target

# Splitting the data - 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2,
random_state = 42)
print("Trainingsplit input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
Trainingsplit input-  (120, 4)
Testing split input-  (30, 4)

# Defining the decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Decision Tree Classifier Created')
Decision Tree Classifier Created

# Predicting the values of test data
y_pred = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True, cmap ='Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)

Text(0.5, 1.0, 'Accuracy Score: 1.0')
```




```
# Visualising the graph without the use of graphvizplt.figure(figsize =
(20,20))
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.columns,
 class_names =["setosa", "vercicolor", "verginica"] , filled = True ,
precision = 4,
rounded = True)
```



Lab Cycle : I
Exp.No : 11

Aim: Write a program to implement Naïve Bayesian Classification.

Solution:

```
dataset = [
           [0,0,1,0,0],
           [0,0,1,1,0],
           [1,0,1,0,1],
           [2,1,1,0,1],
           [2,2,0,0,1],
           [2,2,0,1,0],
           [1,2,0,1,1],
           [0,1,1,0,0],
           [0,2,0,0,1],
           [2,1,0,0,1],
           [0,1,0,1,1],
           [1,1,1,1,1],
           [1,0,0,0,1],
           [2,1,1,1,0]
           ]
mp = dict()
for i in range(len(dataset)):
    row = dataset[i]
    y = row[-1]
    if (y not in mp):
        mp[y] = list()
    mp[y].append(row)

for label in mp:
    print(label)
    for row in mp[label]:
        print(row)

test = [2,1,0,1]

probYes = 1

count = 0
total = 0
for row in dataset:
    if(row[-1] == 1):
        count+=1
    total+=1
print("Total yes: "+str(count)+" / "+str(total))
probYes *= count/total
for i in range(len(test)):
    count = 0
    total = 0
    for row in mp[1]:
        if(test[i] == row[i]):
            count += 1
        total += 1
```

```
    print('for feature '+str(i+1))
    print(str(count)+" / "+str(total))
    probYes *= count/total

probNo = 1
count = 0
total = 0
for row in dataset:
    if(row[-1] == 0):
        count+=1
    total+=1
probNo *= count/total
print("Total no: "+str(count)+" / "+str(total))
for i in range(len(test)):
    count = 0
    total = 0
    for row in mp[0]:
        if(test[i] == row[i]):
            count += 1
        total += 1
    print('for feature '+str(i+1))
    print(str(count)+" / "+str(total))
    probNo *= count/total

print("probability of playing golf YES: "+str(probYes))
print("probability of playing golf NO:"+str(probNo))


prob = probYes/(probYes+probNo)
print("Probability of playing golf: "+str(prob*100)+"%")
0
[0, 0, 1, 0, 0]
[0, 0, 1, 1, 0]
[2, 2, 0, 1, 0]
[0, 1, 1, 0, 0]
[2, 1, 1, 1, 0]
1
[1, 0, 1, 0, 1]
[2, 1, 1, 0, 1]
[2, 2, 0, 0, 1]
[1, 2, 0, 1, 1]
[0, 2, 0, 0, 1]
[2, 1, 0, 0, 1]
[0, 1, 0, 1, 1]
[1, 1, 1, 1, 1]
[1, 0, 0, 0, 1]
Total yes: 9 / 14
for feature 1
3 / 9
for feature 2
4 / 9
for feature 3
6 / 9
for feature 4
3 / 9
Total no: 5 / 14
for feature 1
```

```
2 / 5
for feature 2
2 / 5
for feature 3
1 / 5
for feature 4
3 / 5
probability of playing golf YES: 0.021164021164021163
probability of playing golf NO:0.006857142857142859
Probability of playing golf: 75.5287009063444%
```

Lab Cycle : I
Exp.No : 12

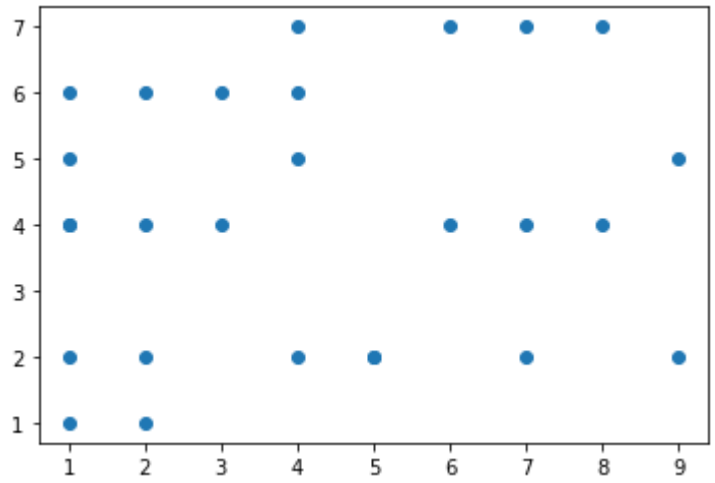
Aim: Write a program to implement k-means clustering algorithm

Solution:

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

x = [i[0] for i in data]
y = [i[1] for i in data]

import matplotlib.pyplot as plt

plt.scatter(x,y)
plt.show()

import math

def dist(center, point):
    d = 0.0
    for i in range(0,len(point)):
        d += (center[i]-point[i])**2
    return math.sqrt(d)

def assignCenters(centers, dataset):
```

```
    clusters = []
    for i in range(len(dataset)):
        distances = []
        for center in centers:
            distances.append(dist(center, dataset[i]))
        temp = [z for z, val in enumerate(distances) if val==min(distances)]
        clusters.append(temp[0])
    return clusters

def mean_center(k, dataset, clusters):
    nCenters = []
    for i in range(k):
        x = 0.0
        y = 0.0
        count = 0
        for j in range(len(clusters)):
            if(i == clusters[j]):
                x += dataset[j][0]
                y += dataset[j][1]
                count += 1
        x = x/count
        y = y/count
        nCenters.append([x,y])
    return nCenters

print("enter k")
k = int(input())
centers = []
for i in range(k):
    print("enter center "+str(i))
    temp = [int(x) for x in input().split()]
    centers.append(temp)

print("Initial centers: ")
print(centers)
print("Initial clusters: ")
clusters = assignCenters(centers, data)
for i in range(k):
    print("cluster "+str(i))
    for j in range(len(clusters)):
        if(i == clusters[j]):
            print(data[j],end=' ')
    print()
print()
for itr in range(10):
    print("Iteration "+str(itr))
    centers = mean_center(k,data,clusters)
    print("Updated centers: ")
    print(centers)
    clusters = assignCenters(centers, data)
    print("Updated clusters: ")
    for i in range(k):
        print("cluster "+str(i))
        for j in range(len(clusters)):
            if(i == clusters[j]):
                print(data[j],end=' ')
        print()
```

```
    print()
```




```
enter k
2
enter center 0
6 4
enter center 1
9 4
Initial centers:
[[6, 4], [9, 4]]
Initial clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [6, 7] [4, 2] [6, 4] [4, 5] [1, 6] [4, 7]
[3, 6] [1, 1] [7, 2] [2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [7, 7] [7, 4] [3, 4]
[1, 4]
cluster 1
[9, 5] [9, 2] [8, 4] [8, 7]

Iteration 0
Updated centers:
[[3.4347826086956523, 4.043478260869565], [8.5, 4.5]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 1
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 2
Updated centers:
```

```
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 3
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 4
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 5
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 6
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 7
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
```

```
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 8
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]

Iteration 9
Updated centers:
[[2.5555555555555554, 3.8333333333333335], [7.444444444444445,
4.666666666666667]]
Updated clusters:
cluster 0
[5, 2] [2, 4] [4, 6] [5, 2] [1, 5] [4, 2] [4, 5] [1, 6] [4, 7] [3, 6] [1, 1]
[2, 2] [2, 1] [1, 2] [1, 4] [2, 6] [3, 4] [1, 4]
cluster 1
[9, 5] [6, 7] [6, 4] [9, 2] [8, 4] [8, 7] [7, 2] [7, 7] [7, 4]
```

Lab Cycle : II
Exp.No : 1

Aim: Study and configure Hadoop for big data.

Step 1 — Installing Java

To get started, we'll update our package list:

- sudo apt-get update

Next, we'll install OpenJDK, the default Java Development Kit on Ubuntu 16.04.

- sudo apt-get install default-jdk

Once the installation is complete, let's check the version.

- java -version

Step 2 — Installing Hadoop

On the server, we'll use wget to fetch it:

- wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz

Again, we'll right-click to copy the file location, then use wget to transfer the file:

- wget https://dist.apache.org/repos/dist/release/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz.mds

Then run the verification:

- shasum -a 256 hadoop-2.7.3.tar.gz

Compare this value with the SHA-256 value in the .mds file:

- cat hadoop-2.7.3.tar.gz.mds
- tar -xzvf hadoop-2.7.3.tar.gz

Finally, we'll move the extracted files into /usr/local, the appropriate place for locally installed software. Change the version number, if needed, to match the version you downloaded.

- sudo mv hadoop-2.7.3 /usr/local/hadoop

With the software in place, we're ready to configure its environment.

Step 3 — Configuring Hadoop's Java Home

To find the default Java path

- readlink -f /usr/bin/java | sed "s:bin/java::"

To begin, open hadoop-env.sh:

- sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Then, choose one of the following options:

Option 1: Set a Static Value

```
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
. . .
```

Option 2: Use Readlink to Set the Value Dynamically

```
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
```

Step 4 — Running Hadoop

Now we should be able to run Hadoop:

- $ /usr/local/hadoop/bin/hadoop
- $ mkdir ~/input
- $ cp /usr/local/hadoop/etc/hadoop/*.xml ~/input

Lab Cycle : II
Exp.No : 2

Aim: Hadoop commands

Using the command line interface

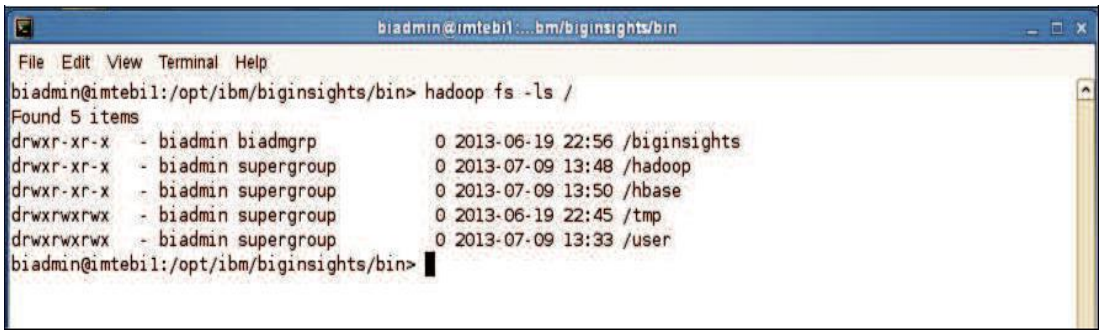
In this part, we will explore some basic HDFS commands. All HDFS commands start with ***hadoop*** followed by ***dfs*** (distributed file system) or ***fs*** (file system) followed by a dash, and the command. Many HDFS commands are similar to UNIX commands. For details, refer to the *Hadoop Command Guide* and *Hadoop FS Shell Guide*.

We will start with the **hadoop fs –ls** command which returns the list of files and directories with permission information.

Ensure the Hadoop components are all started, and from the same Gnome terminal window as before (and logged on as *biadmin*), follow these instructions:

1. **List the contents of the root directory**

   hadoop fs -ls /

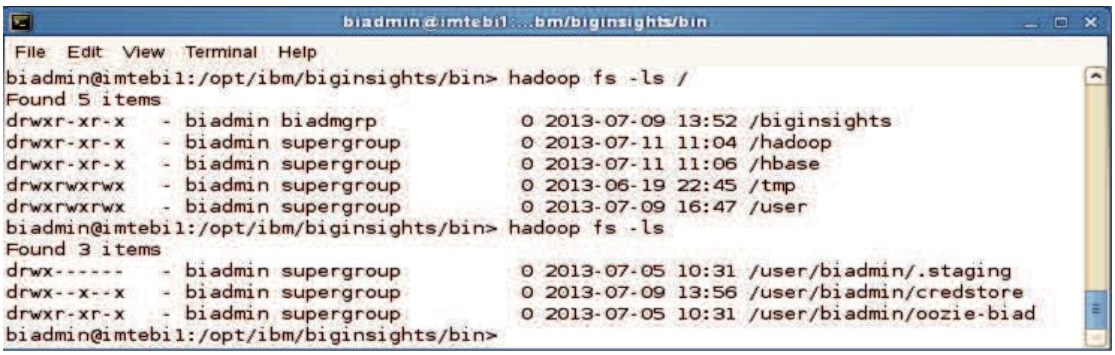


2. **To list the contents of the /user/biadmin directory, execute:**

   hadoop fs –ls

   (or)

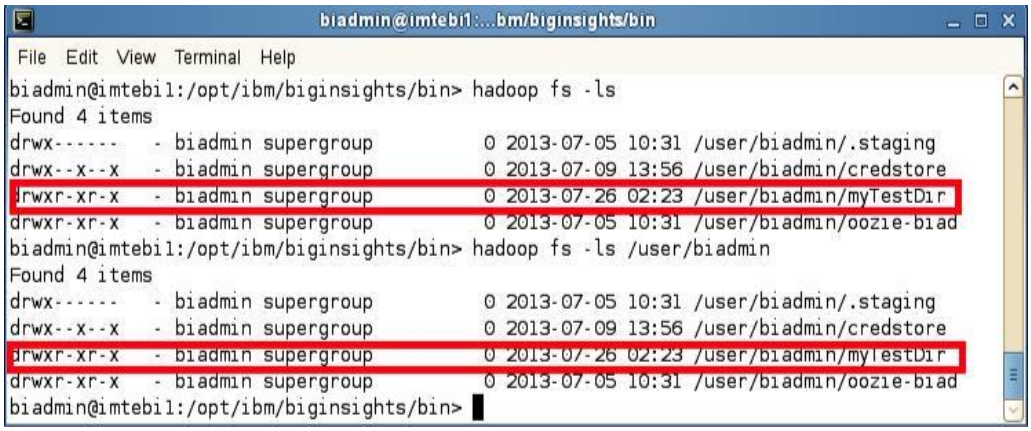
   hadoop fs -ls /user/biadmin



3. **To create the directory *myTestDir* you can issue the following command:**
   hadoop fs -mkdir myTestDir

   Where was this directory created? As mentioned in the previous step, any relative paths will be using the user's home directory.

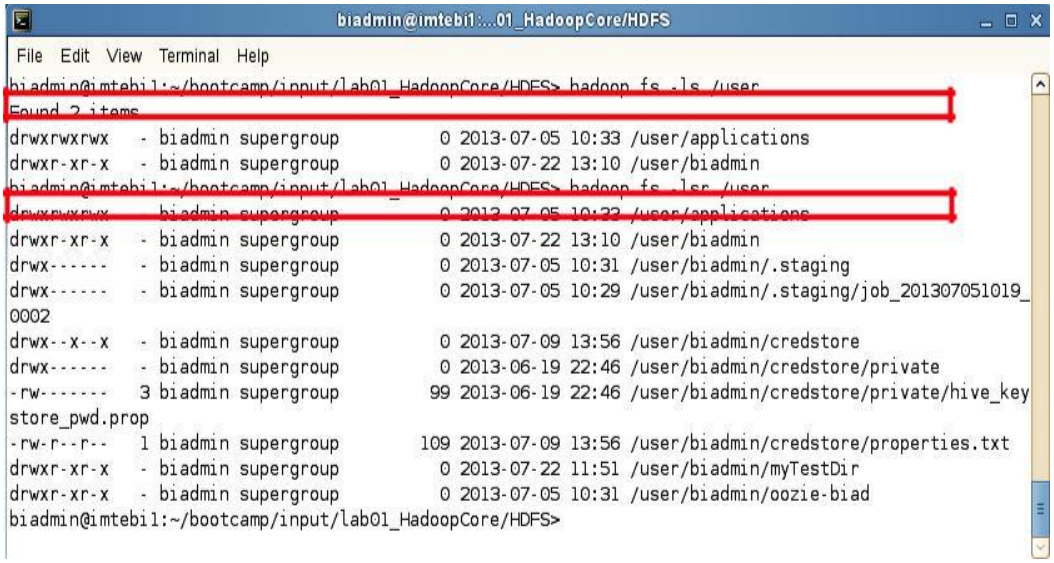
4. **Issue the ls command again to see the subdirectory myTestDir:**
   hadoop fs –ls (or) hadoop fs -ls /user/biadmin



To use HDFS commands recursively generally you add an "r" to the HDFS command (In the Linux shell this is generally done with the "-R" argument).

5. **For example, to do a recursive listing we'll use the –lsr command rather than just –ls, like the examples below:**

   hadoop fs -ls /user
   hadoop fs -lsr /user



6. **You can pipe (using the | character) any HDFS command to be used with the Linux shell. For example, you can easily use *grep* with HDFS by doing the following:**

   hadoop fs -mkdir /user/biadmin/myTestDir2

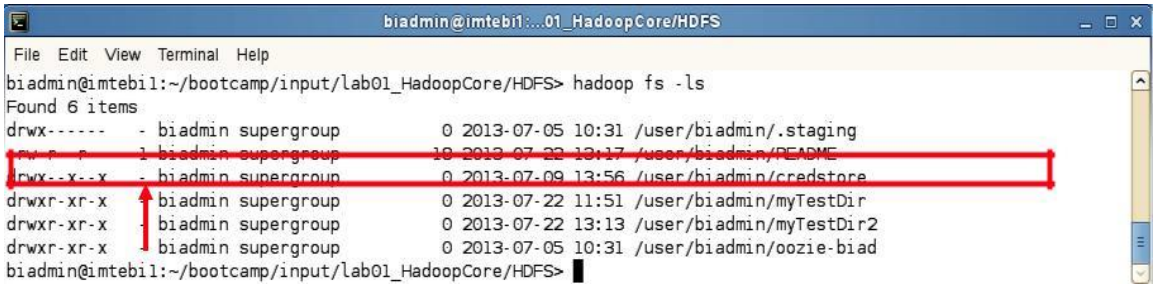
   hadoop fs -ls /user/biadmin | grep Test

As you can see the grep command only returned the lines which had test in them (thus removing the "Found x items" line and the .staging and oozie-biad directories from the listing

7. **To move files between your regular Linux filesystem and HDFS you can use the put and get commands. For example, move the text file README to the hadoop filesystem.**

   hadoop fs -put /home/biadmin/bootcamp/input/lab01_HadoopCore/HDFS/README

   README
   hadoop fs -ls /user/biadmin



You should now see a new file called /user/biadmin/README listed as shown above. Note there is a '1' highlighted in the figure. This represents the replication factor. By default, the replication factor in a BigInsights cluster is 3, but since this laboratory environment only has one node, the replication factor is 1.

8. **In order to view the contents of this file use the –cat command as follows:**

   hadoop fs -cat README

You should see the output of the README file (that is stored in HDFS). We can also use the linux diff command to see if the file we put on HDFS is actually the same as the original on the local filesystem.

9. Execute the commands below to use the diff command:

cd /home/biadmin/bootcamp/input/lab01_HadoopCore/HDFS/
diff <(hadoop fs -cat README) README

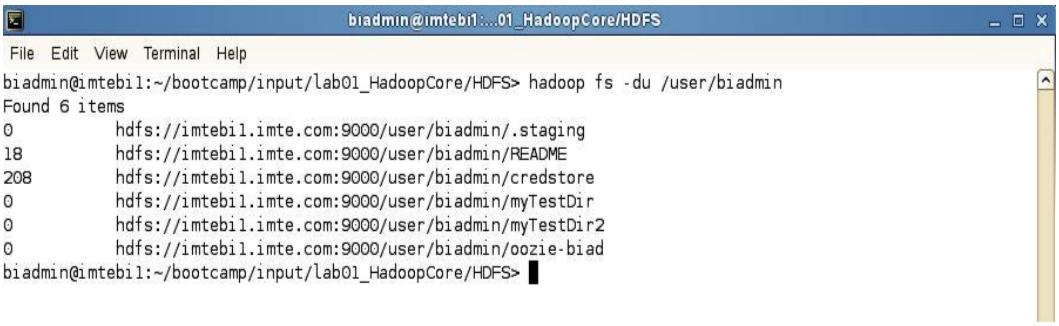
Since the diff command produces no output we know that the files are the same (the diff command prints all the lines in the files that differ).

To find the size of files you need to use the –du or –dus commands. Keep in mind that these commands return the file size in bytes.

10. To find the size of the README file use the following command:

hadoop fs -du README

11. To find the size of all files individually in the /user/biadmin directory use the following command:
hadoop fs -du /user/biadmin



12. To find the size of all files in total of the /user/biadmin directory use the following command:

hadoop fs -dus /user/biadmin

13. If you would like to get more information about hadoop fs commands, invoke –help as follows:

hadoop fs -help

14. For specific help on a command, add the command name after help. For example, to get help on the dus command you'd do the following:

hadoop fs -help dus

Lab Cycle : II
Exp.No : 3

Aim: Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm

Now that we've seen how the FileSystem (fs) shell can be used to execute Hadoop commands to interact with HDFS, the same fs shell can be used to launch MapReduce jobs. In this section, we will walk through the steps required to run a MapReduce program. The source code for a MapReduce program is contained in a compiled .jar file. Hadoop will load the JAR into HDFS and distribute it to the data nodes, where the individual tasks of the MapReduce job will be executed. Hadoop ships with some example MapReduce programs to run. One of these is a distributed WordCount program which reads text files and counts how often words occur.

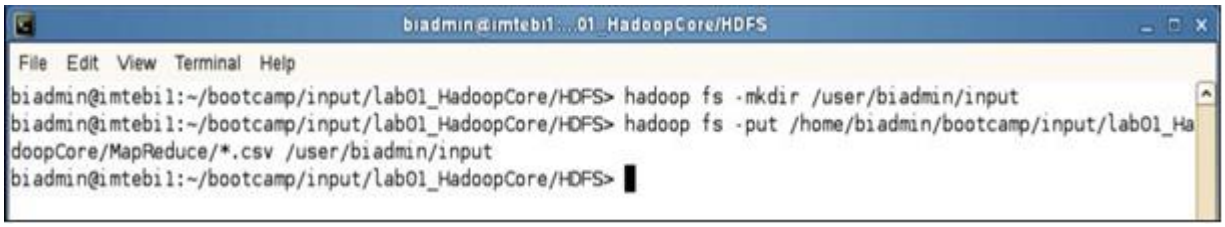
Running the WordCount program

First we need to copy the data files from the local file system to HDFS.

Step 1: Execute the commands below to copy the input files into HDFS.

hadoop fs -mkdir /user/biadmin/input

hadoop fs -put /home/biadmin/bootcamp/input/lab01_HadoopCore/MapReduce/*.csv /user/biadmin/input



Copy input files into HDFS

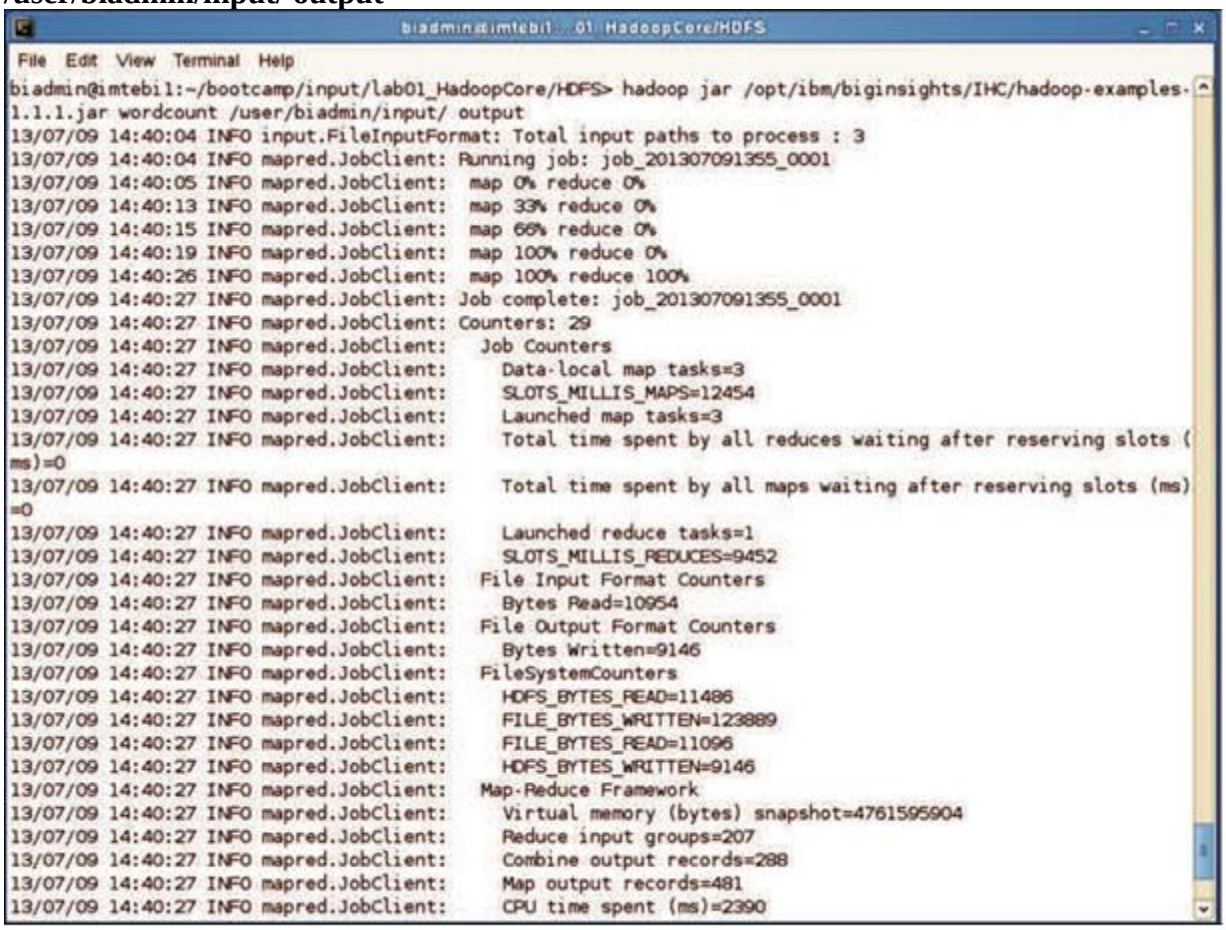
Step 2: Review the files have been copied with the following command:

hadoop fs -ls input

List copied files into HDFS

Step 3: Now we can run the wordcount job with the command below, where "/user/biadmin/input/" is where the input files are, and "output" is the directory where the output of the job will be stored. The "output" directory will be created automatically when executing the command below.

hadoop jar /opt/ibm/biginsights/IHC/hadoop-examples-1.1.1.jar wordcount /user/biadmin/input/ output

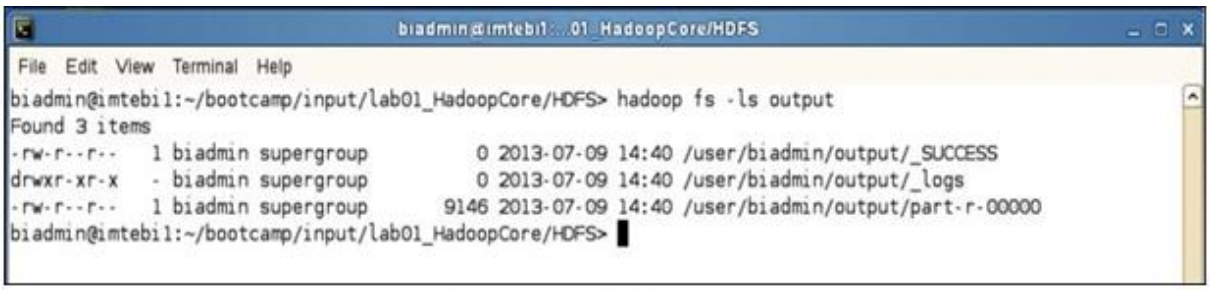


WordCount MapReduce job running

Step 4: Now review the output of step 3:

In this case, the output was not split into multiple files.

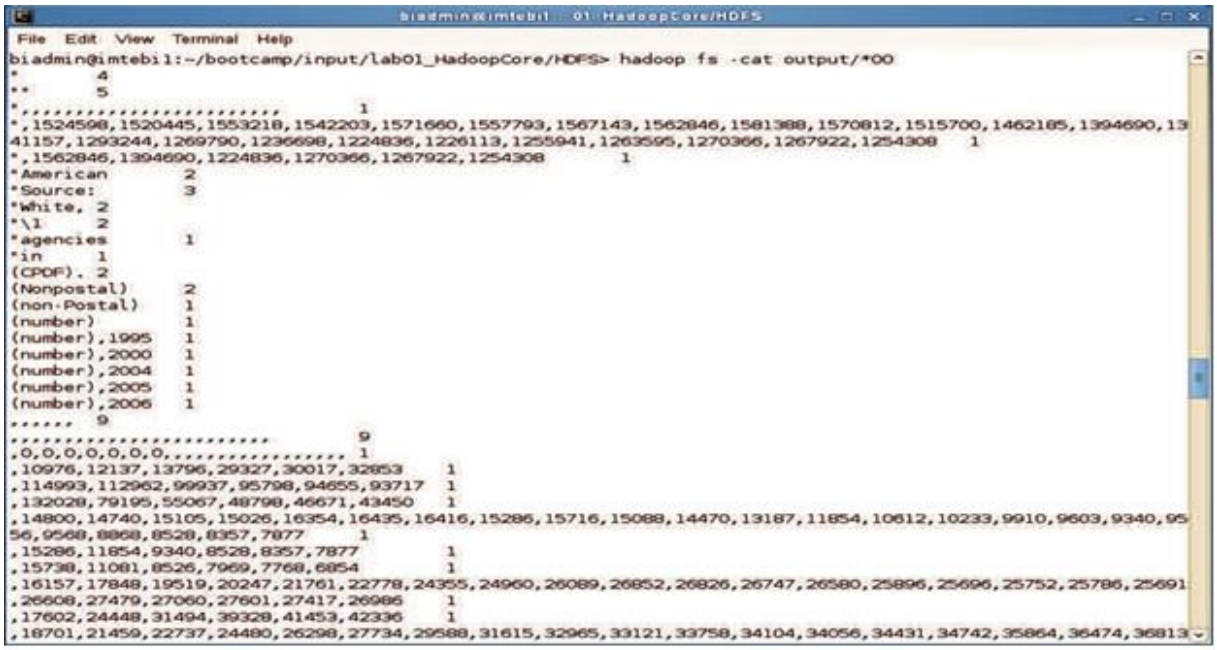
hadoop fs -ls output



MapReduce result files

Step 5: To view the contents of the part-r-0000 file issue the command below:

hadoop fs -cat output/*00



MapReduce output

Lab Cycle : II
Exp.No : 4

Aim: Implement Matrix Multiplication with Hadoop Map Reduce

```
public class MatrixMultiply {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");
            System.exit(2);
        }
        Configuration conf = new Configuration();
        // M is an m-by-n matrix; N is an n-by-p matrix.
        conf.set("m", "1000");
        conf.set("n", "100");
        conf.set("p", "1000");
        @SuppressWarnings("deprecation")
        Job job = new Job(conf, "MatrixMultiply");
        job.setJarByClass(MatrixMultiply.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```