LIVE TO LEARN & LEARN TO SHARE

MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

LABORATORY MANUAL & RECORD

Name:...

Roll No:................Branch:...

Year:...................Sem:...

MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

Certificate

Certified that this is the Bonafide Record of the Work Done by

Mr./Ms...Roll.No..................of

B.Tech............year......................... Semester for Academic year........................

in...Laboratory.

Date: Faculty Incharge HOD

Internal Examiner External Examiner

INDEX

S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

MACHINE LEARNING

LAB MANUAL

B.TECH

(II YEAR – II SEM)

(2022-23)

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

(AIML)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, andthe cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics andresponsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a memberor leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Objectives:

- To introduce the basic concepts and techniques of Machine Learning and the need of Machine Learning techniques in real-world problems.
- To provide understanding of various Machine Learning algorithms and the way to evaluate performance of the Machine Learning algorithms.
- To apply Machine Learning to learn, predict and classify the real-world problems in the Supervised Learning paradigms as well as discover the Unsupervised Learning paradigms of Machine Learning.
- To understand, learn and design simple Artificial Neural Networks of Supervised Learning for the selected problems.
- To understand the concept of Reinforcement Learning and Ensemble Methods.
- To inculcate in students professional and ethical attitude, multidisciplinary approach and an ability to relate real-world issues and provide a cost effective solution to it by developing ML applications.
- To provide student with an academic environment aware of excellence, written ethical codes and guidelines and lifelong learning needed for a successful professional career.

Lab Outcomes:

Upon successful completion of this course, the students will be able to:

- Understand the basic concepts and techniques of Machine Learning and the need of Machine Learning techniques in real-world problems.
- Understand various Machine Learning algorithms and the way to evaluate performance of the Machine Learning algorithms
- Apply Machine Learning to learn, predict and classify the real-world problems in the Supervised Learning paradigms as well as discover the Unsupervised Learning paradigms of Machine Learning.
- Understand, learn and design Artificial Neural Networks of Supervised Learning for the selected problems.
- Understand the concept of Reinforcement Learning and Ensemble Methods.

Introduction about lab

System configurations are as follows:

- **Hardware / Software's installed: I**ntel® CORE™ i3-3240 CPU@3.40GHZRAM: 4GB / Anaconda Navigator or Python and Jupyter Notebook or Google Colab.
- **Packages required to run the programs:** Math, Scipy, Numpy, Matplotlib, Pandas, Sklearn, Tensorflow, Keras etc.
- Systems are provided for students in the **1:1 ratio.**
- Systems are assigned numbers and same system is allotted for students when they do the lab.
- All Systems are configured in LINUX, it is open source and students can use any different programming environments through package installation.

Guidelines to students

A. Standard operating procedure

a) Explanation on today's experiment by the concerned faculty using PPT covering the following aspects:

1) Name of the experiment
2) Aim
3) Software/Hardware requirements
4) Writing the python programs by the students
5) Commands for executing programs

Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

a) Name of the experiment
b) Aim
c) Writing the program
d) Viva-Voce Questions and Answers
e) Errors observed (if any) during compilation/execution

Signature of the Faculty

B. Guide Lines to Students in Lab

Disciplinary to be maintained by the students in the Lab

- Students are required to carry their lab observation book and record book withcompleted experiments while entering the lab.
- Students must use the equipment with care. Any damage is caused student ispunishable.
- Students are not allowed to use their cell phones/pen drives/ CDs in labs.
- Students need to maintain proper dress code along with ID Card
- Students are supposed to occupy the computers allotted to them and are notsupposed to talk or make noise in the lab.
- Students, after completion of each experiment they need to be updated in observation notes and same to be updated in the record.
- Lab records need to be submitted after completion of experiment and get itcorrected with the concerned lab faculty.
- If a student is absent for any lab, they need to be completed the same experiment in the free time before attending next lab.

Steps to perform experiments in the lab by the student

Step1: Students have to write the date, aim and forthat experiment in the observation book.

Step2: Students have to listen and understand the experiment explained by the faculty and note down the important points in the observation book.

Step3: Students need to write procedure/algorithm in the observation book.

Step4: Analyze and Develop/implement the logic of the program by the student in respective platform

Step5: After approval of logic of the experiment by the faculty then the experiment has to be executed on the system.

Step6: After successful execution the results are to be shown to the faculty andnoted the same in the observation book.

Step7: Students need to attend the Viva-Voce on that experiment and write the samein the observation book.

Step8: Update the completed experiment in the record and submit to the concernedfaculty in-charge.

Instructions to maintain the record

- Before start of the first lab they have to buy the record and bring the record to thelab.
- Regularly (Weekly) update the record after completion of the experiment and get itcorrected with concerned lab in-charge for continuous evaluation. In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks (5M) will be deducted.

Awarding the marks for day to day evaluation

Total marks for day to day evaluation is 15 Marks as per Autonomous (JNTUH). These 15 Marks are distributed as:

Regularity	3 Marks
Program written	3 Marks
Execution & Result	3 Marks
Viva-Voce	3 Marks
Dress Code	3 Marks

Allocation of Marks for Lab Internal

Total marks for lab internal are 30 Marks as per Autonomous (JNTUH.)

These 30 Marks are distributed as:

Average of day to day evaluation marks: 15 Marks

Lab Mid exam: 10 Marks

VIVA & Observation: 5 Marks

Allocation of Marks for Lab External

Total marks for lab Internal and External are 70 Marks as per Autonomous / (JNTUH).

These 70 External Lab Marks are distributed as:

Program Written	30 Marks
Program Execution and Result	20 Marks
Viva-Voce	10 Marks
Record	10 Marks

C. General laboratory instructions

1. Students are advised to come to the laboratory at least 5 minutes before (to thestarting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to thelab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computersystem allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the labobservation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, mustmaintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

INDEX

Machine Learning Programs Using Python

Week-1:

1. Implementation of Python Basic Libraries such as Math, Numpy and Scipy

Theory/Description:

- **Python Libraries**

There are a lot of reasons why Python is popular among developers and one of them is that it has an amazingly large collection of libraries that users can work with. In this Python Library, we will discuss Python Standard library and different libraries offered by Python Programming Language: scipy, numpy, etc.

We know that a module is a file with some Python code, and a package is a directory for sub packages and modules. A Python library is a reusable chunk of code that you may want to includein your programs/ projects. Here, a library loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like npm.

- **Python Standard Library**

The Python Standard Library is a collection of script modules accessible to a Python program to simplify the programming process and removing the need to rewrite commonly used commands. They can be used by 'calling/importing' them at the beginning of a script. A list of the Standard Library modules that are most important

- time
- sys
- csv
- math
- random
- pip
- os
- statistics
- tkinter
- socket

To display a list of all available modules, use the following command in the Python console:

```
>>> help('modules')
```

- **List of important Python Libraries**

- Python Libraries for Data Collection
  - Beautiful Soup
  - Scrapy
  - Selenium
- Python Libraries for Data Cleaning and Manipulation
  - Pandas
  - PyOD
  - NumPy
  - Scipy
  - Spacy
- Python Libraries for Data Visualization
  - Matplotlib
  - Seaborn
  - Bokeh
- Python Libraries for Modeling

- Scikit-learn
- TensorFlow
- Keras
- PyTorch

a) Implementation of Python Basic Libraries such as Math, Numpy and Scipy

- **Python Math Library**

The math module is a standard module in Python and is always available. To use mathematical functions under this module, you have to import the module using import math. It gives access tothe underlying C library functions. This module does not support complex datatypes. The cmath module is the complex counterpart.

List of Functions in Python Math Module	
Function	**Description**
ceil(x)	Returns the smallest integer greater than or equal to x.
copysign(x, y)	Returns x with the sign of y
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
fmod(x, y)	Returns the remainder when x is divided by y
frexp(x)	Returns the mantissa and exponent of x as the pair (m, e)
fsum(iterable)	Returns an accurate floating point sum of values in the iterable
isfinite(x)	Returns True if x is neither an infinity nor a NaN (Not a Number)
isinf(x)	Returns True if x is a positive or negative infinity
isnan(x)	Returns True if x is a NaN
ldexp(x, i)	Returns x * (2**i)
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
expm1(x)	Returns e**x - 1

Program-1

```
In [15]: # Import math Library
import math

# Round a number upward to its nearest integer
print(math.ceil(1.4))
print(math.ceil(5.3))
print(math.ceil(-5.3))
print(math.ceil(22.6))
print(math.ceil(10.0))
```

```
2
6
-5
23
10
```

Program-2

In [16]:
```
#Import math Library
import math

#Return factorial of a number
print(math.factorial(9))
print(math.factorial(6))
print(math.factorial(12))
```

```
362880
720
479001600
```

Program-3

In [17]:
```
# Import math Library
import math

# Round numbers down to the nearest integer
print(math.floor(0.6))
print(math.floor(1.4))
print(math.floor(5.3))
print(math.floor(-5.3))
print(math.floor(22.6))
print(math.floor(10.0))
```

```
0
1
5
-6
22
10
```

Program-4

In [18]:
```
#Import math Library
import math

#find the  the greatest common divisor of the two integers
print (math.gcd(3, 6))
print (math.gcd(6, 12))
print (math.gcd(12, 36))
print (math.gcd(-12, -36))
print (math.gcd(5, 12))
print (math.gcd(10, 0))
print (math.gcd(0, 34))
print (math.gcd(0, 0))
```

```
3
6
12
12
1
10
34
0
```

Program-5

- We also have a boolean function isNaN() which returns true if the given argument is a NaN and returns false otherwise. We can also take a value and convert it to float to check whether it is NaN.
- A missing value is denoted as NaN (Stands for Not a Number eg. 0/0).
- Inf: 1/0 is one example of Inf. We can define the negative infinity as -inf and the positive infinity as inf in Python.

In [19]:
```
# Import math Library
import math

# Check whether some values are NaN or not
print (math.isnan (56))
print (math.isnan (-45.34))
print (math.isnan (+45.34))
print (math.isnan (math.inf))
print (math.isnan (float("nan")))
print (math.isnan (float("inf")))
print (math.isnan (float("-inf")))
print (math.isnan (math.nan))
```

```
False
False
False
False
True
False
False
True
```

Program-6

In [25]:
```
# Import math Library
import math

# Print the square root of different numbers
print (math.sqrt(10))
print (math.sqrt (12))
print (math.sqrt (68))
print (math.sqrt (100))

# Round square root downward to the nearest integer
print (math.isqrt(10))
print (math.isqrt (12))
print (math.isqrt (68))
print (math.isqrt (100))
```

```
3.1622776601683795
3.4641016151377544
8.246211251235321
10.0
3
3
8
10
```

- **Python Numpy Library**

NumPy is an open source library available in Python that aids in mathematical, scientific, engineering,and data science programming. NumPy is an incredible library to perform mathematical and statistical operations. It works perfectly well for multi-dimensional arrays and matrices multiplication

For any scientific project, NumPy is the tool to know. It has been built to work with the N-dimensional array, linear algebra, random number, Fourier transform, etc. It can be integrated to C/C++ and Fortran.

NumPy is a programming language that deals with multi-dimensional arrays and matrices. On top of the arrays and matrices, NumPy supports a large number of mathematical operations.

NumPy is memory efficient, meaning it can handle the vast amount of data more accessible than any other library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, TensorFlow and Scikitlearn use NumPy array to compute the matrix multiplication in the back end.

- **Arrays in NumPy:** NumPy's main object is the homogeneous multidimensional array.

- It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- In NumPy dimensions are called axes. The number of axes is rank.
- NumPy's array class is called **ndarray**. It is also known by the alias **array**.

We use python numpy array instead of a list because of the below three reasons:
1. Less Memory
2. Fast
3. Convenient

- **Numpy Functions**

Numpy arrays carry attributes around with them. The most important ones are:
ndim: The number of axes or rank of the array. ndim returns an integer that tells us how many dimensions the array have.
shape: A tuple containing the length in each dimension size: The total number of elements

Program-1

```
In [27]: import numpy        #DEPT OF SoCSE4
         x = numpy.array([[1,2,3], [4,5,6], [7,8,9]]) # 3x3 matrix
         print(x.ndim) # Prints 2
         print(x.shape) # Prints (3L, 3L)
         print(x.size) # Prints 9

         2
         (3, 3)
         9
```

Can be used just like Python lists
x[1] will access the second row
x[-1] will access the last row
x[2]?
x[3]?
x[0]?

Program-2

Arithmetic operations apply element wise

```
In [32]: a = numpy.array( [20,30,40,50,60] )
         b = numpy.arange( 5 )
         c = a-b     #DEPT OF SoCSE4
         #c => array([20, 29, 38, 47])
         c

Out[32]: array([20, 29, 38, 47, 56])
```

- **Built-in Methods**

Many standard numerical functions are available as methods out of the box:

Program-3

```
In [34]: x = numpy.array([1,2,3,4,5])
         avg = x.mean()    #DEPT OF SoCSE4
         sum = x.sum()
         sx = numpy.sin(x)
         sx

Out[34]: array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.95892427])
```

Exercise: Apply x.sort() and display the sorted array as output.

- **Python Scipy Library**

SciPy is an Open Source Python-based library, which is used in mathematics, scientific computing, Engineering, and technical computing. SciPy also pronounced as "Sigh Pi."

- SciPy contains varieties of sub packages which help to solve the most common issuerelated to Scientific Computation.
- SciPy is the most used Scientific library only second to GNU Scientific Library forC/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

Numpy VS SciPy

Numpy:

1. Numpy is written in C and used for mathematical or numeric calculation.
2. It is faster than other Python Libraries
3. Numpy is the most useful library for Data Science to perform basic calculations.
4. Numpy contains nothing but array data type which performs the most basic operation like sorting, shaping, indexing, etc.

SciPy:

1. SciPy is built in top of the NumPy
2. SciPy is a fully-featured version of Linear Algebra while Numpy contains only a few features.
3. Most new Data Science features are available in Scipy rather than Numpy.

Linear Algebra with SciPy

1. Linear Algebra of SciPy is an implementation of BLAS and ATLAS LAPACK libraries.
2. Performance of Linear Algebra is very fast compared to BLAS and LAPACK.
3. Linear algebra routine accepts two-dimensional array object and output is also a two-dimensional array.

Now let's do some test with **scipy.linalg,**

Calculating **determinant** of a two-dimensional matrix,

Program-1

```
from scipy import linalg
import numpy as np #define square matrix
two_d_array = np.array([ [4,5], [3,2] ]) #pass values to det() function
linalg.det( two_d_array )
```

```
-7.0
```

Eigenvalues and Eigenvector – scipy.linalg.eig()

- The most common problem in linear algebra is eigenvalues and eigenvector which can beeasily solved using **eig()** function.
- Now lets we find the Eigenvalue of (**X**) and correspond eigenvector of a two-dimensional square matrix.

Program-2

```
from scipy import linalg
import numpy as np
#define two dimensional array
arr = np.array([[5,4],[6,3]]) #pass value into function
eg_val, eg_vect = linalg.eig(arr) #get eigenvalues
print(eg_val) #get eigenvectors print(eg_vect)
```

```
[ 9.+0.j -1.+0.j]
```

Exercise programs:

1. Consider a list datatype (1D) then reshape it into 2D, 3D matrix using numpy
2. Generate random matrices using numpy
3. Find the determinant of a matrix using scipy
4. Find eigenvalue and eigenvector of a matrix using scipy

Week-2:

Implementation of Python Libraries for ML application such as Pandas and Matplotlib.

- **Pandas Library**

The primary two components of pandas are the Series and DataFrame.
A Series is essentially a column, and a DataFrame is a multi-dimensional table made up of a collection of Series.
DataFrames and Series are quite similar in that many operations that you can do with oneyou can do with the other, such as filling in null values and calculating the mean.

 Reading data from CSVs

With CSV files all you need is a single line to load in the data:
df = pd.read_csv('purchases.csv')df

Let's load in the IMDB movies dataset to begin:
movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="Title")
We're loading this dataset from a CSV and designating the movie titles to be our index.

 Viewing your data
The first thing to do when opening a new dataset is print out a few rows to keep as a visual reference. We accomplish this with .head():
movies_df.head()

Another fast and useful attribute is .shape, which outputs just a tuple of (rows, columns):
movies_df.shape
Note that .shape has no parentheses and is a simple tuple of format (rows, columns). So we have 1000 rows and 11 columns in our movies DataFrame.
You'll be going to .shape a lot when cleaning and transforming data. For example, you might filtersome rows based on some criteria and then want to know quickly how many rows were removed.

Program-1

```
import pandas as pd
S = pd.Series([11, 28, 72, 3, 5, 8])
S
```

```
0    11
1    28
2    72
3     3
4     5
5     8
dtype: int64
```

We haven't defined an index in our example, but we see two columns in our output: The right column contains our data, whereas the left column contains the index. Pandas created a default index starting with 0 going to 5, which is the length of the data minus 1.

dtype('int64'): The type int64 tells us that Python is storing each value within this column as a 64 bit integer

Program-2

We can directly access the index and the values of our Series S:

```
print(S.index)
print(S.values)
```

```
RangeIndex(start=0, stop=6, step=1)
[11 28 72  3  5  8]
```

Program-3

If we compare this to creating an array in numpy, we will find lots of similarities:

```
import numpy as np
X = np.array([11, 28, 72, 3, 5, 8])
print(X)
print(S.values)
# both are the same type:
print(type(S.values), type(X))
```

```
[11 28 72  3  5  8]
[11 28 72  3  5  8]
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

So far our Series have not been very different to ndarrays of Numpy. This changes, as soon as we start defining Series objects with individual indices:

Program-4

```
fruits = ['apples', 'oranges', 'cherries', 'pears']
quantities = [20, 33, 52, 10]
S = pd.Series(quantities, index=fruits)
S
```

```
apples      20
oranges     33
cherries    52
pears       10
dtype: int64
```

Program-5

A big advantage to NumPy arrays is obvious from the previous example: We can use arbitrary indices. If we add two series with the same indices, we get a new series with the same index and the corresponding values will be added:

```
fruits = ['apples', 'oranges', 'cherries', 'pears']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits)
print(S + S2)
print("sum of S: ", sum(S))
```

OUTPUT:

```
apples      37
oranges     46
cherries    83
pears       42
dtype: int64
sum of S:  115
```

Program-6

The indices do not have to be the same for the Series addition. The index will be the "union" of both indices. If an index doesn't occur in both Series, the value for this Series will be NaN:

```
fruits = ['peaches', 'oranges', 'cherries', 'pears']
fruits2 = ['raspberries', 'oranges', 'cherries', 'pears']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits2)
print(S + S2)
```

OUTPUT:

```
cherries       83.0
oranges        46.0
peaches         NaN
pears          42.0
raspberries     NaN
dtype: float64
```

Program-7

In principle, the indices can be completely different, as in the following example. We have two indices. One is the Turkish translation of the English fruit names:

```
fruits = ['apples', 'oranges', 'cherries', 'pears']

fruits_tr = ['elma', 'portakal', 'kiraz', 'armut']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits_tr)
print(S + S2)
```

OUTPUT:

```
apples     NaN
```

armut NaN
cherries NaN
elma NaN
kiraz NaN
oranges NaN
pears NaN
portakal NaN
dtype: float64

Program-8

Indexing

It's possible to access single values of a Series.

```
print (S ['apples'] )
```

OUTPUT:

20

- **Matplotlib Library**

Pyplot is a module of Matplotlib which provides simple functions to add plot elementslike lines, images, text, etc. to the current axes in the current figure.

 Make a simple plot

import matplotlib.pyplot as plt
import numpy as np

List of all the methods as they appeared.

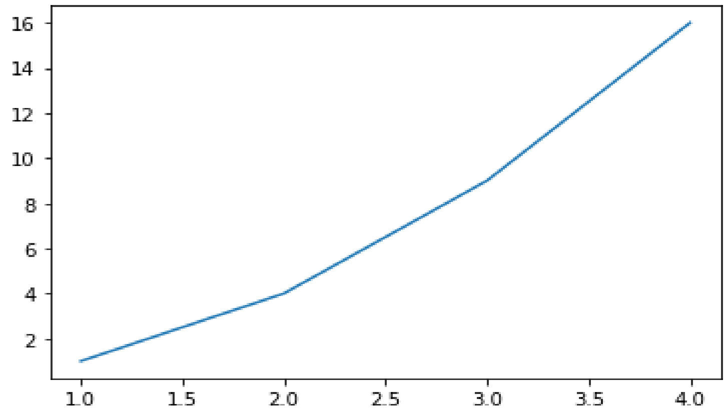
 plot(x-axis values, y-axis values) — plots a simple line graph with x-axis valuesagainst y-axis values
 show() — displays the graph
 title(―string‖) — set the title of the plot as specified by the string
 xlabel(―string‖) — set the label for x-axis as specified by the string
 ylabel(―string‖) — set the label for y-axis as specified by the string
 figure() — used to control a figure level attributes
 subplot(nrows, ncols, index) — Add a subplot to the current figure
 suptitle(―string‖) — It adds a common title to the figure specified by the string
 subplots(nrows, ncols, figsize) — a convenient way to create subplots, in a single call.It returns a tuple of a figure and number of axes.
 set_title(―string‖) — an axes level method used to set the title of subplots in a figure
 bar(categorical variables, values, color) — used to create vertical bar graphs
 barh(categorical variables, values, color) — used to create horizontal bar graphs
 legend(loc) — used to make legend of the graph
 xticks(index, categorical variables) — Get or set the current tick locations and labelsof the x-axis
 pie(value, categorical variables) — used to create a pie chart
 hist(values, number of bins) — used to create a histogram

- xlim(start value, end value) — used to set the limit of values of the x-axis
- ylim(start value, end value) — used to set the limit of values of the y-axis
- scatter(x-axis values, y-axis values) — plots a scatter plot with x-axis values againsty-axis values
- axes() — adds an axes to the current figure
- set_xlabel(―string‖) — axes level method used to set the x-label of the plot specified as a string
- set_ylabel(―string‖) — axes level method used to set the y-label of the plot specified as a string
- scatter3D(x-axis values, y-axis values) — plots a three-dimensional scatter plot withx-axis values against y-axis values
- plot3D(x-axis values, y-axis values) — plots a three-dimensional line graph with x-axis values against y-axis values

Here we import Matplotlib's Pyplot module and Numpy library as most of the data thatwe will be working with arrays only.

Program-1

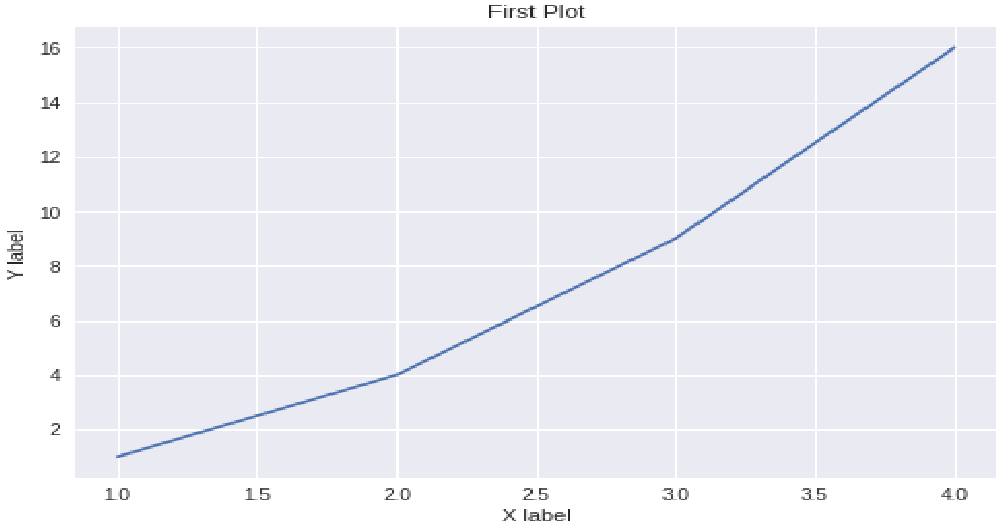
```
import matplotlib.pyplot as plt
import numpy as np
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



We pass two arrays as our input arguments to Pyplot's plot() method and use show() method to invoke the required plot. Here note that the first array appears on the x-axis andsecond array appears on the y-axis of the plot. Now that our first plot is ready, let us add the title, and name x-axis and y-axis using methods title(), xlabel() and ylabel() respectively.

Program-2

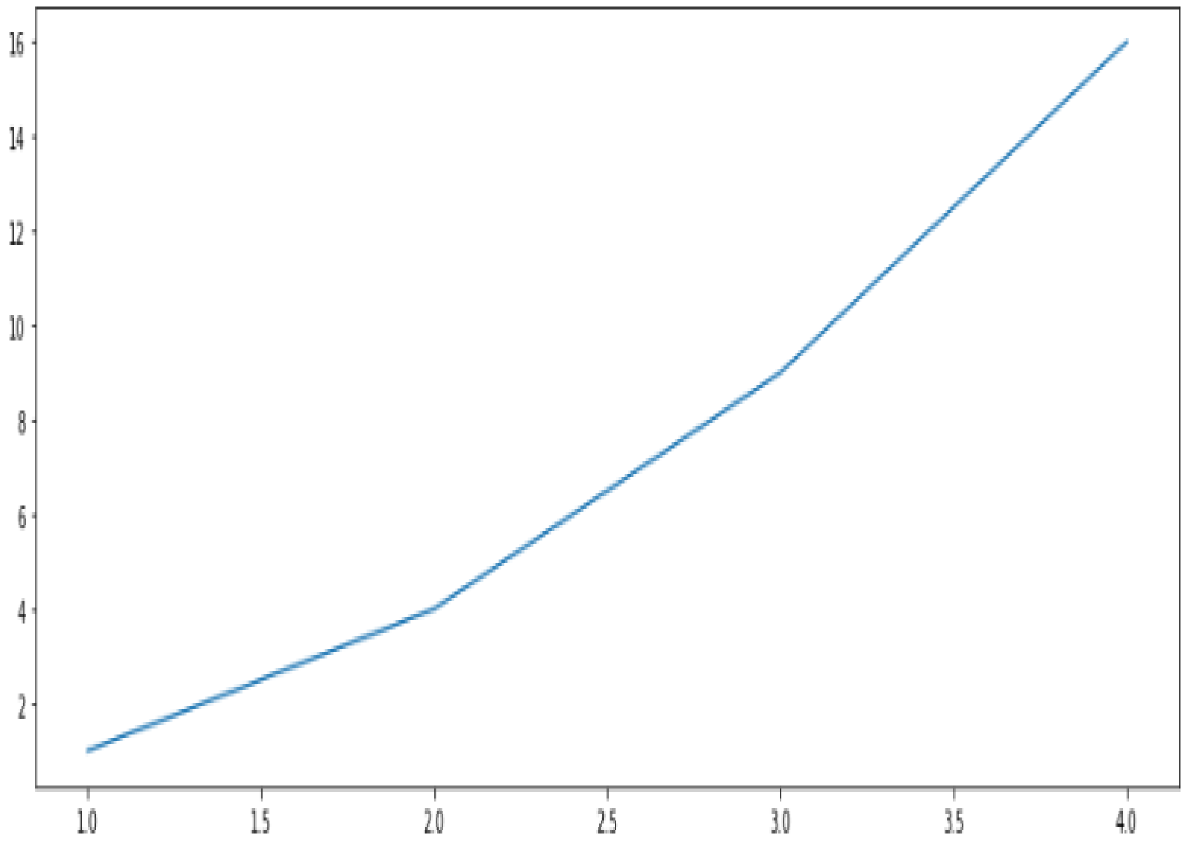
```
plt.plot([1,2,3,4],[1,4,9,16])
plt.title("First Plot")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```



Program-3

We can also specify the size of the figure using method figure()and passing the values as a tuple of the length of rows and columns to the argument figsize

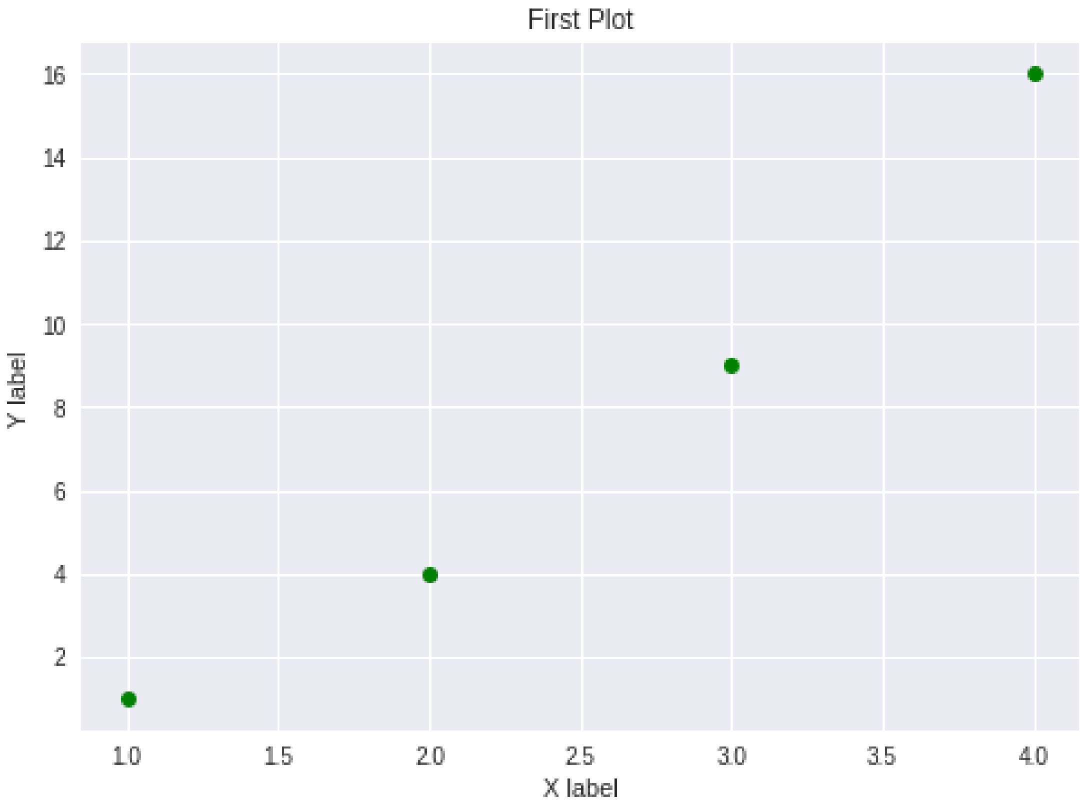
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(15,5))
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



Program-4

With every X and Y argument, you can also pass an optional third argument in the formof a string which indicates the colour and line type of the plot. The default format is **b-** which means a solid blue line. In the figure below we use **go** which means green circles.Likewise, we can make many such combinations to format our plot.

```
plt.plot([1,2,3,4],[1,4,9,16],"go")
plt.title("First Plot")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```



EXERCISE:

1. Write a python program to declare two series data and also add the index names. Use division operator to divide one series by another. In the output one of the series data must be NaN and another Inf.
2. Write a python program to consider some values as (x,y) co-ordinate values and plot the graph using a line graph. The color of the line graph should be red.

Week-3

1. Creation and loading different datasets in Python

Program-1

Method-I

```
# Import pandas package
import pandas as pd

# Assign data
data = {'Name': ['Jai', 'Princi', 'Gaurav',
                 'Anuj', 'Ravi', 'Natasha', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```

	Name	Age	Gender	Marks
0	Jai	17	M	90
1	Princi	17	F	76
2	Gaurav	18	M	NaN
3	Anuj	17	M	74
4	Ravi	18	M	65
5	Natasha	17	F	NaN
6	Riya	17	F	71

Program-2

Method-II:

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
print(boston_dataset.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None
```

Program-3 **Uploading csv file:**

Method-III:

```
import pandas as pd

df = pd.read_csv (r'E:\ml datasets\Machine-Learning-with-Python-master\Datasets\loan_data.csv')
print (df.head())
```

```
   credit.policy             purpose  int.rate  installment  log.annual.inc  \
0              1  debt_consolidation    0.1189       829.10       11.350407
1              1         credit_card    0.1071       228.22       11.082143
2              1  debt_consolidation    0.1357       366.86       10.373491
3              1  debt_consolidation    0.1008       162.34       11.350407
4              1         credit_card    0.1426       102.92       11.299732

     dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  \
0  19.48   737        5639.958333      28854        52.1               0
1  14.29   707        2760.000000      33623        76.7               0
2  11.63   682        4710.000000       3511        25.6               1
3   8.10   712        2699.958333      33667        73.2               1
4  14.97   667        4066.000000       4740        39.5               0

   delinq.2yrs  pub.rec  not.fully.paid
0            0        0               0
1            0        0               0
2            0        0               0
3            0        0               0
4            1        0               0
```

2. Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using Datasets

- **Python Statistics library**

This module provides functions for calculating mathematical statistics of numeric (Real-valued) data. The statistics module comes with very useful functions like: Mean, median, mode, standard deviation,and variance.

The four functions we'll use are common in statistics:

1. mean - average value
2. median - middle value
3. mode - most often value
4. standard deviation - spread of values

- **Averages and measures of central location**

These functions calculate an average or typical value from a population or

sample.mean()	Arithmetic mean (–average‖) of data.
harmonic_mean()	Harmonic mean of data.
median()	Median (middle value) of
data.median_low()	Low median of data.
median_high()	High median of data.
median_grouped()	Median, or 50th percentile, of grouped
data.mode()	Mode (most common value) of discrete

data.

- **Measures of spread**

These functions calculate a measure of how much the population or sample tends to deviate from the typical or average values.

pstdev()	Population standard deviation of data.
pvariance()	Population variance of data.
stdev()	Sample standard deviation of data.
variance()	Sample variance of data.

Program-1

```
# Import statistics Library
import statistics

# Calculate average values
print(statistics.mean([1, 3, 5, 7, 9, 11, 13]))
print(statistics.mean([1, 3, 5, 7, 9, 11]))
print(statistics.mean([-11, 5.5, -3.4, 7.1, -9, 22]))
```

```
7
6
1.8666666666666667
```

Program-2

```
# Import statistics Library
import statistics

# Calculate middle values
print(statistics.median([1, 3, 5, 7, 9, 11, 13]))
print(statistics.median([1, 3, 5, 7, 9, 11]))
print(statistics.median([-11, 5.5, -3.4, 7.1, -9, 22]))
```

```
7
6.0
1.05
```

Program-3

```
# Import statistics Library
import statistics

# Calculate the mode
print(statistics.mode([1, 3, 3, 3, 5, 7, 7, 9, 11]))
print(statistics.mode([1, 1, 3, -5, 7, -9, 11]))
print(statistics.mode(['red', 'green', 'blue', 'red']
```

```
3
1
red
```

Program-4

```
# Import statistics Library
import statistics

# Calculate the standard deviation from a sample of data
print(statistics.stdev([1, 3, 5, 7, 9, 11]))
print(statistics.stdev([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.stdev([-11, 5.5, -3.4, 7.1]))
print(statistics.stdev([1, 30, 50, 100]))
```

```
3.7416573867739413
0.6925797186365384
8.414471660973929
41.67633221226008
```

Program-5

```
# Import statistics Library
import statistics

# Calculate the variance from a sample of data
print(statistics.variance([1, 3, 5, 7, 9, 11]))
print(statistics.variance([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.variance([-11, 5.5, -3.4, 7.1]))
print(statistics.variance([1, 30, 50, 100]))
```

```
14
0.4796666666666667
70.80333333333334
1736.9166666666667
```

Exercise programs:

1. Load two standard ML datasets by using Method II and III shown in the above examples.
2. Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using the first 5 or more rows from Iris dataset.

Week-4

Demonstrate various data pre-processing techniques for a given dataset. Write a python program to compute

a) **Reshaping the data,**
b) **Filtering the data,**
c) **Merging the data**
d) **Handling the missing values in datasets**
e) **Feature Normalization: Min-max normalization**

Program-1

Reshaping the data:

Method-I

```
import numpy as np
array1 = np.arange(8)
print("Original array : \n", array1)

# shape array with 2 rows and 4 columns
array2 = np.arange(8).reshape(2,4)
print("\narray reshaped with 2 rows and 4 columns : \n",array2)

# shape array with 4 rows and 2 columns
array3 = np.arange(8).reshape(4, 2)
print("\narray reshaped with 4 rows and 2 columns : \n",array3)

# Constructs 3D array
array4 = np.arange(8).reshape(2, 2, 2)
print("\nOriginal array reshaped to 3D : \n",array4)
```

```
Original array :
 [0 1 2 3 4 5 6 7]

array reshaped with 2 rows and 4 columns :
 [[0 1 2 3]
 [4 5 6 7]]

array reshaped with 4 rows and 2 columns :
 [[0 1]
 [2 3]
 [4 5]
 [6 7]]

Original array reshaped to 3D :
 [[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]
```

Program-2

Method:II

Assigning the data:

```
#Import pandas package
import pandas as pd

# Assign data
data = {'Name': ['Jai', 'Princi', 'Gaurav',
                 'Anuj', 'Ravi', 'Natasha', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```

	Name	Age	Gender	Marks
0	Jai	17	M	90
1	Princi	17	F	76
2	Gaurav	18	M	NaN
3	Anuj	17	M	74
4	Ravi	18	M	65
5	Natasha	17	F	NaN
6	Riya	17	F	71

Program-3

```
# Categorize gender
df['Gender'] = df['Gender'].map({'M': 0,
                                 'F': 1, }).astype(float)

# Display data
df
```

	Name	Age	Gender	Marks
0	Jai	17	0.0	90
1	Princi	17	1.0	76
2	Gaurav	18	0.0	NaN
3	Anuj	17	0.0	74
4	Ravi	18	0.0	65
5	Natasha	17	1.0	NaN
6	Riya	17	1.0	71

Filtering the data

Suppose there is a requirement for the details regarding name, gender, marks of the top-scoring students. Here we need to remove some unwanted data.

Program-1

```
df.filter(['Name'])
```

	Name
0	Jai
1	Princi
2	Gaurav
3	Anuj
4	Ravi
5	Natasha
6	Riya

Program-2

```
df.filter(['Age'])
```

	Age
0	17
1	17
2	18
3	17
4	18
5	17
6	17

Program-3

```
: df[df['Age'] == 17]
```

:

	Name	Age	Gender	Marks
0	Jai	17	0.0	90
1	Princi	17	1.0	76
3	Anuj	17	0.0	74
5	Natasha	17	1.0	NaN
6	Riya	17	1.0	71

Merge data:
Merge operation is used to merge raw data and into the desired format.
Syntax:
pd.merge(data_frame1,data_frame2, on="field ")

Program-4
First type of data:

```
# import module
import pandas as pd

# creating DataFrame for Student Details
details = pd.DataFrame({
    'ID': [101, 102, 103, 104, 105, 106,
           107, 108, 109, 110],
    'NAME': ['Jagroop', 'Praveen', 'Harjot',
             'Pooja', 'Rahul', 'Nikita',
             'Saurabh', 'Ayush', 'Dolly', "Mohit"],
    'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE', 'CSE',
               'CSE', 'CSE', 'CSE', 'CSE', 'CSE']})

# printing details
print(details)
```

```
    ID     NAME BRANCH
0  101  Jagroop    CSE
1  102  Praveen    CSE
2  103   Harjot    CSE
3  104    Pooja    CSE
4  105    Rahul    CSE
5  106   Nikita    CSE
6  107  Saurabh    CSE
7  108    Ayush    CSE
8  109    Dolly    CSE
9  110    Mohit    CSE
```

Program-5
Second type of data:

```
# Import module
import pandas as pd

# Creating Dataframe for Fees_Status
fees_status = pd.DataFrame(
    {'ID': [101, 102, 103, 104, 105,
            106, 107, 108, 109, 110],
     'PENDING': ['5000', '250', 'NIL',
                 '9000', '15000', 'NIL',
                 '4500', '1800', '250', 'NIL']})

# Printing fees_status
print(fees_status)
```

```
    ID PENDING
0  101    5000
1  102     250
2  103     NIL
3  104    9000
4  105   15000
5  106     NIL
6  107    4500
7  108    1800
8  109     250
9  110     NIL
```

Program-6

```
print(pd.merge(details, fees_status, on='ID'))
```

```
    ID     NAME BRANCH PENDING
0  101  Jagroop    CSE    5000
1  102  Praveen    CSE     250
2  103   Harjot    CSE     NIL
3  104    Pooja    CSE    9000
4  105    Rahul    CSE   15000
5  106   Nikita    CSE     NIL
6  107  Saurabh    CSE    4500
7  108    Ayush    CSE    1800
8  109    Dolly    CSE     250
9  110    Mohit    CSE     NIL
```

Handling the missing values:

Program-1

```
# Import module
import pandas as pd
import numpy as np

# Creating Dataframe for Fees_Status
fees_status = pd.DataFrame(
    {'ID': [101, 102, 103, 104, 105,
            106, 107, 108, 109, 110],
     'PENDING': [5000, 250, np.nan,
                 9000, 15000, np.nan,
                 4500, 1800, 250, np.nan]})

# Printing fees_status
fees_status
```

	ID	PENDING
0	101	5000.0
1	102	250.0
2	103	NaN
3	104	9000.0
4	105	15000.0
5	106	NaN
6	107	4500.0
7	108	1800.0
8	109	250.0
9	110	NaN

Program-2

In order to check null values in Pandas DataFrame, we use isnull() function. This function return dataframe of Boolean values which are True for NaN values.

```
pd.isnull(fees_status["PENDING"])
```

```
0    False
1    False
2     True
3    False
4    False
5     True
6    False
7    False
8    False
9     True
Name: PENDING, dtype: bool
```

Program-3

In order to check null values in Pandas Dataframe, we use notnull() function this function return dataframe of Boolean values which are False for NaN values.

```
print(fees_status.notnull())
```

```
     ID  PENDING
0  True     True
1  True     True
2  True    False
3  True     True
4  True     True
5  True    False
6  True     True
7  True     True
8  True     True
9  True    False
```

Program-4

```
import pandas as pd

df = pd.read_csv (r'E:\ml datasets\Machine_Learning_Data_Preprocessing_Python-master\Sample_real_estate_data.csv')
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3	1.5	100.0
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1	1	NaN
8	100009000.0	215.0	TREMONT	Y	na	2	1800.0

Program-5

```
print(df['ST_NUM'].isnull())
```

```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7    False
8    False
Name: ST_NUM, dtype: bool
```

Program-6

```
print(df.isnull())
```

```
     PID  ST_NUM  ST_NAME  OWN_OCCUPIED  NUM_BEDROOMS  NUM_BATH  SQ_FT
0  False   False    False         False         False     False  False
1  False   False    False         False         False     False  False
2  False    True    False         False          True     False  False
3  False   False    False          True         False      True  False
4   True   False    False         False         False     False  False
5  False   False    False         False          True     False  False
6  False    True    False          True         False     False  False
7  False   False    False         False         False     False   True
8  False   False    False         False         False     False  False
```

Program-7

Method-I

Drop Columns with Missing Values

```
df = df.drop(['ST_NUM'], axis=1)
```

```
df
```

	PID	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	PUTNAM	Y	3	1	1000.0
1	100002000.0	LEXINGTON	N	3	1.5	100.0
2	100003000.0	LEXINGTON	N	NaN	1	850.0
3	100004000.0	BERKELEY	NaN	1	NaN	700.0
4	NaN	BERKELEY	Y	3	2	1600.0
5	100006000.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	WASHINGTON	NaN	2	HURLEY	950.0
7	100008000.0	TREMONT	Y	1	1	NaN
8	100009000.0	TREMONT	Y	na	2	1800.0

Program-8

Method-II

`fillna()` manages and let the user replace NaN values with some value of their own

```
import pandas as pd

# making data frame from csv file
data = pd.read_csv(r'E:\ml datasets\Machine_Learning_Data_Preprocessing_Python-master\Sample_real_estate_data.csv')

# replacing nan values in pid with No id
data["PID"].fillna("No ID", inplace = True)

data
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3	1.5	100.0
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1	NaN	700.0
4	No ID	203.0	BERKELEY	Y	3	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1	1	NaN
8	100009000.0	215.0	TREMONT	Y	na	2	1800.0

Program-9

```
import numpy as np
import pandas as pd

# A dictionary with list as values
GFG_dict = { 'G1': [10, 20,30,40],
                'G2': [25, np.NaN, np.NaN, 29],
                'G3': [15, 14, 17, 11],
                'G4': [21, 22, 23, 25]}

# Create a DataFrame from dictionary
gfg = pd.DataFrame(GFG_dict)


print(gfg)
```

```
   G1    G2  G3  G4
0  10  25.0  15  21
1  20   NaN  14  22
2  30   NaN  17  23
3  40  29.0  11  25
```

Program-10

Filling missing values with mean

```
import numpy as np
import pandas as pd

# A dictionary with list as values
GFG_dict = { 'G1': [10, 20,30,40],
                'G2': [25, np.NaN, np.NaN, 29],
                'G3': [15, 14, 17, 11],
                'G4': [21, 22, 23, 25]}

# Create a DataFrame from dictionary
gfg = pd.DataFrame(GFG_dict)

#Finding the mean of the column having NaN
mean_value=gfg['G2'].mean()

# Replace NaNs in column S2 with the
# mean of values in the same column
gfg['G2'].fillna(value=mean_value, inplace=True)
print('Updated Dataframe:')
print(gfg)
```

```
Updated Dataframe:
   G1    G2  G3  G4
0  10  25.0  15  21
1  20  27.0  14  22
2  30  27.0  17  23
3  40  29.0  11  25
```

Program-11

Filling missing values in csv files:

df=pd.read_csv(r'E:\mldatasets\Machine_Learning_Data_Preprocessing_Python-master\Sample_real_estate_data.csv', na_values='NAN')

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3	1.5	100.0
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1	1	NaN
8	100009000.0	215.0	TREMONT	Y	na	2	1800.0

Program-12

```
df['PID'] = df['PID'].fillna(df['PID'].mean())
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.000000	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.000000	1.5	100.0
2	100003000.0	NaN	LEXINGTON	N	2.166667	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.000000	NaN	700.0
4	100005000.0	203.0	BERKELEY	Y	3.000000	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	2.166667	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.000000	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1.000000	1	NaN
8	100009000.0	215.0	TREMONT	Y	2.166667	2	1800.0

Program-13

Code:

missing_value = ["n/a","na","--"]
data1=pd.read_csv(r'E:\mldatasets\Machine_Learning_Data_Preprocessing_Python-master\Sample_real_estate_data.csv', na_values = missing_value)
df = data1

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.000000	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.000000	1.5	100.0
2	100003000.0	NaN	LEXINGTON	N	2.166667	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.000000	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.000000	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	2.166667	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.000000	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1.000000	1	NaN
8	100009000.0	215.0	TREMONT	Y	2.166667	2	1800.0

Exercise:

Load a real dataset (For example, Iris). Then apply Min-max normalization on the features of the dataset.

WEEK-5

Implement Dimensionality reduction using Principle Component Analysis (PCA) method on a dataset (For example Iris).

Principal Component Analysis (PCA): The idea of principal component analysis (PCA) is to reduce the dimensionality of a dataset consisting of a large number of related variables while retaining as much variance in the data as possible. PCA finds a set of new variables that the original variables are just their linear combinations. The new variables are called Principal Components (PCs). These principal components are orthogonal: In a 3-D case, the principal components are perpendicular to each other. Figure (A) shows the intuition of PCA: it "rotates" the axes to line up better with your data. The first principal component will capture most of the variance in the data, then followed by the second, third, and so on. As a result, the new data will have fewer dimensions.

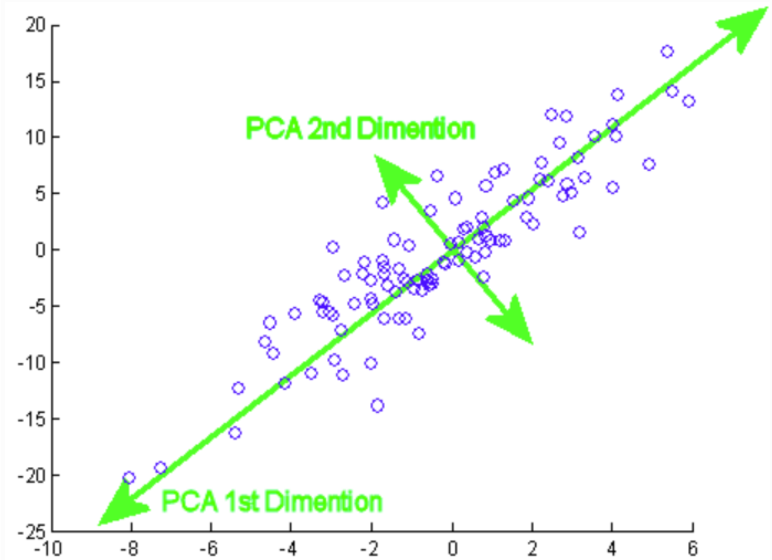
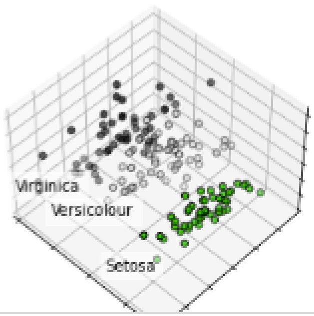


Figure (A)

Below the PCA is applied on Iris datasets.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import decomposition
from sklearn import datasets
import mpl_toolkits.mplot3d

np.random.seed(5)

iris = datasets.load_iris()
X = iris.data
y = iris.target

fig = plt.figure(1, figsize=(4, 3))
plt.clf()

ax = fig.add_subplot(111, projection="3d", elev=48, azim=134)
ax.set_position([0, 0, 0.95, 1])


plt.cla()
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)

for name, label in [("Setosa", 0), ("Versicolour", 1), ("Virginica", 2)]:
    ax.text3D(
        X[y == label, 0].mean(),
        X[y == label, 1].mean() + 1.5,
        X[y == label, 2].mean(),
        name,
        horizontalalignment="center",
        bbox=dict(alpha=0.5, edgecolor="w", facecolor="w"),
    )
# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(float)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, cmap=plt.cm.nipy_spectral, edgecolor="k")

ax.xaxis.set_ticklabels([])
ax.yaxis.set_ticklabels([])
ax.zaxis.set_ticklabels([])

plt.show()
```



Exercise:

Apply the PCA on Ionosphere or Pima dataset.

Week-6

Write a program to demonstrate the working of the decision tree based ID3 algorithm by considering a dataset.

Decision Tree: A decision tree mainly contains of a root node, interior nodes, and leaf nodes which are then connected by branches. The main idea of decision trees (ID3) is to find those descriptive features which contain the most "information" regarding the target feature and then split the dataset along the values of these features such that the target feature values for the resulting sub-datasets are as pure as possible. The descriptive feature which leaves the target feature most purely is said to be the most informative one. This process of finding the "most informative" feature is done until we accomplish a stopping criteria where we then finally end up in so called leaf nodes. Information gain is a measure of how good a descriptive feature is suited to split a dataset on. o be able to calculate the information gain, we have to first introduce the term entropy of a dataset. The entropy of a dataset is used to measure the impurity of a dataset and we will use this kind of informativeness measure in our calculations.

ID3 applied on tennis.csv dataset.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import copy

dataset = pd.read_csv('/content/gdrive/My Drive/tennis.csv')
X = dataset.iloc[:, 1:].values
# print(X)
attribute = ['outlook', 'temp', 'humidity', 'wind']


class Node(object):
    def __init__(self):
        self.value = None
        self.decision = None
        self.childs = None


def findEntropy(data, rows):
    yes = 0
    no = 0
    ans = -1
    idx = len(data[0]) - 1
    entropy = 0
    for i in rows:
        if data[i][idx] == 'Yes':
            yes = yes + 1
        else:
            no = no + 1
```

```
    x = yes/(yes+no)
    y = no/(yes+no)
    if x != 0 and y != 0:
        entropy = -1 * (x*math.log2(x) + y*math.log2(y))
    if x == 1:
        ans = 1
    if y == 1:
        ans = 0
    return entropy, ans


def findMaxGain(data, rows, columns):
    maxGain = 0
    retidx = -1
    entropy, ans = findEntropy(data, rows)
    if entropy == 0:
        """if ans == 1:
            print("Yes")
        else:
            print("No")"""
        return maxGain, retidx, ans

    for j in columns:
        mydict = {}
        idx = j
        for i in rows:
            key = data[i][idx]
            if key not in mydict:
                mydict[key] = 1
            else:
                mydict[key] = mydict[key] + 1
        gain = entropy

        # print(mydict)
        for key in mydict:
            yes = 0
            no = 0
            for k in rows:
                if data[k][j] == key:
                    if data[k][-1] == 'Yes':
                        yes = yes + 1
                    else:
                        no = no + 1
            # print(yes, no)
            x = yes/(yes+no)
            y = no/(yes+no)
```

```
            # print(x, y)
            if x != 0 and y != 0:
                gain += (mydict[key] * (x*math.log2(x) + y*math.log2(y)))/14
        # print(gain)
        if gain > maxGain:
            # print("hello")
            maxGain = gain
            retidx = j

    return maxGain, retidx, ans


def buildTree(data, rows, columns):

    maxGain, idx, ans = findMaxGain(X, rows, columns)
    root = Node()
    root.childs = []
    # print(maxGain
    #
    # )
    if maxGain == 0:
        if ans == 1:
            root.value = 'Yes'
        else:
            root.value = 'No'
        return root

    root.value = attribute[idx]
    mydict = {}
    for i in rows:
        key = data[i][idx]
        if key not in mydict:
            mydict[key] = 1
        else:
            mydict[key] += 1

    newcolumns = copy.deepcopy(columns)
    newcolumns.remove(idx)
    for key in mydict:
        newrows = []
        for i in rows:
            if data[i][idx] == key:
                newrows.append(i)
        # print(newrows)
        temp = buildTree(data, newrows, newcolumns)
        temp.decision = key
        root.childs.append(temp)
```

```
    return root


def traverse(root):
    print(root.decision)
    print(root.value)

    n = len(root.childs)
    if n > 0:
        for i in range(0, n):
            traverse(root.childs[i])


def calculate():
    rows = [i for i in range(0, 14)]
    columns = [i for i in range(0, 4)]
    root = buildTree(X, rows, columns)
    root.decision = 'Start'
    traverse(root)


calculate()
```

Exercise:
Apply ID3 on a different dataset.

Week-7

Consider a dataset, use Random Forest to predict the output class. Vary the number of trees as follows and compare the results:

a. 20
b. 50
c. 100
d. 200
e. 500

Random Forest: The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It collects the votes from different decision trees to decide the final prediction.

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.target_names)
print(iris.feature_names)
# dividing the datasets into two parts i.e. training datasets and test datasets
X, y = datasets.load_iris( return_X_y = True)

# Splitting arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
# i.e. 70 % training dataset and 30 % test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
# creating dataframe of IRIS dataset
data = pd.DataFrame({'sepallength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1],
                     'petallength': iris.data[:, 2], 'petalwidth': iris.data[:, 3],
                     'species': iris.target})
# creating a RF classifier
clf = RandomForestClassifier(n_estimators = 20)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred = clf.predict(X_test)

# metrics are used to find accuracy or error
from sklearn import metrics
print()

# using metrics module for accuracy calculation
```

```
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
```

Exercise:

Apply Random forest by varying the number of trees to 50, 100, 200, 500 and analyze the variation in the accuracies obtained.

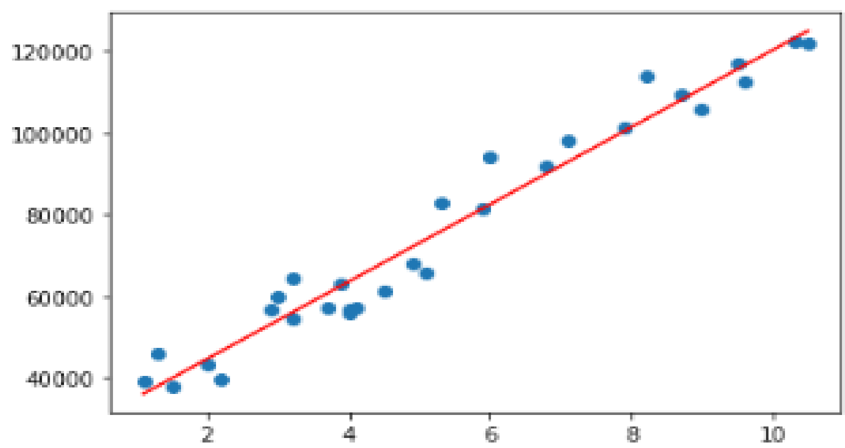
Week-8

Write a Python program to implement Simple Linear Regression and plot the graph.

Linear Regression: Linear regression is defined as an algorithm that provides a linear relationship between an independent variable and a dependent variable to predict the outcome of future events. It is a statistical method used in data science and machine learning for predictive analysis. Linear regression is a supervised learning algorithm that simulates a mathematical relationship between variables and makes predictions for continuous or numeric variables such as sales, salary, age, product price, etc.

Program:

```
# Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression

dataset = pd.read_csv('Salary_Data.csv')
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
liner =LinearRegression()

#x = x.reshape(-1,1)
liner.fit(x,y)
y_pred = liner.predict(X)

plt.scatter(x,y)
plt.plot(x,y_pred,color='red')
plt.show()
```



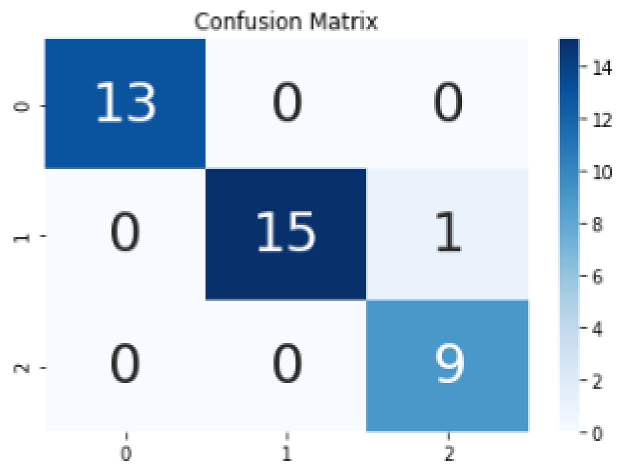
Exercise: Implement Simple Linear Regression on a different dataset and plot the graph.

Week-9

Write a Python program to implement Logistic Regression for iris using sklearn

```
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

dataset = pd.read_csv('iris.csv')

#print(dataset.head())
#dataset.info()
# Splitting the dataset into the Training set and Test set
x = dataset.iloc[:, [0,1,2, 3]].values
#print(x)
y = dataset.iloc[:, 4].values
#print(y)

# Split the dataset into training and test dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)

# Create a Logistic Regression Object, perform Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)

y_pred = log_reg.predict(x_test)

cm =confusion_matrix(y_test,y_pred)

print(cm)


# Plot confusion matrix
import seaborn as sns
import pandas as pd
# confusion matrix sns heatmap
## https://www.kaggle.com/agungor2/various-confusion-matrix-plots
ax = plt.axes()
df_cm = cm
sns.heatmap(df_cm, annot=True, annot_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax )
ax.set_title('Confusion Matrix')
plt.show()
```

```
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```



Exercise: Implement Logistic Regression on a different dataset and plot the confusion matrix.

Week-10

Implement Support Vector Machine for a dataset and compare the accuracy by applying the following kernel functions:

a. **Linear**
b. **Polynomial**
c. **RBF**

```
import matplotlib.pyplot as plt
import pandas as pd
#Load the Dataset
dataset = pd.read_csv('Social_Network_Ads.csv')

#Split Dataset into X and Y
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values

#Split the X and Y Dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

#Perform Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fit SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)

#Predict the Test Set Results
y_pred = classifier.predict(X_test)
print(y_pred)

# predict accuracy
accuracy_score(y_test,y_pred)
```

```
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 1 1 1]
```

```
5]: 0.93
```

Exercise: Write programs to implement the Linear and Polynomial and RBF kernels on IRIS and

compare the results.

Week-11

Build KNN Classification model for a given dataset. Vary the number of k values as follows and compare the results:

a. **1**
b. **3**
c. **5**
d. **7**
e. **11**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
X, y = datasets.load_iris( return_X_y = True)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=1)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

Exercise: Write programs to implement the KNN for k=3,5,7,11 and compare the results.

Week-12

Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets. Vary the activation functions used and compare the results.

```
from keras.models import Sequential
from keras.layers import Dense, Activation
import numpy as np
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
X, y = datasets.load_iris( return_X_y = True)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40)
# Define the network model and its arguments.
# Set the number of neurons/nodes for each layer:
model = Sequential()
model.add(Dense(2, input_shape=(4,)))
model.add(Activation('sigmoid'))
model.add(Dense(1))
model.add(Activation('sigmoid'))
#sgd = SGD(lr=0.0001, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
# Compile the model and calculate its accuracy:
model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['accuracy'])
#model.fit(X_train, y_train, batch_size=32, epochs=3)
# Print a summary of the Keras model:
model.summary()
#model.fit(X_train, y_train)
#model.fit(X_train, y_train, batch_size=32, epochs=300)
model.fit(X_train, y_train, epochs=5)
score = model.evaluate(X_test, y_test)
print(score)
```

Exercise

Vary the number of hidden layers, activation functions used and take the number of epochs as 50, 100, 150, 200 and compare the results.

Week-13

Write a python program to implement K-Means clustering Algorithm. Vary the number of k values as follows and compare the results:

a. **1**
b. **3**
c. **5**
d. **7**
e. **11**

Program:

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

#Import dataset
df = pd.read_csv('Live.csv')

#Check for missing values in dataset
df.isnull().sum()

#Drop redundant columns
df.drop(['status_id', 'status_published','Column1', 'Column2', 'Column3', 'Column4'], axis=1, inplace=True)

#Declare feature vector and target variable
X = df
y = df['status_type']

#Convert categorical variable into integers
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X['status_type'] = le.fit_transform(X['status_type'])
y = le.transform(y)

#Feature Scaling
cols = X.columns
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
X = ms.fit_transform(X)
X= pd.DataFrame(X, columns=[cols])

#K-Means model with four clusters
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(X)
labels = kmeans.labels_

# check how many of the samples were correctly labeled
correct_labels = np.sum(y == labels)
correct_labels
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```

Exercise:

Vary the number of clusters k values as follows on Iris dataset and compare the results. Remove the y-labels from the dataset as pre-processing.

a. **1**
b. **3**
c. **5**
d. **7**
e. **11**

Week-14

Write a python program to implement Hierarchical agglomerative clustering Algorithm.

a. Single linkage

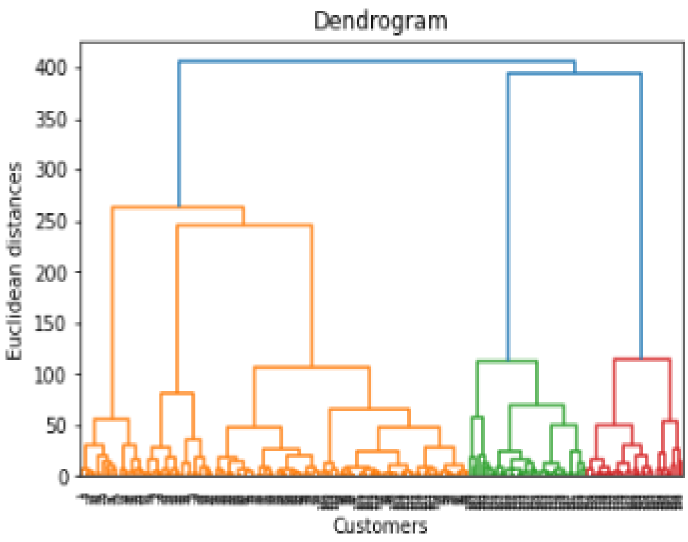
b. Complete linkage

c. Average linkage

d. Ward linkage

b) implementation of hierarchical clustering.

```
#implementation of hierarchical clustering.

#Import the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Mall_Customers.csv')

#Load the Dataset
X = dataset.iloc[:, [3, 4]].values

#Create Dendrogram to find the Optimal Number of Clusters
import scipy.cluster.hierarchy as sch
dendro = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()

#Fitting Agglomerative Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)

print(y_hc)
```




```
[4 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 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 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 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 1 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 1 2
 0 2 0 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
```

Exercise:

- ‘ward’ minimizes the variance of the clusters being merged.
- ‘average’ uses the average of the distances of each observation of the two sets.
- ‘complete’ or ‘maximum’ linkage uses the maximum distances between all observations of the two sets.
  ‘single’ ’ uses the minimum of the distances between all observations of the two sets.

Apply Hierarchical agglomerative clustering on Iris dataset on the following variations and compare the results. Remove the y-labels from the dataset as pre-processing.

e. **Single linkage**
f. **Complete linkage**
g. **Average linkage**
h. **Ward linkage**

Week-15

Write a python program to implement DBSCAN clustering Algorithm.

```
#dbscan

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
df = pd.read_csv('/content/gdrive/My Drive/Mall_Customers.csv')
X_train = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_train)
DBSCAN_dataset = X_train.copy()
DBSCAN_dataset.loc[:,'Cluster'] = clustering.labels_
DBSCAN_dataset.Cluster.value_counts().to_frame()
outliers = DBSCAN_dataset[DBSCAN_dataset['Cluster']==-1]

fig2, (axes) = plt.subplots(1,2,figsize=(12,5))

sns.scatterplot('Annual Income (k$)', 'Spending Score (1-100)',

                data=DBSCAN_dataset[DBSCAN_dataset['Cluster']!=-1],

                hue='Cluster', ax=axes[0], palette='Set2', legend='full', s=200)

sns.scatterplot('Age', 'Spending Score (1-100)',

                data=DBSCAN_dataset[DBSCAN_dataset['Cluster']!=-1],

                hue='Cluster', palette='Set2', ax=axes[1], legend='full', s=200)

axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (1-
100)'], s=10, label='outliers', c="k")

axes[1].scatter(outliers['Age'], outliers['Spending Score (1-
100)'], s=10, label='outliers', c="k")
axes[0].legend()
axes[1].legend()

plt.setp(axes[0].get_legend().get_texts(), fontsize='12')
plt.setp(axes[1].get_legend().get_texts(), fontsize='12')

plt.show()
```

Exercise:

Apply DBSCAN on IRIS dataset. Remove the y-labels from the dataset as pre-processing.

Week-16

Performance Analysis on a specific dataset (Mini Project)

Description:

- **Take any real time Dataset.**
- **Performance analysis with appropriate algorithms.**
- **Compare the results.**
- **Plot the graphs.**