ATME COLLEGE OF ENGINEERING

13th KM Stone, Bannur Road, Mysore - 560 028

A T M E
College of Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(ACADEMIC YEAR 2022-23)

LABORATORY MANUAL

SUBJECT: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY

SUB CODE: 18CSL76

SEMESTER: VII

As per Choice Based Credit System (CBCS)

and

Outcome Based Education (OBE)

(Effective from the academic year 2018 -2019)

INSTITUTIONAL MISSION AND VISION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission.

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

Department of Computer Science & Engineering

Vision of the Department

- To develop highly talented individuals in Computer Science and Engineering to deal with real world challenges in industry, education, research and society.

Mission of the Department

- To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivate students to put their thoughts and ideas adoptable by industry or to pursue higher studies leading to research.

Program Outcomes (POs)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. 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.

PO3. 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, and the cultural, societal, and environmental considerations.

PO4. 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.

PO5. 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.

PO6. 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.

PO7. 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.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. 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.

PO11. 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 multidisciplinary environments.

PO12. 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.

Program Educational Objectives (PEO'S):

1. Empower students with a strong basis in the mathematical, scientific and engineering fundamentals to solve computational problems and to prepare them for employment, higher learning and R&D.
2. Gain technical knowledge, skills and awareness of current technologies of computer science engineering and to develop an ability to design and provide novel engineering solutions for software/hardware problems through entrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects with effective communication skills and leadership qualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by Applying innovative ideas in the latest technology, to become effective professionals in Computer Science to bear a life-long career in related areas.

Program Specific Outcomes (PSOs)

1. Ability to apply skills in the field of algorithms, database design, web design, cloud computing and data analytics.
2. Apply knowledge in the field of computer networks for building network and internet based applications.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LABORATORY

As per Choice Based Credit System (CBCS) AND Outcome Based Education (OBE)

(Effective from the academic year 2018 -2019)

Subject Code	**:**	**18CSL76**	**CIE Marks**	**:**	**40**
Hours/Week	**:**	**0:0:2**	**SEE Marks**	**:**	**60**
Total Hours	**:**	**36**	**Exam Marks**	**:**	**03**

CREDITS – 02

Course Learning Objectives: This course (18CSL76) will enable students to:

- Implement and evaluate AI and ML algorithms in and Python programming language.

Description (If any):

Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Program List

1. Implement A* Search algorithm.

2. Implement AO* Search algorithm.

3. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

4. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

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

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

7. Assuming Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

8. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

9. Apply Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Laboratory outcomes: The students should be able to

1. Implement and demonstrate AI and ML algorithms.
2. Evaluate different algorithms.

Conduction of Practical Examination:

- Experiment distribution

o For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.

o For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.

- Students Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution (*Courseed to change in accoradance with university regulations*)

a) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks

b) For laboratories having PART A and PART B

i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks

ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

CONTENTS

INTRODUCTION

Since the invention of computers or machines, their capability to perform various tasks has experienced an exponential growth. Humans have developed the power of computer systems in terms of their diverse working domains, their increasing speed, and reducing size with respect to time.

A branch of Computer Science named Artificial Intelligence pursues creating the computers or machines as intelligent as human beings.

Basic Concept of Artificial Intelligence (AI)

According to the father of Artificial Intelligence, John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs".

Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think. AI is accomplished by studying how human brain thinks and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, "Can a machine think and behave like humans do?"

Thus, the development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

The Necessity of Learning AI

As we know that AI pursues creating the machines as intelligent as human beings. There are numerous reasons for us to study AI. The reasons are as follows:

AI can learn through data

In our daily life, we deal with huge amount of data and human brain cannot keep track of so much data. That is why we need to automate the things. For doing automation, we need to study AI because it can learn from data and can do the repetitive tasks with accuracy and without tiredness.

AI can teach itself

It is very necessary that a system should teach itself because the data itself keeps changing and the knowledge which is derived from such data must be updated constantly.

We can use AI to fulfill this purpose because an AI enabled system can teach itself.

AI can respond in real time

Artificial intelligence with the help of neural networks can analyze the data more deeply. Due to this capability, AI can think and respond to the situations which are based on the conditions in real time.

AI achieves accuracy

With the help of deep neural networks, AI can achieve tremendous accuracy. AI helps in the field of medicine to diagnose diseases such as cancer from the MRIs of patients.

AI can organize data to get most out of it

The data is an intellectual property for the systems which are using self-learning algorithms. We need AI to index and organize the data in a way that it always gives the best results.

Understanding Intelligence

With AI, smart systems can be built. We need to understand the concept of intelligence so that our brain can construct another intelligence system like itself.

What's Involved in AI

Artificial intelligence is a vast area of study. This field of study helps in finding solutions to real world problems.

Let us now see the different fields of study within AI:

Machine Learning

It is one of the most popular fields of AI. The basic concept of this filed is to make the machine learning from data as the human beings can learn from his/her experience. It contains learning models on the basis of which the predictions can be made on unknown data.

Logic

It is another important field of study in which mathematical logic is used to execute the computer programs. It contains rules and facts to perform pattern matching, semantic analysis, etc.

Searching

This field of study is basically used in games like chess, tic-tac-toe. Search algorithms give the optimal solution after searching the whole search space.

Artificial neural networks

This is a network of efficient computing systems the central theme of which is borrowed from the analogy of biological neural networks. ANN can be used in robotics, speech recognition, speech processing, etc.

Genetic Algorithm

Genetic algorithms help in solving problems with the assistance of more than one program. The result would be based on selecting the fittest.

Knowledge Representation

It is the field of study with the help of which we can represent the facts in a way the machine that is understandable to the machine. The more efficiently knowledge is represented; the more system would be intelligent.

Application of AI

In this section, we will see the different fields supported by AI:

Gaming

AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing

It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems

There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems

These systems understand, interpret, and comprehend visual input on the computer. For example,

- A spying aeroplane takes photographs, which are used to figure out spatial

information or map of the areas.

- Doctors use clinical expert system to diagnose the patient.
- Police use computer software that can recognize the face of criminal with the stored

portrait made by forensic artist.

Speech Recognition

Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition

The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots

Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

1.1 What is Machine Learning?

- It is very hard to write programs that solve problems like recognizing a face.
  - We don't know what program to write because we don't know how our brain does it.
  - Even if we had a good idea about how to do it, the program might be horrendously complicated.
- Instead of writing a program manually, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
  - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
  - If we do it right, the program works for new cases as well as the ones we trained it on.

A classic example of a task that requires machine learning: It is very hard to say what makes a 2

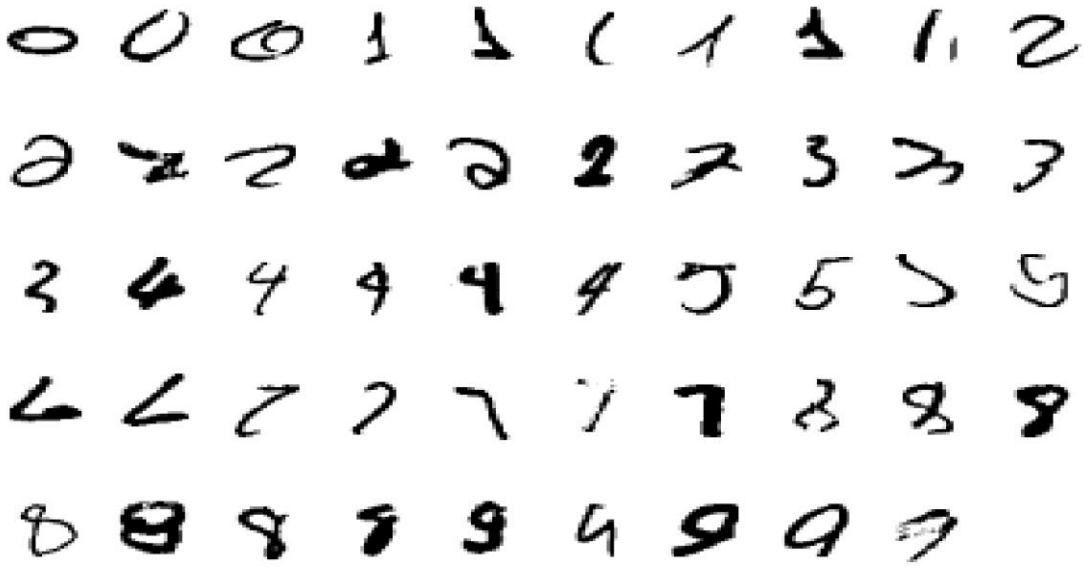


Fig 1: classic example

Some more examples of tasks that are best solved by using a learning algorithm

- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Generating patterns:
  - Generating images or motion sequences (demo)
- Recognizing anomalies:
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant or unusual sound in your car engine.
- Prediction:
  - Future stock prices or currency exchange rates

Some web-based examples of machine learning

- The web contains a lot of data. Tasks with very big datasets often use machine learning
- especially if the data is noisy or non-stationary.
- Spam filtering, fraud detection:
- The enemy adapts so we must adapt too.
- Recommendation systems:
- Lots of noisy data. Million-dollar prize!
- Information retrieval:
- Find documents or images with similar content.
- Data Visualization:
- Display a huge database in a revealing way (demo)

Displaying the structure of a set of documents using Latent Semantic Analysis (a form of PCA)

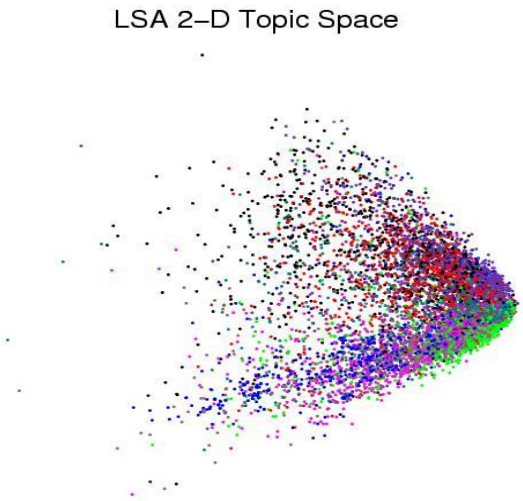


Fig 2: structure of a set of documents

Each document is converted to a vector of word counts. This vector is then mapped to two coordinates and displayed as a colored dot. The colors represent the hand-labeled classes.

When the documents are laid out in 2-D, the classes are not used. So we can judge how good the algorithm is by seeing if the classes are separated.

Displaying the structure of a set of documents using a deep neural network

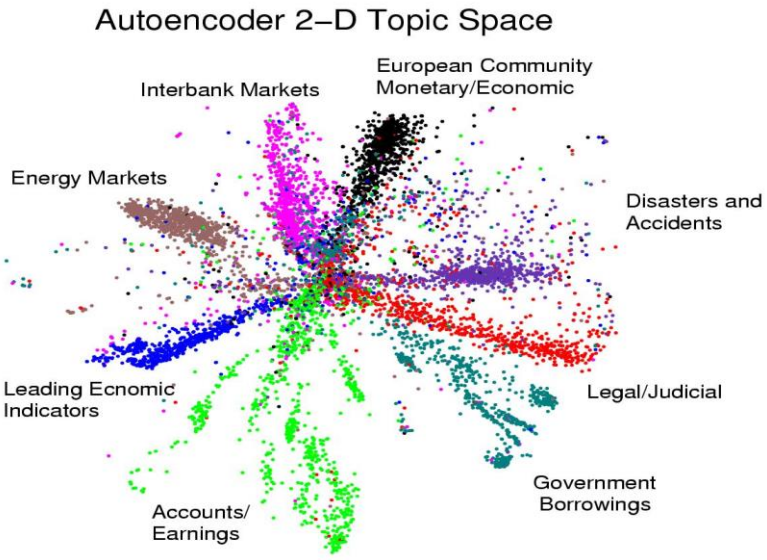


Fig 3: Autoencoder 2-D Topic Space

Machine Learning & Symbolic AI

- Knowledge Representation works with facts/assertions and develops rules of logical inference. The rules can handle quantifiers. Learning and uncertainty are usually ignored.

- Expert Systems used logical rules or conditional probabilities provided by "experts" for specific domains.

- Graphical Models treat uncertainty properly and allow learning (but they often ignore quantifiers and use a fixed set of variables)

  - Set of logical assertions: values of a subset of the variables and local models of the probabilistic interactions between variables.

  - Logical inference: probability distributions over subsets of the unobserved variables (or individual ones)

  - Learning = refining the local models of the interactions.

Machine Learning & Statistics

- A lot of machine learning is just a rediscovery of things that statisticians already knew. This is often disguised by differences in terminology:

  - Ridge regression = weight-decay

  - Fitting = learning

  - Held-out data = test data

- But the emphasis is very different:

  - A good piece of statistics: Clever proof that a relatively simple estimation procedure is asymptotically unbiased.

  - A good piece of machine learning: Demonstration that a complicated algorithm produces impressive results on a specific task.

- Data-mining: Using very simple machine learning techniques on very large databases because computers are too slow to do anything more interesting with ten billion examples.

A spectrum of machine learning tasks

Statistics	Artificial Intelligence
Low-dimensional data (e.g. less than 100 dimensions)	High-dimensional data (e.g. more than 100 dimensions)
Lots of noise in the data	The noise is not sufficient to obscure the structure in the data if we process it right.
There is not much structure in the data, and what structure there is, can be represented by a fairly simple model.	There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.
The main problem is distinguishing true structure from noise.	The main problem is figuring out a way to represent the complicated structure that allows it to be learned.

Types of learning task

- Supervised learning

  - Learn to predict output when given an input vector

  Who provides the correct answer?

- Reinforcement learning

  - Learn action to maximize payoff

  Not much information in a payoff signal

  Payoff is often delayed

  - Reinforcement learning is an important area that will not be covered in this course.

- Unsupervised learning

  - Create an internal representation of the input e.g. form clusters; extract features

  How do we know if a representation is good?

  - This is the new frontier of machine learning because most big datasets do not come with labels.

1.2 What is Python?

- A programming language with strong similarities to PERL, but with powerful typing and object oriented features.

  - Commonly used for producing HTML content on websites. Great for text files.

  - Useful built-in types (lists, dictionaries).

- Clean syntax, powerful extensions.

Why Python?

- Natural Language ToolKit
- Ease of use; interpreter
- AI Processing: Symbolic
  - Python's built-in datatypes for strings, lists, and more.
  - Java or C++ requires the use of special classes for this.
- AI Processing: Statistical
  - Python has strong numeric processing capabilities: matrix operations, etc.
  - Suitable for probability and machine learning code.

Look at a sample of code…

```
x = 34 - 23          # A comment.
y = "Hello"          # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
        x = x + 1
        y = y + "World"          # String concat.
print x
print y
```

Enough to Understand the Code

- Assignment uses = and comparison uses ==.
- For numbers +-*/% are as expected.
  - Special use of + for string concatenation.
  - Special use of % for string formatting.
- Logical operators are words (and, or, not) not symbols (&&, ||!).
- The basic printing command is "print."
- First assignment to a variable will create it.
  - Variable types don't need to be declared.

- Python figures out the variable types on its own.

Basic Datatypes

- Integers (default for numbers)

  z = 5 / 2 # Answer is 2, integer division.

- Floats

  x = 3.456

- Strings

Can use “” or ‘’ to specify. “abc” ‘abc’ (Same thing.) Unmatched ones can occur within the string. “matt’s” Use triple double-quotes for multi-line strings or strings than contain both ‘ and “ inside of them: “””a‘b“c”””

Whitespace

- Whitespace is meaningful in Python: especially indentation and placement of newlines.

  - Use a newline to end a line of code. (Not a semicolon like in C++ or Java.)
  (Use \ when must go to next line prematurely.)

  - No braces { } to mark blocks of code in Python… Use consistent indentation instead. The first line with a new indentation is considered outside of the block.

  - Often a colon appears at the start of a new block. (We’ll see this later for function and class definitions.)

Comments

- Start comments with # – the rest of line is ignored.
- Can include a “documentation string” as the first line of any new function or class that you define.
- The development environment, debugger, and other tools use it: it’s good style to include one.

  def my_function(x, y):

“””This is the docstring. This function does blah blah blah.””” # The code would go here...

Look at a sample of code…

x = 34 - 23 # A comment.

y = “Hello” # Another one.

```
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"   # String concat.
print x
print y
```

Python and Types

Python determines the data types in a program automatically.

"Dynamic Typing"

But Python's not casual about types, it enforces them after it figures them out.

"Strong Typing"

So, for example, you can't just append an integer to a string. You must first convert the integer to a string itself.

```
x = "the answer is "  # Decides x is string.
y = 23 # Decides y is integer.
print x + y  # Python will complain about this.
```

Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

  bob Bob _bob _2_bob_ bob_2 BoB

- There are some reserved words:

  and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while

Accessing Non-existent Name

- If you try to access a name before it's been properly created (by placing it on the left side of an assignment), you'll get an error.

  >>> y

```
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-y
NameError: name 'y' is not defined
```

```
>>> y = 3

>>> y
```

Multiple Assignment

- You can also assign to multiple names at the same time.

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

String Operations

- We can use some methods built-in to the string data type to perform some formatting operations on strings:

```
>>> "hello".upper()

'HELLO'
```

- There are many other handy string operations available. Check the Python documentation for more.

Printing with Python

- You can print a string to the screen using "print."
- Using the % string operator in combination with the print command, we can format our output text.

```
>>> print  "%s xyz %d"  %  ("abc", 34)

abc xyz 34
```

"Print" automatically adds a newline to the end of the string. If you include a list of strings, it will concatenate them with a space between them.

```
>>> print "abc"

>>> print "abc", "def"

abc

abc def
```

Numpy

Let's start with NumPy. Among other things, NumPy contains:

A powerful N-dimensional array object.

Sophisticated (broadcasting/universal) functions.

Tools for integrating C/C++ and Fortran code.

Useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

The key to NumPy is the ndarray object, an n-dimensional array of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size. Modifying the size means creating a new array.

NumPy arrays must be of the same data type, but this can include Python objects.

More efficient mathematical operations than built-in sequence types.

Numpy datatypes

To begin, NumPy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:

intc (same as a C integer) and intp (used for indexing)

int8, int16, int32, int64

uint8, uint16, uint32, uint64

float16, float32, float64

complex64, complex128

bool_, int_, float_, complex_ are shorthand for defaults.

These can be used as functions to cast literals or sequence types, as well as arguments to numpy functions that accept the dtype keyword argument.

To begin, NumPy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:

intc (same as a C integer) and intp (used for indexing)

int8, int16, int32, int64

uint8, uint16, uint32, uint64

float16, float32, float64

complex64, complex128

Some examples:

```
>>> import numpy as np
>>> x = np.float32(1.0)
>>> x
1.0
>>> y = np.int_([1,2,4])
>>> y
array([1, 2, 4])
>>> z = np.arange(3, dtype=np.uint8)
>>> z
array([0, 1, 2], dtype=uint8)
>>> z.dtype
dtype('uint8')
```

Numpy arrays

There are a couple of mechanisms for creating arrays in NumPy:

- Conversion from other Python structures (e.g., lists, tuples).
- Built-in NumPy array creation (e.g., arange, ones, zeros, etc.).
- Reading arrays from disk, either from standard or custom formats (e.g. reading in from a CSV file).
- and others …

In general, any numerical data that is stored in an array-like container can be converted to an ndarray through use of the array() function. The most obvious examples are sequence types like lists and tuples.

```
>>> x = np.array([2,3,1,0])
>>> x = np.array([2, 3, 1, 0])
>>> x = np.array([[1,2.0],[0,0],(1+1j,3.)])
>>> x = np.array([[ 1.+0.j, 2.+0.j], [ 0.+0.j, 0.+0.j], [ 1.+1.j, 3.+0.j]])
```

There are a couple of built-in NumPy functions which will create arrays from scratch.

- zeros(shape) -- creates an array filled with 0 values with the specified shape. The default dtype is float64.

```
>>>np.zeros((2,3))

array([[ 0., 0., 0.], [ 0., 0., 0.]])
```

- ones(shape) -- creates an array filled with 1 values.
- arange() -- creates arrays with regularly incrementing values.

```
>>>np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>np.arange(2,10,dtype=np.float)
```

```
array([2.,3.,4.,5.,6.,7.,8.,9.])
>>>np.arange(2,3,0.1)
array([ 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9])
```

- linspace() -- creates arrays with a specified number of elements, and spaced equally between the specified beginning and end values.

```
>>>np.linspace(1.,4.,6)
array([ 1. , 1.6, 2.2, 2.8, 3.4, 4. ])
```

- random.random(shape) – creates arrays with random floats over the interval [0,1).

```
>>>np.random.random((2,3))
array([[0.75688597,0.41759916,0.35007419],[ 0.77164187, 0.05869089, 0.98792864]])
```

Printing an array can be done with the print statement.

```
>>> import numpy as np
>>> a = np.arange(3)
>>> print a
[0 1 2]
>>> a
array([0, 1, 2])
>>> b = np.arange(9).reshape(3,3)
>>> print b
[[0 1 2]
 [3 4 5]
 [6 7 8]]
>>> c = np.arange(8).reshape(2,2,2)
>>> print c
[[[0 1]
  [2 3]]
 [[4 5]
  [6 7]]]
```

Indexing

Single-dimension indexing is accomplished as usual.

```
>>> x = np.arange(10)
>>> x[2]
2 >>> x[-2]
8
```

```
[ 0  1  2  3  4  5  6  7  8  9 ]
```

Multi-dimensional arrays support multi-dimensional indexing.

```
>>> x.shape = (2,5) # now x is 2-dimensional
>>> x[1,3]
8
```

```
>>> x[1,-1]
9
```

```
⎛ 0  1  2  3  4 ⎞
⎜               ⎟
⎝ 5  6  7  8  9 ⎠
```

Using fewer dimensions to index will result in a subarray.

```
>>> x[0]
array([0, 1, 2, 3, 4])
```

This means that x[i, j] == x[i][j] but the second method is less efficient.

Slicing is possible just as it is for typical Python sequences.

```
>>> x = np.arange(10)
>>> x[2:5]
array([2, 3, 4])
>>> x[:-7]
array([0, 1, 2])
>>> x[1:7:2]
array([1, 3, 5])
>>> y = np.arange(35).reshape(5,7)
>>> y[1:5:2,::3]
array([[ 7, 10, 13], [21, 24, 27]])
```

Array operations

```
>>> a = np.arange(5)
>>> b = np.arange(5)
>>> a+b
array([0, 2, 4, 6, 8])
>>> a-b
array([0, 0, 0, 0, 0])
>>> a**2
array([ 0,  1,  4,  9, 16])
>>> a>3
array([False, False, False, False,  True], dtype=bool)
>>> 10*np.sin(a)
array([ 0., 8.41470985, 9.09297427, 1.41120008, -7.56802495])
>>> a*b
array([ 0,  1,  4,  9, 16])
```

Basic operations apply element-wise. The result is a new array with the resultant elements. Operations like *= and += will modify the existing array.

Since multiplication is done element-wise, you need to specifically perform a dot product to perform matrix multiplication.

```
>>> a = np.zeros(4).reshape(2,2)
>>> a
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> a[0,0] = 1
>>> a[1,1] = 1
>>> b = np.arange(4).reshape(2,2)
>>> b
array([[0, 1],
       [2, 3]])
>>> a*b
array([[ 0.,  0.],
       [ 0.,  3.]])
>>> np.dot(a,b)
array([[ 0.,  1.],
       [ 2.,  3.]])
```

There are also some built-in methods of ndarray objects. Universal functions which may also be applied include exp, sqrt, add, sin, cos, etc…

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.68166391, 0.98943098, 0.69361582],
       [ 0.78888081, 0.62197125, 0.40517936]])
>>> a.sum()
4.1807421388722164
>>> a.min()
0.4051793610379143
>>> a.max(axis=0)
array([ 0.78888081, 0.98943098, 0.69361582])
>>> a.min(axis=1)
array([ 0.68166391, 0.40517936])
```

Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we

can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install NumPy using popular Python package installer, pip.

pip install pandas

If you install Anaconda Python package, Pandas will be installed by default with the following

CSV File

Python has a vast library of modules that are included with its distribution. The csv module gives the Python programmer the ability to parse CSV (Comma Separated Values) files. A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter. You can think of each line as a row and each field as a column. The CSV format has no standard, but they are similar enough that the csv module will be able to read the vast majority of CSV files. You can also write CSV files using the csv module.

Reading a CSV File

There are two ways to read a CSV file. You can use the csv module's reader function or you can use the DictReader class. We will look at both methods. But first, we need to get a CSV file so we have something to parse. There are many websites that provide interesting information in CSV format. We will be using the World Health Organization's (WHO) website to download some information on Tuberculosis. You can go here to get it:

http://www.who.int/tb/country/data/download/en/. Once you have the file, we'll be ready to start. Ready? Then let's look at some code!

```
import csv
#----------------------------------------------------------------------
def csv_reader(file_obj):
    """
    Read a csv file
    """
    reader = csv.reader(file_obj)
    for row in reader:
        print(" ".join(row))
#----------------------------------------------------------------------
if __name__ == "__main__":
    csv_path = "TB_data_dictionary_2014-02-26.csv"
    with open(csv_path, "rb") as f_obj:
        csv_reader(f_obj)
```

Writing a CSV File

The csv module also has two methods that you can use to write a CSV file. You can use the writer function or the DictWriter class. We'll look at both of these as well. We will be with the writer function. Let's look at a simple example:

```
import csv
#----------------------------------------------------------------------
def csv_writer(data, path):
    """
    Write data to a CSV file path
    """
    with open(path, "wb") as csv_file:
        writer = csv.writer(csv_file, delimiter=',')
        for line in data:
            writer.writerow(line)
#----------------------------------------------------------------------
if __name__ == "__main__":
    data = ["first_name,last_name,city".split(","),
            "Tyrese,Hirthe,Strackeport".split(","),
            "Jules,Dicki,Lake Nickolasville".split(","),
            "Dedric,Medhurst,Stiedemannberg".split(",")
            ]
    path = "output.csv"
    csv_writer(data, path)
```

Program 1

1) Implement A* Search algorithm.

Program Objective:

- To find shortest path though search space.
- evaluate version space.

```
from collections import deque


class Graph:
    def __init__(self, adjac_lis):
        self.adjac_lis = adjac_lis


    def get_neighbors(self, v):
        return self.adjac_lis[v]


    # This is heuristic function which is having equal values for all nodes
    def h(self, n):
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1,
            'S': 1,
        }


        return H[n]


    def a_star_algorithm(self, start, stop):
        # In this open_lst is a lisy of nodes which have been visited, but who's
```

```
    # neighbours haven't all been always inspected, It starts off with the start
#node
    # And closed_lst is a list of nodes which have been visited
    # and who's neighbors have been always inspected
    open_lst = set([start])
    closed_lst = set([])

    # poo has present distances from start to all other nodes
    # the default value is +infinity
    poo = {}
    poo[start] = 0

    # par contains an adjac mapping of all nodes
    par = {}
    par[start] = start

    while len(open_lst) > 0:
       n = None

       # it will find a node with the lowest value of f() -
       for v in open_lst:
          if n == None or poo[v] + self.h(v) < poo[n] + self.h(n):
             n = v;

       if n == None:
          print('Path does not exist!')
          return None

       # if the current node is the stop
```

```
# then we start again from start
if n == stop:
    reconst_path = []

    while par[n] != n:
        reconst_path.append(n)
        n = par[n]

    reconst_path.append(start)

    reconst_path.reverse()

    print('Path found: {}'.format(reconst_path))
    return reconst_path

# for all the neighbors of the current node do
for (m, weight) in self.get_neighbors(n):
  # if the current node is not presentin both open_lst and closed_lst
    # add it to open_lst and note n as it's par
    if m not in open_lst and m not in closed_lst:
        open_lst.add(m)
        par[m] = n
        poo[m] = poo[n] + weight

    # otherwise, check if it's quicker to first visit n, then m
    # and if it is, update par data and poo data
    # and if the node was in the closed_lst, move it to open_lst
    else:
        if poo[m] > poo[n] + weight:
```

```
                poo[m] = poo[n] + weight
                par[m] = n

                if m in closed_lst:
                    closed_lst.remove(m)
                    open_lst.add(m)

            # remove n from the open_lst, and add it to closed_lst
            # because all of his neighbors were inspected
            open_lst.remove(n)
            closed_lst.add(n)

        print('Path does not exist!')
        return None
```

Input:

```
adjac_lis = {
        'A': [('B', 2), ('C', 5), ('D', 12)],
        'B': [('C', 2)],
        'C': [('D', 3)],
        'S': [('A', 1), ('B', 4)]
    }
graph1 = Graph(adjac_lis)
graph1.a_star_algorithm('S', 'D')
```

Output:

Path found: ['S', 'A', 'B', 'C', 'D']

['S', 'A', 'B', 'C', 'D']

Program Outcome:

- Understand the implementation procedures for the AI algorithms.
- Students will be able to apply A* Search algorithm and find optimal path

Program 2

2) Implement AO* Search algorithm.

Program Objective:

- To find set of consistent hypothesis.
- evaluate version space.

```
from collections import deque


class Graph:
    def __init__(self, adjac_lis):
        self.adjac_lis = adjac_lis


    def get_neighbors(self, v):
        return self.adjac_lis[v]


    # This is heuristic function which is having equal values for all nodes
    def h(self, n):
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }


        return H[n]


    def a_star_algorithm(self, start, stop):
        # In this open_lst is a lisy of nodes which have been visited, but who's
        # neighbours haven't all been always inspected, It starts off with the start
```

```
#node
    # And closed_lst is a list of nodes which have been visited
    # and who's neighbors have been always inspected
    open_lst = set([start])
    closed_lst = set([])

    # poo has present distances from start to all other nodes
    # the default value is +infinity
    poo = {}
    poo[start] = 0

    # par contains an adjac mapping of all nodes
    par = {}
    par[start] = start

    while len(open_lst) > 0:
       n = None

       # it will find a node with the lowest value of f() -
       for v in open_lst:
          if n == None or poo[v] + self.h(v) < poo[n] + self.h(n):
             n = v;

       if n == None:
          print('Path does not exist!')
          return None

       # if the current node is the stop
       # then we start again from start
```

```
if n == stop:
    reconst_path = []

    while par[n] != n:
        reconst_path.append(n)
        n = par[n]

    reconst_path.append(start)

    reconst_path.reverse()

    print('Path found: {}'.format(reconst_path))
    return reconst_path

# for all the neighbors of the current node do
for (m, weight) in self.get_neighbors(n):
  # if the current node is not presentin both open_lst and closed_lst
    # add it to open_lst and note n as it's par
    if m not in open_lst and m not in closed_lst:
        open_lst.add(m)
        par[m] = n
        poo[m] = poo[n] + weight

    # otherwise, check if it's quicker to first visit n, then m
    # and if it is, update par data and poo data
    # and if the node was in the closed_lst, move it to open_lst
    else:
        if poo[m] > poo[n] + weight:
            poo[m] = poo[n] + weight
```

```
                par[m] = n

                if m in closed_lst:
                    closed_lst.remove(m)
                    open_lst.add(m)

            # remove n from the open_lst, and add it to closed_lst
            # because all of his neighbors were inspected
            open_lst.remove(n)
            closed_lst.add(n)

        print('Path does not exist!')
        return None
```

Input:

```
adjac_lis = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjac_lis)
graph1.a_star_algorithm('A', 'D')
```

Output:

```
Path found: ['A', 'B', 'D']
['A', 'B', 'D']
```

Program Outcome:

- Understand the implementation procedures for the AI algorithms.
- Students will be able to apply AO* algorithm to the real world problem and find optimal path

Program 3

3) For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Program Objective:

- To find set of consistent hypothesis.
- To evaluate version space.
- To make use of Data sets in implementing the machine learning algorithm.

```
import pandas as pd
import numpy as np
data=pd.DataFrame(data=pd.read_csv('finds.csv'))
concepts=np.array(data.iloc[:,0:-1])
target=np.array(data.iloc[:,-1])
def learn(concepts,target):
    specific_h=concepts[0].copy()
    general_h=[["?" for i in range (len(specific_h))]for i in range (len(specific_h))]
    for i,h in enumerate(concepts):
        if target[i]=="yes":
            for x in range(len(specific_h)):
                if h[x]!=specific_h[x]:
                    specific_h[x]='?'
                    general_h[x][x]='?'
        if target[i]=="no":
            for x in range(len(specific_h)):
                if h[x]!=specific_h[x]:
                    general_h[x][x]=specific_h[x]
                else:
                    general_h[x][x]='?'
    indices=[i for i,val in enumerate(general_h)if val==['?','?','?','?','?','?']]
```

```
    for i in indices:
        general_h.remove(['?','?','?','?','?','?'])
    return specific_h,general_h
s_final,g_final=learn(concepts,target)
print("final s:",s_final,sep="\n")
print("final g:",g_final,sep="\n")
data.head()
```

Data Set: finds.csv

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cold	same	yes

Output:

```
final s:
['sunny' 'warm' '?' 'strong' '?' 'same']
final g:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
```

Out[1]:

	sky	temp	humidity	wind	water	forecast	enjoy
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cold	same	yes

Program Outcome:

- Understand the implementation procedures for the machine learning algorithms.
- Design Python programs for Candidate-Elimination algorithm.
- The students will be able to apply candidate elimination algorithm and output a description of the set of all hypotheses consistent with the training examples.

Program 4

4) Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Program Objective:

- To understand working of decision tree based ID3 algorithm.
- To efficiently classify new instances.

```
import math
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"))
    dataset=list(lines)
    headers=dataset.pop(0)
    return dataset,headers


class Node:
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=" "


def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))
    for k in attr:
      dic[k]=[]
    for y in range(len(data)):
       key=data[y][col]
```

```
        if delete:
            del data[y][col]
        dic[key].append(data[y])
    return attr,dic

def entropy(s):
    attr=list(set(s))
    if len(attr)==1:
        return 0
    counts=[0,0]
    for i in range(2):
        counts[i]=sum([1 for x in s if attr[i]==x])/(len(s)*1.0)
    sums=0
    for cnt  in counts:
        sums+=-1*cnt*math.log(cnt,2)
    return sums

def compute_gain(data,col):
    attvalues,dic = subtables(data,col,delete=False)
    total_entropy = entropy([row[-1] for row in data])
    for x in range(len(attvalues)):
        ratio=len(dic[attvalues[x]])/(len(data)*1.0)
        entro=entropy([row[-1] for row in dic[attvalues[x]]])
        total_entropy-=ratio*entro
    return total_entropy

def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
```

```
        node=Node(" ")
        node.answer=lastcol[0]
        return node
    n=len(data[0])-1
    gains=[compute_gain(data,col)for col in range(n)]
    split=gains.index(max(gains))
    node = Node(features[split])
    fea=features[:split]+features[split+1:]
    attr,dic=subtables(data,split,delete=True)
    for x in range(len(attr)):
        child=build_tree(dic[attr[x]],fea)
        node.children.append((attr[x],child))
    return node

def print_tree(node,level):
    if node.answer!=" ":
        print(" "*level,node.answer)
        return
    print(""*level,node.attribute)
    for value,n in node.children:
        print(" "*(level+1),value)
        print_tree(n,level+2)

def classify(node,x_test,features):
    if node.answer!=" ":
        print(node.answer)
        return
    pos=features.index(node.attribute)
    for value,n in node.children:
```

```
        if x_test[pos]==value:
            classify(n,x_test,features)


dataset,features=load_csv("playtennis.csv")
node=build_tree(dataset,features)
print("The decision tree for the dataset using id3 algorithm is")
print_tree(node,0)
testdata,features=load_csv("test_tennis.csv")
for xtest in testdata:
    print("the test instance:",xtest)
    print("the predicted label:",end=" ")
    classify(node,xtest,features)
```

Data Set: playtennis.csv

Outlook	Temperature	Humidity	Wind	Answer
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

test_tennis.csv

Outlook	Temperature	Humidity	Wind
rain	cool	normal	strong
sunny	mild	normal	strong

==========

Output:

```
The decision tree for the dataset using id3 algorithm is
 Outlook
  sunny
 Humidity
    high
     no
    normal
     yes
  rain
 Wind
    strong
     no
    weak
     yes
  overcast
   yes
the test instance: ['rain', 'cool', 'normal', 'strong']
the predicted label: no
the test instance: ['sunny', 'mild', 'normal', 'strong']
the predicted label: yes
```

Program Outcome:

- Understand the implementation procedures for the machine learning algorithms.
- Design Python program for ID3 algorithm.
- The student will be able to demonstrate the working of the decision tree based ID3 algorithm, use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Program 5

5) Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

Program Objective:

- To understand concepts of Artificial Neural Network.
- Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

```
import random

from math import exp

from random import seed

def initialize_network(n_inputs, n_hidden, n_outputs):

    network = list()

    hidden_layer = [{'weights':[random.uniform(-0.5,0.5) for i in range(n_inputs + 1)]} for i in range(n_hidden)]

    network.append(hidden_layer)

    output_layer = [{'weights':[random.uniform(-0.5,0.5) for i in range(n_hidden + 1)]} for i in range(n_outputs)]

    network.append(output_layer)

    return network


def activate(weights, inputs):

    activation = weights[-1]

    for i in range(len(weights)-1):

        activation += weights[i] * inputs[i]

    return activation


def transfer(activation):

    return 1.0 / (1.0 + exp(-activation))
```

```
def forward_propagate(network, row):
    inputs = row
    for layer in network:
        new_inputs = []
        for neuron in layer:
            activation = activate(neuron['weights'], inputs)
            neuron['output'] = transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    return inputs

def transfer_derivative(output):
    return output * (1.0 - output)

def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network)-1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] * neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(expected[j] - neuron['output'])
        for j in range(len(layer)):
```

```
            neuron = layer[j]
            neuron['delta'] = errors[j] * transfer_derivative(neuron['output'])


def update_weights(network, row, l_rate):
    for i in range(len(network)):
        inputs = row[:-1]
        if i != 0:
            inputs = [neuron['output'] for neuron in network[i - 1]]
        for neuron in network[i]:
            for j in range(len(inputs)):
                neuron['weights'][j] += l_rate * neuron['delta'] * inputs[j]
            neuron['weights'][-1] += l_rate * neuron['delta']


def train_network(network, train, l_rate, n_epoch, n_outputs):
        for epoch in range(n_epoch):
            sum_error = 0
            for row in train:
                outputs = forward_propagate(network, row)
                expected = [0 for i in range(n_outputs)]
                expected[row[-1]] = 1
                sum_error += sum([(expected[i]-outputs[i])**2 for i in range(len(expected))])
                backward_propagate_error(network, expected)
                update_weights(network, row, l_rate)
            print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
seed(1)
dataset = [[2.7810836,2.550537003,0],
           [1.465489372,2.362125076,0],
           [3.396561688,4.400293529,0],
           [1.38807019,1.850220317,0],
```

```
        [3.06407232,3.005305973,0],
        [7.627531214,2.759262235,1],
        [5.332441248,2.088626775,1],
        [6.922596716,1.77106367,1],
        [8.675418651,-0.242068655,1],
        [7.673756466,3.508563011,1]]
n_inputs = len(dataset[0]) - 1
n_outputs = len(set([row[-1] for row in dataset]))
network = initialize_network(n_inputs, 2, n_outputs)
print(network)
train_network(network, dataset, 0.5,20, n_outputs)
for layer in network:
    print(layer)
```

Output:

```
[[{'weights': [-0.3656357558875988, 0.3474337369372327, 0.26377461897661403]}, {'weights': [-0.2449309742605783, -0.00456
4912908059049, -0.05050893521126187 4]}], [{'weights': [0.15159297272276295, 0.2887233511355132, -0.4061404132257651]}, {'
weights': [-0.4716525234779937, 0.3357651039198697, -0.06723293209494663]}]]
>epoch=0, lrate=0.500, error=4.763
>epoch=1, lrate=0.500, error=4.558
>epoch=2, lrate=0.500, error=4.316
>epoch=3, lrate=0.500, error=4.035
>epoch=4, lrate=0.500, error=3.733
>epoch=5, lrate=0.500, error=3.428
>epoch=6, lrate=0.500, error=3.132
>epoch=7, lrate=0.500, error=2.850
>epoch=8, lrate=0.500, error=2.588
>epoch=9, lrate=0.500, error=2.348
>epoch=10, lrate=0.500, error=2.128
>epoch=11, lrate=0.500, error=1.931
>epoch=12, lrate=0.500, error=1.753
>epoch=13, lrate=0.500, error=1.595
>epoch=14, lrate=0.500, error=1.454
>epoch=15, lrate=0.500, error=1.329
>epoch=16, lrate=0.500, error=1.218
>epoch=17, lrate=0.500, error=1.120
>epoch=18, lrate=0.500, error=1.033
>epoch=19, lrate=0.500, error=0.956
[{'weights': [-1.435239043819221, 1.8587338175173547, 0.7917644224148094], 'output': 0.029795197360175857, 'delta': -0.00
6018730117768358}, {'weights': [-0.7704959899742789, 0.8257894037467045, 0.21154103288579731], 'output': 0.06771641538441
577, 'delta': -0.00502558510232048}]
[{'weights': [2.223584933362892, 1.2428928053374768, -1.3519548925527454], 'output': 0.23499833662766154, 'delta': -0.042
246618795029306}, {'weights': [-2.509732251870173, -0.5925943219491905, 1.259965727484093], 'output': 0.7543931062537561,
'delta': 0.0455070639255786 2}]
```

Program Outcome:

- Design and develop program to Back propagation algorithm.
- The student will be able to build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

Program 6

6) Write a program to implement the Naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Program Objective:

- To implement the Naive Bayesian classifier.
- To compute the accuracy of the classifier

```
import csv,random,math
import statistics as st
def loadCsv(filename):
    lines=csv.reader(open(filename,"r"));
    dataset=list(lines)
    for i in range(len(dataset)):
        dataset[i]=[float(x) for x in dataset[i]]
    return dataset
def splitDataset(dataset,splitRatio):
    testSize=int(len(dataset)*splitRatio);
    trainSet=list(dataset);
    testSet=[]
    while len(testSet)<testSize:
        index=random.randrange(len(trainSet));
        testSet.append(trainSet.pop(index))
    return [trainSet,testSet]
def separateByClass(dataset):
    separated={}
    for i in range(len(dataset)):
        x=dataset[i]
        if (x[-1] not in separated):
            separated[x[-1]]=[]
        separated[x[-1]].append(x)
```

```
    return separated
def compute_mean(dataset):
    mean_std=[(st.mean(attribute),st.stdev(attribute)) for attribute in zip(*dataset)];
    del mean_std[-1]
    return mean_std
def summarizeByClass(dataset):
    separated=separateByClass(dataset);
    summary={}
    for classValues,instances in separated.items():
        summary[classValues]=compute_mean(instances)
    return summary
def estimateProbability(x,mean,stdev):
    exponent=math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
    return (1/(math.sqrt(2*math.pi)*stdev))*exponent
def calculateClassProbabilities(summaries,testVector):
    p={}
    for classValue,classSummaries in summaries.items():
        p[classValue]=1
        for i in range(len(classSummaries)):
            mean,stdev=classSummaries[i]
            x=testVector[i]
            p[classValue]*=estimateProbability(x,mean,stdev);
    return p
def predict(summaries,testVector):
    all_p=calculateClassProbabilities(summaries,testVector)
    bestLabel,bestProb=None,-1
    for lbl,p in all_p.items():
        if bestLabel is None or p>bestProb:
            bestProb=p
```

```
            bestLabel=lbl
    return bestLabel
def perform_classification(summaries,testSet):
    predictions=[]
    for i in range(len(testSet)):
        result=predict(summaries,testSet[i])
        predictions.append(result)
    return predictions
def getAccuracy(testSet,predictions):
    correct=0
    for i in range(len(testSet)):
        if testSet[i][-1]==predictions[i]:
            correct+=1
    return (correct/float(len(testSet)))*100.0
dataset=loadCsv('pima-indians-diabetes.csv');
print('pima indian diabetes dataset loaded....')
print('total instances available:',len(dataset))
print('total attributes present:',len(dataset[0])-1)
print("first five instances of dataset:")
for i in range(5):
    print(i+1,':',dataset[i])
splitRatio=0.2
trainingSet,testSet=splitDataset(dataset,splitRatio)
print("\n dataset is split into training and testing set")
print('training examples={0}\n testing examples={1}'.format(len(trainingSet),len(testSet)))
summaries=summarizeByClass(trainingSet);
predictions=perform_classification(summaries,testSet)
accuracy=getAccuracy(testSet,predictions)
print('\n Accuracy of the naive bayssive classifier is:',accuracy)
```

Data Set: pima-indians-diabetes.csv

6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1
1	97	66	15	140	23.2	0.487	22	0
13	145	82	19	110	22.2	0.245	57	0
5	117	92	0	0	34.1	0.337	38	0
5	109	75	26	0	36	0.546	60	0
3	158	76	36	245	31.6	0.851	28	1
3	88	58	11	54	24.8	0.267	22	0
6	92	92	0	0	19.9	0.188	28	0
10	122	78	31	0	27.6	0.512	45	0
4	103	60	33	192	24	0.966	33	0
11	138	76	0	0	33.2	0.42	35	0
9	102	76	37	0	32.9	0.665	46	1
2	90	68	42	0	38.2	0.503	27	1
4	111	72	47	207	37.1	1.39	56	1

3	180	64	25	70	34	0.271	26	0
7	133	84	0	0	40.2	0.696	37	0
7	106	92	18	0	22.7	0.235	48	0
9	171	110	24	240	45.4	0.721	54	1
7	159	64	0	0	27.4	0.294	40	0
0	180	66	39	0	42	1.893	25	1
1	146	56	0	0	29.7	0.564	29	0
2	71	70	27	0	28	0.586	22	0
7	103	66	32	0	39.1	0.344	31	1
7	105	0	0	0	0	0.305	24	0
1	103	80	11	82	19.4	0.491	22	0
1	101	50	15	36	24.2	0.526	26	0
5	88	66	21	23	24.4	0.342	30	0
8	176	90	34	300	33.7	0.467	58	1
7	150	66	42	342	34.7	0.718	42	0
1	73	50	10	0	23	0.248	21	0
7	187	68	39	304	37.7	0.254	41	1
0	100	88	60	110	46.8	0.962	31	0
0	146	82	0	0	40.5	1.781	44	0
0	105	64	41	142	41.5	0.173	22	0
2	84	0	0	0	0	0.304	21	0
8	133	72	0	0	32.9	0.27	39	1
5	44	62	0	0	25	0.587	36	0
2	141	58	34	128	25.4	0.699	24	0
7	114	66	0	0	32.8	0.258	42	1
5	99	74	27	0	29	0.203	32	0
0	109	88	30	0	32.5	0.855	38	1
2	109	92	0	0	42.7	0.845	54	0
1	95	66	13	38	19.6	0.334	25	0
4	146	85	27	100	28.9	0.189	27	0
2	100	66	20	90	32.9	0.867	28	1
5	139	64	35	140	28.6	0.411	26	0
13	126	90	0	0	43.4	0.583	42	1
4	129	86	20	270	35.1	0.231	23	0
1	79	75	30	0	32	0.396	22	0
1	0	48	20	0	24.7	0.14	22	0
7	62	78	0	0	32.6	0.391	41	0
5	95	72	33	0	37.7	0.37	27	0
0	131	0	0	0	43.2	0.27	26	1
2	112	66	22	0	25	0.307	24	0
3	113	44	13	0	22.4	0.14	22	0
2	74	0	0	0	0	0.102	22	0
7	83	78	26	71	29.3	0.767	36	0
0	101	65	28	0	24.6	0.237	22	0

5	137	108	0	0	48.8	0.227	37	1
2	110	74	29	125	32.4	0.698	27	0
13	106	72	54	0	36.6	0.178	45	0
2	100	68	25	71	38.5	0.324	26	0
15	136	70	32	110	37.1	0.153	43	1
1	107	68	19	0	26.5	0.165	24	0
1	80	55	0	0	19.1	0.258	21	0
4	123	80	15	176	32	0.443	34	0
7	81	78	40	48	46.7	0.261	42	0
4	134	72	0	0	23.8	0.277	60	1
2	142	82	18	64	24.7	0.761	21	0
6	144	72	27	228	33.9	0.255	40	0
2	92	62	28	0	31.6	0.13	24	0
1	71	48	18	76	20.4	0.323	22	0
6	93	50	30	64	28.7	0.356	23	0
1	122	90	51	220	49.7	0.325	31	1
1	163	72	0	0	39	1.222	33	1
1	151	60	0	0	26.1	0.179	22	0
0	125	96	0	0	22.5	0.262	21	0
1	81	72	18	40	26.6	0.283	24	0
2	85	65	0	0	39.6	0.93	27	0
1	126	56	29	152	28.7	0.801	21	0
1	96	122	0	0	22.4	0.207	27	0
4	144	58	28	140	29.5	0.287	37	0
3	83	58	31	18	34.3	0.336	25	0
0	95	85	25	36	37.4	0.247	24	1
3	171	72	33	135	33.3	0.199	24	1
8	155	62	26	495	34	0.543	46	1
1	89	76	34	37	31.2	0.192	23	0
4	76	62	0	0	34	0.391	25	0
7	160	54	32	175	30.5	0.588	39	1
4	146	92	0	0	31.2	0.539	61	1
5	124	74	0	0	34	0.22	38	1
5	78	48	0	0	33.7	0.654	25	0
4	97	60	23	0	28.2	0.443	22	0
4	99	76	15	51	23.2	0.223	21	0
0	162	76	56	100	53.2	0.759	25	1
6	111	64	39	0	34.2	0.26	24	0
2	107	74	30	100	33.6	0.404	23	0
5	132	80	0	0	26.8	0.186	69	0
0	113	76	0	0	33.3	0.278	23	1
1	88	30	42	99	55	0.496	26	1
3	120	70	30	135	42.9	0.452	30	0
1	118	58	36	94	33.3	0.261	23	0

1	117	88	24	145	34.5	0.403	40	1
0	105	84	0	0	27.9	0.741	62	1
4	173	70	14	168	29.7	0.361	33	1
9	122	56	0	0	33.3	1.114	33	1
3	170	64	37	225	34.5	0.356	30	1
8	84	74	31	0	38.3	0.457	39	0
2	96	68	13	49	21.1	0.647	26	0
2	125	60	20	140	33.8	0.088	31	0
0	100	70	26	50	30.8	0.597	21	0
0	93	60	25	92	28.7	0.532	22	0
0	129	80	0	0	31.2	0.703	29	0
5	105	72	29	325	36.9	0.159	28	0
3	128	78	0	0	21.1	0.268	55	0
5	106	82	30	0	39.5	0.286	38	0
2	108	52	26	63	32.5	0.318	22	0
10	108	66	0	0	32.4	0.272	42	1
4	154	62	31	284	32.8	0.237	23	0
0	102	75	23	0	0	0.572	21	0
9	57	80	37	0	32.8	0.096	41	0
2	106	64	35	119	30.5	1.4	34	0
5	147	78	0	0	33.7	0.218	65	0
2	90	70	17	0	27.3	0.085	22	0
1	136	74	50	204	37.4	0.399	24	0
4	114	65	0	0	21.9	0.432	37	0
9	156	86	28	155	34.3	1.189	42	1
1	153	82	42	485	40.6	0.687	23	0
8	188	78	0	0	47.9	0.137	43	1
7	152	88	44	0	50	0.337	36	1
2	99	52	15	94	24.6	0.637	21	0
1	109	56	21	135	25.2	0.833	23	0
2	88	74	19	53	29	0.229	22	0
17	163	72	41	114	40.9	0.817	47	1
4	151	90	38	0	29.7	0.294	36	0
7	102	74	40	105	37.2	0.204	45	0
0	114	80	34	285	44.2	0.167	27	0
2	100	64	23	0	29.7	0.368	21	0
0	131	88	0	0	31.6	0.743	32	1
6	104	74	18	156	29.9	0.722	41	1
3	148	66	25	0	32.5	0.256	22	0
4	120	68	0	0	29.6	0.709	34	0
4	110	66	0	0	31.9	0.471	29	0
3	111	90	12	78	28.4	0.495	29	0
6	102	82	0	0	30.8	0.18	36	1
6	134	70	23	130	35.4	0.542	29	1

2	87	0	23	0	28.9	0.773	25	0
1	79	60	42	48	43.5	0.678	23	0
2	75	64	24	55	29.7	0.37	33	0
8	179	72	42	130	32.7	0.719	36	1
6	85	78	0	0	31.2	0.382	42	0
0	129	110	46	130	67.1	0.319	26	1
5	143	78	0	0	45	0.19	47	0
5	130	82	0	0	39.1	0.956	37	1
6	87	80	0	0	23.2	0.084	32	0
0	119	64	18	92	34.9	0.725	23	0
1	0	74	20	23	27.7	0.299	21	0
5	73	60	0	0	26.8	0.268	27	0
4	141	74	0	0	27.6	0.244	40	0
7	194	68	28	0	35.9	0.745	41	1
8	181	68	36	495	30.1	0.615	60	1
1	128	98	41	58	32	1.321	33	1
8	109	76	39	114	27.9	0.64	31	1
5	139	80	35	160	31.6	0.361	25	1
3	111	62	0	0	22.6	0.142	21	0
9	123	70	44	94	33.1	0.374	40	0
7	159	66	0	0	30.4	0.383	36	1
11	135	0	0	0	52.3	0.578	40	1
8	85	55	20	0	24.4	0.136	42	0
5	158	84	41	210	39.4	0.395	29	1
1	105	58	0	0	24.3	0.187	21	0
3	107	62	13	48	22.9	0.678	23	1
4	109	64	44	99	34.8	0.905	26	1
4	148	60	27	318	30.9	0.15	29	1
0	113	80	16	0	31	0.874	21	0
1	138	82	0	0	40.1	0.236	28	0
0	108	68	20	0	27.3	0.787	32	0
2	99	70	16	44	20.4	0.235	27	0
6	103	72	32	190	37.7	0.324	55	0
5	111	72	28	0	23.9	0.407	27	0
8	196	76	29	280	37.5	0.605	57	1
5	162	104	0	0	37.7	0.151	52	1
1	96	64	27	87	33.2	0.289	21	0
7	184	84	33	0	35.5	0.355	41	1
2	81	60	22	0	27.7	0.29	25	0
0	147	85	54	0	42.8	0.375	24	0
7	179	95	31	0	34.2	0.164	60	0
0	140	65	26	130	42.6	0.431	24	1
9	112	82	32	175	34.2	0.26	36	1
12	151	70	40	271	41.8	0.742	38	1

5	109	62	41	129	35.8	0.514	25	1
6	125	68	30	120	30	0.464	32	0
5	85	74	22	0	29	1.224	32	1
5	112	66	0	0	37.8	0.261	41	1
0	177	60	29	478	34.6	1.072	21	1
2	158	90	0	0	31.6	0.805	66	1
7	119	0	0	0	25.2	0.209	37	0
7	142	60	33	190	28.8	0.687	61	0
1	100	66	15	56	23.6	0.666	26	0
1	87	78	27	32	34.6	0.101	22	0
0	101	76	0	0	35.7	0.198	26	0
3	162	52	38	0	37.2	0.652	24	1
4	197	70	39	744	36.7	2.329	31	0
0	117	80	31	53	45.2	0.089	24	0
4	142	86	0	0	44	0.645	22	1
6	134	80	37	370	46.2	0.238	46	1
1	79	80	25	37	25.4	0.583	22	0
4	122	68	0	0	35	0.394	29	0
3	74	68	28	45	29.7	0.293	23	0
4	171	72	0	0	43.6	0.479	26	1
7	181	84	21	192	35.9	0.586	51	1
0	179	90	27	0	44.1	0.686	23	1
9	164	84	21	0	30.8	0.831	32	1
0	104	76	0	0	18.4	0.582	27	0
1	91	64	24	0	29.2	0.192	21	0
4	91	70	32	88	33.1	0.446	22	0
3	139	54	0	0	25.6	0.402	22	1
6	119	50	22	176	27.1	1.318	33	1
2	146	76	35	194	38.2	0.329	29	0
9	184	85	15	0	30	1.213	49	1
10	122	68	0	0	31.2	0.258	41	0
0	165	90	33	680	52.3	0.427	23	0
9	124	70	33	402	35.4	0.282	34	0
1	111	86	19	0	30.1	0.143	23	0
9	106	52	0	0	31.2	0.38	42	0
2	129	84	0	0	28	0.284	27	0
2	90	80	14	55	24.4	0.249	24	0
0	86	68	32	0	35.8	0.238	25	0
12	92	62	7	258	27.6	0.926	44	1
1	113	64	35	0	33.6	0.543	21	1
3	111	56	39	0	30.1	0.557	30	0
2	114	68	22	0	28.7	0.092	25	0
1	193	50	16	375	25.9	0.655	24	0
11	155	76	28	150	33.3	1.353	51	1

3	191	68	15	130	30.9	0.299	34	0
3	141	0	0	0	30	0.761	27	1
4	95	70	32	0	32.1	0.612	24	0
3	142	80	15	0	32.4	0.2	63	0
4	123	62	0	0	32	0.226	35	1
5	96	74	18	67	33.6	0.997	43	0
0	138	0	0	0	36.3	0.933	25	1
2	128	64	42	0	40	1.101	24	0
0	102	52	0	0	25.1	0.078	21	0
2	146	0	0	0	27.5	0.24	28	1
10	101	86	37	0	45.6	1.136	38	1
2	108	62	32	56	25.2	0.128	21	0
3	122	78	0	0	23	0.254	40	0
1	71	78	50	45	33.2	0.422	21	0
13	106	70	0	0	34.2	0.251	52	0
2	100	70	52	57	40.5	0.677	25	0
7	106	60	24	0	26.5	0.296	29	1
0	104	64	23	116	27.8	0.454	23	0
5	114	74	0	0	24.9	0.744	57	0
2	108	62	10	278	25.3	0.881	22	0
0	146	70	0	0	37.9	0.334	28	1
10	129	76	28	122	35.9	0.28	39	0
7	133	88	15	155	32.4	0.262	37	0
7	161	86	0	0	30.4	0.165	47	1
2	108	80	0	0	27	0.259	52	1
7	136	74	26	135	26	0.647	51	0
5	155	84	44	545	38.7	0.619	34	0
1	119	86	39	220	45.6	0.808	29	1
4	96	56	17	49	20.8	0.34	26	0
5	108	72	43	75	36.1	0.263	33	0
0	78	88	29	40	36.9	0.434	21	0
0	107	62	30	74	36.6	0.757	25	1
2	128	78	37	182	43.3	1.224	31	1
1	128	48	45	194	40.5	0.613	24	1
0	161	50	0	0	21.9	0.254	65	0
6	151	62	31	120	35.5	0.692	28	0
2	146	70	38	360	28	0.337	29	1
0	126	84	29	215	30.7	0.52	24	0
14	100	78	25	184	36.6	0.412	46	1
8	112	72	0	0	23.6	0.84	58	0
0	167	0	0	0	32.3	0.839	30	1
2	144	58	33	135	31.6	0.422	25	1
5	77	82	41	42	35.8	0.156	35	0
5	115	98	0	0	52.9	0.209	28	1

3	150	76	0	0	21	0.207	37	0
2	120	76	37	105	39.7	0.215	29	0
10	161	68	23	132	25.5	0.326	47	1
0	137	68	14	148	24.8	0.143	21	0
0	128	68	19	180	30.5	1.391	25	1
2	124	68	28	205	32.9	0.875	30	1
6	80	66	30	0	26.2	0.313	41	0
0	106	70	37	148	39.4	0.605	22	0
2	155	74	17	96	26.6	0.433	27	1
3	113	50	10	85	29.5	0.626	25	0
7	109	80	31	0	35.9	1.127	43	1
2	112	68	22	94	34.1	0.315	26	0
3	99	80	11	64	19.3	0.284	30	0
3	182	74	0	0	30.5	0.345	29	1
3	115	66	39	140	38.1	0.15	28	0
6	194	78	0	0	23.5	0.129	59	1
4	129	60	12	231	27.5	0.527	31	0
3	112	74	30	0	31.6	0.197	25	1
0	124	70	20	0	27.4	0.254	36	1
13	152	90	33	29	26.8	0.731	43	1
2	112	75	32	0	35.7	0.148	21	0
1	157	72	21	168	25.6	0.123	24	0
1	122	64	32	156	35.1	0.692	30	1
10	179	70	0	0	35.1	0.2	37	0
2	102	86	36	120	45.5	0.127	23	1
6	105	70	32	68	30.8	0.122	37	0
8	118	72	19	0	23.1	1.476	46	0
2	87	58	16	52	32.7	0.166	25	0
1	180	0	0	0	43.3	0.282	41	1
12	106	80	0	0	23.6	0.137	44	0
1	95	60	18	58	23.9	0.26	22	0
0	165	76	43	255	47.9	0.259	26	0
0	117	0	0	0	33.8	0.932	44	0
5	115	76	0	0	31.2	0.343	44	1
9	152	78	34	171	34.2	0.893	33	1
7	178	84	0	0	39.9	0.331	41	1
1	130	70	13	105	25.9	0.472	22	0
1	95	74	21	73	25.9	0.673	36	0
1	0	68	35	0	32	0.389	22	0
5	122	86	0	0	34.7	0.29	33	0
8	95	72	0	0	36.8	0.485	57	0
8	126	88	36	108	38.5	0.349	49	0
1	139	46	19	83	28.7	0.654	22	0
3	116	0	0	0	23.5	0.187	23	0

3	99	62	19	74	21.8	0.279	26	0
5	0	80	32	0	41	0.346	37	1
4	92	80	0	0	42.2	0.237	29	0
4	137	84	0	0	31.2	0.252	30	0
3	61	82	28	0	34.4	0.243	46	0
1	90	62	12	43	27.2	0.58	24	0
3	90	78	0	0	42.7	0.559	21	0
9	165	88	0	0	30.4	0.302	49	1
1	125	50	40	167	33.3	0.962	28	1
13	129	0	30	0	39.9	0.569	44	1
12	88	74	40	54	35.3	0.378	48	0
1	196	76	36	249	36.5	0.875	29	1
5	189	64	33	325	31.2	0.583	29	1
5	158	70	0	0	29.8	0.207	63	0
5	103	108	37	0	39.2	0.305	65	0
4	146	78	0	0	38.5	0.52	67	1
4	147	74	25	293	34.9	0.385	30	0
5	99	54	28	83	34	0.499	30	0
6	124	72	0	0	27.6	0.368	29	1
0	101	64	17	0	21	0.252	21	0
3	81	86	16	66	27.5	0.306	22	0
1	133	102	28	140	32.8	0.234	45	1
3	173	82	48	465	38.4	2.137	25	1
0	118	64	23	89	0	1.731	21	0
0	84	64	22	66	35.8	0.545	21	0
2	105	58	40	94	34.9	0.225	25	0
2	122	52	43	158	36.2	0.816	28	0
12	140	82	43	325	39.2	0.528	58	1
0	98	82	15	84	25.2	0.299	22	0
1	87	60	37	75	37.2	0.509	22	0
4	156	75	0	0	48.3	0.238	32	1
0	93	100	39	72	43.4	1.021	35	0
1	107	72	30	82	30.8	0.821	24	0
0	105	68	22	0	20	0.236	22	0
1	109	60	8	182	25.4	0.947	21	0
1	90	62	18	59	25.1	1.268	25	0
1	125	70	24	110	24.3	0.221	25	0
1	119	54	13	50	22.3	0.205	24	0
5	116	74	29	0	32.3	0.66	35	1
8	105	100	36	0	43.3	0.239	45	1
5	144	82	26	285	32	0.452	58	1
3	100	68	23	81	31.6	0.949	28	0
1	100	66	29	196	32	0.444	42	0
5	166	76	0	0	45.7	0.34	27	1

1	131	64	14	415	23.7	0.389	21	0
4	116	72	12	87	22.1	0.463	37	0
4	158	78	0	0	32.9	0.803	31	1
2	127	58	24	275	27.7	1.6	25	0
3	96	56	34	115	24.7	0.944	39	0
0	131	66	40	0	34.3	0.196	22	1
3	82	70	0	0	21.1	0.389	25	0
3	193	70	31	0	34.9	0.241	25	1
4	95	64	0	0	32	0.161	31	1
6	137	61	0	0	24.2	0.151	55	0
5	136	84	41	88	35	0.286	35	1
9	72	78	25	0	31.6	0.28	38	0
5	168	64	0	0	32.9	0.135	41	1
2	123	48	32	165	42.1	0.52	26	0
4	115	72	0	0	28.9	0.376	46	1
0	101	62	0	0	21.9	0.336	25	0
8	197	74	0	0	25.9	1.191	39	1
1	172	68	49	579	42.4	0.702	28	1
6	102	90	39	0	35.7	0.674	28	0
1	112	72	30	176	34.4	0.528	25	0
1	143	84	23	310	42.4	1.076	22	0
1	143	74	22	61	26.2	0.256	21	0
0	138	60	35	167	34.6	0.534	21	1
3	173	84	33	474	35.7	0.258	22	1
1	97	68	21	0	27.2	1.095	22	0
4	144	82	32	0	38.5	0.554	37	1
1	83	68	0	0	18.2	0.624	27	0
3	129	64	29	115	26.4	0.219	28	1
1	119	88	41	170	45.3	0.507	26	0
2	94	68	18	76	26	0.561	21	0
0	102	64	46	78	40.6	0.496	21	0
2	115	64	22	0	30.8	0.421	21	0
8	151	78	32	210	42.9	0.516	36	1
4	184	78	39	277	37	0.264	31	1
0	94	0	0	0	0	0.256	25	0
1	181	64	30	180	34.1	0.328	38	1
0	135	94	46	145	40.6	0.284	26	0
1	95	82	25	180	35	0.233	43	1
2	99	0	0	0	22.2	0.108	23	0
3	89	74	16	85	30.4	0.551	38	0
1	80	74	11	60	30	0.527	22	0
2	139	75	0	0	25.6	0.167	29	0
1	90	68	8	0	24.5	1.138	36	0
0	141	0	0	0	42.4	0.205	29	1

12	140	85	33	0	37.4	0.244	41	0
5	147	75	0	0	29.9	0.434	28	0
1	97	70	15	0	18.2	0.147	21	0
6	107	88	0	0	36.8	0.727	31	0
0	189	104	25	0	34.3	0.435	41	1
2	83	66	23	50	32.2	0.497	22	0
4	117	64	27	120	33.2	0.23	24	0
8	108	70	0	0	30.5	0.955	33	1
4	117	62	12	0	29.7	0.38	30	1
0	180	78	63	14	59.4	2.42	25	1
1	100	72	12	70	25.3	0.658	28	0
0	95	80	45	92	36.5	0.33	26	0
0	104	64	37	64	33.6	0.51	22	1
0	120	74	18	63	30.5	0.285	26	0
1	82	64	13	95	21.2	0.415	23	0
2	134	70	0	0	28.9	0.542	23	1
0	91	68	32	210	39.9	0.381	25	0
2	119	0	0	0	19.6	0.832	72	0
2	100	54	28	105	37.8	0.498	24	0
14	175	62	30	0	33.6	0.212	38	1
1	135	54	0	0	26.7	0.687	62	0
5	86	68	28	71	30.2	0.364	24	0
10	148	84	48	237	37.6	1.001	51	1
9	134	74	33	60	25.9	0.46	81	0
9	120	72	22	56	20.8	0.733	48	0
1	71	62	0	0	21.8	0.416	26	0
8	74	70	40	49	35.3	0.705	39	0
5	88	78	30	0	27.6	0.258	37	0
10	115	98	0	0	24	1.022	34	0
0	124	56	13	105	21.8	0.452	21	0
0	74	52	10	36	27.8	0.269	22	0
0	97	64	36	100	36.8	0.6	25	0
8	120	0	0	0	30	0.183	38	1
6	154	78	41	140	46.1	0.571	27	0
1	144	82	40	0	41.3	0.607	28	0
0	137	70	38	0	33.2	0.17	22	0
0	119	66	27	0	38.8	0.259	22	0
7	136	90	0	0	29.9	0.21	50	0
4	114	64	0	0	28.9	0.126	24	0
0	137	84	27	0	27.3	0.231	59	0
2	105	80	45	191	33.7	0.711	29	1
7	114	76	17	110	23.8	0.466	31	0
8	126	74	38	75	25.9	0.162	39	0
4	132	86	31	0	28	0.419	63	0

3	158	70	30	328	35.5	0.344	35	1
0	123	88	37	0	35.2	0.197	29	0
4	85	58	22	49	27.8	0.306	28	0
0	84	82	31	125	38.2	0.233	23	0
0	145	0	0	0	44.2	0.63	31	1
0	135	68	42	250	42.3	0.365	24	1
1	139	62	41	480	40.7	0.536	21	0
0	173	78	32	265	46.5	1.159	58	0
4	99	72	17	0	25.6	0.294	28	0
8	194	80	0	0	26.1	0.551	67	0
2	83	65	28	66	36.8	0.629	24	0
2	89	90	30	0	33.5	0.292	42	0
4	99	68	38	0	32.8	0.145	33	0
4	125	70	18	122	28.9	1.144	45	1
3	80	0	0	0	0	0.174	22	0
6	166	74	0	0	26.6	0.304	66	0
5	110	68	0	0	26	0.292	30	0
2	81	72	15	76	30.1	0.547	25	0
7	195	70	33	145	25.1	0.163	55	1
6	154	74	32	193	29.3	0.839	39	0
2	117	90	19	71	25.2	0.313	21	0
3	84	72	32	0	37.2	0.267	28	0
6	0	68	41	0	39	0.727	41	1
7	94	64	25	79	33.3	0.738	41	0
3	96	78	39	0	37.3	0.238	40	0
10	75	82	0	0	33.3	0.263	38	0
0	180	90	26	90	36.5	0.314	35	1
1	130	60	23	170	28.6	0.692	21	0
2	84	50	23	76	30.4	0.968	21	0
8	120	78	0	0	25	0.409	64	0
12	84	72	31	0	29.7	0.297	46	1
0	139	62	17	210	22.1	0.207	21	0
9	91	68	0	0	24.2	0.2	58	0
2	91	62	0	0	27.3	0.525	22	0
3	99	54	19	86	25.6	0.154	24	0
3	163	70	18	105	31.6	0.268	28	1
9	145	88	34	165	30.3	0.771	53	1
7	125	86	0	0	37.6	0.304	51	0
13	76	60	0	0	32.8	0.18	41	0
6	129	90	7	326	19.6	0.582	60	0
2	68	70	32	66	25	0.187	25	0
3	124	80	33	130	33.2	0.305	26	0
6	114	0	0	0	0	0.189	26	0
9	130	70	0	0	34.2	0.652	45	1

3	125	58	0	0	31.6	0.151	24	0
3	87	60	18	0	21.8	0.444	21	0
1	97	64	19	82	18.2	0.299	21	0
3	116	74	15	105	26.3	0.107	24	0
0	117	66	31	188	30.8	0.493	22	0
0	111	65	0	0	24.6	0.66	31	0
2	122	60	18	106	29.8	0.717	22	0
0	107	76	0	0	45.3	0.686	24	0
1	86	66	52	65	41.3	0.917	29	0
6	91	0	0	0	29.8	0.501	31	0
1	77	56	30	56	33.3	1.251	24	0
4	132	0	0	0	32.9	0.302	23	1
0	105	90	0	0	29.6	0.197	46	0
0	57	60	0	0	21.7	0.735	67	0
0	127	80	37	210	36.3	0.804	23	0
3	129	92	49	155	36.4	0.968	32	1
8	100	74	40	215	39.4	0.661	43	1
3	128	72	25	190	32.4	0.549	27	1
10	90	85	32	0	34.9	0.825	56	1
4	84	90	23	56	39.5	0.159	25	0
1	88	78	29	76	32	0.365	29	0
8	186	90	35	225	34.5	0.423	37	1
5	187	76	27	207	43.6	1.034	53	1
4	131	68	21	166	33.1	0.16	28	0
1	164	82	43	67	32.8	0.341	50	0
4	189	110	31	0	28.5	0.68	37	0
1	116	70	28	0	27.4	0.204	21	0
3	84	68	30	106	31.9	0.591	25	0
6	114	88	0	0	27.8	0.247	66	0
1	88	62	24	44	29.9	0.422	23	0
1	84	64	23	115	36.9	0.471	28	0
7	124	70	33	215	25.5	0.161	37	0
1	97	70	40	0	38.1	0.218	30	0
8	110	76	0	0	27.8	0.237	58	0
11	103	68	40	0	46.2	0.126	42	0
11	85	74	0	0	30.1	0.3	35	0
6	125	76	0	0	33.8	0.121	54	1
0	198	66	32	274	41.3	0.502	28	1
1	87	68	34	77	37.6	0.401	24	0
6	99	60	19	54	26.9	0.497	32	0
0	91	80	0	0	32.4	0.601	27	0
2	95	54	14	88	26.1	0.748	22	0
1	99	72	30	18	38.6	0.412	21	0
6	92	62	32	126	32	0.085	46	0

4	154	72	29	126	31.3	0.338	37	0
0	121	66	30	165	34.3	0.203	33	1
3	78	70	0	0	32.5	0.27	39	0
2	130	96	0	0	22.6	0.268	21	0
3	111	58	31	44	29.5	0.43	22	0
2	98	60	17	120	34.7	0.198	22	0
1	143	86	30	330	30.1	0.892	23	0
1	119	44	47	63	35.5	0.28	25	0
6	108	44	20	130	24	0.813	35	0
2	118	80	0	0	42.9	0.693	21	1
10	133	68	0	0	27	0.245	36	0
2	197	70	99	0	34.7	0.575	62	1
0	151	90	46	0	42.1	0.371	21	1
6	109	60	27	0	25	0.206	27	0
12	121	78	17	0	26.5	0.259	62	0
8	100	76	0	0	38.7	0.19	42	0
8	124	76	24	600	28.7	0.687	52	1
1	93	56	11	0	22.5	0.417	22	0
8	143	66	0	0	34.9	0.129	41	1
6	103	66	0	0	24.3	0.249	29	0
3	176	86	27	156	33.3	1.154	52	1
0	73	0	0	0	21.1	0.342	25	0
11	111	84	40	0	46.8	0.925	45	1
2	112	78	50	140	39.4	0.175	24	0
3	132	80	0	0	34.4	0.402	44	1
2	82	52	22	115	28.5	1.699	25	0
6	123	72	45	230	33.6	0.733	34	0
0	188	82	14	185	32	0.682	22	1
0	67	76	0	0	45.3	0.194	46	0
1	89	24	19	25	27.8	0.559	21	0
1	173	74	0	0	36.8	0.088	38	1
1	109	38	18	120	23.1	0.407	26	0
1	108	88	19	0	27.1	0.4	24	0
6	96	0	0	0	23.7	0.19	28	0
1	124	74	36	0	27.8	0.1	30	0
7	150	78	29	126	35.2	0.692	54	1
4	183	0	0	0	28.4	0.212	36	1
1	124	60	32	0	35.8	0.514	21	0
1	181	78	42	293	40	1.258	22	1
1	92	62	25	41	19.5	0.482	25	0
0	152	82	39	272	41.5	0.27	27	0
1	111	62	13	182	24	0.138	23	0
3	106	54	21	158	30.9	0.292	24	0
3	174	58	22	194	32.9	0.593	36	1

7	168	88	42	321	38.2	0.787	40	1
6	105	80	28	0	32.5	0.878	26	0
11	138	74	26	144	36.1	0.557	50	1
3	106	72	0	0	25.8	0.207	27	0
6	117	96	0	0	28.7	0.157	30	0
2	68	62	13	15	20.1	0.257	23	0
9	112	82	24	0	28.2	1.282	50	1
0	119	0	0	0	32.4	0.141	24	1
2	112	86	42	160	38.4	0.246	28	0
2	92	76	20	0	24.2	1.698	28	0
6	183	94	0	0	40.8	1.461	45	0
0	94	70	27	115	43.5	0.347	21	0
2	108	64	0	0	30.8	0.158	21	0
4	90	88	47	54	37.7	0.362	29	0
0	125	68	0	0	24.7	0.206	21	0
0	132	78	0	0	32.4	0.393	21	0
5	128	80	0	0	34.6	0.144	45	0
4	94	65	22	0	24.7	0.148	21	0
7	114	64	0	0	27.4	0.732	34	1
0	102	78	40	90	34.5	0.238	24	0
2	111	60	0	0	26.2	0.343	23	0
1	128	82	17	183	27.5	0.115	22	0
10	92	62	0	0	25.9	0.167	31	0
13	104	72	0	0	31.2	0.465	38	1
5	104	74	0	0	28.8	0.153	48	0
2	94	76	18	66	31.6	0.649	23	0
7	97	76	32	91	40.9	0.871	32	1
1	100	74	12	46	19.5	0.149	28	0
0	102	86	17	105	29.3	0.695	27	0
4	128	70	0	0	34.3	0.303	24	0
6	147	80	0	0	29.5	0.178	50	1
4	90	0	0	0	28	0.61	31	0
3	103	72	30	152	27.6	0.73	27	0
2	157	74	35	440	39.4	0.134	30	0
1	167	74	17	144	23.4	0.447	33	1
0	179	50	36	159	37.8	0.455	22	1
11	136	84	35	130	28.3	0.26	42	1
0	107	60	25	0	26.4	0.133	23	0
1	91	54	25	100	25.2	0.234	23	0
1	117	60	23	106	33.8	0.466	27	0
5	123	74	40	77	34.1	0.269	28	0
2	120	54	0	0	26.8	0.455	27	0
1	106	70	28	135	34.2	0.142	22	0
2	155	52	27	540	38.7	0.24	25	1

2	101	58	35	90	21.8	0.155	22	0
1	120	80	48	200	38.9	1.162	41	0
11	127	106	0	0	39	0.19	51	0
3	80	82	31	70	34.2	1.292	27	1
10	162	84	0	0	27.7	0.182	54	0
1	199	76	43	0	42.9	1.394	22	1
8	167	106	46	231	37.6	0.165	43	1
9	145	80	46	130	37.9	0.637	40	1
6	115	60	39	0	33.7	0.245	40	1
1	112	80	45	132	34.8	0.217	24	0
4	145	82	18	0	32.5	0.235	70	1
10	111	70	27	0	27.5	0.141	40	1
6	98	58	33	190	34	0.43	43	0
9	154	78	30	100	30.9	0.164	45	0
6	165	68	26	168	33.6	0.631	49	0
1	99	58	10	0	25.4	0.551	21	0
10	68	106	23	49	35.5	0.285	47	0
3	123	100	35	240	57.3	0.88	22	0
8	91	82	0	0	35.6	0.587	68	0
6	195	70	0	0	30.9	0.328	31	1
9	156	86	0	0	24.8	0.23	53	1
0	93	60	0	0	35.3	0.263	25	0
3	121	52	0	0	36	0.127	25	1
2	101	58	17	265	24.2	0.614	23	0
2	56	56	28	45	24.2	0.332	22	0
0	162	76	36	0	49.6	0.364	26	1
0	95	64	39	105	44.6	0.366	22	0
4	125	80	0	0	32.3	0.536	27	1
5	136	82	0	0	0	0.64	69	0
2	129	74	26	205	33.2	0.591	25	0
3	130	64	0	0	23.1	0.314	22	0
1	107	50	19	0	28.3	0.181	29	0
1	140	74	26	180	24.1	0.828	23	0
1	144	82	46	180	46.1	0.335	46	1
8	107	80	0	0	24.6	0.856	34	0
13	158	114	0	0	42.3	0.257	44	1
2	121	70	32	95	39.1	0.886	23	0
7	129	68	49	125	38.5	0.439	43	1
2	90	60	0	0	23.5	0.191	25	0
7	142	90	24	480	30.4	0.128	43	1
3	169	74	19	125	29.9	0.268	31	1
0	99	0	0	0	25	0.253	22	0
4	127	88	11	155	34.5	0.598	28	0
4	118	70	0	0	44.5	0.904	26	0

2	122	76	27	200	35.9	0.483	26	0
6	125	78	31	0	27.6	0.565	49	1
1	168	88	29	0	35	0.905	52	1
2	129	0	0	0	38.5	0.304	41	0
4	110	76	20	100	28.4	0.118	27	0
6	80	80	36	0	39.8	0.177	28	0
10	115	0	0	0	0	0.261	30	1
2	127	46	21	335	34.4	0.176	22	0
9	164	78	0	0	32.8	0.148	45	1
2	93	64	32	160	38	0.674	23	1
3	158	64	13	387	31.2	0.295	24	0
5	126	78	27	22	29.6	0.439	40	0
10	129	62	36	0	41.2	0.441	38	1
0	134	58	20	291	26.4	0.352	21	0
3	102	74	0	0	29.5	0.121	32	0
7	187	50	33	392	33.9	0.826	34	1
3	173	78	39	185	33.8	0.97	31	1
10	94	72	18	0	23.1	0.595	56	0
1	108	60	46	178	35.5	0.415	24	0
5	97	76	27	0	35.6	0.378	52	1
4	83	86	19	0	29.3	0.317	34	0
1	114	66	36	200	38.1	0.289	21	0
1	149	68	29	127	29.3	0.349	42	1
5	117	86	30	105	39.1	0.251	42	0
1	111	94	0	0	32.8	0.265	45	0
4	112	78	40	0	39.4	0.236	38	0
1	116	78	29	180	36.1	0.496	25	0
0	141	84	26	0	32.4	0.433	22	0
2	175	88	0	0	22.9	0.326	22	0
2	92	52	0	0	30.1	0.141	22	0
3	130	78	23	79	28.4	0.323	34	1
8	120	86	0	0	28.4	0.259	22	1
2	174	88	37	120	44.5	0.646	24	1
2	106	56	27	165	29	0.426	22	0
2	105	75	0	0	23.3	0.56	53	0
4	95	60	32	0	35.4	0.284	28	0
0	126	86	27	120	27.4	0.515	21	0
8	65	72	23	0	32	0.6	42	0
2	99	60	17	160	36.6	0.453	21	0
1	102	74	0	0	39.5	0.293	42	1
11	120	80	37	150	42.3	0.785	48	1
3	102	44	20	94	30.8	0.4	26	0
1	109	58	18	116	28.5	0.219	22	0
9	140	94	0	0	32.7	0.734	45	1

13	153	88	37	140	40.6	1.174	39	0
12	100	84	33	105	30	0.488	46	0
1	147	94	41	0	49.3	0.358	27	1
1	81	74	41	57	46.3	1.096	32	0
3	187	70	22	200	36.4	0.408	36	1
6	162	62	0	0	24.3	0.178	50	1
4	136	70	0	0	31.2	1.182	22	1
1	121	78	39	74	39	0.261	28	0
3	108	62	24	0	26	0.223	25	0
0	181	88	44	510	43.3	0.222	26	1
8	154	78	32	0	32.4	0.443	45	1
1	128	88	39	110	36.5	1.057	37	1
7	137	90	41	0	32	0.391	39	0
0	123	72	0	0	36.3	0.258	52	1
1	106	76	0	0	37.5	0.197	26	0
6	190	92	0	0	35.5	0.278	66	1
2	88	58	26	16	28.4	0.766	22	0
9	170	74	31	0	44	0.403	43	1
9	89	62	0	0	22.5	0.142	33	0
10	101	76	48	180	32.9	0.171	63	0
2	122	70	27	0	36.8	0.34	27	0
5	121	72	23	112	26.2	0.245	30	0
1	126	60	0	0	30.1	0.349	47	1
1	93	70	31	0	30.4	0.315	23	0

Output:

```
pima indian diabetes dataset loaded....
total instances available: 768
total attributes present: 8
first five instances of dataset:
1 : [6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0, 1.0]
2 : [1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0, 0.0]
3 : [8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0, 1.0]
4 : [1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0, 0.0]
5 : [0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0, 1.0]

 dataset is split into training and testing set
training examples=615
 testing examples=153

 Accuracy of the naive bayssive classifier is: 79.08496732026144
```

Program Outcome:

- Understand the implementation procedures for the machine learning algorithms
- The student will be able to apply naive baysian classifier for the relevent problem and analyse the results.

Program 7

7) Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

Program Objective:

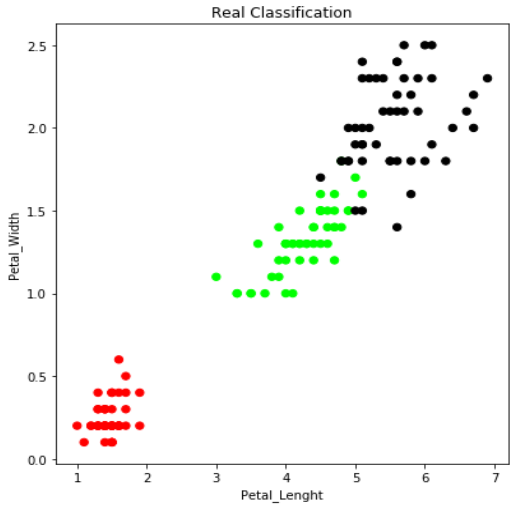
- To implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.
- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

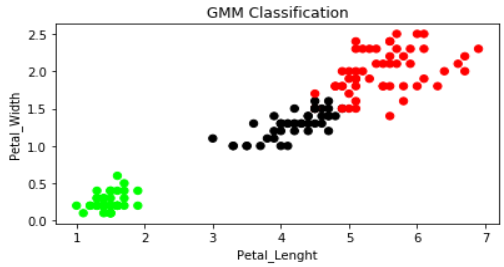
```
import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.cluster import KMeans

import sklearn.metrics as sm

import pandas as pd

import numpy as np

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)

X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)

y.columns = ['Targets']

model = KMeans(n_clusters=3)

model.fit(X)

model.labels_

plt.figure(figsize=(14,7))

colormap = np.array(['red', 'lime', 'black'])

plt.subplot(1, 2, 1)

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

plt.title('Real Classification')

plt.xlabel('Petal_Lenght')
```

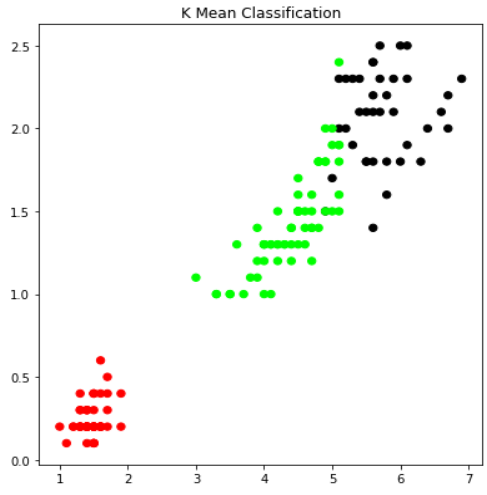
```
plt.ylabel('Petal_Width')
plt.figure(figsize=(14,7))
predY =np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length,X.Petal_Width, c=colormap[predY], s=40)
plt.title('K Mean Classification')
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)
y_cluster_gmm = gmm.predict(xs)
plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_cluster_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal_Lenght')
plt.ylabel('Petal_Width')
print('Observation : The GMM using EM algorithm based clustering matched the True label more closely than the K-MEANS')
```

Output:

```
Observation : The GMM using EM algorithm based clustering matched the True label more closely than the K-MEANS
```







Program Outcome:

- The students will be able to apply EM algorithm and k-Means algorithm for clustering and analyze the results.

Program 8

8) Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Program Objective:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python

```
from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import datasets

iris=datasets.load_iris()

print("Iris dataset loaded...")

x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,test_size=0.1)

print("Dataset is split into training and testing...")

print("Size of training data and its label",x_train.shape,y_train.shape)

print("size of testing data and its label",x_test.shape,y_test.shape)

for i in range(len(iris.target_names)):

    print("Label",i,"-",str(iris.target_names[i]))

classifier=KNeighborsClassifier(n_neighbors=1)

classifier.fit(x_train,y_train)

y_pred=classifier.predict(x_test)

print("Results of Classification using K-nn with K=1")

for r in range(0,len(x_test)):

    print("Sample:",str(x_test[r]),"Actual-label:",str(y_test[r]),"Predicted-
label:",str(y_pred[r]))

print("Classification accuracy:",classifier.score(x_test,y_test));
```

Output:

```
Iris dataset loaded...
Dataset is split into training and testing...
Size of training data and its label (135, 4) (135,)
size of testing data and its label (15, 4) (15,)
Label 0 - setosa
Label 1 - versicolor
Label 2 - virginica
Results of Classification using K-nn with K=1
Sample: [5.5 4.2 1.4 0.2] Actual-label: 0 Predicted-label: 0
Sample: [6.4 2.7 5.3 1.9] Actual-label: 2 Predicted-label: 2
Sample: [6.3 3.3 6.  2.5] Actual-label: 2 Predicted-label: 2
Sample: [6.  3.4 4.5 1.6] Actual-label: 1 Predicted-label: 1
Sample: [5.  3.4 1.6 0.4] Actual-label: 0 Predicted-label: 0
Sample: [7.7 3.8 6.7 2.2] Actual-label: 2 Predicted-label: 2
Sample: [6.7 3.1 5.6 2.4] Actual-label: 2 Predicted-label: 2
Sample: [5.  3.4 1.5 0.2] Actual-label: 0 Predicted-label: 0
Sample: [5.2 3.5 1.5 0.2] Actual-label: 0 Predicted-label: 0
Sample: [6.2 2.2 4.5 1.5] Actual-label: 1 Predicted-label: 1
Sample: [5.9 3.  4.2 1.5] Actual-label: 1 Predicted-label: 1
Sample: [7.4 2.8 6.1 1.9] Actual-label: 2 Predicted-label: 2
Sample: [4.4 2.9 1.4 0.2] Actual-label: 0 Predicted-label: 0
Sample: [6.6 3.  4.4 1.4] Actual-label: 1 Predicted-label: 1
Sample: [6.9 3.1 4.9 1.5] Actual-label: 1 Predicted-label: 1
Classification accuracy: 1.0
```

Program Outcome:

- The student will be able to implement k-Nearest Neighbour algorithm to classify the iris data set and Print both correct and wrong predictions.

Program 9

9) Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Program Objective:

- Make use of Data sets in implementing the machine learning algorithms.
- Implement ML concepts and algorithms in Python
- Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points.

```
import numpy as np

import pandas as pd

from sklearn.datasets import load_boston

import matplotlib.pyplot as plt

%matplotlib inline

import math


boston = load_boston()

features = pd.DataFrame(boston.data, columns=boston.feature_names)

target = pd.DataFrame(boston.target,columns=['target'])

data = pd.concat([features,target],axis=1)

x = data['RM']

X1 = sorted(np.array(x/x.mean()))


X=X1+[i+1 for i in X1]


Y=np.sin(X)

plt.plot(X,Y)

n = int(0.8 * len(X))


x_train = X[:n]
```

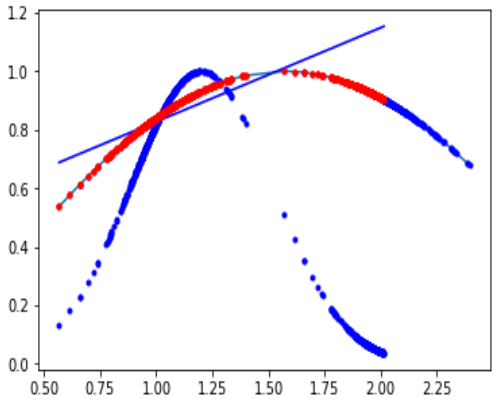
```
y_train = Y[:n]
x_test = X[n:]
y_test = Y[n:]

w=np.exp([-(1.2-i)**2/(2*0.1) for i in x_train])
plt.plot(x_train, y_train,'r.')
plt.plot(x_train,w,'b.')

def h(x,a,b):
    return a*x + b

def error(a,x,b,y,w):
    e = 0
    m = len(x)

    for i in range(m):
        e += np.power(h(x[i],a,b)-y[i],2)*w[i]
    return (1/(2*m)) * e

def step_gradient(a,x,b,y,learning_rate,w):
    grad_a = 0
    grad_b = 0
    m = len(x)
    for i in range(m):
        grad_a += (2/m)*((h(x[i],a,b)-y[i])*x[i])*w[i]
        grad_b += (2/m)*(h(x[i],a,b)-y[i])*w[i]
    a = a - (grad_a * learning_rate)
    b = b - (grad_b * learning_rate)
    return a,b
```

```
def descend(initial_a, initial_b, x, y, learning_rate, iterations,w):
    a = initial_a
    b = initial_b
    for i in range(iterations):
        e = error(a,x,b,y,w)
        if i%1000 == 0:
            print(f"Error: {e}-- a:{a}, b:{b}")
        a, b = step_gradient(a,x,b,y, learning_rate,w)
    return a,b

a = 1.8600662368042573
b = -0.7962243178421666
learning_rate = 0.01
iterations = 10000

final_a, final_b = descend(a,b,x_train,y_train, learning_rate, iterations,w)

H=[i*final_a+final_b for i in x_train]

plt.plot(x_train, y_train,'r.',x_train, H,'b')

print(error(a,x_test,b,y_test,w))
print(error(final_a,x_test, final_b,y_test,w))
plt.plot(x_test,y_test,'b.',x_train,y_train,'r.')
```

Output:

```
Error: 0.06614137226206705-- a:1.8600662368042573, b:-0.7962243178421666
Error: 0.01831248988715221-- a:1.3533605603913972, b:-0.6206735673234249
Error: 0.011422762970211432-- a:1.1032234861838637, b:-0.347590814908577
Error: 0.007176247674245229-- a:0.9068452261129998, b:-0.13319830250762849
Error: 0.0045588881799908-- a:0.7526720746347257, b:0.0351175247039557
Error: 0.0029456664570710403-- a:0.6316334187867452, b:0.16725934893398114
Error: 0.0019513497294632626-- a:0.536608078323685, b:0.2710015934995427
Error: 0.0013384979802249 41-- a:0.46200533867114346, b:0.3524478227325071
Error: 0.0009607639482851428-- a:0.4034360271954487, b:0.41638983867834906
Error: 0.0007279458172072266-- a:0.35745428091221954, b:0.4665896016596849
1.6930984012182055
0.037219754002487955
```

Out[1]: [<matplotlib.lines.Line2D at 0x7f19edb87908>,
<matplotlib.lines.Line2D at 0x7f19edb95400>]



Program Outcome:

- To understand and implement linear regression and analyses the results with change in the parameters.

VIVA QUESTIONS

1. What is the difference between supervised and unsupervised machine learning?

A Supervised learning is a process where it requires training labeled data. When it comes to Unsupervised learning it doesn't require data labeling.

2. How is KNN different from K-means clustering?

KNN stands for K- Nearest Neighbours, it is classified as a supervised algorithm. K-means is an unsupervised cluster algorithm.

4. How to handle or missing data in a dataset?

An individual can easily find missing or corrupted data in a data set either by dropping the rows or columns. On contrary, they can decide to replace the data with another value. In Pandas they are two ways to identify the missing data, these two methods are very useful. isnull() and dropna().

5. What is the difference between an array and Linked list?

Deep An array is an ordered fashion of collection of objects. A linked list is a series of objects that are processed in a sequential order.

6. Explain why Navie Bayes is so Naive?

It is based on an assumption that all of the features in the data set are important, equal and independent.

7. Please state few popular Machine Learning algorithms?

Nearest Neighbour
Neural Networks
Decision Trees etc
Support vector machines

8. What are the different types of algorithm techniques available in machine learning?

Some of them are :
Supervised learning
Unsupervised learning
Semi-supervised learning
Transduction
Learning to learn

9. What are the three stages to build the model in machine learning?

1. Model building
2. Model testing
3. Applying the model

10. Name a few libraries in Python used for Data Analysis and Scientific computations

NumPy, SciPy, Pandas, SciKit, Matplotlib, Seaborn

11. Which is the standard data missing marker used in Pandas?

NaN

12. Write the code to sort an array in NumPy by the nth column?

Using argsort () function this can be achieved. If there is an array X and you would like to sort the nth column then code for this will be x[x [: n-1].argsort ()]

13.What is pylab?

A package that combines NumPy, SciPy and Matplotlib into a single namespace.

14. Is all the memory freed when Python exits?

No it is not, because the objects that are referenced from global namespaces of Python modules are not always de-allocated when Python exits.

15. How can you randomize the items of a list in place in Python?

Shuffle (lst) can be used for randomizing the items of a list in Python

16. Which tool in Python will be used to find bugs if any?

Pylint and Pychecker. Pylint verifies that a module satisfies all the coding standards or not. Pychecker is a static analysis tool that helps find out bugs in the course code.

17. What are the supported data types in Python?

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

19. What are the supported sequence types in Python?

Python supports 7 sequence types. They are str, list, tuple, unicode, byte array, xrange, and buffer. where xrange is deprecated in python 3.5.X.

20. What is pylab?

A package that combines NumPy, SciPy and Matplotlib into a single namespace.

21. What are the supported data types in Python?

SciKit-Learn

23. Is Python a case-sensitive programming language?

Yes, it is a case-sensitive language.

24. What is the difference between a tuple and a list?

The basic difference between a tuple and a list is that the former is immutable and the latter is mutable.

25. What is the difference between Xrange() and range()?

Range() returns a list and xrange() returns an xrange object, which is kind of like an iterator and generates the numbers on demand.

26. Optimize the below python code-

```
word = 'word'
print word.__len__ ()
```

print 'word'._len_ ()

27. What is PEP 8?

PEP 8 is a coding convention that lets us write more readable code. In other words, it is a set of recommendations.

28. What are Python decorators?

A Python decorator is a specific change that we make in Python syntax to alter functions easily.

29. What is Dict and List comprehensions are?

They are syntax constructions to ease the creation of a Dictionary or List based on existing iterable.

30. What is lambda in Python?

It is a single expression anonymous function often used as inline function.

31. What is pass in Python?

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

32. In Python what are iterators?

In Python, iterators are used to iterate a group of elements, containers like list.

33. In Python what is slicing?

A mechanism to select a range of items from sequence types like list, tuple, strings etc. is known as slicing.

34. What is docstring in Python?

A Python documentation string is known as docstring, it is a way of documenting Python functions, modules and classes.

35. How can you copy an object in Python?

To copy an object in Python, you can try copy.copy () or copy.deepcopy() for the general case. You cannot copy all objects but most of them.

36. Explain how to delete a file in Python?

By using a command os.remove (filename) or os.unlink(filename)

37. Is It Mandatory For A Python Function To Return A Value?

It is not at all necessary for a function to return any value. However, if needed, we can use None as a return value.

38. What Is Whitespace In Python?

Whitespace represents the characters that we use for spacing and separation. They possess an "empty" representation. In Python, it could be a tab or space.

39. What Is Isalpha() In Python?

Python provides this built-in isalpha() function for the string handling purpose. It returns True if all characters in the string are of alphabet type, else it returns False.

40. What Does The Join Method Do In Python?

Python provides the join() method which works on strings, lists, and tuples. It combines them and returns a united value.

41. What Makes The CPython Different From Python?

CPython has its core developed in C. The prefix 'C' represents this fact. It runs an interpreter loop used for translating the Python-ish code to C language.

42. What Is A Tuple In Python?

A tuple is a collection type data structure in Python which is immutable. They are similar to sequences, just like the lists. However, There are some differences between a tuple and list; the former doesn't allow modifications whereas the list does.

43. What is Artificial Intelligence?

Artificial Intelligence is an area of computer science that emphasizes the creation of intelligent machine that work and reacts like humans.

44. What are the various areas where AI (Artificial Intelligence) can be used?

Artificial Intelligence can be used in many areas like Computing, Speech recognition, Bio-informatics, Humanoid robot, Computer software, Space and Aeronautics's etc.