

*B.N.M. Institute of Technology*

12th Main, 27th Cross, Banashankari II Stage, Bengaluru 560 070.

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



## **LABORATORY MANUAL**

**2018 - 2019**

**VII SEMESTER B.E**

**Course: Machine Learning Laboratory**  
**Course Code: 15CSL76**

**Lab Incharge**

**Jebah Jaykumar**  
**Priyanka S**  
**Usha C R**

## CONTENTS

Sl. No	Topics	Page No.
1	Program Outcomes	3
2	Course Outcomes	4
3	Scheme of Evaluation & Rubrics for Laboratory	5
4	Introduction to Machine Learning	9
5	Sample Programs	13
6	1. Implement and demonstrate the <b>FIND-S algorithm</b> for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.	31
7	2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the <b>Candidate-Elimination algorithm</b> to output a description of the set of all hypotheses consistent with the training examples.	36
8	3. Write a program to demonstrate the working of the decision tree based <b>ID3 algorithm</b> . Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.	42
9	4. Build an Artificial Neural Network by implementing the <b>Backpropagation algorithm</b> and test the same using appropriate data sets.	48
10	5. Write a program to implement the <b>naïve Bayesian classifier</b> for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.	51
11	6. Assuming a set of documents that need to be classified, use the <b>naïve Bayesian Classifier</b> model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.	56
12	7. Write a program to construct a <b>Bayesian network</b> considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.	60
13	8. Apply <b>EM algorithm</b> to cluster a set of data stored in a .CSV file. Use the same data set for clustering using <b>k-Means algorithm</b> . 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.	67
14	9. Write a program to implement <b>k-Nearest Neighbour algorithm</b> to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.	76
15	10. Implement the non-parametric <b>Locally Weighted Regression algorithm</b> in order to fit data points. Select appropriate data set for your experiment and draw graphs.	81
16	VIVA Q& A	84



## I. PROGRAM OUTCOMES

### Engineering Graduates will be able to:

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

**II. COURSE OUTCOMES:**

Upon successful completion of this Lab the student will be capable of:

1. Understand the implementation procedures for the machine learning algorithms.
2. Design Java/Python programs for various Learning algorithms.
3. Apply appropriate data sets to the Machine Learning algorithms.
4. Identify and apply Machine Learning algorithms to solve real world problems.

**CO mapping to PO/PSOs**

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3											
CO2		3	3	3	3							
CO3		3		3	2							
CO4		3	3	3	2	2						2

	CO –1	CO – 2	CO – 3	CO –4
Program 1	✓	✓	✓	✓
Program 2	✓	✓	✓	✓
Program 3	✓	✓	✓	✓
Program 4	✓	✓	✓	✓
Program 5	✓	✓	✓	✓
Program 6	✓	✓	✓	✓
Program 7	✓	✓	✓	✓
Program 8	✓	✓	✓	✓
Program 9	✓	✓	✓	✓
Program 10	✓	✓	✓	✓

**III. SCHEME OF EVALUATION & RUBRICS FOR LABORATORY****Internal Assessment split up (20)**

<b>a. Final Observation marks</b>	<b>6</b>	<b>Average of all observation marks</b>
<b>b. Lab Record</b>	<b>4</b>	<b>Average of all programs</b>
<b>c. Test</b>	<b>10</b>	<b>Average mark considered from 2 Tests and reduced to 10 Marks for final marks calculation.</b>
<b>Total</b>	<b>20 marks</b>	

**a) Observation**

Every program is evaluated for 6 Marks with the following scheme.

1. Write-up	02 M
2. Execution	02 M
3. Viva-Voce	02 M
<hr/>	
<b>Total: 6 M</b>	

**b) Lab Record**

Every program is evaluated for 4 marks.

**c) Test**

Two internal tests are conducted for 50 marks with the following scheme.

1. Write-up	08 M
2. Execution	35 M
3. Viva-Voce	07 M
<hr/>	
<b>Total: 50 M</b>	

**Rubrics for Evaluation of Observation**

Attribute	Excellent (2)	Good (2)	Satisfactory (1)	Poor (0)
<b>1. Write up (02)</b>	<ul style="list-style-type: none"> <li>Source code without errors.</li> <li>Code with comments.</li> <li>Input and expected output for all test cases.</li> <li>Indentation.</li> </ul>	<ul style="list-style-type: none"> <li>Complete source code with 1 or 2 errors.</li> <li>Input and expected output for all test cases.</li> <li>Indentation</li> </ul>	<ul style="list-style-type: none"> <li>Source code with syntax &amp; logical errors.</li> <li>Incomplete output</li> </ul>	<ul style="list-style-type: none"> <li>Incomplete code.</li> <li>Late submission</li> </ul>

Attribute	Good (2)	Satisfactory (1)	Poor (0)
<b>2. Execution (02)</b>	<ul style="list-style-type: none"> <li>Debugs the program independently.</li> <li>Executes the program for all possible inputs.</li> <li>Able to modify existing program</li> <li>Uses appropriate data set</li> </ul>	<ul style="list-style-type: none"> <li>Debugs the program independently.</li> <li>Executes the program for all possible inputs.</li> <li>Able to modify the programs with hints from faculty</li> <li>Completes execution with help from faculty.</li> <li>Applies the appropriate data set with help from faculty</li> </ul>	<ul style="list-style-type: none"> <li>Not able to complete the execution of program within the lab session.</li> <li>Not able to do any modifications.</li> <li>Does not know to use appropriate data set.</li> </ul>

Attribute	Good (2)	Satisfactory (1)	Poor (0)
<b>3. Viva – voice(02)</b>	Answering 4-6 out of 6 questions.	Answering 1-3 out of 6 questions.	Not Answering any questions.

**Rubrics for Evaluation of Record Book**

Attribute	Excellent (4)	Good (3)	Satisfactory (2)	Poor (1)
<b>Record (04)</b>	<ul style="list-style-type: none"> <li>•Writes the record neatly.</li> <li>•With data set description</li> <li>•Program with comments.</li> <li>•All possible cases of output</li> <li>•Index entry and submits on time.</li> <li>•Indentation.</li> </ul>	<ul style="list-style-type: none"> <li>•Writes the record neatly.</li> <li>•With data set description</li> <li>•Program with comments.</li> <li>•All possible cases of output</li> <li>•Index entry.</li> </ul>	<ul style="list-style-type: none"> <li>• Not written clearly</li> <li>• With data set description</li> <li>• All possible cases of output.</li> <li>• Incomplete index entry</li> </ul>	Late submission.

**Rubrics for Evaluation of Test**

Attribute	Excellent (8)	Good (6)	Satisfactory (4)	Poor (0)
<b>1. Write-up (08)</b>	<ul style="list-style-type: none"> <li>• Algorithm &amp; data set description.</li> <li>• Complete source code.</li> <li>• Free of syntax and logical errors.</li> <li>• All possible cases of input with expected output.</li> <li>• Comments included</li> </ul>	<ul style="list-style-type: none"> <li>• Algorithm &amp; data set description.</li> <li>• Complete source code.</li> <li>• Free of syntax and logical</li> <li>• Minor mistakes</li> </ul>	<ul style="list-style-type: none"> <li>• Algorithm &amp; data set description.</li> <li>• Complete source code</li> <li>• Few syntactical and logical errors</li> </ul>	<ul style="list-style-type: none"> <li>• Incomplete source code.</li> </ul>



Attribute	Excellent (30-35)	Good (20-29)	Satisfactory (11-19)	Poor (0-10)
<b>2. Execution (35)</b>	<ul style="list-style-type: none"> <li>• Execution of the program for all possible inputs</li> <li>• Able to modify the programs as per the examiners direction</li> </ul>	<ul style="list-style-type: none"> <li>• Execution of the program for all possible inputs</li> </ul>	<ul style="list-style-type: none"> <li>• Debugs the program.</li> <li>• Not able to execute program for all possible inputs.</li> </ul>	<ul style="list-style-type: none"> <li>• Not able to execute the program within the lab session.</li> </ul>

Attribute	Excellent (6-7)	Good (4-5)	Satisfactory (2-3)	Poor (0-1)
<b>3.Viva – voce(07)</b>	Answering 4-5 out of 5 questions.	Answering 2-3 out of 5 questions.	Answering 1 out of 5 questions.	Does not answer any question.

**\*Two internal tests are conducted with above scheme for 50 marks and an average mark is considered for scaling down to 10 marks for final IA calculation.**

## Introduction to Machine learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.

### Machine learning tasks

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:

**Supervised learning:** The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback:

**Semi-supervised learning:** the computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

**Active learning:** the computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.

**Reinforcement learning:** training data (in form of rewards and punishments) is given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

**Unsupervised learning:** No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Supervised learning	Un Supervised learning	Instance based learning
Find-s algorithm	EM algorithm	Locally weighted Regression algorithm
Candidate elimination algorithm	K means algorithm	
Decision tree algorithm		
Back propagation Algorithm		
Naïve Bayes Algorithm		
K nearest neighbour algorithm(lazy learning algorithm)		

## Machine learning applications

In **classification**, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised manner. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

In **regression**, also a supervised problem, the outputs are continuous rather than discrete.

In **clustering**, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

**Density estimation** finds the distribution of inputs in some space.

**Dimensionality reduction** simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked with finding out which documents cover similar topics.

## Machine learning Approaches

### Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

### Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

### Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

### Deep learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing.

Some successful applications of deep learning are computer vision and speech recognition.

### **Inductive logic programming**

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

### **Support vector machines**

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

### **Clustering**

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some pre-designated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

### **Bayesian networks**

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

### **Reinforcement learning**

Reinforcement learning is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. Reinforcement learning differs from

supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

### **Similarity and metric learning**

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

### **Genetic algorithms**

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

### **Rule-based machine learning**

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply, knowledge. The defining characteristic of a rule-based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learners that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

### **Feature selection approach**

Feature selection is the process of selecting an optimal subset of relevant features for use in model construction. It is assumed the data contains some features that are either redundant or irrelevant, and can thus be removed to reduce calculation cost without incurring much loss of information. Common optimality criteria include accuracy, similarity and information measures.

---

**SAMPLE PROGRAMS****1. Write a Python Program to Add Two Numbers.**

```
# This program adds two numbers
num1 = 1.5
num2 = 6.3
# Add two numbers
sum = float(num1) + float(num2)
# Display the sum
print("The sum of {0} and {1} is {2}".format(num1, num2, sum))
```

**Output**

```
The sum of 1.5 and 6.3 is 7.8
```

**2. Write a Python Program to Add Two Numbers Provided by The User.**

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')
# Add two numbers
sum = float(num1) + float(num2)
# Display the sum
print("The sum of {0} and {1} is {2}".format(num1, num2, sum))
```

**Output**

```
Enter first number: 1.5
Enter second number: 6.3
The sum of 1.5 and 6.3 is 7.8
```

**3. Write a Python Program to Find the Largest Among Three Number.**

```
# Python program to find the largest number among the three input numbers
# change the values of num1, num2 and num3
# for a different result
```

```
num1 = 10
num2 = 14
num3 = 12

# uncomment following lines to take three numbers from user
#num1 = float(input("Enter first number: "))
#num2 = float(input("Enter second number: "))
#num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3

print("The largest number between",num1,"",num2,"and",num3,"is",largest)
```

### Output

```
The largest number between 10, 14 and 12 is 14.0
```

#### 4. Write a Python Program to find LCM without using GCD function.

```
# Python Program to find the L.C.M. of two input number
# define a function
def lcm(x, y):
    """This function takes two
    integers and returns the L.C.M."""
    # choose the greater number
    if x > y:
        greater = x
    else:
        greater = y
    while(True):
        if((greater % x == 0) and (greater % y == 0)):
            lcm = greater
            break
```

```
        greater += 1
    return lcm

# change the values of num1 and num2 for a different result
num1 = 54
num2 = 24

# uncomment the following lines to take input from the user
#num1 = int(input("Enter first number: "))
#num2 = int(input("Enter second number: "))
print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
```

### Output

```
The L.C.M. of 54 and 24 is 216
```

### 5. Write a Python Program to find LCM using GCD function.

```
# define gcd function
def gcd(x, y):
    """This function implements the Euclidian algorithm
    to find G.C.D. of two numbers"""
    while(y):
        x, y = y, x % y
    return x

# define lcm function
def lcm(x, y):
    """This function takes two
    integers and returns the L.C.M."""
    lcm = (x*y)//gcd(x,y)
    return lcm

# change the values of num1 and num2 for a different result
num1 = 54
num2 = 24

# uncomment the following lines to take input from the user
#num1 = int(input("Enter first number: "))
```



```
#num2 = int(input("Enter second number: "))  
print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
```

### Output

```
The L.C.M. of 54 and 24 is 216
```

### 6. Write a Python Program to simulate a Simple Calculator by making Functions

''' Program make a simple calculator that can add, subtract, multiply and divide using functions '''

```
# This function adds two numbers  
def add(x, y):  
    return x + y  
  
# This function subtracts two numbers  
def subtract(x, y):  
    return x - y  
  
# This function multiplies two numbers  
def multiply(x, y):  
    return x * y  
  
# This function divides two numbers  
def divide(x, y):  
    return x / y  
  
print("Select operation.")  
print("1.Add")  
print("2.Subtract")  
print("3.Multiply")  
print("4.Divide")  
  
# Take input from the user  
choice = input("Enter choice(1/2/3/4):")  
  
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))  
  
if choice == '1':  
    print(num1,"+",num2,"=", add(num1,num2))  
  
elif choice == '2':  
    print(num1,"-",num2,"=", subtract(num1,num2))  
  
elif choice == '3':  
    print(num1,"*",num2,"=", multiply(num1,num2))
```

```
elif choice == '4':
    print(num1,"/",num2,"=", divide(num1,num2))
else:
    print("Invalid input")
```

### Output

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 3
Enter first number: 15
Enter second number: 14
15 * 14 = 210
```

### 7. Write a Python Program to Print all Prime Numbers in an Interval.

# Python program to display all the prime numbers within an interval

# change the values of lower and upper for a different result

```
lower = 900
```

```
upper = 1000
```

# uncomment the following lines to take input from the user

```
#lower = int(input("Enter lower range: "))
```

```
#upper = int(input("Enter upper range: "))
```

```
print("Prime numbers between",lower,"and",upper,"are:")
```

```
for num in range(lower,upper + 1):
```

```
    # prime numbers are greater than 1
```

```
    if num > 1:
```

```
        for i in range(2,num):
```

```
            if (num % i) == 0:
```

```
                break
```

```
        else:
```

```
            print(num)
```

**Output**

```
Prime numbers between 900 and 1000 are:
907
911
919
929
937
941
947
953
967
971
977
983
991
997
```

**8. Write a Python Program to Find the Sum of Natural Numbers.**

# Python program to find the sum of natural numbers up to n where n is provided by user

# change this value for a different result

num = 16

# uncomment to take input from the user

#num = int(input("Enter a number: "))

if num < 0:

print("Enter a positive number")

else:

sum = 0

# use while loop to iterate until zero

while(num > 0):

sum += num

num -= 1

print("The sum is",sum)

**Output**

```
The sum is 136
```

**9. Python Program to Find Sum of Natural Numbers Using Recursion.**

# Python program to find the sum of natural numbers up to n using recursive function

```
def recur_sum(n):  
    """Function to return the sum  
    of natural numbers using recursion"""  
    if n <= 1:  
        return n  
    else:  
        return n + recur_sum(n-1)  
# change this value for a different result  
num = 16  
# uncomment to take input from the user  
#num = int(input("Enter a number: "))  
if num < 0:  
    print("Enter a positive number")  
else:  
    print("The sum is",recur_sum(num))
```

**Output**

```
The sum is 136
```

**10. Write a Python Program to Print the Fibonacci sequence.**

```
# Program to display the Fibonacci sequence up to n-th term where n is provided by the user  
# change this value for a different result  
nterms = 10  
# uncomment to take input from the user  
#nterms = int(input("How many terms? "))  
  
# first two terms  
n1 = 0  
n2 = 1
```

```
count = 0
# check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print("Fibonacci sequence upto",nterms,":")
    print(n1)
else:
    print("Fibonacci sequence upto",nterms,":")
    while count < nterms:
        print(n1,end=' ', )
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```

### Output

```
Fibonacci sequence upto 10 :
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

### 11. Write a Python Program to create a string in Python.

```
# all of the following are equivalent
my_string = 'Hello'
print(my_string)
my_string = "Hello"
print(my_string)

my_string = """Hello"""
print(my_string)
# triple quotes string can extend multiple lines
```

```
my_string = """Hello, welcome to the world of Python"""  
print(my_string)
```

**Output:**

```
Hello  
Hello  
Hello  
Hello, welcome to  
the world of Python
```

**12. Write a Python program to access characters in a String.**

```
str = 'programiz'  
print('str = ', str)  
#first character  
print('str[0] = ', str[0])  
#last character  
print('str[-1] = ', str[-1])  
#slicing 2nd to 5th character  
print('str[1:5] = ', str[1:5])  
#slicing 6th to 2nd last character  
print('str[5:-2] = ', str[5:-2])
```

**Output:**

```
>>> # two string literals together  
>>> 'Hello ' 'World!'  
'Hello World!'  
  
>>> # using parentheses  
>>> s = ('Hello '  
...     'World')  
>>> s  
'Hello World'
```

**13. Write a Python program to count the number of characters (character frequency) in a string.**

```
def char_frequency(str1):  
    dict = { }  
    for n in str1:  
        keys = dict.keys()  
        if n in keys:  
            dict[n] += 1  
        else:  
            dict[n] = 1  
    return dict  
print(char_frequency('google.com'))
```

**Output:**

{‘o’:3, ‘.’:1, ‘g’:2, ‘l’:1, ‘e’:1, ‘c’:1, ‘m’:1}

**14. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.**

```
def string_both_ends(str):  
    if len(str) < 2:  
        return ""  
  
    return str[0:2] + str[-2:]  
  
print(string_both_ends('pythonresource'))  
print(string_both_ends('py'))  
print(string_both_ends('p'))
```

**Output:**

pyce  
pypy

**15. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '\$', except the first char itself.**

```
def change_char(str1):  
    char = str1[0]  
    str1 = str1.replace(char, '$')  
    str1 = char + str1[1:]
```

```
return str1
```

```
print(change_char('restart'))
```

**Output:**

```
resta$t
```

**16. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.**

```
def chars_mix_up(a, b):  
    new_a = b[:2] + a[2:]  
    new_b = a[:2] + b[2:]  
  
    return new_a + ' ' + new_b  
print(chars_mix_up('abc', 'xyz'))
```

**Output:**

```
xyz abz
```

**17. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged**

```
def add_string(str1):  
    length = len(str1)  
  
    if length > 2:  
        if str1[-3:] == 'ing':  
            str1 += 'ly'  
        else:  
            str1 += 'ing'  
  
    return str1  
print(add_string('ab'))  
print(add_string('abc'))  
print(add_string('string'))
```

**Output:**

```
ab  
abcing
```



stringly

**18. Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'bad' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'.**

```
def not_poor(str1):
    snot = str1.find('not')
    sbad = str1.find('poor')

    if sbad > snot:
        str1 = str1.replace(str1[snot:(sbad+4)], 'good')

    return str1
print(not_poor("The lyrics is not that poor!"))
```

**Output:**

The lyrics is good

**19. Write a Python program to sum all the items in a list.**

```
def sum_list(items):
    sum_numbers = 0
    for x in items:
        sum_numbers += x
    return sum_numbers
print(sum_list([1,2,-8]))
```

**Output:**

-5

**20. Write a Python program to get the largest number from a list.**

```
def max_num_in_list( list ):
    max = list[ 0 ]
    for a in list:
        if a > max:
            max = a
    return max
print(max_num_in_list([1, 2, -8, 0]))
```

**Output:**

2

**21. Write a Python program to count the number of strings where the string length is 2 or more**

**and the first and last character are same from a given list of strings.**

```
def match_words(words):  
    ctr = 0  
  
    for word in words:  
        if len(word) > 1 and word[0] == word[-1]:  
            ctr += 1  
    return ctr  
  
print(match_words(['abc', 'xyz', 'aba', '1221']))
```

**Output:**

2

**22. Write a Python function that takes two lists and returns True if they have at least one common member.**

```
def common_data(list1, list2):  
    result = False  
    for x in list1:  
        for y in list2:  
            if x == y:  
                result = True  
    return result  
  
print(common_data([1,2,3,4,5], [5,6,7,8,9]))  
print(common_data([1,2,3,4,5], [6,7,8,9]))
```

**Output:**

True

None

**23. Write a Python program to replace the last element in a list with another list**

```
num1 = [1, 3, 5, 7, 9, 10]  
num2 = [2, 4, 6, 8]  
num1[-1:] = num2  
print(num1)
```

**Output:**

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

**24. Write a Python program to add a key to a dictionary.**

```
d = {0:10, 1:20}
```

```
print(d)
d.update({2:30})
print(d)
```

**Output:**

```
{0: 10, 1: 20}
{0: 10, 1: 20, 2: 30}
```

**25. Write a Python program to concatenate following dictionaries to create a new one.**

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

**Output:**

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

**26. Write a Python program to check if a given key already exists in a dictionary.**

```
d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
def is_key_present(x):
    if x in d:
        print('Key is present in the dictionary')
    else:
        print('Key is not present in the dictionary')
is_key_present(5)
is_key_present(9)
```

**Output:**

```
Key is present in the dictionary
Key is not present in the dictionary
```

**27. Write a Python program to iterate over dictionaries using for loops.**

```
d = {'x': 10, 'y': 20, 'z': 30}
for dict_key, dict_value in d.items():
    print(dict_key,'->',dict_value)
```

**Output:**

```
y -> 20
z -> 30
x -> 10
```

**28. Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.**

```
d=dict()
for x in range(1,16):
    d[x]=x**2
print(d)
```

**Output:**

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}
```

**29. Write a Python program to create a tuple.**

```
#Create an empty tuple

x = ()
print(x)

#Create an empty tuple with tuple() function built-in Python
tuplex = tuple()
print(tuplex)
```

**Output:**

```
()
()
```

**30. Write a Python program to create a tuple with different data types.**

```
#Create a tuple with different data types
tuplex = ("tuple", False, 3.2, 1)
print(tuplex)
```

**Output:**

```
('tuple', False, 3.2, 1)
```

**31. Write a Python program to create a tuple with numbers and print one item.**

```
#Create a tuple with numbers
tuplex = 5, 10, 15, 20, 25
print(tuplex)
#Create a tuple of one item
tuplex = 5,
print(tuplex)
```

**Output:**

```
(5,10,15,20,25)
(5)
```

**32. Write a Python program to unpack a tuple in several variables.**

```
#create a tuple
tuplex = 4, 8, 3
print(tuplex)
n1, n2, n3 = tuplex
#unpack a tuple in variables
print(n1 + n2 + n3)
#the number of variables must be equal to the number of items of the tuple
n1, n2, n3, n4= tuplex
```

**Output:**

```
(4, 8, 3)
15
Traceback (most recent call last):
  File "32fd05c0-3096-11e7-a6a0-0b37d4d0b2c6.py", line 8, in <module>
    n1, n2, n3, n4 = tuplex
ValueError: not enough values to unpack (expected 4, got 3)
```

**33. Write a Python program to add an item in a tuple.**

```
#create a tuple
tuplex = (4, 6, 2, 8, 3, 1)
print(tuplex)
#tuples are immutable, so you cannot add new elements
#using merge of tuples with the + operator you can add an element and it will create a new tuple
```

```
tuplex = tuplex + (9,)
print(tuplex)
#adding items in a specific index
tuplex = tuplex[:5] + (15, 20, 25) + tuplex[:5]
print(tuplex)
#converting the tuple to list
listx = list(tuplex)
#use different ways to add items in list
listx.append(30)
tuplex = tuple(listx)
print(tuplex)
```

**Output:**

```
(4, 6, 2, 8, 3, 1)
(4, 6, 2, 8, 3, 1, 9)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3, 30)
```

**34. Write a Python program to remove an item from a tuple.**

```
#create a tuple
tuplex = "p", "y", "t", "h", "o", "n", "s", "o", "u", "r", "c", "e"
print(tuplex)
#tuples are immutable, so you cannot remove elements
#using merge of tuples with the + operator you can remove an item and it will create a new tuple
tuplex = tuplex[:2] + tuplex[3:]
print(tuplex)
#converting the tuple to list
listx = list(tuplex)
#use different ways to remove an item of the list
listx.remove("c")
#converting the tuple to list
tuplex = tuple(listx)
print(tuplex)
```

**Output:**

```
('p', 'y', 't', 'h', 'o', 'n', 's', 'o', 'u', 'r', 'c', 'e')
('p', 'y', 'h', 'o', 'n', 's', 'o', 'u', 'r', 'c', 'e')
('p', 'y', 'h', 'o', 'n', 's', 'o', 'u', 'r', 'e')
```

## LABORATORY EXPERIMENTS

**Lab Program 1:** Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

**Concept learning** - A learning task in which a human or machine learner is trained to classify objects by being shown a set of example objects along with their class labels. The learner simplifies what has been observed by condensing it in the form of an example.

### **FIND-S Algorithm:**

Find-S algorithm is a searching algorithm.

**Purpose:** To find the maximally specific hypothesis from the set of hypothesis space.

**Method:** Begin with most specific possible hypothesis in  $H$ , then generalize this hypothesis each time it fails to cover an observed positive training example.

#### **Notations:**

$D$  - Training data set

$X$  - Set of instances with in training data set

$x$  - particular instance in training example

$H$  - Set of possible hypothesis

$h$  - particular hypothesis described by conjunction of constraints on the attributes

$a_i$  - constraint attribute of hypothesis,  $a_i$  can have a value of  $\theta$  (no value), or any value( $a_i = \text{sunny}$ ), or ?(any value)

$c$  - target concept

1. Initialize  $h$  to the most specific hypothesis in  $H$
2. For each positive training instance  $x$ 
  - For each attribute constraint  $a_i$  in  $h$  :
    - If the constraint  $a_i$  in  $h$  is satisfied by  $x$  Then do nothing
    - Else replace  $a_i$  in  $h$  by the next more general constraint that is satisfied by  $x$
3. Output hypothesis  $h$

**Example:**

Positive and negative training data set for the target concept “Enjoy sport”

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	Enjoy sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	War3m	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

**Step 1:** Initialize  $h$  to most specific hypothesis

$$h = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$$

**Step 2:** 1<sup>st</sup> positive training example

$$h = (\text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same})$$

**Step 3:** 2<sup>nd</sup> Positive training example

$$h = (\text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same})$$

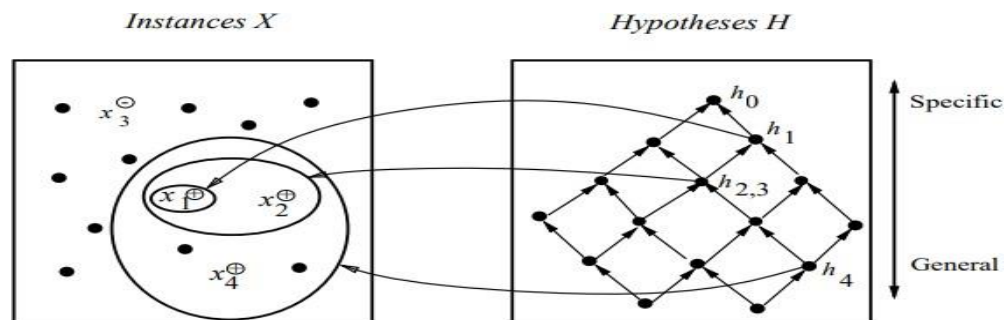
**Step 4:** 3<sup>rd</sup> negative training example

$$h = (\text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same})$$

**Step 5:** 4<sup>th</sup> positive training example

$$h = (\text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ?)$$

Hypothesis Space Search by Find-S



$$\begin{aligned}
 x_1 &= \langle \text{Sunny Warm Normal Strong Warm Same} \rangle, + & h_0 &= \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \\
 x_2 &= \langle \text{Sunny Warm High Strong Warm Same} \rangle, + & h_1 &= \langle \text{Sunny Warm Normal Strong Warm Same} \rangle \\
 x_3 &= \langle \text{Rainy Cold High Strong Warm Change} \rangle, - & h_2 &= \langle \text{Sunny Warm ? Strong Warm Same} \rangle \\
 x_4 &= \langle \text{Sunny Warm High Strong Cool Change} \rangle, + & h_3 &= \langle \text{Sunny Warm ? Strong Warm Same} \rangle \\
 & & h_4 &= \langle \text{Sunny Warm ? Strong ? ?} \rangle
 \end{aligned}$$



**Properties:**

- Find-S is guaranteed to output the most specific hypothesis within H that is consistent with positive training examples.
- Find-S algorithm ignores the negative examples.

**Drawbacks of Find-S:**

- Algorithm can't tell whether it has learned the right concept because it picks one hypothesis out of many possible ones
- Can't tell when training data inconsistent because it ignores the negative examples: doesn't account for noise
- Picks a maximally specific h, depending on H, there might be several correct hypothesis

**Example of training data set:**

**Task:** To determine whether the risk of defaulting on a loan is "high" or "low," based on a set of binary attributes. The task is to learn a hypothesis that best fits the data.

Gender	Age	Student	Previously Declined	HairLength	Employed	Type of collateral	First loan	Life Insurance	Risk
Male	Young	No	Yes	Short	No	House	No	No	Low
Male	Young	No	No	Long	Yes	Car	Yes	Yes	High
Male	Young	No	No	Short	Yes	Car	Yes	No	High
Male	Old	No	No	Short	No	House	No	Yes	High
Male	Old	No	No	Long	No	House	No	Yes	Low
Female	Young	Yes	No	Long	Yes	Car	No	Yes	High
Male	Old	No	Yes	Short	No	Car	Yes	No	Low
Male	Young	Yes	Yes	Long	No	House	No	No	High
Female	Old	No	Yes	Short	Yes	House	No	Yes	Low
Male	Old	Yes	Yes	Long	No	House	No	Yes	High
Male	Young	No	No	Short	Yes	Car	Yes	No	Low
Female	Old	No	No	Long	No	House	No	No	High
Male	Young	No	Yes	Short	No	Car	Yes	No	Low
Male	Old	No	No	Long	Yes	Car	No	No	High

**PROGRAM:**

```
import csv
#!/usr/bin/python
#list creation
#Most specific hypothesis
hypo=['%','%','%','%','%','%']
#Reading the csv file
with open("Training_examples.csv") as csv_file:
    readcsv = csv.reader(csv_file, delimiter=',')
    print(readcsv)
    data = []
    print("\nThe given training examples are:")
    for row in readcsv:
        print(row)
        #Copying only the positive training data to list'data'
        if row[-1].upper() == "YES":
            data.append(row)
    print("\nThe positive examples are:")
    for x in data:
        print(x)
    TotalEx = len(data)
    i=0
    j=0
    print("The steps of the Find-s algorithm are\n", hypo)
    list = []
    d=len(data[i])-1
    #Refining the hypothesis with the first training data
    hypo=data[0]
    #Refining the hypothesis for each training data encountered
    for i in range(1,TotalEx):
        for j in range(d):
            if hypo[j]!=data[i][j]:
```

```

        hypo[j]='?'

    print(hypo)

print("\nThe maximally specific Find-s hypothesis for the given training examples is")

list=[]

#Copying the hypothesis to the list'list'

print(hypo)

```

### Training examples.csv

```

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,Cool,Change,Yes

```

### OUTPUT:

```

Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD
64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\ML\ML Lab Programs\l-find-s\Find-s.py =====
<_csv.reader object at 0x00000208FBFEAF50>

The given training examples are:
['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', 'Cool', 'Change', 'Yes']

The positive examples are:
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
The steps of the Find-s algorithm are
['%', '%', '%', '%', '%', '%']
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', '?', 'Strong', '?', '?']

The maximally specific Find-s hypothesis for the given training examples is
['Sunny', 'Warm', '?', 'Strong', '?', '?']
>>> |

```

**Lab Program 2:** 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.

The candidate elimination algorithm incrementally builds the version space given a hypothesis space  $H$  and a set  $D$  of examples. The final output hypothesis are consistent with the given the training example. To find such hypotheses the simplest way is as follow:

- a. List all the hypotheses in  $H$
- b. Get each training example  $d$  from  $D$ 
  - i. Remove all the hypotheses which are not consistent with the training example  $d$ .

After execution of all the examples, the remaining hypotheses are all consistent with the given training example set. This procedure is only useful when the hypotheses set  $H$  is finite. This procedure is called as List-Then-Eliminate Algorithm.

The other algorithm is Candidate-Elimination Algorithm. The Candidate-Elimination Algorithm works on the same principle as the above List-Then-Eliminate Algorithm. However, it uses more compact representation of the version space. The version space is represented in term of most general and most specific members. To understand this following definitions are used.

**General Boundary  $G$ ,** with respect to hypothesis space  $H$  and training data  $D$ , is the set of maximally general hypotheses consistent with  $D$ .

$$G \equiv \{g \in H | \text{Consistent}(g, D) \wedge (\neg \exists g'' \in H) [(g'' > g) \wedge \text{Consistent}(g'', D)]\}$$

**Specific Boundary  $S$ ,** with respect to hypothesis space  $H$  and training data  $D$ , is the set of maximally specific hypotheses consistent with  $D$ .

$$S \equiv \{s \in H | \text{Consistent}(s, D) \wedge (\neg \exists s'' \in H) [(s > s'') \wedge \text{Consistent}(s'', D)]\}$$

### Candidate-Elimination Learning Algorithm

The Candidate-Elimination algorithm computes the version space containing all hypothesis from  $H$  that are consistent with an observed sequence of training examples. It begins by initializing the version space to the set of all hypotheses in  $H$ ; that is by initializing the  $G$  boundary set to contain most general hypothesis in  $H$

$$G_0 = \{(\text{?}, \text{?}, \text{?}, \text{?})\}$$

Then initialize the  $S$  boundary set to contain most specific hypothesis in  $H$

$$S_0 = \{(\text{?}, \text{?}, \text{?}, \text{?})\}$$

For each training example, these  $S$  and  $G$  boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypothesis found inconsistent with the new training examples. After execution of all the training examples, the computed version space contain all the hypotheses consistent with these training examples. The algorithm is summarized as below:

**Algorithm:**

$S \leftarrow$  minimally general hypotheses in  $H$ ,  $G \leftarrow$  maximally general hypotheses in  $H$  Initially any hypothesis is still possible

$S_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \quad G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$

- Initialize  $G$  to the set of maximally general hypotheses in  $H$
- Initialize  $S$  to the set of maximally specific hypotheses in  $H$
- For each training example  $d$ , do
  - If  $d$  is a positive example
    - Remove from  $G$  any hypothesis inconsistent with  $d$
    - For each hypothesis  $s$  in  $S$  that is not consistent with  $d$ 
      - Remove  $s$  from  $S$
      - Add to  $S$  all minimal generalizations  $h$  of  $s$  such that
        - $h$  is consistent with  $d$ , and some member of  $G$  is more general than  $h$
    - Remove from  $S$  any hypothesis that is more general than another hypothesis in  $S$
  - If  $d$  is a negative example
    - Remove from  $S$  any hypothesis inconsistent with  $d$
    - For each hypothesis  $g$  in  $G$  that is not consistent with  $d$ 
      - Remove  $g$  from  $G$
      - Add to  $G$  all minimal specializations  $h$  of  $g$  such that
        - $h$  is consistent with  $d$ , and some member of  $S$  is more specific than  $h$
    - – Remove from  $G$  any hypothesis that is less general than another hypothesis in  $G$

The CANDIDATE-ELIMINATION algorithm utilizes general-to-specific ordering of hypotheses to compute the version space by incrementally computing the sets of maximally specific ( $S$ ) and maximally general ( $G$ ) hypotheses.

**Table 1. Example data sample of EnjoySport data set**

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
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	Cool	Change	Yes

**PROGRAM:**

```
import random
import csv
def g_0(n):
    return ("?",)*n
def s_0(n):
    return (' $\phi$ ',)*n
def more_general(h1, h2):
    more_general_parts = []
    for x, y in zip(h1, h2):
        mg = x == "?" or (x != " $\phi$ " and (x == y or y == " $\phi$ "))
        more_general_parts.append(mg)
    return all(more_general_parts)
def fulfills(example, hypothesis):
    """ the implementation is the same as for hypotheses:
    return more_general(hypothesis, example)
def min_generalizations(h, x):
    h_new = list(h)
    for i in range(len(h)):
        if not fulfills(x[i], h[i]):
            h_new[i] = '?' if h[i] != ' $\phi$ ' else x[i]
    return [tuple(h_new)]
def min_specializations(h, domains, x):
    results = []
    for i in range(len(h)):
        if h[i] == "?":
            for val in domains[i]:
                if x[i] != val:
                    h_new = h[:i] + (val,) + h[i+1:]
                    results.append(h_new)
        elif h[i] != " $\phi$ ":
            h_new = h[:i] + (' $\phi$ ',) + h[i+1:]
```

```

        results.append(h_new)

    return results

with open('trainingexamples.csv') as csvFile:
    examples = [tuple(line) for line in csv.reader(csvFile)]

def get_domains(examples):
    d = [set() for i in examples[0]]
    for x in examples:
        for i, xi in enumerate(x):
            d[i].add(xi)
    return [list(sorted(x)) for x in d]

#get_domains(examples)

def candidate_elimination(examples):
    domains = get_domains(examples)[-1]
    G = set([g_0(len(domains))])
    S = set([s_0(len(domains))])
    i = 0
    print("\n G[{0}]:".format(i), G)
    print("\n S[{0}]:".format(i), S)
    for xcx in examples:
        i = i + 1
        x, cx = xcx[:-1], xcx[-1] # Splitting data into attributes and decisions
        if cx == 'Y': # x is positive example
            G = {g for g in G if fulfills(x, g)}
            S = generalize_S(x, G, S)
        else: # x is negative example
            S = {s for s in S if not fulfills(x, s)}
            G = specialize_G(x, domains, G, S)
        print("\n G[{0}]:".format(i), G)
        print("\n S[{0}]:".format(i), S)
    return

def generalize_S(x, G, S):
    S_prev = list(S)

```

```
    for s in S_prev:
        if not fulfills(x, s):
            S.remove(s)
            Splus = min_generalizations(s, x)
            ## keep only generalizations that have a counterpart in G
            S.update([h for h in Splus if any([more_general(g,h)
                                             for g in G])])
            ## remove hypotheses less specific than any other in S
            S.difference_update([h for h in S if
                                any([more_general(h, h1)
                                     for h1 in S if h != h1])])

    return S

def specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if fulfills(x, g):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
            ## keep only specializations that have a counterpart in S
            G.update([h for h in Gminus if any([more_general(h, s)
                                             for s in S])])
            ## remove hypotheses less general than any other in G
            G.difference_update([h for h in G if
                                any([more_general(g1, h)
                                     for g1 in G if h != g1])])

    return G

candidate_elimination(examples)
```

**Training examples.csv**

```
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,Cool,Change,Yes
```



**OUTPUT:**


```

Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD 64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: D:\ML\ML Lab Programs\2-candidate\Candidate with Package.py ====
initialization of specific_h and general_h
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
steps of Candidate Elimination Algorithm 1
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
steps of Candidate Elimination Algorithm 2
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
steps of Candidate Elimination Algorithm 3
['Sunny' 'Warm' 'High' 'Strong' '?' '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['Sunny' 'Warm' 'High' 'Strong' '?' '?']
Final General_h:
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
>>>

```

**Program 3:** 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.

Following terminologies are used in this algorithm

**Entropy :** Entropy is a measure of impurity

It is defined for a binary class with values a/b as:

$$\text{Entropy} = -p(a) \cdot \log(p(a)) - p(b) \cdot \log(p(b))$$

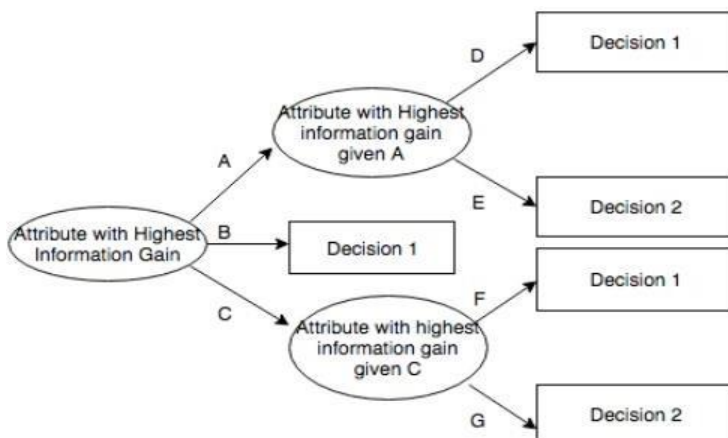
**Information Gain :** measuring the expected reduction in Entropy

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v=1}^n (|S_v|/|S|) * \text{Entropy}(S_v)$$

### THE PROCEDURE

- 1) In the ID3 algorithm, begin with the original set of attributes as the root node.
- 2) On each iteration of the algorithm, iterate through every unused attribute of the remaining set and calculates the entropy (or information gain) of that attribute.
- 3) Then, select the attribute which has the smallest entropy (or largest information gain) value.
- 4) The set of remaining attributes is then split by the selected attribute to produce subsets of the data.
- 5) The algorithm continues to recurse on each subset, considering only attributes never selected before.

#### Flow of Decision Tree



**Dataset Details**

playtennis dataset which has following structure  
Total number of instances=15  
Attributes=Outlook, Temperature, Humidity, Wind,  
Answer Target Concept=Answer

**PROCEDURE:**

Load learning sets S first, create decision tree root node 'rootNode', add learning set S into root node as its subset

For rootNode,

1. Calculate entropy of every attribute using the dataset
2. Split the set into subsets using the attribute for which entropy is minimum (or information gain is maximum)
3. Make a decision tree node containing that attribute
4. Recurse on subsets using renaming attributes

End

- This approach employs a top-down, greedy search through the space of possible decision trees.
- Algorithm starts by creating root node for the tree
- If all the examples are positive then return node with positive label
- If all the examples are negative then return node with negative label
- If Attributes is empty, Return the single-node tree Root, with label = most common value of Targetattribute in Example
- Otherwise -
  1. Calculate the entropy of every attribute using the data set S using formula
$$\text{Entropy} = -p(a) \cdot \log(p(a)) - p(b) \cdot \log(p(b))$$
  2. Split the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum) using formula
$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \text{ from } 1 \text{ to } n} (|S_v|/|S|) * \text{Entropy}(S_v)$$
  3. Make a decision tree node containing that attribute
  4. Recurring on subsets using remaining attributes.

Example:

Consider a dataset in which target concept has 9 positive and 5 negative examples. Then entropy is calculated as

$$\text{Entropy}(9+,5-)=(-9/14)\log(-9/14)-(5/14)\log(5/14)=0.940$$

## **SOFTWARE, PACKAGES AND LIBRARY REQUIREMENT**

Software packages- python2.7

Library – numpy, math

**Numpy**- Stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. You can import this library as:

Import numpy

**Math**- It provides access to the mathematical functions defined. you can import this library as:

Import math

## **INPUTS AND OUTPUTS**

**Input**- Input to the decision algorithm is a dataset stored in .csv file which consists of attributes, examples, target concept.

**Output**- For the given dataset decision tree algorithm produces the decision tree starting with rootnode which has highest information gain.

## **PROGRAM:**

```
import numpy as np
import math
from data_loader import read_data
class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""
    def __str__(self):
        return self.attribute
def subtables(data, col, delete):
```

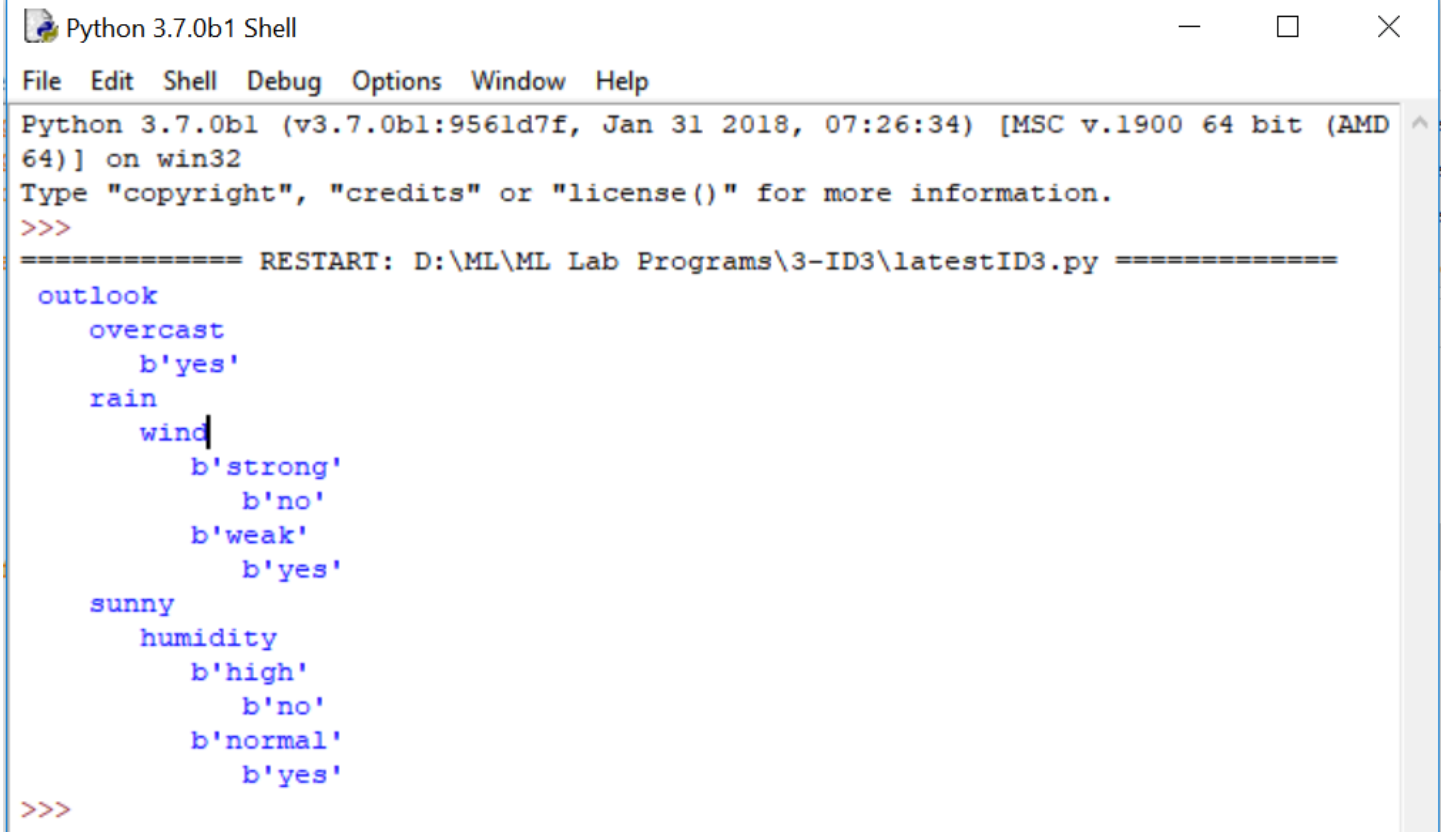
```
dict = {}
items = np.unique(data[:, col])
count = np.zeros((items.shape[0], 1), dtype=np.int32)
for x in range(items.shape[0]):
    for y in range(data.shape[0]):
        if data[y, col] == items[x]:
            count[x] += 1
for x in range(items.shape[0]):
    #dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
    dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
    pos = 0
    for y in range(data.shape[0]):
        if data[y, col] == items[x]:
            dict[items[x]][pos] = data[y]
            pos += 1
    if delete:
        dict[items[x]] = np.delete(dict[items[x]], col, 1)
return items, dict

def entropy(S):
    items = np.unique(S)
    if items.size == 1:
        return 0
    counts = np.zeros((items.shape[0], 1))
    sums = 0
    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)
    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums

def gain_ratio(data, col):
    items, dict = subtables(data, col, delete=False)
    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))
    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0]/(total_size * 1.0)
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])
        intrinsic[x] = ratio * math.log(ratio, 2)
    total_entropy = entropy(data[:, -1])
    iv = -1 * sum(intrinsic)
```

```
for x in range(entropies.shape[0]):
    total_entropy -= entropies[x]

return total_entropy / iv
def create_node(data, metadata):
    #TODO: Co jeśli information gain jest zerowe?
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node
    gains = np.zeros((data.shape[1] - 1, 1))
    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)
    split = np.argmax(gains)
    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)
    items, dict = subtables(data, split, delete=True)
    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))
    return node
def empty(size):
    s = ""
    for x in range(size):
        s += "  "
    return s
def print_tree(node, level):
    if node.answer != "":
        print(empty(level), node.answer)
        return
    print(empty(level), node.attribute)
    for value, n in node.children:
        print(empty(level + 1), value)
        print_tree(n, level + 2)
metadata, traindata = read_data("tennis.data")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
```

**OUTPUT:**A screenshot of a Python 3.7.0b1 Shell window. The window title is "Python 3.7.0b1 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The status bar at the bottom shows "Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD 64)] on win32". The main text area shows the following output:

```
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD 64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\ML\ML Lab Programs\3-ID3\latestID3.py =====
outlook
  overcast
    b'yes'
  rain
    wind
      b'strong'
      b'no'
      b'weak'
      b'yes'
  sunny
    humidity
      b'high'
      b'no'
      b'normal'
      b'yes'
>>>
```

**Program 4. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.**

Artificial neural networks (ANNs) are powerful tools for machine learning with applications in many areas including speech recognition, image classification, medical diagnosis, and spam filtering. It has been shown that ANNs can approximate any function to any degree of accuracy given enough neurons and training time. However, there is no guarantee on the number of neurons required or the time it will take to train them. These are the main disadvantages of using ANNs. Here we develop the BackPropogation algorithm which learns the weights for a multilayer network, given a network with a fixed set of units and interconnections. It employs gradient descent to attempt to minimize the squared error between the network output values and target values for these outputs.

**BACK PROPAGATION ALGORITHM:**

Multiple layer perceptron are effectively applied to handle tricky problems if trained with a vastly accepted algorithm identified as the back-propagation algorithm (error) in a supervised manner. It functions on learning law with error-correction. It is also a simplified version for the least mean square (LMS) filtering algorithm which is equally popular to error back-propagation algorithm.

In Error back-propagation training there are two computational passes via several network layers:

- A forward pass □ A backward pass.

In forward pass, vector input is applied to the nodes of the system propagating each layer,,s outcome to the next layer via network. To get the accurate response of the network, these outputs pass on from several layers and arrive at a set of outputs. In forward pass network weights are permanent. On other hand in the backward pass, weights are adjusted according to rule for error correction. Error signal is the actual response of the network minus the desired response.

The propagation of this error signal through the network is towards backward in direction opposite to the connections of synaptic. The move the real response of network closer to the favored response, tuning of weights is to be done. There are three unique features of a multilayer perception:

- 1)For each neuron in any system, its illustration has an activation function that is non- linear. The logistical function is used to define a function which is sigmoid.
- 2)There are layer(s) of hidden neurons not contained in the input or the output present in the neural network. The study over complex tasks is facilitated by these hidden neurons.

Connectivity degree is high in network. Weight's population should be changed if there is a requirement to alter the connectivity of the network.

The stochastic gradient descent version of the BACKPROPAGATION algorithm for feed forward networks containing two layers of sigmoid units.

Step 1: begins by constructing a network with the desired number of hidden and output units and initializing all network weights to small random values. . For each training example, it applies the network to the example, calculates the error of the network output for this example, computes the gradient with respect to the error on this example, then updates all weights in the network. This gradient descent step is iterated (often thousands of times, using the same training examples multiple



times) until the network performs acceptably well.

Step 2: The gradient descent weight-update rule is similar to the delta training rule. The only difference is that the error  $(t - o)$  in the delta rule is replaced by a more complex error term  $\delta_j$ .

Step 3: updates weights incrementally, following the Presentation of each training example. This corresponds to a stochastic approximation to gradient descent. To obtain the true gradient of  $E$  one would sum the  $\delta_j, x_{ji}$  values over all training examples before altering weight values.

Step 4: The weight-update loop in BACKPROPAGATION may be iterated thousands of times in a typical application. A variety of termination conditions can be used to halt the procedure.

One may choose to halt after a fixed number of iterations through the loop, or once the error on the training examples falls below some threshold.

### **PROGRAM:**

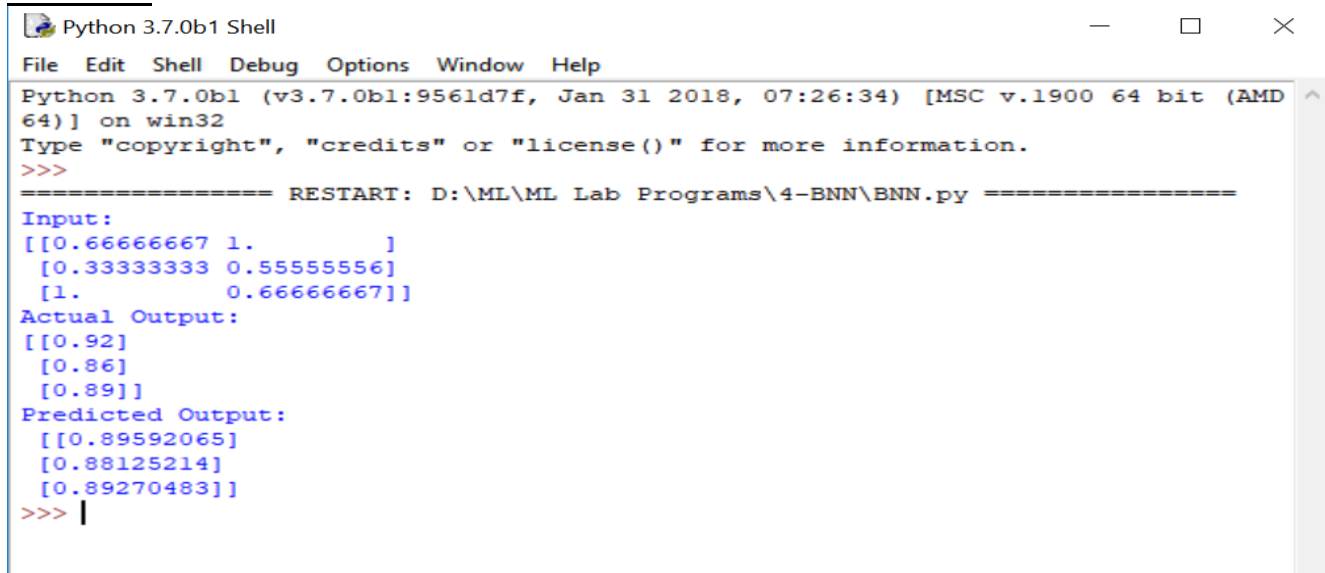
```
import numpy as np
X = np.array([[2, 9], [1, 5], [3, 6]], dtype=float)
y = np.array([92], [86], [89]), dtype=float)

X = X/np.amax(X,axis=0) # maximum of X array longitudinally
y = y/100
#Sigmoid Function
def sigmoid (x):
    return 1/(1 + np.exp(-x))
#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)
#Variable initialization
epoch=7000 #Setting training iterations
lr=0.1 #Setting learning rate
inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
#draws a random range of numbers uniformly of dimx*y
for i in range(epoch):
```

```
#Forward Propagation
hinp1=np.dot(X,wh)
hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp= outinp1+ bout
output = sigmoid(outinp)

#Backpropagation
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO* outgrad
EH = d_output.dot(wout.T)
hiddengrad = derivatives_sigmoid(hlayer_act)
#how much hidden layer wts contributed to error
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_output) *lr
# dotproduct of nextlayererror and currentlayerop
#bout += np.sum(d_output, axis=0,keepdims=True)*lr
wh += X.T.dot(d_hiddenlayer) *lr
#bh += np.sum(d_hiddenlayer, axis=0,keepdims=True) *lr
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

### **OUTPUT:**



```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD
64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\ML\ML Lab Programs\4-BNN\BNN.py =====
Input:
[[0.66666667 1.          ]
 [0.33333333 0.55555556]
 [1.          0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
[[0.89592065]
 [0.88125214]
 [0.89270483]]
>>> |
```

**Program 5:** 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.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as „Naive“.

Bayes theorem provides a way of calculating posterior probability  $P(c|x)$  from  $P(c)$ ,  $P(x)$  and  $P(x|c)$ . Look at the equation below:

### **1)Handling Of Data:**

- load the data from the CSV file and split in to training and test data set.
- Training data set can be used to by Naïve Bayes to make predictions.
- And Test data set can be used to evaluate the accuracy of the model.

### **2)Summarize Data:**

The summary of the training data collected involves the mean and the standard deviation for each attribute, by class value.

- These are required when making predictions to calculate the probability of specific attribute values belonging to each class value.
- summary data can be break down into the following sub-tasks:

**a)Separate Data By Class:** The first task is to separate the training dataset instances by class value so that we can calculate statistics for each class. We can do that by creating a map of each class value to a list of instances that belong to that class and sort the entire dataset of instances into the appropriate lists.

**b)Calculate Mean:**We need to calculate the mean of each attribute for a class value. The mean is the central middle or central tendency of the data, and we will use it as the middle of our gaussian distribution when calculating probabilities.

**c)Calculate Standard Deviation:** We also need to calculate the standard deviation of each attribute for a class value. The standard deviation describes the variation of spread of the data, and we will use it to characterize the expected spread of each attribute in our Gaussian distribution when calculating probabilities.

**d)Summarize Dataset:** For a given list of instances (for a class value) we can calculate the mean and the standard deviation for each attribute. The zip function groups the values for each attribute across our data instances into their own lists so that we can compute the mean and standard deviation values for the attribute.

**e)Summarize Attributes By Class:** We can pull it all together by first separating our training dataset into instances grouped by class. Then calculate the summaries for each attribute.

**3) Make Predictions:**

Making predictions involves calculating the probability that a given data instance belongs to each class, then selecting the class with the largest probability as the prediction.

Finally, estimation of the accuracy of the model by making predictions for each data instance in the test dataset.

**4) Evaluate Accuracy:** The predictions can be compared to the class values in the test dataset and a classification\ accuracy can be calculated as an accuracy ratio between 0% and 100%.

**PROGRAM:**

```
print("\nNaive Bayes Classifier for concept learning problem")
import csv
import random
import math
import operator
def safe_div(x,y):
    if y == 0:
        return 0
    return x / y

def loadCsv(filename):
    lines = csv.reader(open(filename))
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    i=0
    while len(trainSet) < trainSize:
        #index = random.randrange(len(copy))

        trainSet.append(copy.pop(i))
    return [trainSet, copy]

def separateByClass(dataset):
    separated = { }
```

```
for i in range(len(dataset)):
    vector = dataset[i]
    if (vector[-1] not in separated):
        separated[vector[-1]] = []
    separated[vector[-1]].append(vector)
return separated
```

```
def mean(numbers):
    return safe_div(sum(numbers),float(len(numbers)))
```

```
def stdev(numbers):
    avg = mean(numbers)
    variance = safe_div(sum([pow(x-avg,2) for x in numbers]),float(len(numbers)-1))
    return math.sqrt(variance)
```

#Calculate mean and stddev for each attribute in the dataset

```
def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries
```

```
def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = { }
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries
```

```
def calculateProbability(x, mean, stdev):
    exponent = math.exp(-safe_div(math.pow(x-mean,2),(2*math.pow(stdev,2))))
    final = safe_div(1 , (math.sqrt(2*math.pi) * stdev)) * exponent
    return final
```

```
def calculateClassProbabilities(summaries, inputVector):
    probabilities = { }
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
```

```
        probabilities[classValue] *= calculateProbability(x, mean, stdev)
    return probabilities

def predict(summaries, inputVector):
    probabilities = calculateClassProbabilities(summaries, inputVector)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions

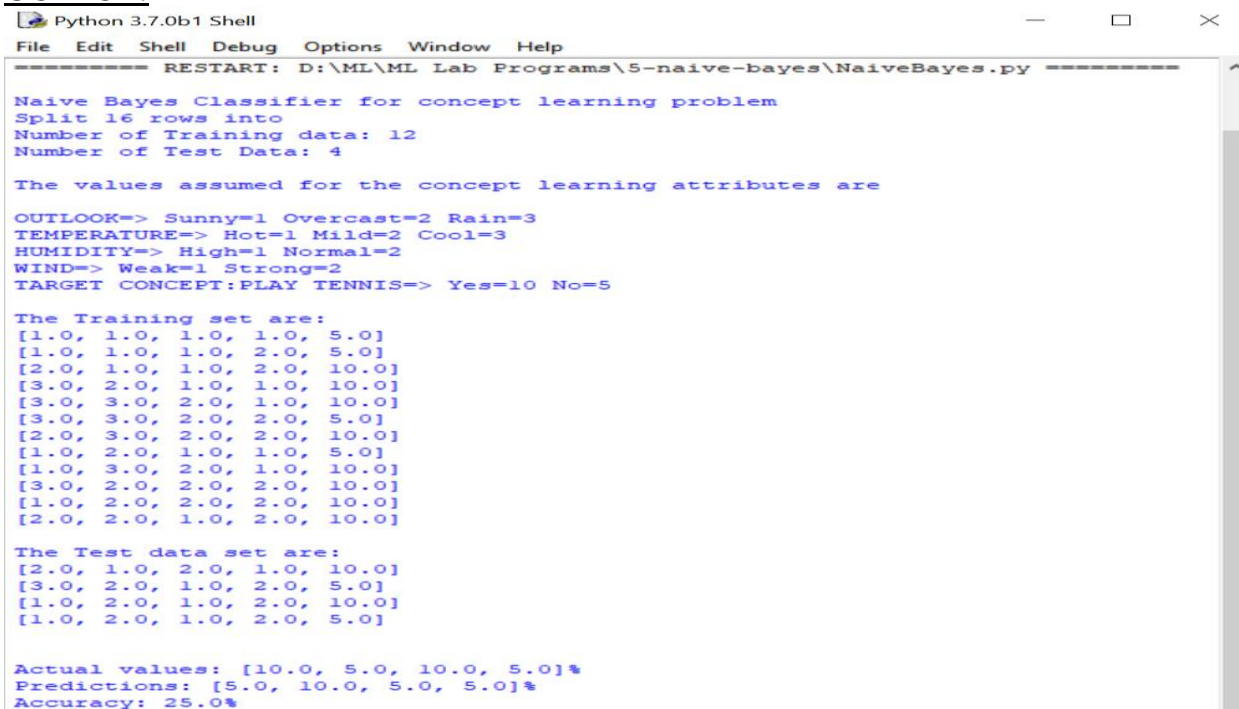
def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
        if testSet[i][-1] == predictions[i]:
            correct += 1
    accuracy = safe_div(correct, float(len(testSet))) * 100.0
    return accuracy

def main():
    filename = 'ConceptLearning.csv'
    splitRatio = 0.80
    dataset = loadCsv(filename)
    trainingSet, testSet = splitDataset(dataset, splitRatio)
    print('Split {0} rows into'.format(len(dataset)))
    print('Number of Training data: ' + (repr(len(trainingSet))))
    print('Number of Test Data: ' + (repr(len(testSet))))
    print("\nThe values assumed for the concept learning attributes are\n")
    print("OUTLOOK=> Sunny=1 Overcast=2 Rain=3\nTEMPERATURE=> Hot=1 Mild=2\nCool=3\nHUMIDITY=> High=1 Normal=2\nWIND=> Weak=1 Strong=2")
    print("TARGET CONCEPT:PLAY TENNIS=> Yes=10 No=5")
    print("\nThe Training set are:")
```

```
for x in trainingSet:
    print(x)
print("\nThe Test data set are:")
for x in testSet:
    print(x)
print("\n")
# prepare model
summaries = summarizeByClass(trainingSet)
# test model
predictions = getPredictions(summaries, testSet)
actual = []
for i in range(len(testSet)):
    vector = testSet[i]
    actual.append(vector[-1])

# Since there are five attribute values, each attribute constitutes to 20% accuracy. So if all attributes
match with predictions then 100% accuracy
print('Actual values: {0}%'.format(actual))
print('Predictions: {0}%'.format(predictions))
accuracy = getAccuracy(testSet, predictions)
print('Accuracy: {0}%'.format(accuracy))

main()
```

**OUTPUT:**

```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
===== RESTART: D:\ML\ML Lab Programs\5-naive-bayes\NaiveBayes.py =====

Naive Bayes Classifier for concept learning problem
Split 16 rows into
Number of Training data: 12
Number of Test Data: 4

The values assumed for the concept learning attributes are
OUTLOOK=> Sunny=1 Overcast=2 Rain=3
TEMPERATURE=> Hot=1 Mild=2 Cool=3
HUMIDITY=> High=1 Normal=2
WIND=> Weak=1 Strong=2
TARGET CONCEPT:PLAY TENNIS=> Yes=10 No=5

The Training set are:
[1.0, 1.0, 1.0, 1.0, 5.0]
[1.0, 1.0, 1.0, 2.0, 5.0]
[2.0, 1.0, 1.0, 2.0, 10.0]
[3.0, 2.0, 1.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 2.0, 5.0]
[2.0, 3.0, 2.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 1.0, 5.0]
[1.0, 3.0, 2.0, 1.0, 10.0]
[3.0, 2.0, 2.0, 2.0, 10.0]
[1.0, 2.0, 2.0, 2.0, 10.0]
[2.0, 2.0, 1.0, 2.0, 10.0]

The Test data set are:
[2.0, 1.0, 2.0, 1.0, 10.0]
[3.0, 2.0, 1.0, 2.0, 5.0]
[1.0, 2.0, 1.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 2.0, 5.0]

Actual values: [10.0, 5.0, 10.0, 5.0]*
Predictions: [5.0, 10.0, 5.0, 5.0]*
Accuracy: 25.0%
```

**Program 6:** Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set

### Naïve bayes Classifier

This classifier can be used when moderate to large training set is available.

- **Problem statement:**
- Given features  $X_1, X_2, \dots, X_n$
- Predict a label  $Y$

### Bayes Classifier

$$P(c/\mathbf{x}) \propto P(\mathbf{x}/c)P(c) = P(x_1, \dots, x_n | c)P(c) \text{ for } c = c_1, \dots, c_L.$$

Determining the joint distribution is infeasible so Naïve Bayes classifier works on the assumption that features are independent  
Naïve Bayes Classifier

$$\begin{aligned} P(x_1, x_2, \dots, x_n | c) &= P(x_1 | x_2, \dots, x_n, c)P(x_2, \dots, x_n | c) \\ &= \underline{P(x_1 | c)}P(x_2, \dots, x_n | c) \\ &= P(x_1 | c)P(x_2 | c) \cdots P(x_n | c) \end{aligned}$$

Apply the MAP classification rule: assign to  $c^*$  if

$$[P(a_1 | c^*) \cdots P(a_n | c^*)]P(c^*) > [P(a_1 | c) \cdots P(a_n | c)]P(c), \quad c \neq c^*, c = c_1, \dots, c_L$$

### Learning Phase

For each target value of  $c_i (c_i = c_1, \dots, c_L)$

$\hat{P}(c_i) \leftarrow$  estimate  $P(c_i)$  with examples in  $S$ ;

For every feature value  $x_{jk}$  of each feature  $x_j (j = 1, \dots, F; k = 1, \dots, N_j)$

$\hat{P}(x_j = x_{jk} | c_i) \leftarrow$  estimate  $P(x_{jk} | c_i)$  with examples in  $S$ ;

### Test Phase

$$[\hat{P}(a'_1 | c^*) \cdots \hat{P}(a'_n | c^*)]\hat{P}(c^*) > [\hat{P}(a'_1 | c) \cdots \hat{P}(a'_n | c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \dots, c_L$$



Algorithm to train and derive inference from the Naïve Bayes model

```
TRAINMULTINOMIALNB(C, ID)
1  V ← EXTRACTVOCABULARY(ID)
2  N ← COUNTDOCS(ID)
3  for each c ∈ C
4    do Nc ← COUNTDOCSINCLASS(ID, c)
5      prior[c] ← Nc / N
6      textc ← CONCATENATETEXTOFALLDOCSINCLASS(ID, c)
7      for each t ∈ V
8        do Tct ← COUNTTOKENSOFTERM(textc, t)
9        for each t ∈ V
10       do condprob[t][c] ←  $\frac{T_{ct}+1}{\sum_{t'} (T_{ct'}+1)}$ 
11  return V, prior, condprob
```

```
APPLYMULTINOMIALNB(C, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3    do score[c] ← log prior[c]
4      for each t ∈ W
5        do score[c] += log condprob[t][c]
6  return arg maxc ∈ C score[c]
```

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

**Note** :To eliminate zeros, we use where

$$B = |V|$$

is the number of terms in the vocabulary.

*add-one* or *Laplace smoothing*, which simply adds one to each count

**Accuracy** = No of test samples correctly classified /Total no of samples\*100

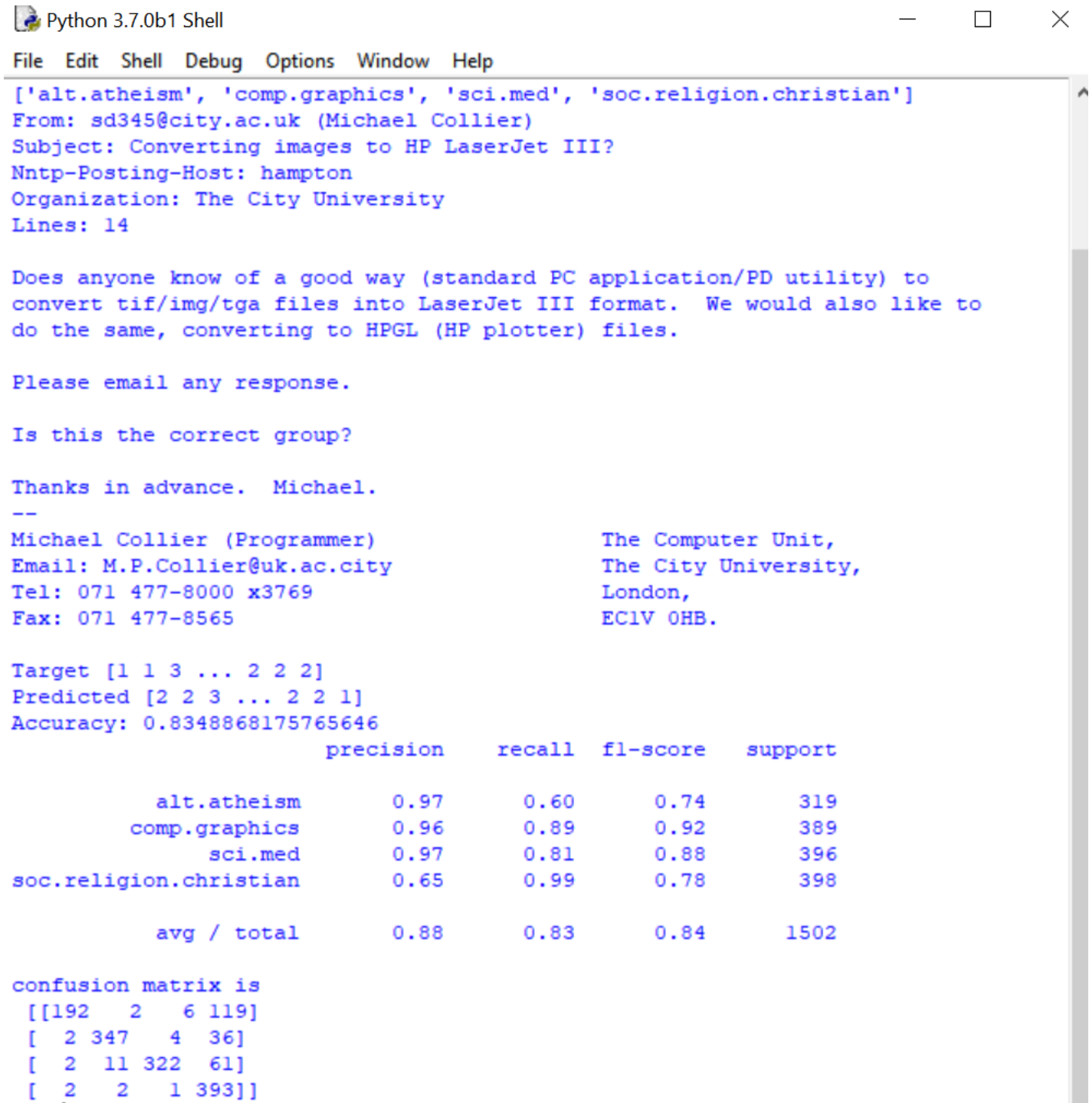
**Precision** = true positives / (true positives + false positives)

**Recall**= true positives / (true positives + false negatives)

**PROGRAM:**

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import numpy as np
categories = ['alt.atheism', 'soc.religion.christian', 'comp.graphics', 'sci.med']
twenty_train = fetch_20newsgroups(subset='train',categories=categories,shuffle=True)
twenty_test = fetch_20newsgroups(subset='test',categories=categories,shuffle=True)
print(len(twenty_train.data))
print(len(twenty_test.data))
print(twenty_train.target_names)
print("\n".join(twenty_train.data[0].split("\n")))
print("Target",twenty_train.target)
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_tf = count_vect.fit_transform(twenty_train.data)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_tf)
#X_train_tfidf.shape
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn import metrics
mod = MultinomialNB()
mod.fit(X_train_tfidf, twenty_train.target)
X_test_tf = count_vect.transform(twenty_test.data)
X_test_tfidf = tfidf_transformer.transform(X_test_tf)
predicted = mod.predict(X_test_tfidf)
print("Predicted",predicted)
print("Accuracy:", accuracy_score(twenty_test.target, predicted))
print(classification_report(twenty_test.target,predicted,target_names=twenty_test.target_names))

print("confusion matrix is \n",metrics.confusion_matrix(twenty_test.target, predicted))
```

**OUTPUT:**


```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']
From: sd345@city.ac.uk (Michael Collier)
Subject: Converting images to HP LaserJet III?
Nntp-Posting-Host: hampton
Organization: The City University
Lines: 14

Does anyone know of a good way (standard PC application/PD utility) to
convert tif/img/tga files into LaserJet III format. We would also like to
do the same, converting to HPGL (HP plotter) files.

Please email any response.

Is this the correct group?

Thanks in advance. Michael.
--
Michael Collier (Programmer)           The Computer Unit,
Email: M.P.Collier@uk.ac.city          The City University,
Tel: 071 477-8000 x3769                London,
Fax: 071 477-8565                      EC1V 0HB.

Target [1 1 3 ... 2 2 2]
Predicted [2 2 3 ... 2 2 1]
Accuracy: 0.8348868175765646

              precision    recall  f1-score   support

    alt.atheism           0.97      0.60      0.74       319
  comp.graphics           0.96      0.89      0.92       389
        sci.med           0.97      0.81      0.88       396
soc.religion.christian     0.65      0.99      0.78       398

    avg / total           0.88      0.83      0.84      1502

confusion matrix is
[[192  2  6 119]
 [ 2 347  4  36]
 [ 2  11 322  61]
 [ 2  2  1 393]]
```

**Program 7:** Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

**A Bayesian network** – also called a belief network or causal probabilistic network- is a graphical representation of probabilistic information: It is a directed acyclic graph in which nodes represent random (stochastic) variables, and links between nodes represent direct probabilistic influences between the variables. In this formalism, propositions are given numerical probability values signifying the degree of belief accorded them, and the values are combined and manipulated according to the rules of probability theory. Typically, the direction of a connection between nodes indicates a causal influence or class-property relationship

Bayesian statistical inference uses probabilities for both prior and future events to estimate the uncertainty that is inevitable with prediction. The fundamental concept in Bayesian networks is that probabilities can be assigned to parameter values,

**Bayes theorem**, these probabilities can be updated given new data. In Bayesian models the parameter is viewed as a domain variable, with a probability distribution, since the actual value of the parameter is unknown. The causal links between the variables are represented by arrows in the model. The model is strong if the arrows can be interpreted as causal mechanisms. The relationships can, alternatively, be considered an association and this type of model would be viewed more cautiously. Bayesian networks can express the relationships between diagnoses, physical findings, laboratory test results, and imaging study findings. Physicians can determine the a priori (“pre-test”) probability of a disease, and then incorporate laboratory and imaging results to calculate the a posteriori (“post-test”) probability

**Tool boxes** <http://bayespy.org/>

BayesPy provides tools for Bayesian inference with Python. The user constructs a model as a Bayesian network, observes data and runs posterior inference. The goal is to provide a tool which is efficient, flexible and extendable enough for expert use but also accessible for more casual users.

Bayesian Belief Network is a specific type of Causal belief network. Nodes represent Stochastic Variables(features) and arcs identify direct causal influences between linked variables. Bayesian Calculus is used to determine state probabilities of each node or variable from conditional and prior probabilities.

Following steps are used to build the Bayesian network

Step 1: Identify the variables which is set of attributes specified in the dataset(ex Medical Dataset)

Step2: Determine the domain of each variable that is set of values a variable may take

Step3: Create a directed graph network of nodes where each node represents the attribute and edges represent parent child relationship. Edge represents that the child variable is conditionally dependent on the parent.

Step4 : Determine the prior and conditional probability for each attribute

Step5 : Perform the inference on the model and determine the marginal probabilities

**PROGRAM:**

```
import bayespy as bp
import numpy as np
import csv
from colorama import init
from colorama import Fore, Back, Style
init()

# Define Parameter Enum values
#Age
ageEnum = {'SuperSeniorCitizen':0, 'SeniorCitizen':1, 'MiddleAged':2, 'Youth':3, 'Teen':4}
# Gender
genderEnum = {'Male':0, 'Female':1}
# FamilyHistory
familyHistoryEnum = {'Yes':0, 'No':1}
# Diet(Calorie Intake)
dietEnum = {'High':0, 'Medium':1, 'Low':2}
# LifeStyle
lifeStyleEnum = {'Athlete':0, 'Active':1, 'Moderate':2, 'Sedetary':3}
# Cholesterol
cholesterolEnum = {'High':0, 'BorderLine':1, 'Normal':2}
# HeartDisease
heartDiseaseEnum = {'Yes':0, 'No':1}
#heart_disease_data.csv
with open('heart_disease_data.csv') as csvfile:
    lines = csv.reader(csvfile)
    dataset = list(lines)
    data = []
    for x in dataset:

data.append([ageEnum[x[0]],genderEnum[x[1]],familyHistoryEnum[x[2]],dietEnum[x[3]],lifeStyleEnum[
x[4]],cholesterolEnum[x[5]],heartDiseaseEnum[x[6]]])
# Training data for machine learning todo: should import from csv
data = np.array(data)
N = len(data)

# Input data column assignment

# Group assignment probabilities
```

```
p_age = bp.nodes.Dirichlet(1.0*np.ones(5))
# Group assignments for nodes
age = bp.nodes.Categorical(p_age, plates=(N,))
age.observe(data[:,0])

p_gender = bp.nodes.Dirichlet(1.0*np.ones(2))
gender = bp.nodes.Categorical(p_gender, plates=(N,))
gender.observe(data[:,1])

p_familyhistory = bp.nodes.Dirichlet(1.0*np.ones(2))
familyhistory = bp.nodes.Categorical(p_familyhistory, plates=(N,))
familyhistory.observe(data[:,2])

p_diet = bp.nodes.Dirichlet(1.0*np.ones(3))
diet = bp.nodes.Categorical(p_diet, plates=(N,))
diet.observe(data[:,3])

p_lifestyle = bp.nodes.Dirichlet(1.0*np.ones(4))
lifestyle = bp.nodes.Categorical(p_lifestyle, plates=(N,))
lifestyle.observe(data[:,4])

p_cholesterol = bp.nodes.Dirichlet(1.0*np.ones(3))
cholesterol = bp.nodes.Categorical(p_cholesterol, plates=(N,))
cholesterol.observe(data[:,5])

# Prepare nodes and establish edges
# np.ones(2) -> HeartDisease has 2 options Yes/No
# plates(5, 2, 2, 3, 4, 3) -> corresponds to options present for domain values
p_heartdisease = bp.nodes.Dirichlet(np.ones(2), plates=(5, 2, 2, 3, 4, 3))
heartdisease = bp.nodes.MultiMixture([age, gender, familyhistory, diet, lifestyle, cholesterol],
bp.nodes.Categorical, p_heartdisease)
heartdisease.observe(data[:,6])
p_heartdisease.update()

# Sample Test with hardcoded values
#print("Sample Probability")
#print("Probability(HeartDisease|Age=SuperSeniorCitizen, Gender=Female, FamilyHistory=Yes,
DietIntake=Medium, LifeStyle=Sedetary, Cholesterol=High)")
#print(bp.nodes.MultiMixture([ageEnum['SuperSeniorCitizen'], genderEnum['Female'],
familyHistoryEnum['Yes'], dietEnum['Medium'], lifeStyleEnum['Sedetary'], cholesterolEnum['High']],
```

```
bp.nodes.Categorical, p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']])
```

```
# Interactive Test
```

```
m = 0
```

```
while m == 0:
```

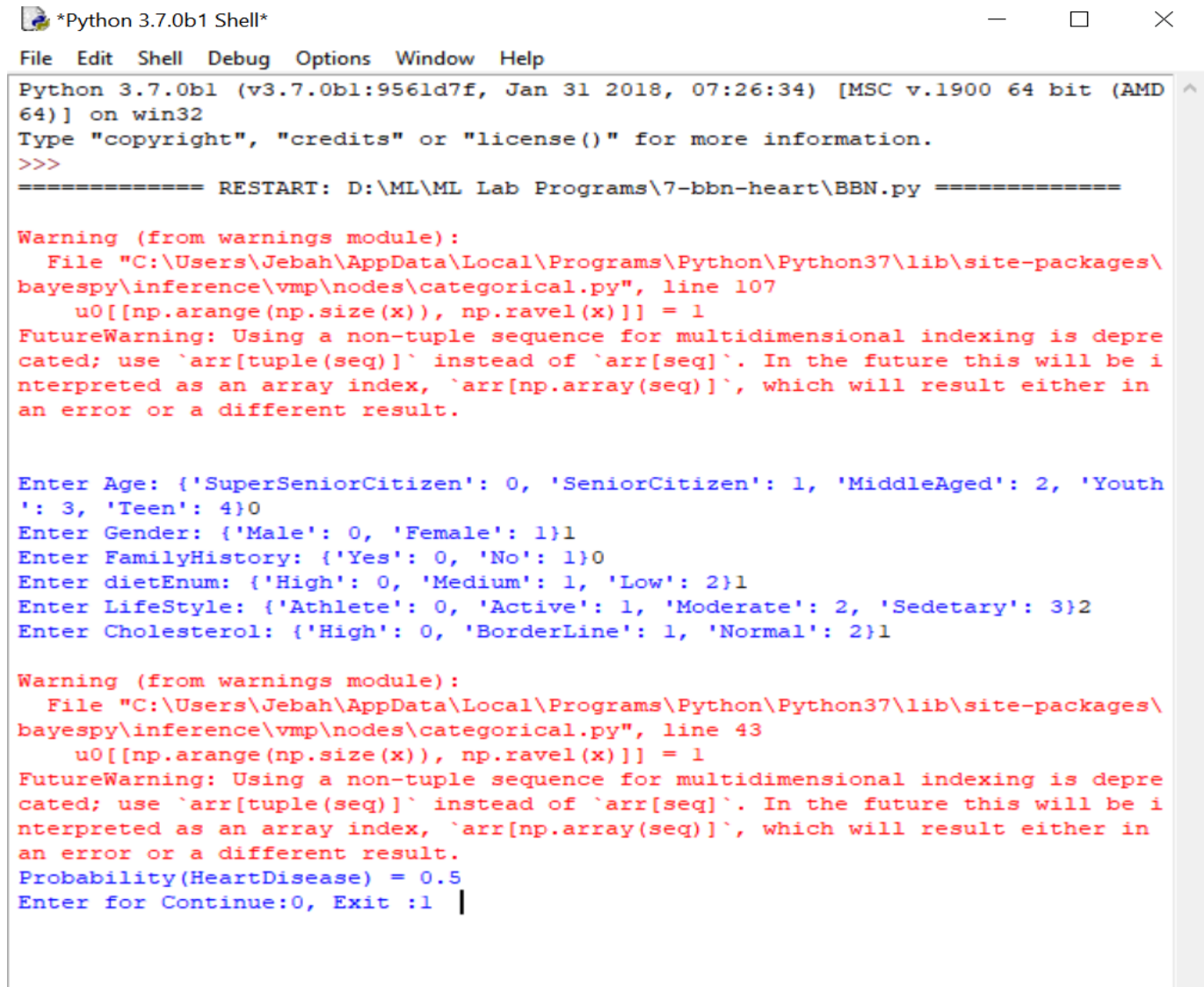
```
    print("\n")
```

```
    res = bp.nodes.MultiMixture([int(input('Enter Age: ' + str(ageEnum))), int(input('Enter Gender: ' + str(genderEnum))), int(input('Enter FamilyHistory: ' + str(familyHistoryEnum))), int(input('Enter dietEnum: ' + str(dietEnum))), int(input('Enter LifeStyle: ' + str(lifeStyleEnum))), int(input('Enter Cholesterol: ' + str(cholesterolEnum))), bp.nodes.Categorical, p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']])
```

```
    print("Probability(HeartDisease) = " + str(res))
```

```
    #print(Style.RESET_ALL)
```

```
    m = int(input("Enter for Continue:0, Exit :1 "))
```

**OUTPUT:**


```

Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\ML\ML Lab Programs\7-bbn-heart\BBN.py =====

Warning (from warnings module):
  File "C:\Users\Jebah\AppData\Local\Programs\Python\Python37\lib\site-packages\
bayespy\inference\vmp\nodes\categorical.py", line 107
    u0[[np.arange(np.size(x)), np.ravel(x)]] = 1
FutureWarning: Using a non-tuple sequence for multidimensional indexing is depre
cated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be i
nterpreted as an array index, `arr[np.array(seq)]`, which will result either in
an error or a different result.

Enter Age: {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1, 'MiddleAged': 2, 'Youth
': 3, 'Teen': 4}0
Enter Gender: {'Male': 0, 'Female': 1}1
Enter FamilyHistory: {'Yes': 0, 'No': 1}0
Enter dietEnum: {'High': 0, 'Medium': 1, 'Low': 2}1
Enter LifeStyle: {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}2
Enter Cholesterol: {'High': 0, 'BorderLine': 1, 'Normal': 2}1

Warning (from warnings module):
  File "C:\Users\Jebah\AppData\Local\Programs\Python\Python37\lib\site-packages\
bayespy\inference\vmp\nodes\categorical.py", line 43
    u0[[np.arange(np.size(x)), np.ravel(x)]] = 1
FutureWarning: Using a non-tuple sequence for multidimensional indexing is depre
cated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be i
nterpreted as an array index, `arr[np.array(seq)]`, which will result either in
an error or a different result.
Probability(HeartDisease) = 0.5
Enter for Continue:0, Exit :1 |

```



**Program 8: 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. Add Python ML library classes/API in the program.**

### Unsupervised Learning:

In machine learning, unsupervised learning is a class of problems in which one seeks to determine how the data are organized. It is distinguished from supervised learning (and reinforcement learning) is that the learner is given only unlabeled examples.

#### ➤ Clustering Algorithms -

##### 1. K-means clustering:

- It is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups).
- The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K.

Data points are clustered based on feature similarity.

The results of the K-means clustering algorithm are:

- The centroids of the K clusters, which can be used to label new data
- Labels for the training data (each data point is assigned to a single cluster) Each centroid of a cluster is a collection of feature values which define the resulting groups.
- Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

The k-means is a partitional clustering algorithm.

Let the set of data points (or instances) be as follows:

$D = \{x_1, x_2, \dots, x_n\}$ , where

$x = (x_{i1}, x_{i2}, \dots, x_{ir})$ , is a vector in a real-valued space  $X \subseteq \mathbb{R}^r$ , and  $r$  is the number of attributes in the data.

The k-means algorithm partitions the given data into  $k$  clusters with each cluster having a center called a centroid.

$k$  is specified by the user.

Given  $k$ , the k-means algorithm works as follows:

##### Algorithm K-means( $k, D$ )

1. Identify the  $k$  data points as the initial centroids (cluster centers).

2. Repeat step 1.
3. For each data point  $x \in D$  do.
4. Compute the distance from  $x$  to the centroid.
5. Assign  $x$  to the closest centroid (a centroid represents a cluster).
6. Re-compute the centroids using the current cluster memberships until the stopping criterion is met.

## **2. Expectation–maximization (EM Algorithm)**

- EM algorithm is an iterative method to find maximum likelihood estimates of parameters in statistical models, where the model depends on unobserved latent variables.
- Iteratively learn probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories “Randomly label” data
- Learn initial probabilistic model by estimating model parameters  $\theta$  from randomly labeled data
- Iterate until convergence:
  1. Expectation (E-step): Compute  $P(c_i | E)$  for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  2. Maximization (M-step): Re-estimate the model parameters,  $\theta$ , from the probabilistically re-labeled data.

### **The EM Algorithm for Gaussian Mixtures**

#### **Algorithm:**

An arbitrary initial hypothesis  $h = \langle \mu_1, \mu_2, \dots, \mu_k \rangle$  is chosen. The

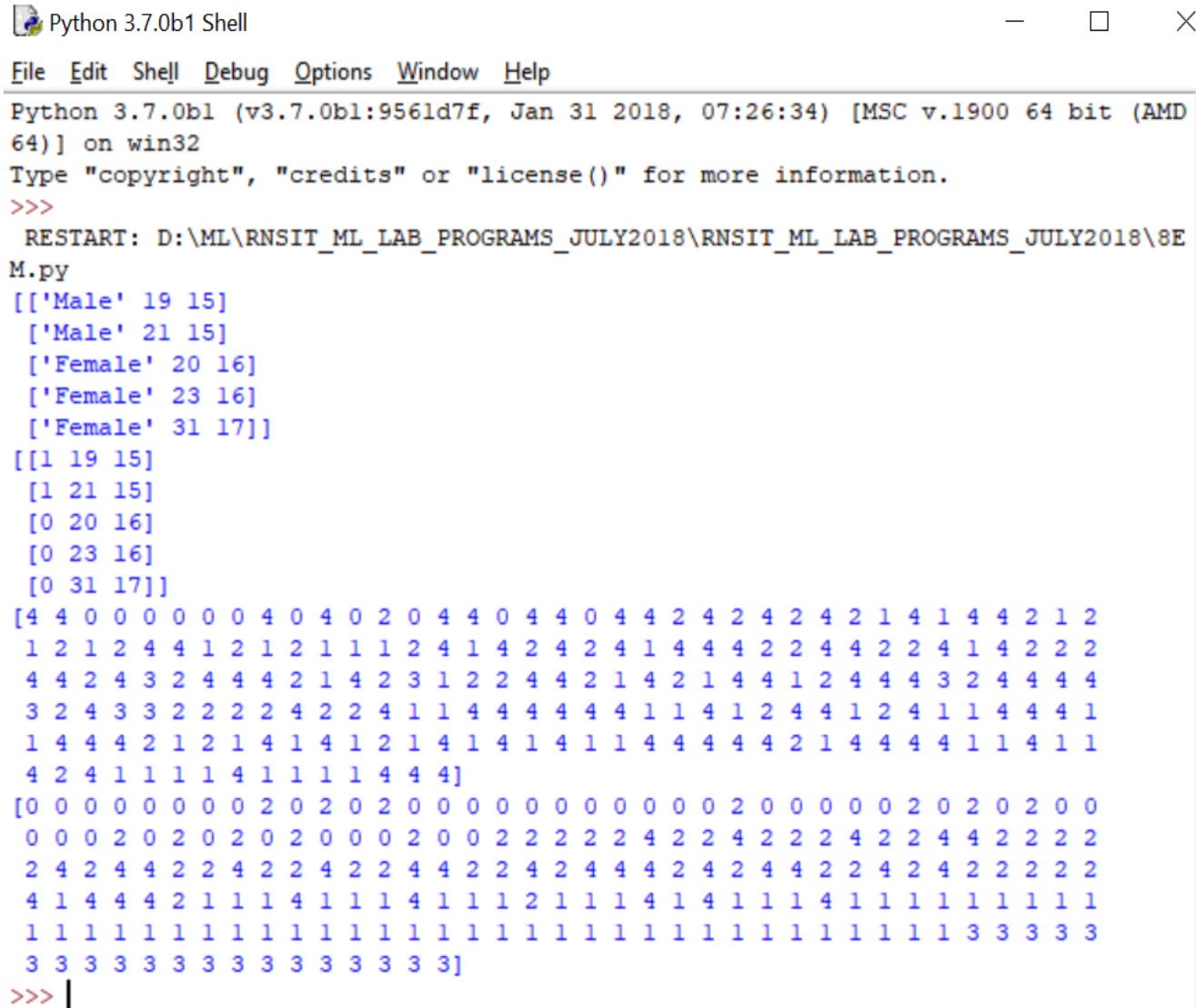
EM Algorithm iterates over two steps:

Step 1 (Estimation, E): Calculate the expected value  $E[z_{ij}]$  of each hidden variable  $z_{ij}$ , assuming that the current hypothesis  $h = \langle \mu_1, \mu_2, \dots, \mu_k \rangle$  holds.

Step 2 (Maximization, M): Calculate a new maximum likelihood hypothesis  $h' = \langle \mu_1', \mu_2', \dots, \mu_k' \rangle$ , assuming the value taken on by each hidden variable  $z_{ij}$  is its expected value  $E[z_{ij}]$  calculated in step 1. Then replace the hypothesis  $h = \langle \mu_1, \mu_2, \dots, \mu_k \rangle$  by the new hypothesis  $h' = \langle \mu_1', \mu_2', \dots, \mu_k' \rangle$  and iterate.

**CHOICE 1- PROGRAM:**

```
# coding: utf-8
# ### Comparison of Expectation Maximization(EM) vs K-Means Clustering
Algorithm
# ### Step 1 : Load required packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# ### Step 2 : Load the inbuilt data or the csv/excel file into pandas data frame and
clean the data
df = pd.read_csv("../data/Mall_Customers.csv")
df.head()
# ### Step 3 : Create the Feature Matrix and check the top 5 rows
x = df.iloc[:, 1:-1].values
print(x[:5]) #Encode Gender Column using LabelEncoder
from sklearn.preprocessing import LabelEncoder
x[:, 0] = LabelEncoder().fit_transform(x[:, 0])
print(x[:5])
# ### Step 4 : Instantiate a EM Model and K-Means Model and train them
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
em_cluster = GaussianMixture(n_components = 5)
km_cluster = KMeans(n_clusters=5)
em_cluster.fit(x)
km_cluster.fit(x)
# ### Step 6 : Use the model to predict the cluster labels
em_predictions = em_cluster.predict(x)
km_predictions = km_cluster.predict(x)
print(em_predictions)
print(km_predictions)
# ### Step 7 : Visualize the Clusters from both EM and K-Means
import matplotlib.pyplot as plt
plt.scatter(x[:, 1], x[:, 2], c=em_predictions)
plt.scatter(x[:, 1], x[:, 2], c=km_predictions)
```

**OUTPUT:**

```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: D:\ML\RNSIT_ML_LAB_PROGRAMS_JULY2018\RNSIT_ML_LAB_PROGRAMS_JULY2018\8EM.py
[['Male' 19 15]
 ['Male' 21 15]
 ['Female' 20 16]
 ['Female' 23 16]
 ['Female' 31 17]]
[[1 19 15]
 [1 21 15]
 [0 20 16]
 [0 23 16]
 [0 31 17]]
[4 4 0 0 0 0 0 0 4 0 4 0 2 0 4 4 0 4 4 0 4 4 2 4 2 4 2 4 2 1 4 1 4 4 2 1 2
 1 2 1 2 4 4 1 2 1 2 1 1 1 2 4 1 4 2 4 2 4 1 4 4 4 2 2 4 4 2 2 4 1 4 2 2 2
 4 4 2 4 3 2 4 4 4 2 1 4 2 3 1 2 2 4 4 2 1 4 2 1 4 4 1 2 4 4 4 3 2 4 4 4 4
 3 2 4 3 3 2 2 2 2 4 2 2 4 1 1 4 4 4 4 4 4 1 1 4 1 2 4 4 1 2 4 1 1 4 4 4 1
 1 4 4 4 2 1 2 1 4 1 4 1 2 1 4 1 4 1 4 1 1 4 4 4 4 4 2 1 4 4 4 4 1 1 4 1 1
 4 2 4 1 1 1 1 4 1 1 1 1 4 4 4]
[0 0 0 0 0 0 0 0 2 0 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 2 0 2 0 2 0 0
 0 0 0 2 0 2 0 2 0 2 0 0 0 2 0 0 2 2 2 2 2 4 2 2 4 2 2 2 4 2 2 4 4 2 2 2 2
 2 4 2 4 4 2 2 4 2 2 4 2 2 4 4 2 2 4 2 4 4 4 2 4 2 4 4 2 2 4 2 4 2 2 2 2 2
 4 1 4 4 4 2 1 1 1 4 1 1 1 4 1 1 1 2 1 1 1 4 1 4 1 1 1 4 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
>>> |
```

**CHOICE 2- PROGRAM 8A (EM Algorithm):**

```
import csv
import math
import copy
k=2

class cluster:
    def __init__(self,cluster_head_data):
        self.head=cluster_head_data
        self.l=[]
        self.mean=cluster_head_data[0]
        self.variance=1.0
    def display_head(self):
        print("m=",self.mean)
        print("v=",self.variance)
    def add_ele_cluster(self,data):
        self.l.append(data)
        print(self.l)
    def display_ele(self):
        print('list contains',self.l)

def find_insert_individual_cluster(cluster,element,n):
    ele=float(element[0])
    prob=[]

    for i in range(len(cluster)):
        pxb= 1/math.sqrt(2*3.142*cluster[i].variance)* math.exp(-1* ( ele - float(cluster[i].mean) )**2 /
        (2*float(cluster[i].variance)))
        print('pxb exact==',pxb)
        prob.append(pxb)
    print('prob elem',prob)
    bi_den=0
    for i in range(len(prob)):
        bi_den=bi_den+prob[i]*1/n
        print('bi den', bi_den)
    #insert ele in to the cluster-- ele+pxb+bi
    for i in range(len(cluster)):
        clust_data=[]
        clust_data.append(ele)
        bi=(prob[i]*1/n)/bi_den
        clust_data.append(bi)
        #add the contents on to the cluster
        cluster[i].add_ele_cluster(clust_data)

def recalculate_cluster_mean_variance(cluster):
    ll=cluster.l
```

```

print('list enteries',l1)
#recalculating mean
mean_num=0.0
mean_den=0.0
var_num=0.0
var_den=0.0
for i in range(len(l1)):
    mean_num=mean_num+l1[i][0]*l1[i][1]
    mean_den=mean_den+l1[i][1]
mean=mean_num/mean_den
cluster.mean=mean
#recalculating varaiaance
for i in range(len(l1)):
    var_num=var_num+l1[i][1]*(l1[i][0]-mean)**2
    var_den=var_den+l1[i][1]
variance=var_num/var_den
cluster.variance=mean

def find_nearest(cluster,ele):
    ele=float(ele[0])
    prob=[]
    nearest_prob=None
    index=1
    for i in range(len(cluster)):
        pxb= 1/math.sqrt(2*3.142*cluster[i].variance)* math.exp(-1* ( ele - float(cluster[i].mean) )**2 /
        (2*float(cluster[i].variance)))
        print('pxb for cluster i',i,'=',pxb)
        if nearest_prob is None:
            nearest_prob=pxb
            index=i
        else:
            if nearest_prob < pxb:
                nearest_prob=pxb
                index=i
    print('index',index,'nearest_prob=',nearest_prob)
    cluster[index].l.append(ele)

#read the contents of CSV file
with open('cluster.csv') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',')
    #inserting ellements in to the list
    db=[]
    for row in spamreader:
        db.append(row)
    #creating individual cluster heads
#displaying ellements of the list
print('Db entries')
print(db)

```

```
#initialize the cluster
c=[]

#initialize the cluster heads
for i in range(k):
    new_clust=cluster(db[i])
    c.append(new_clust)
print('initial cluster Mean Variance')
#print cluster mean and variance
for i in range(k):
    print('----cluster',i,'-----')
    c[i].display_head()

error_ratio=1
# Iteration and including elements in the cluster
while error_ratio>0:
    error_ratio=0
    prevc=copy.deepcopy(c)
    #estimation
    for i in range(len(db)):
        find_insert_individual_cluster(c,db[i],len(db))
    #recalculate cluster mean and variance
    for i in range(len(c)):
        recalculate_cluster_mean_variance(c[i])
    #display recalculated mean and variance of cluster
    for i in range(k):
        print('----cluster',i,'-----')
        c[i].display_head()
    #clear all the values of the cluster list for the next iteration
    for i in range(k):
        c[i].l=[]
    #calculate the error
    error_ratio=0
    for i in range(len(c)):
        if abs(c[i].variance - prevc[i].variance)>0.1:
            error_ratio=error_ratio+1
    #display the cluster elements
    #clear all the values of the cluster list for the next iteration
    for i in range(k):
        c[i].l=[]
    #calculate the nearest prob for each element and include it in the resultant cluster
    for i in range(len(db)):
        find_nearest(c,db[i])

    #display cluster elements
    for i in range(len(c)):
        print(c[i].l)
```

**Training Data:**

1	1
3	3
5	5
8	8
2	2
11	11
14	14
18	18

**OUTPUT:**

```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
[[1.0, 0.0000000000000000], [3.0, 0.0000000000000000], [5.0, 0.0000000000000000], [8.0, 0.0000000000000000], [2.0, 0.0000000000000000], [11.0, 0.0000000000000000], [14.0, 0.0000000000000000], [18.0, 0.0000000000000000]]
list entries [[1.0, 0.9999999999999999], [3.0, 0.9872346760913435], [5.0, 0.8845366857156121], [8.0, 0.026608765578653434], [2.0, 0.9937611153166809], [11.0, 0.7237162031421714e-06], [14.0, 1.4215896682110703e-10], [18.0, 1.317723888723432e-18]]
list entries [[1.0, 0.004053118004248754], [3.0, 0.012765323908656461], [5.0, 0.11546331428438791], [8.0, 0.9733912344213466], [2.0, 0.006238884683319154], [11.0, 0.9999927628379686], [14.0, 0.999999999857841], [18.0, 1.0]]
---cluster 0 ---
m= 2.7213340517024402
v= 2.7213340517024402
---cluster 1 ---
m= 12.504957731742246
v= 12.504957731742246
pxb for cluster i 0 = 0.14030103544739295
pxb for cluster i 1 = 0.0005673529507970876
index 0 nearest_prob= 0.14030103544739295
pxb for cluster i 0 = 0.23839358908896524
pxb for cluster i 1 = 0.003044550243209151
index 0 nearest_prob= 0.23839358908896524
pxb for cluster i 0 = 0.09314788538736465
pxb for cluster i 1 = 0.01186516649339238
index 0 nearest_prob= 0.09314788538736465
pxb for cluster i 0 = 0.0014456877959066348
pxb for cluster i 1 = 0.05011024116755676
index 1 nearest_prob= 0.05011024116755676
pxb for cluster i 0 = 0.2197718847381247
pxb for cluster i 1 = 0.0013678961391086488
index 0 nearest_prob= 0.2197718847381247
pxb for cluster i 0 = 8.216307351891605e-07
pxb for cluster i 1 = 0.10304124381579008
index 1 nearest_prob= 0.10304124381579008
pxb for cluster i 0 = 1.709934543851912e-11
pxb for cluster i 1 = 0.10316387232255203
index 1 nearest_prob= 0.10316387232255203
pxb for cluster i 0 = 5.707857566318809e-20
pxb for cluster i 1 = 0.033728682873350235
index 1 nearest_prob= 0.033728682873350235
[1.0, 3.0, 5.0, 2.0]
[8.0, 11.0, 14.0, 18.0]
```



**PROGRAM 8B (K-Means Algorithm) :**

```
import csv
import math
import copy
k=2

class cluster:
    def __init__(self,cluster_head_data):
        self.head=cluster_head_data
        self.l=[]
    def display_head(self):
        print(self.head)
    def add_ele_cluster(self,data):
        self.l.append(data)
    def display_ele(self):
        print('list contains',self.l)

def compare_the_values(first,secound):
    x=float(first[0])
    y=float(first[1])
    x1=float(secound[0])
    x2=float(secound[1])
    val=math.sqrt( math.pow(math.fabs(x-x1),2)+math.pow(math.fabs(x-x1),2) )
    return val

def compare_the_nearest_cluster(cluster,data):
    #intialize the nearest cluster to 0
    dist_measure=None
    nearest=0
    for i in range(len(cluster)):
        dist=compare_the_values(cluster[i].head,data)
        if dist_measure is None:
            dist_measure=dist
            nearest=i
        if dist<dist_measure:
            dist_measure=dist
            nearest=i
    return nearest

def recal_head(cluster):
    for i in range(len(cluster)):
        l1=cluster[i].l
        xval=0.0
```

```
yval=0.0
for j in l1:
    xval=xval+float(j[0])
    yval=yval+float(j[1])
xavg=xval/len(l1)
yavg=yval/len(l1)
avgl=[]
avgl.append(xavg)
avgl.append(yavg)
cluster[i].head=avgl
```

```
#read the contents of CSV file
with open('cluster.csv') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',')
    #inserting elements in to the list
    db=[]
    for row in spamreader:
        db.append(row)
    #creating individual cluster heads
#displaying elements of the list
print('Db entries')
print(db)
```

```
#initialize the cluster
c=[]
```

```
#initialize the cluster heads
for i in range(k):
    new_clust=cluster(db[i])
    c.append(new_clust)
print('initial cluster head values')
#print display heads
for i in range(k):
    print('---cluster',i,'-----')
    c[i].display_head()
```

```
error_ratio=1
# Iteration and including elements in the cluster
while error_ratio>0:
    prevc=copy.deepcopy(c)
    for ele in db:
        r=compare_the_nearest_cluster(c,ele)
        c[r].add_ele_cluster(ele)

    # display all the elements
```

```
for clust in c:
    clust.display_ele()
#recalculate the avg value
    recal_head(c)

for i in range(k):
    print('---cluster',i,'-----')
    c[i].display_head()
#remove the ele of cluter head for the next iter
for i in range(k):
    c[i].l=[]
#calculate the error
    error_ratio=0
for i in range(k):
    if c[i].head != prevc[i].head:
        error_ratio=error_ratio+1
#final cluster ele
```

**Training Data:**

1	1
3	3
5	5
8	8
2	2
11	11
14	14
18	18

**OUTPUT:**

```

Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD 64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\ML\ML Lab Programs\8-k means\k means.py =====
Db entries
[['1', '1'], ['3', '3'], ['5', '5'], ['8', '8'], ['2', '2'], ['11', '11'], ['14', '14'], ['18', '18']]
initial cluster head values
---cluster 0 -----
['1', '1']
---cluster 1 -----
['3', '3']
list contains [['1', '1'], ['2', '2']]
list contains [['3', '3'], ['5', '5'], ['8', '8'], ['11', '11'], ['14', '14'], ['18', '18']]
---cluster 0 -----
[1.5, 1.5]
---cluster 1 -----
[9.833333333333334, 9.833333333333334]
list contains [['1', '1'], ['3', '3'], ['5', '5'], ['2', '2']]
list contains [['8', '8'], ['11', '11'], ['14', '14'], ['18', '18']]
---cluster 0 -----
[2.75, 2.75]
---cluster 1 -----
[12.75, 12.75]
list contains [['1', '1'], ['3', '3'], ['5', '5'], ['2', '2']]
list contains [['8', '8'], ['11', '11'], ['14', '14'], ['18', '18']]
---cluster 0 -----
[2.75, 2.75]
---cluster 1 -----
[12.75, 12.75]
>>>

```

**Program 9.** Write a program to implement K-nearest neighbor algorithm to classify iris dataset. Print both correct and wrong predication using python machine learning.

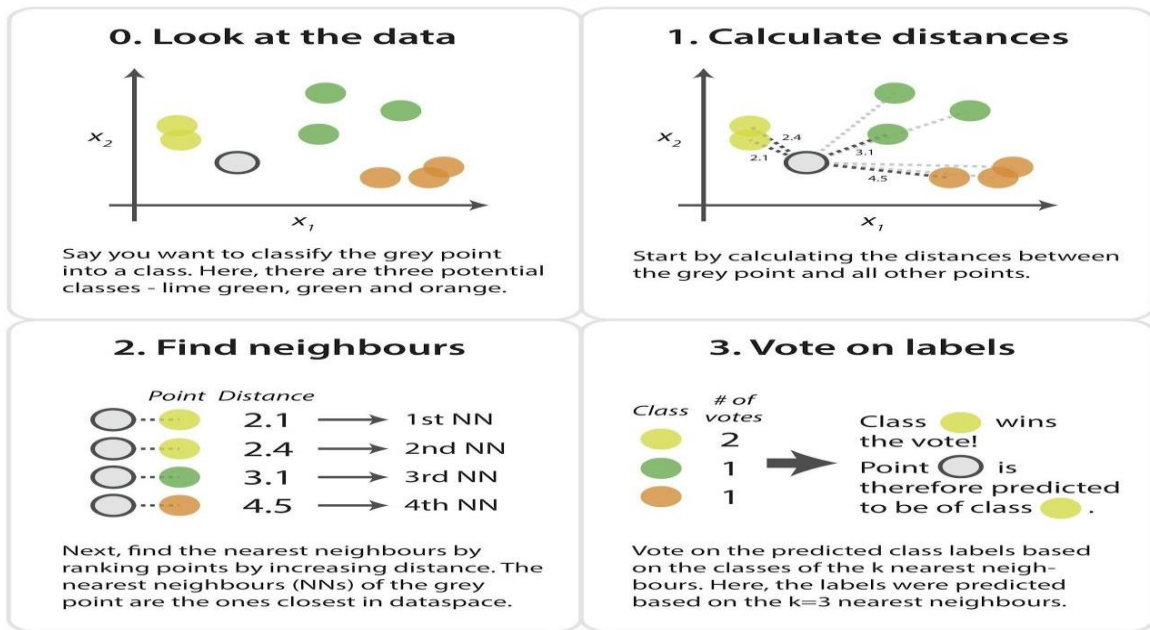
K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.<sup>[1]</sup> In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression.

K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

The kNN task can be broken down into writing 3 primary functions:

1. Calculate the distance between any two points
2. Find the nearest neighbours based on these pair wise distances
3. Majority vote on a class labels based on the nearest neighbour list

## kNN Algorithm



### Dataset

Iris dataset, consists of flower measurements for three species of iris flower. Our task is to predict the species labels of a set of flowers based on their flower measurements. Since you'll be building a predictor based on a set of known correct classifications

The data set contains 3 classes of 151 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

**Predicted attribute:** class of iris plant

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Class:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

**Algorithm**

1. Load the data and split into train and test sets.
2. Need a measure of similarity.
3. Compute the distance function  $d(a,b)$ , where  $a,b$  are the scenarios composed of  $N$  features, such that  $a=\{a_1,\dots,a_n\}$ ,  $b=\{b_1,\dots,b_n\}$ .
4. Euclidean distance measuring: where distance  $d$  between two points  $(a_1, a_2)$  and  $(b_1, b_2)$  is given by  $d = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2}$ . Each flower in the iris dataset has 4 dimensions, need to find the distance between each flower.  
 $D=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+(a_3-b_3)^2+(a_4-b_4)^2}$ .
5. The zip function aggregates elements from lists to return a list of tuples. List comprehensions are a powerful Pythonic construct that facilitate quick computations on lists.
6. Iterating over values from the corresponding dimensions in the two data points, calculating the differences squared, and storing each dimension's. These are then summed and returned.
7. This pair wise calculation is done for every train instance and the given test instance.
8. Next, the distances are sorted in order to find the  $k$  closest neighbours to the test instance.

**PROGRAM:**

```
import csv
import random
import math
import operator
def loadDataset(filename, split, trainingSet=[], testSet=[]):
    with open(filename) as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x][y] = float(dataset[x][y])
            if random.random() < split:
                trainingSet.append(dataset[x])
            else:
                testSet.append(dataset[x])
def euclideanDistance(instance1, instance2, length):
    distance = 0
```

```
    for x in range(length):
        distance += pow((instance1[x] - instance2[x]), 2)
    return math.sqrt(distance)

def getNeighbors(trainingSet, testInstance, k):
    distances = []
    length = len(testInstance)-1
    for x in range(len(trainingSet)):
        dist = euclideanDistance(testInstance, trainingSet[x], length)
        distances.append((trainingSet[x], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors

def getResponse(neighbors):
    classVotes = {}
    for x in range(len(neighbors)):
        response = neighbors[x][-1]
        if response in classVotes:
            classVotes[response] += 1
        else:
            classVotes[response] = 1
    sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)
    return sortedVotes[0][0]

def getAccuracy(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return (correct/float(len(testSet))) * 100.0

def main():
    # prepare data
    trainingSet=[]
    testSet=[]
    split = 0.67
    loadDataset('iris_data.csv', split, trainingSet, testSet)
    print ('\n Number of Training data: ' + (repr(len(trainingSet))))
    print (' Number of Test Data: ' + (repr(len(testSet))))
    # generate predictions
    predictions=[]
    k = 3
    print("\n The predictions are: ")
    for x in range(len(testSet)):
        neighbors = getNeighbors(trainingSet, testSet[x], k)
```

```

        result = getResponse(neighbors)
        predictions.append(result)
    print(' predicted=' + repr(result) + ', a  getAccuracy(testSet, predictions)
print('\n The Accuracy is: ' + repr(accuracy) + '%')

```

```
main()
```

**OUTPUT:**

 Python 3.7.0b1 Shell

```
File Edit Shell Debug Options Window Help
Python 3.7.0b1 (v3.7.0b1:9561d7f, Jan 31 2018, 07:26:34) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
```

```
>>>
===== RESTART: D:\ML\ML Lab Programs\9-knn\KNN.py =====
```

```
Number of Training data: 91
Number of Test Data: 58
```

The predictions are:

[illegible]

```
The Accuracy is: 98.27586206896551%
>>>
```



**Program 10: Implement the non-parametric Locally Weighted Regression algorithm to fit data points. Select appropriate data set for your experiment and draw graphs.**

**Locally Weighted Regression –**

- Nonparametric regression is a category of regression analysis in which the predictor does not take a predetermined form but is constructed according to information derived from the data(training examples).
- Nonparametric regression requires larger sample sizes than regression based on parametric models. Because larger the data available ,accuracy will be high.

**Locally Weighted Linear Regression –**

- ☐ Locally weighted regression is called local because the function is approximated based a only on data near the query point, weighted because the contribution of each training example is weighted by its distance from the query point.

- Query point is nothing but the point nearer to the target function , which will help in finding the actual position of the target function.

Let us consider the case of locally weighted regression in which the target function  $f$  is approximated near  $x$ , using a linear function of the form

1. Minimize the squared error over just the  $k$  nearest neighbors:

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2$$

2. Minimize the squared error over the entire set  $D$  of training examples, while weighting the error of each training example by some decreasing function  $K$  of its distance from  $x_q$ :

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

3. Combine 1 and 2:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

### **PROGRAM:**

```
from math import ceil
import numpy as np
from scipy import linalg
```

```
def lowess(x, y, f=2./3., iter=3):
    n = len(x)
    r = int(ceil(f*n))
    h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]
    w = np.clip(np.abs((x[:,None] - x[None,:]) / h), 0.0, 1.0)
    w = (1 - w**3)**3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iter):
        for i in range(n):
            weights = delta * w[:,i]
            b = np.array([np.sum(weights*y), np.sum(weights*y*x)])
            A = np.array([[np.sum(weights), np.sum(weights*x)],
                          [np.sum(weights*x), np.sum(weights*x*x)]])
```

```
    beta = linalg.solve(A, b)
    yest[i] = beta[0] + beta[1]*x[i]

    residuals = y - yest
    s = np.median(np.abs(residuals))
    delta = np.clip(residuals / (6.0 * s), -1, 1)
    delta = (1 - delta**2)**2

    return yest

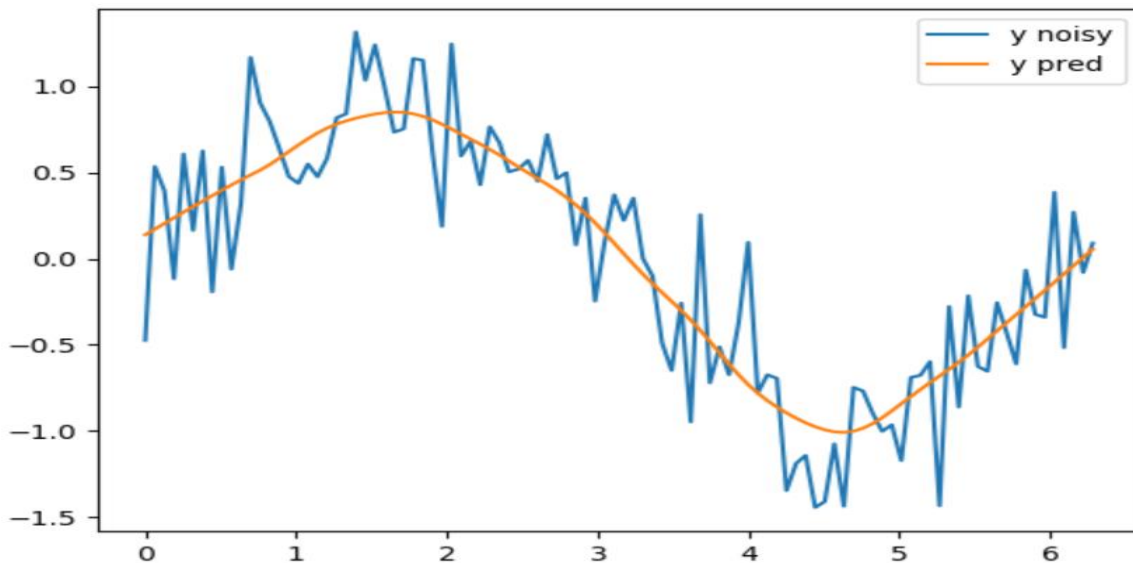
if __name__ == '__main__':
    import math
    n = 100
    x = np.linspace(0, 2 * math.pi, n)
    print("=====values of x=====")
    print(x)
    y = np.sin(x) + 0.3*np.random.randn(n)
    print("=====Values of          y=====")
    print(y)
    f = 0.25
    yest = lowess(x, y, f=f, iter=3)

    import pylab as pl
    pl.clf()
    pl.plot(x, y, label='y noisy')
    pl.plot(x, yest, label='y pred')
    pl.legend()
    pl.show()
```

**OUTPUT:**

```
Python 3.7.0b1 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: D:\ML\ML Lab Programs\l0-regression\regression.py =====
=====values of x=====
[0. 0.06346652 0.12693304 0.19039955 0.25386607 0.31733259
0.38079911 0.44426563 0.50773215 0.57119866 0.63466518 0.6981317
0.76159822 0.82506474 0.88853126 0.95199777 1.01546429 1.07893081
1.14239733 1.20586385 1.26933037 1.33279688 1.3962634 1.45972992
1.52319644 1.58666296 1.65012947 1.71359599 1.77706251 1.84052903
1.90399555 1.96746207 2.03092858 2.0943951 2.15786162 2.22132814
2.28479466 2.34826118 2.41172769 2.47519421 2.53866073 2.60212725
2.66559377 2.72906028 2.7925268 2.85599332 2.91945984 2.98292636
3.04639288 3.10985939 3.17332591 3.23679243 3.30025895 3.36372547
3.42719199 3.4906585 3.55412502 3.61759154 3.68105806 3.74452458
3.8079911 3.87145761 3.93492413 3.99839065 4.06185717 4.12532369
4.1887902 4.25225672 4.31572324 4.37918976 4.44265628 4.5061228
4.56958931 4.63305583 4.69652235 4.75998887 4.82345539 4.88692191
4.95038842 5.01385494 5.07732146 5.14078798 5.2042545 5.26772102
5.33118753 5.39465405 5.45812057 5.52158709 5.58505361 5.64852012
5.71198664 5.77545316 5.83891968 5.9023862 5.96585272 6.02931923
6.09278575 6.15625227 6.21971879 6.28318531]
=====Values of y=====
[ 6.88106030e-04 -1.53527916e-01 -4.98737481e-01 1.66052067e-02
 4.86586342e-01 5.59773603e-01 1.24770523e+00 -2.69282742e-01
 3.83486911e-01 4.78805302e-01 7.18032090e-01 1.06634463e-01
 8.42933418e-01 1.04366178e+00 9.30413454e-01 1.01345445e+00
 9.82183103e-01 9.45759176e-01 4.75508863e-01 1.46202254e+00
 9.63727398e-01 7.29364864e-01 1.08280866e+00 1.34037389e+00
 7.84259178e-01 7.66175421e-01 1.31624987e+00 1.47139621e+00
 8.04798599e-01 8.12591280e-01 1.22360577e+00 1.15719055e+00
 6.84623550e-01 7.69221924e-01 2.78289358e-01 5.81659677e-01
 1.37037559e+00 9.40211539e-01 5.31299720e-01 6.26264032e-01
 7.49614838e-01 8.13147491e-01 5.46453233e-01 4.81099301e-01
 6.22229293e-01 3.21262080e-02 2.53146620e-01 -3.44801420e-02
 4.89829335e-01 8.75060677e-02 -6.03342638e-01 -2.17350719e-01
 2.32082134e-01 -4.38989269e-01 3.99743480e-02 -1.89615065e-01
 -1.91664335e-01 -6.42399360e-01 -3.72766160e-01 -2.82936950e-01
 -6.77408578e-01 -9.65816954e-01 -7.16975131e-01 -4.65038775e-01
 -2.35525338e-01 -7.51120676e-01 -5.92095239e-01 -4.98764638e-01
 -9.37475246e-01 -7.01832144e-01 -8.12613956e-01 -1.06577126e+00]
```

Figure 1



## VIVA QUESTION & ANSWERS

Q1- What's the trade-off between bias and variance?

Bias is error due to erroneous or overly simplistic assumptions in the learning algorithm you're using. This can lead to the model underfitting your data, making it hard for it to have high predictive accuracy and for you to generalize your knowledge from the training set to the test set.

Variance is error due to too much complexity in the learning algorithm you're using. This leads to the algorithm being highly sensitive to high degrees of variation in your training data, which can lead your model to overfit the data. You'll be carrying too much noise from your training data for your model to be very useful for your test data.

The bias-variance decomposition essentially decomposes the learning error from any algorithm by adding the bias, the variance and a bit of irreducible error due to noise in the underlying dataset. Essentially, if you make the model more complex and add more variables, you'll lose bias but gain some variance — in order to get the optimally reduced amount of error, you'll have to tradeoff bias and variance. You don't want either high bias or high variance in your model.

### **Q2- What is the difference between supervised and unsupervised machine learning?**

Supervised learning requires training labeled data. For example, in order to do classification (a supervised learning task), you'll need to first label the data you'll use to train the model to classify data into your labeled groups. Unsupervised learning, in contrast, does not require labeling data explicitly.

### **Q3- How is KNN different from k-means clustering?**

K-Nearest Neighbors is a supervised classification algorithm, while k-means clustering is an unsupervised clustering algorithm. While the mechanisms may seem similar at first, what this really means is that in order for K-Nearest Neighbors to work, you need labeled data you want to classify an unlabeled point into (thus the nearest neighbor part). K-means clustering requires only a set of unlabeled points and a threshold: the algorithm will take unlabeled points and gradually learn how to cluster them into groups by computing the mean of the distance between different points.

The critical difference here is that KNN needs labeled points and is thus supervised learning, while k-means doesn't — and is thus unsupervised learning.

### **Q4- Define precision and recall.**

Recall is also known as the true positive rate: the amount of positives your model claims compared to the actual number of positives there are throughout the data. Precision is also known as the positive predictive value, and it is a measure of the amount of accurate positives your model claims compared to the number of positives it actually claims. It can be easier to think of recall and precision in the context of a case where you've predicted that there were 10 apples and 5 oranges in a case of 10 apples. You'd have perfect recall

(there are actually 10 apples, and you predicted there would be 10) but 66.7% precision because out of the 15 events you predicted, only 10 (the apples) are correct.

**Q5 What is Bayes' Theorem? How is it useful in a machine learning context?**

Bayes' Theorem gives you the posterior probability of an event given what is known as prior knowledge. Mathematically, it's expressed as the true positive rate of a condition sample divided by the sum of the false positive rate of the population and the true positive rate of a condition. Say you had a 60% chance of actually having the flu after a flu test, but out of people who had the flu, the test will be false 50% of the time, and the overall population only has a 5% chance of having the flu. Would you actually have a 60% chance of having the flu after having a positive test?

Bayes' Theorem says no. It says that you have a  $(.6 * 0.05)$  (True Positive Rate of a Condition Sample) /  $(.6*0.05)(\text{True Positive Rate of a Condition Sample}) + (.5*0.95)$  (False Positive Rate of a Population) = 0.0594 or 5.94% chance of getting a flu.

Bayes' Theorem is the basis behind a branch of machine learning that most notably includes the Naive Bayes classifier. That's something important to consider when you're faced with machine learning interview questions.

**Q6- Why is "Naive" Bayes naive?**

Despite its practical applications, especially in text mining, Naive Bayes is considered "Naive" because it makes an assumption that is virtually impossible to see in real-life data: the conditional probability is calculated as the pure product of the individual probabilities of components. This implies the absolute independence of features — a condition probably never met in real life.

As a Quora commenter put it whimsically, a Naive Bayes classifier that figured out that you liked pickles and ice cream would probably naively recommend you a pickle ice cream.

**Q7- What's your favorite algorithm, and can you explain it to me in less than a minute?**

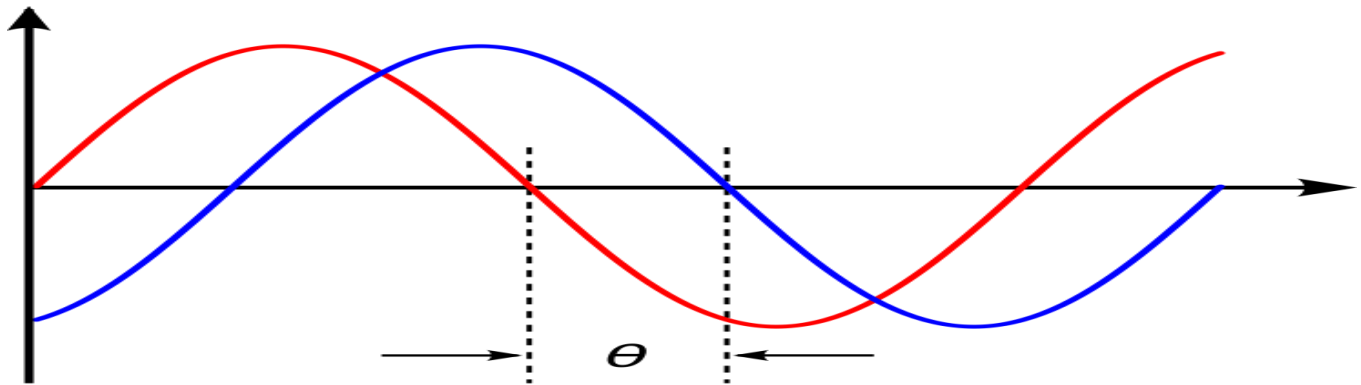
This type of question tests your understanding of how to communicate complex and technical nuances with poise and the ability to summarize quickly and efficiently. Make sure you have a choice and make sure you can explain different algorithms so simply and effectively that a five-year-old could grasp the basics!

**Q8- What's the difference between Type I and Type II error?**

Don't think that this is a trick question! Many machine learning interview questions will be an attempt to lob basic questions at you just to make sure you're on top of your game and you've prepared all of your bases. Type I error is a false positive, while Type II error is a false negative. Briefly stated, Type I error means claiming something has happened when it hasn't, while Type II error means that you claim nothing is happening when in fact something is.

A clever way to think about this is to think of Type I error as telling a man he is pregnant, while Type II error means you tell a pregnant woman she isn't carrying a baby.

**Q9- What's the difference between probability and likelihood?**



**Q10- What is deep learning, and how does it contrast with other machine learning algorithms?**

Deep learning is a subset of machine learning that is concerned with neural networks: how to use backpropagation and certain principles from neuroscience to more accurately model large sets of unlabelled or semi-structured data. In that sense, deep learning represents an unsupervised learning algorithm that learns representations of data through the use of neural nets.

**Q11- How is a decision tree pruned?**

Pruning is what happens in decision trees when branches that have weak predictive power are removed in order to reduce the complexity of the model and increase the predictive accuracy of a decision tree model. Pruning can happen bottom-up and top-down, with approaches such as reduced error pruning and cost complexity pruning.

Reduced error pruning is perhaps the simplest version: replace each node. If it doesn't decrease predictive accuracy, keep it pruned. While simple, this heuristic actually comes pretty close to an approach that would optimize for maximum accuracy.

**Q12- Which is more important to you– model accuracy, or model performance?**

This question tests your grasp of the nuances of machine learning model performance! Machine learning interview questions often look towards the details. There are models with higher accuracy that can perform worse in predictive power — how does that make sense?

Well, it has everything to do with how model accuracy is only a subset of model performance, and at that, a sometimes misleading one. For example, if you wanted to detect fraud in a massive dataset with a sample of millions, a more accurate model would most likely predict no fraud at all if only a vast minority of cases were fraud. However, this would be useless for a predictive model — a model designed to find fraud that

asserted there was no fraud at all! Questions like this help you demonstrate that you understand model accuracy isn't the be-all and end-all of model performance.

### **Q13- What's the F1 score? How would you use it?**

The F1 score is a measure of a model's performance. It is a weighted average of the precision and recall of a model, with results tending to 1 being the best, and those tending to 0 being the worst. You would use it in classification tests where true negatives don't matter much.

### **Q14- How would you handle an imbalanced dataset?**

An imbalanced dataset is when you have, for example, a classification test and 90% of the data is in one class. That leads to problems: an accuracy of 90% can be skewed if you have no predictive power on the other category of data! Here are a few tactics to get over the hump:

- 1- Collect more data to even the imbalances in the dataset.
- 2- Resample the dataset to correct for imbalances.
- 3- Try a different algorithm altogether on your dataset.

What's important here is that you have a keen sense for what damage an unbalanced dataset can cause, and how to balance that.

### **Q15- When should you use classification over regression?**

Classification produces discrete values and dataset to strict categories, while regression gives you continuous results that allow you to better distinguish differences between individual points. You would use classification over regression if you wanted your results to reflect the belongingness of data points in your dataset to certain explicit categories (ex: If you wanted to know whether a name was male or female rather than just how correlated they were with male and female names.)

### **Q16- How do you handle missing or corrupted data in a dataset?**

You could find missing/corrupted data in a dataset and either drop those rows or columns, or decide to replace them with another value.

In Pandas, there are two very useful methods: `isnull()` and `dropna()` that will help you find columns of data with missing or corrupted data and drop those values. If you want to fill the invalid values with a placeholder value (for example, 0), you could use the `fillna()` method.