Maharaja Education Trust (R), Mysuru

Maharaja Institute of Technology Mysore

Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Approved by AICTE, New Delhi,
Affiliated to VTU, Belagavi & Recognized by Government of Karnataka

Laboratory Manual on

Machine Learning Laboratory (15CS76)

Prepared by

Department of Information Science & Engineering

Maharaja Education Trust (R), Mysuru
Maharaja Institute of Technology Mysore
Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Vision/ ಆಶಯ

"To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude"

ಸಂಶೋಧನೆ, ಆವಿಷ್ಕಾರ ಹಾಗೂ ಉದ್ಯಮಶೀಲತೆಯನ್ನು ಉತ್ತೇಜಿಸುವ ಅಗ್ರಮಾನ್ಯ ತಾಂತ್ರಿಕ ಮತ್ತು ಆಡಳಿತ ವಿಜ್ಞಾನ ಶಿಕ್ಷಣ ಕೇಂದ್ರವಾಗಿ ಗುರುತಿಸಿಕೊಳ್ಳುವುದು.

Mission/ ಧ್ಯೇಯ

- To empower students with indispensable knowledge through dedicated teaching and collaborative learning.
  ಸಮರ್ಪಣಾ ಮನೋಭಾವದ ಬೋಧನೆ ಹಾಗೂ ಸಹಭಾಗಿತ್ವದ ಕಲಿಕಾಕ್ರಮಗಳಿಂದ ವಿದ್ಯಾರ್ಥಿಗಳನ್ನು ಅತ್ಯತ್ಕೃಷ್ಟ ಜ್ಞಾನಸಂಪನ್ನರಾಗಿಸುವುದು.
- To advance extensive research in science, engineering and management disciplines.
  ವೈಜ್ಞಾನಿಕ, ತಾಂತ್ರಿಕ ಹಾಗೂ ಆಡಳಿತ ವಿಜ್ಞಾನ ವಿಭಾಗಗಳಲ್ಲಿ ವಿಸ್ತೃತ ಸಂಶೋಧನೆಗಳೊಡನೆ ಬೆಳವಣಿಗೆ ಹೊಂದುವುದು.
- To facilitate entrepreneurial skills through effective institute - industry collaboration and interaction with alumni.
  ಉದ್ಯಮ ಕ್ಷೇತಗಳೊಡನೆ ಸಹಯೋಗ, ಸಂಸ್ಥೆಯ ಹಿರಿಯ ವಿದ್ಯಾರ್ಥಿಗಳೊಂದಿಗೆ ನಿರಂತರ ಸಂವಹನಗಳಿಂದ ವಿದ್ಯಾರ್ಥಿಗಳಿಗೆ ಉದ್ಯಮಶೀಲತೆಯ ಕೌಶಲ್ಯ ಪಡೆಯಲು ನೆರವಾಗುವುದು.
- To instill the need to uphold ethics in every aspect.
  ಜೀವನದಲ್ಲಿ ನೈತಿಕ ಮೌಲ್ಯಗಳನ್ನು ಅಳವಡಿಸಿಕೊಳ್ಳುವುದರ ಮಹತ್ವದ ಕುರಿತು ಅರಿವು ಮೂಡಿಸುವುದು.
- To mould holistic individuals capable of contributing to the advancement of the society.
  ಸಮಾಜದ ಬೆಳವಣಿಗೆಗೆ ಗಣನೀಯ ಕೊಡುಗೆ ನೀಡಬಲ್ಲ ಪರಿಪೂರ್ಣ ವ್ಯಕ್ತಿತ್ವವುಳ್ಳ ಸಮರ್ಥ ನಾಗರೀಕರನ್ನು ರೂಪಿಸುವುದು.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Vision / ಆಶಯ

To be recognized as the best centre for technical education and research in the field of information science and engineering.

Mission / ಧ್ಯೇಯ

- To facilitate adequate transformation in students through a proficient teaching learning process with the guidance of mentors and all-inclusive professional activities.
- To infuse students with professional, ethical and leadership attributes through industry collaboration and alumni affiliation.
- To enhance research and entrepreneurship in associated domains and to facilitate real time problem solving.

.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Program Outcomes

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.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Course Overview

Subject: Machine Learning Laboratory **Subject Code: 15CS76**

Machine learning is a branch of artificial intelligence, a science that researches machines to acquire new knowledge and new skills and to identify existing knowledge. Machine learning has been widely used in data mining, computer vision, natural language processing, biometrics, search engines, medical diagnostic etc.

Machine learning laboratory aims at practicing and achieving this aim by using various machine learning algorithms, these require a through practice of python programming language.

Machine Learning Laboratory aims at understating and implementing machine learning algorithms on data sets Using java or python as programing languages.

Course Objectives

The objectives of this course is to make students to learn-

- Make use of Data sets in implementing the machine learning algorithms.
- Implement the machine learning concepts and algorithms in any suitable language of choice.

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Course Outcomes

CO's	DESCRIPTION OF THE OUTCOMES
15CSL76.1	Apply suitable algorithms to solve real world problem and present the same.
15CSL76.2	Analyze the usage of appropriate computing and mathematical models to write machine learning algorithm..
15CSL76.3	Examine different working processes of machine learning concepts to meet desired need and present them appropriately.
15CSL76.4	Evaluate the machine learning algorithm using python tools for a given data set.

Dr.Y.H.SHARATH KUMAR	DR. PUSHPA D	DR. PUSHPA D
Faculty		**Course Coordinator**

Criterion 3 Coordinator	**NBA coordinator**	**HOD**
Convener		**Principal**

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Syllabus

Subject: Machine Learning Laboratory **Subject Code: 15CS76**

Topics Covered as per Syllabus
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. 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. 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. 4. Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets. 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. 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. 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. 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. You can add Java/Python ML library classes/API in the program. 9. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem. 10. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs. **Conduction of Practical Examination:** • All laboratory experiments are to be included for practical examination. • Students are allowed to pick one experiment from the lot. • Strictly follow the instructions as printed on the cover page of answer script. • Marks distribution: Procedure + Conduction + Viva: 20 + 50 +10 (80). **Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.**
List of Text Books
1. Tom M. Mitchell, Machine Learning, India Edition 2013, McGraw Hill Education.

List of Reference Books
1. Trevor Hastie, Robert Tibshirani, Jerome Friedman, h The Elements of Statistical Learning, 2nd edition, springer series in statistics. 2. Ethem Alpaydın, Introduction to machine learning, second edition, MIT press.
List of URLs, Text Books, Notes, Multimedia Content, etc
1. https://medium.com/ml-research-lab/machine-learning-algorithm-overview 2. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

Maharaja Institute of Technology Mysore

Department of Information Science & Engineering

Index

Subject: Machine Learning Laboratory **Subject Code: 15CS76**

Python Machine Learning – Introduction

Python is a popular platform used for research and development of production systems. It is a vast language with number of modules, packages and libraries that provides multiple ways of achieving a task.

Python and its libraries like NumPy, SciPy, Scikit-Learn, Matplotlib are used in data science and data analysis. They are also extensively used for creating scalable machine learning algorithms. Python implements popular machine learning techniques such as Classification, Regression, Recommendation, and Clustering.

Python offers ready-made framework for performing data mining tasks on large volumes of data effectively in lesser time. It includes several implementations achieved through algorithms such as linear regression, logistic regression, Naïve Bayes, k-means, K nearest neighbour, and Random Forest.

Python in Machine Learning Python has libraries that enables developers to use optimized algorithms. It implements popular machine learning techniques such as recommendation, classification, and clustering. Therefore, it is necessary to have a brief introduction to machine learning before we move further.

What is Machine Learning? Data science, machine learning and artificial intelligence are some of the top trending topics in the tech world today. Data mining and Bayesian analysis are trending and this is adding the demand for machine learning. This tutorial is your entry into the world of machine learning.

Machine learning is a discipline that deals with programming the systems so as to make them automatically learn and improve with experience. Here, learning implies recognizing and understanding the input data and taking informed decisions based on the supplied data. It is very difficult to consider all the decisions based on all possible inputs. To solve this problem, algorithms are developed that build knowledge from a specific data and past experience by applying the principles of statistical science, probability, logic, mathematical optimization, reinforcement learning, and control theory.

Applications of Machine Learning Algorithms The developed machine learning algorithms are used in various applications such as:

- Vision processing
- Language processing
- Forecasting things like stock market trends, weather
- Pattern recognition
- Games
- Data mining
- Expert systems
- Robotics

Steps Involved In Machine Learning A Machine Learning Programs Involves The Following Steps:

- Defining a Problem
- Preparing Data
- Evaluating Algorithms
- Improving Results
- Presenting Results

The best way to get started using Python for machine learning is to work through a project end-to-end and cover the key steps like loading data, summarizing data, evaluating algorithms and making some predictions. This gives you a replicable method that can be used dataset after dataset. You can also add further data and improve the results.

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE
BELAWADI, SRIRANGAPATNA TALUK, MANDYA-571477
DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

General Lab Guidelines:

- Conduct yourself in a responsible manner at all times in the laboratory. Intentional misconduct will lead to the exclusion from the lab.
- Do not wander around, or distract other students, or interfere with the laboratory experiments of other students.
- Read the handout and procedures before starting the experiments. Follow all written and verbal instructions carefully. If you do not understand the procedures, ask the instructor or teaching assistant.
- The workplace has to be tidy before, during and after the experiment.
- Do not eat food, drink beverages or chew gum in the laboratory.

Do's and Don'ts w.r.t Lab Practice. (Computer Lab)

- Avoid stepping on electrical wires or any other computer cables.
- Do not insert metal objects such as clips, pins and needles into the computer casings(They may cause fire) and should not attempt to repair, open, tamper or interfere with any of the computer, printing, cabling, or other equipment in the laboratory.
- Do not open any irrelevant internet sites on lab computer.
- Do not use a flash drive on lab computers without the consent of lab instructor.
- Do not upload, delete or alter any software on the lab PC.
- Students are not allowed to work in Laboratory alone or without presence of the instructor/ teaching assistant.
- Turn off the machine once you are done using it.

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 csv file.

Solution: 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

* ai - constraint attribute of hypothesis, ai can have a value of 0 (no value), or any value(ai =sunny), or ?(any value)

* c - target concept

Algorithm: 1. Initialize h to the most specific hypothesis in H

1. For each positive training instance x

• For each attribute constraint ai is satisfied by x
• Then do nothing
• Else replace ai in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

```
import csv
#!usr/bin/python #list creatin
hypo=['%','%','%','%','%','%']
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)
      if row[len(row)-1] =='Yes':
         data.append(row)
print("\nThe positive examples are:")
for x in data:
   print(x)
print("\n")
TotalExamples=len(data)
i=0
j=0
k=0
print("The steps of the Find-s algorithm are\n",hypo)
```

```
list =[]
p=0
d=len(data[p])-1
for j in range(d):
   list.append(data[i][j])
hypo=list
for i in range(1,TotalExamples):
   for k in range(d):
      if hypo[k]!=data[i][k]:
         hypo[k]='?'
      else:
         hypo[k]
   print(hypo)
   print("....................................................")
print("\nThe maximally specific Find-s hypothesis for the given training examples is");
list=[]
for i in range(d):
   list.append(hypo[i])
print(list)
```

OUTPUT

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', '?', 'Strong', 'Warm', 'Same']

['Sunny', 'Warm', '?', 'Strong', '?', '?']

The maximally specific Find-s hypothesis for the given training examples is

['Sunny', 'Warm', '?', 'Strong', '?', '?']

Program 2: For a given set of training data example stored in .csv file, implement and demonstrate the Candidate-Elimination Algorithm to output and describes the set of all hypotheses consistent with training example.

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

G0 = {(?, ?, ?, ?)}

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

S0 = {(Θ, Θ, Θ, Θ)}

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 contains all the hypotheses consistent with these training examples. The algorithm is summarized as below:

Candidate-Elimination Algorithm

Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For every learning 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 generalization 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 is S
- If d is 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

For the implementation of Candidate-Elimination Algorithm the EnjoySport data set can be used from UCI repository. The data set contains around 700 data samples. Few of the samples are showed in Table 1.

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

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('trainingexamples.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print(general_h)
    for i, h in enumerate(concepts):
        if target[i] == "Y":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
                print(specific_h)
        print(specific_h)
        if target[i] == "N":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
```

```
print("Final General_h:", g_final, sep="\n")
#data.head()
```

OUTPUT

```
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
['Y' 'Y' 'N' 'Y']
initialization of specific_h and general_h
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
steps of Candidate Elimination Algorithm 1
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

steps of Candidate Elimination Algorithm 2
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]
```

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

steps of Candidate Elimination Algorithm 3

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' '?' 'Strong' '?' 'Same']

['Sunny' 'Warm' '?' 'Strong' '?' '?']

['Sunny' 'Warm' '?' 'Strong' '?' '?']

steps of Candidate Elimination Algorithm 4

['Sunny' 'Warm' '?' 'Strong' '?' '?']

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific_h:

['Sunny' 'Warm' '?' '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:

$$Entropy = - p(a)*\log(p(a)) - p(b)*\log(p(b))$$

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

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

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 recurs on each subset, considering only attributes never selected before.

Dataset Details

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

ID3 (Learning Sets S, Attributes Sets A, Attributes values V) Return Decision Tree

Begin
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
- I f 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

Entropy = - p(a)*log(p(a)) - p(b)*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

Gain(S,A)= Entropy(S) - Sum for v from 1 to n of (|Sv|/|S|) * Entropy(Sv)

3. Make a decision tree node containing that attribute

4. Recurring on subsets using remaining attributes.

```
import ast
import csv
#import sys
import math
import os
def load_csv_to_header_data(filename):
    path = os.path.normpath(os.getcwd() + filename)
    ''' os.path.normpath(path)
Normalize a pathname by collapsing redundant separators and up-level references so that A//B,
A/B/, A/./B and A/foo/../B all become A/B. This string manipulation may change the meaning
of a path that contains symbolic links. On Windows, it converts forward slashes to backward
slashes. To normalize case, use normcase().'''
    print(path)
    fs = csv.reader(open(path))
    all_row = []
    for r in fs:
        all_row.append(r)
    headers = all_row[0]
    idx_to_name, name_to_idx = get_header_name_to_idx_maps(headers)
    data = { 'header': headers,'rows': all_row[1:],'name_to_idx': name_to_idx,'idx_to_name':
idx_to_name}
    return data
def get_header_name_to_idx_maps(headers):
    name_to_idx = {}
    idx_to_name = {}
```

```
    for i in range(0, len(headers)):
        name_to_idx[headers[i]] = i
        idx_to_name[i] = headers[i]
        #print(name_to_idx)
        #print(idx_to_name)
    return idx_to_name, name_to_idx
def project_columns(data, columns_to_project):
    data_h = list(data['header'])
    data_r = list(data['rows'])
    all_cols = list(range(0,len(data_h)))
    columns_to_project_ix = [data['name_to_idx'][name] for name in columns_to_project]
    #print(columns_to_project_ix)
    columns_to_remove = [cidx for cidx in all_cols if cidx not in columns_to_project_ix]
    #print(columns_to_remove)
    for delc in sorted(columns_to_remove, reverse=True):
        del data_h[delc]
        for r in data_r:
            del r[delc]
    idx_to_name, name_to_idx = get_header_name_to_idx_maps(data_h)
    return {'header': data_h, 'rows': data_r,'name_to_idx': name_to_idx,'idx_to_name':
idx_to_name}
def get_uniq_values(data):
    idx_to_name = data['idx_to_name']
    idxs = idx_to_name.keys()
    #print(idxs)
    val_map = {}
    for idx in iter(idxs):
        val_map[idx_to_name[idx]] = set()
    #print(val_map)
    for data_row in data['rows']:
        for idx in idx_to_name.keys():
            att_name = idx_to_name[idx]
            val = data_row[idx]
            if val not in val_map.values():
                val_map[att_name].add(val)
    #print(val_map)
    return val_map
```

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.

```
{ 'data_file' : '//tennis.csv', 'data_mappers' : [], 'data_project_columns' :
['Outlook', 'Temperature', 'Humidity', 'Windy', 'PlayTennis'], 'target_attribute' :
'PlayTennis'}
```

```
{'data_file': '//tennis.csv', 'data_mappers': [], 'data_project_columns':
['Outlook',
'Temperature', 'Humidity', 'Windy', 'PlayTennis'], 'target_attribute': 'PlayTennis'}
E:\suthan\codes\Machine Learning Lab\3\tennis.csv
{'Outlook', 'Windy', 'Humidity', 'Temperature'}
IF Outlook EQUALS Sunny AND Humidity EQUALS Normal THEN Yes
IF Outlook EQUALS Sunny AND Humidity EQUALS High THEN No
IF Outlook EQUALS Rainy AND Windy EQUALS True THEN No IF
Outlook EQUALS Rainy AND Windy EQUALS False THEN Yes IF
Outlook EQUALS Overcast THEN 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:

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.

3) 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 aj.

Step 3: updates weights incrementally, following the Presentation of each training example. This corresponds to a stochastic approxi- mation to gradient descent. To obtain the true gradient of E one would sum the Sj, xji 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.

```
from math import exp
from random import seed
from random import random

# Initialize a network
def initialize_network(n_inputs, n_hidden, n_outputs):
   network = list()
   hidden_layer = [{'weights':[random() for i in range(n_inputs + 1)]} for i in range(n_hidden)]
   network.append(hidden_layer)
   #print(network)
   output_layer = [{'weights':[random() for i in range(n_hidden + 1)]} for i in range(n_outputs)]
   network.append(output_layer)
   #print(network)

   return network

# Calculate neuron activation for an input
def activate(weights, inputs):
        activation = weights[-1]
        for i in range(len(weights)-1):
                activation += weights[i] * inputs[i]
        return activation

# Transfer neuron activation
def transfer(activation):
        return 1.0 / (1.0 + exp(-activation))

# Forward propagate input to a network output
def forward_propagate(network, row):
   inputs = row
   for layer in network:
      #print(layer)
      new_inputs = []
      for neuron in layer:
         activation = activate(neuron['weights'], inputs)
         neuron['output'] = transfer(activation)
         new_inputs.append(neuron['output'])
      inputs = new_inputs
      #print(inputs)
```

```
    #print(inputs)
    return inputs

# Calculate the derivative of an neuron output
def transfer_derivative(output):
        return output * (1.0 - output)

# Backpropagate error and store in neurons
def backward_propagate_error(network, expected):

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

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

# Train a network for a fixed number of epochs
def train_network(network, train, l_rate, n_epoch, n_outputs):
    for epoch in range(n_epoch):

        sum_error = 0
        for row in train:
            outputs = forward_propagate(network, row)
            #print(outputs)
            expected = [0 for i in range(n_outputs)]
            #print(expected)
            expected[row[-1]] = 1
```

```
            #print(expected)
            sum_error += sum([(expected[i]-outputs[i])**2 for i in range(len(expected))])
            #print(sum_error)
            backward_propagate_error(network, expected)
            update_weights(network, row, l_rate)
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))

# Test training backprop algorithm
seed(1)
dataset = [[2.7810836,2.550537003,0],
            [1.465489372,2.362125076,0],
            [3.396561688,4.400293529,0],
            [1.38807019,1.850220317,0],
            [3.06407232,3.005305973,0],
            [7.627531214,2.759262235,1],
            [5.332441248,2.088626775,1],
            [6.922596716,1.77106367,1],
            [8.675418651,-0.242068655,1],
            [7.673756466,3.508563011,1]]
n_inputs = len(dataset[0]) - 1
#print(n_inputs)
n_outputs = len(set([row[-1] for row in dataset]))
#print(n_outputs)
network = initialize_network(n_inputs, 2, n_outputs)
train_network(network, dataset, 0.5, 20, n_outputs)
for layer in network:
            print(layer)
```

OUTPUT

```
>epoch=0, lrate=0.500, error=6.350
>epoch=1, lrate=0.500, error=5.531
>epoch=2, lrate=0.500, error=5.221
>epoch=3, lrate=0.500, error=4.951
>epoch=4, lrate=0.500, error=4.519
>epoch=5, lrate=0.500, error=4.173
>epoch=6, lrate=0.500, error=3.835
>epoch=7, lrate=0.500, error=3.506
>epoch=8, lrate=0.500, error=3.192
>epoch=9, lrate=0.500, error=2.898
>epoch=10, lrate=0.500, error=2.626
>epoch=11, lrate=0.500, error=2.377
>epoch=12, lrate=0.500, error=2.153
>epoch=13, lrate=0.500, error=1.953
>epoch=14, lrate=0.500, error=1.774
>epoch=15, lrate=0.500, error=1.614
```

```
>epoch=16, lrate=0.500, error=1.472
>epoch=17, lrate=0.500, error=1.346
>epoch=18, lrate=0.500, error=1.233
>epoch=19, lrate=0.500, error=1.132
[{'weights': [-1.4688375095432327, 1.850887325439514, 1.0858178629550297], 'output':
0.029980305604426185, 'delta': -0.0059546604162323625}, {'weights':
[0.37711098142462157, -0.0625909894552989, 0.2765123702642716], 'output':
0.9456229000211323, 'delta': 0.0026279652850863837}]
[{'weights': [2.515394649397849, -0.3391927502445985, -0.9671565426390275],
'output':
0.23648794202357587, 'delta': -0.04270059278364587}, {'weights': [-
2.5584149848484263, 1.0036422106209202, 0.42383086467582715], 'output':
0.7790535202438367, 'delta': 0.0380313259643735 4}]
```

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.

3) 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.

4) 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.

5) **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%.

```
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)
    #print(separated)
    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)

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

def summarizeByClass(dataset):
   separated = separateByClass(dataset)
   summaries = {}
   #p=separated.items();
   #print(p)
   for classValue, instances in separated.items():
     # print(classValue)
      #print(instances)
      summaries[classValue] = summarize(instances)
   #print(summaries)
   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 = 0.75
        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 Cool=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

```
Split 6 rows into
Number of Training data: 4
Number of Test Data: 2
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]
The Test data set are:
[3.0, 3.0, 2.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 2.0, 5.0]
Actual values: [10.0]%
Actual values: [10.0, 5.0]%
Predictions: [5.0, 5.0]%
Accuracy: 50.0%
```

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

```
rom sklearn.datasets import fetch_20newsgroups
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
#pprint(fetch_20newsgroups)
categories = ['alt.atheism', 'soc.religion.christian','comp.graphics', 'sci.med']
twenty_train = fetch_20newsgroups(subset='train',categories=categories,shuffle=True)
#pprint(twenty_train)
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(twenty_train.target[0])
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)
print(X_train_tfidf)
X_train_tfidf.shape
print(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("Accuracy:", accuracy_score(twenty_test.target, predicted))
print(classification_report(twenty_test.target,predicted,target_names=twenty_test.target_names
))
#pprint(predicted)

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

OUTPUT

```
2257
1502
```

['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']

1

(0, 14887) 0.016797806021219684

(0, 29022) 0.1348710554299733

(0, 8696) 0.314400065528974

(0, 4017) 0.12491817585060791

(0, 33256) 0.11819702490105698

(0, 21661) 0.1962279892331408

(0, 9031) 0.3841803935867984

(0, 31077) 0.016797806021219684

(0, 9805) 0.21567205914741705

(0, 17366) 0.0744441018788533

(0, 32493) 0.07283773941616518

(0, 16916) 0.17358472047671197

(0, 19780) 0.24645540709354397

(0, 17302) 0.18626015109199115

(0, 23122) 0.036374916362300114

(0, 25663) 0.034290706362898604

(0, 16881) 0.0360441471878483

(0, 16082) 0.11382738609462074

(0, 23915) 0.017762318563562172

(0, 32142) 0.08865416253721688

(0, 33597) 0.06567578043186388

(0, 20253) 0.016864892977128034

(0, 587) 0.05966162012870271

(0, 12051) 0.037793189755988436

(0, 5201) 0.04316199700711876

: :

(2256, 13740) 0.08503348972488488

(2256, 14662) 0.10675645600140808

(2256, 20201) 0.07380572206614645

(2256, 12443) 0.5533848656066114

(2256, 30325) 0.2851629991538151

(2256, 4610) 0.09276072426248369

(2256, 33844) 0.09841352581656573

(2256, 17354) 0.10256037122854149
(2256, 26998) 0.1016498753357488
(2256, 20277) 0.1016498753357488
(2256, 20695) 0.10256037122854149
(2256, 20702) 0.08369317190645711
(2256, 9649) 0.09916899209319777
(2256, 9086) 0.10561084702611609
(2256, 26254) 0.09330694515090646
(2256, 17133) 0.19682705163313147
(2256, 4490) 0.11395377721095844
(2256, 13720) 0.09699270546460859
(2256, 5016) 0.12302132956698501
(2256, 9632) 0.11395377721095844
(2256, 11824) 0.12028503503707107
(2256, 29993) 0.12302132956698501
(2256, 1298) 0.12625767908616806
(2256, 2375) 0.12625767908616806
(2256, 3921) 0.13532523144219463
(2257, 35788)
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 aBayesian 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

Dataset : https://archive.ics.uci.edu/ml/datasets/Heart+Disease

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 identity direct causal influences between linked variables.Bayesian Calculus is used to determine state probabilities of each node or variable from conditional and prior probabilities

```
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]],lifeS
tyleEnum[x[4]],cholesterolEnum[x[5]],heartDiseaseEnum[x[6]]])
# Training data for machine learning todo: should import from csv
data = np.array(data)
print(data)
N = len(data)

# Input data column assignment
p_age = bp.nodes.Dirichlet(1.0*np.ones(5))
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

Few examples from the dataset are given below

age sex cp trestbps ... slope ca thal heartdisease

0 63.0 1.0 1.0 145.0 ... 3.0 0.0 6.0 0

1 67.0 1.0 4.0 160.0 ... 2.0 3.0 3.0 2

2 67.0 1.0 4.0 120.0 ... 2.0 2.0 7.0 1

3 37.0 1.0 3.0 130.0 ... 3.0 0.0 3.0 0

4 41.0 0.0 2.0 130.0 ... 1.0 0.0 3.0 0

[5 rows x 14 columns]

Attributes and datatypes

age float64
sex float64
cp float64
trestbps float64
chol float64
fbs float64
restecg float64
thalach float64
exang float64
oldpeak float64
slope float64
ca object
thal object
heartdisease int64
dtype: object
Learning CPDs using Maximum Likelihood Estimators...
Inferencing with Bayesian Network:
1.Probability of HeartDisease given Age=20

heartdisease	phi(heartdisease)
heartdisease_0	0.6791
heartdisease_1	0.1212
heartdisease_2	0.0810
heartdisease_3	0.0939
heartdisease_4	0.0247

2. Probability of HeartDisease given chol (Cholestoral) =100

heartdisease	phi(heartdisease)

heartdisease_0	0.5400
heartdisease_1	0.1533
heartdisease_2	0.1303
heartdisease_3	0.1259
heartdisease_4	0.0506

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.

Dataset:

➢ Iris dataset

➢ Number of Attributes:2 1. sepal length 2. sepal width

➢ Number of instances:150

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 = {x1, x2, …, xn}, where

x = (xi1, xi2, …, xir), is a vector in a real-valued space X ⊆ Rr, 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 ϵ 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 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 θ from randomly labeled data

➢ Iterate until convergence:

1. Expectation (E-step): Compute P(ci | 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, θ, from the probabilistically re-labeled data.

The EM Algorithm for Gaussian Mixtures

➢ The probability density function for multivariate_normal is

where mu is the mean, Sigma the covariance matrix, and k is the dimension of the space where x takes values.

Algorithm:

An arbitrary initial hypothesis h=<μ1, μ2 ,.., μk> is chosen.

The EM Algorithm iterates over two steps:

Step 1 (Estimation, E): Calculate the expected value E[zij] of each hidden variable zij, assuming that the current hypothesis h=<μ1, μ2 ,.., μk> holds.

Step 2 (Maximization, M): Calculate a new maximum likelihood hypothesis h‟=<μ1‟, μ2‟ „„ μk‟>, assuming the value taken on by each hidden variable zij is its expected value E[zij] calculated in step 1. Then replace the hypothesis h=<μ1, μ2 ,.., μk> by the new hypothesis h‟=<μ1‟, μ2‟ ,.., μk‟> and iterate.

EM ALGORITHM

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

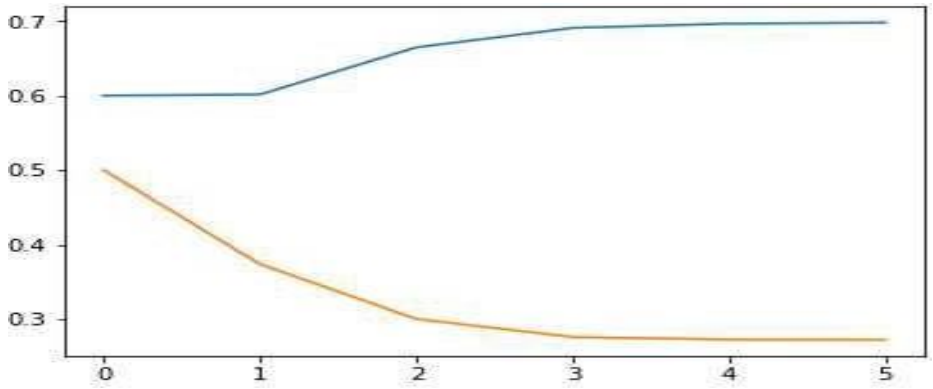
```
def get_binomial_log_likelihood(obs,probs):
    N = sum(obs);
    k = obs[0]
    # number of heads 12
    binomial_coeff = math.factorial(N) / (math.factorial(N-k) *math.factorial(k))
    prod_probs = obs[0]*math.log(probs[0]) + obs[1]*math.log(1-probs[0])
    log_lik = binomial_coeff + prod_probs
    return log_lik
data=[]
with open("cluster.csv") as tsv:
     for line in csv.reader(tsv):
         data=[int(i) for i in line]
head_counts = np.array(data)
tail_counts = 10-head_counts
#print(tail_counts)
experiments = list(zip(head_counts,tail_counts))
pA_heads = np.zeros(100); pA_heads[0] = 0.60
pB_heads = np.zeros(100); pB_heads[0] = 0.50
#print(pA_heads)
#print(pB_heads)
delta = 0.001
j = 0
improvement = float('inf')
print(improvement)
while (improvement>delta):
    expectation_A = np.zeros((len(experiments),2), dtype=float)
    expectation_B = np.zeros((len(experiments),2), dtype=float)
    for i in range(0,len(experiments)):
        e = experiments[i]
        ll_A =get_binomial_log_likelihood(e,np.array([pA_heads[j],1-pA_heads[j]]))
        ll_B = get_binomial_log_likelihood(e,np.array([pB_heads[j],1-pB_heads[j]]))
        weightA = math.exp(ll_A) / ( math.exp(ll_A) + math.exp(ll_B) )
        weightB = math.exp(ll_B) / ( math.exp(ll_A) + math.exp(ll_B) )
        expectation_A[i] = np.dot(weightA, e)
        expectation_B[i] = np.dot(weightB, e)
    pA_heads[j+1] = sum(expectation_A)[0] / sum(sum(expectation_A));
    pB_heads[j+1] = sum(expectation_B)[0] / sum(sum(expectation_B));
    improvement = ( max( abs(np.array([pA_heads[j+1],pB_heads[j+1]])-
np.array([pA_heads[j],pB_heads[j]]) )) )
    print(np.array([pA_heads[j+1],pB_heads[j+1]])-np.array([pA_heads[j],pB_heads[j]]) )
    j = j+1
plt.figure();
plt.plot(range(0,j),pA_heads[0:j])
plt.plot(range(0,j),pB_heads[0:j])
plt.show()
```

EM ALGORITHM

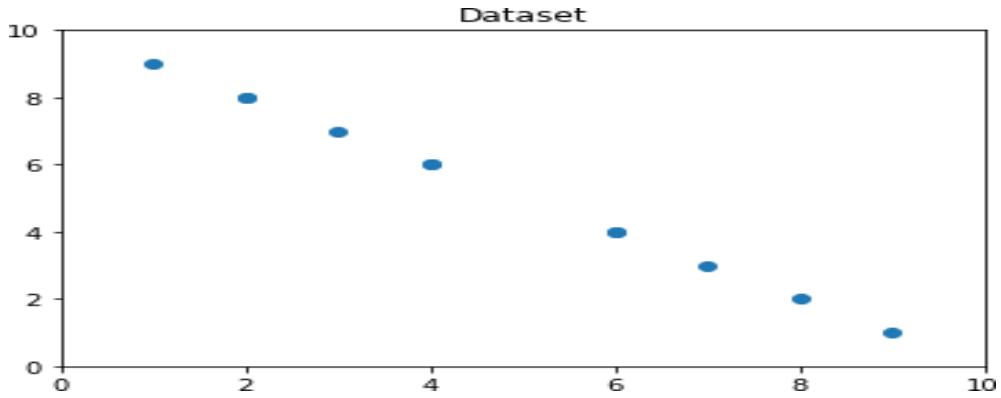
```
from sklearn.cluster import KMeans
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
import csv
data=[]
ydata=[]
with open("cluster.csv") as tsv:
    for line in csv.reader(tsv):
        data=[int(i) for i in line]
        print(data)
        ydata=[10-int(i) for i in line]
        print(ydata)
x1 = np.array(data)
x2 = np.array(ydata)
print(x1)
print(x2)
plt.plot()
plt.xlim([0, 10])
plt.ylim([0, 10])
plt.title('Dataset')
plt.scatter(x1, x2)
plt.show()
plt.plot()
X = np.array(list(zip(x1, x2))).reshape(len(x1), 2)
print(X)
colors = ['b', 'g','r']
markers = ['o', 'v', 's']
K = 3
kmeans_model = KMeans(n_clusters=K).fit(X)
#print(kmeans_model)
plt.plot()
for i, l in enumerate(kmeans_model.labels_):
    print(i,l)
    plt.plot(x1[i], x2[i], color=colors[l],marker=markers[l],ls='None')
    plt.xlim([0, 10])
    plt.ylim([0, 10])
```

OUTPUT

```
 [ 0.00151004 -0.12639463]
[ 0.06329684 -0.07364559]
[ 0.02612144 -0.02430469]
[ 0.00573256 -0.00349713]
[ 0.00137583 -0.00012734]
[0.00044257 0.00015722]
```

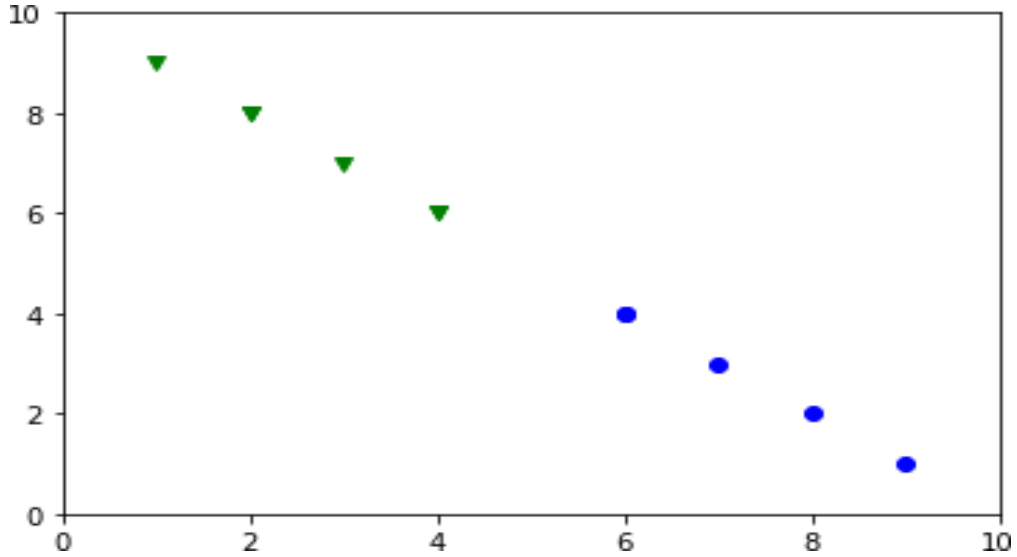


Program 8KM
[2, 9, 8, 4, 7, 2, 3, 1, 6, 4, 6]
[8, 1, 2, 6, 3, 8, 7, 9, 4, 6, 4]
[2 9 8 4 7 2 3 1 6 4 6]
[8 1 2 6 3 8 7 9 4 6 4]



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

8 0
9 1
10 0



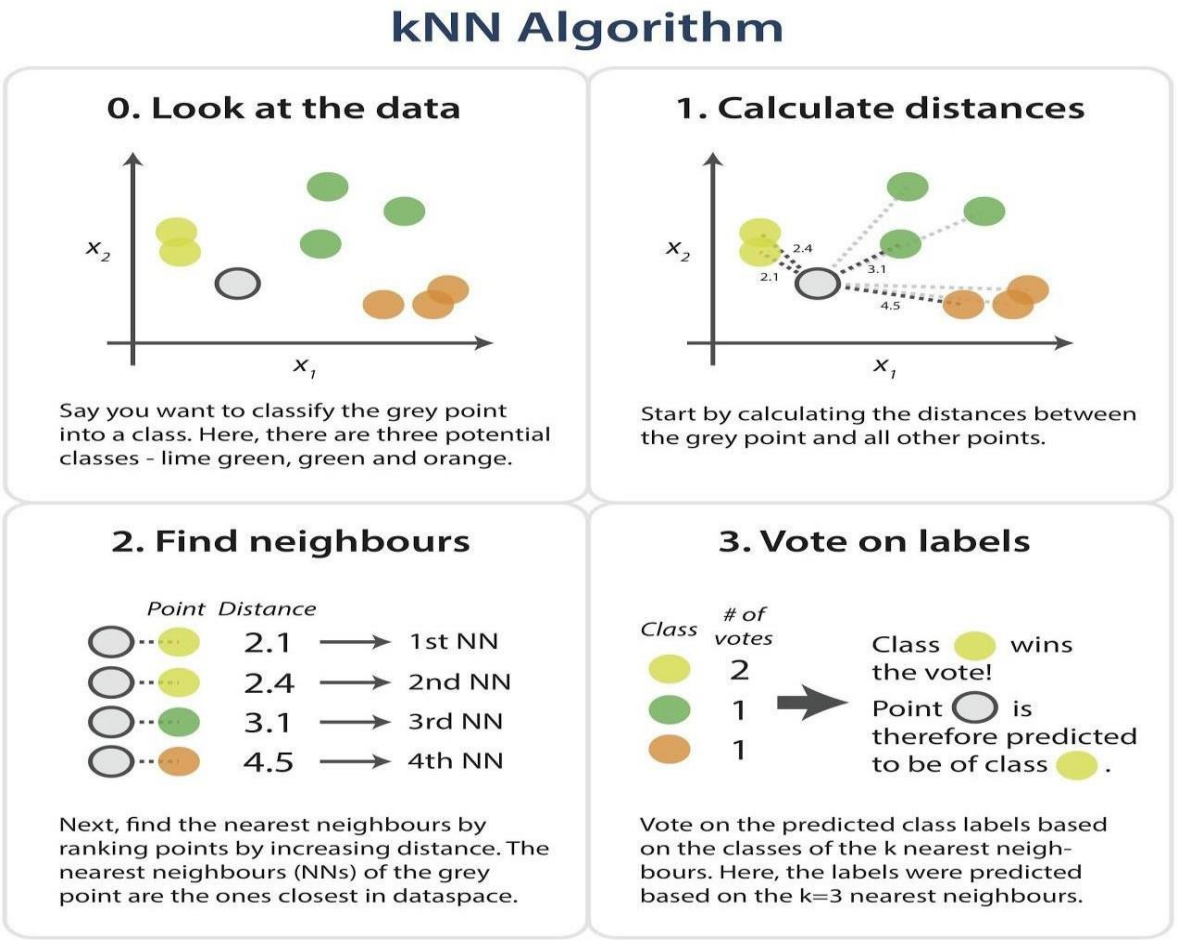
Program 9. Write a program to implement K-nearest neighbour algorithm to classify iris dataset. Print both correct and wrong predication using python machine learning.

k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] 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



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

```
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) + ', actual=' + repr(testSet[x][-1]))
        accuracy = getAccuracy(testSet, predictions)
        print('\n The Accuracy is: ' + repr(accuracy) + '%')

main()
```

OUTPUT

```
Number of Training data: 99
Number of Test Data: 50
The predictions are:
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
```

predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-setosa', actual='Iris-setosa'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-virginica', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-versicolor', actual='Iris-versicolor'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-versicolor', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'

predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
predicted='Iris-virginica', actual='Iris-virginica'
The Accuracy is: 96.0%

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:

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

```
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)
```

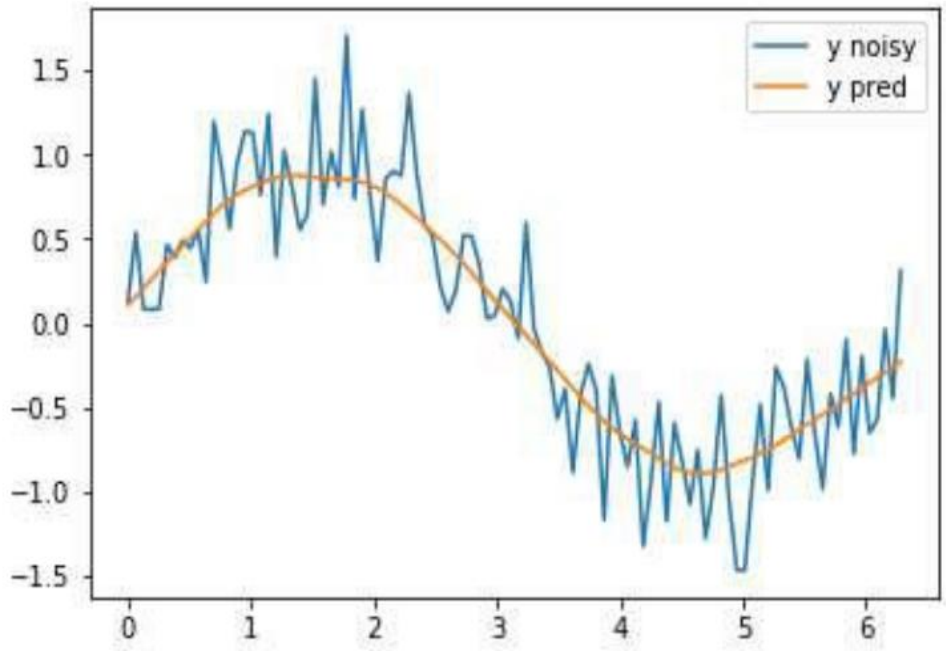
```
    w = (1 - w**3)**3
    yest = np.zeros(n)
    delta = np.ones(n)
    for iteration in range(iter):
        for i in range(n):
            weights = delta * w[:,i]
            b = np.array([np.sum(weights*y), np.sum(weights*y*x)])
            A = np.array([[np.sum(weights), np.sum(weights*x)],
                  [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

```
==========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==========
[ 0.12909628 0.5378001 0.08507775 0.08261955 0.08748326 0.46390454
0.39007129 0.49168683 0.44534231 0.55328598 0.24690547 1.19597387
0.92244303 0.56004488 0.9561929 1.13800942 1.12911587 0.76110236
1.23982502 0.39462141 1.02459433 0.81259471 0.55535331 0.64550225
1.45040127 0.70659902 1.01732347 0.81062276 1.706929 0.73681414
1.26884138 0.76529336 0.36909709 0.85530574 0.90229748 0.87607598
1.36419146 0.88365564 0.58595606 0.51983462 0.2214239 0.07172939
0.18989997 0.51956736 0.51702737 0.35407817 0.02826523 0.04505194
0.20336912 0.13206237 -0.08791493 0.59561087 -0.02677494 -0.17386743
-0.25492254 -0.5663511 -0.38921533 -0.88414287 -0.41859126 -0.23967376
```

-0.39954993 -1.16303626 -0.31169895 -0.63970167 -0.852607 -0.57347235
-1.32173606 -0.9393813 -0.46767613 -1.17360841 -0.59243843 -0.81945994
-1.07030086 -0.75048246 -1.27563295 -0.96278885 -0.42760459 -1.04809671
-1.46120317 -1.46428521 -0.94471363 -0.47953744 -0.98878457 -0.26220194
-0.37523066 -0.61756717 -0.80162952 -0.21397883 -0.65632663 -0.98149854
-0.41874218 -0.61484032 -0.09496718 -0.77232454 -0.19296106 -0.64690202
-0.56981287 -0.03604453 -0.44445902 0.31323543]



VIVO VOCE QUESTIONS AND ANSWER

1) What is Machine learning?

Machine learning is a branch of computer science which deals with system programming in order to automatically learn and improve with experience. For example: Robots are programed so that they can perform the task based on data they gather from sensors. It automatically learns programs from data.

2) Mention the difference between Data Mining and Machine learning?

Machine learning relates with the study, design and development of the algorithms that give computers the capability to learn without being explicitly programmed. While, data mining can be defined as the process in which the unstructured data tries to extract knowledge or unknown interesting patterns. During this process machine, learning algorithms are used.

3) What is 'Overfitting' in Machine learning?

In machine learning, when a statistical model describes random error or noise instead of underlying relationship 'overfitting' occurs. When a model is excessively complex, overfitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been overfit.

4) Why overfitting happens?

The possibility of overfitting exists as the criteria used for training the model is not the same as the criteria used to judge the efficacy of a model.

5) How can you avoid overfitting ?

By using a lot of data overfitting can be avoided, overfitting happens relatively as you have a small dataset, and you try to learn from it. But if you have a small database and you are forced to come with a model based on that. In such situation, you can use a technique known as **cross validation**. In this method the dataset splits into two section, testing and training datasets, the testing dataset will only test the model while, in training dataset, the datapoints will come up with the model.

6) What is inductive machine learning?

The inductive machine learning involves the process of learning by examples, where a system, from a set of observed instances tries to induce a general rule.

7) What are the five popular algorithms of Machine Learning?

a) Decision Trees

b) Neural Networks (back propagation)

c) Probabilistic networks

d) Nearest Neighbor

e) Support vector machines

8) What are the different Algorithm techniques in Machine Learning?

The different types of techniques in Machine Learning are

a) Supervised Learning

b) Unsupervised Learning

c) Semi-supervised Learning

d) Reinforcement Learning

e) Transduction

f) Learning to Learn

9) What are the three stages to build the hypotheses or model in machine learning?

a) Model building

b) Model testing

c) Applying the model

10) What is the standard approach to supervised learning?

The standard approach to supervised learning is to split the set of example into the training set and the test.

11) What is 'Training set' and 'Test set'?

In various areas of information science like machine learning, a set of data is used to discover the potentially predictive relationship known as 'Training Set'. Training set is an examples

given to the learner, while Test set is used to test the accuracy of the hypotheses generated by the learner, and it is the set of example held back from the learner. Training set are distinct from Test set.

12) List down various approaches for machine learning?

The different approaches in Machine Learning are

a) Concept Vs Classification Learning

b) Symbolic Vs Statistical Learning

c) Inductive Vs Analytical Learning

13) What is not Machine Learning?

a) Artificial Intelligence

b) Rule based inference

14) Explain what is the function of 'Unsupervised Learning'?

a) Find clusters of the data

b) Find low-dimensional representations of the data

c) Find interesting directions in data

d) Interesting coordinates and correlations

e) Find novel observations/ database cleaning

15) Explain what is the function of 'Supervised Learning'?

a) Classifications

b) Speech recognition

c) Regression

d) Predict time series

e) Annotate strings

16) What is algorithm independent machine learning?

Machine learning in where mathematical foundations is independent of any particular classifier or learning algorithm is referred as algorithm independent machine learning?

17) What is the difference between artificial learning and machine learning?

Designing and developing algorithms according to the behaviours based on empirical data are known as Machine Learning. While artificial intelligence in addition to machine learning, it also covers other aspects like knowledge representation, natural language processing, planning, robotics etc.

18) What is classifier in machine learning?

A classifier in a Machine Learning is a system that inputs a vector of discrete or continuous feature values and outputs a single discrete value, the class.

19) What are the advantages of Naive Bayes?

In Naïve Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. The main advantage is that it can't learn interactions between features.

20) In what areas Pattern Recognition is used?

Pattern Recognition can be used in

a) Computer Vision

b) Speech Recognition

c) Data Mining

d) Statistics

e) Informal Retrieval

f) Bio-Informatics

21) What is Genetic Programming?

Genetic programming is one of the two techniques used in machine learning. The model is based on the testing and selecting the best choice among a set of results.

22) What is Inductive Logic Programming in Machine Learning?

Inductive Logic Programming (ILP) is a subfield of machine learning which uses logical programming representing background knowledge and examples.

23) What is Model Selection in Machine Learning?

The process of selecting models among different mathematical models, which are used to describe the same data set is known as Model Selection. Model selection is applied to the fields of statistics, machine learning and data mining.

24) What are the two methods used for the calibration in Supervised Learning?

The two methods used for predicting good probabilities in Supervised Learning are

a) Platt Calibration

b) Isotonic Regression

These methods are designed for binary classification, and it is not trivial.

25) Which method is frequently used to prevent overfitting?

When there is sufficient data 'Isotonic Regression' is used to prevent an overfitting issue.

26) What is the difference between heuristic for rule learning and heuristics for decision trees?

The difference is that the heuristics for decision trees evaluate the average quality of a number of disjointed sets while rule learners only evaluate the quality of the set of instances that is covered with the candidate rule.

27) What is Perceptron in Machine Learning?

In Machine Learning, Perceptron is an algorithm for supervised classification of the input into one of several possible non-binary outputs.

28) Explain the two components of Bayesian logic program?

Bayesian logic program consists of two components. The first component is a logical one ; it consists of a set of Bayesian Clauses, which captures the qualitative structure of the domain. The second component is a quantitative one, it encodes the quantitative information about the domain.

29) What are Bayesian Networks (BN) ?

Bayesian Network is used to represent the graphical model for probability relationship among a set of variables .

30) Why instance based learning algorithm sometimes referred as Lazy learning algorithm?

Instance based learning algorithm is also referred as Lazy learning algorithm as they delay the induction or generalization process until classification is performed.

31) What are the two classification methods that SVM (Support Vector Machine) can handle?

a) Combining binary classifiers

b) Modifying binary to incorporate multiclass learning

32) What is ensemble learning?

To solve a particular computational program, multiple models such as classifiers or experts are strategically generated and combined. This process is known as ensemble learning.

33) Why ensemble learning is used?

Ensemble learning is used to improve the classification, prediction, function approximation etc of a model.

34) When to use ensemble learning?

Ensemble learning is used when you build component classifiers that are more accurate and independent from each other.

35) What are the two paradigms of ensemble methods?

The two paradigms of ensemble methods are

a) Sequential ensemble methods

b) Parallel ensemble methods

36) What is the general principle of an ensemble method and what is bagging and boosting in ensemble method?

The general principle of an ensemble method is to combine the predictions of several models built with a given learning algorithm in order to improve robustness over a single model. Bagging is a method in ensemble for improving unstable estimation or classification schemes. While boosting method are used sequentially to reduce the bias of the combined model. Boosting and Bagging both can reduce errors by reducing the variance term.

37) What is bias-variance decomposition of classification error in ensemble method?

The expected error of a learning algorithm can be decomposed into bias and variance. A bias term measures how closely the average classifier produced by the learning algorithm matches the target function. The variance term measures how much the learning algorithm's prediction fluctuates for different training sets.

38) What is an Incremental Learning algorithm in ensemble?

Incremental learning method is the ability of an algorithm to learn from new data that may be available after classifier has already been generated from already available dataset.

39) What is PCA, KPCA and ICA used for?

PCA (Principal Components Analysis), KPCA (Kernel based Principal Component Analysis) and ICA (Independent Component Analysis) are important feature extraction techniques used for dimensionality reduction.

40) What is dimension reduction in Machine Learning?

In Machine Learning and statistics, dimension reduction is the process of reducing the number of random variables under considerations and can be divided into feature selection and feature extraction

41) What are support vector machines?

Support vector machines are supervised learning algorithms used for classification and regression analysis.

42) What are the components of relational evaluation techniques?

The important components of relational evaluation techniques are

a) Data Acquisition

b) Ground Truth Acquisition

c) Cross Validation Technique

d) Query Type

e) Scoring Metric

f) Significance Test

43) What are the different methods for Sequential Supervised Learning?

The different methods to solve Sequential Supervised Learning problems are

a) Sliding-window methods

b) Recurrent sliding windows

c) Hidden Markow models

d) Maximum entropy Markow models

e) Conditional random fields

f) Graph transformer networks

44) What are the areas in robotics and information processing where sequential prediction problem arises?

The areas in robotics and information processing where sequential prediction problem arises are

a) Imitation Learning

b) Structured prediction

c) Model based reinforcement learning

45) What is batch statistical learning?

Statistical learning techniques allow learning a function or predictor from a set of observed data that can make predictions about unseen or future data. These techniques provide guarantees on the performance of the learned predictor on the future unseen data based on a statistical assumption on the data generating process.

46) What is PAC Learning?

PAC (Probably Approximately Correct) learning is a learning framework that has been introduced to analyze learning algorithms and their statistical efficiency.

47) What are the different categories you can categorized the sequence learning process?

a) Sequence prediction

b) Sequence generation

c) Sequence recognition

d) Sequential decision

48) What is sequence learning?

Sequence learning is a method of teaching and learning in a logical manner.

49) What are two techniques of Machine Learning ?

The two techniques of Machine Learning are

a) Genetic Programming

b) Inductive Learning

50) Give a popular application of machine learning that you see on day to day basis?

The recommendation engine implemented by major ecommerce websites uses Machine Learning

Viva Questions

1. What is machine learning?
2. Define supervised learning
3. Define unsupervised learning
4. Define semi supervised learning
5. Define reinforcement learning
6. What do you mean by hypotheses

7. What is classification

8. What is clustering

9. Define precision, accuracy and recall

10. Define entropy

11. Define regression

12. How Knn is different from k-means clustering

13. What is concept learning

14. Define specific boundary and general boundary

15. Define target function

16. Define decision tree

17. What is ANN

18. Explain gradient descent approximation

19. State Bayes theorem

20. Define Bayesian belief networks

21. Differentiate hard and soft clustering

22. Define variance

23. What is inductive machine learning

24. Why K nearest neighbour algorithm is lazy learning algorithm

25. Why naïve Bayes is naïve 26. Mention classification algorithms

27. Define pruning

28. Differentiate Clustering and classification

29. Mention clustering algorithms

30. Define Bias

REFERENCES

List of Text Books
1. Tom M. Mitchell, Machine Learning, India Edition 2013, McGraw Hill Education.
List of Reference Books
1. Trevor Hastie, Robert Tibshirani, Jerome Friedman, h The Elements of Statistical Learning, 2nd edition, springer series in statistics. 2. Ethem Alpaydın, Introduction to machine learning, second edition, MIT press.
List of URLs, Text Books, Notes, Multimedia Content, etc
1. https://medium.com/ml-research-lab/machine-learning-algorithm-overview 2. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/