

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELGAVI-590018, KARNATAKA



LABORATORY MANUAL

Semester: VI (CBCS SCHEME)
MACHINE LEARNING- 21AIL66

Prepared by:

Prof. MANJULA S S
Assistant Professor
Dept. of CSE

Scrutinized by,
Prof. VASUDEVA R



C BYREGOWDA INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

An ISO 9001: 2015 Certified Institute

Kolar – Srinivasapur Road

Kolar-563101

SL. NO	<u>CONTENTS</u>
1	Vision and Mission of the Department
2	Course Details: Syllabus
3	Introduction
4	<p>Illustrate and Demonstrate the working model and principle of Find-S algorithm.</p> <p>Program: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.</p>
5	<p>Demonstrate the working model and principle of candidate elimination algorithm.</p> <p>Program: 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</p>
6	<p>Aim: To construct the Decision tree using the training data sets under supervised learning concept.</p> <p>Program: 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.</p>
7	<p>Aim: To understand the working principle of Artificial Neural network with feed forward and feed backward principle.</p> <p>Program: Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.</p>
8	<p>Aim: Demonstrate the text classifier using Naïve bayes classifier algorithm.</p> <p>Program: Write a program to implement the naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.</p>
9	<p>Aim: Demonstrate and Analyze the results sets obtained from Bayesian belief network Principle.</p> <p>Program: 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 Python ML library classes/API.</p>
10	<p>Aim: Implement and demonstrate the working model of K-means clustering algorithm with Expectation Maximization Concept.</p> <p>Program: 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 Python ML library classes/API in the program.</p>

11	<p>Aim: Demonstrate and analyze the results of classification based on KNN Algorithm.</p> <p>Program: Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.</p>
12	<p>Aim: Understand and analyze the concept of Regression algorithm techniques.</p> <p>Program: Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.</p>
13	<p>Aim: Implement and demonstrate classification algorithm using Support vector machine Algorithm.</p> <p>Program: Implement and demonstrate the working of SVM algorithm for classification.</p>

C BYRE GOWDA INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

VISION

- Prepare competent professionals embedded with human values to meet the challenges in the field of Computer Science & Engineering.

MISION

MD1	To empower the students with the fundamentals for a successful career in the field of Computer Science & Engineering.
MD2	To impart Quality technical education and training to the students to make them Competent Engineers and thereby contribute to research and scientific development.
MD3	To create technical manpower for meeting the current and future demands of Industry

Syllabus

MACHINE LEARNING LABORATORY

Course Code	21AIL66	CIE Marks	50
Teaching Hours/Week(L:T:P:S)	0:0:2:0	SEE Marks	50
Total Hours of Pedagogy	24	Total Marks	100
Credits	1	Exam Hours	03

Course Learning Objectives:

CLO 2. To learn and understand the Importance Machine learning Algorithms

CLO 3. Compare and contrast the learning techniques like ANN approach, Bayesian learning and reinforcement learning.

CLO 4. Able to solve and analyse the problems on ANN, Instance based learning and Reinforcement learning techniques.

CLO 5. To impart the knowledge of clustering and classification Algorithms for predictions and evaluating Hypothesis.

	Prerequisite
	<ul style="list-style-type: none"> Students should be familiarized about Python installation and setting Python environment Usage and installation of Anaconda should be introduced https://www.anaconda.com/products/individual Should have the knowledge about Probability theory, Statistics theory and linear Algebra. Should have the knowledge of numpy, pandas, scikit-learn and scipy library packages.
Sl. No.	PART A – List of problems for which student should develop program and execute in the Laboratory
1	<p>Aim: Illustrate and Demonstrate the working model and principle of Find-S algorithm.</p> <p>Program: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.</p> <p>Text Book 1: Ch2</p>
2	<p>Aim: Demonstrate the working model and principle of candidate elimination algorithm.</p> <p>Program: 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.</p> <p>Text Book 1: Ch2</p> <p>Reference: https://www.youtube.com/watch?v=tfpAm4kxGQI</p>
3	<p>Aim: To construct the Decision tree using the training data sets under supervised learning concept.</p> <p>Program: 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.</p> <p>Text Book 1: Ch 3</p>
4	<p>Aim: To understand the working principle of Artificial Neural network with feed forward and feed backward principle.</p> <p>Program: Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.</p> <p>Text Book 1: Ch 4</p>

5	<p>Aim: Demonstrate the text classifier using Naïve bayes classifier algorithm.</p> <p>Program: Write a program to implement the naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.</p> <p>Text Book 1: Ch6</p>
6	<p>Aim: Demonstrate and Analyse the results sets obtained from Bayesian belief network Principle.</p> <p>Program:- 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 Python ML library classes/API.</p> <p>Text Book 1: Ch 6</p>
7	<p>Aim: Implement and demonstrate the working model of K-means clustering algorithm with Expectation Maximization Concept.</p> <p>Program: 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 Python ML library classes/API in the program.</p> <p>Text Book 1: Ch 8</p>
8	<p>Aim: Demonstrate and analyse the results of classification based on KNN Algorithm.</p> <p>Program: 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.</p> <p>Text Book 1: Ch 8</p>
9	<p>Aim: Understand and analyse the concept of Regression algorithm techniques.</p> <p>Program: Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.</p> <p>Text Book 1: Ch8</p>
10	<p>Aim: Implement and demonstrate classification algorithm using Support vector machine Algorithm.</p> <p>Program: Implement and demonstrate the working of SVM algorithm for classification.</p> <p>Text Book 2: Ch6</p>
Pedagogy	For the above experiments the following pedagogy can be considered. Problem based learning, Active learning, MOOC, Chalk & Talk
PART B	
	A problem statement for each batch is to be generated in consultation with the co-examiner and student should develop an algorithm, program and execute the Program for the given problem with appropriate outputs.
<p>Course Outcomes: At the end of the course the student will be able to:</p> <p>CO 1. Understand the Importance of different classification and clustering algorithms.</p> <p>CO 2. Demonstrate the working of various algorithms with respect to training and test data sets.</p> <p>CO 3. Illustrate and analyze the principles of Instance based and Reinforcement learning techniques. CO 4. Elicit the importance and Applications of Supervised and unsupervised machine learning.</p> <p>CO 5. Compare and contrast the Bayes theorem principles and Q learning approach.</p>	

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student

shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks). The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in - 60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Students can pick one experiment from the questions lot of PART A with equal choice to all the students in a batch.

For PART B examiners should frame a question for each batch, student should

develop an algorithm, program, execute and demonstrate the results with appropriate output for the given problem.

- *Weightage of marks for PART A is 80% and for PART B is 20%. General rubrics suggested to be followed for part A and part B.*
- Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero (Not allowed for Part B).
- The duration of SEE is 03 hours
- Rubrics suggested in Annexure-II of Regulation book

Text Books:

1. Tom M Mitchell, “Machine Learning”, 1st Edition, McGraw Hill Education, 2017.
2. Nello Cristianini, John Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2013
3. Allen B. Downey, “Think Python: How to Think Like a Computer Scientist”, 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at <http://greenteapress.com/thinkpython2/thinkpython2.pdf>)

Suggested Web Links / E Resource

1. <https://www.kaggle.com/general/95287>
2. <https://web.stanford.edu/~hastie/Papers/ESLII.pdf>

Prerequisites

1. Programming experience in Python
2. Knowledge of basic Machine Learning Algorithms
3. Knowledge of common statistical methods and data analysis best practices.

Lab outcomes:

At the end of the course, the student will be able to;

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

Software Requirements

1. Python version 3.5 and above
2. Machine Learning packages
 - Scikit-Learn
 - Numpy - matrices and linear algebra
 - Scipy - many numerical routines
 - Matplotlib- creating plots of data
 - Pandas –facilitates structured/tabular data manipulation and visualisations
 - Pomegranate –for fast and flexible probabilistic models
3. An Integrated Development Environment (IDE) for Python Programming

Anaconda

It contains a list of Python packages; tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages. Anaconda is used for scientific computing, data science, statistical analysis, and machine learning

Operating System

Windows/Linux

Anaconda Python distribution is compatible with Linux or windows.

Python

It is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-

oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is open source, multi-paradigm, Object-oriented and structured programming supported, Language. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution, which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions.

Python packages for machine learning

NumPy

NumPy (stands for Numerical Python) is one of the most famous and commonly used python package among data scientists and ML engineers. This is a part of Python's SciPy Stack, which is basically a collection of software specially designed for scientific computations. It provides several features to work with n-dimensional arrays and matrices in python. This library provides vectorization of mathematical operations on the NumPy array type which adds up to the performance of the execution.

Pandas

The Pandas library is too a well-known library in the world of Analytics and Data Sciences. This package is primarily designed to work with simple and relational data. This is one of the favorite libraries among the data scientists for easy data manipulation, visualization as well as aggregation.

If talking about the data structures, there are basically two prime data structures available in the library which are Series (one-dimensional) & Data Frames (two-dimensional) and we think these are not that significant to talk about as of now.

The basic functionalities that Pandas provides:

- We can very easily delete as well as add a columns from DataFrame
- Pandas can be used to convert the Data Structures in to DataFrame objects.
- If we have any redundancy in the dataset in the form of missing data represented as „NaN“, this is the perfect tool to remove that
- Can be used for grouping of the attributes based strictly on their functionality.

Libraries for Machine Learning

Scikit-Learn: The library is focused on modeling data. Some popular groups of models provided by scikit-learn include:

- Clustering: for grouping unlabeled data such as KMeans.
- Cross Validation: for estimating the performance of supervised models on unseen data.
- Datasets: for test datasets and for generating datasets with specific properties for investigating

model behavior.

- Dimensionality Reduction: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
- Ensemble methods: for combining the predictions of multiple supervised models.
- Feature extraction: for defining attributes in image and text data.
- Feature selection: for identifying meaningful attributes from which to create supervised models.
- Parameter Tuning: for getting the most out of supervised models.
- Manifold Learning: For summarizing and depicting complex multi-dimensional data.
- Supervised Models: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

- Use Keras if you need a deep learning library that:
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

TensorFlow

TensorFlow is an open source software library for numerical computation using dataflow graphs. Nodes in the graph represents mathematical operations, while graph edges represent multi-dimensional data arrays (aka tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API

Best Python Machine Learning IDEs

Spyder is coming to our very first focus i.e. *Spyder*. This IDE got this short name from it's name itself: "Scientific Python Development Environment". Pierre Raybaut is the author of Spyder and it got officially released on October 18, 2009 and is written solely in Python.

Features at a glance:

- Very simple and light-weight IDE with detailed documentation and quite easy to install.
- This is an open source editor and supports code completion, introspection, goto definition as well as horizontal and vertical splitting.
- This editor comes with a *Documentation Viewer* where you can see the documentation related to classes or functions you gotta use.
- Like most of the IDEs, this also supports *Variable Explorer* which is a helpful tool to explore and edit the variables that we have created during file execution.

- It supports runtime debugging i.e. the errors will be seen on the screen as soon as you type them.

Geany is primarily a Python machine learning IDE authored by Enrico Troger and got officially released on October 19, 2005. It has been written in C & C++ and is a light-weight IDE. Despite of being a small IDE it is as capable as any other IDE present out there.

Features at a glance:

- Geany's editor supports highlighting of the Syntax and line numbering.
- It comes equipped with the features like code completion, auto closing of braces, auto HTML and XML tags closing.
- It also comes with code folding.
- This IDE supports code navigation.

Rodeo: This is special we got here. This is a Python IDE that primarily focuses and built for the purpose of machine learning and data science. This particular IDE use IPython kernel (you will know this later) and was authored by Yhat.

Features at a glance

- It is mainly famous due to its ability to let users explore, compare and interact with the data frames & plots.
- Like Geany's editor this also comes with a editor that has capability of auto-completion, syntax highlighting.
- This also provides a support for IPython making the code writing fast.
- Also Rodeo comes with Python tutorials integrated within which makes it quite favourable for the users.
- This IDE is well known for the fact that for the data scientists and engineers who work on RStudio IDE can very easily adapt to it

PyCharm is the IDE which is most famous in the professional world whether it is for data science or for conventional Python programming. This IDE is built by one of the big company out there that we all might have heard about: JetBrains, company released the official version of PyCharm in October 2010.

PyCharm comes in two different editions: Community Edition which we all can have access to essentially for free and second one is the Professional Edition for which you will need to pay some bucks.

Features at a glance

- It includes code completion, auto-indentation and code formatting.
- This also comes with runtime debugger i.e. will display the errors as soon as you type them.
- It contains PEP-8 that enables writing neat codes.
- It consist of debugger for Javascript and Python with a GUI.

- It has one of the most advanced documentation viewer along with video tutorials.
- PyCharm being accepted widely among big companies for the purpose of Machine Learning is due to its ability to provide support for important libraries like Matplotlib, NumPy and Pandas.

JuPyter Notebook or IPython Notebook

It is simple and this became a sensational IDE among the data enthusiasts as it is the descendant of IPython. Best thing about JuPyter is that there you can very easily switch between the different versions of python (or any other language) according to your preference.

Features at a glance

- It's an open source platform
- It can support up to 40 different languages to work on including languages beneficial for data sciences like R, Python, Julia, etc.
- It supports sharing live codes, and even documents with equations and visualizations.
- In JuPyter you can produce outputs in the form of images, videos and even LaTeX with the help of several useful widgets.

Program 1

1	Aim	Illustrate and demonstrate the working model and principle of Find-S algorithm
	Program	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples

CONCEPT - FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS**FIND-S Algorithm**

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

To illustrate this algorithm, assume the learner is given the sequence of training examples from the

EnjoySport task

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

- The first step of FIND-S is to initialize h to the most specific hypothesis in H
 $h = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
- Consider the first training example
 $x_1 = \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle, +$

Observing the first training example, it is clear that hypothesis h is too specific. None of the "0" constraints in h are satisfied by this example, so each is replaced by the next *more general*

constraint that fits the example

$h_1 = \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle$

- Consider the second training example

$x_2 = \langle \text{Sunny, Warm, High, Strong, Warm, Same} \rangle, +$

The second training example forces the algorithm to further generalize h , this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example

$h_2 = \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle$

- Consider the third training example

$x_3 = \langle \text{Rainy, Cold, High, Strong, Warm, Change} \rangle, -$

Upon encountering the third training the algorithm makes no change to h . The FIND-S algorithm simply ignores every negative example.

$h_3 = \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle$

- Consider the fourth training example

$x_4 = \langle \text{Sunny Warm High Strong Cool Change} \rangle, +$

The fourth example leads to a further generalization of h

$h_4 = \langle \text{Sunny, Warm, ?, Strong, ?, ?} \rangle$

The key property of the FIND-S algorithm

- It is incremental learning i.e., algorithm learns by processing one training example at a time, updating its hypothesis based on each example
- FIND-S is computationally efficient, especially for small to medium-sized datasets and can handle noisy data and incomplete training sets
- FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples
- FIND-S algorithm's final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H , and provided the training examples are correct.

```

import csv

a = []
with open('enjoysport.csv', 'r') as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)

print("\n The total number of training instances are : ", len(a))

num_attribute = len(a[0]) - 1
print("\n The initial hypothesis is : ")
hypothesis = ['0'] * num_attribute
print(hypothesis)

for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'
    print("\n The hypothesis for the training instance { } is : \n".format(i + 1), hypothesis)

print("\n The Maximally specific hypothesis for the training instance is ")
print(hypothesis)

```

Output:

```

PS C:\Users\naray\Downloads\MLL> & "C:/Program Files/Python312/python.exe" c:/Users/naray/Downloads/MLL/prg1finds.py
[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forecast', 'enjoysport'], ['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 total number of training instances are : 5

The initial hypothesis is :
['0', '0', '0', '0', '0', '0']

The hypothesis for the training instance 1 is :
['0', '0', '0', '0', '0', '0']

The hypothesis for the training instance 2 is :
['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

The hypothesis for the training instance 3 is :
['sunny', 'warm', '?', 'strong', 'warm', 'same']

The hypothesis for the training instance 4 is :
['sunny', 'warm', '?', 'strong', 'warm', 'same']

The hypothesis for the training instance 5 is :
['sunny', 'warm', '?', 'strong', '?', '?']

The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
PS C:\Users\naray\Downloads\MLL>

```


Program 2

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

Objective	Incrementally builds the version space given a hypothesis space H and a set E of examples.
Dataset	Tennis data set: This data set contain the set of example days on which playing of tennis is possible or not. Based on attribute Sky, AirTemp, Humidity, Wind, Water and Forecast. The dataset has 14 instances.
ML algorithm	Supervised Learning-Candidate elimination algorithm
Description	The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. ... The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

```
import numpy as np
import pandas as pd

data = pd.read_csv('enjoysport.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):
        print("For Loop Starts")
        if target[i] == "yes":
            print("If instance is Positive ")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'

        if target[i] == "no":
            print("If instance is Negative ")
            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)
print("\n")
print("\n")
```

```
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")
```

DATASET:

enjoysport.csv

(Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport)
sunny	warm	normal	strong	warm	same	Y
sunny	warm	high	strong	warm	same	Y
rainy	cold	high	strong	warm	change	N
sunny	warm	high	strong	cool	change	Y

Output:

```
● cbit@cbit-H410M-H-V3:~/MACHINE LEARNING LAB/21AIL66/21ail6666$ python3 prg2candy.py
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
 ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
 ['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
['yes' 'yes' 'no' 'yes']
initialization of specific h and general h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
 ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
For Loop Starts
If instance is Positive
steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
 ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

For Loop Starts
If instance is Positive
steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
 ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
```

```

For Loop Starts
If instance is Negative
    steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['same']]

For Loop Starts
If instance is Positive
    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.

Objective	To demonstrate the working of the decision tree based ID3 algorithm.
Dataset	Tennis data set: This data set contain the set of example days on which playing of tennis is possible or not. Based on attribute Sky, AirTemp, Humidity, Wind, Water and Forecast.
ML algorithm	Supervised Learning-Decision Tree algorithm
Description	Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

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

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

def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))

    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
        for y in range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1

    for x in range(len(attr)):
        dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
        pos=0
        for y in range(r):
            if data[y][col]==attr[x]:
                if delete:
                    del data[y][col]
```

```
        dic[attr[x]][pos]=data[y]
        pos+=1
    return attr,dic
```

```
def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0

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

```
def compute_gain(data,col):
    attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])
    for x in range(len(attr)):
        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
        total_entropy-=ratio[x]*entropies[x]
    return total_entropy
```

```
def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol))==1:
        node=Node("")
        node.answer=lastcol[0]
        return node

    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
        gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]

    attr,dic=subtables(data,split,delete=True)
```

```

for x in range(len(attr)):
    child=build_tree(dic[attr[x]],fea)
    node.children.append((attr[x],child))
return node

def print_tree(node,level):
    if node.answer!="":
        print(" "*level,node.answer)
        return

    print(" "*level,node.attribute)
    for value,n in node.children:
        print(" "*(level+1),value)
        print_tree(n,level+2)

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

dataset,features=load_csv("id3.csv")
node1=build_tree(dataset,features)
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
testdata,features=load_csv("id3_test.csv")
for xtest in testdata:
    print("The test instance:",xtest)
    print("The label for test instance:",end=" ")
    classify(node1,xtest,features)

```

DATASET:

Outlook	Temperat	Humidity	Wind	Target
sunny	hot	high	weak	yes
sunny	hot	high	strong	yes
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	yes
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

data3.csv

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

data3_test.csv

Output:

```
(base) cbitccp@cbitccp-H410M-H-V3:~/Documents$ python prgm4.py
The decision tree for the dataset using ID3 algorithm is
Temperature
  hot
  yes
  cool
  yes
  mild
  Humidity
    normal
    yes
    high
    Outlook
      overcast
      yes
      sunny
      no
      rain
      Wind
        weak
        yes
        strong
        no
The test instance : ['rain', 'cool', 'normal', 'strong']
The predicted label : yes
The test instance : ['sunny', 'mild', 'normal', 'strong']
The predicted label : yes
(base) cbitccp@cbitccp-H410M-H-V3:~/Documents$
```

Program: 4

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

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

Objective	To build an artificial neural network using the backpropagation algorithm.
Dataset	Data stored as a list having two features- number of hours slept, number of hours studied with the test score being the class label
ML algorithm	Supervised Learning –Backpropagation algorithm
Description	The neural network using back propagation will model a single hidden layer with three inputs and one output. The network will be predicting the score of an exam based on the inputs of number of hours studied and the number of hours slept the day before. The test score is the output.

```
import numpy as np
X = np.array([2, 9], [1, 5], [3, 6]), dtype=float) # two inputs [sleep,study]
y = np.array([92], [86], [89]), dtype=float) # one output [Expected % in Exams]
X = X/np.amax(X,axis=0) # maximum of X array longitudinally
y = y/100

#Sigmoid Function
def sigmoid(x):
    return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)

#Variable initialization
epoch=5000 #Setting training iterations
lr=0.1 #Setting learning rate
inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer

#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons)) #weight of the link from input
node to hidden node
bh=np.random.uniform(size=(1,hiddenlayer_neurons)) # bias of the link from input node to hidden node
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons)) #weight of the link from hidden
node to output node
bout=np.random.uniform(size=(1,output_neurons)) #bias of the link from hidden node to output node

#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):
```



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

#Backpropagation
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO* outgrad
EH = d_output.dot(wout.T)

#how much hidden layer weights contributed to error
hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad

# dotproduct of nextlayererror and currentlayerop
wout += hlayer_act.T.dot(d_output) *lr
wh += X.T.dot(d_hiddenlayer) *lr

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

Output:

```
● cbit@cbit-H410M-H-V3:~/MACHINE LEARNING LAB/21AIL66/21ail6666$ python3 prg4ann.py
Input:
[[0.66666667 1.          ]
 [0.33333333 0.55555556]
 [1.          0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
[[0.92337029]
 [0.9173532 ]
 [0.92589065]]
```

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 datasets.

Objective	To implement a classification model for a sample training dataset and computing the accuracy of the classifier for test data.
Dataset	Pima Indian Diabetes dataset stored as .CSV .The attributes are Number of times pregnant, Plasma glucose concentration, Blood Pressure, Triceps skin fold thickness, serum insulin, Body mass index, Diabetes pedigree function, Age
ML algorithm	Supervised Learning -Naïve Bayes Algorithm
Description	The Naïve Bayes classifier is a probabilistic classifier that is based on Bayes Theorem. The algorithm builds a model assuming that the attributes in the dataset are independent of each other.

```
import csv
import random
import math

def loadcsv(filename):
    lines = csv.reader(open(filename, "r"));
    dataset = list(lines)
    for i in range(len(dataset)):
        #converting strings into numbers for processing
        dataset[i] = [float(x) for x in dataset[i]]

    return dataset

def splitdataset(dataset, splitratio):
    #67% training size
    trainsize = int(len(dataset) * splitratio);
    trainset = []
    copy = list(dataset);
    while len(trainset) < trainsize:
        #generate indices for the dataset list randomly to pick ele for training data
        index = random.randrange(len(copy));
        trainset.append(copy.pop(index))
    return [trainset, copy]

def separatebyclass(dataset):
    separated = { } #dictionary of classes 1 and 0
    #creates a dictionary of classes 1 and 0 where the values are
    #the instances belonging to each class
    for i in range(len(dataset)):
        vector = dataset[i]
```

```
    if (vector[-1] not in separated):
        separated[vector[-1]] = []
    separated[vector[-1]].append(vector)
    return separated

def mean(numbers):
    return sum(numbers)/float(len(numbers))

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

def summarize(dataset): #creates a dictionary of classes
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)];
    del summaries[-1] #excluding labels +ve or -ve
    return summaries

def summarizebyclass(dataset):
    separated = separatebyclass(dataset);
    #print(separated)
    summaries = { }
    for classvalue, instances in separated.items():
    #for key,value in dic.items()
    #summaries is a dic of tuples(mean,std) for each class value
        summaries[classvalue] = summarize(instances) #summarize is used to cal to mean and std
    return summaries

def calculateprobability(x, mean, stdev):
    exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
    return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

def calculateclassprobabilities(summaries, inputvector):
    probabilities = { } # probabilities contains the all prob of all class of test data
    for classvalue, classsummaries in summaries.items():#class and attribute information as mean and sd
        probabilities[classvalue] = 1
        for i in range(len(classsummaries)):
            mean, stdev = classsummaries[i] #take mean and sd of every attribute for class 0 and 1 sepearely
            x = inputvector[i] #testvector's first attribute
            probabilities[classvalue] *= calculateprobability(x, mean, stdev);#use normal dist
    return probabilities

def predict(summaries, inputvector): #training and test data is passed
    probabilities = calculateclassprobabilities(summaries, inputvector)
    bestLabel, bestProb = None, -1
    for classvalue, probability in probabilities.items():#assigns that class which has he highest prob
        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
    return (correct/float(len(testset))) * 100.0

def main():
    filename = 'naivedata.csv'
    splitratio = 0.67
    dataset = loadcsv(filename);

    trainingset, testset = splitdataset(dataset, splitratio)
    print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingset), len(testset)))
    # prepare model
    summaries = summarizebyclass(trainingset);
    #print(summaries)
    # test model
    predictions = getpredictions(summaries, testset) #find the predictions of test data with the training data
    accuracy = getaccuracy(testset, predictions)
    print('Accuracy of the classifier is : {0}%'.format(accuracy))

main()
```

Dataset:naivedata.csv

```
naivedata.csv > data
1 6,148,72,35,0,33.6,0.627,50,1
2 1,85,66,29,0,26.6,0.351,31,0
3 8,183,64,0,0,23.3,0.672,32,1
4 1,89,66,23,94,28.1,0.167,21,0
5 0,137,40,35,168,43.1,2.288,33,1
6 5,116,74,0,0,25.6,0.201,30,0
7 3,78,50,32,88,31.0,0.248,26,1
8 10,115,0,0,0,35.3,0.134,29,0
9 2,197,70,45,543,30.5,0.158,53,1
10 8,125,96,0,0,0.0,0.232,54,1
11 4,110,92,0,0,37.6,0.191,30,0
12 10,168,74,0,0,38.0,0.537,34,1
13 10,139,80,0,0,27.1,1.441,57,0
14 1,189,60,23,846,30.1,0.398,59,1
15 5,166,72,19,175,25.8,0.587,51,1
16 7,100,0,0,0,30.0,0.484,32,1
17 0,118,84,47,230,45.8,0.551,31,1
18 7,107,74,0,0,29.6,0.254,31,1
19 1,103,30,38,83,43.3,0.183,33,0
20 1,115,70,30,96,34.6,0.529,32,1
21 3,126,88,41,235,39.3,0.704,27,0
22 8,99,84,0,0,35.4,0.388,50,0
23 7,196,90,0,0,39.8,0.451,41,1
24 9,119,80,35,0,29.0,0.263,29,1
25 11,143,94,33,146,36.6,0.254,51,1
```

naivedata.csv

Output:

```
cbit@cbit-H410M-H-V3:~/MACHINE LEARNING LAB/21AIL66/21ail6666$ python3 prg5nbc.py
Split 768 rows into train=514 and test=254 rows
Accuracy of the classifier is : 69.68503937007874%
cbit@cbit-H410M-H-V3:~/MACHINE LEARNING LAB/21AIL66/21ail6666$
```

Program: 6

6. 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.

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianNetwork
from pgmpy.inference import VariableElimination

heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)

#print('Sample instances from the dataset are given below')
#print(heartDisease.head())

#print("\n Attributes and datatypes")
#print(heartDisease.dtypes)

model=BayesianNetwork([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),
('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','chol')])
print("\nLearning CPD using Maximum likelihood estimators")
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

print("\n Inferencing with Bayesian Network:")
HeartDiseasetest_infer = VariableElimination(model)

print("\n 1. Probability of HeartDisease given evidence= restecg")
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)

print("\n 2. Probability of HeartDisease given evidence= cp ")
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

DATASET:**heart.csv**

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heart disease
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
44	1	2	120	263	0	0	173	0	0	1	0	7	0

Output:

```
PS C:\Users\naray\Downloads\MLLFinal\21AIL66> & "C:/Program Files/Python/Python39-64/Python.exe" C:/Program Files/Python/Python39-64/Scripts/python.exe C:/Users/naray/Downloads/MLLFinal/21AIL66/prg6bbn.py
```

```
Learning CPD using Maximum likelihood estimators
```

```
Inferencing with Bayesian Network:
```

```
1. Probability of HeartDisease given evidence= restecg
```

```
+-----+
| heartdisease | phi(heartdisease) |
+-----+
| heartdisease(0) | 0.1016 |
+-----+
| heartdisease(1) | 0.0000 |
+-----+
| heartdisease(2) | 0.2361 |
+-----+
| heartdisease(3) | 0.2017 |
+-----+
| heartdisease(4) | 0.4605 |
+-----+
```

```
2. Probability of HeartDisease given evidence= cp
```

```
+-----+
| heartdisease | phi(heartdisease) |
+-----+
| heartdisease(0) | 0.3742 |
+-----+
| heartdisease(1) | 0.2018 |
+-----+
| heartdisease(2) | 0.1375 |
+-----+
| heartdisease(3) | 0.1541 |
+-----+
| heartdisease(4) | 0.1323 |
+-----+
```

```
PS C:\Users\naray\Downloads\MLLFinal\21AIL66>
```

Program 7

Apply EM algorithm to cluster a set of data stored in a .csv file. Use the same data set for clustering 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.

Objective	To group a set of unlabelled data into similar classes/clusters and label them and to compare the quality of algorithm.
Dataset	Delivery fleet driver dataset Data set in .csv file with features “Driver_ID”, “distance_feature”, ”speeding_feature” having more than 20 instances
ML algorithm	EM algorithm, K means algorithm – Unsupervised clustering
Packages	Scikit learn(sklearn),pandas
Description	EM algorithm – soft clustering - can be used for variable whose value is never directly observed, provided the general probability distribution governing these variable is known. EM algorithm can be used to train Bayesian belief networks as well as radial basis function network. K-Means – Hard Clustering - to find groups in the data, with the number of groups represented by the variable K . The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = KMeans(n_clusters=3)
model.fit(X)

# Plot the Original Classifications
plt.figure(figsize=(8,8))
colormap = np.array(['red', 'lime', 'black'])
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```



```
# Plot the Models Classifications
plt.figure(figsize=(8,8))
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrix of K-Mean: ',sm.confusion_matrix(y, model.labels_))

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)

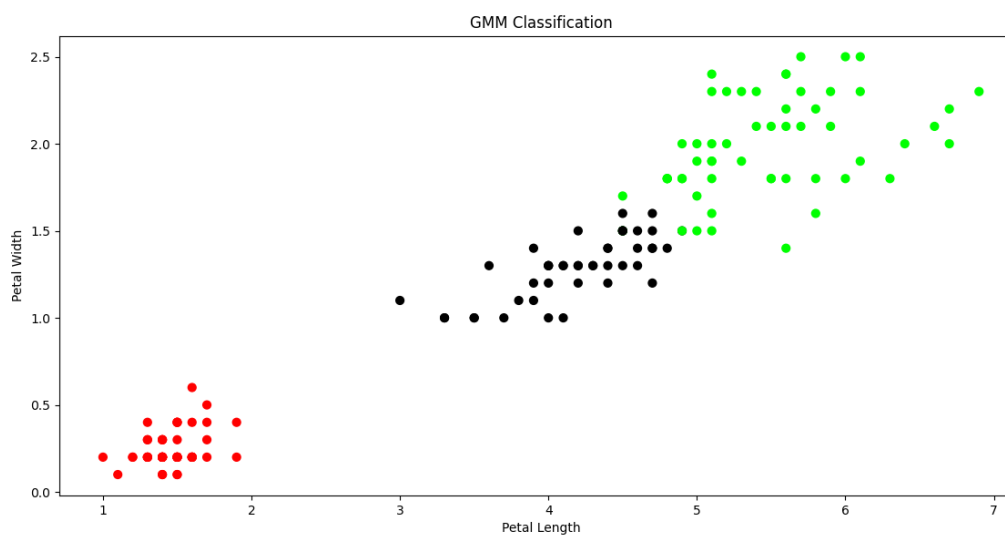
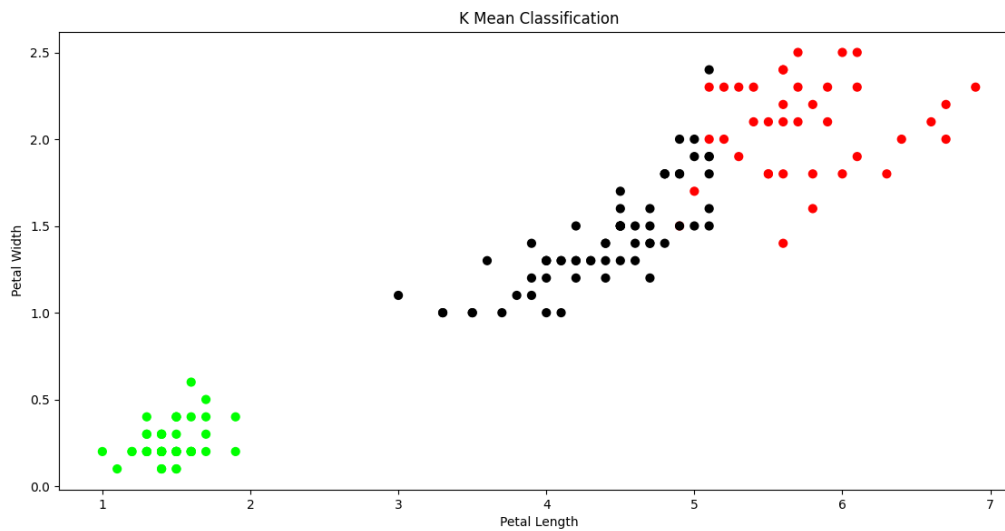
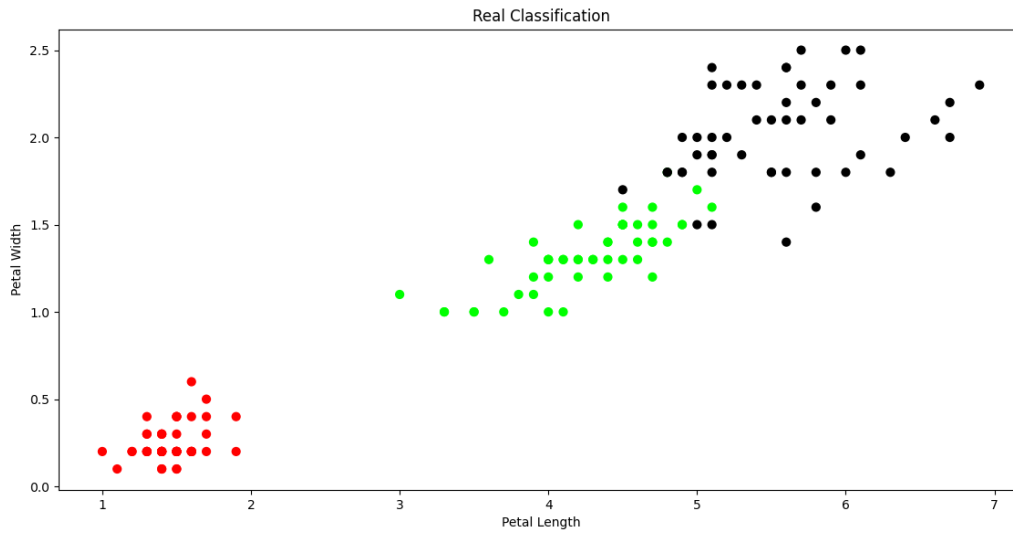
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)

y_gmm = gmm.predict(xs)

# Plot the GMM Classification
plt.figure(figsize=(8,8))
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_gmm))
```

Output:

```
PS C:\Users\naray\Downloads\MLLFinal\21AIL66> & "C:/Program Files/Python312/python.exe"
c:/Users/naray/Downloads/MLLFinal/21AIL66/prg7em_kMeans.py
The accuracy score of K-Mean: 0.44666666666666666
The Confusion matrix of K-Mean: [[50  0  0]
 [ 0  3 47]
 [ 0 36 14]]
The accuracy score of EM: 0.03333333333333333
The Confusion matrix of EM: [[ 0  0 50]
 [45  5  0]
 [ 0 50  0]]
PS C:\Users\naray\Downloads\MLLFinal\21AIL66> □
```

Plots:

Program: 8

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

```

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets

iris=datasets.load_iris()

x = iris.data
y = iris.target
"""
print('sepal-length', 'sepal-width', 'petal-length', 'petal-width')
print(x)
print('class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica')
print(y)
"""

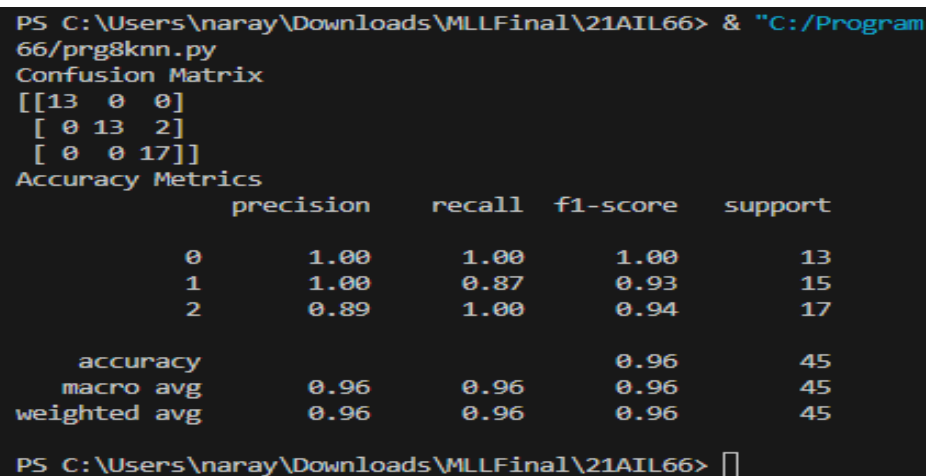
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)

#To Training the model and Nearest neighbors K=5
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)

#To make predictions on our test data
y_pred=classifier.predict(x_test)

print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))

```

Output:


```

PS C:\Users\naray\Downloads\MLLFinal\21AIL66> & "C:/Program
66/prg8knn.py
Confusion Matrix
[[13  0  0]
 [ 0 13  2]
 [ 0  0 17]]
Accuracy Metrics
              precision    recall  f1-score   support

      0         1.00        1.00        1.00        13
      1         1.00        0.87        0.93        15
      2         0.89        1.00        0.94        17

 accuracy         0.96
 macro avg        0.96
weighted avg        0.96
PS C:\Users\naray\Downloads\MLLFinal\21AIL66> 

```

Program: 9

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

```
import numpy as np
import os
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import pearsonr

def kernel(point, xmat, k):
    m, n = xmat.shape
    weights = np.mat(np.eye(m))
    for j in range(m):
        diff = point - xmat[j]
        weights[j, j] = np.exp(diff.dot(diff.T) / (-2.0 * k**2))
    return weights

def local_weight(point, xmat, ymat, k):
    wei = kernel(point, xmat, k)
    mata = xmat.T * wei * xmat
    W = np.linalg.pinv(mata).dot(xmat.T * wei * ymat)
    return W

def local_weight_regression(xmat, ymat, k):
    m, n = xmat.shape
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i].dot(local_weight(xmat[i], xmat, ymat, k))
    return ypred

# Load data points
data = pd.read_csv('tips.csv')
bill = np.array(data.total_bill)
tip = np.array(data.tip)

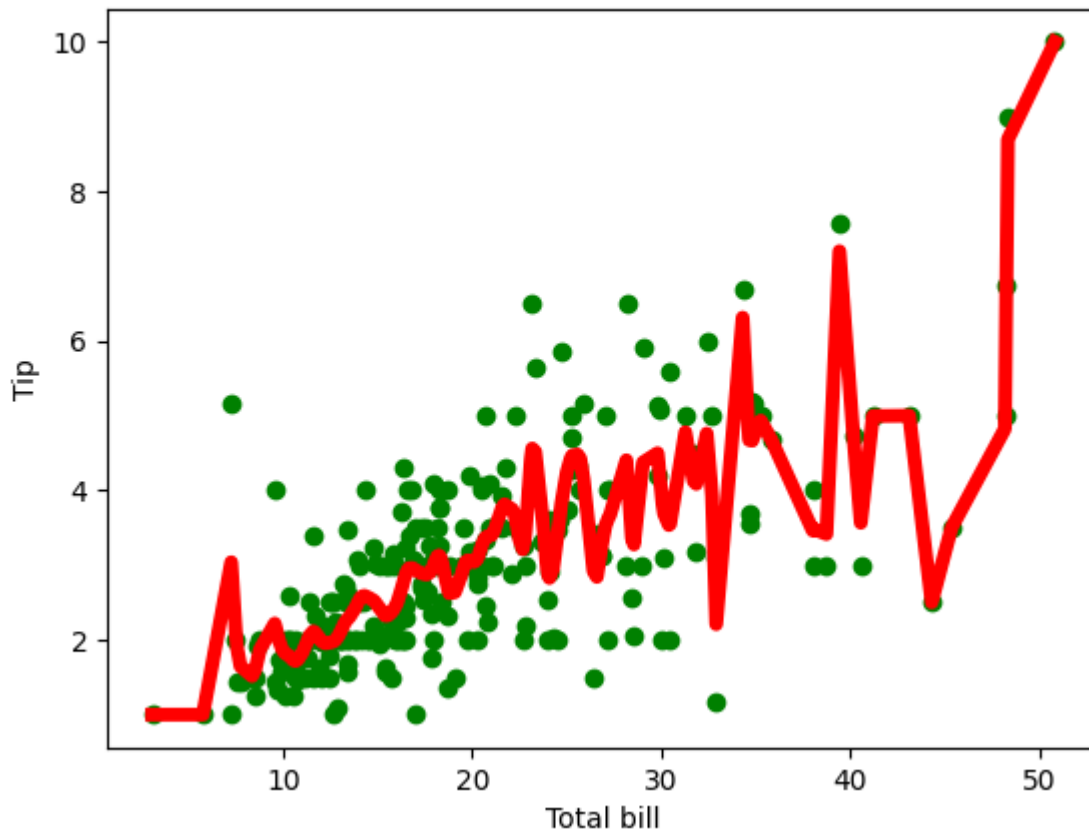
# Prepare and add 1 to bill
mbill = bill.reshape(-1, 1)
one = np.ones((mbill.shape[0], 1))
X = np.hstack((one, mbill)) # Create a stack of bill from ONE

# Set k here
ypred = local_weight_regression(X, tip.reshape(-1, 1), 0.3)
SortIndex = X[:, 1].argsort()
xsort = X[SortIndex]

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(bill, tip, color='green')
ax.plot(xsort[:, 1], ypred[SortIndex], color='red', linewidth=5)
plt.xlabel("Total bill")
```

```
plt.ylabel('Tip')  
plt.show()
```

Output:



Program: 10

Implement and demonstrate the working of SVM algorithm for classification.

```
# Importing necessary libraries

import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the iris dataset

iris = datasets.load_iris()

X = iris.data[:, :2] # Taking only the first two features for
simplicity

y = iris.target

# Splitting the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Feature scaling

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Training the SVM model

svm_classifier = SVC(kernel='linear', random_state=42)

svm_classifier.fit(X_train, y_train)

# Predicting the test set results

y_pred = svm_classifier.predict(X_test)

# Calculating the accuracy of the model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

# Visualizing the decision boundary

def plot_decision_boundary(classifier, X, y):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
```

```
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
np.arange(y_min, y_max, 0.1))
Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolors='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('SVM Decision Boundary')
plt.show()
# Plotting decision boundary
plot_decision_boundary(svm_classifier, X_train, y_train)
```

Output:

Accuracy: 0.7333333333333333

