DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- VIII

MACHINE LEARNING WITH PYTHON LAB
Subject code: LC-CSE-412G

Prepared by:	Checked by:	Approved by:
Prof. Sukrati Chaturvedi	Dr. Ashima Mehta	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

TABLE OF CONTENTS

VISION AND MISSION OF THE INSTITUTE

Vision:

"Dronacharya College of Engineering, Gurugram aims to become an Institution of excellence in imparting quality Outcome Based Education that empowers the young generation with Knowledge, Skills, Research, Aptitude and Ethical values to solve Contemporary Challenging Problems"

Mission:

- M1: Develop a platform for achieving globally acceptable level of intellectual acumen and technological competence.
- M2: Create an inspiring ambience that raises the motivation level for conducting quality research.
- M3: Provide an environment for acquiring ethical values and positive attitude.

VISION AND MISSION OF THE DEPARTMENT

Vision:

"To become a Centre of Excellence in teaching and research in Information Technology for producing skilled professionals having a zeal to serve society"

Mission:

M1: To create an environment where students can be equipped with strong fundamental concepts, programming and problem-solving skills.

M2: To provide an exposure to emerging technologies by providing hands on experience for generating competent professionals.

M3: To promote Research and Development in the frontier areas of Information Technology and encourage students for pursuing higher education

M4: To inculcate in students ethics, professional values, team work and leadership skills.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1: To provide students with a sound knowledge of mathematical, scientific and engineering fundamentals required to solve real world problems.

PEO2: To develop research oriented analytical ability among students and to prepare them for making technical contribution to the society.

PEO3: To develop in students the ability to apply state-of-the–art tools and techniques for designing software products to meet the needs of Industry with due consideration for environment friendly and sustainable development.

PEO4: To prepare students with effective communication skills, professional ethics and managerial skills.

PEO5: To prepare students with the ability to upgrade their skills and knowledge for life-long learning.

PROGRAMME OUTCOMES (POs)

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

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

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

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

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Analyse, identify and clearly define a problem for solving user needs by selecting, creating and evaluating a computer based system through an effective project plan.

PSO2: Design, implement and evaluate processes, components and/or programs using modern techniques, skills and tools of core Information Technologies to effectively integrate secure IT-based solutions into the user environment.

PSO3: Develop impactful IT solutions by using research based knowledge and research methods in the fields of integration, interface issues, security & assurance and implementation.

UNIVERSITY SYLLABUS

1.The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the probability that it is Friday is 20 %. What is theprobability that a student is absent given that today is Friday?
Apply Baye's rule in python to get the result. (Ans: 15%)

2. Extract the data from database using python

3. Implement k-nearest neighbours classification using python

4. Implement linear regression using python

5.Implement K-Means_Clustering using python

6. Implement Naive Bayes Theorem to Classify the English Text using python

7. Implement an algorithm to demonstrate the significance of Genetic Algorithm in python

8. Implement an algorithm to demonstrate Back Propagation Algorithm in python

9. Implementing FIND-S algorithm using python

10. Implementing Candidate Elimination algorithm using python

COURSE OUTCOMES (COs)

Upon successful completion of the course, the students will:

1. To learn the basic concepts of machine learning and types of machine learning.
2. To design and analyze various machine learning algorithms and techniques with a modern outlook focusing on recent advances.
3. Explore supervised and unsupervised learning paradigms of machine learning.

CO-PO Mapping:

	PSO1	PSO2	PSO3	PSO4	PSO5	PSO6	PSO7	PSO8	PSO9	PSO10	PSO11	PSO12
C412.1	3	3	2	2	2	1	-	1	-	1	1	3
C412.2	3	3	3	2	3	1	-	1	-	1	1	3
C412.3	3	3	2	1	2	1	-	1	-	1	1	3

CO-PSO Mapping:

	PSO1	PSO2	PSO3
C412.1	3	2	1
C412.2	3	3	2
C412.3	3	2	3

COURSE OVERVIEW

Machine Learning (ML) is basically that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do. In simple words, ML is a type of artificial intelligence that extract patterns out of raw data by using an algorithm or method. The key focus of ML is to allow computer systems to learn from experience without being explicitly programmed or human intervention.

LIST OF EXPERIMENTS MAPPED WITH COs

DOs and DON'Ts

DOs

1. Login-on with your username and password.

2. Log off the Computer every time when you leave the Lab.

3. Arrange your chair properly when you are leaving the lab.

4. Put your bags in the designated area.

5. Ask permission to print.

DON'Ts

1. Do not share your username and password.

2. Do not remove or disconnect cables or hardware parts.

3. Do not personalize the computer setting.

4. Do not run programs that continue to execute after you log off.

5. Do not download or install any programs, games or music on computer in Lab.

6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.

7. No Internet gaming activities allowed.

8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

GENERAL SAFETY PRECAUTIONS

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency : 200 (Reception)

Security : 248 (Gate No.1)

GUIDELINES TO STUDENTS FOR REPORT PREPARATION

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows:-

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment

(ii) Pre-experiment work (as given by the faculty)

(iii) Lab assignment questions and their solutions

(iv) Test Cases (if applicable to the course)

(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab record is regularly evaluated.

LAB ASSESSMENT CRITERIA

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	**Exemplary (4)**	**Competent (3)**	**Needs Improvement (2)**	**Poor (1)**
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Modeling	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3:	Able to identify	Able to identify	Is dependent	Unable to

Identification & Removal of errors/ bugs	errors/ bugs and remove them	errors/ bugs and remove them with little bit of guidance	totally on someone for identification of errors/ bugs and their removal	understand the reason for errors/ bugs even after they are explicitly pointed out
AC4:Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5:Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done

	and output	with all variants of input and output	with all variants of input and output	with all variants of input and output

LAB EXPERIMENTS

LAB EXPERIMENT 1

OBJECTIVE:

The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the
probability that it is Friday is 20 %. What is theprobability that a student is absent given that today is Friday?
Apply Baye's rule in python to get the result. (Ans: 15%)

ALGORITHM:
Step 1: Calculate probability for each word in a text and filter the words which have a probability less than threshold
probability. Words with probability less than threshold probability are irrelevant.
Step 2: Then for each word in the dictionary, create a probability of that word being in insincere questions and its
probability insincere questions. Then finding the conditional probability to use in naive Bayes classifier.
Step 3: Prediction using conditional probabilities.
Step 4: End.

PROGRAM:
} PFIA=float(input("Enter probability that it is Friday and that a student is absent="))
PF=float(input(" probability that it is Friday="))
PABF=PFIA / PF
print("probability that a student is absent given that today is Friday using conditional probabilities=",PABF)

OUTPUT:
Enter probability that it is Friday and that a student is absent= 0.03
probability that it is Friday= 0.2
probability that a student is absent given that today is Friday using conditional probabilities=

0.15

LAB EXPERIMENT 2

OBJECTIVE:

Extract the data from database using python

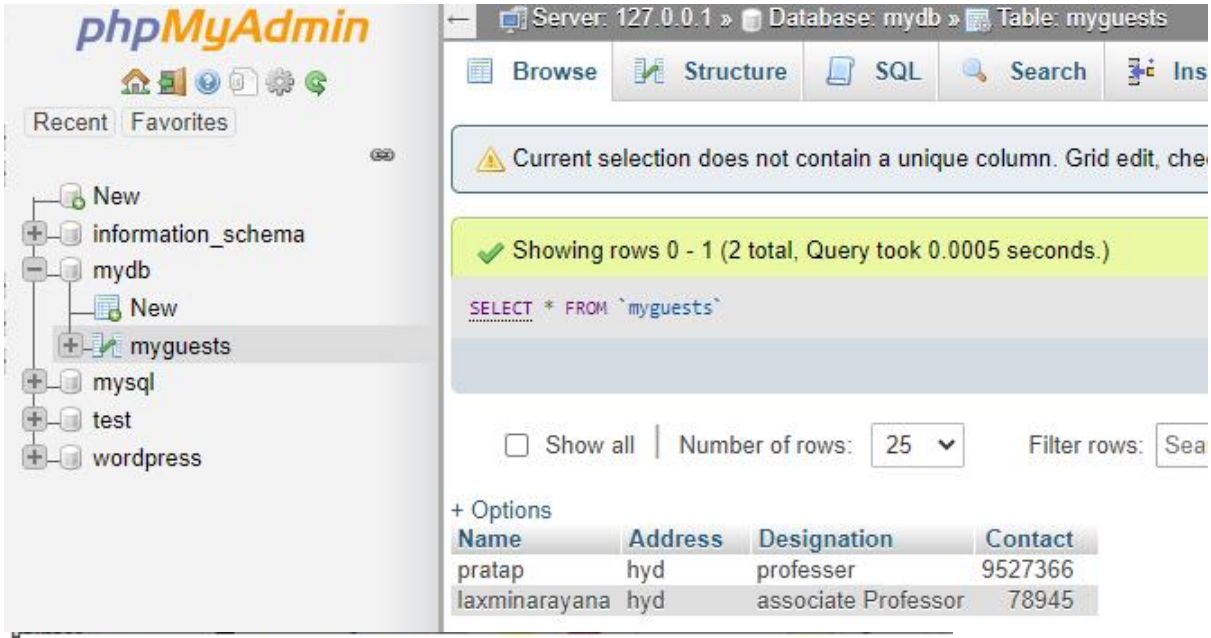
ALGORITHM:
Step 1: Connect to MySQL from Python
Step 2: Define a SQL SELECT Query
Step 3: Get Cursor Object from Connection
Step 4: Execute the SELECT query using execute() method
Step 5: Extract all rows from a result
Step 6: Iterate each row
Step 7: Close the cursor object and database connection object
Step 8: End.

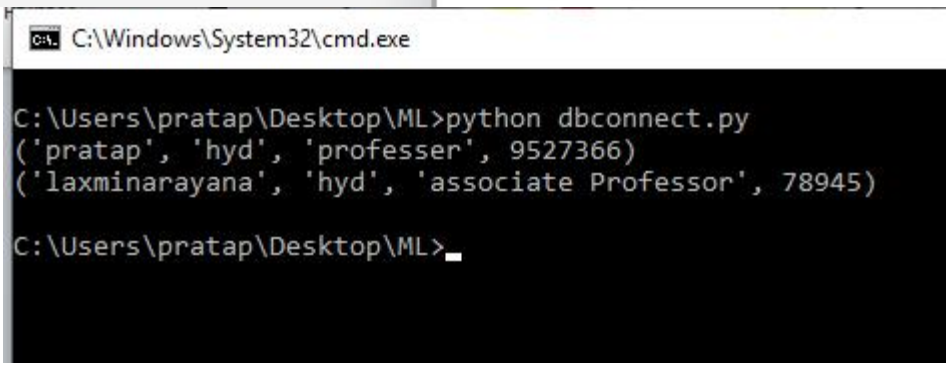
PROCEDURE
CREATING A DATABASE IN MYSQL AS FOLLOWS:
CREATE DATABASE myDB;
SHOW DATABASES;
USE myDB
CREATE TABLE MyGuests (id INT, name VARCHAR(20), email VARCHAR(20));
SHOW TABLES;
INSERT INTO MyGuests (id,name,email) VALUES(1,"sairam","xyz@abc.com");
… **SELECT ***
FROM authors;
We need to install mysql-connector to connect Python with MySQL. You can use the below command to
install this in your system.
pip install mysql-connector-python-rf

PYTHON SOURCE CODE:

```
import mysql.connector
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
database="myDB"


)
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM MyGuests")
myresult = mycursor.fetchall()
for x in myresult:
print(x)
```

OUTPUT:





Extracting data from Excel sheet using Python

Step1: First convert dataset present in excel to CSV file using online resources, then execute following

program:

consider dataset excel consists of 14 input columns and 3 output columns (C1, C2, C3)as follows:

Python Souce Code:

```
import pandas as pd
dataset=pd.read_csv("Mul_Label_Dataset.csv", delimiter=',')
print(dataset) #Print entire dataset
X =
```

```
dataset[['Send','call','DC','IFMSCV','MSCV','BA','MBZ','TxO','RS','CA','AL','IFWL','WWL','
FWL']].values
Y = dataset[['C1','C2','C3']].values
print(Y) #Prints output values
print(X) #Prints intput values
X1 = dataset[['Send','call','DC','IFMSCV','MSCV']].values
print(X1) #Prints first 5 columns of intput values
print(X[0:5]) # Prints only first 5 rows of input values
```

OUTPUT SCREENS:

Excel Format: CSV

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Send	call	DC	IFMSCV	MSCV	BA	MBZ	TxO	RS	CA	AL	IFWL	WWL	FWL	C1	C2	C3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	0	2	0	0	0	0	0	1	0	1
0	0	0	1	2	2	1	0	0	2	0	0	0	0	0	0	1
0	0	0	2	2	2	0	0	0	0	0	2	0	0	1	0	1
2	2	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1
0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1
2	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	1

Format:

```
Send,call,DC,IFMSCV,MSCV,BA,MBZ,TxO,RS,CA,AL,IFWL,WWL,FWL,C1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,2,2,2,2,0,2,0,0,0,0,0,1,0,1
0,0,0,1,2,2,1,0,0,2,0,0,0,0,0,0,1
0,0,0,2,2,2,0,0,0,0,0,2,0,0,1,0,1
2,2,0,0,0,0,0,0,0,0,0,0,2,0,0,0,1
0,2,0,0,0,0,0,0,0,0,0,0,0,2,0,0,1
2,0,0,0,0,0,0,0,2,0,1,0,0,0,0,0,1
```

LAB EXPERIMENT 3

OBJECTIVE:

Implement k-nearest neighbours classification using python

ALGORITHM:

Step 1: Load the data
Step 2: Initialize the value of k
Step 3: For getting the predicted class, iterate from 1 to total number of training data points
i) Calculate the distance between test data and each row of training data. Here we will use Euclidean
distance as our distance metric since it's the most popular method. The other metrics that can be
used are Chebyshev, cosine, etc.
ii) Sort the calculated distances in ascending order based on distance values 3. Get top k rows from the
sorted array
iii) Get the most frequent class of these rows i.e. Get the labels of the selected K entries
iv) Return the predicted class If regression, return the mean of the K labels If classification, return
the mode of the K labels
If regression, return the mean of the K labels
If classification, return the mode of the K labels
Step 4: End.

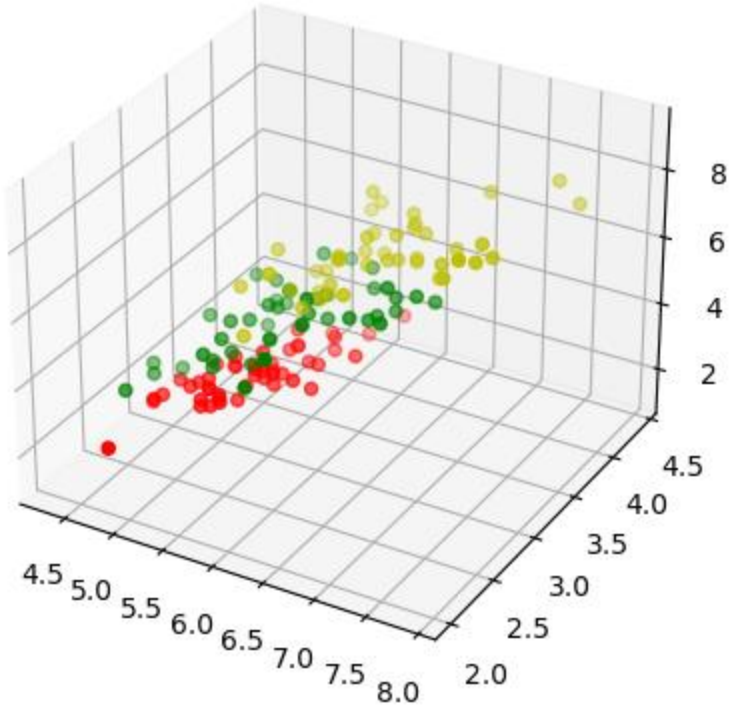
PROGRAM

```
import numpy as np
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
labels = iris.target
for i in [0, 79, 99, 101]:
print(f"index: {i:3}, features: {data[i]}, label: {labels[i]}")
np.random.seed(42)
indices = np.random.permutation(len(data))
n_training_samples = 12
learn_data = data[indices[:-n_training_samples]]
learn_labels = labels[indices[:-n_training_samples]]


test_data = data[indices[-n_training_samples:]]
test_labels = labels[indices[-n_training_samples:]]
print("The first samples of our learn set:")
print(f"{'index':7s}{'data':20s}{'label':3s}")
for i in range(5):
print(f"{i:4d} {learn_data[i]} {learn_labels[i]:3}")
print("The first samples of our test set:")
print(f"{'index':7s}{'data':20s}{'label':3s}")
for i in range(5):
print(f"{i:4d} {learn_data[i]} {learn_labels[i]:3}")
```

```
#The following code is only necessary to visualize the data of our learnset
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
colours = ("r", "b")
X = []
for iclass in range(3):
X.append([[], [], []])
for i in range(len(learn_data)):
if learn_labels[i] == iclass:
X[iclass][0].append(learn_data[i][0])
X[iclass][1].append(learn_data[i][1])
X[iclass][2].append(sum(learn_data[i][2:]))


colours = ("r", "g", "y")
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for iclass in range(3):
ax.scatter(X[iclass][0], X[iclass][1], X[iclass][2], c=colours[iclass])
plt.show()
#--------------------------------------------------
def distance(instance1, instance2):
""" Calculates the Eucledian distance between two instances"""
return np.linalg.norm(np.subtract(instance1, instance2))
def get_neighbors(training_set, labels, test_instance, k, distance):
"""
get_neighors calculates a list of the k nearest neighbors of an instance 'test_instance'.
The function returns a list of k 3-tuples. Each 3-tuples consists of (index, dist, label)
"""
distances = []
for index in range(len(training_set)):
dist = distance(test_instance, training_set[index])
distances.append((training_set[index], dist, labels[index]))
distances.sort(key=lambda x: x[1])
neighbors = distances[:k]
return neighbors


for i in range(5):
neighbors = get_neighbors(learn_data, learn_labels, test_data[i], 3, distance=distance)
print("Index: ",i,'\n',
"Testset Data: ",test_data[i],'\n',
"Testset Label: ",test_labels[i],'\n',
"Neighbors: ",neighbors,'\n')
```

OUTPUT:

```
(base) dohathi@dohathi-Compaq-15-Notebook-PC:~/ML_LAB$ python KNN.py
index:   0, features: [5.1 3.5 1.4 0.2], label: 0
index:  79, features: [5.7 2.6 3.5 1. ], label: 1
index:  99, features: [5.7 2.8 4.1 1.3], label: 1
index: 101, features: [5.8 2.7 5.1 1.9], label: 2
The first samples of our learn set:
index  data                label
   0   [6.1 2.8 4.7 1.2]     1
   1   [5.7 3.8 1.7 0.3]     0
   2   [7.7 2.6 6.9 2.3]     2
   3   [6.  2.9 4.5 1.5]     1
   4   [6.8 2.8 4.8 1.4]     1
The first samples of our test set:
index  data                label
   0   [6.1 2.8 4.7 1.2]     1
   1   [5.7 3.8 1.7 0.3]     0
   2   [7.7 2.6 6.9 2.3]     2
   3   [6.  2.9 4.5 1.5]     1
   4   [6.8 2.8 4.8 1.4]     1
```




```
Index:          2
 Testset Data:   [6.3 2.3 4.4 1.3]
 Testset Label:  1
 Neighbors:       [(array([6.2, 2.2, 4.5, 1.5]), 0.26457513110645864, 1),
 (array([6.3, 2.5, 4.9, 1.5]), 0.574456264653803, 1), (array([6. , 2.2, 4
. , 1. ]), 0.5916079783099617, 1)]

Index:          3
 Testset Data:   [6.4 2.9 4.3 1.3]
 Testset Label:  1
 Neighbors:       [(array([6.2, 2.9, 4.3, 1.3]), 0.20000000000000018, 1),
 (array([6.6, 3. , 4.4, 1.4]), 0.2645751311064587, 1), (array([6.6, 2.9,
4.6, 1.3]), 0.3605551275463984, 1)]

Index:          4
 Testset Data:   [5.6 2.8 4.9 2. ]
 Testset Label:  2
 Neighbors:       [(array([5.8, 2.7, 5.1, 1.9]), 0.31622776601683755, 2),
 (array([5.8, 2.7, 5.1, 1.9]), 0.31622776601683755, 2), (array([5.7, 2.5,
 5. , 2. ]), 0.33166247903553986, 2)]
```

LAB EXPERIMENT 4

OBJECTIVE:

Implement linear regression using python

ALGORITHM:

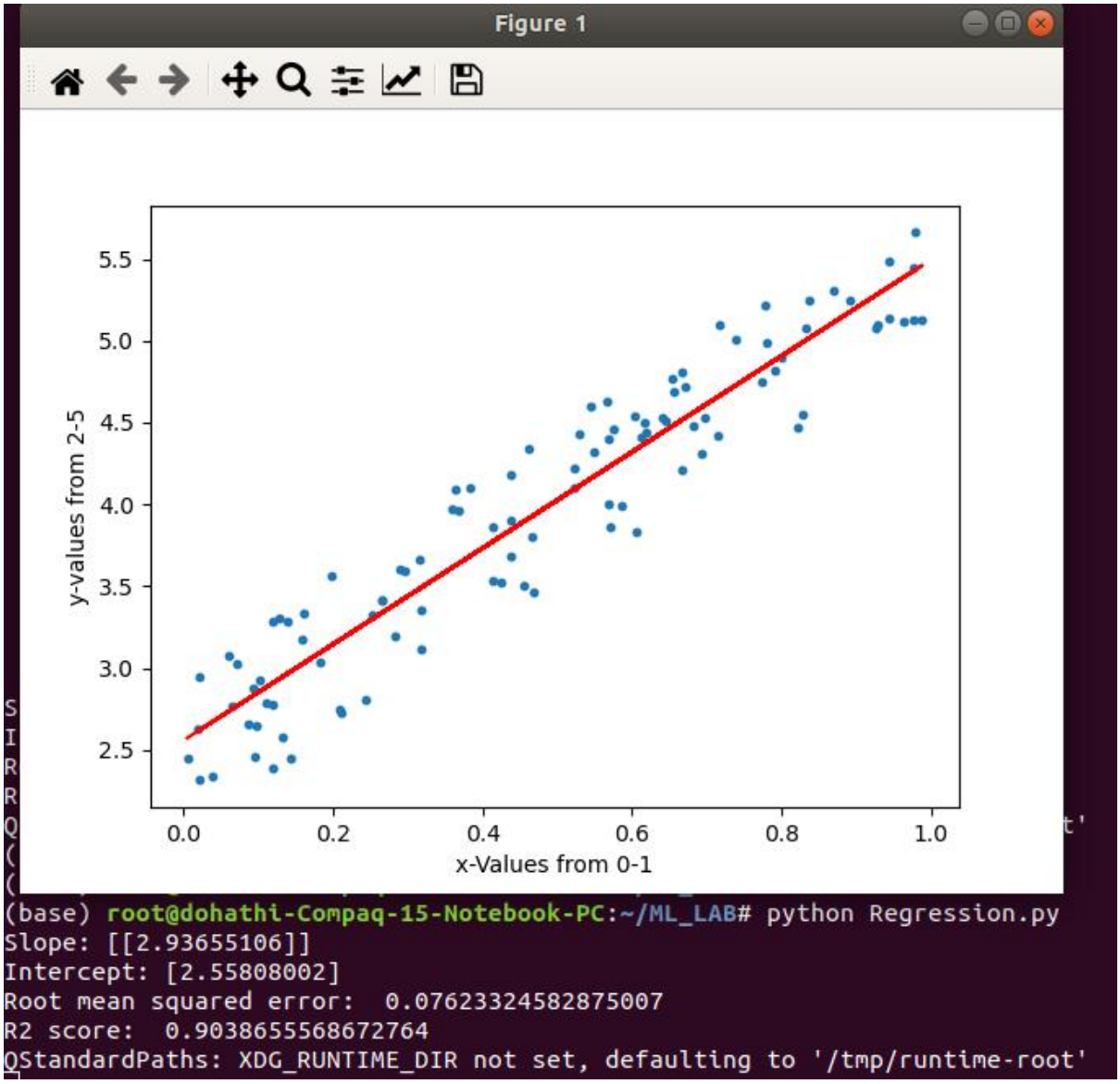
Step 1: Create Database for Linear Regression
Step 2:Finding Hypothesis of Linear Regression
Step 3:Training a Linear Regression model
Step 4:Evaluating the model
Step 5: Scikit-learn implementation
Step 6: End

PROGRAM:

Write a program that implement Queue (its operations)using

```
# Importing Necessary Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# generate random data-set
np.random.seed(0)
x = np.random.rand(100, 1) #Generate a 2-D array with 100 rows, each row containing 1
random numbers:
y = 2 + 3 * x + np.random.rand(100, 1)
regression_model = LinearRegression() # Model initialization
regression_model.fit(x, y) # Fit the data(train the model)
y_predicted = regression_model.predict(x) # Predict
# model evaluation
rmse = mean_squared_error(y, y_predicted)
r2 = r2_score(y, y_predicted)
# printing values
print('Slope:' ,regression_model.coef_)
print('Intercept:', regression_model.intercept_)


print('Root mean squared error: ', rmse)
print('R2 score: ', r2)
# plotting values # data points
plt.scatter(x, y, s=10)
plt.xlabel('x-Values from 0-1')
plt.ylabel('y-values from 2-5')
# predicted values
plt.plot(x, y_predicted, color='r')
plt.show() )
```

OUTPUT:



LAB EXPERIMENT 5

OBJECTIVE:

Implement K-Means_Clustering using python

ALGORITHM:

Step 1: Read the Given data Sample to X
Step 2: Train Dataset with K=5
Step 3: Find optimal number of clusters(k) in a dataset using Elbow method
Step 4: Train Dataset with K=3 (optimal K-Value)
Step 4: Compare results
Step 6: End

PROGRAM:

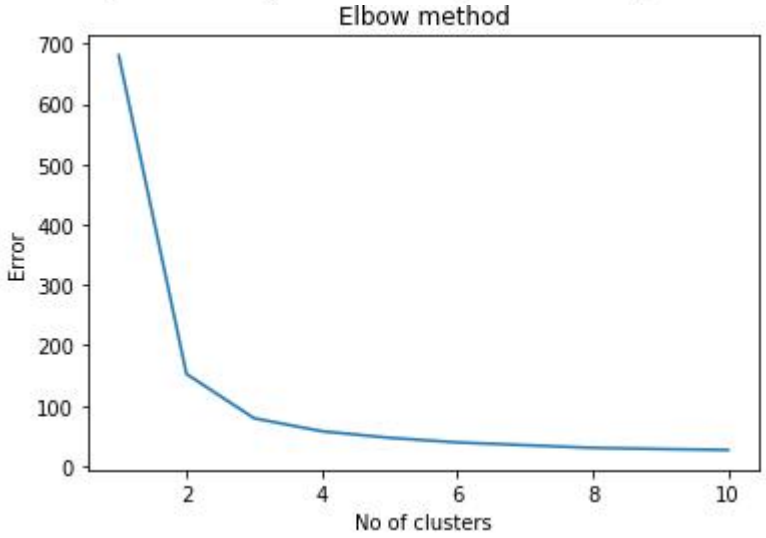
```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets
#Read DataSet
df = datasets.load_iris()
x = df.data
y = df.target
print(x)
print(y)
#Lets try with k=5 initially
kmeans5 = KMeans(n_clusters=5)
y_kmeans5 = kmeans5.fit_predict(x)
print(y_kmeans5)
print(kmeans5.cluster_centers_)
# To find optimal number of clusters(k) in a dataset
Error =[ ]
for i in range(1, 11):
kmeans = KMeans(n_clusters = i).fit(x)
kmeans.fit(x)
Error.append(kmeans.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1, 11), Error)


plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
#Now try with k=3 finally
kmeans3 = KMeans(n_clusters=3)
y_kmeans3 = kmeans3.fit_predict(x)
print(y_kmeans3)
print(kmeans3.cluster_centers_)
```

OUTPUT:

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

```
kmeans5.cluster_centers_
```

```
array([[5.006     , 3.418     , 1.464     , 0.244     ],
       [6.52916667, 3.05833333, 5.50833333, 2.1625    ],
       [5.508     , 2.6       , 3.908     , 1.204     ],
       [7.475     , 3.125     , 6.3       , 2.05      ],
       [6.20769231, 2.85384615, 4.74615385, 1.56410256]])
```




```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
```

```
kmeans3.cluster_centers_
```

```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.418     , 1.464     , 0.244     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

LAB EXPERIMENT 6

OBJECTIVE:

Implement Naive Bayes Theorem to Classify the English Text using python

The Naive Bayes algorithm

Bayes' Theorem

feature matrix

response/target vector

Feature matrix

dependent features **d** **X = (x1,x2,x2, xd).**

Response/target vector **class/group variable** **each row of feature matrix.**

Now the "naïve" conditional independence assumptions come into play: assume that all features in X are mutually independent, conditional on the category y:

Dealing with text data

```
from sklearn.feature_extraction.text import CountVectorizer
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)

print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
'this']

print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

The values 0,1,2, encode the frequency of a word that appeared in the initial text data.

E.g [0 1 1 1 0 0 1 0 1] unique vocabulary ['and', 'document',
'first', 'is', 'one', 'second', 'the', 'third', 'this'],

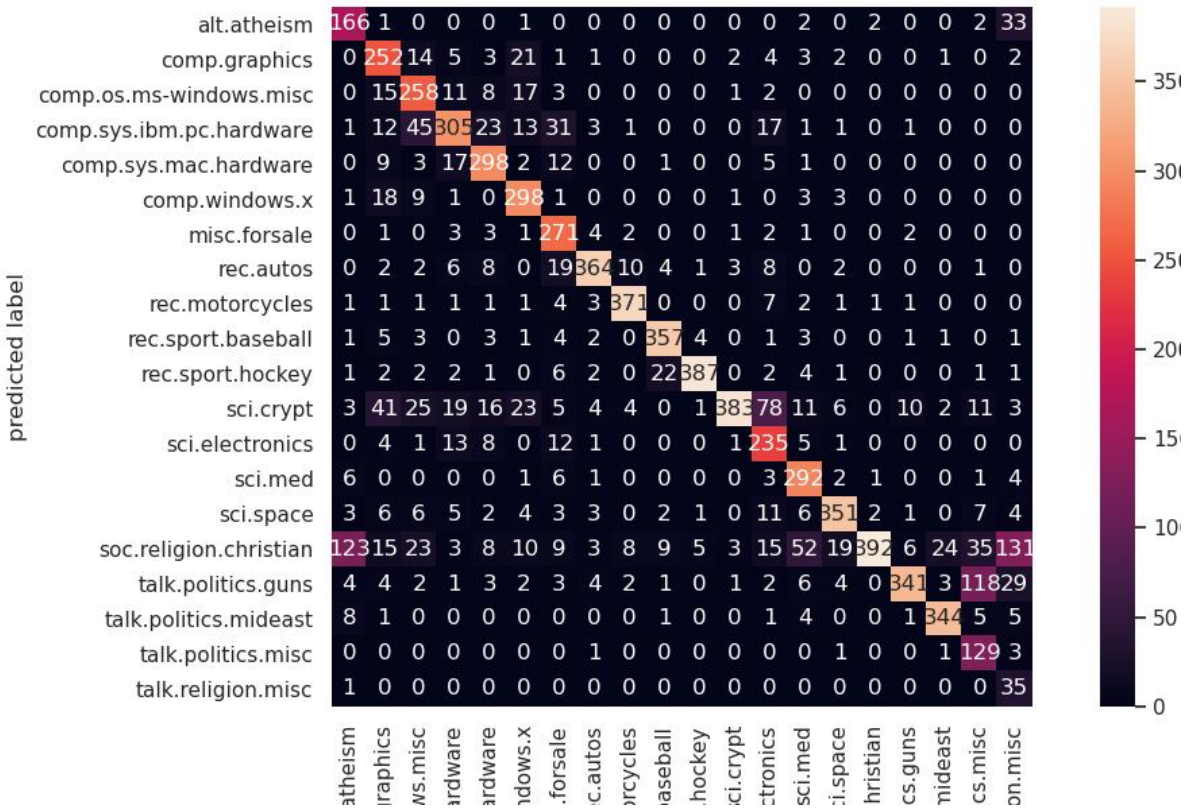
make_pipeline

using the transformed data

Source Code

OUTPUT:

```
L_Programs$ python NB_NaiveBayes.py
Whether: [2 2 0 1 1 1 0 2 2 1 2 0 0 1]
Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Features: [(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0),
(1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
Predicted Value for the input 0:Overcast, 2:Mild: [1]
NAIVE BAYES ENGLISH TEST CLASSIFICATION
We have 20 unique classes
We have 11314 training samples
We have 7532 test samples
```



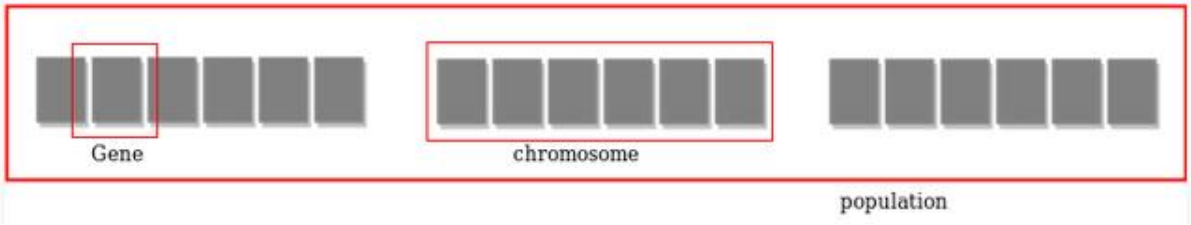
LAB EXPERIMENT 7

OBJECTIVE:

Implement an algorithm to demonstrate the significance of Genetic Algorithm in python

ALGORITHM:

1. Individual in population compete for resources and mate
2. Those individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from "fittest" parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. Thus each successive generation is more suited for their environment.



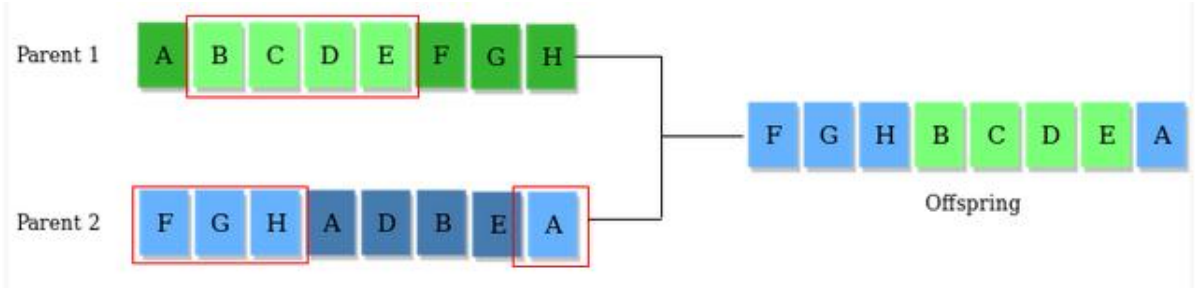
Operators of Genetic Algorithms

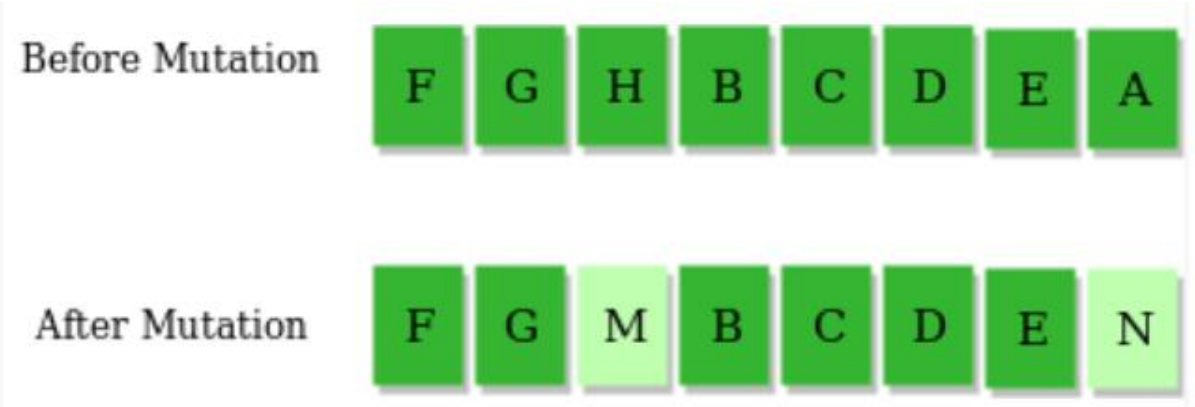
Once the initial generation is created, the algorithm evolve the generation using following operators –

1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass there genes to the successive generations.

2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).

3) Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence.





Given a target string, the goal is to produce target string starting from a random string of the same length. In
the following implementation, following analogies are made –

Characters A-Z, a-z, 0-9 and other special symbols are considered as genes

A string generated by these character is considered as chromosome/solution/Individual

Fitness score is the number of characters which differ from characters in target string at a particular index. So
individual having lower fitness value is given more preference.

Source Code

```
# Python3 program to create target string, starting from
# random string using Genetic Algorithm
import random
# Number of individuals in each generation
POPULATION_SIZE = 100
# Valid genes
GENES = '''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ 1234567890, .-;:_!"#%&/()=?@${[]}'''
# Target string to be generated
TARGET = "I love GeeksforGeeks"
class Individual(object):
'''
Class representing individual in population '''
def __init__(self, chromosome):
self.chromosome = chromosome
self.fitness = self.cal_fitness()
@classmethod
def mutated_genes(self):
```

```
'''
create random genes for mutation
'''
global GENES
gene = random.choice(GENES)
return gene
@classmethod
def create_gnome(self):
'''
create chromosome or string of genes
'''
global TARGET


gnome_len = len(TARGET)
return [self.mutated_genes() for _ in range(gnome_len)]
def mate(self, par2):
''' Perform mating and produce new offspring '''
# chromosome for offspring
child_chromosome = []
for gp1, gp2 in zip(self.chromosome, par2.chromosome):
# random probability
prob = random.random()
# if prob is less than 0.45, insert gene
# from parent 1
if prob < 0.45:
child_chromosome.append(gp1)
# if prob is between 0.45 and 0.90, insert
# gene from parent 2
elif prob < 0.90:
child_chromosome.append(gp2)
# otherwise insert random gene(mutate),
# for maintaining diversity
else:
child_chromosome.append(self.mutated_genes())
# create new Individual(offspring) using
# generated chromosome for offspring
return Individual(child_chromosome)
def cal_fitness(self):
''' Calculate fittness score, it is the number of
characters in string which differ from target string. '''
global TARGET
fitness = 0
for gs, gt in zip(self.chromosome, TARGET):
if gs != gt: fitness+= 1
return fitness
# Driver code
def main():
global POPULATION_SIZE
#current generation
```

```
generation = 1


found = False
population = []
# create initial population
for _ in range(POPULATION_SIZE):
gnome = Individual.create_gnome()
population.append(Individual(gnome))
while not found:
# sort the population in increasing order of fitness score
population = sorted(population, key = lambda x:x.fitness)
# if the individual having lowest fitness score ie.
# 0 then we know that we have reached to the target
# and break the loop
if population[0].fitness <= 0:
found = True
break
# Otherwise generate new offsprings for new generation
new_generation = []
# Perform Elitism, that mean 10% of fittest population
# goes to the next generation
s = int((10*POPULATION_SIZE)/100)
new_generation.extend(population[:s])
# From 50% of fittest population, Individuals
# will mate to produce offspring
s = int((90*POPULATION_SIZE)/100)
for _ in range(s):
parent1 = random.choice(population[:50])
parent2 = random.choice(population[:50])
child = parent1.mate(parent2)
new_generation.append(child)
population = new_generation
print("Generation: {}\tString: {}\tFitness: {}".\
format(generation,
"".join(population[0].chromosome),
population[0].fitness))
generation += 1


print("Generation: {}\tString: {}\tFitness: {}".\
format(generation,
"".join(population[0].chromosome),
population[0].fitness))
if __name__ == '__main__':
main()
```

OUTPUT:

```
Generation: 1    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 2    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 3    String: .#lRWf9k_Ifslw #O$k_    Fitness: 17
Generation: 4    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 5    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 6    String: A#ldW) #lIkslw cVek)    Fitness: 14
Generation: 7    String: A#ldW) #lIkslw cVek)    Fitness: 14
Generation: 8    String: (, o x _x%Rs=, 6Peek3   Fitness: 13
                         .
                         .
                         .
Generation: 29    String: I lope Geeks#o, Geeks   Fitness: 3
Generation: 30    String: I loMe GeeksfoBGeeks    Fitness: 2
Generation: 31    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 32    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 33    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 34    String: I love GeeksforGeeks    Fitness: 0
```

LAB EXPERIMENT 8

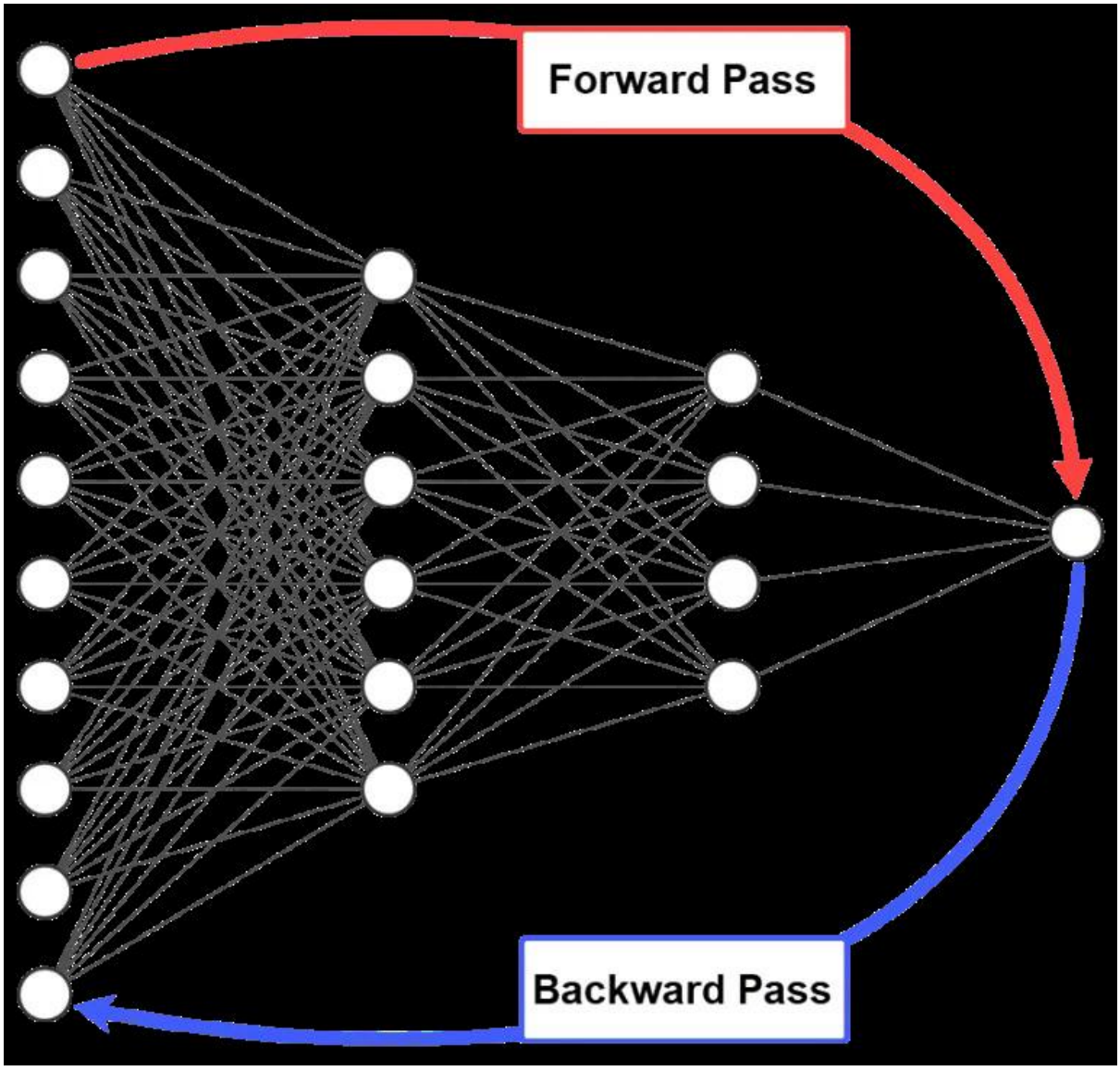
OBJECTIVE:

Implement an algorithm to demonstrate Back Propagation Algorithm in python

ALGORITHM:

It is the most widely used algorithm for training artificial neural networks.
In the simplest scenario, the architecture of a neural network consists of some sequential layers, where the
layer numbered *i* is connected to the layer numbered *i+1*. The layers can be classified into 3 classes:

1. Input
2. Hidden
3. Output



Usually, each neuron in the hidden layer uses an activation function like sigmoid or rectified linear unit
(ReLU). This helps to capture the non-linear relationship between the inputs and their outputs. The neurons in the output layer also use activation functions like sigmoid (for regression) or SoftMax (for
classification).
To train a neural network, there are 2 passes (phases):
Forward
Backward
The forward and backward phases are repeated from some epochs. In each epoch, the following occurs:
1. The inputs are propagated from the input to the output layer.
2. The network error is calculated.
3. The error is propagated from the output layer to the input layer.
Knowing that there's an error, what should we do? We should minimize it. To minimize network error, we
must change something in the network. Remember that the only parameters we can change are the weights
and biases. We can try different weights and biases, and then test our network.

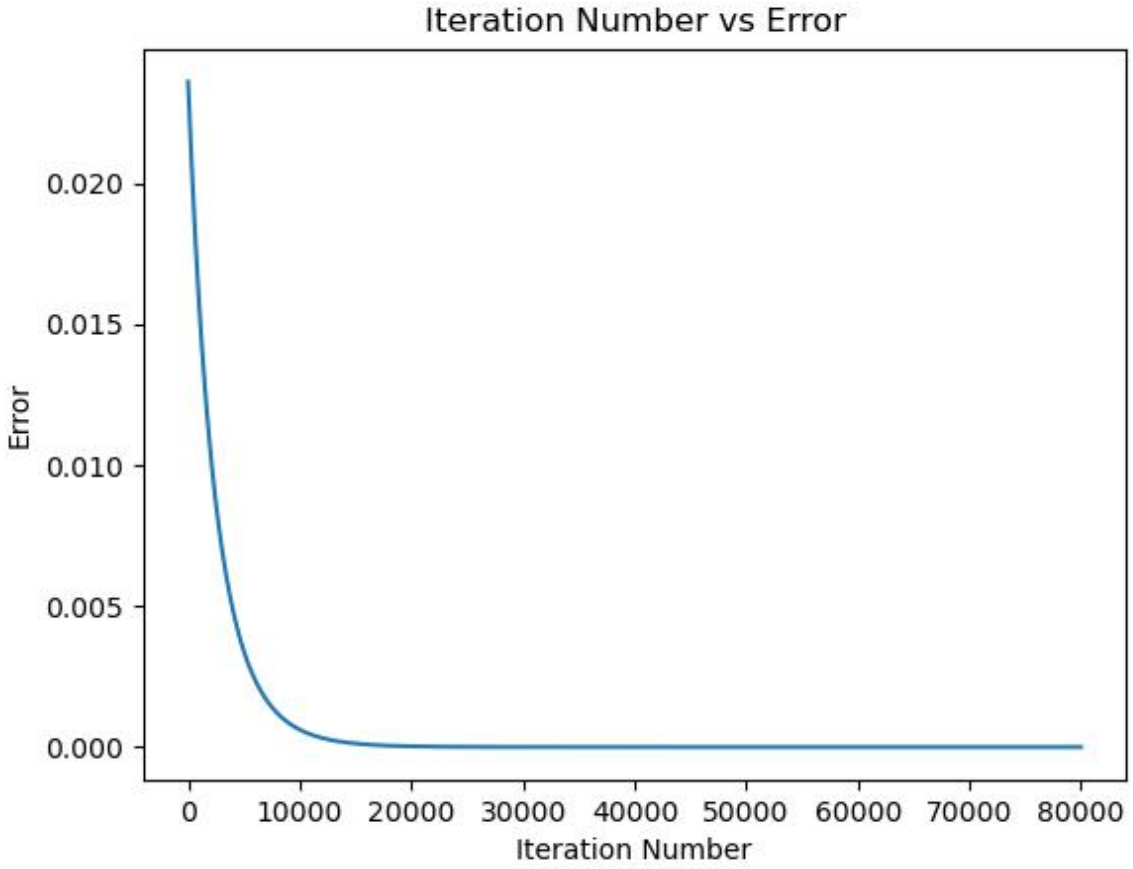
Source Code:

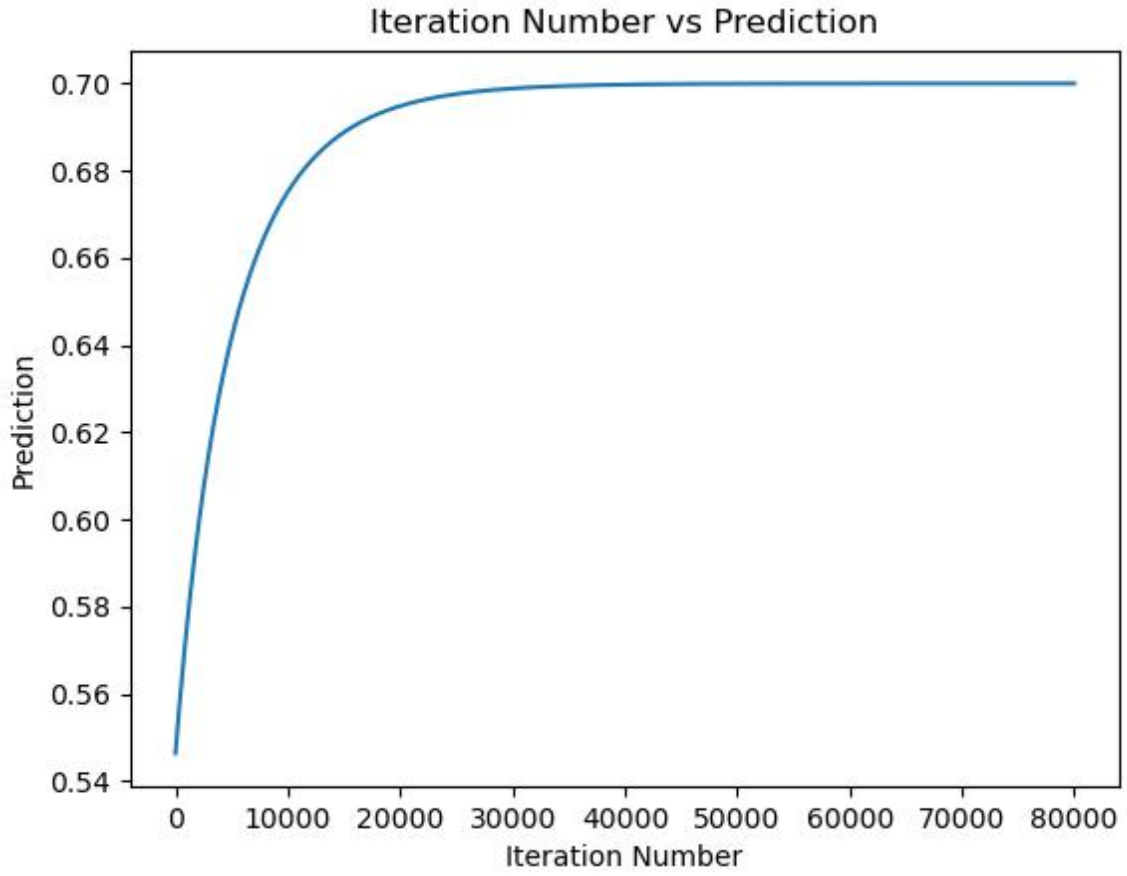
```
import numpy
import matplotlib.pyplot as plt
def sigmoid(sop):
return 1.0/(1+numpy.exp(-1*sop))
def error(predicted, target):
return numpy.power(predicted-target, 2)
def error_predicted_deriv(predicted, target):
return 2*(predicted-target)
def sigmoid_sop_deriv(sop):
return sigmoid(sop)*(1.0-sigmoid(sop))
def sop_w_deriv(x):
return x
def update_w(w, grad, learning_rate):
return w - learning_rate*grad
x1=0.1
x2=0.4
target = 0.7
learning_rate = 0.01
w1=numpy.random.rand()
w2=numpy.random.rand()
print("Initial W : ", w1, w2)
predicted_output = []
network_error = []
old_err = 0
for k in range(80000):
# Forward Pass
y = w1*x1 + w2*x2
predicted = sigmoid(y)
```

```
err = error(predicted, target)
predicted_output.append(predicted)
network_error.append(err)
# Backward Pass


g1 = error_predicted_deriv(predicted, target)
g2 = sigmoid_sop_deriv(y)
g3w1 = sop_w_deriv(x1)
g3w2 = sop_w_deriv(x2)
gradw1 = g3w1*g2*g1
gradw2 = g3w2*g2*g1
w1 = update_w(w1, gradw1, learning_rate)
w2 = update_w(w2, gradw2, learning_rate)
#print(predicted)
plt.figure()
plt.plot(network_error)
plt.title("Iteration Number vs Error")
plt.xlabel("Iteration Number")
plt.ylabel("Error")
plt.show()
plt.figure()
plt.plot(predicted_output)
plt.title("Iteration Number vs Prediction")
plt.xlabel("Iteration Number")
plt.ylabel("Prediction")
plt.show()
```

OUTPUT:

```
Initial W : 0.08698924153243281 0.4532713230157145
```


Iteration Number vs Error
Error
0.020
0.015
0.010
0.005
0.000
0
10000
20000
30000
40000
50000
60000
70000
80000
Iteration Number



LAB EXPERIMENT 9

OBJECTIVE:

Implementing FIND-S algorithm using python

Training Database

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

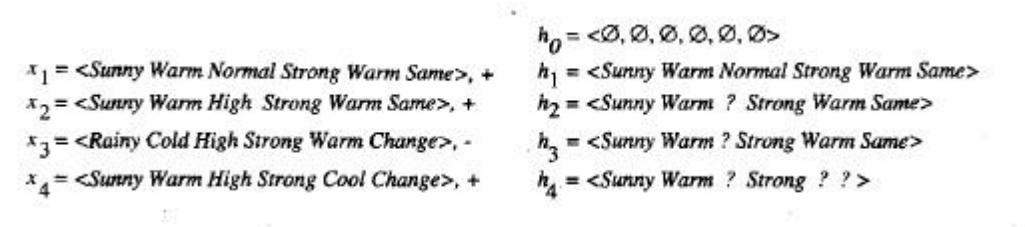
TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Algorithm

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x

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

Hypothesis Construction



Source Code:

```
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] == 'TRUE': #for each positive example only
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:

```
[kln@localhost ML_Programs]$ python FindS.py
[['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'TRUE'], ['Sunny', 'Warm', 'High', 'Strong'
, 'Warm', 'Same', 'TRUE'], ['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'FALSE'], ['Sunny
', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'TRUE']]
('\n The total number of training instances are : ', 4)

 The initial hypothesis is :
['0', '0', '0', '0', '0', '0']
('\n The hypothesis for the training instance 1 is : \n', ['Sunny', 'Warm', 'Normal', 'Strong', 'W
arm', 'Same'])
('\n The hypothesis for the training instance 2 is : \n', ['Sunny', 'Warm', '?', 'Strong', 'Warm',
 'Same'])
('\n The hypothesis for the training instance 3 is : \n', ['Sunny', 'Warm', '?', 'Strong', 'Warm',
 'Same'])
('\n The hypothesis for the training instance 4 is : \n', ['Sunny', 'Warm', '?', 'Strong', '?', '?
'])

 The Maximally specific hypothesis for the training instance is
['Sunny', 'Warm', '?', 'Strong', '?', '?']
```

LAB EXPERIMENT 10

OBJECTIVE:

Implementing Candidate Elimination algorithm using python

Training Database

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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Algorithm

Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do

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

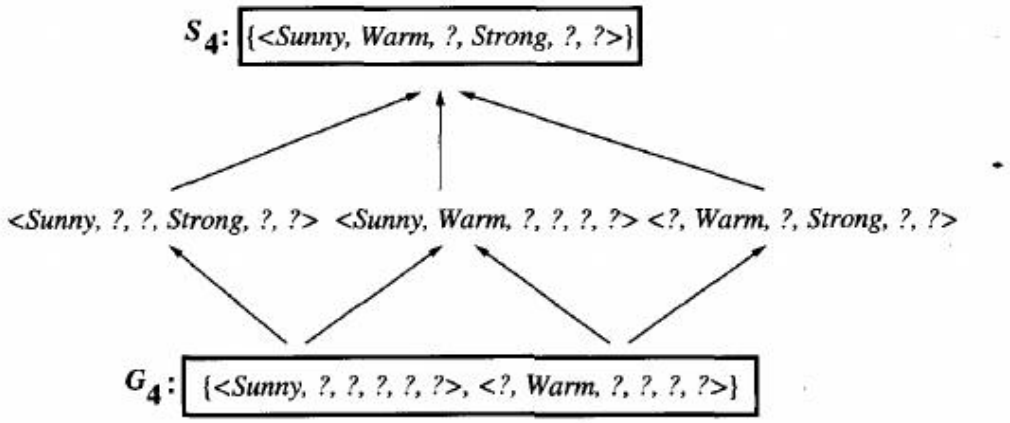


FIGURE 2.7
The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

Source Code:

```
import csv
with open("enjoysport.csv") as f:
csv_file=csv.reader(f)
data=list(csv_file)
print(data)
print("---------------------")
s=data[1][:-1] #extracting one row or instance or record
g=[['?' for i in range(len(s))] for j in range(len(s))]
print(s)
print("---------------------")
print(g)
print("---------------------")
for i in data:
if i[-1]=="TRUE": # For each positive training record or instance
for j in range(len(s)):
if i[j]!=s[j]:
s[j]='?'
g[j][j]='?'
elif i[-1]=="FALSE": # For each negative training record or example
for j in range(len(s)):
if i[j]!=s[j]:
g[j][j]=s[j]


else:
g[j][j]="?"
print("\nSteps of Candidate Elimination Algorithm",data.index(i)+1)
print(s)
print(g)
gh=[]
for i in g:
for j in i:
if j!='?':
gh.append(i)
break
print("\nFinal specific hypothesis:\n",s)
print("\nFinal general hypothesis:\n",gh)
```

OUTPUT:

```
[kln@localhost ML_Programs]$ python CandidateElimination.py
[['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'TRUE'], ['Sunny', 'Warm', 'High', 'Strong'
, 'Warm', 'Same', 'TRUE'], ['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'FALSE'], ['Sunny
', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'TRUE']]
--------------------
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same']
--------------------
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], [
'?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
--------------------
('\nSteps of Candidate Elimination Algorithm', 1)
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], [
'?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
('\nSteps of Candidate Elimination Algorithm', 2)
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], [
'?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
('\nSteps of Candidate Elimination Algorithm', 3)
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'S
ame']]
('\nSteps of Candidate Elimination Algorithm', 4)
['Sunny', 'Warm', '?', 'Strong', '?', '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?
']]
('\nFinal specific hypothesis:\n', ['Sunny', 'Warm', '?', 'Strong', '?', '?'])
('\nFinal general hypothesis:\n', [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?'
, '?']])
```

This lab manual has been updated by

Prof. Sukrati Chaturvedi

(sukrati.chaturvedi@ggnindia.dronacharya.info)

Crosschecked By

HOD CSE

Please spare some time to provide your valuable feedback.