|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
| **Sr. No.** | **List of Practical** | **Date** | **Page No** | **Sign** |
| 1. | Design an Expert system using AIML. | 25/08/23 | 1 |  |
| 2. | Design a bot using AIML. | 31/08/23 | 6 |  |
| 3. | Implement Bayes Theorem using Python | 01/09/23 | 19 |  |
| 4. | Implement Conditional Probability and joint probability using Python | 08/09/23 | 21 |  |
| 5. | Write a program to implement Rule based system. | 15/09/23 | 23 |  |
| 6. | Design a Fuzzy based application using Python / R. | 29/09/23 | 25 |  |
| 7. | Write an application to implement clustering algorithm. | 06/10/23 | 28 |  |
| 8. | Write an application to implement support vector machine algorithm. | 13/10/23 | 30 |  |
| 9. | Simulate artificial neural network model with both feed forward and backpropogation approach. | 20/10/23 | 33 |  |
| 10. | Simulate genetic algorithm with suitable example using Python / R or any other platform. | 27/10/23 | 36 |  |
| 11. | Design an application to simulate language parser. | 03/11/23 | 41 |  |

**Practical 1**

**AIM: Design an Expert system using AIML.**

**Theory:** An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using **both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science,** etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

**basic\_chat.aiml**

<aiml version="1.0.1" encoding="UTF-8">

<!-- basic\_chat.aiml -->

<category>

<pattern>HELLO</pattern>

<template>

Well, hello!

</template>

</category>

<category>

<pattern>WHAT ARE YOU</pattern>

<template>

I'm a bot, silly!

</template>

</category>

<category>

<pattern>MY NAME IS \*</pattern>

<template>

<set name = "username">

<star/>

</set> is the nice name.

</template>

</category>

<category>

<pattern>I LIKE \*</pattern>

<template>

<set name = "liking">

<star/>

</set> is also my favourite.

</template>

</category>

<category>

<pattern>MY DOG NAME IS \*</pattern>

<template>

THAT IS INTERESTING THAT YOU HAVE A DOG NAMED

<set name ="dog">

<star/>

</set> .

</template>

</category>

<category>

<pattern>BYE</pattern>

<template>

Bye!!!

<get name = "username"/> Thanks for talking with me.

</template>

</category>

</aiml>

**Std-startup.xml**

<aiml version="1.0.1" encoding="UTF-8">

<!-- std-startup.xml -->

<!-- Category is an atomic AIML unit -->

<category>

<!-- Pattern to match in user input -->

<!-- If user enters "LOAD AIML B" -->

<pattern>LOAD AIML B</pattern>

<!-- Template is the response to the pattern -->

<!-- This learn an aiml file -->

<template>

<learn>basic\_chat.aiml</learn>

<!-- You can add more aiml files here -->

<!--<learn>more\_aiml.aiml</learn>-->

</template>

</category>

</aiml>

**ChatBot.py**

import aiml

# Create the kernel and learn AIML files

kernel = aiml.Kernel()

kernel.learn("std-startup.xml")

kernel.respond("load aiml b")

# Press CTRL-C to break this loop

while True:

message = input("Enter your message to the bot: ")

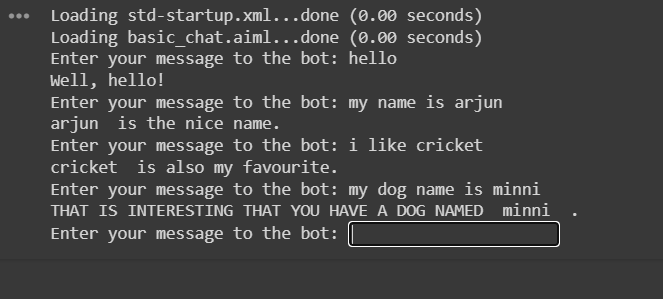
if message == "quit":

break

else:

bot\_response = kernel.respond(message)

print(bot\_response)



**Practical 2**

**AIM: Design a bot using aiml.**

**Theory:** The main aim of the AIML language is to simplify the modelling process of dialogue. It gives access to stimulus response. In addition to that, AIML is an XML-based markup language. AIML works by defining a class object that is responsible for modelling the patterns of conversation. It is the most-used chat-bot language due to its simplicity, ease of learning, ease of implementation and the availability of pre-authored AIML sets. It is a simple word pattern-matcher that generates a response based on a query. The AIML robot responds according to connection between the questions set by the user and knowledge located in AIML files. AIML has its advantages and disadvantages. Some of the advantages are: Easy to implement and learn, user-friendliness and simplicity of the system of dialogue, the use of XML for the formal, computer readable representation. Some of the disadvantages are that the knowledge is presented as an instance of AIML files. If knowledge is created based on data collected from the internet, it will not be automatically updated and must be updated periodically. Original AIML has no extension possible.

**academics.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<meta name="language" content="en"/>

<category>

<pattern>WHERE IS ACADEMICS</pattern>

<template>

link:https://upgcm.ac.in/PO,%20PSO%20and%20CO/M\_\_146

</template>

</category>

<category>

<pattern>Principle's View</pattern>

<template>

Link: https://upgcm.ac.in/Presidents%20Message/M\_\_25

</template>

</category>

<category>

<pattern>CENTRAL LIBRARY</pattern>

<template> Central Library

The Library of Usha Pravin Gandhi College of Arts, Science and Commerce, Vile Parle (West), Mumbai was established in 2003, the same year in which the college established. In the year 2007, the library shifted to its present location in the new college premises. The Library has occupied a spacious area on the second floor and mezzanine floor of the main building. It is entirely air conditioned and is under CCTV surveillance.

The Library which was started with an initial collection of 175 books in the academic year 2003-2004 has now more than 6000 books in its collection. Computerization of the Library is done. It provides various services to its users and conducts activities for the benefit of its users.

To know more please follow the link given below:

Link: https://upgcm.ac.in/Library/M\_\_37

</template>

</category>

<category>

<pattern>ACADEMIC CALENDAR BE</pattern>

<template> You can get the academic calendar in the link given below:

https://upgcm.ac.in/Academic%20Calendar/M\_\_18

</template>

</category>

</aiml>

**Admission.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<meta name="language" content="en"/>

<category>

<pattern> ADMISSIONS</pattern>

<template>

https://upgcm.ac.in/Admissions/M\_\_2</template>

</category>

<category>

<pattern>Training and Placement Cell</pattern>

<template>

https://upgcm.ac.in/Training%20and%20Placement%20Cell/M\_\_130

</template></category>

</aiml>

**department.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<!-- basic\_chat.aiml -->

<meta name="language" content="en"/>

<category>

<pattern>ONE TIME I \*</pattern>

<template>

<random>

<li>Go on.</li>

<li>How old are you?</li>

<li>Be more specific.</li>

<li>I did not know that.</li>

<li>Are you telling the truth?</li>

<li>I don't know what that means.</li>

<li>Try to tell me that another way.</li>

<li>Are you talking about an animal, vegetable or mineral?</li>

<li>What is it?</li>

</random>

</template>

</category>

<category>

<pattern>HELLO</pattern>

<template>

Well, hello!

</template>

</category>

<category>

<pattern>HOW ARE YOU</pattern>

<template>

I'm a bot, silly!

</template>

</category>

<category>

<pattern>UPG</pattern>

<template>

Yes

</template>

</category>

<!--IT-->

<category>

<pattern>BSCIT DEPARTMENT</pattern>

<template>

BSCIT is here

Link: https://upgcm.ac.in/BSc%20IT/M\_\_16

</template>

</category>

<category>

<pattern>MSCIT DEPARTMENT</pattern>

<template>

MSCIT course is here

Link: https://upgcm.ac.in/MSc%20IT/M\_\_17

</template>

</category>

<!--BMS-->

<category>

<pattern>BMS DEPARTMENT</pattern>

<template>

BSCIT is here

Link: https://upgcm.ac.in/BMS/M\_\_14

</template>

</category>

</aiml>

**hi.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<meta name="language" content="en"/>

<category>

<pattern>HI</pattern>

<template>

<random>

<li>Hello there!</li>

<li>Hey</li>

</random>

</template>

</category>

<category>

<pattern>HELLO</pattern>

<template>

<srai>HI</srai>

</template>

</category>

<category>

<pattern>WHAT IS YOUR NAME</pattern>

<template>

You suggest something!

</template>

</category>

<category>

<pattern>WHAT IS YOUR NAME?</pattern>

<template>

You suggest something!

</template>

</category>

<category>

<pattern>LET YOUR NAME BE \*</pattern>

<template>

Okay, <set name = "username"> <star/></set> is a good name!

</template>

</category>

<category>

<pattern>HOW ABOUT \*</pattern>

<template>

Okay, <set name = "botname"> <star/></set> is a good name!

</template>

</category>

<category>

<pattern>MY NAME IS \*</pattern>

<template>

Oh! Nice to meet you<set name = "username"> <star/></set>

</template>

</category>

<category>

<pattern>THANK YOU \*</pattern>

<template>

Your most welcome!

</template>

</category>

<category>

<pattern>THANK YOU</pattern>

<template>

Your most welcome!

</template>

</category>

<category>

<pattern>BYE \*</pattern>

<template>

Goodbye!

</template>

</category>

<category>

<pattern>BYE</pattern>

<template>

Goodbye!

</template>

</category>

</aiml>

**Results.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<meta name="language" content="en"/>

<category>

<pattern>WHEN ARE THE RESULTS</pattern>

<template>

https://upgcm.ac.in/Result-(B.M.M.%20,B.M.S.%20,%20B.Sc.(I.T.),%20B.A.%20,%20M.A.)/M\_\_62

</template>

</category>

<category>

<pattern>DO YOU HAVE ANY CONTACT</pattern>

<template>

Yes. link:https://upgcm.ac.in/Contact%20Us/M\_\_7

</template>

</category>

</aiml>

**UPG.aiml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<aiml version="1.0">

<meta name="language" content="en"/>

<category>

<pattern>What are the official office hours of UPG College</pattern>

<template>

The college office operates from 10 AM to 1:30 PM

</template>

</category>

<category>

<pattern>Can you provide the complete address for UPG College</pattern>

<template>

Usha Pravin Gandhi College of Arts, Science and Commerce, Bhakti Vedanta Swami Marg, JVPD Scheme, Vile Parle (West), Mumbai 400 056, Maharashtra, India.

</template>

</category>

<category>

<pattern>How much is the fees for different undergraduate courses in this college</pattern>

<template>

1st year-

BMS: 17988₹

BAMMC: 17988₹

BSC.IT: 24794₹

BA(FTNMP): 62388₹

2nd year-

BMS: 16968₹

BAMMC: 16968₹

BSC.IT: 24274₹

BA(FTNMP): 51368₹

3rd year-

BMS: 17618₹

BAMMC: 17618₹

BSC.IT: 26424₹

BA(FTNMP): 51618₹

</template>

</category>

<category>

<pattern>How much is the fees for different postgraduate courses in this college</pattern>

<template>

1st year-

M.Sc. IT: INR 40,687

M.A. (EMA): INR 68,087

M.Com.: INR 15,587

2nd year-

M.Sc. IT: INR 39,112

M.A. (EMA): INR 56,912

M.Com.: INR 14,612.

</template>

</category>

<category>

<pattern>Who is the principal of UPG college</pattern>

<template>

The principal of UPG College is Dr. A. Kapoor.

</template>

</category>

</aiml>

**Std-startup.xml**

**<aiml version="1.0">**

**<category>**

**<pattern>LOAD AIML B</pattern>**

**<template>**

**<!-- Load standard AIML set -->**

**<learn>admission.aiml</learn>**

**<learn>academics.aiml</learn>**

**<learn>department.aiml</learn>**

**<learn>result.aiml</learn>**

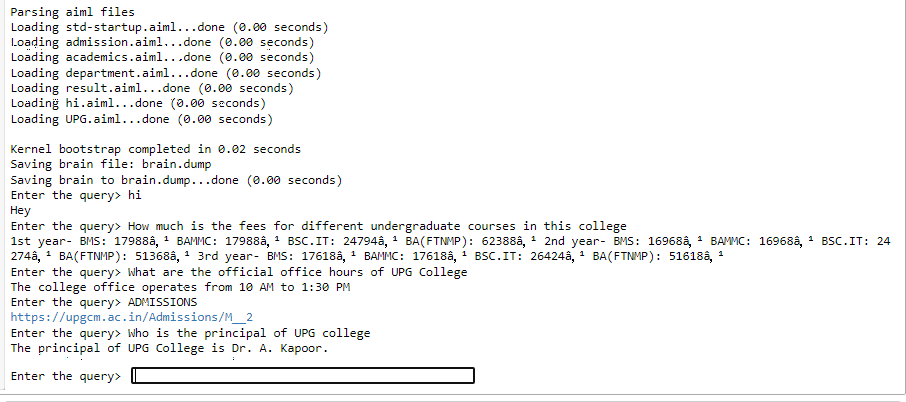
**<learn>hi.aiml</learn>**

**<learn>UPG.aiml</learn>**

**</template>**

**</category>**

**</aiml>**



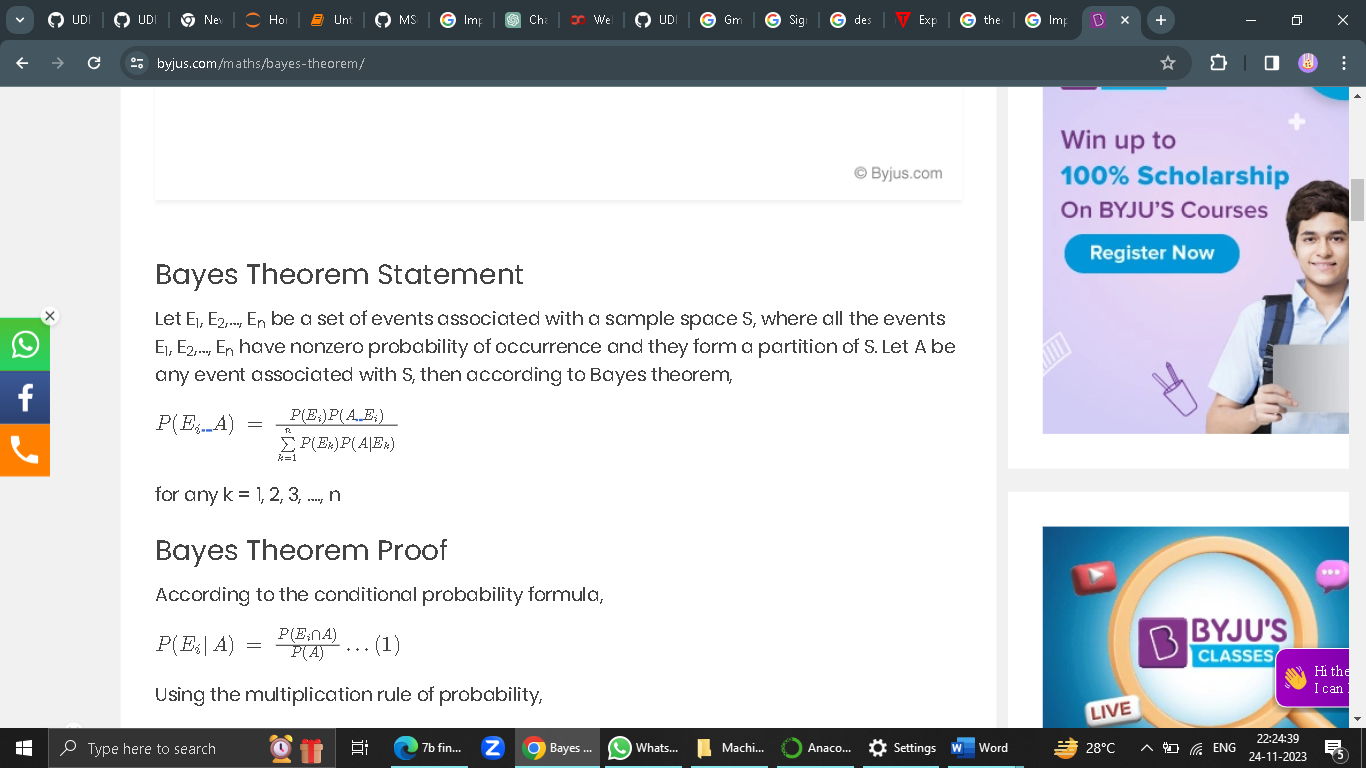
**Practical 3**

**AIM: Implement Bayes Theorem using Python**

**Theory: Bayes’ theorem** describes the probability of occurrence of an event related to any condition. It is also considered for the case of [conditional probability](https://byjus.com/maths/conditional-probability-and-conditional-probability-examples/). Bayes theorem is also known as the formula for the probability of “causes”. For example: if we have to calculate the probability of taking a blue ball from the second bag out of three different bags of balls, where each bag contains three different colour balls viz. red, blue, black. In this case, the probability of occurrence of an event is calculated depending on other conditions is known as conditional probability. In this article, let us discuss the statement and proof for Bayes theorem, its derivation, formula, and many solved examples.

**Bayes Theorem Statement**

Let E1, E2,…, En be a set of events associated with a sample space S, where all the events E1, E2,…, En have nonzero probability of occurrence and they form a partition of S. Let A be any event associated with S, then according to Bayes theorem.



**Code:**

# calculate the probability of cancer patient and diagnostic test

# calculate P(A|B) given P(A), P(B|A), P(B|not A)

def bayes\_theorem(p\_a, p\_b\_given\_a, p\_b\_given\_not\_a):

# calculate P(not A)

not\_a = 1 - p\_a

# calculate P(B)

p\_b = p\_b\_given\_a \* p\_a + p\_b\_given\_not\_a \* not\_a

# calculate P(A|B)

p\_a\_given\_b = (p\_b\_given\_a \* p\_a) / p\_b

return p\_a\_given\_b

# P(A)

p\_a = 0.0002

# P(B|A)

p\_b\_given\_a = 0.85

# P(B|not A)

p\_b\_given\_not\_a = 0.05

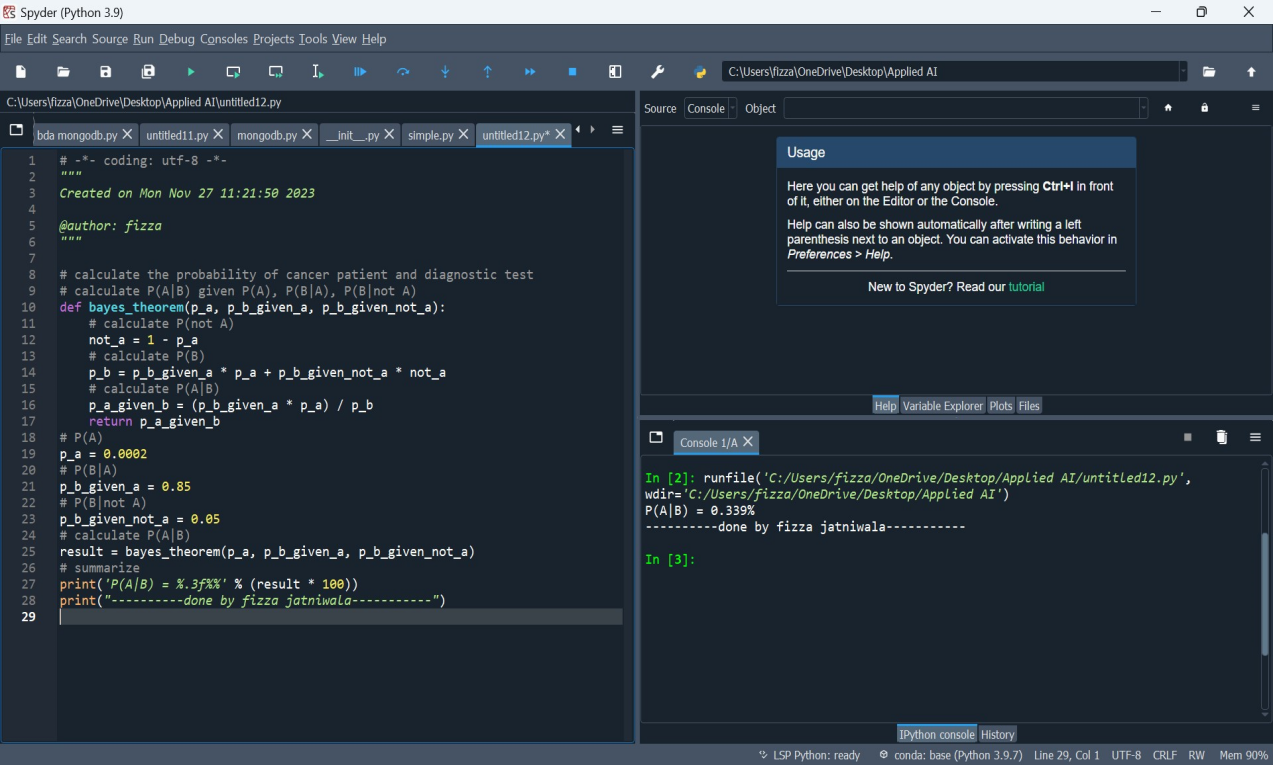
# calculate P(A|B)

result = bayes\_theorem(p\_a, p\_b\_given\_a, p\_b\_given\_not\_a)

# summarize

print('P(A|B) = %.3f%%' % (result \* 100))

Print(“\_\_\_\_\_\_\_\_\_\_\_done by Fizza Jatniwala\_\_”)



**Practical 4**

**AIM: Implement Conditional Probability and joint probability using Python**

**Theory:** Conditional probability is defined as the likelihood of an event or outcome occurring, based on the occurrence of a previous event or outcome. Conditional probability is calculated by multiplying the [probability](https://www.investopedia.com/terms/c/compound-probability.asp) of the preceding event by the updated probability of the succeeding, or conditional, event.

Conditional probability can be contrasted with [unconditional probability](https://www.investopedia.com/terms/u/unconditional_probability.asp). Unconditional probability refers to the likelihood that an event will take place irrespective of whether any other events have taken place or any other conditions are present.

The term joint probability refers to a statistical measure that calculates the likelihood of two events occurring together and at the same point in time. Put simply, a joint probability is the probability of event Y occurring at the same time that event X occurs. In order for joint probability to work, both events must be independent of one another, which means they aren't conditional or don't rely on each other. Joint probabilities can be visualized using Venn diagrams.

**Code:**

import enum, random

class Kid(enum.Enum):

BOY = 0

GIRL = 1

def random\_kid() -> Kid:

return random.choice([Kid.BOY, Kid.GIRL])

both\_girls = 0

older\_girl = 0

either\_girl = 0

random.seed(0)

for \_ in range(10000):

younger = random\_kid()

older = random\_kid()

if older == Kid.GIRL:

older\_girl += 1

if older == Kid.GIRL and younger == Kid.GIRL:

both\_girls += 1

if older == Kid.GIRL or younger == Kid.GIRL:

either\_girl += 1

print("older girl: ", older\_girl)

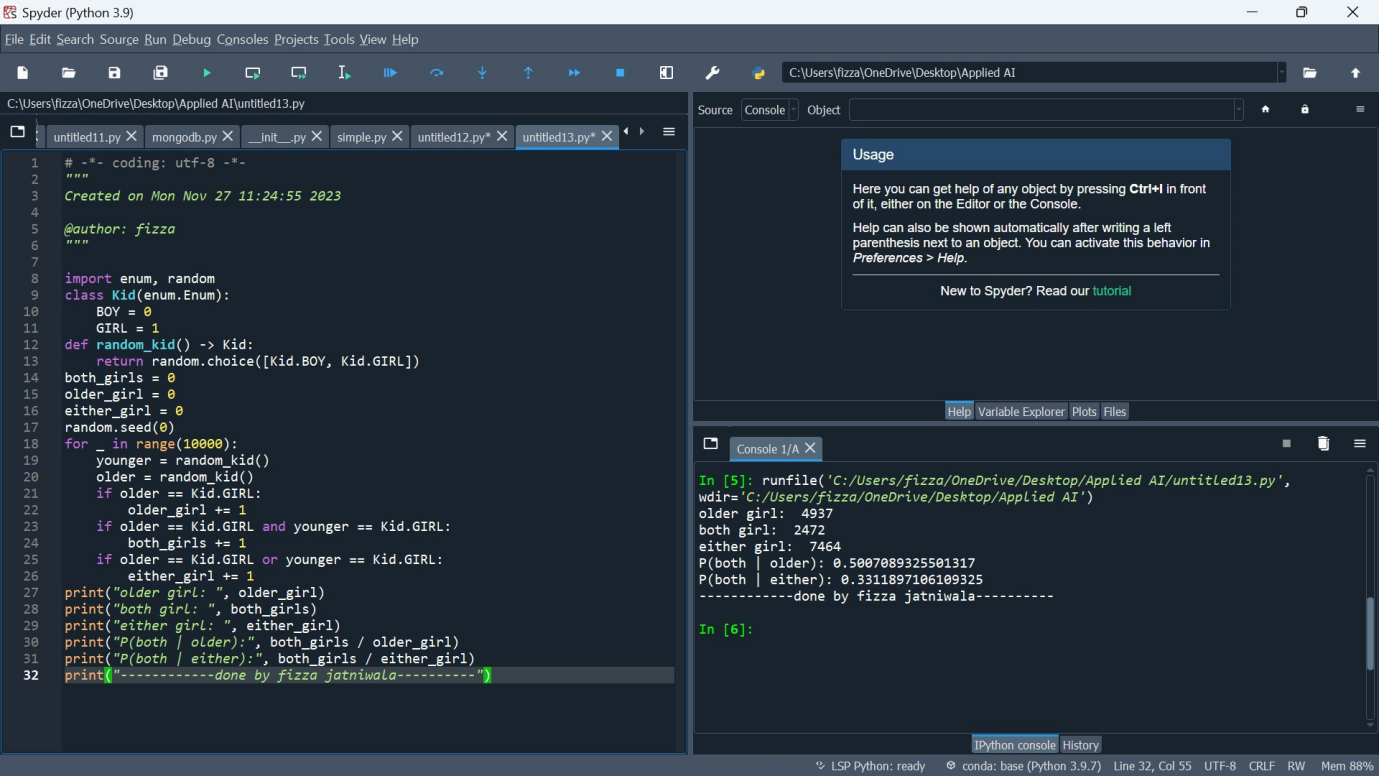
print("both girl: ", both\_girls)

print("either girl: ", either\_girl)

print("P(both | older):", both\_girls / older\_girl)

print("P(both | either):", both\_girls / either\_girl)

Print(“\_\_\_\_\_\_\_\_\_\_done by Fizza Jatniwala\_\_\_\_\_\_\_\_\_\_”)



**Practical 5**

**AIM: Write a program to implement Rule based system.**

**Theory:**  rule-based system in AI is a system that applies human-made rules to store, sort and manipulate data. In doing so, it mimics human intelligence.

Rule based system in AI require a set of facts or source of data, and a set of rules for manipulating that data. These rules are sometimes referred to as ‘If statements’ as they tend to follow the line of ‘IF X happens THEN do Y’.

The steps can be simplified to:

* First comes the data or new business event
* Then comes the analysis: the part where the system conditionally processes the data against its rules
* Then comes any subsequent automated follow-up actions

Code:

father(joe,paul).

father(joe,mary).

father(joe,hope).

mother(jane,paul).

mother(jane,mary).

mother(jane,hope).

male(paul).

male(joe).

male(raphl).

male(X):-father(X,Y).

female(mary).

female(jane).

female(hope).

female(X):-mother(X,Y).

son\_of(X,Y):- father(Y,X),male(X).

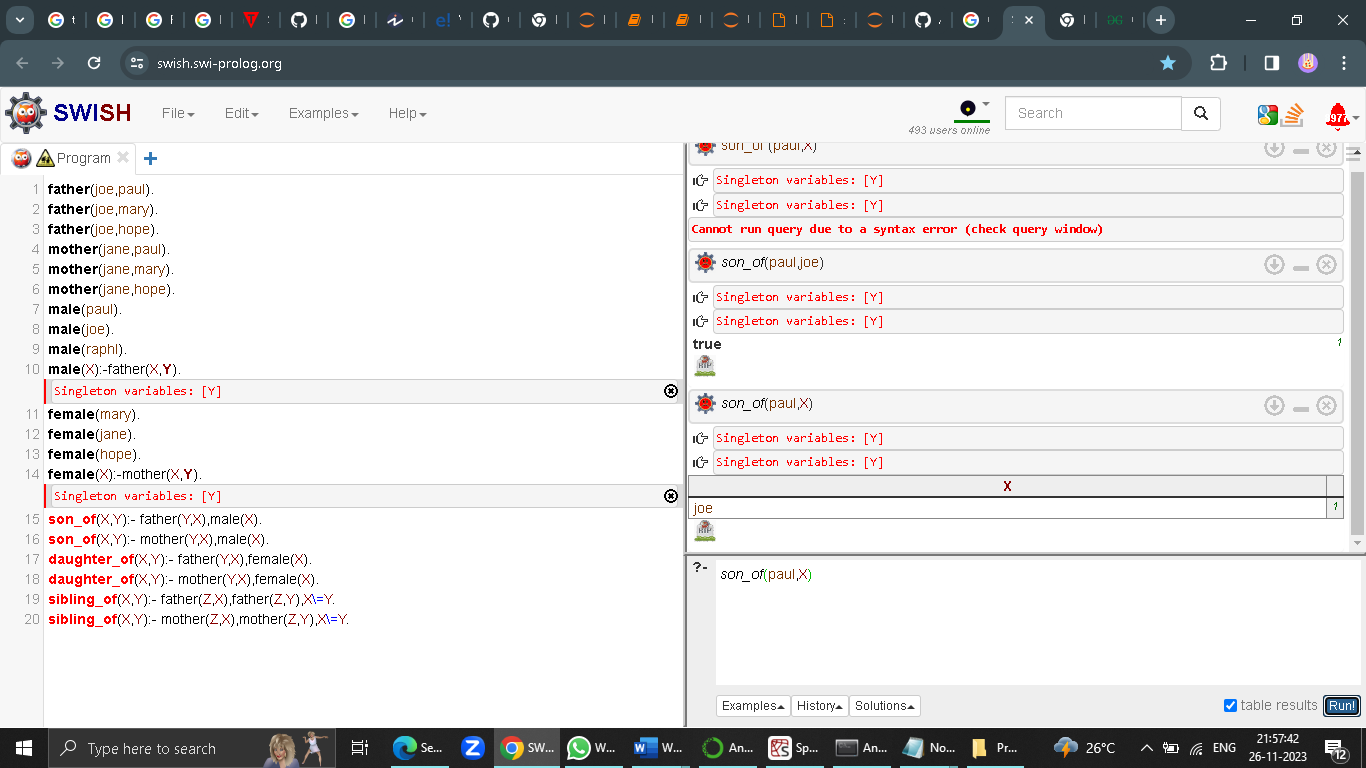
son\_of(X,Y):- mother(Y,X),male(X).

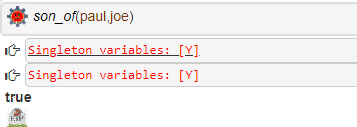
daughter\_of(X,Y):- father(Y,X),female(X).

daughter\_of(X,Y):- mother(Y,X),female(X).

sibling\_of(X,Y):- father(Z,X),father(Z,Y),X\=Y.

sibling\_of(X,Y):- mother(Z,X),mother(Z,Y),X\=Y.





**Practical 6**

**AIM: Design a Fuzzy based application using Python / R.**

**Theory:** **Applications of Fuzzy Logic**

The Fuzzy logic is used in various fields such as automotive systems, domestic goods, environment control, etc. Some of the common applications are:

* It is used in the **aerospace field** for**altitude control** of spacecraft and satellite.
* This controls the **speed and traffic** in the**automotive systems.**
* It is used for **decision making support systems** and personal evaluation in the large company business.
* It also controls the pH, drying, chemical distillation process in the **chemical industry**.
* Fuzzy logic is used in **Natural language processing** and various intensive [applications in Artificial Intelligence](https://www.edureka.co/blog/artificial-intelligence-applications/" \t "_blank).
* It is extensively used in **modern control systems** such as expert systems.
* Fuzzy Logic mimics how a person would make decisions, only much faster. Thus, you can use it with [Neural Networks](https://www.edureka.co/blog/what-is-a-neural-network/" \t "_blank).

These were some of the common applications of the Fuzzy Logic. Now, let’s have a look at the advantages and disadvantages of using Fuzzy Logic in AI.

**Code:**

import numpy as np

import skfuzzy as fuzz

import matplotlib.pyplot as plt

from skfuzzy import control as ctrl

from mpl\_toolkits.mplot3d import Axes3D # Required for 3D plotting

# New Antecedent/Consequent objects hold universe variables and membership

# functions

quality = ctrl.Antecedent(np.arange(0, 10, 0.1), 'quality')

service = ctrl.Antecedent(np.arange(0, 10, 0.1), 'service')

tip = ctrl.Consequent(np.arange(0, 25, 0.1), 'tip')

quality['poor'] = fuzz.zmf(quality.universe, 0,5)

quality['average'] = fuzz.gaussmf(quality.universe,5,1)

quality['good'] = fuzz.smf(quality.universe,5,10)

service['poor'] = fuzz.zmf(service.universe, 0,5)

service['average'] = fuzz.gaussmf(service.universe,5,1)

service['good'] = fuzz.smf(service.universe,5,10)

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])

tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])

tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

quality['average'].view()

plt.title('Quality')

service['poor'].view()

plt.title('Service')

tip['medium'].view()

plt.title('Tip Medium')

rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])

rule2 = ctrl.Rule(service['average'], tip['medium'])

rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

rule1.view()

plt.title('Rule 1')

rule2.view()

plt.title('Rule 2')

rule3.view()

plt.title('Rule 3')

tipping\_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])

tipping = ctrl.ControlSystemSimulation(tipping\_ctrl)

tipping.input['quality'] = 6.5

tipping.input['service'] = 9.8

tipping.compute()

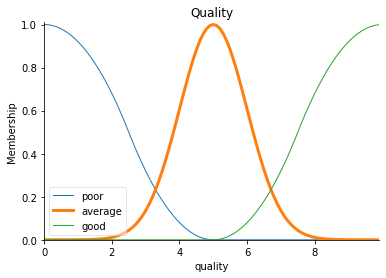
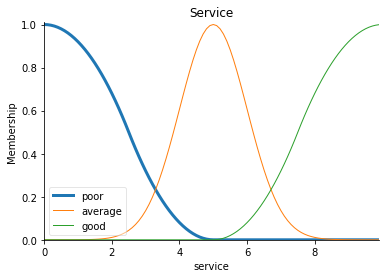
print(tipping.output['tip'])

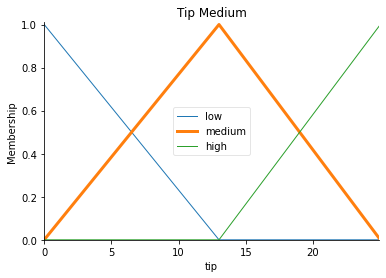
tip.view(sim=tipping)

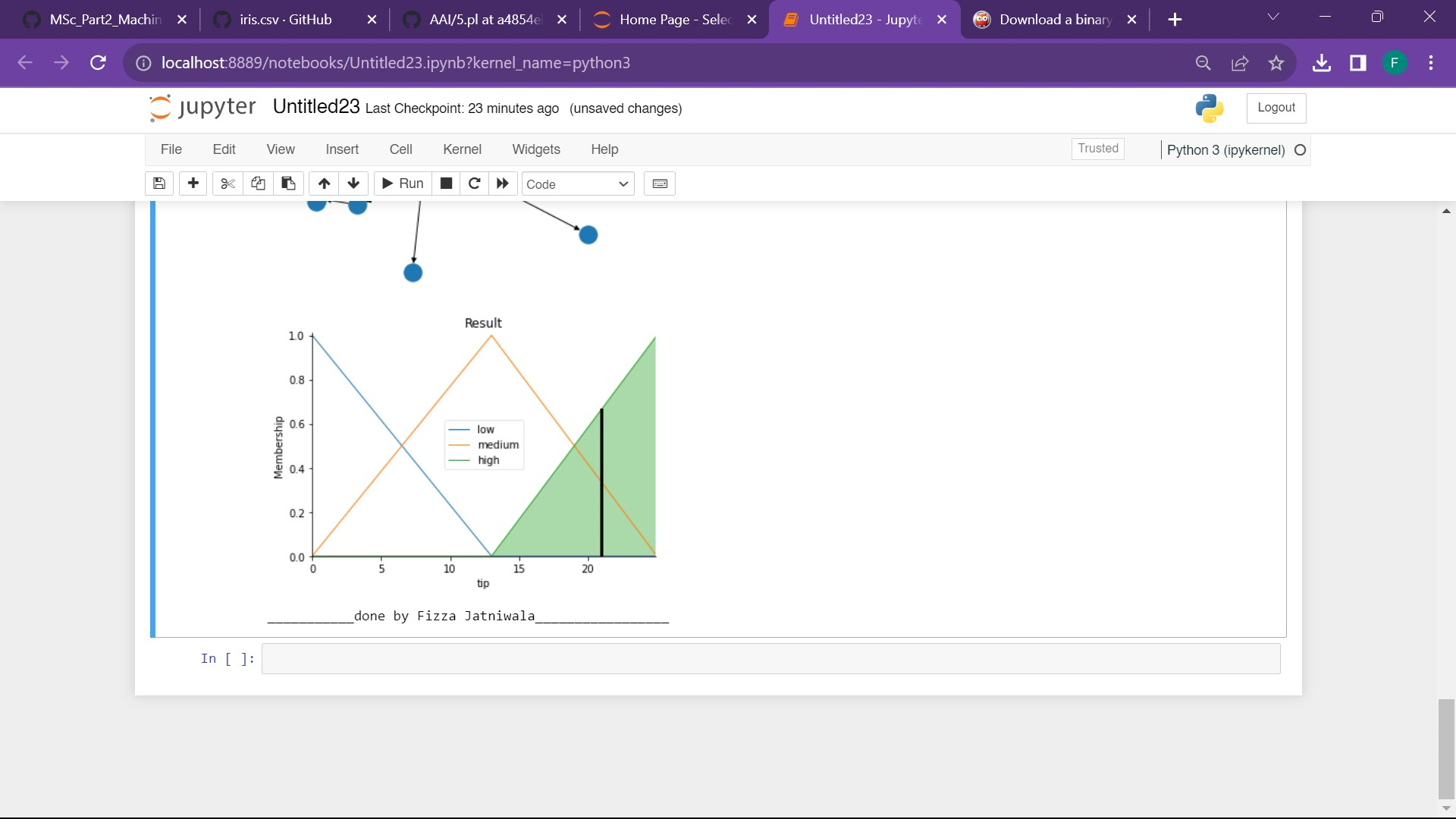
plt.title('Result')

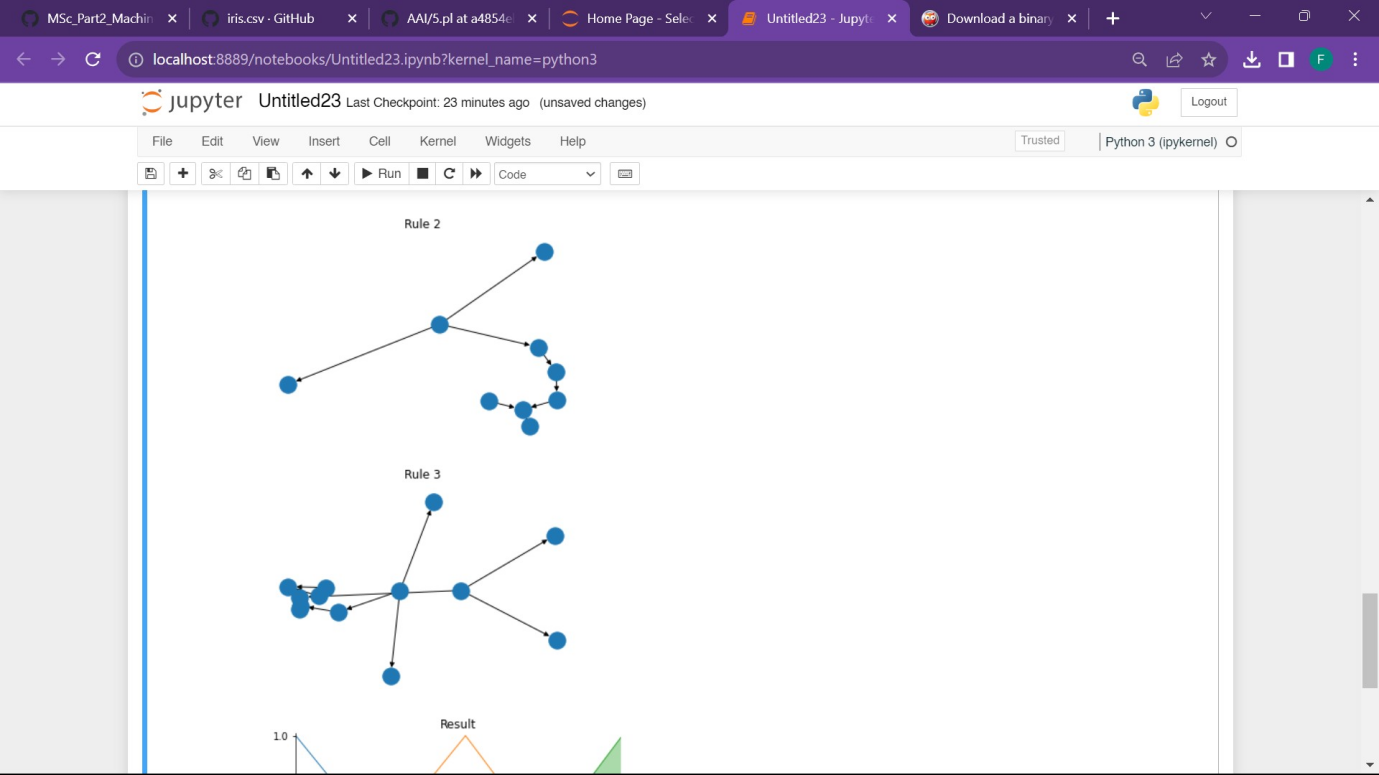
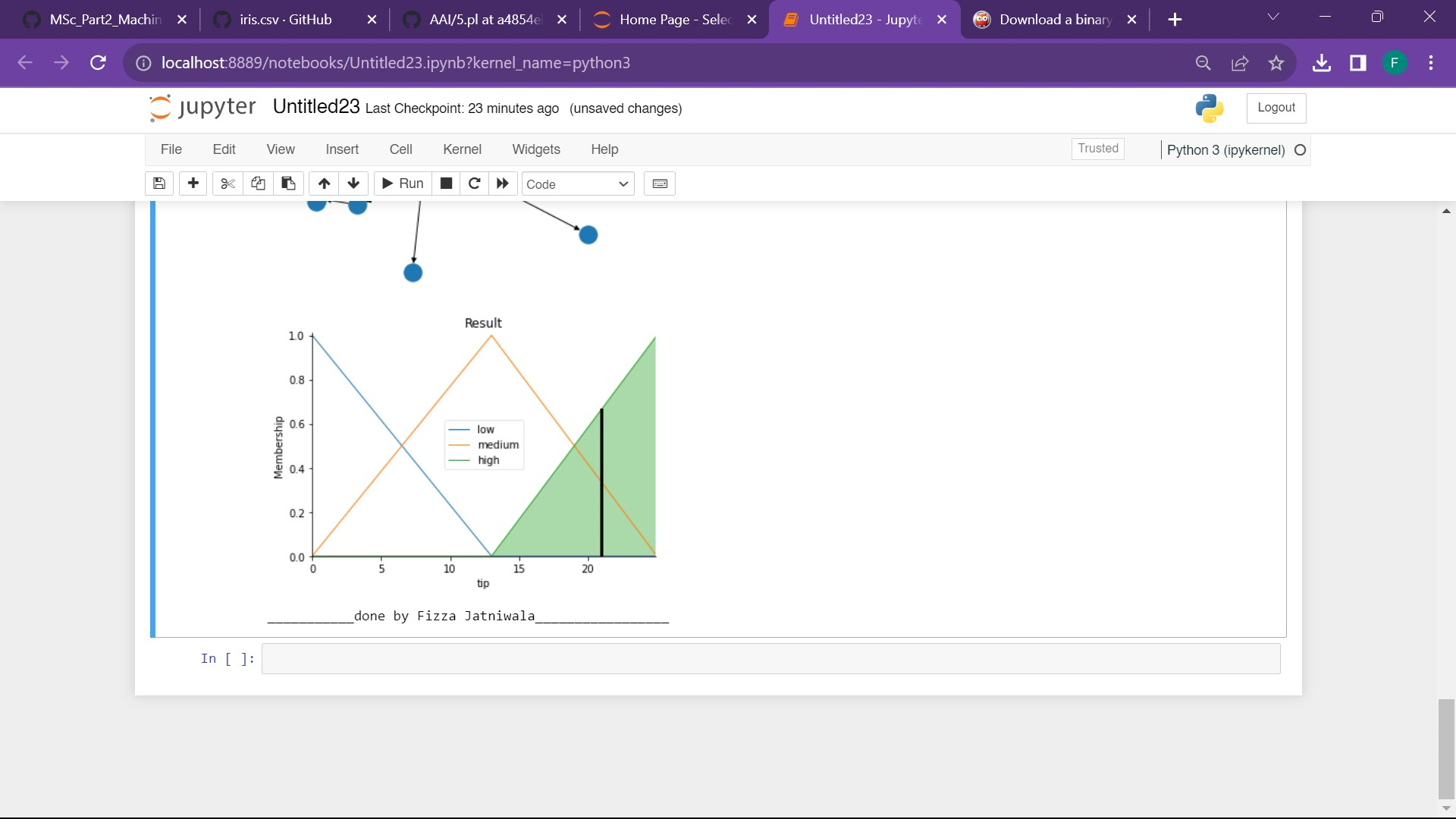
plt.show(block=True)

Print(“\_\_\_\_\_\_\_done by Fizza Jatniwala\_\_\_\_\_”)







**Practical 7**

**AIM: Write an application to implement clustering algorithm.**

**Theory:  Applications of K-Means Clustering**

K-Means clustering is used in a variety of examples or business cases in real life, like:

* Academic performance
* Diagnostic systems
* Search engines
* Wireless sensor networks

**Academic Performance**

Based on the scores, students are categorized into grades like A, B, or C.

**Diagnostic systems**

The medical profession uses k-means in creating smarter medical decision support systems, especially in the treatment of liver ailments.

**Search engines**

Clustering forms a backbone of search engines. When a search is performed, the search results need to be grouped, and the search engines very often use clustering to do this.

**Wireless sensor networks**

The clustering algorithm plays the role of finding the cluster heads, which collect all the data in its respective cluster.

**Code:**

# Importing Modules

from scipy.cluster.hierarchy import linkage, dendrogram

import matplotlib.pyplot as plt

import pandas as pd

# Reading the DataFrame

seeds\_df = pd.read\_csv("seeds-less-rows.csv")

# Remove the grain species from the DataFrame, save for later

varieties = list(seeds\_df.pop('grain\_variety'))

# Extract the measurements as a NumPy array

samples = seeds\_df.values

"""

Perform hierarchical clustering on samples using the

linkage() function with the method='complete' keyword argument.

Assign the result to mergings.

"""

mergings = linkage(samples, method='complete')

"""

Plot a dendrogram using the dendrogram() function on mergings,

specifying the keyword arguments labels=varieties, leaf\_rotation=90,

and leaf\_font\_size=6.

"""

dendrogram(mergings,

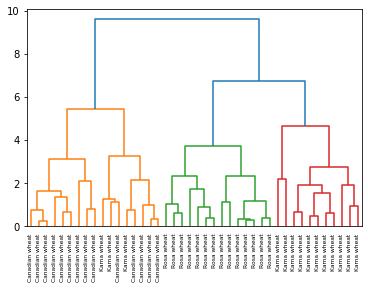
labels=varieties,

leaf\_rotation=90,

leaf\_font\_size=6,

)

plt.show()



**Practical 8**

**AIM: Write an application to implement support vector machine algorithm.**

**Theory:** Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

**SVM can be of two types:**

* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**Code:**

#Import scikit-learn dataset library

from sklearn import datasets

#Import svm model

from sklearn import svm

# Import train\_test\_split function

from sklearn.model\_selection import train\_test\_split

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

#Load dataset

cancer = datasets.load\_breast\_cancer()

# print the names of the 13 features

print("Features: ", cancer.feature\_names)

# print the label type of cancer('malignant' 'benign')

print("Labels: ", cancer.target\_names)

# print data(feature)shape

cancer.data.shape

# print the cancer data features (top 5 records)

print(cancer.data[0:5])

# print the cancer labels (0:malignant, 1:benign)

print(cancer.target)

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target, test\_size=0.3,random\_state=109) # 70% training and 30% test

#Create a svm Classifier

clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets

clf.fit(X\_train, y\_train)

#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

# Model Accuracy: how often is the classifier correct?

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

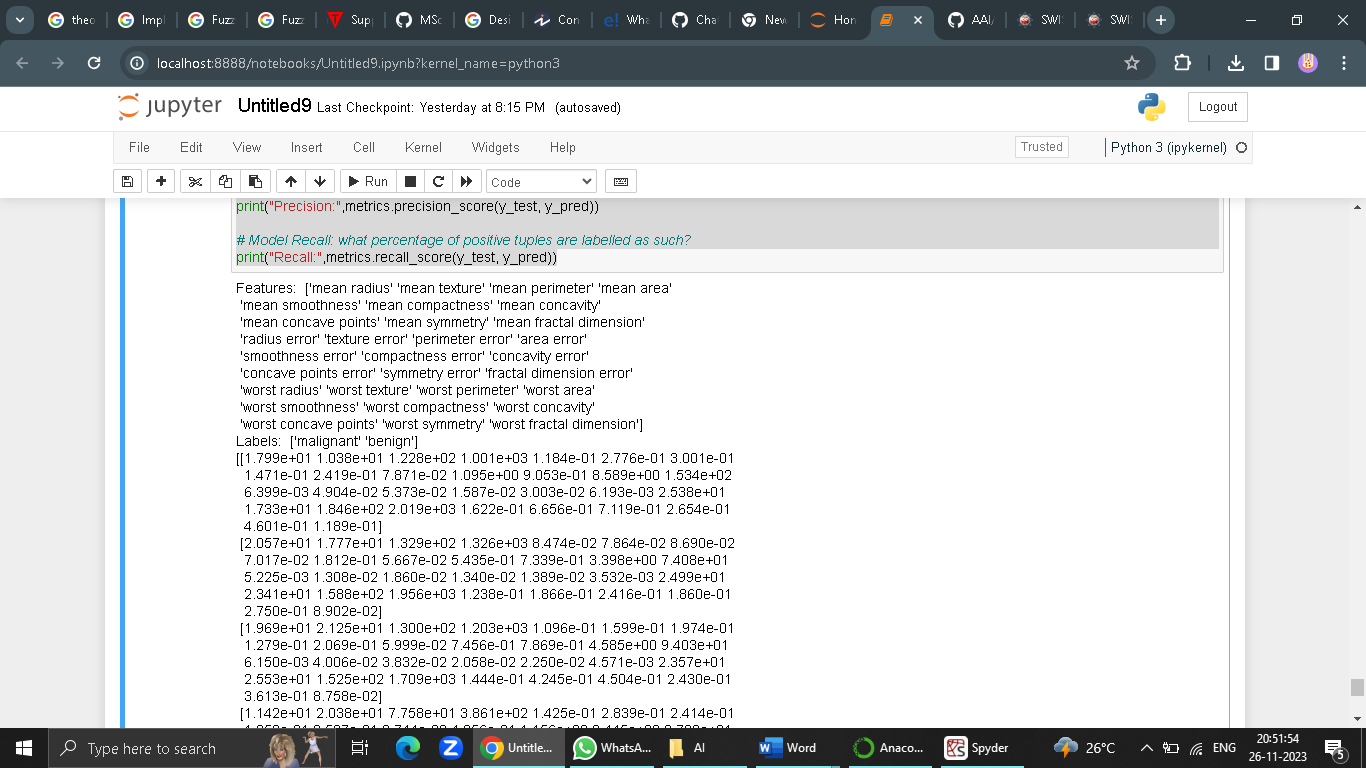
# Model Precision: what percentage of positive tuples are labeled as such?

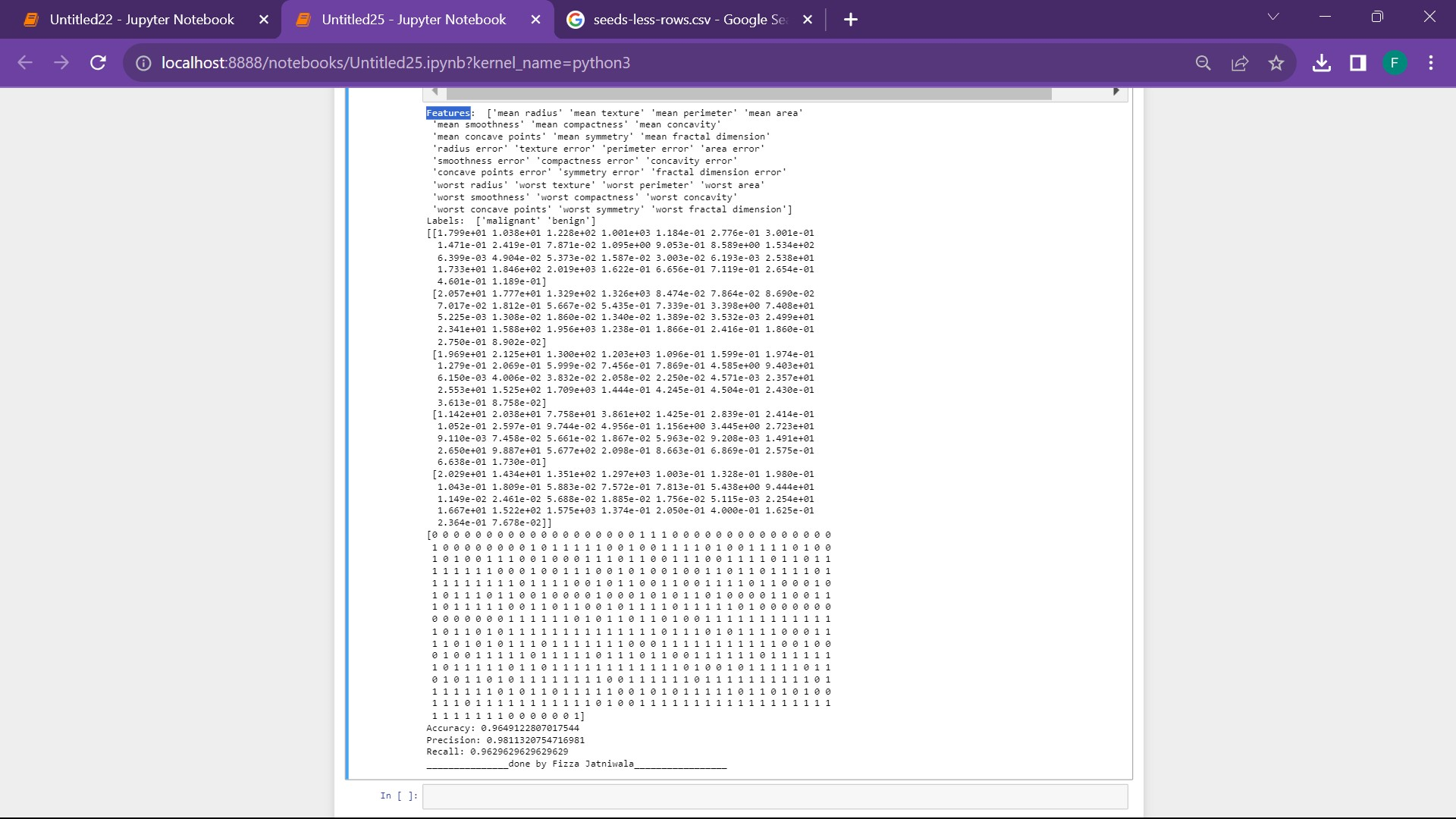
print("Precision:",metrics.precision\_score(y\_test, y\_pred))

# Model Recall: what percentage of positive tuples are labelled as such?

print("Recall:",metrics.recall\_score(y\_test, y\_pred))

print(“\_\_\_\_\_\_\_\_\_\_done by Fizza Jatniwala\_”)





**Practical 9**

**AIM: Simulate artificial neural network model with both feedforward and backpropagation approach.**

**Theory:** A **neural network** is a simplified model of the way the human brain processes information. It works by simulating a large number of interconnected processing units that resemble abstract versions of neurons.

The processing units are arranged in layers. There are typically three parts in a neural network: an **input layer**, with units representing the input fields; one or more **hidden layers**; and an **output layer**, with a unit or units representing the target field(s). The units are connected with varying connection strengths (or **weights**). Input data are presented to the first layer, and values are propagated from each neuron to every neuron in the next layer. Eventually, a result is delivered from the output layer.

The network learns by examining individual records, generating a prediction for each record, and making adjustments to the weights whenever it makes an incorrect prediction. This process is repeated many times, and the network continues to improve its predictions until one or more of the stopping criteria have been met.

Backpropagation, or backward propagation of errors, is an [algorithm](https://www.techtarget.com/whatis/definition/algorithm) that is designed to test for errors working back from output nodes to input nodes. It's an important mathematical tool for improving the accuracy of predictions in [data mining](https://www.techtarget.com/searchbusinessanalytics/definition/data-mining) and [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML). Essentially, backpropagation is an algorithm used to quickly calculate derivatives in a [neural network](https://www.techtarget.com/searchenterpriseai/definition/neural-network), which are the changes in output because of tuning and adjustments.

There are two leading types of backpropagation networks:

**Static backpropagation:** Static backpropagation is a network developed to map static inputs for static outputs. Static networks can solve static classification problems, such as optical character recognition ([OCR](https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition)).

**Recurrent backpropagation:** The recurrent backpropagation network is used for fixed-point learning. This means that during neural network training, the weights are numerical values that determine how much nodes -- also referred to as neurons -- influence output values. They're adjusted so that the network can achieve stability by reaching a fixed value.

**Code:**

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 Propogation

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

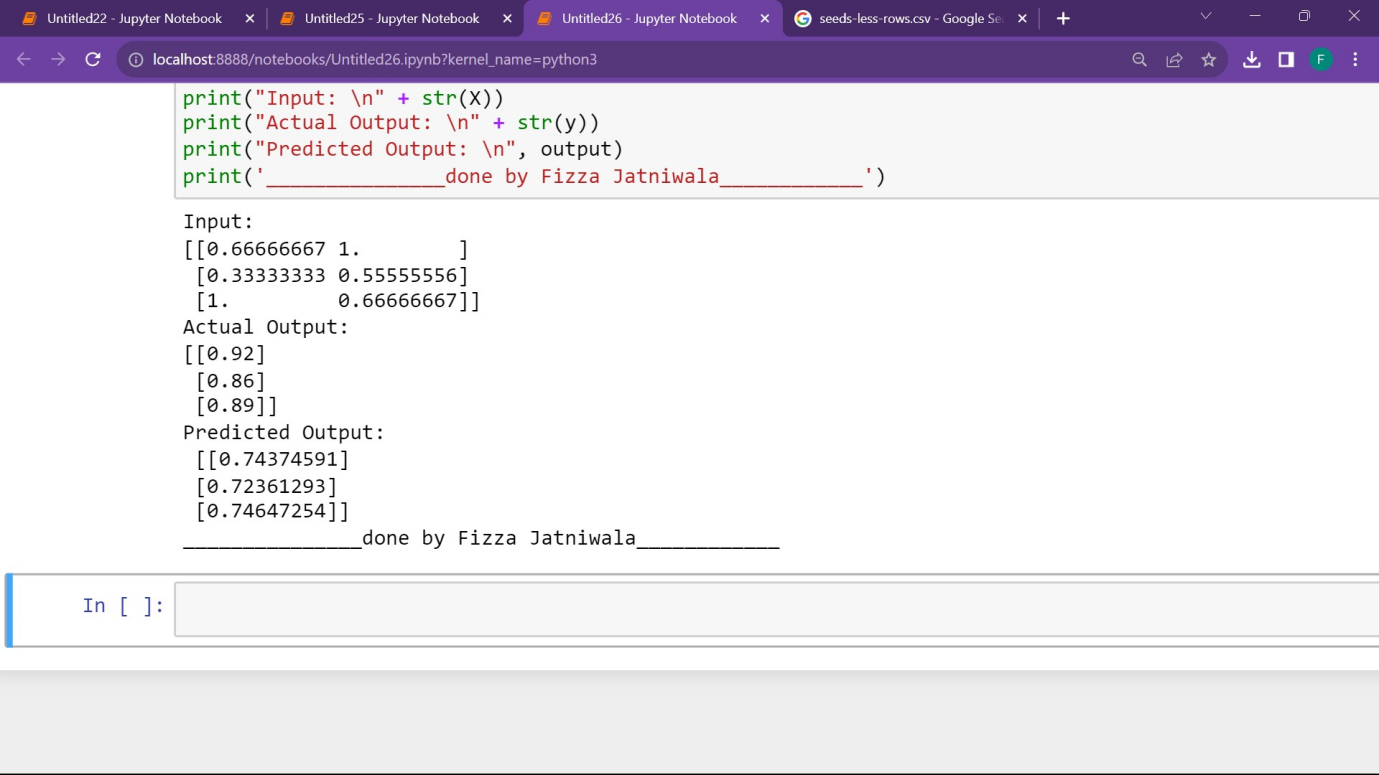
wh += X.T.dot(d\_hiddenlayer) \* lr

print("Input: \n" + str(X))

print("Actual Output: \n" + str(y))

print("Predicted Output: \n", output)

**Print(“\_\_\_\_\_\_done by Fizza Jatniwala\_\_\_\_\_\_\_”)**



**Practical 10**

**AIM: Simulate genetic algorithm with suitable example using Python / R or any other platform.**

**Theory:** Genetic algorithms (GAs) are a type of computational optimization technique inspired by the principles of natural selection and genetics. They are used to solve complex problems by mimicking the process of evolution to improve a population of potential solutions iteratively. These algorithms operate on a set of candidate solutions encoded as strings of binary digits or other data structures.

At the core of a genetic algorithm is the concept of a population, representing a collection of potential solutions to the problem at hand. Each individual within the population corresponds to a particular solution, and a set of parameters called genes defines its characteristics. These genes encode the properties or features of the solution, and they can be represented as binary strings, real-valued numbers, or other data types.

The genetic algorithm begins with an initial population of individuals, typically generated randomly. It then goes through a series of iterations, known as generations or epochs, in which the individuals undergo operations such as selection, crossover, and mutation. These operations mimic the processes of natural selection, reproduction, and genetic variation observed in biological evolution.

**Applications of Genetic Algorithms**

**1. Optimization problems**

GAs excel at solving optimization problems, aiming to find the best solution among a large set of possibilities. These problems include mathematical function optimization, parameter tuning, portfolio optimization, resource allocation, and more. GAs explore the solution space by enabling the evolution of a population of candidate solutions using genetic operators such as selection, crossover, and mutation, gradually converging towards an optimal or close-to-optimal solution.

**2. Machine learning**

GAs have applications in machine learning, particularly to optimize the configuration and parameters of machine learning models. GAs can be used to optimize hyperparameters, such as learning rate, regularization parameters, and network architectures in[neural networks](https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-a-neural-network/" \o "neural networks). They can also be employed for feature selection, where the algorithm evolves a population of feature subsets to identify the most relevant subset for a given task.

**3. Design and creativity**

GAs have been used for design and creativity tasks, such as generating artistic designs, music composition, and game design. By representing design elements or musical notes as genes, GAs can evolve populations of designs or compositions and evaluate their quality using fitness functions tailored to the specific domain. GAs have demonstrated the ability to generate novel and innovative solutions in creative domains.

**Code:**

import random

# Number of individuals in each generation

POPULATION\_SIZE = 100

# Valid genes

GENES = '''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP

QRSTUVWXYZ 1234567890, .-;:\_!"#%&/()=?@${[]}'''

# Target string to be generated

TARGET = "FIZZA JATNIWALA"

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: {}".\

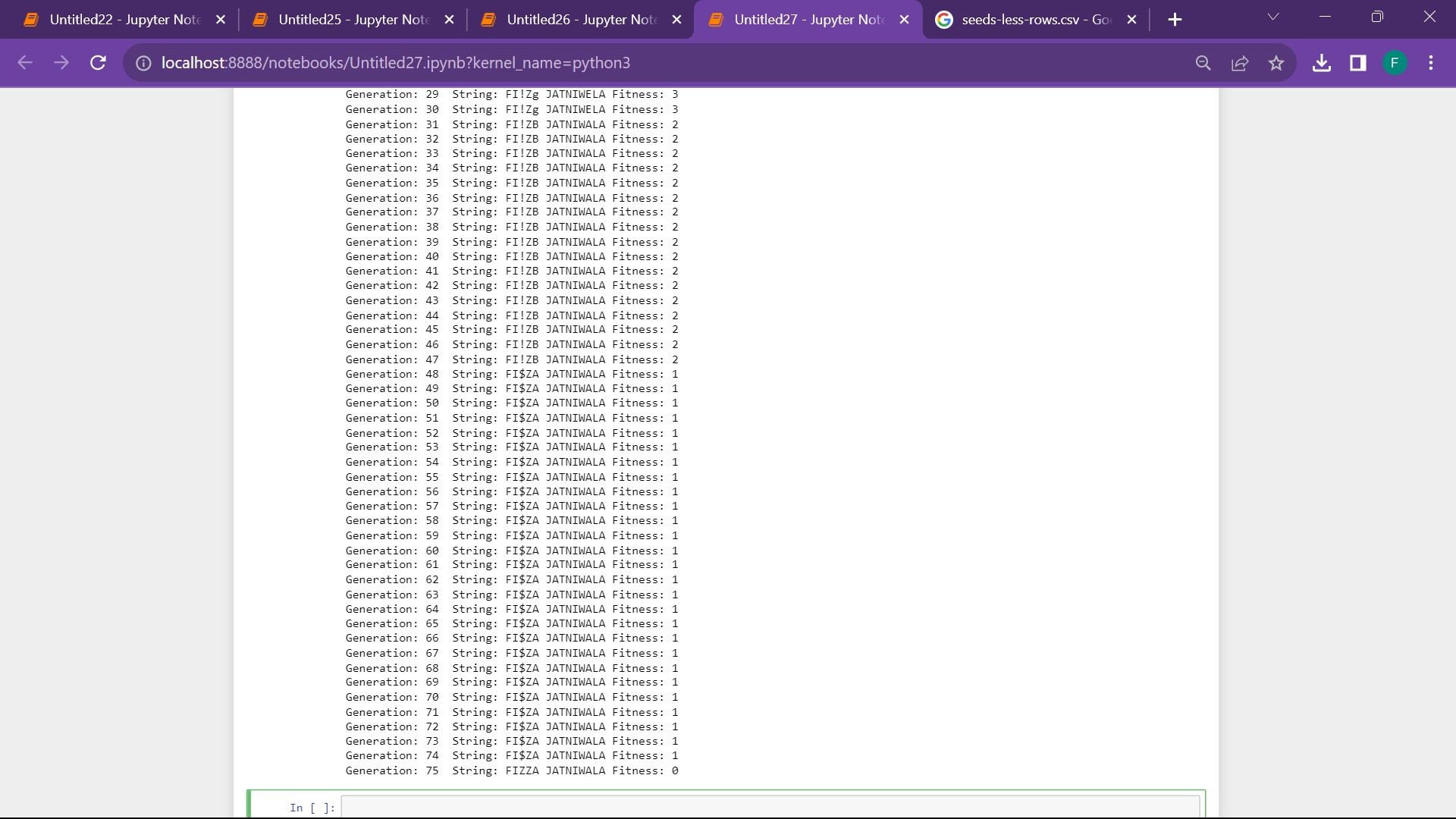
format(generation,

"".join(population[0].chromosome),

population[0].fitness))

if \_\_name\_\_ == '\_\_main\_\_':

main()



**Practical 11**

**AIM: Design an application to simulate language parser.**

**Theory:** Creating a language parser involves several steps, including designing the grammar, implementing parsing algorithms, and building a user interface. Below is a simplified outline for an application to simulate a language parser. The example assumes you're interested in a command-line application, but you can adapt it for other interfaces.

**1. Define Grammar:**

Define the grammar for the language you want to parse. Use a formal notation like Backus-Naur Form (BNF) to represent the syntax rules.

**2. Choose a Parsing Algorithm:**

Choose a parsing algorithm based on your grammar. Common choices include recursive descent parsing or using parser generators like ANTLR, yacc, or PLY (Python Lex-Yacc).

**3. Implement the Parser:**

Write code to implement the parser based on the chosen algorithm. If using a parser generator, follow the tool-specific guidelines. If building a recursive descent parser manually, create functions for each non-terminal in the grammar.

**4. Input Handling:**

Create a function to tokenize the input string into a sequence of tokens that the parser can understand.

**5. Testing:**

Test the parser with various inputs to ensure it handles different cases correctly.

**6. Refinement and Expansion:**

Refine your parser based on feedback and requirements. You can also expand it to support more features or a broader language.

Remember that this is a simplified example, and in a real-world scenario, you may need to handle more complex grammars and error scenarios.

**Code:**

# Importing the required libraries

import nltk

from nltk import CFG

# Defining the grammar rules

grammar = CFG.fromstring("""

S -> NP VP

NP -> Det N | Det N PP

VP -> V NP | V NP PP

PP -> P NP

Det -> 'The' | 'a' | 'the'

N -> 'dog' | 'cat' | 'house' | 'car'

V -> 'chased' | 'ate' | 'drove'

P -> 'in' | 'on' | 'at'

""")

# Creating the parser

parser = nltk.ChartParser(grammar)

# Parsing a sentence

sentence = "The dog chased the cat"

for tree in parser.parse(sentence.split()):print(tree)

Print(“\_\_\_\_\_\_\_\_\_\_done by Fizza Jatniwala\_\_\_\_\_”)

