Applied Artificial Intelligence

Index

Sr.No	Title	Date	Sign
1	Design a bot using AIML.		
2	Design an Expert system using AIML.		
3	Implement Bayes Theorem using Python.		
4	Implement Conditional Probability and Joint Probability using Python.		
5	Write a program to implement Rule based system (Prolog).		
6	Design a Fuzzy based application using Python/R.		
7	[A] Write an application to simulate supervised learning model. [B] Write an application to simulate unsupervised learning model.		
8	Write an application to implement clustering algorithm.		
9	Write an application to implement Support Vector Machine Algorithm.		
10	Simulate artificial neural network model with both feedforward and backpropagation approach.		

Practical No: 1

Aim: Design a bot using AIML.

Code:

Step 1: Create an XML file.

Open the notepad, write the following code and save it as *std-startup.xml*

```
<aiml version="1.0.1" encoding="UTF-8">
  <category>
   <pattern>LOAD AIML B</pattern>
   <template>
    <learn>basic_chat.aiml</learn>
   </template>
  </category>
</aiml>
```

Step 2: Create an aiml file.

Open the notepad, write the following code and save it as *basic_chat.aiml.*

```
<aiml version="1.0.1" encoding="UTF-8">
 <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 also my favourite.
  </template>
</category>
<category>
  <pattern>I LIKE *</pattern>
  <template>
    <set name = "liking"><star/></set> is a nice name.
  </template>
</category>
<category>
  <pattern>MY DOG NAME IS *</pattern>
  <template>
    <set name = "dog"><star/></set> THAT IS INTERESTING THAT YOU HAVE A DOG NAMED.
  </template>
</category>
<category>
  <pattern>Bye *</pattern>
  <template>
    <set name = "Bye!!!"><star/></set> Thanks for talking with me.
```

```
  </template>
 </category>
</aiml>
```

Step 3: Install aiml package through command prompt.

"pip install aiml"

or

"pip3 install aiml"

Step 4: Create chatbot.py file.

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
while True:
 message = input("Enter your message to bot: ")
 if message == "quit":
  break
 else:
  bot_response = kernel.respond(message)
  print(bot_response)
```

Output:

```
Loading std-startup.xml...done (0.02 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
Enter your message to the bot: hello
Well, hello!
Enter your message to the bot: what are you
I'm a bot, silly!
Enter your message to the bot: my dog name is sam
sam THAT IS INTERESTING THAT YOU HAVE A DOG NAMED.
Enter your message to the bot: bye bot
bot Thanks for talking with me.
Enter your message to the bot: quit
```

Practical No: 2

Aim: Design an Expert system using AIML.

Code:

Step 1: Create an XML file.

Open the notepad, write the following code and save it as *std-startup1.xml*

```
<aiml version="1.0.1" encoding="UTF-8">
  <category>
   <pattern>LOAD AIML B</pattern>
   <template>
    <learn>basic_chat1.aiml</learn>
   </template>
 </category>
</aiml>
```

Step 2: Create an aiml file.

Open the notepad, write the following code and save it as *basic_chat1.aiml*

```
<aiml version="1.0.1" encoding="UTF-8">
 <category>
   <pattern>HELLO</pattern>
   <template>
    WHAT WOULD YOU LIKE TO DISCUSS? :HEALTH,MOVIES
   </template>
```

```
  </category>
  <category>
    <pattern>MOVIES</pattern>
    <template>
      YES <set name = "topic">MOVIES</set>
    </template>
  </category>
<category>
    <pattern>HEALTH</pattern>
    <template>
      YES <set name = "topic">HEALTH</set>
    </template>
  </category>
  <topic name = "MOVIES">
    <category>
      <pattern>*</pattern>
      <template>
        DO YOU LIKE COMEDY MOVIES?
      </template>
    </category>
    <category>
      <pattern>YES</pattern>
      <template>
        I TOO LIKE COMEDY MOVIES!
      </template>
    </category>
    <category>
      <pattern>NO</pattern>
```

```
    <template>
      BUT I LIKE COMEDY MOVIES
    </template>
  </category>
</topic>
<topic name = "HEALTH">
  <category>
    <pattern>*</pattern>
    <template>
      DO YOU HAVE FEVER?
    </template>
  </category>
  <category>
    <pattern>YES</pattern>
    <template>
      PLEASE TAKE MEDICINES AND PROPER REST
    </template>
  </category>
  <category>
    <pattern>NO</pattern>
    <template>
      GO OUT FOR A WALK AND LISTEN MUSIC
    </template>
  </category>
</topic>
<category>
  <pattern>NICE TALKING TO YOU</pattern>
  <template>
```

```
    SAME HERE...!
   </template>
  </category>
</aiml>
```

Step 3: Install aiml package through command prompt.

"pip install aiml"

or

"pip3 install aiml"

Step 4: Create chatbot.py file.

```
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup1.xml")
kernel.respond("load aiml b")
while True:
  message = input("Enter your message to bot: ")
  if message == "quit":
    break
  else:
    bot_response = kernel.respond(message)
    print(bot_response)
```

Output:

```
Loading basic_chat1.aiml...done (0.00 seconds)
Enter a message for chatbot: hello
WHAT WOULD YOU LIKE TO DISCUSS? :HEALTH,MOVIES
Enter a message for chatbot: health
YES HEALTH
Enter a message for chatbot: i am feeling tired
DO YOU HAVE FEVER?
Enter a message for chatbot: no
GO OUT FOR A WALK AND LISTEN MUSIC
Enter a message for chatbot: movies
YES MOVIES
Enter a message for chatbot: i love movies
DO YOU LIKE COMEDY MOVIES?
Enter a message for chatbot: yes
I TOO LIKE COMEDY MOVIES!
Enter a message for chatbot: nice talking to you
SAME HERE...!
Enter a message for chatbot: quit
```

Practical No: 3

Aim: Implement Bayes Theorem using Python.

Code:

```
def bayes_theorem(p_a, p_b_given_a, p_b_given_not_a):
    not_a = 1 - p_a
    p_b = p_b_given_a * p_a + p_b_given_not_a * not_a
    p_a_given_b = (p_b_given_a * p_a) / p_b
    return p_a_given_b

p_a = 0.0002
p_b_given_a = 0.85
p_b_given_not_a = 0.05

result = bayes_theorem(p_a, p_b_given_a, p_b_given_not_a)
print('P(A|B): %.3f%%' % (result * 100))
```

Output:

```
P(A|B): 0.339%
```

Practical No: 4

Aim: Implement Conditional Probability and Joint Probability using Python

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 girls: ", both_girls)
print("Either girl: ", either_girl)
print("P(both | older): ", both_girls/older_girl)
print("P(both | either): ",both_girls/either_girl)
```

Output:

```
Older girl:  4937
Both girls:  2472
Either girl:  7464
P(both | older):  0.5007089325501317
P(both | either):  0.3311897106109325
```

Practical No: 5

Aim: Write a program to implement Rule based system (Prolog).

Code:

```
   hypothesis(Disease),
   write('I believe that the patient have '),
   write(Disease),
   nl,
   write('TAKE CARE '),
   undo.

/*Hypothesis that should be tested*/
hypothesis(cold) :- cold, !.
hypothesis(flu) :- flu, !.
hypothesis(typhoid) :- typhoid, !.
hypothesis(measles) :- measles, !.
hypothesis(malaria) :- malaria, !.
hypothesis(unknown). /* no diagnosis*/

/*Hypothesis Identification Rules*/

cold :-
verify(headache),
verify(runny_nose),
verify(sneezing),
verify(sore_throat),
write('Advices and Sugestions:'),
```

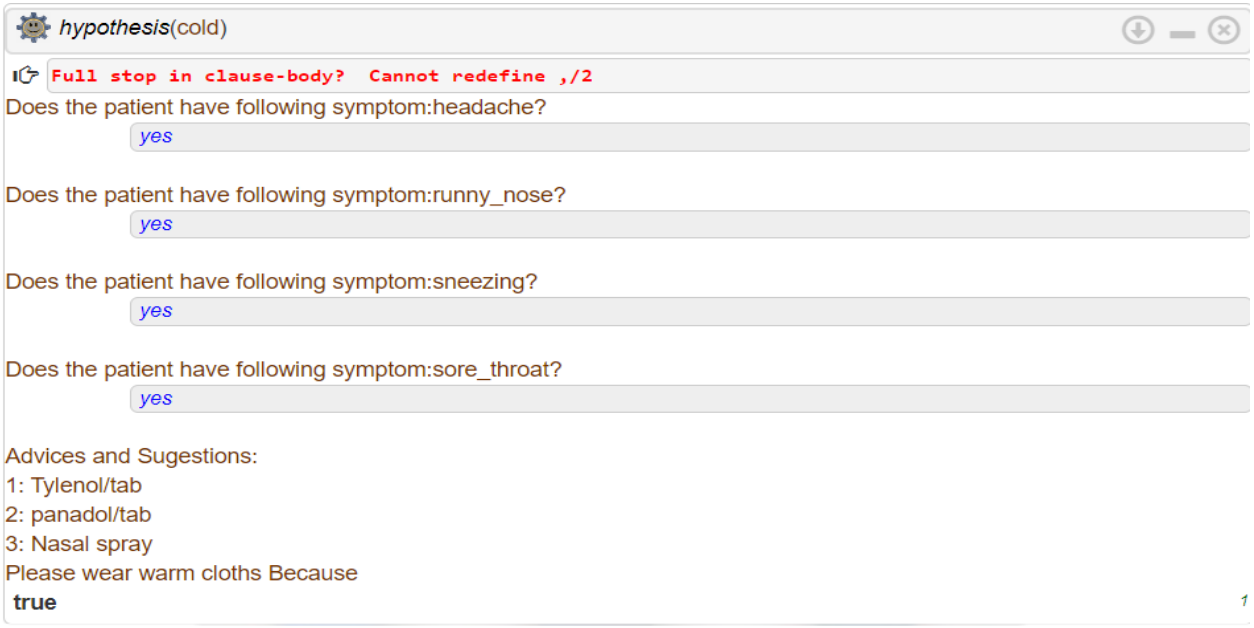
```
nl,
write('1: Tylenol/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Nasal spray'),
nl,
write('Please wear warm cloths Because'),
nl.

flu :-
verify(fever),
verify(headache),
verify(chills),
verify(body_ache),
write('Advices and Sugestions:'),
nl,
write('1: Tamiflu/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Zanamivir/tab'),
nl,
write('Please take a warm bath and do salt gargling Because'),
nl.

typhoid :-
verify(headache),
```

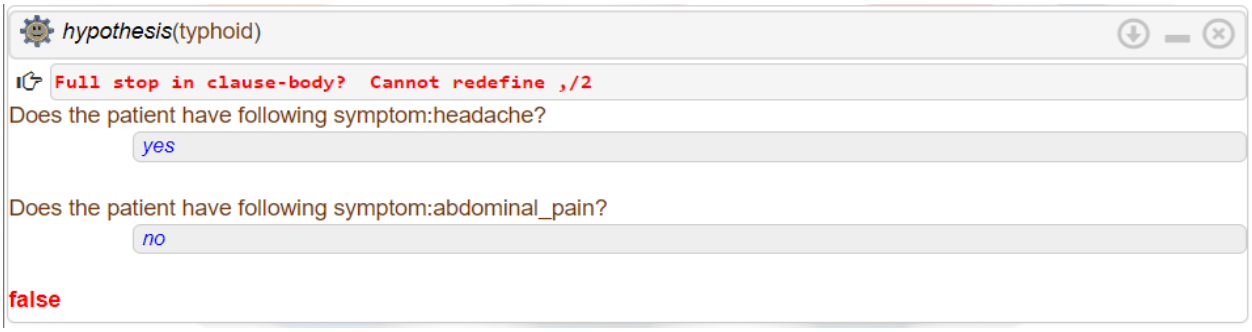
```
verify(abdominal_pain),
verify(poor_appetite),
verify(fever),
write('Advices and Sugestions:'),
nl,
write('1: Chloramphenicol/tab'),
nl,
write('2: Amoxicillin/tab'),
nl,
write('3: Ciprofloxacin/tab'),
nl,
write('4: Azithromycin/tab'),
nl,
write('Please do complete bed rest and take soft Diet Because'),
nl.

measles :-
verify(fever),
verify(runny_nose),
verify(rash),
verify(conjunctivitis),
write('Advices and Sugestions:'),
nl,
write('1: Tylenol/tab'),
nl,
write('2: Aleve/tab'),
nl,
write('3: Advil/tab'),
```

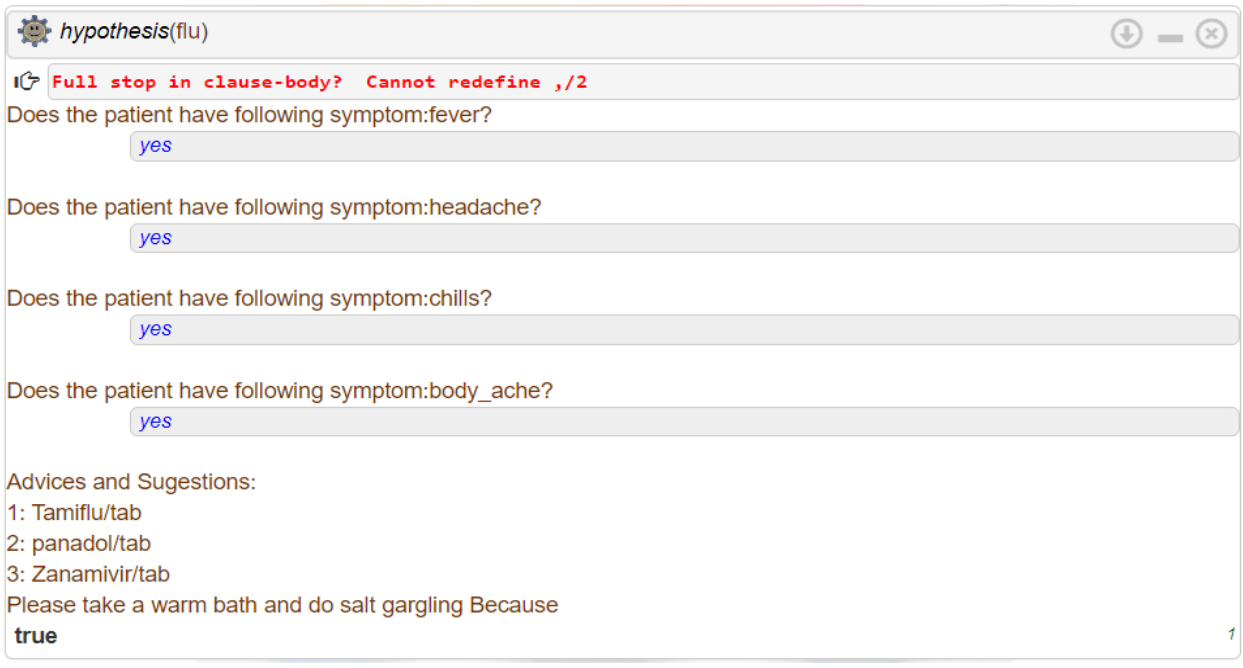
```
nl,
write('4: Vitamin A'),
nl,
write('Please Get rest and use more liquid Because'),
nl.

malaria :-
verify(fever),
verify(sweating),
verify(headache),
verify(nausea),
verify(vomiting),
verify(diarrhea),
write('Advices and Sugestions:'),
nl,
write('1: Aralen/tab'),
nl,
write('2: Qualaquin/tab'),
nl,
write('3: Plaquenil/tab'),
nl,
write('4: Mefloquine'),
nl,
write('Please do not sleep in open air and cover your full skin Because'),
nl.

/* how to ask questions */
ask(Question) :-
```

```
write('Does the patient have following symptom:'),
write(Question),
write('? '),
read(Response),
nl,
( (Response == yes ; Response == y)
->
assert(yes(Question)) ;
assert(no(Question)), fail).

:- dynamic yes/1,no/1.
/*How to verify something */
verify(S) :-
(yes(S)
->
true ;
(no(S)
->
fail ;
ask(S))).
/* undo all yes/no assertions*/
undo :- retract(yes(_)),fail.
undo :- retract(no(_)),fail.
undo.
```

Output:


hypothesis(cold)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:runny_nose?
yes
Does the patient have following symptom:sneezing?
yes
Does the patient have following symptom:sore_throat?
yes
Advices and Sugestions:
1: Tylenol/tab
2: panadol/tab
3: Nasal spray
Please wear warm cloths Because
true
1


hypothesis(typhoid)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:abdominal_pain?
no
false


hypothesis(flu)
Full stop in clause-body? Cannot redefine ,/2
Does the patient have following symptom:fever?
yes
Does the patient have following symptom:headache?
yes
Does the patient have following symptom:chills?
yes
Does the patient have following symptom:body_ache?
yes
Advices and Sugestions:
1: Tamiflu/tab
2: panadol/tab
3: Zanamivir/tab
Please take a warm bath and do salt gargling Because
true
1

Practical No: 6

Aim: Design a Fuzzy based application using Python/ R.

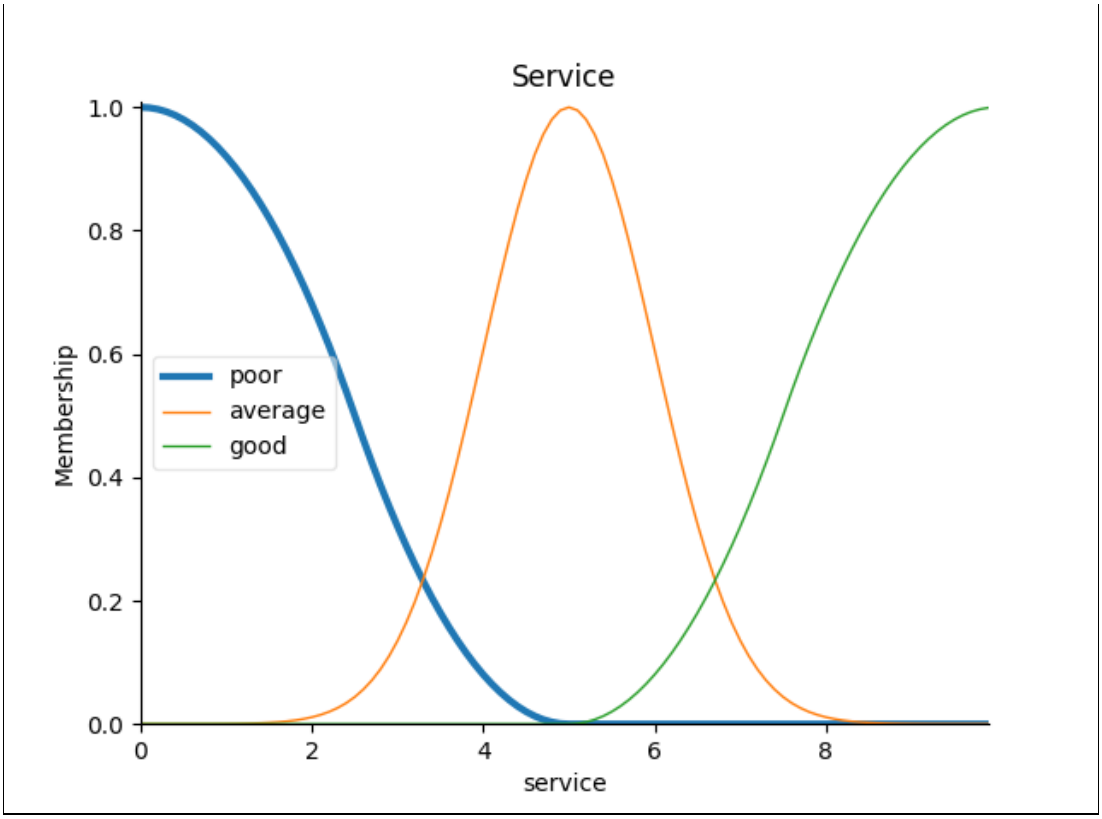
Code:

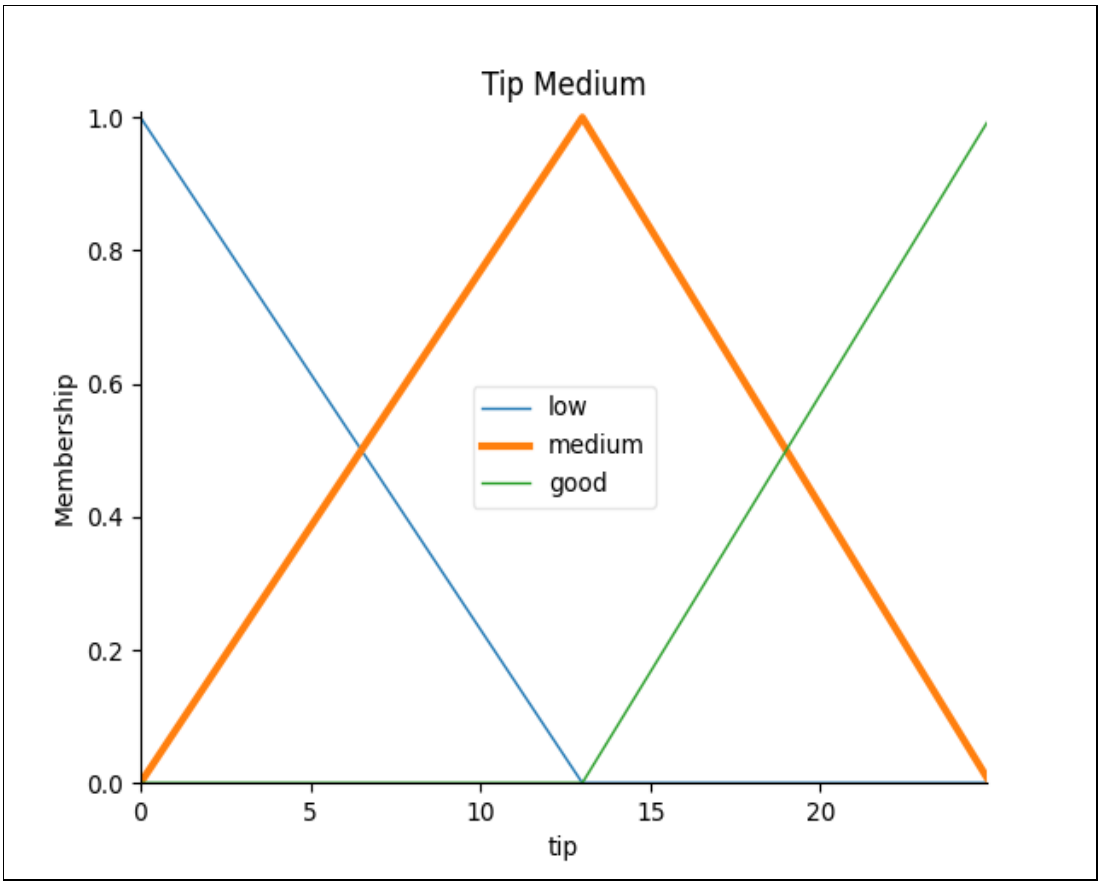
```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
from skfuzzy import control as ctrl
from mpl_toolkits.mplot3d import Axes3D

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

quality['average'].view()
```

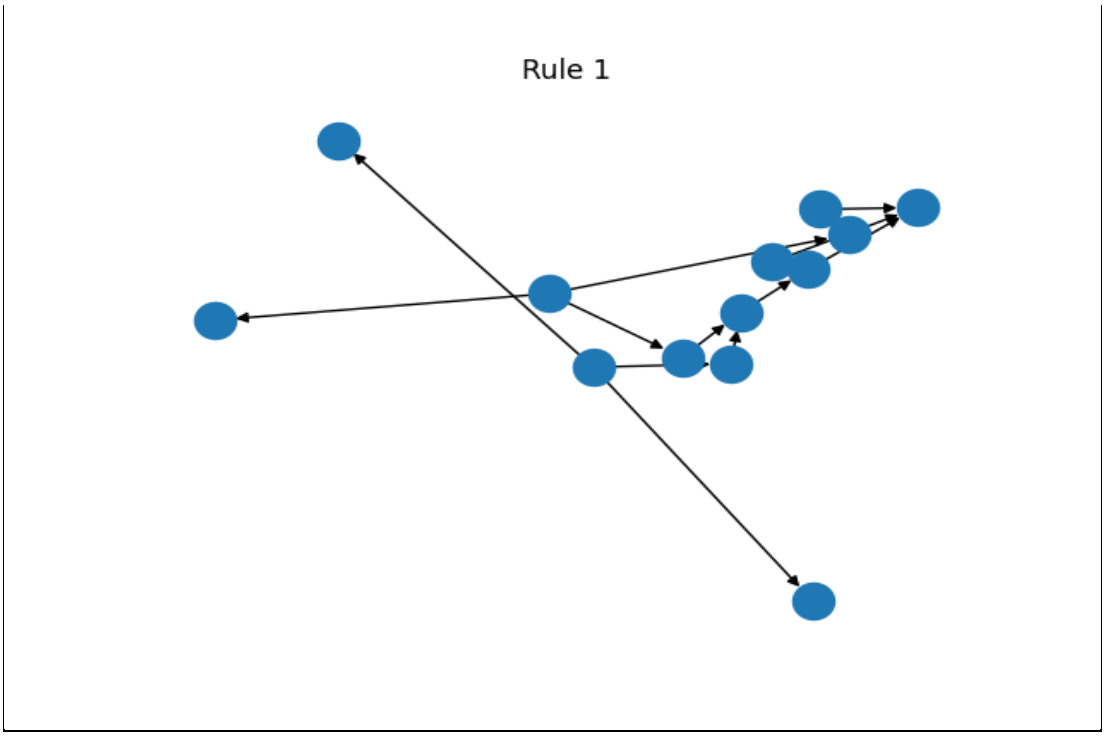
```
plt.title('Quality')
service['poor'].view()
plt.title('Service')
tip['medium'].view()
plt.title('Tip Medium')

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()
tip.view(sim=tipping)

plt.title('Result')
plt.show(block=True)
```

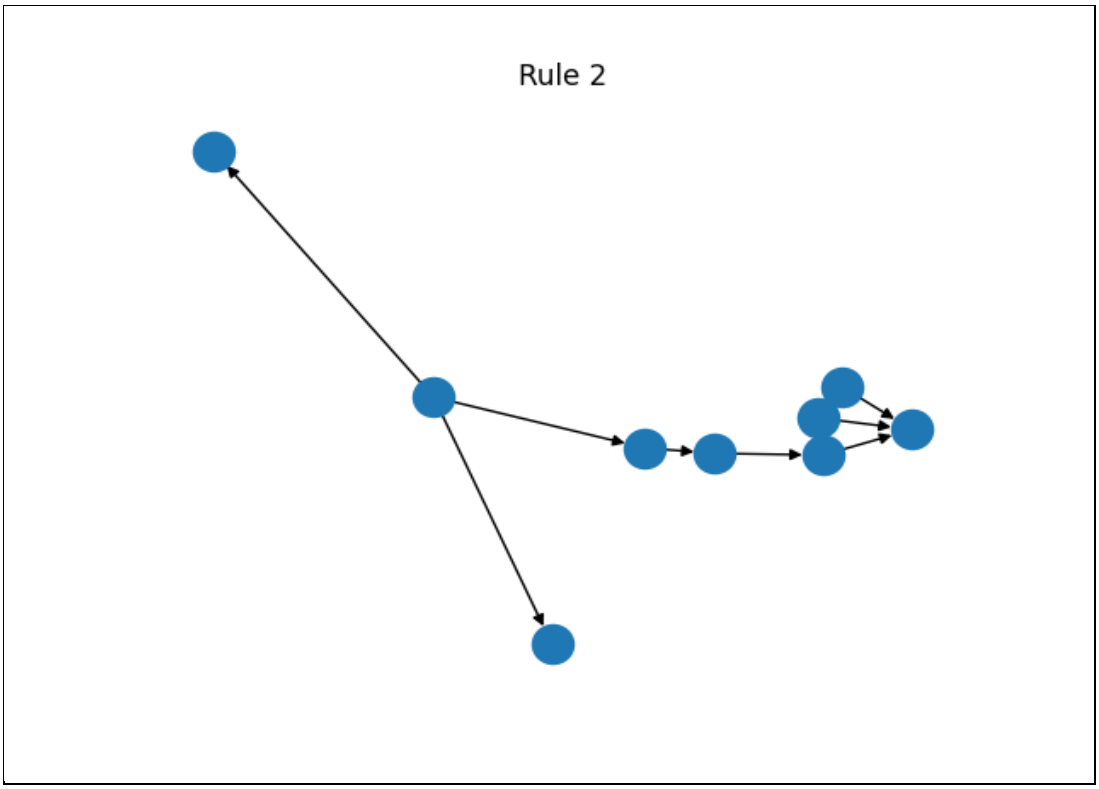
Output:

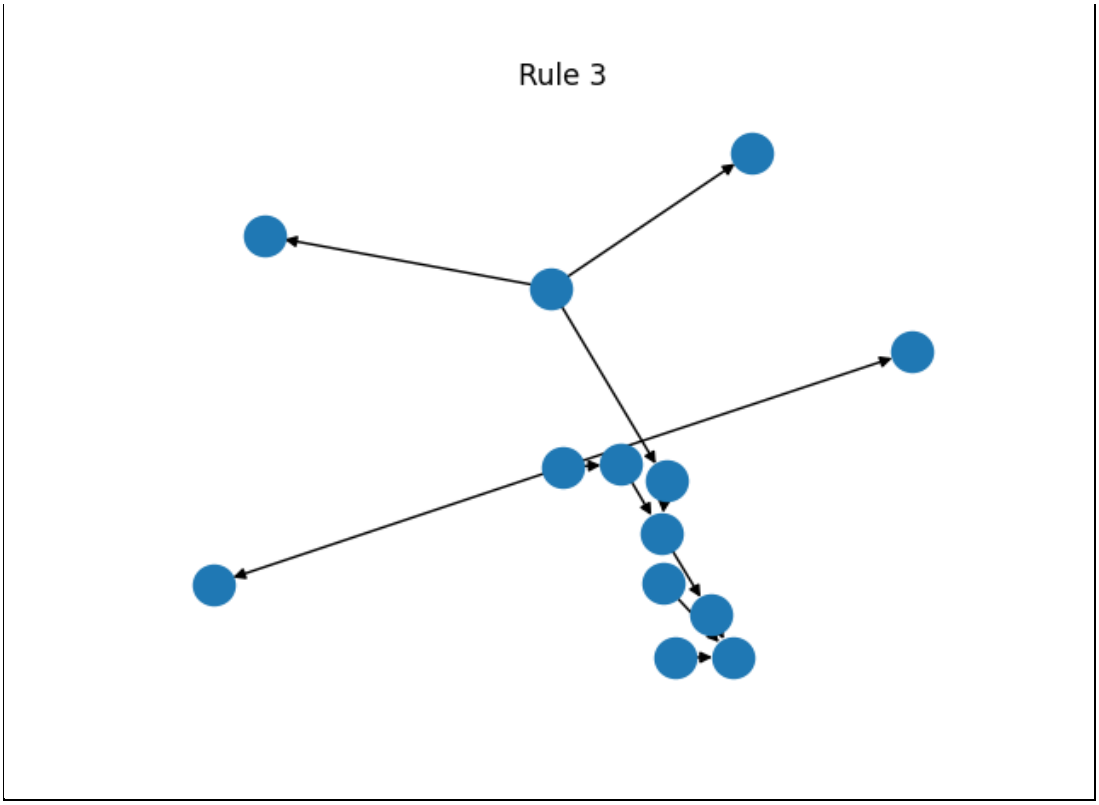

Quality
1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
2
4
6
8
quality

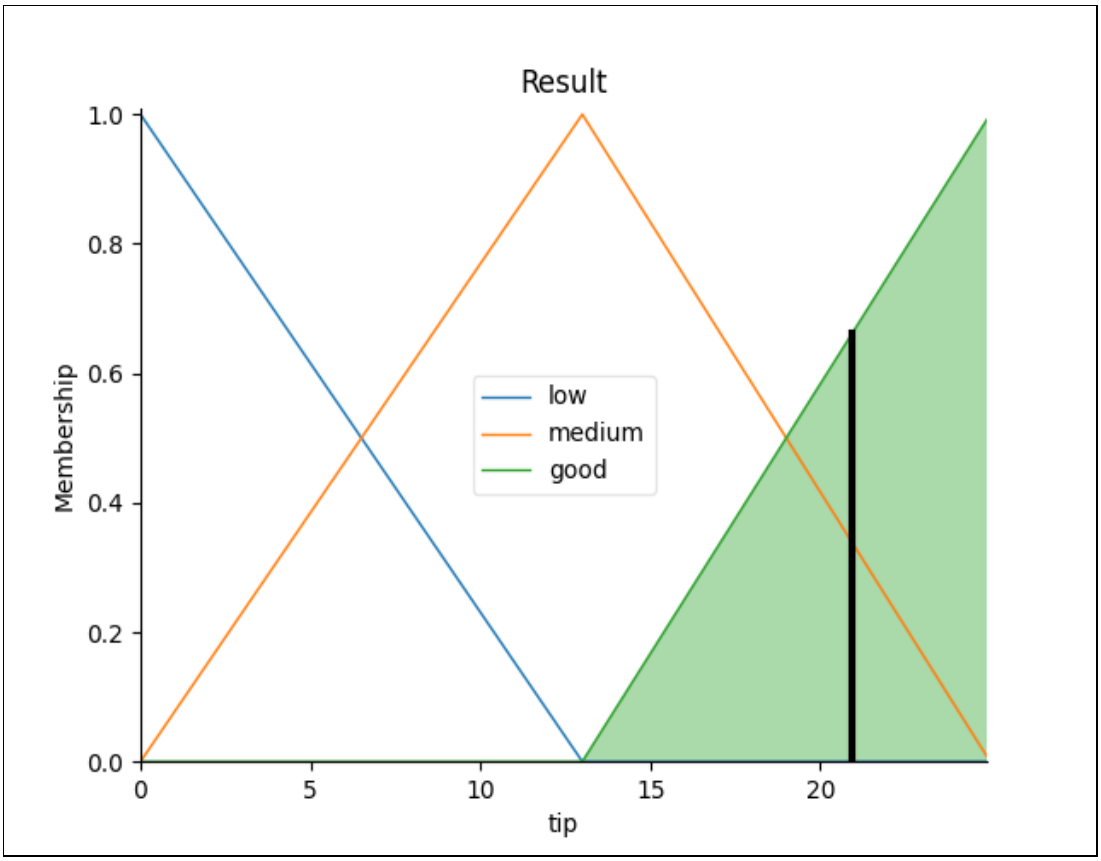

Service
1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
2
4
6
8
service


Tip Medium
1.0
0.8
0.6
0.4
0.2
0.0
Membership
low
medium
good
0
5
10
15
20
tip


Rule 1


Rule 2


Rule 3


Result
1.0
0.8
0.6
0.4
0.2
0.0
Membership
low
medium
good
0
5
10
15
20
tip

Practical No: 7

Aim: [A] Write an application to simulate supervised learning model.

Code:

```
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)

classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(x_train, y_train)

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:

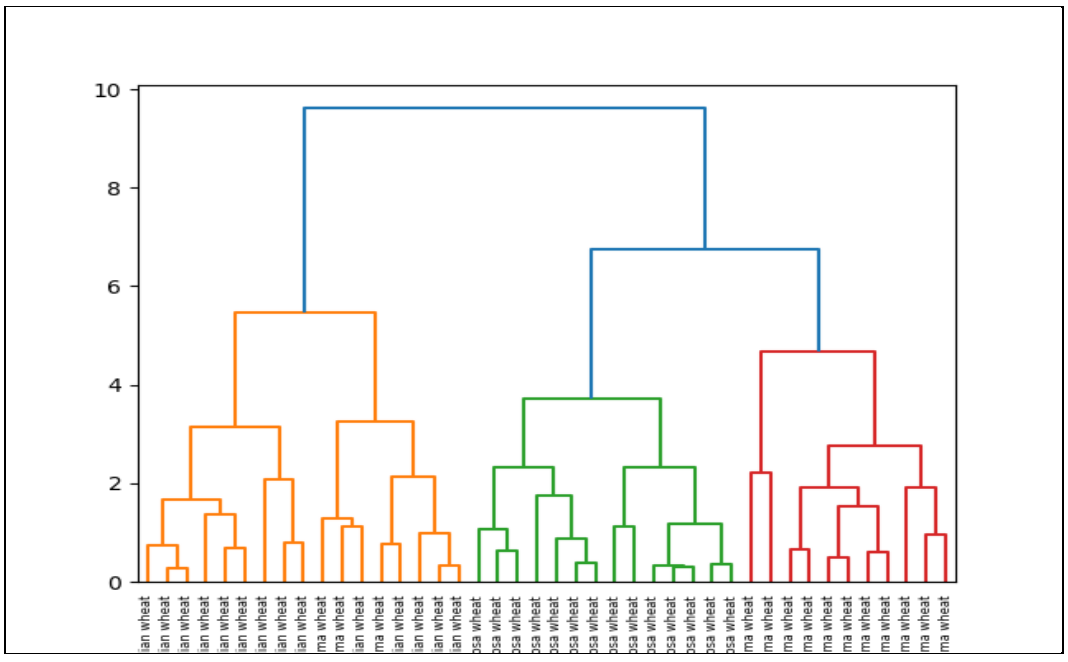
```
sepal-length sepal-width petal-length petal-width
Squeezed text (150 lines).
Class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Confusion Matrix
[[15  0  0]
 [ 0 13  1]
 [ 0  0 16]]
Accuracy Metrics
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.93      0.96        14
           2       0.94      1.00      0.97        16

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Aim: [B] Write an application to simulate unsupervised learning model.

Code:

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import pandas as pd

seeds_df = pd.read_csv("C:\\Users\\mihir\\Downloads\\seeds-less-rows.csv")
varieties = list(seeds_df.pop('grain_variety'))
samples = seeds_df.values

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

dendrogram(mergings, labels=varieties, leaf_rotation=90, leaf_font_size=6)
plt.show()
```

Output:



Practical No: 8

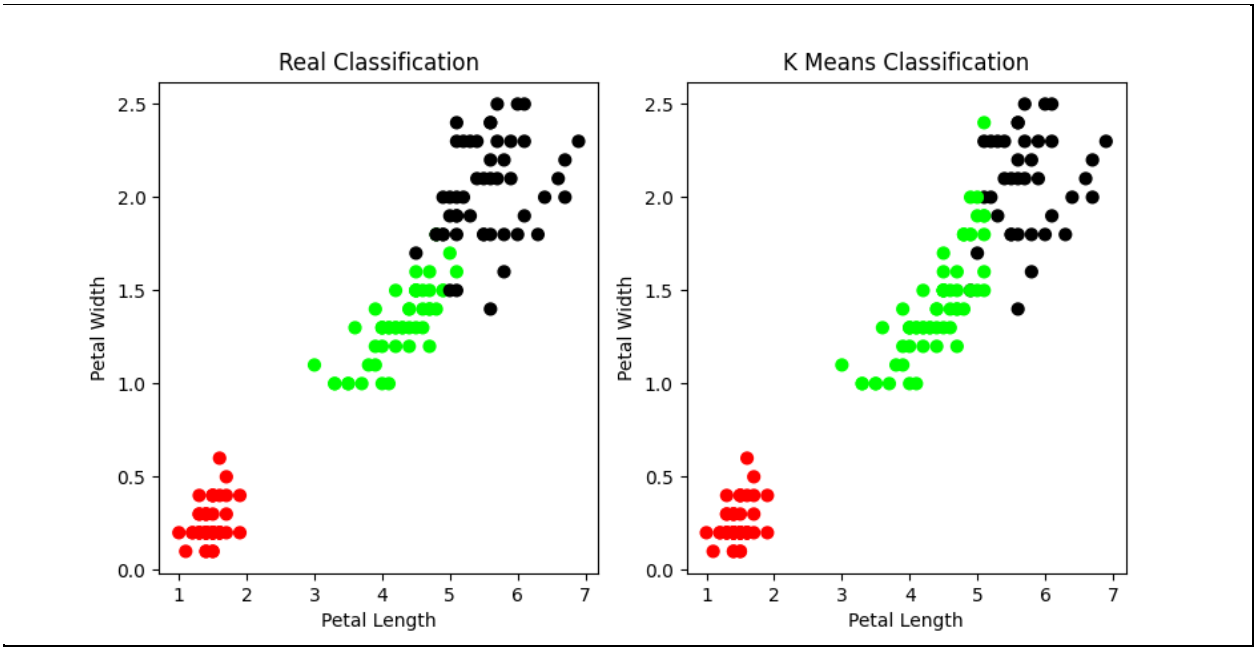
Aim: Write an application to implement clustering algorithm.

Code:

```
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, columns=['Sepal_Length', 'Sepal_Width', 'Petal_Length',
'Petal_Width'])
y = pd.DataFrame(iris.target, columns=['Targets'])

model = KMeans(n_clusters=3, n_init=10)
model.fit(X)

colormap = np.array(['red', 'lime', 'black'])
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[iris.target], s=40)
plt.title('Red Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
```

```
plt.title('K Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
print('The Accuracy score of K-Means: ', sm.accuracy_score(iris.target, model.labels_))
print('The Confusion matrix of K-Means: ')
print(sm.confusion_matrix(iris.target, model.labels_))
```

Output:




```
The Accuracy score of K-Means:  0.8933333333333333
The Confusion matrix of K-Means:
[[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
```

Practical No: 9

Aim: Write an application to implement Support Vector Machine algorithm.

Code:

```
from sklearn import datasets

from sklearn import svm

from sklearn.model_selection import train_test_split

from sklearn import metrics


cancer = datasets.load_breast_cancer()

print('Features: ', cancer.feature_names)

print('Labels: ', cancer.target_names)

print('Data shape: ', cancer.data.shape)


X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,
random_state=109)

clf = svm.SVC(kernel='linear')

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))

print('Precision: ', metrics.precision_score(y_test, y_pred))

print('Recall: ', metrics.recall_score(y_test, y_pred))
```

Output:

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
Data shape:  (569, 30)
Accuracy:  0.9649122807017544
Precision:  0.9811320754716981
Recall:  0.9629629629629629
```

Practical No: 10

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

Code:

```
import numpy as np

X = np.array(([2,9], [1,5], [3,6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X / np.amax(X, axis=0)
y = y / 100

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

def derivatives_sigmoid(x):
    return x * (1 - x)

epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
outputlayer_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, outputlayer_neurons))
```

```
bout = np.random.uniform(size=(1, outputlayer_neurons))

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)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad

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 Outpur: \n', output)
```

Output:

```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Outpur:
 [[0.91611219]
 [0.90155494]
 [0.91941754]]
```