**PRACTICAL:1**

**AIM: Design An Expert System Using AIML**

pract1.py 1 X

**pract1.py >...**

import aiml

kernel- aiml.Kernel()

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

kernel.respond ("load aiml b′′)

while True:

input\_text = input(">Human: ")

response = kernel.respond (input\_text)

print(">Bot: "+response)

pract1.py 1

**basic\_chat.aiml**

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

<!--std-startup.xml

<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>

Basic\_chat.aiml

1 <aiml version=”1.0.1” encoding=”UTF-8”>

<!—basic\_chat.aiml 🡪

<category>

<pattern>HELLO \*</pattern> <template>

Well, hello govind!

</template>

</category>

<category>

<pattern>WHAT ARE YOU</pattern>

<template>

I’m a bot, and I’m silly!

</template>

</category>

<category>

<pattern>WHAT DO YOU DO</pattern>

<template>

I’m here to annoy you!

</template>

</category>

<pattern>WHO I AM</pattern>

<template>

<category>

You are Govind Saini, and you working on Web Developer…

</template>

</category>

</aiml>

**PRACTICAL:2**

**AIM : Design A Bot Using AIML.**

**Code:-**

**Pract2.py > …**

Import aiml

Kernel = aiml.Kernel()

Kernel.learn(“std2-startup.xml”) kernel.respond (“load prac\_2”)

While True:

Input\_text = input(“> Human: “)

Response = kernel.respond(input\_text)

print(“> Bot: “+response)

Std2-startup.xml

<aiml version=”1.0.1” encoding=”UTF-8”> <!—std-startup.xml 🡪

<category>

<!—Pattern to match in user input <!—If user enters “LOAD AIML B” 🡪 <pattern>LOAD PRAC 2</pattern>

<!—Template is the response to the pattern

<!—This learn an aiml file 🡪

<template>

<learn>prac2\_chat.aiml</learn>

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

<!--<learn>more\_aiml.aiml</learn>🡪

</template>

</category>

/aiml

Basic chat siml

<category>

<pattern>MONDAY</pattern>

<template>the day of the week before Tuesday and following Sunday</template> </category>

<category>

<pattern>TUESDAY</pattern>

</template>

</category>

<template>the day of the week before Wednesday and following Monday</template> </category>

<category>

<pattern>WEDNESDAY</pattern>

<template>the day of the week before Thursday and following Tuesday</template>

</category>

<category>

<pattern>THURSDAY</pattern>

<template>the day of the week before Friday and following Wednesday</template>

</category>

<pattern>FRIDAY</pattern>

<category>

<template>the day of the week before Saturday and following Thursday</template>

</category>

<category>

<pattern>SATURDAY</pattern>

<template>the day of the week before Sunday and following Friday. </template> </category>

</aiml>

**PRACTICAL:3**

**AIM : Implement Bayes Theorem Using Python.**

**Code:-**

Drug\_user

1 def drug user(

Prob\_th=0.8,

Sensitivity-0.79,

Specificity=0.79,

Prevelance-0.02,

Verbose=True):

#Computes the posterior using Bayes’ rule

P\_user = prevelance

P\_non\_user= 1-prevelance

P\_neg\_user = specificity

P\_pos\_non\_user = 1-specificity

P\_pos\_user sensitivity

Num = p\_pos\_user\*p\_user

Den = p\_pos\_user\*p\_user+p\_pos\_non\_user\*p\_non\_user

Prob = num/den

If verbose:

If prob> prob\_th:

Else:

Print(“The test-taker could be an user”)

Print(“The test-taker may not be an user”)

Return prob

Print(“Govind Saini”)

p-drug\_user(prob\_th-0.5, sensitivity-0.97, specificity-0.95, prevelance=0.005) 30 print(“Probability of the test-taker being a drug user is:”, round (p,3))

**PRACTICAL:4**

**AIM : Implement Conditional Probability And Joint Probability. Using**

**Python.**

**Code**

Pract4.py >…

Def conditional():

Pass\_stats 0.15

Pass\_codingWStats – 0.60

pass\_codingWOStats-0.40

Prob\_both -pass\_stats

Pass\_codingwStats

Print(“The probability that applicant passes both is”, round (prob\_both, 3)) prob\_coding = (prob\_both) + ((1-pass\_stats)\*pass\_codingWOStats)

Print(“Probability that he/she passes only coding is”, round (prob\_coding, 3)) stats\_given\_coding = prob\_both/prob\_coding

Print(“Hey Govind”)

Conditional()

Print(“Conditional probabilty is”, round (stats\_given\_coding, 3))

Import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

Sns.set()

Import pandas as pd

Data pd.read\_csv(‘data.csv’, header-None, names-[‘x’, ‘y’])

Sns.jointplot(data[‘x’], data[‘y’], kind-‘kde’).plot\_joint (sns.scatterplot)

Plt.show()

**Practical no. 5**

**Aim:- : A Program To Implement Rule Based System.**

**Code:-**

Man (lou).

Man (pete).

Man (ian).

Man (peter).

Woman (pauline).

Woman (cathy).

Woman (lucy).

Parent (ian, lucy).

Parent (ian, peter).

Parent (cathy, ian).

Parent (pete, ian).

Parent (lou, pete).

Parent (lou, pauline).

Mother (X, Y) :- woman (x), parent (X, Y), (X\=Y)

.Father (X, Y) :- man (X), parent (X, Y), (X\=Y).

Sibling (X, Y) :-Parent (Z, X), parent (Z, Y), (X\=Y).

Brother (X, Y) :- man (X), sibling (X, Y), (X\=Y).

sister (X, Y):-woman (X), sibling (X, Y), (X\=Y).

Grandfather (X, Y) :- father (X, Z), parent (Z, Y), (X\=Y).

grandmother (X, Y) :- mother (X, 2), parent (Z, Y), (X\=Y).

Ancestor (X, Y):Parent (X, Y), (X\=Y).

Ancestor (X, Y) :-Parent (X, Z), ancestor (Z, Y), (X\=Y).

**PRACTICAL:6**

**AIM : Design A Fuzzy Based Application Using Python.**

**Code:**

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and mer quality = ctrl.Antecedent (np.arange(0, 11, 1), 'quality')

service = ctrl.Antecedent (np.arange(0, 11, 1), 'service') tip=ctrl.Consequent (np.arange(0, 26, 1), 'tip')

quality.automf (3)

service.automf (3)

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()

service.view() tip.view()

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()

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()

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)

**PRACTICAL:7**

**AIM : Write And Application To Stimulate Supervised And Un-Supervised Learning Model.**

**Code:-**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.linear model import Logistic Regression

from sklearn import datasets

# Importing the dataset

dataset = pd.read\_csv("iris.csv")

dataset.describe()

# Splitting the dataset into the Training set and Test set

X = dataset.iloc[:, [0,1,2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model selection import train\_test\_split

X\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.25, random\_state from sklearn.preprocessing import StandardScaler

SC =StandardScaler()

X\_train X\_test

sc.fit\_transform(X\_train) sc.transform(X\_test)N0)=0, solver='lbfgs', multi\_class='auto')

# Fitting Logistic Regression to the Training set classifier Logistic Regression (random\_state classifier.fit(x\_train, y\_train)

# Predicting the Test set results y\_pred = classifier.predict(X\_test) # Predict probabilities

probs\_y=classifier.predict\_proba (X\_test) from sklearn.metrics import confusion\_matrix cm = confusion\_matrix (y\_test, y\_pred) print(cm)

# Plot confusion matrix

import seaborn as sns

import pandas as pd

# confusion matrix sns heatmap

ax - plt.axes()

df\_cm = cm

sns.heatmap(df\_cm, annot=True, annot\_kws={"size": 30), fmt='d', cmap="Blues", ax = ax )

ax.set\_title('Confusion Matrix')

plt.show()

import matplotlib.pyplot as plt import pandas as pd

import numpy as np

customer\_data = pd.read\_csv(‘Mall\_Customers.csv’)

customer\_data.shape customer\_data.head()

data = customer\_data.iloc[:, 3:5].values import scipy.cluster.hierarchyshc

plt.figure(figsize=(10, 7))

plt.title(“Customer Dendograms”)

dend = shc.dendrogram(shc.linkage (data, method=’ward’)) from sklearn.cluster import AgglomerativeClusteringcluster

AgglomerativeClustering (n\_clusters-5, affinity-‘euclidean’, linkage=’wa

Cluster.fit\_predict(data)

Plt.figure(figsize=(10, 7))

Plt.scatter (data[:,0], data[:,1], c-cluster.labels\_, cmap=”rainbow’) plt.show()

**Practical no. 8**

**Aim:-:Write An Application To Implement Clustering Algorithm.**

**Code:-**

﻿# synthetic classification dataset

From numpy import where

From sklearn.datasets import make\_classification

From matplotlib import pyplot

# define dataset

X, y = make\_classification (n\_samples-1000, n\_features=2, n\_informative-2, n\_redundant=0,

N\_clusters\_per\_class=1, random\_state=4)

# create scatter plot for samples from each class

For class\_value in range(2):

# get row indexes for samples with this class

Row\_ix = where(y == class\_value)

# create scatter of these samples

Pyplot.scatter (X[row\_ix, 0], X[row\_ix, 1])

# show the plot pyplot.show()

**PRACTICAL: 9**

**AIM:WriteAnProgramToImplement BFSAlgorithm.**

**Code:-**

Import collections

# BFS algorithm

Def bfs (graph, root):

Visited, queue set(), collections.deque([root])

Visited.add(root)

While queue:

Vertex =

Queue.popleft()

Print(str(vertex) + “”, end=””)

For neighbour in graph[vertex]: if neighbour not in visited: visited.add(neighbour) queue.append(neighbour)

If name

Graph=\_\_main\_\_’:{0: [1, 2], 1: [2], 2: [3], 3: [1, 2]} print(“Following is Breadth First Traversal: “)

bfs (graph, 0)

**PRACTICAL: 10**

**AIM:WriteAnProgramToImplement DFS Algorithm.**

**Code:-**

**#** DFS algorithm

Def dfs(graph, start, visited=None):

If visited is None:

Visited set()

Visited.add(start)

Print(start)

For next in graph[start] – visited: dfs(graph, next, visited)

Return visited

=

Graph ‘0’: set([‘1’, ‘2’]),

‘1’: set([‘0’,

‘2’: set([‘0’]),

‘3’: set([‘1’]),

‘3’, ‘4’]),

‘4’: set([‘2’, ‘3’])}

Dfs(graph, ‘0’)