|  |  |  |  |
| --- | --- | --- | --- |
| S.No. |  | **Page No.** | **Teacher’s sign** |
| 1. | W.A.P. for Family Tree | 1-2 |  |
| 2. | W.A.P. for Counter Program | 3 |  |
| 3. | W.A.P. using List Functions | 4-8 |  |
| 4. | W.A.P. for Route Finding Problem | 9 |  |
| 5. | W.A.P. for Tower of Hanoi | 10 |  |
| 6. | W.A.P. to solve the Monkey Banana Problem | 11-12 |  |
| 7. | W.A.P. for Grammar Problem | 13-14 |  |
| 8. | W.A.P. to Check Identifiers | 15-17 |  |
| 9. | W.A.P. for Finite State Automata | 18-20 |  |

**ML lab Index:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No**. |  | **Page No.** | **Teacher’s sign** |
| 1. | W.A.P. for Print Pattern | 21 |  |
| 2. | W.A.P. to Compute Grades | 22-23 |  |
| 3. | W.A.P. for String Functions | 23-26 |  |
| 4. | W.A.P. for Discount | 26-27 |  |
| 5. | W.A.P. for Dictionary | 27 |  |
| 6. | W.A.P. for Students Records | 28-29 |  |
| 7. | W.A.P. Using List,Set and Tuple Functions | 29-32 |  |
| 8. | W.A.P. for Statistical Linear Regression | 32-34 |  |
| 9. | W.A.P. for KNN | 35-36 |  |
| 10. | W.A.P. for Linear Regression MSE | 37-39 |  |
| 11. | W.A.P. for KNN(Loop) | 40-41 |  |
| 12. | W.A.P. for KMeans Clustering | 42-43 |  |
| 13. | W.A.P. for Linear Regression Tensorflow | 44-46 |  |
| 14. | W.A.P. Support Vector Machine(SVM) | 47-49 |  |
| 15. | W.A.P. Decision Tree | 50-53 |  |

**AI Lab Programs:**1

PROGRAM -1

**W.A.P. for Family Tree:**

**INPUT:**

domains

name=symbol

predicates

male(name)

female(name)

husband(name , name)

wife(name , name)

father(name , name)

son(name , name)

daughter(name , name)

brother(name , name)

sister(name , name)

mother(name , name)

grandfather(name , name)

grandmother(name , name)

cousin(name , name)

sibling(name , name)

Clauses

male(jai).

male(govind).

male(krishna).

male(shiv).

male(arjun).

male(parth).

male(ganesh).

female(kavita).

female(rukmani).

female(radha).

female(parvati).

female(divya).

female(mansi).

father(jai , govind).

father(jai , krishna).

father(jai , parvati).

father(govind , arjun).

father(govind , divya). 2

father(krishna , parth).

father(shiv , ganesh).

father(shiv , mansi).

husband(jai , kavita).

husband(govind , rukmani).

husband(krishna , radha).

husband(shiv , parvati).

wife(X,Y):-husband(Y,X),female(X),male(Y).

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

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

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

sister(X,Y):-female(X),father(Z,X),father(Z,Y).

mother(X,Y):-female(X),wife(X,Z),father(Z,Y).

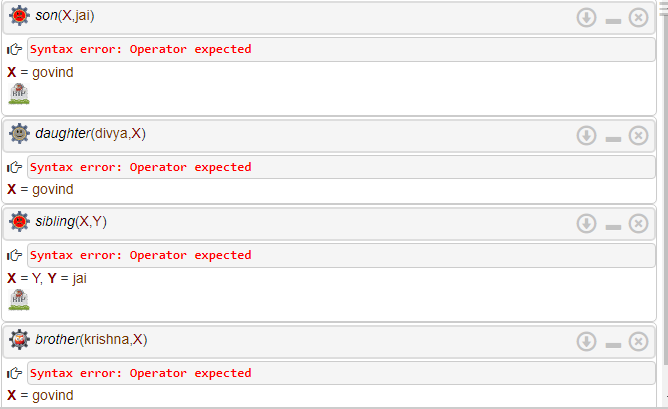
grandfather(X,Y):-male(X),father(X,Z),father(Z,Y).

grandmother(X,Y):-female(X),mother(X,Z),father(Z,Y),mother(Z,Y).

sibling(X,Y):-father(X,Z),father(Y,Z).

cousin(X,Y):-grandfather(X,Z),grandfather(Y,Z).

**OUTPUT:**



.

.

PROGRAM -23

**W.A.P. for Counter Program:**

**INPUT:**

domains

n = integer

predicates

count\_to\_n(n)

clauses

count\_to\_n(10):-

write(10),nl.

count\_to\_n(X):-

write(X),nl,

Y = X+1,

count\_to\_n(Y).

**Output:-**



PROGRAM -34

**W.A.P. for List Functions:**

**INPUT:**

domains

character = symbol

list = character\*

n = integer

predicates

member(character,list)

length(list,n)

concate(list,list,list)

delete(character,list,list)

append(character,list,list)

list\_insert(character,list,list)

permut(list,list)

reverse\_list(list,list)

shift(list,list)

order(list)

subset(list,list)

union(list,list,list)

intersec(list,list,list)

partition(list,list,list)

mergesort(list,list)

merge(list,list,list)

clauses

member(X,[X|\_]).

member(X,[\_|T]):-

member(X,T).

length([],0).

length([\_|T],N):- 5

length(T,N1),

N = N1+1.

concate([],L,L).

concate([X1|L1],L2,[X1|L3]):-

concate(L1,L2,L3).

delete(X,[X],[]).

delete(X,[X|L],L).

delete(X,[Y|L2],[Y|L1]):-

delete(X,L2,L1).

append(A,T,T):-

member(A,T).

append(A,T,[A|T]).

list\_insert(X,L,R):-

delete(X,R,L).

permut([],[]).

permut(L,[X|P]):-

delete(X,L,L1),

permut(L1,P).

reverse\_list([],[]).

reverse\_list([H|T],Rev):-

reverse\_list(T,R),

concate(R,[H],Rev).

shift([H|T],S):-

concate(T,[H],S).

6

order([X]).

order([X,Y|Tail]):-

X <= Y,

order([Y|Tail]).

subset([],[]).

subset([H|T],[H|Sub]):-

subset(T,Sub).

subset([\_|T],Sub):-

subset(T,Sub).

union([],Z,Z).

union([X|Y],Z,W):-

member(X,Z),

union(Y,Z,W).

union([X|Y],Z,[X|W]):-

not(member(X,Z)),

union(Y,Z,W).

intersec([],Z,[]).

intersec([X|Y],Z,[X|W]):-

member(X,Z),

intersec(Y,Z,W).

partition([],[],[]).

partition([X],[X],[]).

partition([X,Y|T],[X|L1],[Y|L2]):-

partition(T,L1,L2).

mergesort([],[]).

mergesort([X],[X]).

mergesort([A,B|R],S):-

partition([A,B|R],L1,L2),

7

mergesort(L1,S1),

mergesort(L2,S2),

merge(S1,S2,S).

merge(A,[],A).

merge([],B,B).

merge([A|RA],[B|RB],[A|M]):-

A<=B,

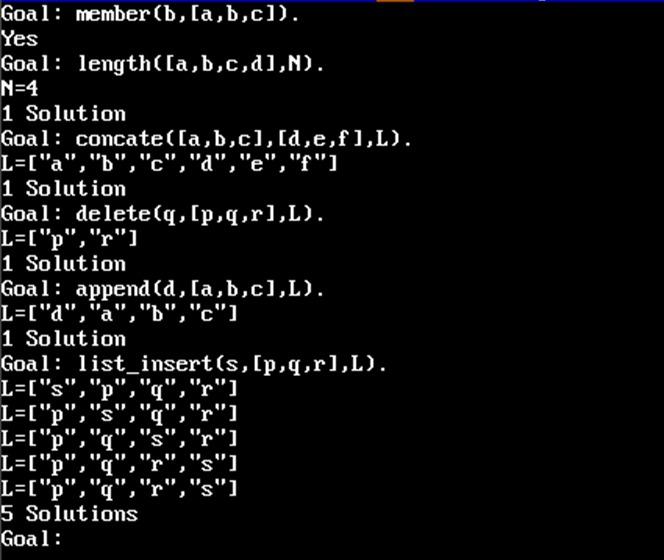
merge(RA,[B|RB],M).

merge([A|RA],[B|RB],[B|M]):-

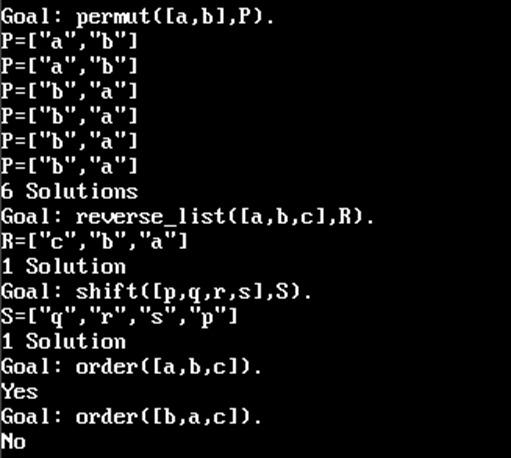
B<=A,

merge([A|RA],RB,M).

**Output:-**

****

8





9PROGRAM -4

**W.A.P. for Route Finding Problem:**

**INPUT:**

road(noida,delhi,20).

road(delhi,rohtak,50).

road(delhi,faridabad,30).

road(faridabad,palwal,35).

road(faridabad,gurugram,45).

road(palwal,sohna,25).

road(sohna,gurugram,40).

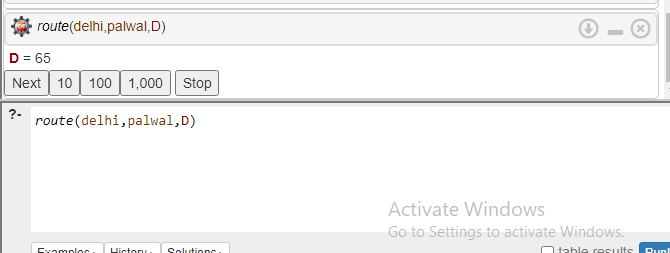
road(gurugram,rohtak,55).

route(X,Y,D):-road(X,Y,D).

route(X,Y,D):-road(X,Z,D1),

route(Z,Y,D2),

D is D1+D2.

**Output:-**

PROGRAM -510

**W.A.P. for Tower of Hanoi:**

**INPUT:**

move(1,X,Y,\_) :-

write('Move top disk from '), write(X), write(' to '), write(Y), nl.

move(N,X,Y,Z) :-

N>1,

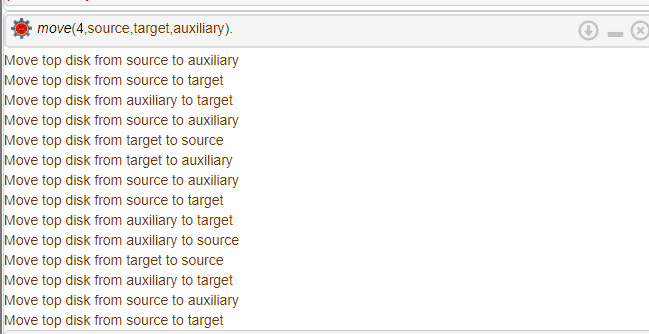
M is N-1,

move(M,X,Z,Y),

move(1,X,Y,\_),

move(M,Z,Y,X)

**Output:-**



PROGRAM -611

**W.A.P for Monkey Banana Problem:**

**INPUT:**

move(state(middle,onbox,middle,hasnot),

grasp,

state(middle,onbox,middle,has)).

move(state(P,onfloor,P,H),

climb,

state(P,onbox,P,H)).

move(state(P1,onfloor,P1,H),

drag(P1,P2),

state(P2,onfloor,P2,H)).

move(state(P1,onfloor,B,H),

walk(P1,P2),

state(P2,onfloor,B,H)).

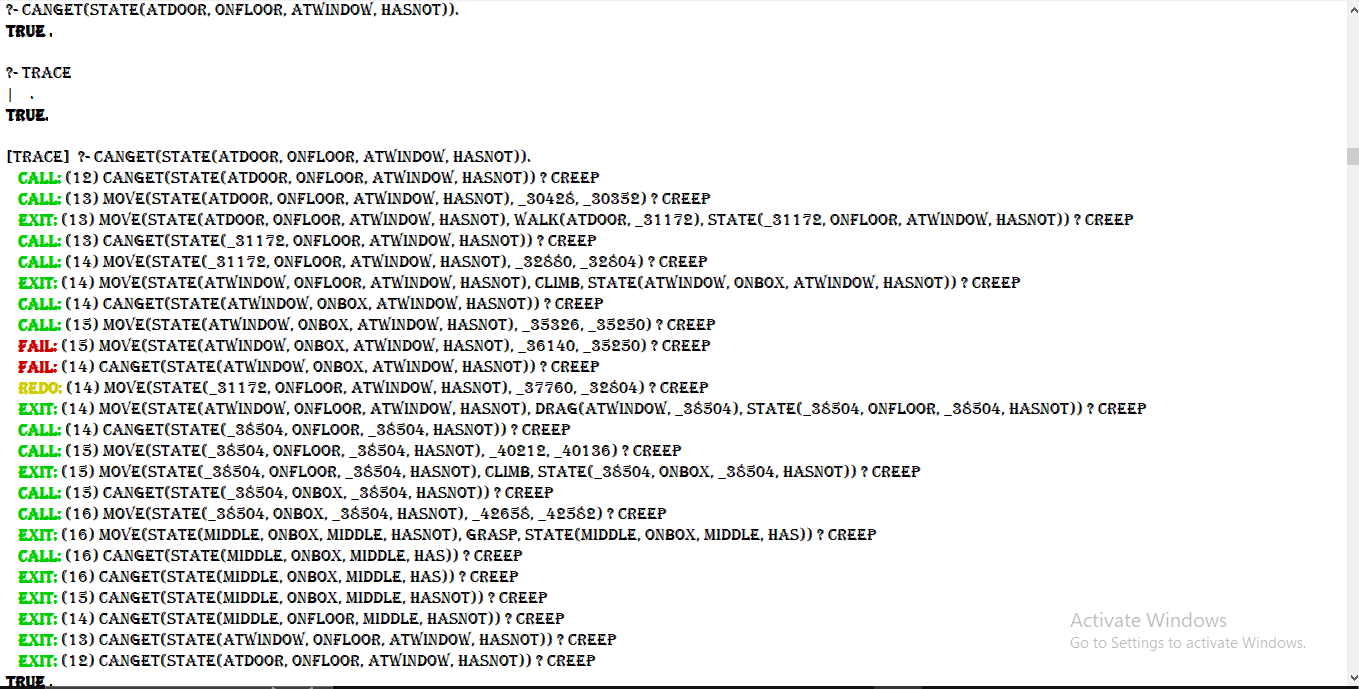
canget(state(\_,\_,\_,has)).

canget(State1) :-

move(State1,\_,State2),

canget(State2).

Output *:-* 12



.

PROGRAM -713

**W.A.P. to solve Grammer Problem:**

**INPUT:**

% Definite clause grammar

advp(A, Z) :- adv(A, Z).

advp(A, Z) :- adv(A, B), advp(B, Z).

adv([quickly | X], X).

adv([very | X], X).

s(A, Z) :- np(A, B), vp(B, Z).

s(A, Z) :- np(A, B), vp(B, C), np(C, Z).

s(A, Z) :- np(A, B), vp(B, C), np(C, D), np(D, Z).

np(A, Z) :- n(A, Z).

np(A, Z) :- dp(A, B), n(B, Z).

dp(A, Z) :- d(A, Z).

vp(A, Z) :- v(A, Z).

vp(A, Z) :- advp(A, B), v(B, Z).

n([he | X], X).

n([her | X], X).

n([book | X], X).

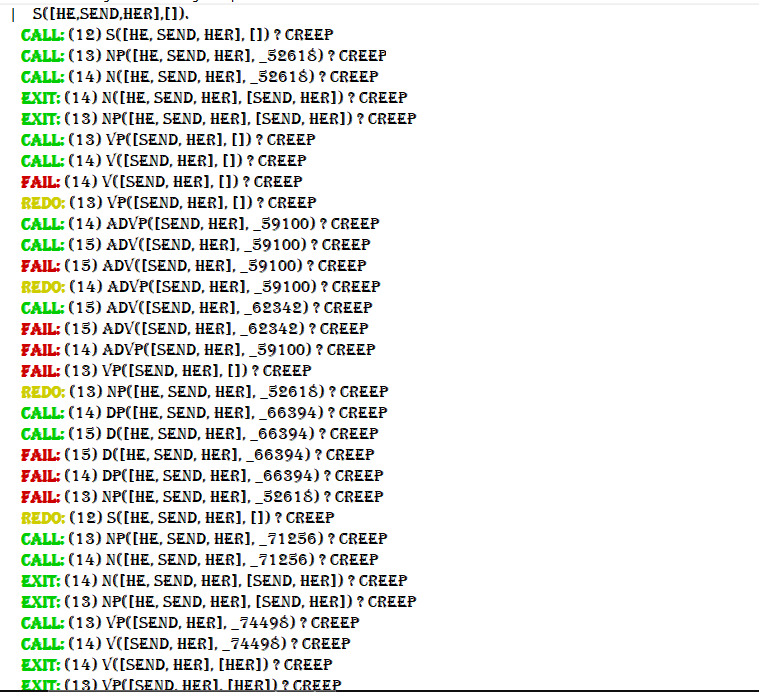
d([a | X], X).

d([the | X], X).

d([an | X], X).

v([send | X], X).

Output *:-*14



.

PROGRAM -815

**W.A.P. to check identifiers:**

**INPUT:**

domains

var = char

s = integer

list = var\*

predicates

delta(s,var,s)

delta(s,var,s)

start(s)

final(s)

dfa(list)

trans(s,list)

isletter(var)

isdigit(var)

clauses

delta(0,A,1):-

isletter(A).

delta(1,A,1):-

isletter(A) or isdigit(A).

isletter(A):-

A >= 'A' and A <= 'Z' or

A >= 'a' and A <= 'z' or

16

A = '\_'.

isdigit(A):-

A >= '0' and A <= '9'.

start(0).

final(1).

dfa(L):-

start(S),

trans(S,L).

trans(X,[A|B]):-

delta(X,A,Y),

write(X),

write(' '),

write(A),

nl,

trans(Y,B).

trans(X,[]):-

final(X),

write(X),

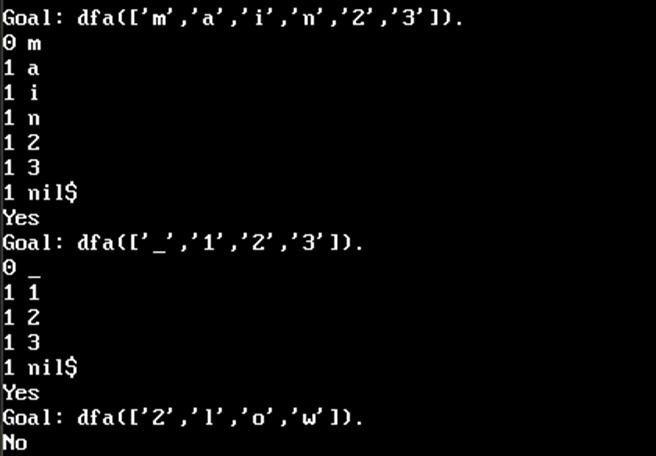
write(' '),

write([]),

nl.

17

**Output:**

****

PROGRAM -918

**W.A.P. for Finite State Automata:**

**INPUT:**

start(0).

final(1).

dfa(L):-

start(S),

trans(S,L).

trans(X,[A|B]):-

delta(X,A,Y),

write(X),

write(' '),

write(A),

nl,

trans(Y,B).

trans(X,[]):-

final(X),

write(X),

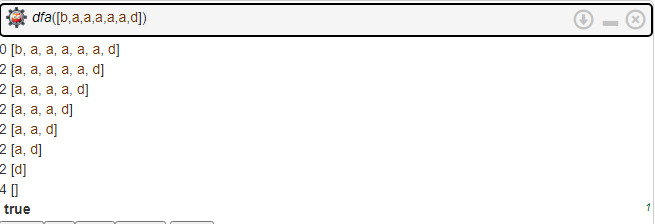
write(' '),

write([]),

Nl.

20

**Output:-**

****

.

.

**ML programs:**21

PROGRAM -1

**W.A.P for Print Pattern:**

**INPUT:**

num=int(input("enter the no of symbol:"))

num\_of\_row=(num//2) + 1

s=(num//2)

e=(num//2 )

for i in range(0,num+2):

for j in range(0,num):

if i==0 or i==num+1:

print("\*", end = "")

else:

if(j<=s or j>=e):

print("\*", end = "")

else:

print(" ", end = "")

if i<num\_of\_row:

s=s-1

e=e+1

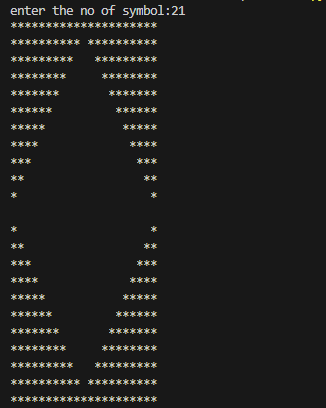
else:

s=s+1

e=e-1

print( )

**Output:-**



PROGRAM -2 22

**W.A.P. to Compute Grades:**

**INPUT:**

deffind\_grade(percentage):

if percentage >= 90:

grade = 'A'

elif percentage >= 80:

grade = 'B'

elif percentage >= 70:

grade = 'C'

elif percentage >= 60:

grade = 'D'

else:

grade = 'F'

return grade

deffind\_percentage(marks):

total\_marks = sum(marks)

total\_subjects = len(marks)

percentage = (total\_marks / (500 )) \* 100

return percentage

# Get marks of 5 subjects

subject1 = int(input("Enter marks of Subject 1: "))

subject2 = int(input("Enter marks of Subject 2: "))

subject3 = int(input("Enter marks of Subject 3: "))

subject4 = int(input("Enter marks of Subject 4: "))

subject5 = int(input("Enter marks of Subject 5: "))

# Calculate the percentage

percentage = find\_percentage([subject1, subject2, subject3, subject4, subject5])

# Find the grade

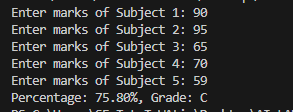
grade = find\_grade(percentage)

# Print the result

print("Percentage: {:.2f}%, Grade: {}".format(percentage, grade))

23

**Output:-**



.

.

PROGRAM -3

**W.A.P for String Functions:**

**INPUT:**

message='hello sheetal'

print(message)

print(message[6])

print(message[-12])

print(len(message))

print(message[len(message)-1])

print(message[len(message)-13])

print(message[-len(message)])

n\_message='hello'+' sir'

print(n\_message)

print(max('AZ','C','BZ'))

print(max('AZ','C','bz'))

print(message[0:5])

print(message[-10:-5])

print(message[:5])

print(message[5:])

print(message[5:None])

print(n\_message[5:])

#.....membership..........#

print('h' in message)

print('H' in message)

print(message.count("l"))

#...... vowels.........#

vowels=" a e i o u"

v\_count=0

for ch in vowels:

v\_count+='Encyolopedia'.count(ch)

print(v\_count)

#.......find and refind.....#

colour='red green yellow grey'

print(colour.find ('red'))

print(colour.find ('green'))

print(colour.rfind('yellow'))

24

#............functions...............#

print(message.capitalize())

print(message.upper())

print(message.lower())

print(message.title())

print(message.swapcase())

print(message.isupper())

print(message.islower())

print(message.istitle())

print(message.replace('hel','leh'))

print(message.strip())

print(message.lstrip())

print(message.rstrip())

#.....split and partition....#

print(colour.split('r'))

print(colour.split('e'))

print(message.partition(','))

print(colour.partition('e'))

#......join........#

print(' '.join(['I','am','ok']))

print('>'.join(['I','am','ok']))

#......function......#

print(message.isalpha())

print(message.isdigit())

print(message.isalnum())

print(message.isspace())

name=input("Enter your name: ")

print(name.isalpha())

#...... ......#

print(message.startswith('H'))

print(message.startswith(' '))

print(message.endswith('i'))

#.......encode decode.....#

str='pen'.encode('utf')

print(str.decode('utf'))

#..ques=counting th no. of matching characters in a pair of string....#

count=0

for char in message:

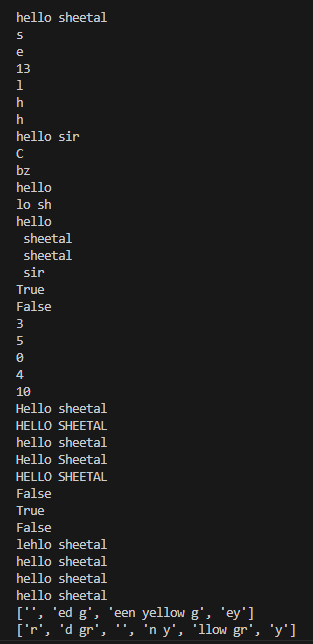
if char in n\_message:

25

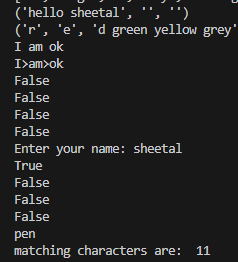
count+=1

print("matching characters are: ",count)

**Output:-**



26



.

. PROGRAM -4

**W.A.P. for Discount:**

**Input:-**

defcalculate\_final\_price(original\_price,discount\_percentage):

discount\_amount=original\_price\*(discount\_percentage/100)

final\_price=original\_price-discount\_amount

return final\_price,discount\_amount

defmain():

try:

original\_price=float(input("enter the original price:"))

discount\_percentage=float(input("Enter the percentage of discount:"))

final\_price,discount\_amount=calculate\_final\_price(original\_price,discount\_percentage)

print("\n original proce:",original\_price)

print("\n discount percentage:",discount\_percentage,"%")

print("\n discount amount:",discount\_amount)

print("\n final price:",final\_price)

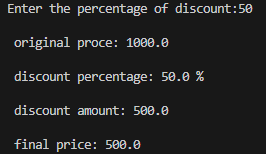
except ValueError:

print("please enter valid numeric values for the original price and discount percentage")

if \_\_name\_\_=="\_\_main\_\_":

main()

**Output:-**27

****

PROGRAM -5

**W.A.P. for Dictionary:**

**Input:-**

num\_name\_month={1:'Jan',2:'Feb',3:'March'}

print("1. ",num\_name\_month.keys())

print("2. ",num\_name\_month.values())

print("3. ",num\_name\_month.items())

more\_months={4:'April',5:'May',6:'June'}

num\_name\_month.update(more\_months)

print("4. After Update:",num\_name\_month)

print("5. After Update:",num\_name\_month.items())

del num\_name\_month[4]

print("6. After deletion:",num\_name\_month)

dec={1:"priya",2: "Rahul"}

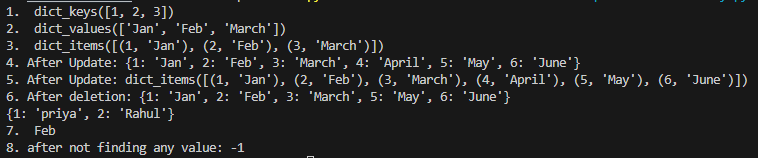
print(dec)

del dec

print("7. ",num\_name\_month.get(2,-1))

print("8. after not finding any value:",num\_name\_month.get(4,-1))

**Output:-**

****

PROGRAM -628

**W.A.P. for Student Records:**

**Input:-**

students=[['Sheetal','52',['AI','Java','cloud','SoftEngg']],

['Priyanka','44',['Python','Data Science']],

['vaishali','60',['C++','Web Dev']],

['Piyush','49',[ 'Android App Development' ]]]

print(students)

for student in students:

print([student[0], student[1], student[2]])

# Adding a new subject to the first student's list of subjects

students[0][2].append('Data Structures')

# Printing the updated list of students

print(students)

new\_student = ['Raj', '50', ['Python', 'Java']]

students.insert(4, new\_student)

print(students)

# Removing the last student from the list

removed\_student = students.pop()

# Printing the removed student and the updated list of students

print("Removed student:", removed\_student)

print("Updated list of students:", students)

# Adding a new student to the list using extend function

new\_student = ['Piyush','49',[]]

students.extend([new\_student])

# Printing the updated list of students

print(students)

# Sorting the students list in reverse order based on roll numbers

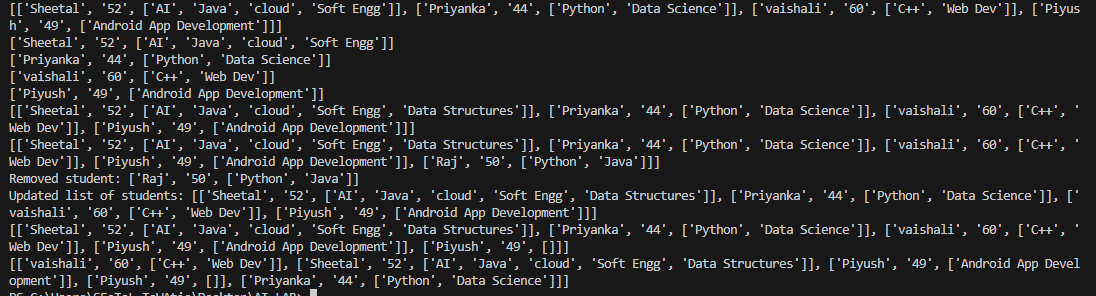
29

students.sort(key=lambda x: x[1], reverse=True)

# Printing the sorted list of students

print(students)

**Output:-**

****

PROGRAM -7

**W.A.P. using List,Set and Tuple Functions:**

**INPUT:**

cube=lambda x:x\*\*3

print(cube(3))

list1=[1,2,3,4,5]

list(map(cube,list1))

print(list(map(lambda x:x\*\*3,list1)))

list2=[1,2,3,4,5,6,7,8,9,10]

print(list(filter(lambda x:x%2==0,list2)))

30

# sets

vowels={'a','e', 'i', 'o', 'u'}

for v in vowels:

print(v,end=' ')

# set functions

s1={1,2,3}

s2={4,5,6}

s1.add(10)

print(s1)

s1.update(s2)

print(s1)

s1.remove(2)

print(s1)

x=s1.pop()

print(x)

y=s1.union(s2)

print(y)

z=s1.intersection(s2)

print(z)

31

p=s1.difference(s2)

print(p)

digits={1,2,3,4,5,6,7,8,9,10}

new\_digits=digits.copy()

print(new\_digits)

#superset

print(s1>=s2)

#subset

print(s1<=s2)

vowels1='aeiou'

vow\_tup=tuple(vowels1)

print(vow\_tup)

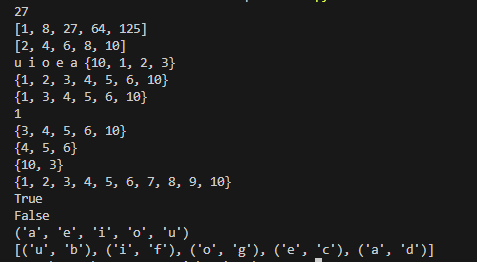
tup1={'a','e','i','o','u'}

tup2={'b','c','d','f','g'}

print(list(zip(tup1,tup2)))

32

**Output:-**

****

.

.

PROGRAM -8

**W.A.P. for Statistical Linear Regression:**

**INPUT:**

import numpy as np

import matplotlib.pyplot as plt

defest\_coff(x, y):

n = np.size(x)

mx, my = np.mean(x), np.mean(y)

ss\_xy = np.sum(y \* x) - n \* mx \* my

ss\_xx = np.sum(x \* x) - n \* mx \* mx

b1 = ss\_xy / ss\_xx

b0 = my - b1 \* mx

return b0, b1

defplot\_reg\_line(x, y, b):

plt.scatter(x, y, color='m', marker='d', s=60)

y\_pred = b[0] + b[1] \* x

plt.plot(x, y\_pred, color='g')

33

plt.xlabel("X")

plt.ylabel("Y")

plt.show()

defmain():

test\_data\_size=50

train\_x=[]

train\_y=[]

for n in range(test\_data\_size):

tx=n

ty=n\*0.5+1/np.random.rand()

train\_x.append([tx])

train\_y.append([ty])

x=np.array(train\_x)

y=np.array(train\_y)

print(type(x))

print(type(y))

b = est\_coff(x, y)

print(x)

print(y)

print("Estimated coefficients:")

print("b0 = {}\nb1 = {}".format(b[0], b[1]))

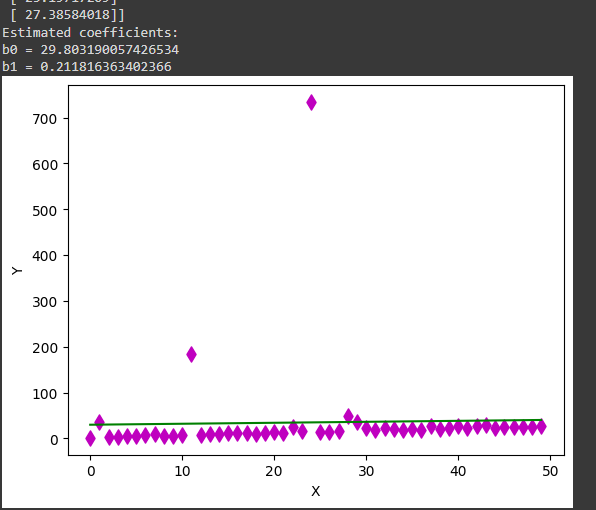
plot\_reg\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

main()

34

**Output:-**

****

35

PROGRAM -9

**W.A.P. for KNN:**

**INPUT:**

# import necessary libraries

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.datasets import load\_iris

#loading the data

irisData = load\_iris()

#create feature vector and target

X = irisData.data

Y = irisData.target

print(X)

print(Y)

#splitting the dataset

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size = 0.3, random\_state = 42)

#model creation

knn = KNeighborsClassifier(n\_neighbors = 7)

#fit the data

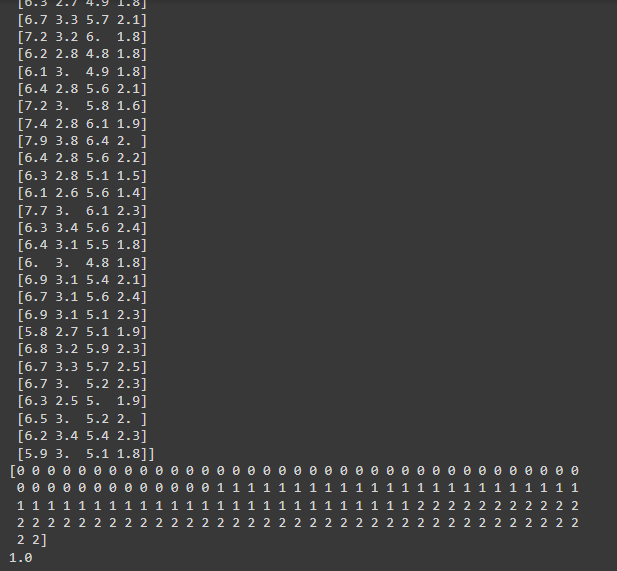
knn.fit(X\_train,Y\_train)

#calculating accuracy

print(knn.score(X\_test,Y\_test))

36

**OUTPUT:**

****

PROGRAM -1037

**W.A.P. for Linear Regression MSE:**

**INPUT:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

import requests

from io import StringIO

# Define the URL

data\_url = "http://lib.stat.cmu.edu/datasets/boston"

# Fetch the data from the URL

response = requests.get(data\_url)

# Check if the request was successful

if response.status\_code == 200:

# Read the content of the response into a pandas DataFrame

raw\_data = pd.read\_csv(StringIO(response.text), sep="\s+", skiprows=22, header=None)

# Extract features and target variable

X = np.hstack([raw\_data.values[::2, :], raw\_data.values[1::2, :2]])

Y = raw\_data.values[1::2, 2]

# Split the data into training and testing sets with a fixed random state

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0)

# Initialize the Linear Regression model

38

reg = LinearRegression()

# Training the model

reg.fit(X\_train, Y\_train)

# Extracting coefficients

coefficients = reg.coef\_

# Printing the coefficients

print("Coefficients:", coefficients)

print("Bias/Intercept:", reg.intercept\_)

print("Variance Score:", reg.score(X\_test, Y\_test))

# Plotting of residual error

plt.style.use('fivethirtyeight')

# Plotting residual errors

plt.scatter(reg.predict(X\_train), reg.predict(X\_train) - Y\_train,

color="green", s=10, label="Train Data")

plt.scatter(reg.predict(X\_test), reg.predict(X\_test) - Y\_test,

color="blue", s=10, label="Test Data")

# Adding a horizontal line at y=0

plt.axhline(y=0, color='r', linestyle='-', linewidth=3)

# Adding labels and legend

plt.legend(loc='upper right')

plt.title("Residual error")

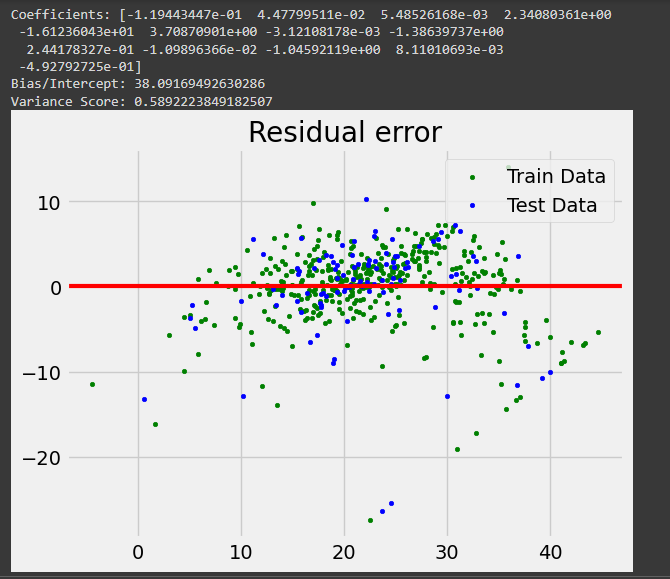
plt.show()

else:

print("Failed to fetch the data from the URL.")

39

**OUTPUT:**

****

PROGRAM -1140

**W.A.P. for KNN(Loop):**

**INPUT:**

#import necessary modules

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.datasets import load\_iris

import numpy as np

import matplotlib.pyplot as plt

irisData=load\_iris()

X=irisData.data

y=irisData.target

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42)

neighbors=np.arange(1,9)

train\_acc=np.empty(len(neighbors))

test\_acc=np.empty(len(neighbors))

#loop

for i,k in enumerate(neighbors):

knn=KNeighborsClassifier(n\_neighbors=1)

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

train\_acc[i]=knn.score(X\_train,y\_train)

test\_acc[i]=knn.score(X\_test,y\_test)

#plot graph

plt.plot(neighbors,test\_acc,label='Test Acc')

plt.plot(neighbors,train\_acc,label='Train Acc')

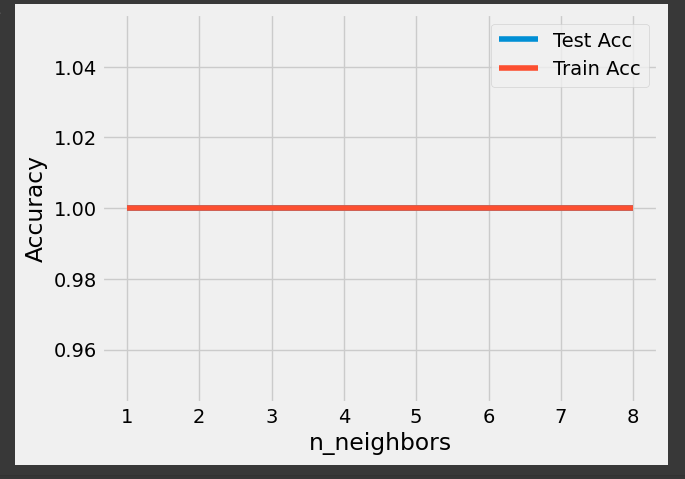
plt.legend()

plt.xlabel('n\_neighbors')

plt.ylabel('Accuracy')

plt.show()

**OUTPUT:** 41

****

PROGRAM -1242

**W.A.P. for KMeans Clustering:**

**INPUT:**

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

#make blobs

X,y=make\_blobs(n\_samples=150,n\_features=2,centers=4,cluster\_std=0.5,shuffle=True,random\_state=0)

#plot

plt.scatter(X[:,0],X[:,1],c='white',marker='o',edgecolor='black',s=50)

plt.show(block=False)

from sklearn.cluster import KMeans

km=KMeans(n\_clusters=4,init='random',n\_init=10,max\_iter=300,tol=1e04,random\_state=0)

y\_km=km.fit\_predict(X)

plt.scatter(X[y\_km==0,0],X[y\_km==0,1],s=50,c='lightgreen',marker='s',edgecolor=

'black',label='Cluster 1')

plt.scatter(X[y\_km==1,0],X[y\_km==1,1],s=50,c='orange',marker='o',edgecolor=

'blue',label='Cluster 2')

plt.scatter(X[y\_km==2,0],X[y\_km==2,1],s=50,c='lightblue',marker='v',edgecolor=

'yellow',label='Cluster 3')

plt.scatter(X[y\_km==3,0],X[y\_km==3,1],s=50,c='red',marker='d',edgecolor=

'pink',label='Cluster 4')

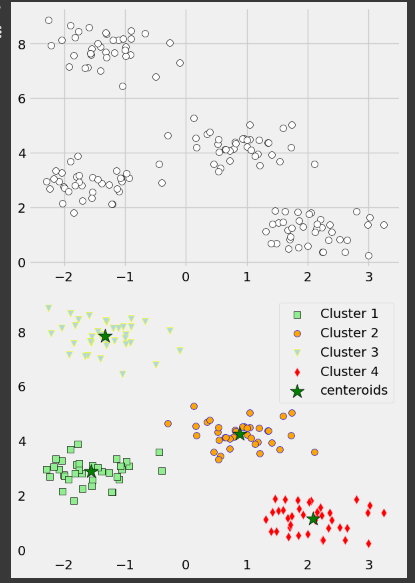
plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1],s=250,marker='\*',c='green',edgecolor='black',label='centeroids')

plt.legend(scatterpoints=1)

plt.grid()

plt.show()

**OUTPUT:** 43

****

PROGRAM -1344

**W.A.P. for Linear Regression Tensorflow:**

**INPUT:**

### Importing required libraries

import tensorflow.compat.v1 as tf

tf.disable\_v2\_behavior()

import numpy as np

import matplotlib.pyplot as plt

### Defining data size, no. of iterations and learning rate

test\_data\_size = 2000

iterations = 10000

learn\_rate = 0.005

### Creating a function to generate test values randomly

defgenerate\_test\_values():

train\_x = []

train\_y = []

for \_ in range(test\_data\_size):

x1 = np.random.rand()

x2 = np.random.rand()

x3 = np.random.rand()

y\_f = 2\*x1 + 3\*x2 + 7\*x3 + 4

train\_x.append([x1,x2,x3])

train\_y.append(y\_f)

return np.array(train\_x),np.transpose([train\_y])

### Defining the placeholders and variables

x = tf.placeholder(tf.float32,[None,3],name = 'x')

w = tf.Variable(tf.zeros([3,1]),name = 'w')

b = tf.Variable(tf.zeros([1]),name = 'b')

y = tf.placeholder(tf.float32,[None,1])

model = tf.add(tf.matmul(x,w),b) #Applying the model using y = x\*w + b

cost = tf.reduce\_mean(tf.square(y - model)) ## calculating the cost (loss function)

45

train = tf.train.GradientDescentOptimizer(learn\_rate).minimize(cost)

#using gradient descent optimizer to minimize cost function

init = tf.global\_variables\_initializer() #initializing the variables

### Generating the training and test values

train\_dataset,train\_value = generate\_test\_values()

### Creating the session and feeding the data

costs=[]

with tf.Session() as session:

session.run(init)

for \_ in range(iterations):

session.run(train,feed\_dict = {

x:train\_dataset,

y:train\_value

})

costs.append(session.run(cost,feed\_dict = {x:train\_dataset,y:train\_value}))

print("Cost = {}".format(session.run(cost,feed\_dict = {

x:train\_dataset,

y:train\_value

})))

### Plot graph between the no. of iterations and the cost

x\_values = np.array([i for i in range(iterations)])

y\_values = np.array(costs)

plt.plot(x\_values,y\_values)

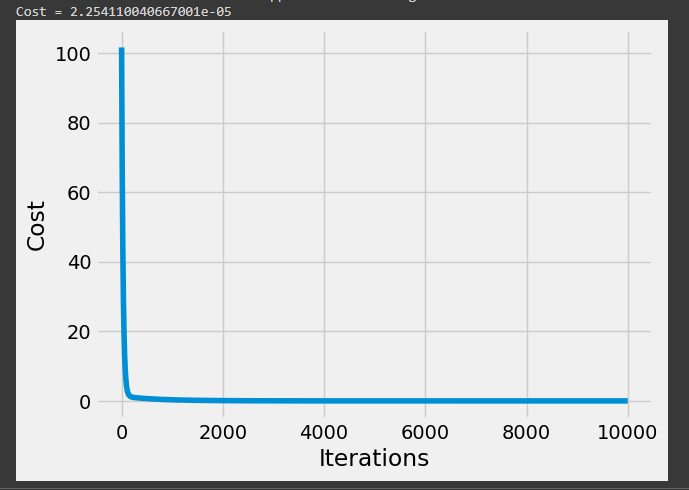
plt.xlabel("Iterations")

plt.ylabel("Cost")

plt.show()

46

**OUTPUT:**

****

PROGRAM -1447

**W.A.P. for Support Vector Machine:**

**INPUT:**

#Import scikit-learn dataset library

from sklearn import datasets

#Load dataset

cancer = datasets.load\_breast\_cancer()

#Exploring the data

# 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)

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']

# print data(feature)shape

cancer.data.shape

(569, 30)

# print the cancer data features (top 5 records)

print(cancer.data[0:5])

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

print(cancer.target)

48

#Splitting Data

# Import train\_test\_split function

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

# 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

#Generating Model

#Import svm model

from sklearn import svm

#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)

#Evaluating the Model

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# 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?

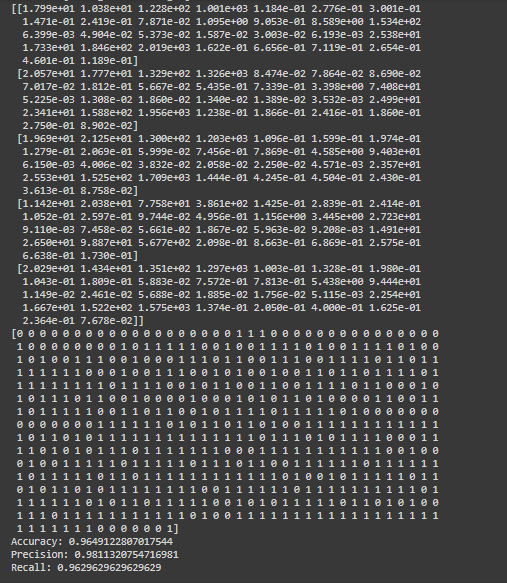
print("Precision:",metrics.precision\_score(y\_test, y\_pred))

49

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

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

**OUTPUT:**

****

PROGRAM -1550

**W.A.P. for Decision Tree:**

**INPUT:**

# Importing modules

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

import sklearn.datasets as datasets

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve

from sklearn.tree import plot\_tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# Loading the dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

df = pd.read\_csv(url, header=None, names=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',

'petal width (cm)', 'Species'])

# Printing the first 10 rows

df.head(10)

# Shape of the dataset

df.shape

print("Rows : ", df.shape[0])

51

print("COlumns : ", df.shape[1])

# Check the dataframe information

df.info()

# Counting null values

df.isnull().sum()

# Summary statistics

df.describe()

# Finding the outliers in the dataset

columns = df.columns[0 : -1]

for i in columns:

sns.boxplot(y = df[i])

plt.show()

# Removing outliers from 'sepal width (cm)'

q1 = df['sepal width (cm)'].quantile(0.25)

q3 = df['sepal width (cm)'].quantile(0.75)

iqr = q3 - q1

df = df[(df['sepal width (cm)'] >= q1-1.5\*iqr) & (df['sepal width (cm)'] <= q3+1.5\*iqr)]

df.shape # To find out the number of rows and column after outlier treatment

# Blocplot for sepal width (cm) after outlier treatment

sns.boxplot(y=df['sepal width (cm)'])

plt.show()

# Splitting the data into train and test sets

X = df.drop("Species",axis=1)

y = df["Species"]

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3, random\_state= 1)

# Defining an object for DTC and fitting for whole dataset

dt = DecisionTreeClassifier(max\_depth=3, min\_samples\_leaf=10, random\_state=1 )

dt.fit(X, y)

# Plotting of decission tree

from IPython.display import Image

from sklearn.tree import export\_graphviz

52

!pip install pydotplus

import pydotplus

features = X.columns

dot\_data = export\_graphviz(dt, out\_file=None, feature\_names=features)

graph = pydotplus.graph\_from\_dot\_data(dot\_data)

Image(graph.create\_png())

# Defining an object for DTC and fitting for train dataset

dt = DecisionTreeClassifier(random\_state=1)

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

y\_pred\_train = dt.predict(X\_train)

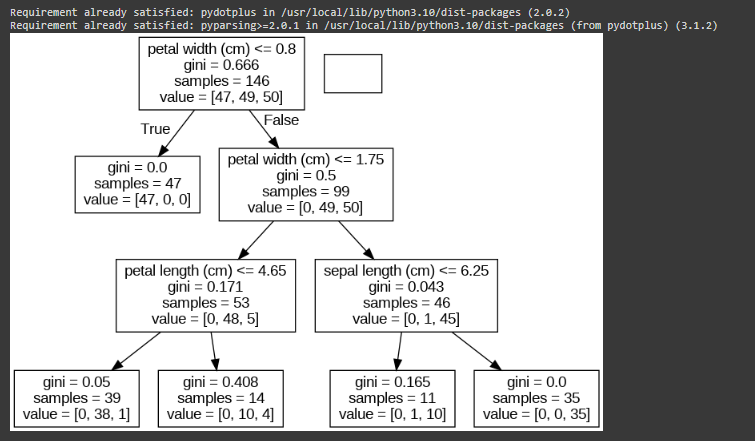
y\_pred = dt.predict(X\_test)

y\_prob = dt.predict\_proba(X\_test)

print('Accuracy of Decision Tree-Train: ', accuracy\_score(y\_pred\_train, y\_train))

print('Accuracy of Decision Tree-Test: ', accuracy\_score(y\_pred, y\_test))

**OUTPUT: 53**

****

****