Artificial Intelligence

krunal dhavle
TYCS-713

Name : Krunal Dhavle

Class : TYCS

Roll No : 713

Subject : Artificial Intelligence Practicals

Teacher : Manali Patil Maam

INDEX

Date: 24/08/2020

Practical no 1

AIM: Implement Breadth first search algorithm for Romanian map problem or any other map.

CODE:-

```
from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph = defaultdict(list)
    def addEdge(self,u,v):
        self.graph[u].append(v)
    def BFS(self, s):
        visited = [False] * (len(self.graph))
        queue = []
        queue.append(s)
        visited[s-1] = True
        while queue:
            s = queue.pop(0)
            print (s, end = " ")
            for i in self.graph[s]:
                if visited[i-1] == False:
                    queue.append(i)
                    visited[i-1] = True
g = Graph()
g.addEdge(1, 2)
g.addEdge(1, 3)
g.addEdge(2, 1)
g.addEdge(2, 4)
g.addEdge(2, 5)
g.addEdge(3, 1)
g.addEdge(3, 5)
g.addEdge(4, 2)
g.addEdge(4, 6)
g.addEdge(5, 2)
g.addEdge(6, 5)

print ("Following is Breadth First Traversal"
" (starting from vertex 1)")
g.BFS(1)

print("\nperformed by krunal 713")
```

OUTPUT:-

```
==================== RESTART: E:\tycs\al prac\pracl\pracl.py ====================
Following is Breadth First Traversal (starting from vertex 1)
1 2 3 4 5 6
performed by krunal 713
>>>
```

Date:31/08/2020

Practical no 2

AIM: Implement Iterative deep depth first search for Romanian map problem or any other map

CODE:-

```
from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph = defaultdict(list)
    def addEdge(self, u, v):
        self.graph[u].append(v)
    def DFSUtil(self, v, visited):
        visited[v] = True
        print(v, end = ' ')
        for i in self.graph[v]:
            if visited[i] == False:
                self.DFSUtil(i, visited)
    def DFS(self, v):
        visited = [False] * (max(self.graph)+1)
        self.DFSUtil(v, visited)
g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)
g.addEdge(3, 4)
g.addEdge(4, 4)
g.addEdge(4, 5)
g.addEdge(5, 4)
g.addEdge(5, 5)
g.addEdge(4, 6)
g.addEdge(5, 6)
g.addEdge(6, 6)

print("Following is DFS from (starting from vertex 0)")
print("Performed By krunal 713")
g.DFS(0)
```

OUTPUT:-

```
================ RESTART: C:/Users/BlackBot/Desktop/AI_prac2.py ================
Following is DFS from (starting from vertex 0)
Performed By krunal 713
0 1 2 3 4 5 6
>>>
```

Date:07/09/2020

Practical no 3

AIM: Implement A* search algorithm for Romanian map problem or any other map.

CODE:

```
from simpleai.search import SearchProblem, astar

GOAL = 'KRUNAL DHAVLE'
class HelloProblem(SearchProblem):
    def actions(self, state):
        if len(state) < len(GOAL):
            return list(' ABCDEFGHIJKLMNOPQRSTUVWXYZ')
        else:
            return []

    def result(self, state, action):
        return state + action
    def is_goal(self, state):
        return state == GOAL
    def heuristic(self, state):

        wrong = sum([1 if state[i] != GOAL[i]
                     else 0
                     for i in range(len(state))])
        missing = len(GOAL) - len(state)
        return wrong + missing

problem = HelloProblem(initial_state='')
result = astar(problem)
print(result.state)
print(result.path())
```

OUTPUT:

```
================== RESTART: C:/Users/BlackBot/Desktop/aiprac3.py ================
KRUNAL DHAVLE
[(None, ''), ('K', 'K'), ('R', 'KR'), ('U', 'KRU'), ('N', 'KRUN'), ('A', 'KRUNA'), ('L', 'KR
UNAL'), (' ', 'KRUNAL '), ('D', 'KRUNAL D'), ('H', 'KRUNAL DH'), ('A', 'KRUNAL DHA'), ('V',
'KRUNAL DHAV'), ('L', 'KRUNAL DHAVL'), ('E', 'KRUNAL DHAVLE')]
>>>
```

Date:05/10/2020

Practical no 4

AIM: Implement recursive best-first search algorithm for Romanian map problem.

CODE:

```
import queue as q
dict_hn={
 'A':336,
 'B':0,
 'C':160,
 'D':242,
 'E':161,
 'F':176
 }
dict_gn={
 'A':{'B':75,'C':118},
 'B':{'A':85,'D':211,'E':90},
 'C':{'A':120,'F':146},
 'D':{'B':75},
 'E':{'B':86},
 'F':{'C':99}
 }
def get_fn(citystr):
 cities=citystr.split(',')
 hn=0
gn=0
 ctr=0
 while ctr!=len(cities)-1:
 gn=gn+dict_gn[cities[ctr]][cities[ctr+1]]
 ctr+=1
 hn=dict_hn[cities[len(cities)-1]]
 return hn+gn
def expand(mycities,cityq,goal):
 tot,citystr=mycities
 cities=citystr.split(',')
 city2expand=cities[len(cities)-1]
 if(city2expand==goal):
 ans="The RBST Path is "+citystr+"with the value as "+str(tot);
 while not cityq.empty():
 cityq.get()
 return ans
 print("Expanded City -------",city2expand)
 tempq=q.PriorityQueue()
 for city in dict_gn[city2expand]:
 tempq.put((get_fn(citystr+','+city),citystr+','+city))
 print('First Best and Second Best inserted into tempq')
 ctr=1
 if(cityq.empty()):
```

```
 while not tempq.empty():

 if ctr==1 or ctr==2:
tempgn,tempcitystr=tempq.get()
 print('Inserting into cityqueue :',tempgn,tempcitystr)
 cityq.put((tempgn,tempcitystr))
 ctr=ctr+1
 else:
 #pass
 tempq.get()
 else:
 fn=0
 citystr=""
 fn=getSecondBest(cityq,fn,citystr)
 while not tempq.empty():
 if ctr==1 or ctr==2:
 tempgn,tempcitystr=tempq.get()
 if tempgn>ctr:
 if ctr==1:
 print('Inserting into cityqueue :',tempgn,tempcitystr)
 cityq.put((tempgn,tempcitystr))
 ctr=3
 continue
 else:
 #break
 print("Inserting into CityQueue:",tempgn,citystr)
 cityq.put((tempgn,tempcitystr))
 ctr+=1
 else:
tempq.get()
 while not tempq.empty():
 tempq.get()
def getSecondBest(cityq,fn,citystring):
 fn,citystring=cityq.get()
 cityq.put((fn,citystring))
 return fn
def main():
 start="A"
 goal="F"
 cityq=q.PriorityQueue()
 cityq.put((get_fn(start),start))
 while not cityq.empty():
 mycities=cityq.get()
 ans=expand(mycities,cityq,goal)
 print(ans)
 print('performed by krunal 713')
main()
```

OUTPUT:

```
Expanded City ------- A
First Best and Second Best inserted into tempq
Inserting into cityqueue : 75 A,B
Inserting into cityqueue : 278 A,C
Expanded City ------- B
First Best and Second Best inserted into tempq
Inserting into cityqueue : 326 A,B,E
Expanded City ------- C
First Best and Second Best inserted into tempq
Inserting into cityqueue : 440 A,C,F
Expanded City ------- E
First Best and Second Best inserted into tempq
Inserting into cityqueue : 251 A,B,E,B
Expanded City ------- B
First Best and Second Best inserted into tempq
Inserting into cityqueue : 502 A,B,E,B,E
The RBST Path is A,C,Fwith the value as 440
performed by krunal 713
>>>
```

Date:09/09/2020

Practical no 5

AIM: Implement decision tree learning algorithm for the restaurant waiting problem.

STEPS:

Step1: Download the graph viz file from below link and extract it.

https://graphviz.gitlab.io/_pages/Download/windows/graphviz-2.38.zip

Step2: Install the sklearn , ipython and pydotplus packages.First copy the path of script in python folder and then change the path of cmd.

Step3: Now install the packages by writing pip install and the packages name.

Step4: Next you have to change the environment variable. Copy the path of graphiz..Then go to environment and add new path.

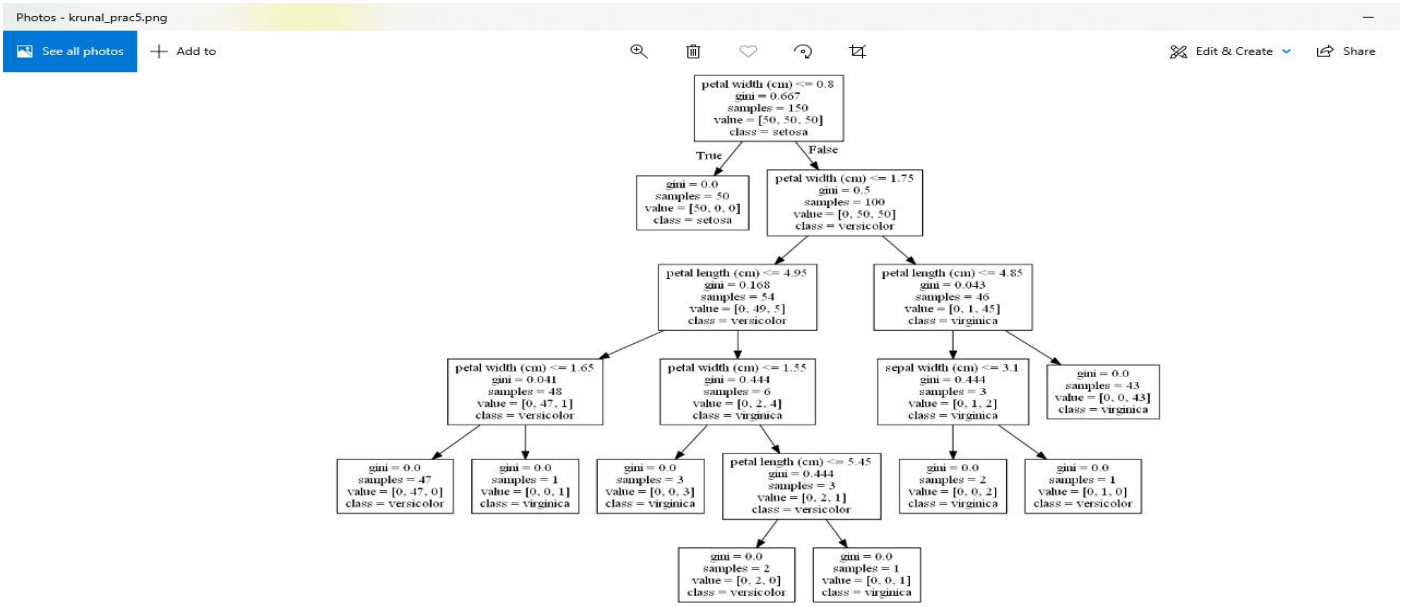
Step5: After all this is done write the code and run it . output will be in pdf and png format.

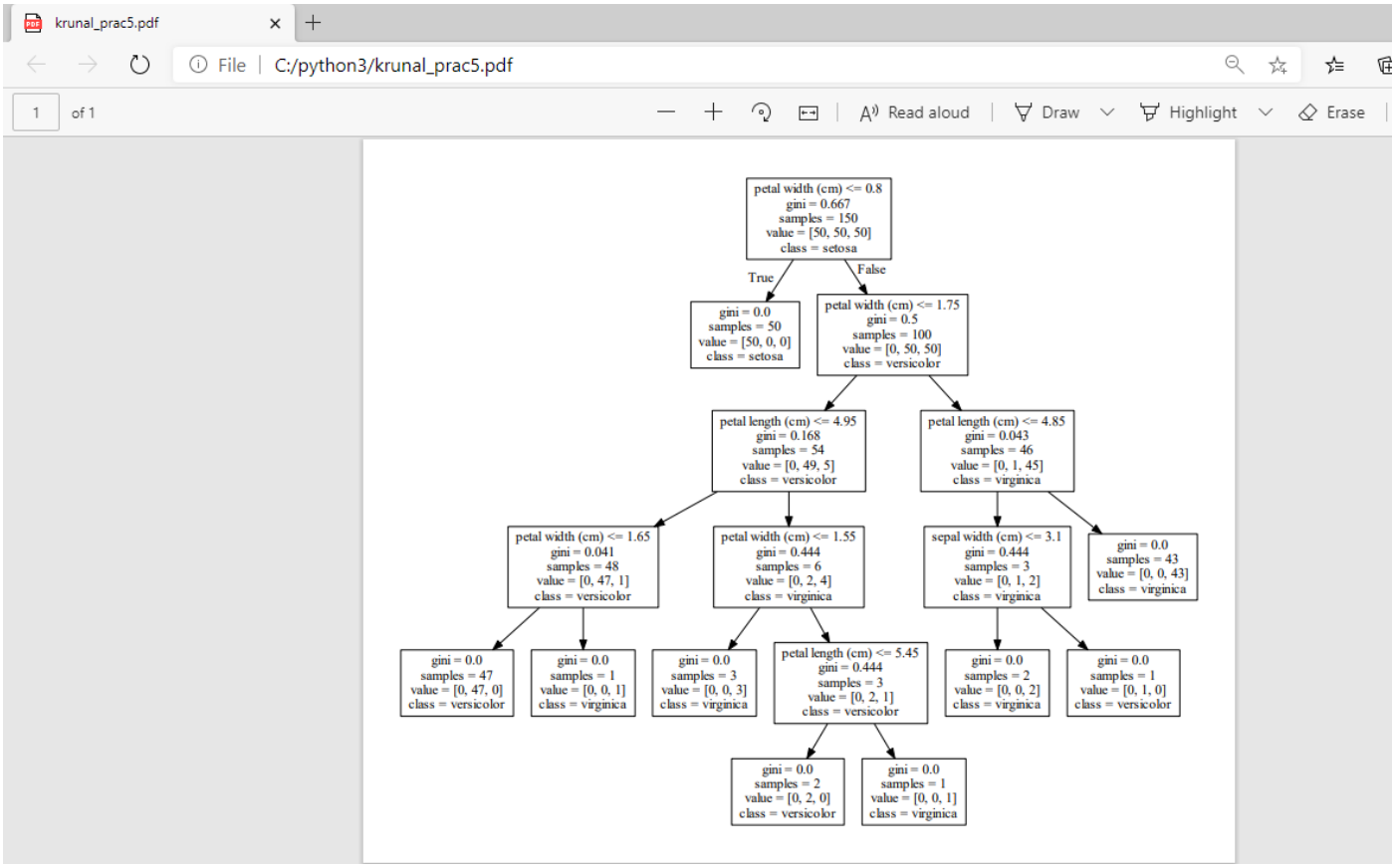
CODE:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from IPython.display import Image
from sklearn import tree
import os #only for windows
import pydotplus
os.environ['PATH'] += os.pathsep+ "C:/graphviz-2.38/release/bin/"
iris=datasets.load_iris()
x=iris.data
y=iris.target
clf=DecisionTreeClassifier(random_state=0)
model=clf.fit(x,y)
dot_data=tree.export_graphviz(clf,out_file=None,feature_names=iris.feature_nam
es,class_names=iris.target_names)
graph =pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
graph.write_pdf("krunal_prac5.pdf")
graph.write_png("krunal_prac5.png")
```

OUTPUT:

iris.pdf	09/09/2020 02:44 PM	Microsoft Edge P...	27 KB
iris.png	09/09/2020 02:44 PM	PNG File	78 KB
krunal_prac5.pdf	09/09/2020 02:51 PM	Microsoft Edge P...	27 KB
krunal_prac5.png	09/09/2020 02:51 PM	PNG File	78 KB





Date:11/10/2020

Practical no 6

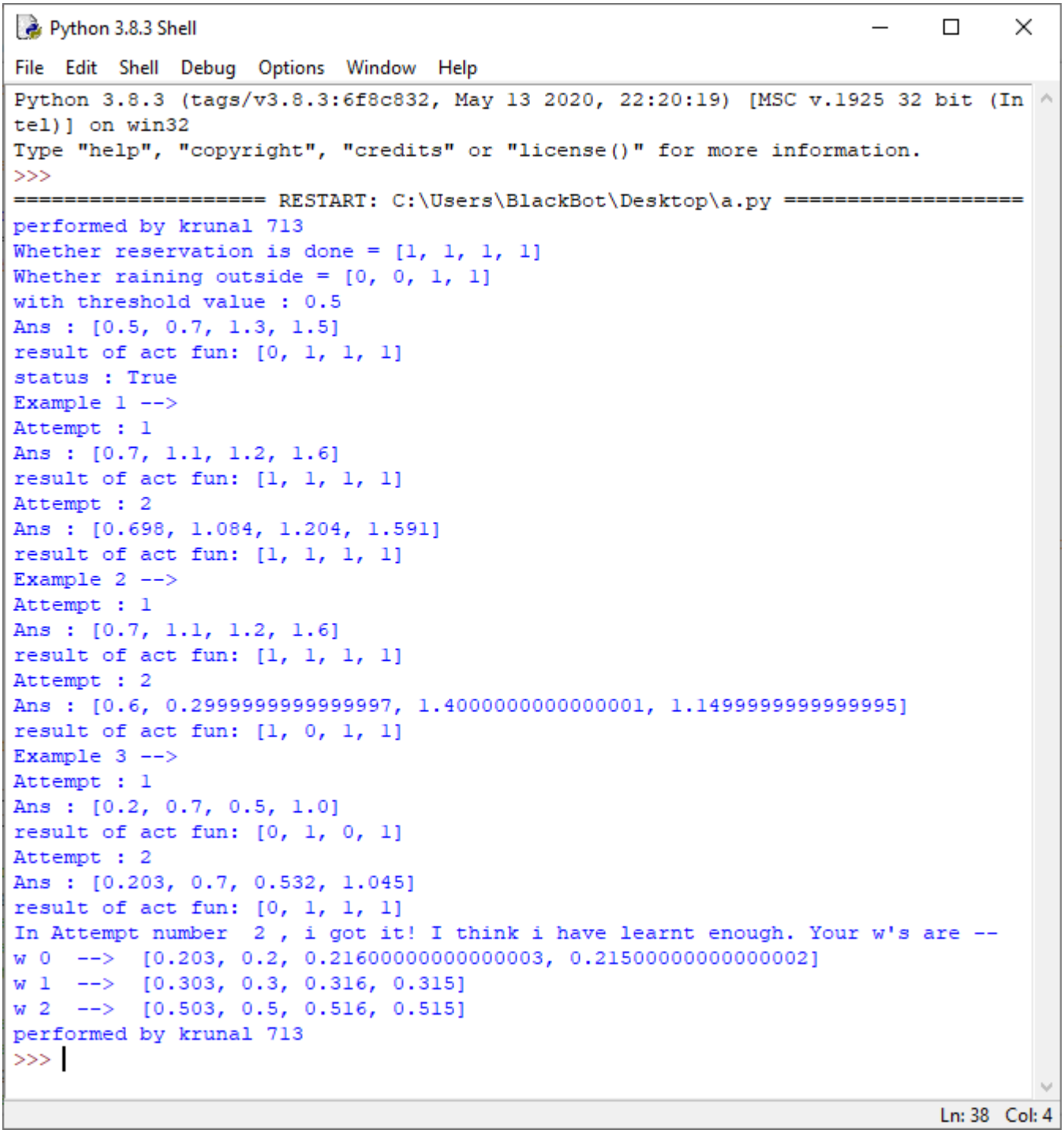
AIM: Implement feed forward back propagation neural network learning algorithm for the restaurant waiting problem.

CODE:

```
Implement feed forward back propagation neural network learning algorithm for the restaurant waiting problem

class Perceptron : # With 2 inputs and 1 output
    def __init__(self, a,b, c, tval):
        self.x = a # input vector
        self.result = b # activation result
        self.cresult = c # summation result
        self.threshold = tval # threshold value used by activation function
        self.w = []
    def h(self, tw): # calculating summation(hypothesis function)
        hresult= []
        for i in range(0 , len(self.result)):
            hresult.append(0)
            #print("index - ", i, ";", hresult)
            for j in range(0,len(tw)):
                #print("i=",i, ",j=",j)
                hresult[i] = hresult[i] + ( tw[j][i]*self.x[j][i] )
        return hresult
    def checkthreshold(self, hresult):  # applying activation function on summation result using threshold value
        #flag = True
        actfun =[]
        for i in range(0 , len(self.result)) :
            if (hresult[i] <= self.threshold ):
                actfun.append(0)
            else :
                actfun.append( 1)
        print("Ans :", hresult)
        print("result of act fun:", actfun)

        for i in range(0 , len(self.x)) :
            if (actfun[i] != self.result[i]) :
                return False
        return True

    def training(self, tw, alpha): #passing w vector and alpha value
        i=1
        while i<=2 : # Max 100 attempts
                print("Attempt :", i)
                hresult = self.h(tw)


                    if(self.checkthreshold(hresult)) :  #if training result matches the test result
```

```
                self.w = tw

 print("In Attempt number ", i,  ", i got it! I think i have learnt enough. Your w's are --" )
                for x in range(0,len(self.w)):
                    print("w", x, " --> ", self.w[x])
                break

            i = i +1
            # Changing values of w to reduce error/loss using batch gradient descent learning rule given on
page 721 eqn 18.6
            for j in range(0,len(self.result)) :
                for k in range(0, len(tw)):
                    sum = 0
                    for n in range(0, len(tw)):
                        sum = sum + (self.cresult[j] - hresult[j]) *self.x[n][j]
                    tw[k][j] = tw[k][j] + alpha*sum

        if(i>=100):
            print("I am exhausted, tried 100 iterations! plz change something else...")
a = [ [1,1,1,1], [0,0,1,1] , [0,1,0,1] ] # x vector, x0 is dummy
b = [0,1,1,1] # result of activation function
c = [0.5, 0.7, 1.3, 1.5] # sample h values

p = Perceptron(a,b,c, 0.5) # threshold = 0.5
print("Whether reservation is done =", p.x[0])
print("Whether raining outside =", p.x[1])
print("with threshold value :", p.threshold)
r =  p.h([ [0.5,0.5,0.5,0.5], [0.8, 0.8, 0.8, 0.8], [0.2, 0.2, 0.2, 0.2]])
print("status :", p.checkthreshold(r))
print("Example 1 -->") #with alpha as 0.01, you will not get result
p.training( [ [0.7,0.7,0.7,0.7], [0.5, 0.5, 0.5, 0.5], [0.4, 0.4, 0.4, 0.4]], 0.01)
print("Example 2 -->")  #with alpha as 0.5, you will not get result
p.training( [ [0.7,0.7,0.7,0.7], [0.5, 0.5, 0.5, 0.5], [0.4, 0.4, 0.4, 0.4]], 0.5)
print("Example 3 -->")
p.training( [ [0.2,0.2,0.2,0.2], [0.3, 0.3, 0.3, 0.3], [0.5, 0.5, 0.5, 0.5]], 0.01)
```

OUTPUT :


Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\BlackBot\Desktop\a.py ==========
performed by krunal 713
Whether reservation is done = [1, 1, 1, 1]
Whether raining outside = [0, 0, 1, 1]
with threshold value : 0.5
Ans : [0.5, 0.7, 1.3, 1.5]
result of act fun: [0, 1, 1, 1]
status : True
Example 1 -->
Attempt : 1
Ans : [0.7, 1.1, 1.2, 1.6]
result of act fun: [1, 1, 1, 1]
Attempt : 2
Ans : [0.698, 1.084, 1.204, 1.591]
result of act fun: [1, 1, 1, 1]
Example 2 -->
Attempt : 1
Ans : [0.7, 1.1, 1.2, 1.6]
result of act fun: [1, 1, 1, 1]
Attempt : 2
Ans : [0.6, 0.2999999999999997, 1.4000000000000001, 1.1499999999999995]
result of act fun: [1, 0, 1, 1]
Example 3 -->
Attempt : 1
Ans : [0.2, 0.7, 0.5, 1.0]
result of act fun: [0, 1, 0, 1]
Attempt : 2
Ans : [0.203, 0.7, 0.532, 1.045]
result of act fun: [0, 1, 1, 1]
In Attempt number 2 , i got it! I think i have learnt enough. Your w's are --
w 0 --> [0.203, 0.2, 0.21600000000000003, 0.21500000000000002]
w 1 --> [0.303, 0.3, 0.316, 0.315]
w 2 --> [0.503, 0.5, 0.516, 0.515]
performed by krunal 713
>>>
Ln: 38 Col: 4

Date:02/11/2020

Practical no 7

AIM: Implement Adaboost ensemble learning algorithm for the restaurant waiting problem Or any other problem.

CODE:

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import AdaBoostClassifier
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
num_trees = 30
kfold = model_selection.KFold(n_splits=10)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

OUTPUT:

```
>>>
================== RESTART: C:/Users/BlackBot/Desktop/prac7.py ==================
0.760457963089542
performed by krunal 713
>>>
```

Date:23/11/2020

Practical no 8

Aim: Implement Naive Bayes learning algorithm for the restaurant waiting problem.

CODE:

```
class NaiveBayes:
    def __init__(self, f, r):
        self.features = f
        self.response = r
    def predict(self,custcase):
        anskeys = list(self.response.keys())
        ansvalues = dict.fromkeys(anskeys,0)

        for key in anskeys :
            ansvalues[key] = self.response[key]
            for ikey, ival in custcase.items() :
                ansvalues[key] = ansvalues[key] * self.features[ikey][ival][key]
            print(ansvalues)
            maxkey=""
            maxans=-1
            for ikey, ival in ansvalues.items():
                if ival > maxans :
                    maxans= ival
                    maxkey = ikey
                return maxkeyresponse = {"Wait":0.4, "Leave":0.6}
features = {
                                                "Reservation":
                                                {
                                                "Yes" : {"Wait":0.5, "Leave":0.666667},
                                                "No" : {"Wait":0.5, "Leave":0.333333}
                                                } ,
                                                "Time>30":
                                                {
                                                "Yes" : {"Wait":0.25, "Leave":0.83333},
                                                "No" : {"Wait":0.75, "Leave":0.16667}
                                                }
}
nb = NaiveBayes(features, response)
print("Probability :", nb.features["Reservation"]["Yes"]["Wait"])
print("Probability :", nb.features["Time>30"]["No"]["Leave"])
resstatus = input("Manager asks Customer, Have you reserved the table?(Yes/No):")
timestatus = input("Customer asks Manager, Will it take more than 30 mins?(Yes/No):")
custcase = {"Reservation":resstatus, "Time>30":timestatus}
print("Manager predicts that Customer will: " , nb.predict(custcase) )
print("Performed By 713 krunal dhavle")
```

OUTPUT:

```
Probability : 0.5
Probability : 0.16667
Manager asks Customer, Have you reserved the table?(Yes/No):Yes
Customer asks Manager, Will it take more than 30 mins?(Yes/No):Yes
{'Wait': 0.05, 'Leave': 0.33333216666599996}
Manager predicts that Customer will:  Leave
Performed By 713 krunal dhavle
>>>
```

```
Probability : 0.5
Probability : 0.16667
Manager asks Customer, Have you reserved the table?(Yes/No):No
Customer asks Manager, Will it take more than 30 mins?(Yes/No):No
{'Wait': 0.15000000000000002, 'Leave': 0.033333966666}
Manager predicts that Customer will:  Wait
Performed By 713 krunal dhavle
>>>
```

```
Probability : 0.5
Probability : 0.16667
Manager asks Customer, Have you reserved the table?(Yes/No):Yes
Customer asks Manager, Will it take more than 30 mins?(Yes/No):No
{'Wait': 0.15000000000000002, 'Leave': 0.066668033334}
Manager predicts that Customer will:  Wait
Performed By 713 krunal dhavle
>>>
```

```
Probability : 0.5
Probability : 0.16667
Manager asks Customer, Have you reserved the table?(Yes/No):No
Customer asks Manager, Will it take more than 30 mins?(Yes/No):Yes
{'Wait': 0.05, 'Leave': 0.16666583333399998}
Manager predicts that Customer will:  Leave
Performed By 713 krunal dhavle
>>>
```