Python: without numpy or sklearn

Q1: Given two matrices please print the product of those two matrices

```
Ex 1: A   = [[1 3 4],
             [2 5 7],
             [5 9 6]]
      B   = [[1 0 0],
             [0 1 0],
             [0 0 1]]
      A*B = [[1 3 4]
             [2 5 7]
             [5 9 6]]


Ex 2: A   = [[1 2]
             [3 4]]
      B   = [[1 2 3 4 5]
             [5 6 7 8 9]]
      A*B = [[11 14 17 20 23]
             [23 30 36 42 51]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
             [9 6]]
      A*B =Not possible
```

In [308…

```python
def matrixmultiplication(a,b):
    c=[]
    if(len(a[0]) == len(b)):
        #tlist=[]
        for x in range(len(a)):
            tlist=[]
            for y in range(len(b[0])):
                sum_m=0
                for z in range(len(a)):
                    sum_m +=(a[x][z]*b[z][y])
                tlist.append(sum_m)
            c.append(tlist)
        for alpha in c:
            print(alpha)
    else:
        print("Not Possible")
```

In [309…

```python
matrixmultiplication([[1, 2],[3, 4]],[[1,2,3,4,5],[5,6,7,8,9]])
```

```
[11, 14, 17, 20, 23]
[23, 30, 37, 44, 51]
```

In [310…

```python
matrixmultiplication([[1, 2],[3, 4]],[[1,4],[5,6],[7,8],[9,6]])
```

```
Not Possible
```

In [311…

```python
matrixmultiplication([[1,3,4],[2,5,7],[5,9,6]],[[1,0,0],[0,1,0],[0,0,1]])
```

```
[1, 3, 4]
[2, 5, 7]
[5, 9, 6]
```

Q2: Select a number randomly with probability proportional to its magnitude from the given array of n elements

consider an experiment, selecting an element from the list A randomly with probability proportional to its magnitude. assume we are doing the same experiment for 100 times with replacement, in each experiment you will print a number that is selected randomly from A.

```
Ex 1: A = [0 5 27 6 13 28 100 45 10 79]
let f(x) denote the number of times x getting selected in 100 experiments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f(0)
```

In [312…

```python
from random import uniform
def pick_a_number_from_list(A):
    A.sort()
    sum_s=0
    sum_c = 0
    normalize_l=[]
    list_c=[]
    for x in range(len(A)):
        sum_s += A[x]

    for x in range(len(A)):
        normalize = A[x]/sum_s
        normalize_l.append(normalize)
        sum_c +=normalize_l[x]
        list_c.append(sum_c)

    r= random.uniform(0.0,1.0)

    for x in range(len(list_c)):
        if(r>=list_c[x] and r<list_c[x+1]):
            return(A[x+1])

def sampling_based_on_magnitued():
    dict1=dict()
    dict1[0]=0
    for i in range(0,100):
        number = pick_a_number_from_list(A)
        if number not in dict1:
            dict1[number] = 1

        else:
            dict1[number] += 1
    print("OUTPUT:",end=("\n"))
    for i in sorted (dict1.keys()) :
        print("<f(",i,")", end = " ")
```

In [313…

```python
A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
sampling_based_on_magnitued()
```

```
OUTPUT:
<f( 0 ) <f( 5 ) <f( 6 ) <f( 10 ) <f( 13 ) <f( 27 ) <f( 28 ) <f( 45 ) <f( 79 ) <f( 100 )
```

Q3: Replace the digits in the string with

consider a string that will have digits in that, we need to remove all the not digits and replace the digits with #

```
Ex 1: A = 234                Output: ###
Ex 2: A = a2b3c4             Output: ###
Ex 3: A = abc                Output:   (empty string)
Ex 5: A = #2a$#b%c%561#      Output: ####
```

In [314…

```python
def replace_digit(String):
    count = 0
    tlist = list(String)
    for x in range(len(tlist)):
        if(tlist[x].isnumeric()):
            count += 1
    print("OUTPUT:",end='\n')
    print("#"*count)
```

In [315…

```python
replace_digit('#2a$#b%c%561#')
```

```
OUTPUT:
####
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]
from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on

your task is to print the name of students **a. Who got top 5 ranks, in the descending order of marks**
b. Who got least 5 ranks, in the increasing order of marks
d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
Ex 1:
Students=
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10'

Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

In [316…

```python
def display_dash_board(students,Marks):
    stud_n =[]
    stud_m =[]
    dict1 = dict(zip(students,Marks))
    stud_am = sorted(Marks)
    stud_m = sorted(Marks,reverse = True)
    print("a.",end='\n')
    for x in range(0,5):
        for Students,Marks in dict1.items():
            if Marks == stud_m[x]:
                print(Students,":",Marks)
    print("  ")
    print("b.",end='\n')
    for x in range(0,5):
        for Students,Marks in dict1.items():
            if Marks == stud_am[x]:
                print(Students,":",Marks)
    print("  ")
    print("c.",end='\n')
    s = int(.25*len(stud_am))
    e = int(.75*len(stud_am))
    for x in range(s,e):
        for Students,Marks in dict1.items():
            if Marks == stud_am[x]:
                print(Students,":",Marks)
```

In [317…

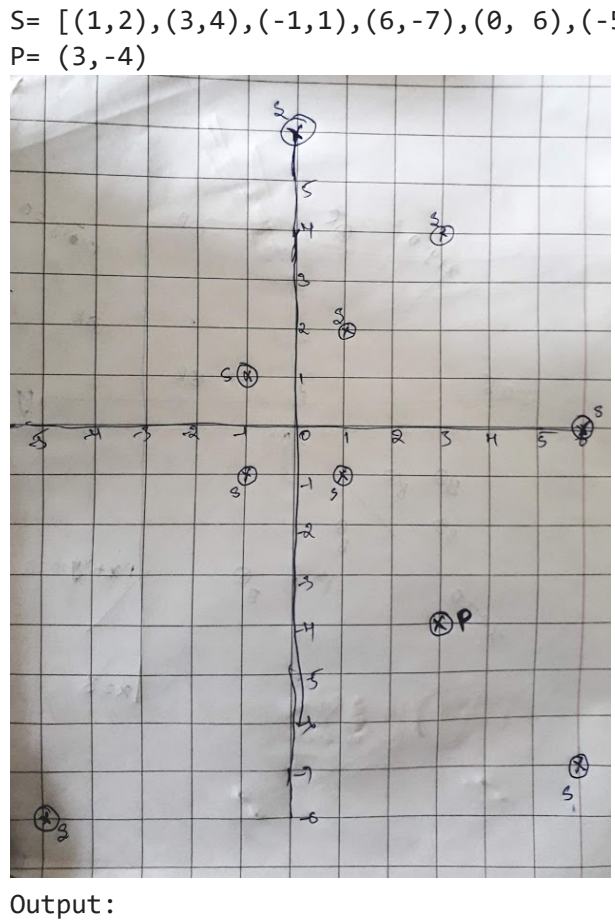
```python
Students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','s
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
display_dash_board(Students,Marks)
```

```
a.
student8 : 98
student10 : 80
student2 : 78
student5 : 48
student7 : 47

b.
student3 : 12
student4 : 14
student9 : 35
student6 : 43
student1 : 45

c.
student9 : 35
student6 : 43
student1 : 45
student7 : 47
student5 : 48
```

Q5: Find the closest points

consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)
your task is to find 5 closest points(based on cosine distance) in S from P
cosine distance between two points (x,y) and (p,q) is defind as $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

```
Ex:

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1)(6,0),(1,-1)]
P= (3,-4)
```



```
Output:
(6,-7)
(1,-1)
(6,0)
(-5,-8)
(-1,-1)
```

In [318…

```python
import math
def closest_points_to_p(S, P):
    dis_list = []
    sort_dis = []
    for a in range(len(S)):
        point = S[a]
        x =point[0]
        y = point[1]
        p = P[0]
        q = P[1]
        distance = math.acos(((x*p)+(y*q))/(math.sqrt((x**2)+(y**2)) * math.sqrt((p**2)+(q**2))))
        dis_list.append(distance)
    dict_1 = dict(zip(S,dis_list))
    sort_dis = sorted(dis_list)
    print("Output:",end='\n')
    for b in range(0,5):
        for S,dis_list in dict_1.items():
            if dis_list == sort_dis[b]:
                print(S)
```

In [319…

```python
S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
closest_points_to_p(S, P)
```

```
Output:
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

Q6: Find Which line separates oranges and apples

consider you have given two set of data points in the form of list of tuples like

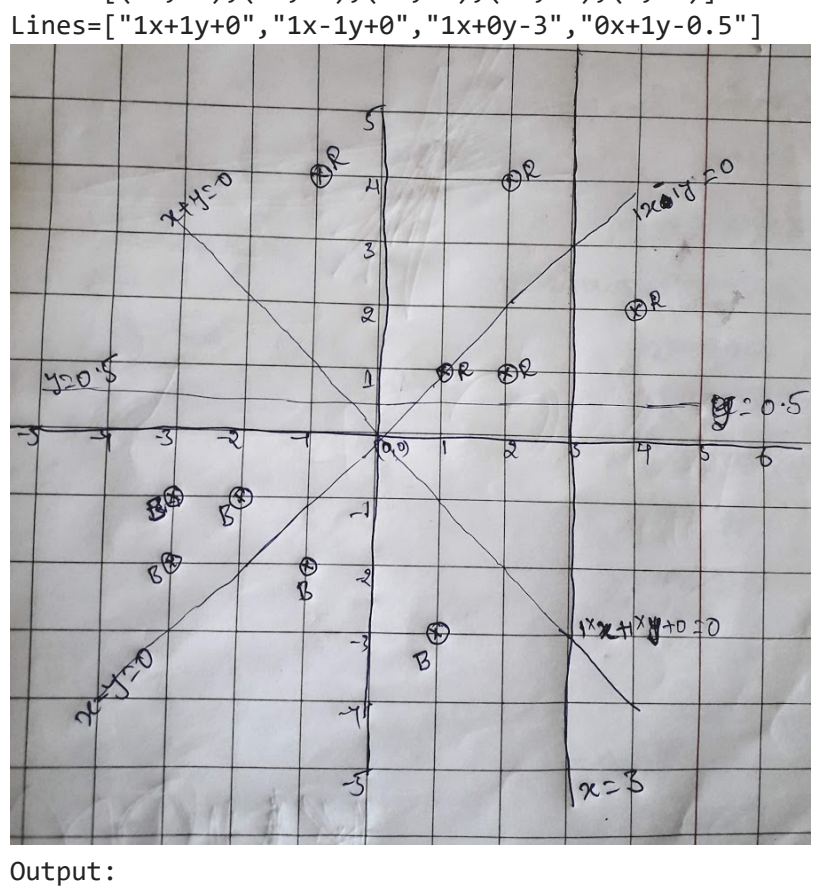
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

and set of line equations(in the string formate, i.e list of strings)

```
Lines = [a1x+b1y+c1,a2x+b2y+c2,a3x+b3y+c3,a4x+b4y+c4,..,K lines]
Note: you need to string parsing here and get the coefficients of x,y and intercept
```

your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

```
Ex:
Red= [(1,1),(2,1),(4,2),(2,4),(-1,4)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]
```



```
Output:
YES
NO
NO
YES
```

In [320…

```python
import math
import re
def i_am_the_one(red,blue,line):
    print("Output:",end='\n')
    for equ in range(len(line)):
        list_red=[]
        list_blue=[]
        count = 0
        coff = re.findall(r'[\d\.\-\+]+', line[equ])
        for Red in range(len(red)):
            r_x = red[Red][0]
            r_y = red[Red][1]
            r_value = (r_x*(float(coff[0])))+(r_y*(float(coff[1])))+float(coff[2])
            list_red.append(r_value)
        for Blue in range(len(blue)):
            b_x = blue[Blue][0]
            b_y = blue[Blue][1]
            b_value = (b_x*(float(coff[0])))+(b_y*(float(coff[1])))+float(coff[2])
            list_blue.append(b_value)
        for value in range(len(list_blue)):
            if(list_red[value]>=0.0 and list_blue[value]<0.0 or list_red[value]<0.0 and list_blue[value]>=0.0):
                count += 1
        if(count == len(list_blue)):
            print('YES')
        else:
            print('NO')
```

In [321…

```python
red= [(1,1),(2,1),(4,2),(2,4),(-1,4)]
blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
line=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]
i_am_the_one(red,blue,line)
```

```
Output:
YES
NO
NO
YES
```

Q7: Filling the missing values in the specified formate

You will be given a string with digits and '_'(missing value) symbols you have to replace the '_' symbols as explained

```
Ex 1: _, _, _, 24 ==> 24/4, 24/4, 24/4, 24/4 i.e we. have distributed the 24 equally to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 20, 20, 20, 20, 20 i.e.
the sum of (60+40) is distributed qually to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. the 80 is distributed
qually to all 5 missing values that are right to it

Ex 4: _, _, 30, _, _, _, 50, _, _
==> we will fill the missing values from left to right
    a. first we will distribute the 30 to left two missing values (10, 10, 10, _, _, _, 50, _, _)
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12, 12, _, _)
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 4, 4, 4)
```

for a given string with comma seprate values, which will have both missing values numbers like ex: "_, _, x, _, _, _" you need fill the missing values Q: your program reads a string like ex: "_, _, x, _, _, _" and returns the filled sequence Ex:

```
Input1: "_,_,_,24"
Output1: 6,6,6,6

Input2: "40,_,_,_,60"
Output2: 20,20,20,20,20

Input3: "80,_,_,_,_"
Output3: 16,16,16,16,16

Input4: "_,_,30,_,_,_,50,_,_"
Output4: 10,10,12,12,12,12,4,4,4
```

In [322…

```python
def curve_smoothing(S):
    string = S.split(",")
    for x in range(len(string)):  # coverting into the integer from string
        if(string[x] == "_" ):
            string[x] = 0
        if(string[x]!="_"):
            string[x]=int(string[x])
    String=string

    q,sum_m,j=0,0,0
    while(q<len(String)):
        if(String[q] != 0 or q==(len(String)-1)):
            sum_m+=(String[q])
            z = (q-j)-1
            for a in range(q,z,-1):
                String[a]=int(sum_m/(j+1))
            j=0
            sum_m=(String[q])
        j+=1
        q+=1
    print("OUTPUT:",end='\n')
    print((String))
```

In [323…

```python
S=  "_,_,30,_,_,_,50,_,_"
smoothed_values= curve_smoothing(S)
```

```
OUTPUT:
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

Q8: Filling the missing values in the specified formate

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

1. the first column F will contain only 5 uniques values (F1, F2, F3, F4, F5)
2. the second column S will contain only 3 uniques values (S1, S2, S3)

```
your task is to find
a. Probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3)
b. Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3)
c. Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3)
d. Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3)
e. Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3)
```

Ex:

```
[[F1,S1],[F2,S2],[F3,S3],[F1,S2],[F2,S3],[F3,S2],[F2,S1],[F4,S1],[F4,S3],[F5,S1]]

a. P(F=F1|S==S1)=1/4, P(F=F1|S==S2)=1/3, P(F=F1|S==S3)=0/3
b. P(F=F2|S==S1)=1/4, P(F=F2|S==S2)=1/3, P(F=F2|S==S3)=1/3
c. P(F=F3|S==S1)=0/4, P(F=F3|S==S2)=1/3, P(F=F3|S==S3)=1/3
d. P(F=F4|S==S1)=1/4, P(F=F4|S==S2)=0/3, P(F=F4|S==S3)=1/3
e. P(F=F5|S==S1)=1/4, P(F=F5|S==S2)=0/3, P(F=F5|S==S3)=0/3
```

In [324…

```python
def compute_conditional_probabilites(A):
    f,s,e,f_s,s_s,s_e,f_e=[],[],[],[],[],[],[]  # required list using to make the logic
    nr,dr = 0,0
    for x in range(len(A)):
        ele,ele_s,ele_f = 0,0,0       # counting the element(e,f,s)
        for y in range(len(A)):
            if(A[x]==A[y]):
                ele += 1
            if(A[x][0] == A[y][0]):
                ele_f += 1
                f.append(A[x][0])
            if(A[x][1] == A[y][1]):
                ele_s += 1
                s.append(A[x][1])
        e.append(str(ele))
        f_s.append(str(ele_f))
        s_s.append(str(ele_s))
    f = list(set(f))
    s = list(set(s))
#     print("f:",f)
#     print("s:",s)
#     print("e:",e)
#     print("f_s:",f_s)
#     print("s_s:",s_s)

    for x in range(len(s)):                  # for make dict as key(s) and value(count(s))
        scount = 0
        for y in range(len(A)):
            if(s[x] == A[y][1]):
                scount +=1
        s_e.append(scount)
    d_s = dict(zip(s,s_e))
#     print(d_s)

#     for x in range(len(f)):
#         fcount = 0
#         for y in range(len(A)):
#             if(f[x] == A[y][0]):
#                 fcount +=1
#         f_e.append(fcount)
#     d_f = dict(zip(f,f_e))
#     print(d_f)


    T_list=[]                      # list of all combination of f&s
    for x in range(len(f)):
        for y in range(len(s)):
            if([f[x],s[y]] in A):
                pass
            T_list.append([f[x],s[y]])
    print("OUTPUT:",end='\n')
    for x in range(len(T_list)): # checking which element is present in the given A and according to the prese
        if(T_list[x] in A):
            nr = e[A.index(T_list[x])]
            dr = s_s[A.index(T_list[x])]
            print("P(F=",T_list[x][0],"|","S==",T_list[x][1],")=",nr,"/",dr,end="\n")
        else:
            nr = 0
            dr = d_s.get(T_list[x][1])
            print("P(F=",T_list[x][0],"|","S==",T_list[x][1],")=",nr,"/",dr,end="\n")
```

In [325…

```python
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'
compute_conditional_probabilites(A)
```

```
OUTPUT:
P(F= F2 | S== S2 )= 1 / 3
P(F= F2 | S== S3 )= 1 / 3
P(F= F2 | S== S1 )= 1 / 4
P(F= F4 | S== S2 )= 0 / 3
P(F= F4 | S== S3 )= 1 / 3
P(F= F4 | S== S1 )= 1 / 4
P(F= F5 | S== S2 )= 0 / 3
P(F= F5 | S== S3 )= 0 / 3
P(F= F5 | S== S1 )= 1 / 4
P(F= F3 | S== S2 )= 1 / 3
P(F= F3 | S== S3 )= 1 / 3
P(F= F3 | S== S1 )= 0 / 4
P(F= F1 | S== S2 )= 1 / 3
P(F= F1 | S== S3 )= 0 / 3
P(F= F1 | S== S1 )= 1 / 4
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

```
a. Number of common words between S1, S2
b. Words in S1 but not in S2
c. Words in S2 but not in S1
```

Ex:

```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
Output:
a. 7
b. ['first','F','5']
c. ['second','S','3']
```

In [326…

```python
def string_features(s1,s2):
    s3 = []
    s1 = s1.split()
    s2 = s2.split()
    s3 = s1+s2
    print("OUTPUT:",end='\n')
    print("a.",len(set(s1).intersection(set(s2))))
    print("b.",list(set(s3)-set(s2)))
    print("c.",list(set(s3)-set(s1)))
```

In [327…

```python
string_features("the first column F will contain only 5 uniques values","the second column S will contain only
```

```
OUTPUT:
a. 7
b. ['F', '5', 'first']
c. ['S', '3', 'second']
```

Q10: Given two sentances S1, S2

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

a. the first column Y will contain interger values
b. the second column Y_{score} will be having float values
Your task is to find the value of $f(Y,Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score})+(1-Y)log10(1-Y_{score}))$ here n is the number of rows in the matrix

```
Ex:
[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
output:
0.4243099
```

$$\frac{-1}{8}\cdot((1\cdot log_{10}(0.4)+0\cdot log_{10}(0.6))+(0\cdot log_{10}(0.5)+1\cdot log_{10}(0.5))+\ldots+(1\cdot log_{10}(0.8)+0\cdot log_{10}(0.2)))$$

In [328…

```python
import math
def compute_log_loss(A):
    pro = 0
    for x in range(len(A)):
        pro += (A[x][0]*math.log10(A[x][1]))+((1-A[x][0])*(math.log10(1-A[x][1])))
    print("OUTPUT:",end='\n')
    print(-1*(pro/len(A)))

A = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
compute_log_loss(A)
```

```
OUTPUT:
0.42430993457031635
```