Assignment_3

November 9, 2021

Python: without numpy or sklearn

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

[1]:
```
### write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given␣
↪input examples
# you can take matrix input from user or you can directly define the matrix and␣
↪give input to the function
# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(r1,c1,A,r2,c2,B):
    """
    This function will multiply Two Matrices.
    """
    if c1 != r2:
        return "Not Possible"
    else:
        result = [[0 for i in range(c2)] for j in range(r1)]

        for i in range(len(A)):
            for j in range(c2):
                for k in range(r1):
                    result[i][j] += A[i][k]*B[k][j]

        return result

def createMat(r,c,lst):
    """
    This function will create a matrix using a List
    """
    mat = []
    for i in range(r):
        row = []
        for j in range(c):
            row.append(lst[c*i+j])
        mat.append(row)
    return mat

def createList(r,c):
    """
    This Function will takes input from user and creates a list.
    """
    n = []
    print("Enter the elements of the matrix:")
    for i in range(r*c):
        n.append(int(input()))

    return n


r1 = int(input("enter the no of rows of matrix A:"))
c1 = int(input("enter the no of columns of matrix A:"))
A = createMat(r1,c1,createList(r1,c1))
r2 = int(input("enter the no of rows of matrix B:"))
c2 = int(input("enter the no of columns of matrix B:"))
B = createMat(r2,c2,createList(r2,c2))
print("A = ",A)
print("\nB = ",B)
print("\n A*B = ",matrix_mul(r1,c1,A,r2,c2,B))
```

```
enter the no of rows of matrix A:2
enter the no of columns of matrix A:2
Enter the elements of the matrix:
1
2
3
4
enter the no of rows of matrix B:2
enter the no of columns of matrix B:5
Enter the elements of the matrix:
1
2
3
4
5
5
6
7
8
9
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, 37, 44, 51]]
```

Q2: Proportional Sampling - 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.

[2]:
```
from random import uniform

A = [0,5,27,6,13,28,100,45,10,79]
b = []

def pick_a_number_from_list(A):
    sum = 0
    for i in range(len(A)):
        sum += A[i]

    d_dash = []                           #Calculate normalized sum of each␣
 ↪variable in list
    for j in range(len(A)):
        d_dash.append(A[j]/sum)

    d_bar =[0 for i in range(len(A))]      #Calculate Cumulative sum
    d_bar[0] = d_dash[0]
    for k in range(len(d_dash)-1):
        d_bar[k+1] = d_bar[k] + d_dash[k+1]

    r = uniform(0.0,1.0)
    num = 0
    for m in range(len(d_bar)):
        if r <= d_bar[m]:
            num = A[m]
            break
    return num

def sampling_based_on_magnitude():
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        b.append(number)
    return number

sampling_based_on_magnitude()

c=(sorted(A, reverse=True))
print(c)
```

```
for x in c:
    print("count of {} is".format(x),b.count(x))
```

```
[100, 79, 45, 28, 27, 13, 10, 6, 5, 0]
count of 100 is 32
count of 79 is 23
count of 45 is 9
count of 28 is 13
count of 27 is 14
count of 13 is 3
count of 10 is 1
count of 6 is 2
count of 5 is 2
count of 0 is 0
```

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 #

[3]:
```
import re

def replace_digits(String):
    m = re.sub(r'\D','',String)     # new string only with digits
    return('#' * len(m))            # replacing that string with #

Str = ("234","a2b3c4","abc","#2a$#b%c%561#")
for i in Str:
    print("Original String:{} and Replaced String:{}\n".
 ↪format(i,replace_digits(i)))
```

```
Original String:234 and Replaced String:###

Original String:a2b3c4 and Replaced String:###

Original String:abc and Replaced String:

Original String:#2a$#b%c%561# and Replaced String:####
```

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, 22, 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

```
[4]: def display_dash_board(Students, Marks):
         Scores = dict(zip(Students,Marks))

         des = sorted(Scores.items(), key =
                lambda kv:(kv[1], kv[0]))   # Sorting in Ascending order

         top_5_students = [des[-i] for i in range(1,6)]

         least_5_students = [des[i] for i in range(5)]

         max_mark = max(Marks)                   # Student's marks between 25th and␣
      ↪75th percentile
         min_mark = min(Marks)
         diff =  max_mark - min_mark
         per_25 = diff*0.25
         per_75 = diff*0.75
         students_within_25_and_75= list(filter(lambda x:x[1]>per_25 and␣
      ↪x[1]<per_75, des))

         return top_5_students, least_5_students, students_within_25_and_75

     Students=['student1','student2','student3','student4','student5','student6','student7','studen
     Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]

     top_5_students, least_5_students, students_within_25_and_75 =␣
      ↪display_dash_board(Students, Marks)
     #s.sort()
     print("Top 5 Students:",top_5_students)
     print("\nLast 5 Students:",least_5_students)
     print("\nStudents between 25th and 75th percentile:",students_within_25_and_75)
```

```
Top 5 Students: [('student8', 98), ('student10', 80), ('student2', 78),
('student5', 48), ('student7', 47)]

Last 5 Students: [('student3', 12), ('student4', 14), ('student9', 22),
('student6', 43), ('student1', 45)]

Students between 25th and 75th percentile: [('student9', 22), ('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 $cos^{-1}(\frac{(x \cdot p+y \cdot q)}{\sqrt{(x^2+y^2)} \cdot \sqrt{(p^2+q^2)}})$

Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) =

0.065

[5]:
```
import math as m

def closest_points_to_p(S, P):
    dis = []
    P_sqrt = m.sqrt(float(P[0]*P[0] + P[1]*P[1]))
    for i in range(len(S)):
        X = float(S[i][0])
        Y = float(S[i][1])
        dis.append(m.acos((X*P[0] + Y*P[1])/(m.sqrt(X*X + Y*Y)*P_sqrt)))

    dict_dis = dict(zip(S,dis))
    sort_dict = sorted(dict_dis.items(), key=
                       lambda kv:(kv[1],kv[0]))
    close_pts = [sort_dict[i][0] for i in range(5)]

    return close_pts

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
points = closest_points_to_p(S, P)
print("Closest points in S from P are:",points)
```

```
Closest points in S from P are: [(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

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

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

[6]:
```
import math

def coef(s):
    """
    This function will return coefficients of given expression.
    """
    c_x = s.split('x')
    a = float(c_x[0])

    c_y = c_x[1].split('y')
    b = float(c_y[0])
    c = float(c_y[1])

    return (a,b,c)
```

```
def evaluate(x_y,p):
    """
    This function will evaluate coef and Pionts.
    """
    e = x_y[0]*p[0] + x_y[1]*p[1] + x_y[2]

    if e>=0:
        return 1
    else:
        return -1

def i_am_the_one(red,blue,line):

    for j in range(len(red)):
        if evaluate(coef(line),red[j]) <= 0:
            return "NO"
    for k in range(len(blue)):
        if evaluate(coef(line),blue[k]) > 0:
            return "NO"

    return "YES"

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"]

for i in Lines:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print(yes_or_no) # the returned value
```

```
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

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:

[7]:
```
def replace(x,a,b):
    """
    This function will replace the values.
    """
    if a == -1:                        # if first index has a digit
        v = float(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = v
    elif b == -1:                      # if last index has a digit
        v = float(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = v
    else:                              # all other scenarios
        v = (float(x[a])+float(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = v
    return x

def curve_smoothing(string):
    """
    This function will identify the position of missing value symbols.
    """
    splt = string.split(',')
    d = [i for i, v in enumerate(splt) if v != '_']  # gives the indices of the␣
 ↪elements where value is not '_'
    if d[0] != 0:          # checks whether first index has a digit
        d = [-1] + d
    if d[-1] != len(splt)-1:  # checks whether last index has a digit
        d = d + [-1]
    for (a,b) in zip(d[:-1],d[1:]):
        replace(splt,a,b)
    return splt

S=  ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]
for i in S:
    smoothed_values = curve_smoothing(i)
    print(smoothed_values)
```

```
[6.0, 6.0, 6.0, 6.0]
[20.0, 20.0, 20.0, 20.0, 20.0]
[16.0, 16.0, 16.0, 16.0, 16.0]
[10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
```

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)

Ex:

```
[8]: def compute_conditional_probabilities(F,S):
         """
         This function will print conditional probability.
         """
         num=0
         den=0
         for i in range(len(A)):
             if(A[i][1]==S):
                 den=den+1
                 if(A[i][0]==F):
                     num=num+1
         print('P(F={}|S=={})={}/{}'.format(F, S, str(num), str(den)))


     A =␣
      ↪[['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4',

     K = []
     M = []
     for i in range(len(A)):
         K.append(A[i][0])
         M.append(A[i][1])
     K = list(set(K))
     M = list(set(M))
     K.sort()
     M.sort()

     for k in K:
         for m in M:
             compute_conditional_probabilities(k,m)
```

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

Q9: Given two sentances S1, S2

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

Ex:

[9]:
```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"

# split by space
a = S1.split(' ')
b = S2.split(' ')

#convert to set
c = set(a)
d = set(b)

print("Number of common words between S1, S2 : ",len(c & d)) #Intersection
print("Words in S1 but not in S2",list(c-d))                  #Set difference
print("Words in S2 but not in S1",list(d-c))                  #Set difference
```

```
Number of common words between S1, S2 :  7
Words in S1 but not in S2 ['5', 'first', 'F']
Words in S2 but not in S1 ['3', 'S', '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 $\frac{-1}{8}\cdot((1\cdot log_{10}(0.4)+0\cdot log_{10}(0.6))+(0\cdot log_{10}(0.5)+1\cdot log_{10}(0.5))+...+(1\cdot log_{10}(0.8)+0\cdot log_{10}(0.2)))$

[10]:
```
import math as m

def compute_log_loss(A):
    """
    This function will calculate  ( ,    ) and returns the loss.
    """
    sum = 0
    for i in range(len(A)):
        sum += (A[i][0]*m.log10(A[i][1]))+((1 - A[i][0])*m.log10(1-A[i][1]))   ␣
 ↪#   10(   )+(1- )  10(1-    ))
    loss = -(sum/len(A))
    return loss

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]]
loss = compute_log_loss(A)
```

```python
print("log loss:",loss)
```

```
log loss: 0.42430993457031635
```