parth.pandey13103447@gmail.com_3

December 9, 2019

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 free to change all these codes/structure
     # here A and B are list of lists
     def matrix_mul(A, B):
         # write your code
         if  len(A[0]) != len(B):
             return 'Not Possible'
         p = len(A)
         q = len(A[0])
         r = len(B[0])
         fm = []
         for i in range(p):
             mat = []
             for k in range(r):
                 temp = 0
                 for j in range(q):
                     temp = temp + A[i][j]*B[j][k]
                 mat.append(temp)
             fm.append(mat)
         return fm
     A = [[1,2],
          [3,4]]

     B = [[1 ,4],
                  [5, 6],
                  [7 ,8],
                  [9 ,6]]
     matrix_mul(A, B)
```

```
[1]: 'Not Possible'
```

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.

```
[2]: from random import uniform
# 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 free to change all these codes/structure
def pick_a_number_from_list(A):
    S = sum(A)
    normal_A = [a/S for a in A]
    temp = 0
    cum_sum = []
    for i in normal_A:
        temp = temp + i
        cum_sum.append(temp)
    random_number = uniform(0,1)
    for i in range(len(cum_sum)):
        if random_number < cum_sum[i]:
            index = i
            break

    return A[index]


    # your code here for picking an element from with the probability␣
 ↪propotional to its magnitude
    #.
    #.
    #.
#     return #selected_random_number
A = [0 ,5, 27, 6, 13, 28, 100, 45, 10, 79]
d = {}
def sampling_based_on_magnitued():
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        if number in d.keys():
            d[number] = d[number] + 1
        else:
            d[number] = 1
    print('''This shows the number of times a number if selected at random
    is based upon the magnitude''')
    print('This is also called the propotional sampling')
    for key in sorted(d.keys(),reverse=True):
        print(key,d[key])


sampling_based_on_magnitued()
```

```
This shows the number of times a number if selected at random
    is based upon the magnitude
This is also called the propotional sampling
100 24
79 26
45 12
28 10
27 11
13 7
10 4
5 5
```

Q3: Replace the digits in the string with #

Consider a string that will have digits in that, we need to remove all the characters which are not digits and replace the digits with #

[3]:
```
import re
# 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 free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    # write your code
    #
    temp = []
    for i in String:
        if re.match(r'[a-zA-Z]',i) or not re.match(r'[a-zA-Z0-9]',i):
            continue
        else:
            temp.append('#')


    return ''.join(temp) # modified string which is after replacing the # with␣
↪digits
```

```
String = '#2a$#b%c%561#'
replace_digits(String)
```

[3]:
```
'####'
```

Q4: Students marks dashboard

Consider the marks list of class students given in 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.

[32]:
```
# 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 free to change all these codes/structure
def display_dash_board(students, marks):
    # write code for computing top top 5 students

    x = list(zip(students,marks))
    x.sort(key = lambda z : z[1])
    x
    marks.sort(reverse=True)
    top_5 = []
    for m in marks[:5]:
        for k ,v in x:
            if v == m:
                top_5.append((k,v))
    marks.sort()
    last_5 = []
    for m in marks[:5]:
        for k ,v in x:
            if v == m:
                last_5.append((k,v))
    no_students = len(marks)
    first_percentile = int(round((no_students * 0.25),0))-1
    third_percentile = int(round((no_students * 0.75),0))-1
    percentile = []
    for m in marks[first_percentile+1 : third_percentile]:
        for k,v in x:
            if v == m:
                percentile.append((k,v))
    return (top_5, last_5 , percentile)


    #     top_5_students = # compute this
    # write code for computing top least 5 students
#     least_5_students = # compute this
    # write code for computing top least 5 students
#     students_within_25_and_75 = # compute this

#     return top_5_students, least_5_students, students_within_25_and_75
students =␣
 ↪['student1','student2','student3','student4','student5','student6','student7','student8','s
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
top_5_students, least_5_students, students_within_25_and_75 =␣
 ↪display_dash_board(students, marks)
print('\nTop 5 Students\n')
for student,marks in top_5_students:
    print(student,marks)
print('\nLast 5 Students\n')
for student,marks in least_5_students:
    print(student,marks)
print('\nStudents within 25 and 75\n')
for student,marks in students_within_25_and_75:
    print(student,marks)
```

```
Top 5 Students

student8 98
student10 80
student2 78
student5 48
student7 47

Last 5 Students

student3 12
student4 14
student9 35
student6 43
student1 45

Students within 25 and 75

student9 35
student6 43
```

```
student1 45
student7 47
student5 48
```

Q5: Find the closest points

Consider you are 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 defined as $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

[38]:
```python
import math

# 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 free to change all these codes/structure


# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    # write your code here
    temp = {}
    for x,y in S:
        num = x*P[0] + x*P[1]
        den = math.sqrt(x**2 + y**2 ) * math.sqrt(P[0]**2+ P[1]**2)
        temp[math.acos(num/den)] = (x,y)
    points = []
    for i in sorted(temp.keys())[:5]:
        points.append(temp[i])
    return points  # its list of tuples

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)
for pts in points:
    print(pts) #print the returned values
```

```
(-1, -1)
(-5, -8)
(0, 6)
(1, 2)
(3, 4)
```

Q6: Find which line separates oranges and apples

Consider you are given two set of data points in the form of list of tuples like and set of line equations(in the string format, i.e list of strings)

Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print NO.

[6]:
```python
import math
# 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 strings



# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
#     EXTRACTING COEFFICIENTS
    x_coff = line.split('x')[0]
    y_coff = line.split('x')[1].split('y')[0]
    c_coff = line.split('y')[1]
#     CONVERTING COEFFICIENTS TO FLOAT
    if len(x_coff) > 0 and x_coff[0] == '-':
        x_coff = -1 * float(x_coff[1])
    else:
        x_coff = float(x_coff)

    if y_coff[0] == '-':
        y_coff = -1 * float(y_coff[1])
    else:
        y_coff = float(y_coff[1])

    if c_coff[0] == '-':
        c_coff = -1 * float(c_coff[1:])
    else:
        c_coff = float(c_coff[1:])
#     CALCULATING THE  DIRECTION OF EACH POINT
    dist = {'red':[],'blue':[]}

    for r,b in zip(red,blue):
        temp = r
        dist_r = x_coff * temp[0] + y_coff * temp[1] + c_coff
        temp = b
        dist_b = x_coff * temp[0] + y_coff * temp[1] + c_coff
        dist['red'].append(dist_r)
        dist['blue'].append(dist_b)
#     IF ALL POINTS FOR EITHER OF THE COLOR ARE IN SAME DIREACTION
#     THEN THE LINE SEPERATES THEM PERFECTLY
    temp_r = list(filter(lambda x: x > 0 , dist['red']))
    temp_b = list(filter(lambda x: x < 0 , dist['blue']))
    return len(temp_r) == len(temp_b)


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:
    if i_am_the_one(Red, Blue, i) == True: yes_or_no = 'YES'
    else: yes_or_no = 'NO'
    print(yes_or_no) # the returned value
```

```
YES
NO
NO
YES
```

Q7: Filling the missing values in the specified format

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:

[10]:
```
# 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 strings


# you can free to change all these codes/structure
def curve_smoothing(string):
    # your code
    count = 1
    temp_num = 0
    temp = string.split(',')
    no_of_blanks = len(list(filter(lambda x: x == '_' , temp)))
    dist_num = []
    for x in temp:
        if x!= '_' and count > 0:
            num = int(x) + temp_num
            dist_num.append(num//count)
            temp_num = num//count
            count = 1
            count+=1
        else:
            count += 1
        if len(dist_num) > 0:
            final_dist = temp_num//(count-1)
    dist_num.append(final_dist)
    i =0
    for j in range(len(temp)):
        if temp[j] == '_':
            temp[j] = dist_num[i]
        else:
            i+=1
            temp[j] = dist_num[i]
    return temp

S=  "_,_,30,_,_,_,50,_,_"
# S = "_,_,_,24"
# S = "40,_,_,_,60"
# S = "80,_,_,_,_"
smoothed_values= curve_smoothing(S)
print(','.join(list(map(str,smoothed_values))))
```

```
10,10,12,12,12,12,4,4,4
```

Q8: Find the probabilities

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:

[11]:
```
# 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 strings




# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    # your code
    # print the output as per the instructions
    condition = ['S1','S2','S3']
    probables = ['F1','F2','F3','F4','F5']
    final_ans = []
    for prob in probables:
        temp = []
        for cond in condition:
            total = 0
            fav = 0
            for element in A:
                if element[1] == cond:
                    total+=1
                    if element[0] == prob:
                        fav+=1
            temp.append('P(F={}|S=={})={}/{}'.format(prob,cond,fav,total))
        final_ans.append(temp)
    print(A)
    for i in final_ans:
        print(', '.join(i))
A =␣
 ↪[['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4',
compute_conditional_probabilites(A)
```

```
[['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S3'], ['F3',
'S2'], ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S1']]
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: Operations on sentences

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

Ex:

[12]:
```
# 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 strings

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    count = 0
    diff_word = []
    diff_word1 = []
    for word in S1.split(' '):
        if word in S2.split(' '):
            count+=1
        if word not in S2.split(' '):
            diff_word.append(word)

    for word in S2.split(' '):
        if word not in S1.split(' '):
            diff_word1.append(word)
    return count , diff_word , diff_word1

S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
a,b,c = string_features(S1, S2)
print(a)
print(b)
print(c)
```

```
7
['first', 'F', '5']
['second', 'S', '3']
```

Q10: Error Function

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

[13]:
```
# 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 strings
# IMPORTING MATH FOR LOG
from math import log
# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    loss = 0
    for pair in A:
        loss = loss  + ( pair[0] * log(pair[1],10) + (1- pair[0]) *␣
↪log((1-pair[1]),10))
    return loss * (-1/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]]
loss = compute_log_loss(A)
print(loss)
```

```
0.42430993457031635
```