Python_Mandatory_Assignment

December 20, 2019

Python: without numpy or sklearn

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

[3]:
```
# 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):
    num_rows_A= len(A)
    num_columns_A=len(A[0])
    num_rows_B=len(B)
    num_columns_B=len(B[0])
    if(num_columns_A!=num_rows_B):
        return 0

    else:
        prod=[]
        for n in range(num_rows_A):
            prod_row=[]
            for m in range(num_columns_B):
                prod_row.append(0)
            prod.append(prod_row)

        for i in range(num_rows_A):
            for j in range(num_columns_B):
                for k in range(num_columns_A):
                    prod[i][j]+= (A[i][k]*B[k][j])
        return prod

    #return(#multiplication_of_A_and_B)

A = [[1,2],[3,4]]
B= [[5,6],[7,8]]
product=matrix_mul(A, B)
```

```
if product!=0:
    print('A*B={}'.format(product))
elif product==0:
    print('A*B is not possible')
```

```
A*B=[[19, 22], [43, 50]]
```

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.

[10]:
```
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

count=0
# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    # your code here for picking an element from with the probability␣
 ↪propotional to its magnitude
    sum=0
    prob_lst=[]
    cumulative_sum=0
    cumulative_lst=[]
    for i in A:
        sum+=i
    for i in A:
        prob_lst.append(i/sum)
    for i in prob_lst:
        cumulative_sum+=i
        cumulative_lst.append(cumulative_sum)
    num=uniform(0,1)
    for n in cumulative_lst:
        if num<n:
            index=cumulative_lst.index(n)
            number=A[index]
            return number


def sampling_based_on_magnitued(A):
    A.sort()
    count=[]
    for i in range(len(A)):
```

```
        count.append(0)
    for i in range(1,101):
        number = pick_a_number_from_list(A)
        print(number)
        for n in A:
            if number==n:
                count[A.index(number)]+=1
    for i in range(len(A)):
        print('Number of times {} got selected is {}'.format(A[i],count[i]))

        #print(number)

A = [1,5,27,16,60,35,130,105,10,79]
sampling_based_on_magnitued(A)
```

```
130
105
130
79
60
79
60
105
60
130
130
130
130
130
27
105
27
105
105
79
60
105
105
105
130
130
130
79
130
105
60
60
```

79
130
79
27
60
130
105
105
79
35
1
27
10
105
130
79
27
130
27
105
130
60
79
16
130
79
60
105
60
130
130
10
130
79
79
130
130
16
27
105
130
130
105
130
105
105
60
130

```
10
105
130
130
105
130
10
130
79
130
60
79
79
79
105
130
27
79
105
105
Number of times 1 got selected is 1
Number of times 5 got selected is 0
Number of times 10 got selected is 4
Number of times 16 got selected is 2
Number of times 27 got selected is 8
Number of times 35 got selected is 1
Number of times 60 got selected is 12
Number of times 79 got selected is 17
Number of times 105 got selected is 23
Number of times 130 got selected is 32
```

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 #

[11]:
```python
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
    x='#'*len(re.findall('\d',String))
    return x
```

```
    # modified string which is after replacing the # with digits

A = input('Enter sequence: ')
print('Output: {}'.format(replace_digits(A)))
```

```
Enter sequence: a2b34
Output: ###
```

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.

percentile function is taken from https://code.activestate.com/recipes/511478-finding-the-percentile-of-the-values/

[12]:
```
def percentile(N, percent, key=lambda x:x):
    """
    Find the percentile of a list of values.

    @parameter N - is a list of values. Note N MUST BE already sorted.
    @parameter percent - a float value from 0.0 to 1.0.
    @parameter key - optional key function to compute value from each element␣
 ↪of N.

    @return - the percentile of the values
    """
    if not N:
        return None
    k = (len(N)-1) * percent
    f = k//1
    c = -(-k//1)
    if f == c:
        return key(N[int(k)])
    d0 = key(N[int(f)]) * (c-k)
    d1 = key(N[int(c)]) * (k-f)
    return d0+d1
```

[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 examples
```

```
# you can free to change all these codes/structure
def display_dash_board(students, marks):
    #putting the students and marks in a dictionary
    stud_dict=dict(zip(students,marks))

    stud_list=list(stud_dict.keys())
    marks_list=list(stud_dict.values())
    sorted_marks_list=sorted(marks_list,reverse=True)

    #printing the list of top 5 students with highest marks
    top_5_students_marks=[]
    print('a.')
    for m in sorted_marks_list[:5]:
        print('{} {}'.format(stud_list[marks_list.index(m)],m))

    #printing the list of bottom 5 students with least marks
    bottom_5_students_marks=[]
    sorted_marks_list=sorted(marks_list)
    print('b.')
    for m in sorted_marks_list[:5]:
        print('{} {}'.format(stud_list[marks_list.index(m)],m))


    #students_within_25_and_75
    print('c.')
    percentile_75=percentile(sorted_marks_list,.75)
    percentile_25=percentile(sorted_marks_list,.25)
    percentile_25_to_75=[]
    for i in marks_list:
        if (i>=percentile_25) and (i<=percentile_75):
            percentile_25_to_75.append(i)
    for i in percentile_25_to_75:
        print('{} {}'.format(stud_list[marks_list.index(i)],i))


Students=['student1','student2','student3','student4','student5','student6','student7','studen
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.
student1 45
student5 48
student6 43
student7 47
```

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 $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

[19]:
```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):
    cos_distance_lst=[]
    closest_points_to_p=[]
    for point in S:
        x=point[0]
        y=point[1]
        numerator=x*P[0]+y*P[1]
        denominator= (((x**2)+(y**2))**0.5)*(((P[0]**2)+(P[1]**2))**0.5)
        cos_distance=math.acos(numerator/denominator)
        cos_distance_lst.append(cos_distance)

    sorted_cos_distance_lst=sorted(cos_distance_lst)

    for distance in sorted_cos_distance_lst[:5]:
        closest_points_to_p.append(S[cos_distance_lst.index(distance)])


    return closest_points_to_p  # 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)
print('The closest points to p are')
for p in points:
    print(p) #print the returned values
```

```
The closest points to p are
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

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.

[20]:
```
import math
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 strings


# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    line=(re.split("x|y|=",line))
    Red_in_line=[]
    Blue_in_line=[]
    count_Positive_r=0
    count_Negative_r=0
    count_Positive_b=0
    count_Negative_b=0
    for r in red:
        Red_in_line.
 ↪append(r[0]*float(line[0])+r[1]*float(line[1])+float(line[2]))
    for b in blue:
        Blue_in_line.
 ↪append(b[0]*float(line[0])+b[1]*float(line[1])+float(line[2]))

    for r in Red_in_line:
```

```
        if r>0:
            count_Positive_r+=1
        elif r<0:
            count_Negative_r+=1
    for b in Blue_in_line:
        if b>0:
            count_Positive_b+=1
        elif b<0:
            count_Negative_b+=1
    if count_Positive_r == len(red)and count_Negative_b==len(blue):
        return 'YES'
    elif count_Negative_r == len(red)and count_Positive_b==len(blue):
        return 'YES'
    else:
        return 'NO'



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:
    Ans=i_am_the_one(Red, Blue, i)
    print(Ans)
```

```
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:

[21]:
```
# 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):
    string_lst=string.split(',')
    print(string_lst)
    _count=0
    num_count=0
    _index=None
    num1=None
    num1_index=None

    for i in range(len(string_lst)):
        if string_lst[i]=='_':
            if _count==0:
                _index=i
                _count=_count+1
            elif _count>0 and num_count==1 and i==len(string_lst)-1:
                _count=_count+1
                val=int(num1/(_count+num_count))
                for c in range(num1_index,i+1):
                    string_lst[c]=val
            else:
                _count=_count+1

        else:
            if _count>0 and num_count==0:
                num1=float(string_lst[i])
                num1_index=i
                num_count=num_count+1

                val=int(float(string_lst[i])/(_count+num_count))
                num1=val
                _count=0
                for c in range(_index,num1_index+1):
                    string_lst[c]=val
            elif _count>0 and num_count>0:
                num_count=num_count+1
                val=int((float(string_lst[i])+num1)/(_count+num_count))

                for c in range(num1_index,(i+1)):
                    string_lst[c]=val
                num_count=1
                num1=float(string_lst[i])
                num1_index=i
                _count=0
            elif _count==0 and num_count==0:
                num1=float(string_lst[i])
```

```
                num1_index=i
                num_count=num_count+1

    return string_lst


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

```
['_', '_', '30', '_', '_', '_', '50', '_', '_']
[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:

[22]:
```
# 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):
    FS={
        'F1S1':0,
        'F1S2':0,
        'F1S3':0,
        'F2S1':0,
        'F2S2':0,
        'F2S3':0,
        'F3S1':0,
        'F3S2':0,
        'F3S3':0,
        'F4S1':0,
        'F4S2':0,
        'F4S3':0,
        'F5S1':0,
        'F5S2':0,
        'F5S3':0
    }
    S=[0,0,0]
```

```
    for i in A:
        FS_combined=i[0]+i[1]
        FS[FS_combined]+=1
        if i[1]=='S1':
            S[0]+=1
        if i[1]=='S2':
            S[1]+=1
        if i[1]=='S3':
            S[2]+=1

    print("a. P(F=F1|S==S1)={}/{}, P(F=F1|S==S2)={}/{}, P(F=F1|S==S3)={}/{}"
          .format(FS['F1S1'],S[0],FS['F1S2'],S[1],FS['F1S3'],S[2]))
    print("b. P(F=F2|S==S1)={}/{}, P(F=F2|S==S2)={}/{}, P(F=F2|S==S3)={}/{}"
          .format(FS['F2S1'],S[0],FS['F2S2'],S[1],FS['F2S3'],S[2]))
    print("c. P(F=F3|S==S1)={}/{}, P(F=F3|S==S2)={}/{}, P(F=F3|S==S3)={}/{}"
         .format(FS['F3S1'],S[0],FS['F3S2'],S[1],FS['F3S3'],S[2]))
    print("d. P(F=F4|S==S1)={}/{}, P(F=F4|S==S2)={}/{}, P(F=F4|S==S3)={}/{}"
         .format(FS['F4S1'],S[0],FS['F4S2'],S[1],FS['F4S3'],S[2]))
    print("e. P(F=F5|S==S1)={}/{}, P(F=F5|S==S2)={}/{}, P(F=F5|S==S3)={}/{}"
         .format(FS['F5S1'],S[0],FS['F5S2'],S[1],FS['F5S3'],S[2]))




A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],
     ['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]

compute_conditional_probabilites(A)
```

```
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
```

Q9: Operations on sentences

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

Ex:

[23]:
```
# 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):
```

```
    S1=S1.split()
    S2=S2.split()
    common_words=[]
    common_words_count=0
    S1_unique_words=[]
    S2_unique_words=[]
    for s in S1:
        if s in S2:
            common_words.append(s)
            common_words_count+=1
    for s in S1:
        if s not in common_words:
            S1_unique_words.append(s)
    for s in S2:
        if s not in common_words:
            S2_unique_words.append(s)

    return common_words_count,S1_unique_words,S2_unique_words


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.{}'.format(a))
print('b.{}'.format(b))
print('c.{}'.format(c))
```

```
a.7
b.['first', 'F', '5']
c.['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 $\frac{-1}{8}\cdot((1\cdot log_{10}(0.4)+0\cdot log_{10}(0.6))+(0\cdot log_{10}(0.5)+1\cdot log_{10}(0.5))+...+(1\cdot log_{10}(0.8)+0\cdot log_{10}(0.2)))$

[24]:
```
# 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
import math as m
```

```
# you can free to change all these codes/structure
def compute_log_loss(A):
    n=len(A)
    lst=[]
    sum=0
    for i in A:
        val=(i[0]*m.log10(i[1]))+(1-i[0])*m.log10(1-i[1])
        lst.append(val)
    for i in lst:
        sum=sum+i
    return (-1/n)*sum




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
```