1_Python_Assignment (1)

November 8, 2021

Python: without numpy or sklearn

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

[ ]:
```
A=[
    [1,2,3],
    [4,5,6],
    [7,8,9]
]
B=[
    [1,2,0,4],
    [0,1,0,4],
    [0,0,1,4]
]
#creating the empty matrix of values 0 with A rows and B colums
C=[]
for a in range(len(A)):
    emp=[]
    for b in range(len(B[0])):
        emp.append(0)
    C.append(emp)

print("The matrix A dimension is",len(A),len(A[0]))
print("The matrix B dimension is",len(B),len(B[0]))
# "A & B.shape" in numpy

def mul_matx(A,B):
    for i in range(len(A)):#iteration on rows of A
        for j in range(len(B[0])):#iteration on colums of B
            for x in range(len(B)):#iteration on raws of B
                C[i][j]+=A[i][x]*B[x][j] #mul of A(i,x) with B(x,j) for C(i,j)

    return C;

if len(B)==len(A[0]):
    mul_matx(A,B)
    print(C)
else:
```

```
    print("multipication of given matrix is not possible")
```

```
The matrix A dimension is 3 3
The matrix B dimension is 3 4
[[1, 4, 3, 24], [4, 13, 6, 60], [7, 22, 9, 96]]
```

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.

[29]:
```
from random import uniform

def step1(A):
  #compute the sum
  s=sum(A)
  #print(s)
  ##Normalizing using the sum
  d=[]
  for i in range(0,len(A)):
    d.append(A[i]/s)
  #print(d)
  ###cumulative normalizing sum
  c=[]
  c.append(d[0])
  for i in range(1,len(A)):
    c.append(c[i-1] + d[i])
  #print(c)
  ################################## sample one values
  r=uniform(0,1.0)
  ################################## proportional sampling
  for i in range(len(A)):
    if r<= c[i]:
      return i



A = [0,5,27,6,13,28,100,45,10,79]
prob=[]
for i in range(len(A)):
  prob.append(0)

for i in range(100): #sampling hundred times
  index=step1(A)
  prob[index]=prob[index]+1
  #print("number=",A[index],"index=",index)
```

```
for i in range(len(A)):
  print("number",A[i],"is sampled",prob[i],"time")
print(prob)
```

```
number 0 is sampled 0 time
number 5 is sampled 3 time
number 27 is sampled 10 time
number 6 is sampled 1 time
number 13 is sampled 10 time
number 28 is sampled 4 time
number 100 is sampled 29 time
number 45 is sampled 16 time
number 10 is sampled 2 time
number 79 is sampled 25 time
[0, 3, 10, 1, 10, 4, 29, 16, 2, 25]
```

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 #

[ ]:
```
def replace_digits(S):
  for i in range(0,len(S)):
    x=S[i]
    if x.isnumeric():
      print(end="#")

S = input("enter the string ")
replace_digits(S)
```

```
enter the string #2a$#b%c%561#
####
```

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

[36]:
```
class stud:
  name=""
  marks=0
  def __init__(self, name, marks):#class variables
    self.name=name
    self.marks=marks
```

```
def display_dash_board(stud):
  stud.sort(key=lambda s:s.marks,reverse=True)
  for i in range(0,5):
    print(stud[i].name,stud[i].marks)
  print(sep='\n')
  max=stud[0]
  students.sort(key=lambda s:s.marks)
  for i in range(0,5):
    print(stud[i].name,stud[i].marks)
  print(sep='\n')
  min=stud[0]

  diff=max.marks-min.marks
  pre25=diff*.25
  pre75=diff*.75
  #print(diff,pre25,pre75)

  for i in range(len(stud)):
    if pre25 < stud[i].marks < pre75:
      print(stud[i].name,stud[i].marks)

  #for i in range(0,len(stud)):
    #x=int((i/len(stud))*100)  #Formula for percentile
    #print(x)
    #if (x>25) & (x<75):
      #print(stud[i].name,stud[i].marks)
      #print(x)



s1=stud("a",9)
s2=stud("b",89)
s3=stud("c",45)
s4=stud("e",34)
s5=stud("f",2)
s6=stud("g",98)
s7=stud("h",67)
s8=stud("i",87)
s9=stud("j",32)
s10=stud("k",76)

students=[s1,s2,s3,s4,s5,s6,s7,s8,s9,s10]

display_dash_board(students)
```

```
g 98
b 89
i 87
```

```
k 76
h 67

f 2
a 9
j 32
e 34
c 45

96 24.0 72.0
j 32
e 34
c 45
h 67
```

https://www.tutorialsteacher.com/python/list-sort

https://www.youtube.com/watch?v=ZDa-Z5JzLYM

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

[ ]:
```python
import math
def dist(S,P):
  #print(S,P)
  x=S[0]
  y=S[1]
  p=P[0]
  q=P[1]
  r1=math.sqrt(x**2+y**2)
  r2=math.sqrt(p**2+q**2)
  a=(x*p+y*q)/(r1*r2)
  k=math.acos(a)
  return k



def clo_poin(S,P):
  #list=[]
  l1=[]
  #print(len(S))

  for i in range(0,len(S)):
    l1.append(dist(S[i],P))
```

```
  for i in range(0,5):
    x=S.pop(l1.index(min(l1))) # poping S list index where the min value of l1 list
    print(x)



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

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

[ ]:
```
import math
x=2
y=3
p=1
q=5
r1=math.sqrt(x**2+y**2)
r2=math.sqrt(p**2+q**2)
a=(x*p+y*q)/(r1*r2)
k=math.acos(a)
print(a,r1,r2,k)
```

```
0.924678098474716 3.605551275463989 5.0990195135927845 0.3906070436976867
```

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

[ ]:
```
import math


def value_generator(S):     # this function returns the values of X1 X2 and X3(y-intercept)
  o=1
  for i in range (len(S)):
    #print(i)
    #print(S[i])
    a=S[i]
    if a.isnumeric():
```

```
      b=int(a)
      b=b*o            # To determine whether number is +Ve or -Ve
      if i==3:
        y=b
      elif i==6:
        c=b
      elif i==0:
        x=b
    if a == "-":
      o=-1
  return x,y,c;
  ##############################

def dist(x,y,c,p):      # just to calculate the distance between point to a␣
 ↪line
  a=p[0]
  b=p[1]
  n=x*a+y*b+c
  de=math.sqrt(a**2+b**2)
  dis=n/de
  return dis
  ##############################

def cal(red,blue,lines):         #Calculating the distance of point from the␣
 ↪line and based on +ve or -ve value we differentiate the points are onr side␣
 ↪or the other
  for k in range (0,len(lines)):
    x,y,c=value_generator(lines[k])
    print(x,y,c)
    red_list=[]
    blue_list=[]
    for i in range(0,len(Red)):
      red_list.append(dist(x,y,c,Red[i])) # getting the distance and appending␣
 ↪it in list (red point)
    print(red_list)
    for i in range(0,len(Blue)):
      blue_list.append(dist(x,y,c,Blue[i])) # same as above but for blue point
    print(blue_list)
    if (min(red_list)>0) & (max(blue_list)<0): # checking it is one sided or␣
 ↪not of line
      print("############----->yes")
    elif (max(red_list)<0) & (min(blue_list)>0): # checking on other side of␣
 ↪line
      print("############----->yes")
    else:
      print("xxxxxxxxx----->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"]

cal(Red, Blue, lines)
```

```
1 1 0
[1.414213562373095, 1.3416407864998738, 1.3416407864998738, 1.3416407864998738,
0.7276068751089989]
[-1.3416407864998738, -1.3416407864998738, -1.386750490563073,
-1.2649110640673518, -0.6324555320336759]
############----->yes
1 -1 0
[0.0, 0.4472135954999579, 0.4472135954999579, -0.4472135954999579,
-1.212678125181665]
[-0.4472135954999579, 0.4472135954999579, -0.2773500981126146,
-0.6324555320336759, 1.2649110640673518]
xxxxxxxxx----->no
1 0 -3
[-1.414213562373095, -0.4472135954999579, 0.22360679774997896,
-0.22360679774997896, -0.9701425001453319]
[-2.23606797749979, -1.7888543819998317, -1.6641005886756874,
-1.8973665961010275, -0.6324555320336759]
xxxxxxxxx----->no
0 1 0
[0.7071067811865475, 0.4472135954999579, 0.4472135954999579, 0.8944271909999159,
0.9701425001453319]
[-0.4472135954999579, -0.8944271909999159, -0.5547001962252291,
-0.31622776601683794, -0.9486832980505138]
############----->yes
```

https://brilliant.org/wiki/distance-between-point-and-line/

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:

[ ]:
```
def curve_smoothing(S):
  space=0
  l1=[]
  for i in range(0,len(S)):
```

```
    x=S[i]
    if x.isnumeric():
      l1.append(int(x))      #get value in l1
      #print(l1,"l1 in num")
    elif x == "_":
      space=space+1
      print(space,"|^space")
    elif x.startswith(","):
      val=0
      for j in range(0,len(l1)):
        val=10*j*l1.pop()                  #converting it to a proper value
      if val>0:
        v=val
        v=v//space
        print((v,)*space)
        print(v,"v")
      print(val,"val")

  return 0

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

```
1 |^space
0 val
2 |^space
0 val
(15, 15)
15 v
30 val
3 |^space
0 val
4 |^space
0 val
5 |^space
0 val
(10, 10, 10, 10, 10)
10 v
50 val
6 |^space
0 val
7 |^space
0
```

[ ]:
```
# takes an array x and two indices a,b.
# Replaces all the _'s with (x[a]+x[b])/(b-a+1)
```

```
def fun(x, a, b):
    if a == -1:
        v = float(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = v
    elif b == -1:
        v = float(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = v
    else:
        v = (float(x[a])+float(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = v
    return x

def replace(text):
    # Create list from the string
    x = text.replace(" ","").split(",")
    #print(x,"x")
    # Get all the pairs of indices having number
    y = [i for i, v in enumerate(x) if v != '_']
    #print(y,"y")
    if y[0] != 0:
        y = [-1] + y
        #print(y)
    # Ending with _ ?
    if y[-1] != len(x)-1:
        y = y + [-1]
    # run over all the pairs
    for (a, b) in zip(y[:-1], y[1:]):
        fun(x,a,b)
        #print(x,a,b,"x a b")
    return x

# Test cases
tests = [
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]

for i in tests:
    print (replace(i))
```

```
[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]
```

I had tryed for logic but couldn't able to come with any logic so gone with this link https://stackoverflow.com/questions/57179618/filling-the-missing-values-in-the-specified-format-python/57180534 and try to reverse engineer the code and understood the same.

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:

[ ]:
```
def denominator(a,s1,s2,s3): # calculating the denominator
  b=a[0]
  c=a[1]
  #print(b,c)
  if c=="S1":
    s1=s1+1
  elif c=="S2":
    s2=s2+1
  elif c=="S3":
    s3=s3+1
  return s1,s2,s3


def numerator(a,l1): # numerator
  b=a[0]
  c=a[1]
  for i in range(1,6):
    if  str(b).endswith(str(i)):
      for j in range(1,4):
        if str(c).endswith(str(j)):
          l1[i-1][j-1]=l1[i-1][j-1]+1

def print_pr(n,s1,s2,s3): #printing the values
  a=n[0]
  b=n[1]
  c=n[2]
  for i in range(3):
    if i==0:
            print("with S1",a/s1)
    elif i==1:
            print("with S2",b/s2)
    elif i==2:
            print("with S3",c/s3)
```

```
def compute_conditional_probabilites(A,l1):
  s1=s2=s3=0
  for i in range(len(A)):
    s1,s2,s3=denominator(A[i],s1,s2,s3)
  for i in range(len(A)):
    numerator(A[i],l1)
  for i in range(1,6):
    print("probablity of F",i)
    print_pr(l1[i-1],s1,s2,s3)


l1 = [[0,0,0],
      [0,0,0],
      [0,0,0],
      [0,0,0],
      [0,0,0]
      ]

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

compute_conditional_probabilites(A,l1)
```

```
probablity of F 1
with S1 0.25
with S2 0.3333333333333333
with S3 0.0
probablity of F 2
with S1 0.25
with S2 0.3333333333333333
with S3 0.3333333333333333
probablity of F 3
with S1 0.0
with S2 0.3333333333333333
with S3 0.3333333333333333
probablity of F 4
with S1 0.25
with S2 0.0
with S3 0.3333333333333333
probablity of F 5
with S1 0.25
with S2 0.0
with S3 0.0
```

Q9: Given two sentances S1, S2

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

Ex:

[ ]:
```
def string_features(S1, S2):
  l1=S1.split()
  l2=S2.split()
  a=0
  b=l1.copy()
  c=l2.copy()
  for i in range(len(l1)):
    for j in range(len(l2)):
      if l1[i]==l2[j]:
        a=a+1                   #number of comman words
        c.remove(l1[i])         #uni to s1
        b.remove(l2[j])         #uni to s2
  print(a,b,c)


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

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

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

[ ]:
```
import math

def sub_cal(a): #calculate each element
  y=a[0]
  y_s=a[1]
  x=(y*(math.log10(y_s)))+((1-y)*(math.log10(1-y_s)))
  return x

def compute_log_loss(A):
  l1=[]
  for i in range(len(A)):
    l1.append(sub_cal(A[i]))
  #print(len(A))
  x= -sum(l1)/len(A)
  print(x)
  #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]]
compute_log_loss(A)
#print(# the above loss)
```

```
0.42430993457031635
```