1.Python Assignment

June 25, 2020

Python: without numpy or sklearn

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

[1]:
```
def matrix_mul(A, B):
    print("Given A matrix is {} * {} matrix".format(len(A), len(A[0])))
    print("Given B matrix is {} * {} matrix".format(len(B), len(B[0])))

    records_A = len(A)
    columns_B = len(B[0])
    rows_B = len(B)

    if len(A[0]) != len(B):
        print("Not possible")

    else:
        print("Resultant matrix is {} * {} matrix".format(records_A ,␣
 ↪columns_B))

        matrix = []

        for i in range(records_A):
            matrix.append([int(i) for i in ['0']*columns_B])

        for i in range(records_A):
            # iterating by coloum by B
            for j in range(columns_B):
                # iterating by rows of B
                for k in range(rows_B):
                    matrix[i][j] += A[i][k] * B[k][j]
        return matrix

#Note: Above example answer is wrong for below first one, I've cross checked␣
 ↪with online calculator

A   = [[1, 2],
       [3, 4]]
B   = [[1, 2, 3, 4, 5],
       [5, 6, 7, 8, 9]]

# A   = [[1, 2],[3, 4]]
# B   = [[1, 4], [5, 6], [7, 8], [9, 6]]

# A   = [[1, 3, 4], [2, 5, 7], [5, 9, 6]]
# B   = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

matrix_mul(A, B)
```

```
Given A matrix is 2 * 2 matrix
Given B matrix is 2 * 5 matrix
Resultant matrix is 2 * 5 matrix
```

[1]:
```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

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]:
```
import random
from random import uniform

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

def random_sampling_based_on_magnitued(x):
    x.sort()
    weights = [i/sum(x) for i in x]
    return random.choices(x,weights,k=100)

print(random_sampling_based_on_magnitued(A))
```

```
[79, 79, 79, 100, 79, 79, 100, 79, 45, 27, 79, 100, 28, 13, 45, 100, 45, 79, 79,
79, 100, 79, 79, 79, 28, 45, 45, 100, 100, 10, 79, 28, 100, 79, 79, 79, 45, 100,
100, 100, 100, 79, 100, 10, 27, 79, 27, 27, 100, 100, 100, 28, 100, 27, 100,
100, 79, 45, 100, 13, 79, 10, 79, 100, 28, 79, 45, 10, 45, 79, 28, 100, 79, 28,
13, 100, 28, 100, 100, 100, 79, 100, 28, 45, 45, 100, 100, 45, 100, 100, 45, 79,
79, 100, 100, 45, 79, 100, 28, 79]
```

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(A):
    A = re.sub('[A-Za-z]+',"", A)
    if len(A) > 0:
        A = re.sub("\W", "", A, flags=re.I)
        A = re.sub("\d",'#', A)
        return A

    else:
        return print('empty string')

String = "#2a$#b%c%561#"      #Output: ####

replace_digits(String)
```

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

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

[4]:
```
import math

Students=['student1','student2','student3','student4','student5','student6','student7','studer
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]


def percentile(marks, length):
    percentiles=list()
    size = len(marks)
    for j in range(length[0], length[1]):
        percentiles.append(sorted(marks)[int(math.ceil((size * j) / 100)) - 1])
    return sorted(list(set(percentiles)))[:5]

def display_dash_board(students, marks):
    d=dict(zip(students, marks))

    # write code for computing top top 5 students
    top_5_students = sorted(d.items(), key=lambda x: x[1], reverse=True)[:5]

    # write code for computing top least 5 students
    least_5_students = sorted(d.items(), key=lambda x: x[1], reverse=False)[:5]

    # write code for computing top least 5 students
    index = [marks.index(i) for i in percentile(marks, (25,76))]
    d = dict(zip(marks, students))
    marks_percntle = sorted([marks[i] for i in index])
    stdns_percntle = [d.get(i) for i in marks_percntle]
    students_within_25_and_75 = dict(zip(stdns_percntle, marks_percntle))

    return top_5_students, least_5_students, students_within_25_and_75



top_5_students, least_5_students, students_within_25_and_75 =␣
 ↪display_dash_board(Students, Marks)

print(top_5_students, "\n", least_5_students, "\n", students_within_25_and_75,␣
 ↪"\n")
```

```
[('student8', 98), ('student10', 80), ('student2', 78), ('student5', 48),
('student7', 47)]
 [('student3', 12), ('student4', 14), ('student9', 35), ('student6', 43),
('student1', 45)]
 {'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 $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

[5]:
```
import math

def closest_points_to_p(X, Y):
    nearest_points=[]
    p, q = Y
    for i in X:
        x, y = i[0], i[1]
        temp1 = (*+*)
        temp2 = (math.sqrt(**2+**2))*(math.sqrt(**2+q**2))
        coords = math.acos(temp1 /temp2)
        nearest_points.append([i, coords])
    nearest_points.sort(key=lambda x: x[1])
    return [i[0] for i in nearest_points][:5]

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(points)
```

```
[(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]:
```python
import math
import re

def i_am_the_one(red,blue,line):
    a,b,c = line[0], line[1], line[2]
    r, bl = [],[]

    sign = lambda x: True if x>0 else False if x<0 else 0

    for i in red:
        x,y = float(i[0]),float(i[1])
        temp = a*x+b*y+c
        r.append(sign(temp))
    for i in blue:
        x,y = float(i[0]),float(i[1])
        temp = a*x+b*y+c
        bl.append(sign(temp))

    if all(r) == True and all(bl) == False or all(r) == False and all(bl) ==␣
 ↪True:
        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:
    x,y,c = [float(i) for i in re.split("x|y", i)]

    yes_or_no = i_am_the_one(Red, Blue, [x,y,c])

    print(yes_or_no)
```

```
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:

`https://stackoverflow.com/questions/57179618/filling-the-missing-values-in-the-specified-fo`

[7]:
```
# 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 array from the string
    x = text.replace(" ","").split(",")
    # Get all the pairs of indices having number
    y = [i for i, v in enumerate(x) if v != '_']
    # Starting with _ ?
    if y[0] != 0:
        y = [-1] + y
    # Ending with _ ?
    if y[-1] != len(x)-1:
        y = y + [-1]
    # run over all the pairs
    print(dict(zip(y[:-1], y[1:])))
    for (a, b) in zip(y[:-1], y[1:]):
        fun(x,a,b)
    return x

# Test cases
string = ["20,_,_,30,_,_,_,50,_,_"]

for i in string:
    print (replace(i))
    break
```

```
{0: 3, 3: 7, 7: -1}
```

```
[12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 12.5, 4.166666666666667, 4.166666666666667,
4.166666666666667]
```

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]:
```
import re

pairs = []
condition = []

def compute_conditional_probabilites(A):
    for i in range(len(A)):
        k = A[i][0]+A[i][1]
        condition.append(A[i][1])
        pairs.append(k)
    print(pairs)
    print(condition)

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

compute_conditional_probabilites(A)

conditions = ["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"]

conditions = [i.split(",") for i in conditions][0]
conditions = [i.strip() for i in conditions]

for i in conditions:
    i = i.strip("P(F")
    i = i.replace('S==', "")
    i = i.strip("=")
    i = i.replace(")=", ",")
    prop = i.split(",")[1]
    pair = i.split(",")[0].replace("|", "")
    cond = i.split(",")[0].split("|")[1]

    print("conditional probability {} ie.,".format(i),(pairs.count(pair)/
 ↪condition.count(cond)))
```

```
['F1S1', 'F2S2', 'F3S3', 'F1S2', 'F2S3', 'F3S2', 'F2S1', 'F4S1', 'F4S3', 'F5S1']
['S1', 'S2', 'S3', 'S2', 'S3', 'S2', 'S1', 'S1', 'S3', 'S1']
```

```
conditional probability F1|S1,1/4 ie., 0.25
conditional probability F1|S2,1/3 ie., 0.3333333333333333
conditional probability F1|S3,0/3 ie., 0.0
conditional probability F2|S1,1/4 ie., 0.25
conditional probability F2|S2,1/3 ie., 0.3333333333333333
conditional probability F2|S3,1/3 ie., 0.3333333333333333
conditional probability F3|S1,0/4 ie., 0.0
conditional probability F3|S2,1/3 ie., 0.3333333333333333
conditional probability F3|S3,1/3 ie., 0.3333333333333333
conditional probability F4|S1,1/4 ie., 0.25
conditional probability F4|S2,0/3 ie., 0.0
conditional probability F4|S3,1/3 ie., 0.3333333333333333
conditional probability F5|S1,1/4 ie., 0.25
conditional probability F5|S2,0/3 ie., 0.0
conditional probability F5|S3,0/3 ie., 0.0
```

Q9: Given two sentances S1, S2
You will be given two sentances S1, S2 your task is to find
Ex:

[9]:
```python
def string_features(S1, S2):

    S1 = S1.split(" ")
    S2 = S2.split(" ")

    corpus = []
    tot_len = len(S1) + len(S2)
    corpus = corpus + S1 + S2
    length = tot_len - len(list(set(corpus)))

    S1_unique = []
    for i in [i if i not in S2 else None for i in S1]:
        if i != None:
            S1_unique.append(i)

    S2_unique = []
    for i in [i if i not in S1 else None for i in S2]:
        if i != None:
            S2_unique.append(i)

    return length, S1_unique, S2_unique


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. Number of common words between S1, S2 : ', a)
print("b. Words in S1 but not in S2 : ", b)
print("c. Words in S2 but not in S1 : ", c)
```

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

[10]:
```
import math

def compute_log_loss(A):
    n = len(A)
    y, y_score = [i for i in zip(*A)]
    sum = 0
    for i in range(len(y)):
        sum += y[i] * math.log10(y_score[i]) + (1-y[i]) * math.
 ↪log10((1-y_score[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
```