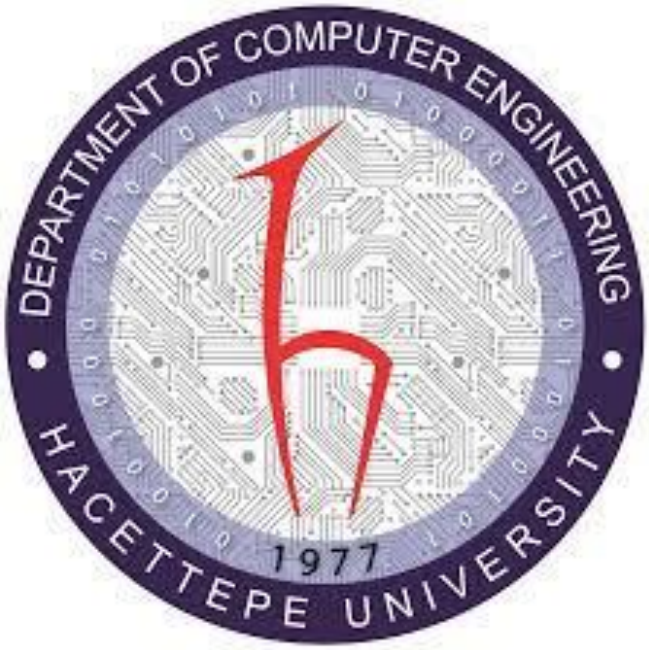
HACETTEPE UNIVERSITY
Department of Computer Science

Course: BBM103 Fall 2022

Assignment 2: Doctor's Aid

Mustafa Ege
2210356088

22.11.2022



Analysis

In this assignment, the main problem is to get the input from an external text file and defining some functions to create an output which is another text file. The input in the text file is not regular and needs some adjustments in order to become usable for the code. For the create, remove, list, probability and recommendation functions, there needs to be a list for the patients' data. Create function needs to add the patients to this list and remove needs to delete from the list. List function needs to get the data from the list and write it to the output file as an organized table. Probability function is not the basic possibility, it needs to calculate false positive value by using some extra data from the user. Recommendation function needs to call probability function to get the probability of having the disease, but while doing this it shouldn't write the probability function's output to the text file. For all the functions except list, functions need to check for other possible situations, such as create function trying to create the same patient or remove function trying to delete a non-existing patient. Lastly, there should be an output function to write all the things functions do throughout the code. After all the functions are completed, these functions need to be called one by one in the order of the input file.

Design

Reading the Input

I created the read_the_input_file function to get the input from doctors_aid_inputs.txt
It returns the input line as a list whose elements are the lines.

Organizing the Input

read_the_input_file function returns every line as an element but I need them as another list. (I need a nested list) Therefore, I created organized_input function. It transforms every word in a line to an element in a list by splitting from ", ". Also; in the input file, between the first and the second word there is no comma so it creates a problem, but with connecting 2 lists I solved the problem.

Adding the Patients to the Pystem

Before creating a create function, I defined a patients list to keep all the patients data throughout the code. Inside the create function, there is another list called patients_list_name and this list is to check if the person which is being created is already in the list or not. It enters a loop, if the input name exists in the patient's name list, function writes it can't create due to duplication. If doesn't exist in the list, function takes the input without its first element which is the 'create' word and appends all of it to the patient variable. After that, patient gets appended to the general patients list. This is for making it a nested list. Finally, function writes that the patient is recorded.

Deleting Patients from the System

This function works in the same way with create function but doing the opposite thing. Basically, it checks if the patient who is wanted to be removed exists in the list. If it does, it enters a for loop, if the name in the input and the patient name matches, whole information about that person gets deleted, at the same time function writes that the patient is removed. If the patient doesn't exist in the list, function writes that the patient can't be removed due to absence.

Listing the Patients

The way I needed to write the patient information is different than the way it is in the input file. So first I changed diagnosis accuracy and treatment risk in the create function and made the numbers same with the required output. In the list function I added '%' at the end of every item of diagnosis accuracy and treatment risk.
To make a table I needed the info titles like patient name, diagnosis accuracy and goes on. So I wrote them and appended them to 2 different list by the first word and second word.
The last problem is making the this table align properly with its columns. In order to achieve that, I used format function and right padding. I took the patients from patients list one by one with a for loop and added them to format. With {:n} this kind of writing format function allows me to make every column have definite amount of space.

Calculating Probability

To calculate the probability of having the disease, I need to get the diagnosis accuracy and disease incidence by its numerator and denominator. To do this, I need to use the index of these informations but I need the patient's index inside the patients list as well. I created another list called which_patient, it returns the index of the wanted patient. By using this function and using the other indexes, I calculated the false positive by this method:
Total number of diagnosed people = (correctly diagnosed people) + (total number of people * (100 – diagnose accuracy)) / 100
False positive = (number of correctly diagnosed people) / (total number of diagnosed people) *100
There is an extra step and parameter in the function because the recommendation function calls for probability function. When we call the function the first parameter is for the the patient name, and the second is a parameter with a default none parameter and it is for making the function enter an if block and making it return the calculated value.
After all the steps function writes the probability value of having the certain type of disease patient has the information about.
If the which_patient returns None value, probability function writes that probability can't be calculated due to absence.

Recommendation for a Treatment

There are 2 values we need in order to recommend something to the patient. I get the treatment risk directly from the list by using which_patient and I get the patient's probability of having the disease by calling the probability function. To get the false positive value from probability function, we need to give 2 parameters to the function which are the patient name and 'return' values.

If probability of having the disease if greater than the treatment risk probability, function writes that system suggests patient to have the treatment and vice versa.
Just like the other functions, if the patient doesn't exist, it writes that recommendation cannot be calculated due to absence.

Saving to Output File

Instead of writing the saving to output file code everytime, I created save_output function. It opens doctors_aid_output.txt with 'a' parameter, this allows the function to add text everytime it gets called.

Calling Functions

After definxing all the functions, I need to call them one by one in the order of the input file. To do this, I get the inputs line by line from the list which organized_input returns. If the first element of the input is 'create', it calls the create function; if it is 'remove' it calls the remove function and same for the list.
For probability and recommendation there is extra person variable to get the name of the patient. Person variable gives the function patient's name and function uses that name.

Programmer's Catalogue

In this section there are the whole code I've written, the time I spent for analysis implementation and testing and how this code can be reused by the others.

Time Spent on This Assignment

Time Spent for Analysis	Analysis part for me includes understanding what is wanted and how to do it. I had to learn how to open another file in python and using read commands. Also I had to learn about false positive possibility. I can say I spent 3 hours for this part.
Time Spent for Design and Implementation	This part was the most time consuming part. After getting the text file, finding out how to convert it to a input for functions took my time. I created every function in a similar amount of time. I took long time understanding the logic but after I got what I was actually doing, it became easier and I got faster. This part took 8-12 hours.
Time Spent for Testing and Reporting	While implementing the code, I tested every step but after I finished everything I had some small issues that I had to solve, such as tab spaces needed in list function, or '%' symbols that I had to add to some parts. These took around 2 hours for me to solve. For the reporting part I wrote this report in 4-5 hours.

```
def read_the_input_file():
    with open('doctors_aid_inputs.txt','r') as inputfile:
        return inputfile.readlines()

def organized_input():
    organized_list= []
    for line in read_the_input_file():
        input_patient_list = line.split(', ')
        input_patient_list = input_patient_list[0].split()+ (input_patient_list[1:])
        organized_list.append(input_patient_list)
    return organized_list

def save_output(entry):
    with open('doctors_aid_output.txt','a') as outputfile:
        outputfile.write(entry)

patients_list = []

def create(input_line):
    global patients_list
    patients_list_name = [i[0] for i in patients_list]
    input_line[2] = format(float(input_line[2])*100, '.2f')
    input_line[6] = int(float(input_line[6])*100)

    while True:
        if input_line[1] in patients_list_name:
            save_output(f'Patient {input_line[1]} cannot be recorded due to duplication.\n')
            break
        else:
            patient = [i for i in input_line if i!='create']
            patients_list.append(patient)
            save_output(f'Patient {input_line[1]} is recorded.\n')
            break

def remove(input_line):
    patients_list_names = [i[0] for i in patients_list]

    while True:
        if input_line[1] in patients_list_names:
            for patient in patients_list:
                if input_line[1] in patient:
                    save_output(f'Patient {patient[0]} is removed.\n')
                    patients_list.remove(patient)
                    break
        else:
            save_output(f'Patient {input_line[1]} cannot be removed due to absence\n')
        break

def list():
    global patients_list
    titles = ['Patient Name', 'Diagnosis Accuracy', 'Disease Name', 'Disease Incidence' ,'Treatment Name', 'Treatment
Risk']
    titlesfirst = [i.split()[0] for i in titles]
    titlessecond = [i.split()[1] for i in titles]
    patients_list_2 = [i.copy() for i in patients_list]
```

```
    for i in patients_list_2:
        i[1] = i[1] + '%'
        i[5] = str(i[5]) + '%'

    save_output('{:<8}{:<16}{:<24}{:<16}{:<24}{}\n'.format(*titlesfirst))
    save_output('{:<8}{:<16}{:<24}{:<16}{:<24}{}\n'.format(*titlessecond))
    save_output('-'*110)
    save_output('\n')
    for i in patients_list_2:
        save_output('{:<8}{:<16}{:<24}{:<16}{:<24}{:<20}\n'.format(*i))

def which_patient(name):
    for i in patients_list:
        if i[0] == name:
            return patients_list.index(i)
        else:
            pass

def probability(the_patient,type=None):
    if which_patient(the_patient)!= None:
        patient_index = which_patient(the_patient)
        total_number_of_people = patients_list[patient_index][3].split('/')[1]
        correctly_diagnosed_people = int(patients_list[which_patient(the_patient)][3].split('/')[0])
        diagnose_accuracy = patients_list[which_patient(the_patient)][1]
        wrongly_diagnosed_people = (int(total_number_of_people) * (100-float(diagnose_accuracy)))/100
        total_number_of_diagnosed_people = correctly_diagnosed_people + wrongly_diagnosed_people
        false_positive_possibility = round((correctly_diagnosed_people / total_number_of_diagnosed_people *100),2)

        if false_positive_possibility-int(false_positive_possibility)== 0:
            false_positive_possibility = int(false_positive_possibility)

        if type == 'return':
            return false_positive_possibility
        save_output(f'Patient {the_patient} has a probability of {false_positive_possibility}% of having
{patients_list[patient_index][2].lower()}.\n')
    else:
        save_output(f'Probability for {the_patient} cannot be calculated due to absence.\n')

def recommendation(the_patient):
    if which_patient(the_patient)!= None:
        probability_of_having_the_disease = probability(the_patient,'return')
        treatment_risk_probability =patients_list[which_patient(the_patient)][-1]
        if probability_of_having_the_disease > treatment_risk_probability:
            save_output(f'System suggests {the_patient} to have the treatment.\n')
        elif probability_of_having_the_disease < treatment_risk_probability:
            save_output(f'System suggests {the_patient} NOT to have the treatment.\n')
    else:
        save_output(f'Recommendation for {the_patient} cannot be calculated due to absence.\n')

with open('doctors_aid_output.txt','w') as outputfile:
    outputfile.write('')

for whole_info in organized_input():
    if whole_info[0] == 'create':
        create(whole_info)
    elif whole_info[0] == 'remove':
```

```
        remove(whole_info)
    elif whole_info[0] == 'list':
        list()
    elif whole_info[0] == 'probability':
        person = whole_info[1]
        probability(person)
    elif whole_info[0] == 'recommendation':
        person = whole_info[1]
        recommendation(person)
    else:
        pass
```

How to use this code?

The code I've written works with an external text file, so if you want to use the code for the same purpose, it is enough to change the external text file the way you want the code to work.
Also the code consists of many functions, these functions can be used individually as well. Some adjustments can be made the way you want. For instance you can change the spaces in list function to have table with different look.

User Catalogue

As I mentioned in the how to use this code, this code can be reused in some ways. To use this program as a user, users need to make changes in the doctors_aid_inputs.txt
To be able to use the program properly, inputs should be in an order.

To use the create function, users need to write the inputs in the given order :

create Patient Name, Diagnosis Accuracy, Disease Name, Disease Incidence , Treatment Name, Treatment Risk

create Ateş, 0.99, Thyroid Cancer, 16/100000, Chemotherapy, 0.02

For remove, probability, recommendation functions, only name is needed. Example:
remove Hayriye
recommendation Su
probability Toprak

For list function no input is needed, only list is enough. Example:
list

Evaluation	Points	Evaluate Yourself / Guess Grading
Indented and Readable Codes	5...	5
Using Meaningful Naming	5...	5
Using Explanatory Comments	5...	5
Efficiency (avoiding unnecessary actions)	5...	5
Function Usage	25 ...	25
Correctness	35 ...	35
Report	20 ...	18
There are several negative evaluations	.....	