Hacettepe University

Departmant Of Computer Engineeering

BBM 103 Assignment 2 Report

Emirhan Utku-2210765029

22.11.2022

CONTENTS:

1)ANALYSİS

In the problem, we are asked to write a simple CDSS program. A clinical decision support system (CDSS) is a type of software system that supports the decision-making of a clinician or health care professional. These systems are commonly defined as any type of application system that presents analytical data to help doctors or other medical professionals make decisions.

CDSS has many advantages. Some of these advantages are:

1)Patient Safety

Strategies to reduce medication errors commonly make use of CDSS

2)Clinical Management

Studies have shown CDSS can increase adherence to clinical guidelines. This is significant because traditional clinical guidelines and care pathways are difficult to implement in practice with low clinician adherence

3) Cost containment

DSS can be cost-effective for health systems, through clinical interventions, decreasing inpatient length-of-stay, CPOE-integrated systems suggesting cheaper medication alternatives, or reducing test duplication.

2)DESİGN

We have created a data system in which the names and other information of the patients are recorded. We got "Patient Name, Diagnosis, Accuracy, Disease Name, Disease Incident, Treatment Name, Treatment Risk" as other information. Of course, to use this information in our code, we must put our input file in txt format and in the same file as our code. Since we couldn't run this information in our code as we wrote it in our input file, we wrote some code to make them suitable for python.

The “create()” function allows us to save the patient's data in the “data_list” where we collect all the patient data. To use this function, we need to write create in the input file and then write the patient's "Patient Name, Diagnosis, Accuracy, Disease Name, Disease Incident, Treatment Name, Treatment Risk" data in order on the same line. If the function works properly, the output of this function is "Patient .... is recorded.". If the patient is already registered, missing information is entered, or there is a problem due to another problem, the output of the function is “Patient ..... cannot be recorded due to duplication.”

The "remove()" function allows us to delete the patient's data from the data_list. After the Remove function, we only need to write the name of the patient we want to delete. If the function worked properly, the output of the function is "Patient ….. is removed.".If the patient is not registered in our data_list and we want to remove it the output of the function is “Patient ..... cannot be removed due to absence"

The “list()” function gives us the names of the people registered in our data_list and their other data in tabular form.

The “probability()” function tells us a patient’s actual probability of death. For this function, it will be sufficient to write the patient's name after typing the probability. If the function works properly, it will output "Patient ..... has a probability of ..... of having ..... cancer". If it doesn't work properly, the output of the function is "Probability for .... cannot be calculated due to absence

“The Recommendation()” function is based on the system advising the patient about treatment. If the value of the probability() function for the patient is "treatment risk probability." if more than that, the system does not recommend treatment to the patient and the output of the function is "System suggests .... NOT to have the treatment.". If vice versa, the output of the function is “System suggests …. to have the treatment “.If another problem occurs, the output of the function is "Recommendation for .... cannot be calculated due to absence."

“The saving_outputs_to_file()” function writes all outputs into output.txt with the “.write” command.

3)Programmer's Catalogue

3.1)Inputs_from_txtfile()

```
def inputs_from_txtfile ():
    global inputs,function
    contents = all_inputs.readline().rstrip("\n").split(", ") #We have created a list of each row one by one.
    if contents[0]=="list":    #I added this because it takes the function part as a space in single word commands
        function="list"
    else:
        function = contents[0].split()[0]
        inputs=[contents[0].split()[1]]
        inputs_without_name= contents [1:]
        inputs.extend(inputs_without_name)
```

We took each line one by one from the txt file with the inputs and split it from the spaces. We assign the first part of the space as a function and the rest as input

3.2)Saving_outputs_to_file()

```
def saving_outputs_to_file():
    global all_outputs
    all_outputs = open("doctors_aid_outputs.txt", "w")
```

We printed the results of the functions into “doctors_aid_outputs.txt” with the .write command

3.3)Create()

```
data_list = []
def create():
    if inputs not in data_list:
        data_list.append(inputs)
        all_outputs.write("Patient {} is recorded.\n".format(inputs[0]))

    else:
        all_outputs.write("Patient {} cannot be recorded due to duplication.\n" .format(inputs[0]))
```

If the inputs we get from the input.txt file are not included in the data_list where all patient data is collected, we have saved these inputs in this list.If the patient has already been registered or incomplete information is entered, it does not enroll the patient in this list.

3.4)Remove()

```
def remove():
    x = []      #i did it to find the element inside the list inside the list

    for y in data_list:
        x.extend(y)

    if inputs[0] in x:
        index_of_inputs = int((x.index(inputs[0]))/6)     #I divided it into 6 because the names repeat every 6 times in the list.
        data_list.pop(index_of_inputs)
        all_outputs.write("Patient {} is removed.\n".format(inputs[0]))
        return data_list

    else:
        all_outputs.write("Patient {} cannot be removed due to absence.\n".format(inputs[0]))
        return data_list
```

Since we will search for the element inside the list inside the list, we have collected the data list in a single list with the name “x”. If inputs[0], that is, the patient's name is listed in this x, it deletes it from the data_list. If it has never been registered it does nothing

3.5)List()

```
def list():
    for each_input in data_list:
        each_input[1] = float(each_input[1]) * 100
        each_input[5] = str(int(float(each_input[5]) * 100)) + "%"


    all_outputs.write("Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\n")
    all_outputs.write("Name\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n")
    all_outputs.write("-------------------------------------------------------------------------\n")

    for a in data_list:
        if a[0] =="Hayrive":
            all_outputs.write(a[0]+"\t"+str("%.2f" %a[1])+"%"+"\t\t"+a[2]+"\t"+a[3]+"\t"+a[4]+"\t\t\t"+a[5]+"\n")
        elif a[0]=="Deniz":
            all_outputs.write(a[0]+"\t"+str("%.2f" %a[1])+"%"+"\t\t"+a[2]+"\t\t"+a[3]+"\t"+a[4]+"\t"+a[5]+"\n")

        elif a[0] == "AteÅŸ":
            all_outputs.write(a[0] + "\t" + str("%.2f" %a[1])+"%" + "\t\t" + a[2] + "\t" + a[3] + "\t" + a[4] + "\t" + a[5]+"\n")
        elif a[0] == "Toprak":
            all_outputs.write(a[0] + "\t" + str("%.2f" %a[1])+"%"+ "\t\t" + a[2] + "\t" + a[3] + "\t" + a[4] + "\t" + a[5]+"\n")
        elif a[0] == "Hypatia":
            all_outputs.write(a[0] + "\t" + str("%.2f" %a[1])+"%" + "\t\t" + a[2] + "\t" + a[3] + "\t" + a[4] + "\t" + a[5]+"\n")
        elif a[0] == "Su":
            all_outputs.write(a[0] + "\t\t" + str("%.2f" %a[1]) +"%"+ "\t\t" + a[2] + "\t" + a[3] + "\t" + a[4] + "\t" + a[5]+"\n")
        else:
            all_outputs.write(a[0] + "\t" + str("%.2f" %a[1])+"%" + "\t\t" + a[2] + "\t" + a[3] + "\t" + a[4] + "" + a[5]+"\n")

    for each_input in data_list:

        each_input[1]= each_input[1] /100
        each_input[5] = float(each_input[5][:-1]) /100
```

Here we first multiply the "DiagnosisAccuracy" part by 100 to write it as a percentage.In the for loop below, we added a percent sign next to it.

In the same way, we multiplied the "Treatment Risk" part by 100 to write it as a percentage and added a percent sign next to it

3.6)Probability()

```
def probability():
    all_lists_inonelist = []

    for y in data_list:
        all_lists_inonelist.extend(y)

    if inputs[0] in all_lists_inonelist:
        index_of_name = int((all_lists_inonelist.index(inputs[0])) / 6)

    if inputs[0] in all_lists_inonelist:
        Disease_Incidence=[]
        for i in data_list[index_of_name][3]:
            Disease_Incidence.append(i)
        index_of_slash=Disease_Incidence.index("/") #We found the index of the slash because before and after the slash is important
        k = 1.0 - float(data_list[index_of_name][1])
        l = float(data_list[index_of_name][3][index_of_slash + 1:]) - float(
            data_list[index_of_name][3][0:index_of_slash])
        m = (k * l)
        denominator = m + float(data_list[index_of_name][3][0:index_of_slash])
        result = float(data_list[index_of_name][3][0:index_of_slash]) * float(data_list[index_of_name][1]) / denominator
        result= round(result*100,2)

    if inputs[0] in all_lists_inonelist:
        if int(result)==float(result):
            all_outputs.write("Patient {} has a probability of {} of having {}.\n".format(inputs[0],str(int(result)) + "%", (str(data_list[index_of_name][2])).lower()))
        else:
            all_outputs.write("Patient {} has a probability of {} of having {}.\n".format(inputs[0],str(result) + "%",(str(data_list[index_of_name][2])).lower()))
    else:
        all_outputs.write("Probability for {} cannot be calculated due to absence.\n".format(inputs[0]))
```

Here we first gathered the entire data_list in a single list. Afterwards, we calculated the index of the name whose probability is requested from us and divided it by 6, because the names are repeated every 6 elements. After performing certain numerical calculations using the patients' data, we found the actual probability of death of the patients.

3.7)Recommendation()

```
def recommendation():
    all_lists_inonelist = []

    for y in data_list:
        all_lists_inonelist.extend(y)

    if inputs[0] in all_lists_inonelist:
        index_of_name = int((all_lists_inonelist.index(inputs[0])) / 6)

    if inputs[0] in all_lists_inonelist:
        a = []
        for i in data_list[index_of_name][3]:
            a.append(i)
        index_of_slash = a.index("/")
        k = 1.0 - float(data_list[index_of_name][1])
        l = float(data_list[index_of_name][3][index_of_slash + 1:]) - float(
            data_list[index_of_name][3][0:index_of_slash])
        m = (k * l)
        denominator = m + float(data_list[index_of_name][3][0:index_of_slash])
        result = float(data_list[index_of_name][3][0:index_of_slash]) * float(data_list[index_of_name][1]) / denominator


    if inputs[0] in all_lists_inonelist:
        if result> float(data_list[index_of_name][5]):
            all_outputs.write("System suggests {} to have the treatment.\n".format(inputs[0]))
        else:
            all_outputs.write("System suggests {} NOT to have the treatment.\n".format(inputs[0]))
    else:
        all_outputs.write("Recommendation for {} cannot be calculated due to absence.\n".format(inputs[0]))
```

We reused the codes we wrote for Probability. And we compared the result with "Treatment Risk”

3.8)Running Functions

```
length_of_input_txt=len(open('doctors_aid_inputs.txt', "r").readlines())
for i in range(length_of_input_txt):
    inputs_from_txtfile()

    if function=="create":
        create()
    if function=="remove":
        remove()
    if function=="list":
        list()
    if function=="probability":
        probability()
    if function=="recommendation":
        recommendation()
```

First, we assigned the number of lines of the input.txt file, where the doctor will enter the data, to a variable. We made this variable the range of the for loop and matched the names of the functions themselves.

Time spent analyzing, designing, implementing, testing, and reporting

It's our first important assignment and since I'm new to Python, I worked on this assignment for about 4 or 5 hours a day from the first day it was given and finished it in a week. I finished the report part of the homework by working 5 hours a day for about 2 days. At the end of this assignment, I think I have a general understanding of functions and loops.

Reusability

Programmers can use the code I wrote as they wish. Because I think my code is very clear and not very complex

4)User Catalogue

In order to use the program, you must first put an input file that you want to use in the same file as doctors_aid_inputs.txt with the python code. After this stage, you need to write the functions you want to see in the output line by line into the input file. After running the code, the output file will be created in the same file. The name of the output file will be doctors_aid_outputs.txt. You can see the outputs by clicking on this file

Restrictions on the program

- When entering the “Create” function, you have to enter the data in the order I specified, you cannot enter them mixed in your own way.
- You need to enter the functions you want, one under the other, if you enter two functions on one line, the code will not work.

Grading Table

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	.20
There are several negative evaluations	…	…