Hacettepe University Department of Computer Engineering

BBM103 Assignment 2 Report

Aykut Alp GÜRLER – 2210356024

26.11.2022

Contents

Analysis

Today, healthcare organizations benefit from information systems in a wide variety of areas such as management services, diagnosis of diseases, support of physicians’ decisions about patients, guidance in nurses’ and physicians' work, signal interpretation, laboratory services and patient management. CDSS is at the forefront of the systems used for this purpose. CDSS is a computer program that provides support to clinicians and other health care providers in clinical decisions. From a point of view, these systems are computer systems that deal with clinical data or medical information to provide decision support.

CDSS systems eliminate the problems called over-diagnosis and overtreatment which can be as harmful as being cancer.

This assignment is about a CDSS system which is probability-based decision system called Doctor’s Aid for clinicians and other healthcare employee.

Doctor’s Aid’s purpose is providing tools for clinicians to improve healthcare.

Design

In Assignment 2, we have 3 main steps:

1.Reading input file,

2.Deciding the command,

3.Processing input file,

4.Writing output to a file.

1.Reading Input File

First, open doctors_aid_inputs.txt file for reading by using the open() function. Second, read text from the file using readline() method of the file object and use “with” operator to make it more efficiently.I used “utf-8” character encoding system to avoid some undefined character situations.

2.Deciding the Command

Below our code, with conditional statements(if, elif, else) structure, program decides which command is going to be processed.

3.Processing Input File

First of all, I have defined 5 different functions in this stage: create() ,remove() , probability(), recommendation(), list(). These functions are responsible from main duties. Now I will explain what these functions do.

create()

Simply, we create a patient’s profile with this function. We append patient profile to a list named “patient_list” to reach it at other stages and we make “patient_list” our database. “create()” function is able to handle a problem like creating same patient again falsely. Function gives a message “Patient ‘patient_name’ cannot be recorded due to duplication.”. For normal situations, function prints “Patient ‘patient_name’ is recorded.”

remove()

The function we will consider now is “remove()”. The functions is responsible from removing patient and its profile from “patient_list” database. Function makes our CDSS system more efficient and simple with its application. For example, the patient may have been discharged from the hospital and in this case it is necessary to remove her/him from the system, “remove()” function handles this situation. For normal situation, its output “Patient ‘patient_name’ is removed.” But if there is not patient profile and we want to remove that patient, its output is “Patient ‘patient_name’ cannot be removed due to absence.”.

probability()

Our next function is maybe the most important mission of Doctor's Aid system. The task of this function is vitally important. Function calculates patient's probability of having disease with given informations which are "diagnosis accuracy" and "disease incidence". We use an unusual but correct method to calculate probability. Its output is "Patient 'patient_name' has a probability of 'probability' of having 'disease name'." . If you want to calculate probability of patient who does not exist in our database, output is "Probability for 'patient name' cannot be calculated due to absence.". We use this calculated probability value at next function which is named as "recommendation()".

recommendation()

The function I am going to talk about it is "recommendation()". This function's output is important for avoiding overdiagnosis and overtreatment issues which can be as harmful as being cancer. This function looks "patient_list" database and compares patient's probability of having disease and treatment risk. If patient's probability of having disease is larger than treatment risk, function's output is "System suggests 'patient name' to have the treatment.". Else, output is "System suggests 'patient name' NOT to have the treatment.". If you try to use function for patient who does not exist, output is "Recommendation for 'patient name' cannot be calculated due to absence."

listt()

Given function displays the patients profile which are in "patient_list" in the table. The function provides visual way to view the database.

4.Writing Output to a File

We write all outputs we take from other functions to doctors_aid_outputs.txt file with function which is called "write_output()". This function makes it able to handle the situation of writing outputs to text file efficiently. We again use "utf-8" to avoid undefined character situations.

Programmer's Catalogue

I will explain the code in 9 parts.

1. **Appending New Line**

We append one new line to text file to be able processing last line of text file.

```
with open("doctors_aid_inputs.txt", "a", encoding="utf-8") as file:
    file.write("\n")
```

2. **read_inputtxt() Function**

We create a string with lines of "doctors_aid_inputs.txt" file and append lines to "inputs_lists"

```
def read_inputtxt(): #function to read doctors_aid_inputs.txt file
    char="a"
    inputs_lists=[]
    with open("doctors_aid_inputs.txt", "r", encoding = "utf-8") as f:
        while char!="":
            char=f.readline()
            inputs_lists.append(char)
        f.close()
        return inputs_lists
```

3. **write_output() Function**

We write outputs to "doctors_aid_outputs.txt file".

```
def write_output(input): #function to add outputs to doctors_aid_outputs.txt file
    with open("doctors_aid_outputs.txt", "a", encoding = "utf-8") as f:
        f.write(input)
        f.close()
```

4. **create() Function**

Allows us to create a database with the name of "patient_list"

```
def create(): #function to create patient profile
    patient_name= line.split(", ")[0].split()[1]
    duplicate_found = False
    if len(patient_list) > 0:
        for y in range (len(patient_list)):
            if patient_name in patient_list[y]:
                write_output("Patient "+patient_name+ " cannot be recorded due to
duplication.\n")
                duplicate_found = True
        if not duplicate_found:
            patient=[]
            patient.append(patient_name)
            for i in range (1,6):
```

```
                patient.append(line.split(", ")[i])
            patient_list.append(patient)
            write_output("Patient "+patient_name+ " is recorded.\n")
    else:
        patient=[]
        patient.append(patient_name)
        for i in range (1,6):
            patient.append(line.split(", ")[i])
        patient_list.append(patient)
        write_output("Patient "+patient_name+ " is recorded.\n")
```

5. **remove() Function**

Allows us to remove patient profile from database.

```
def remove(patient_name): #function to remove patient from patient list
    is_removed = False
    for i in range(len(patient_list)):
        if patient_list[i][0] == patient_name:
            is_removed = True
            patient_list.pop(i)
            write_output("Patient "+ patient_name + " is removed.\n")
            break
    if is_removed == False:
        write_output("Patient "+patient_name+ " cannot be removed due to
absence.\n")
```

6. **probability() Function**

Function calculates the probability of having disease.

```
def probability(): #function to calculate patient's probability of having
disease.
    patient_name= line.split(", ")[0].split()[1]
    absence_patient= False
    for i in range (len(patient_list)):
        if  patient_list[i][0] == patient_name:
            absence_patient= True
            diagnosis_accuracy=float(patient_list[i][1])
            disease_incidence=patient_list[i][3]
            upper=float(disease_incidence[0:2])
            lower=float(disease_incidence[3:])
            global patient_probability
            patient_probability=100 * upper/((lower-upper)*(1-
diagnosis_accuracy)+upper)
            patient_probability=round(patient_probability,2)
            write_output(f"Patient {patient_name} has a probability of
{patient_probability}% of having {patient_list[i][2]}\n")
    if absence_patient == False:
```

```
        write_output(f"Probability for {patient_name} cannot be calculated due to
absence.\n")
```

7. recommendation() Function

Function compares probability of having disease and treatment risk and gives a recommendation to patient about having the treatment or not.

```
def recommendation(): #function to recommend whether patient should have the
treatment or not
    patient_name= line.split(", ")[0].split()[1]
    absence_recommendation= False
    for i in range (len(patient_list)):
        if  patient_list[i][0] == patient_name:
            absence_recommendation= True
            treatment_risk = patient_list[i][5].replace("\n", "")
            if float(patient_probability) > float(treatment_risk)*100:
                write_output("System suggests "+ patient_name + " to have the
treatment."+ "\n")
            else:
                write_output("System suggests "+ patient_name + " NOT to have the
treatment."+ "\n")
    if absence_recommendation== False:
        write_output(f"Recommendation for {patient_name} cannot be calculated due
to absence.\n")
```

8. listt() Function

Function diplays patient profile as a table and allows us to see patient profiles in a visual way.

```
def listt(): #function to show patient profile as a list
    first_line = f'{"Patient": <8}{"Diagnosis": <16}{"Disease": <16}{"Disease":
<12}{"Treatment": <16}{"Treatment"}\n'
    second_line = f'{"Name": <8}{"Accuracy": <16}{"Name": <16}{"Incidence":
<12}{"Name": <16}{"Risk"}\n'
    third_line = "-----------------------------------------------------------------
----------\n"
    write_output(first_line)
    write_output(second_line)
    write_output(third_line)
    for x in range(len(patient_list)):
        number1 = str(float(patient_list[x][1]) * 100) + "%"
        number2 = (patient_list[x][5])
        number2 = number2[0:-1]
        number2 = float(number2) * 100
        c =
f'{patient_list[x][0]:<8}{number1:<16}{patient_list[x][2]:<16}{patient_list[x][3]
:<12}{patient_list[x][4]:<16}{number2}%\n'
        write_output(c)
```

9. Deciding the commands

These lines of codes, allow to deciding which command is going to be processed.

```
data= read_inputtxt()
patient_list=list(list())
for line in data:

    command1=line.split(", ")
    if len(command1)!=1:
        create()
    else:
        try:
            command=line.split()[0]
            if command =="remove":
                patient_name= line.split()[1]
                remove(patient_name)
            elif command =="list":
                listt()
            elif command =="probability":
                probability()
            else:
                recommendation()

        except:
            pass
```

User's Catalogue

1. Create a text file named "doctors_aid_inputs.txt".
2. Fill the file with commands "create, remove, probability, recommendation, list"
   2.1 Command "create" is for creating a patient profile. Example input text file line is "*create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40"*
   2.2 Command "remove" is for removing patient from database. Example input text file line is "*remove Ateş"*
   2.3 Command "probability" is for calculating the probability of having disease. Example input text file line is "*probability Deniz"*
   2.4 Command "recommendation" is for deciding patient should have the treatment or not. Example input text file line is "*recommendation Su"*
   2.5 Command "list" is for showing patient profile visually as a table. Example text file line is "*list"*
3. Look output file named "doctors_aid_outputs.txt" for tools you used in our CDSS Doctor's Aid.