HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

BBM 103 Assignment 2 Report

Erdinç Arıcı – 2210356035

22.11.2022

CONTENTS

ANALYSİS

Almost every year, millions of new cancer cases and millions of cancer deaths occur worldwide.

Early detection plays a critical role in preventing cancer, so it is possible to cure cancer, but experiments to diagnose cancer are not always 100% accurate. For this reason, some people diagnosed with cancer are identified as cancerous even though they do not have cancer. In addition to harming these people through treatment, they are labeled as cancerous, with psychological, financial, and time costs.

We know that these people are labeled as cancerous. So, how do we know whether a patient selected among these cancer patients really has cancer? We have been asked to help with this issue. Thanks to the necessary information about the patients and their diseases, we calculate the probability that someone labeled as a patient will actually have the disease, thus avoiding unnecessary treatments.

DESİGN

The beginning of everything is to get the necessary entries from the txt file for the codes to work.

In all inputs in the txt file, the first word is the name of function, other words are shown as information, if any. A *readingputfile()* function must be created to read such inputs and parse them into function name and information. This function also executes the information obtained outside of this process in the required function.

A separate function must be created for each of the obtained function names. These can be sorted as create, remove, list, probability, recommendation, and apart from these, there is a writingoutputfile that writes the outputs from these functions to a new txt file.

The function named *create()* adds the patient whose information is entered to a list named *ListOfPatients*. If patient is already registered in this list, it informs you and finishes the process.

The function named *remove()* removes the patient whose name is entered from the *ListOfPatients.* If the patient whose name is entered is not in the list, it informs you and finishes the process.

The function named *list()* returns all the arguments in the *ListOfPatients* into a table.

The function named *probability()* gets the information of the patient whose name is entered from *ListOfPatients* and calculates the probability that the patient is the actual patient by making the necessary calculations.

The function named *recommendation()* calculates whether the patient should have surgery based on the result of *probability()* of the patient whose name is entered. If the patient is not in *ListOfPatients*, it informs you and finishes the process.

The function named *writingoutputfile(x)* prints the feedback from other functions to a new txt file.

PROGRAMMER'S CATALOGUE

Accessing Inputs

The program starts with this code. With the help of the *open* command, the txt file is accessed and the first step is taken to receive the inputs.

If there is no file named "doctors_aid_inputs.txt" the code will stop without any output.

ListOfPatients

This list is where the patient information that will be given to us is stored.

def readinginputfile()

```
def readinginputfile():
    """Connects to the txt file and takes the inputs in order and runs them
in the required function."""
    global Inputs
    for i in range(len(open("doctors_aid_inputs.txt", "r").readlines())):
        try:
            Inputs = ReadTXT.readline().rstrip("\n").split(", ")
            if Inputs == ["list"]: Function = "list"
            else:
                Function = Inputs[0].split()[0]
                Inputs[0] = Inputs[0].split()[1]
            if Function == "create": create()
            if Function == "remove": remove()
            if Function == "list": list()
            if Function == "probability": probability()
            if Function == "recommendation": recommendation()
        except:
            pass
```

This function does most of the whole operation. First of all, I made an *Inputs* variable that will be created inside this function global so that we can use it outside the function. Then I opened a *for* loop in this function. I made the number of iterations of this loop as much as the number of lines in the input file given to us thanks to the *open* command. then I deleted the "\n" at the end of each line to get the information easier and used the ", " on each line to split that line.

I used the first space character in the input to parse the information in the received input. In this way, I could easily separate the name of the function and the information. But there was a line in the input file that didn't fit what I was doing: *list.* Since the only problem is in this line, if the input is "list", I got it without looping it with the others.

The last job I gave to this function was to pass the information to the correct function according to the received function name. I did this with the *if/else* command.

If the lines in the input file are not as desired, I set the function to stop without output.

def create()

```
def create():
    """Adds the patient whose information is entered to a list where all
information is stored. A patient cannot be registered more than once."""
    NewPatients = [Inputs[0], Inputs[1], Inputs[2], Inputs[3], Inputs[4],
Inputs[5]]
    if NewPatients not in ListOfPatients:
        ListOfPatients.append(NewPatients)
        return writingoutputfile(f"Patient {Inputs[0]} is recorded.\n")
    else: return writingoutputfile(f"Patient {Inputs[0]} cannot be recorded
due to duplication.\n")
```

The name entered with this code is searched in the *ListOfPatients* list with a for loop. If the information of the entered name is not in the list, the information is added. If the entered name is in the list, the function informs you and the process is completed.

def remove()

```
def remove():
    """Deletes the patient whose name is entered from the list. A patient
not on the list cannot be deleted."""
    for i in range(len(ListOfPatients)):
        if Inputs[0] in ListOfPatients[i]:
            ListOfPatients.pop(i)
            return writingoutputfile(f"Patient {Inputs[0]} is removed.\n")
    return writingoutputfile(f"Patient {Inputs[0]} cannot be removed due to
absence.\n")
```

The name entered with this code is searched in the *ListOfPatients* list with a for loop. If the information of the entered name is in the list, the information is deleted. If the entered name is not in the list, the function will informs you and the process is finished.

def list()

```
def list():
    """Converts all the information in the list where the information is saved into a table."""
    writingoutputfile("Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\n")
    writingoutputfile("Name\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n")
    writingoutputfile("-------------------------------------------------------------------------------\n")
    for Patient in ListOfPatients:
        if len(Patient[0]) < 4: writingoutputfile(Patient[0] + "\t\t")
        else: writingoutputfile(Patient[0] + "\t")
        if len(Patient[1]) < 6: writingoutputfile(str(float(Patient[1]) * 100) + "0%" + "\t\t")
        else: writingoutputfile(str(float(Patient[1]) * 100) + "%" + "\t\t")
        if len(Patient[2]) < 12: writingoutputfile(Patient[2] + "\t\t")
        else: writingoutputfile(Patient[2] + "\t")
        if len(Patient[3]) < 8: writingoutputfile(Patient[3] + "\t\t")
        else: writingoutputfile(Patient[3] + "\t")
        if len(Patient[4]) < 4: writingoutputfile(Patient[4] + "\t\t\t\t")
        elif len(Patient[4]) < 8: writingoutputfile(Patient[4] + "\t\t\t")
        elif len(Patient[4]) < 12: writingoutputfile(Patient[4] + "\t\t")
        elif len(Patient[4]) < 16: writingoutputfile(Patient[4] + "\t")
        else: writingoutputfile(Patient[4])
        if True: writingoutputfile(str(int(float(Patient[5]) * 100)) + "%" + "\n")
```

The job of this function is to put the information found in *ListOfPatiens* into a table. In this function, I first printed the headers of the table. Then I hyphenated the titles and information to avoid confusion. Then I set the number of *tab* characters to be inserted according to the character length of the information given to us.

In the information given to us, the numbers were in float style. These numbers should have been written in the list in percentile form. That's why I converted the numbers to strings and added *"%"* next to them.

def probability()

```
def probability():
    """Indicates the probability of the patient contracting the disease."""
    for i in range(len(ListOfPatients)):
        if Inputs[0] in ListOfPatients[i]:
            DiagnosisAccuracy = float(ListOfPatients[i][1])
            DiseaseIncidenceDividing = int(ListOfPatients[i][3].split("/")[0]) * DiagnosisAccuracy
            DiseaseIncidenceDivisor = int(ListOfPatients[i][3].split("/")[1])
            Process = DiseaseIncidenceDividing / ((DiseaseIncidenceDivisor - DiseaseIncidenceDividing) * (1 - DiagnosisAccuracy) + DiseaseIncidenceDividing)
            NameOfDisease = ListOfPatients[i][2].lower()
            Result = str(round(Process * 100, 2)) + "%"
            if ".0%" in Result: Result = Result[:-3] + "%"
            return writingoutputfile("Patient {} has a probability of {} of having {}.\n".format(Inputs[0], Result, NameOfDisease))
    return writingoutputfile(f"Probability for {Inputs[0]} cannot be calculated due to absence.\n")
```

This function calculates the probability that the named patient actually has the disease. First, it checks whether the patient whose name is entered is registered in the system with the help of the for loop. If it is not registered, it informs you and finishes the process. If it is registered, it first takes the information to be calculated from the patient information and performs the operation. It then converts the result to a string to display the result in percentage style and appends *"%"* next to it.

def recommendation()

```
def recommendation():
    """Indicates whether the patient should have surgery according to the probability of getting the disease."""
    for i in range(len(ListOfPatients)):
        if Inputs[0] in ListOfPatients[i]:
            DiagnosisAccuracy = float(ListOfPatients[i][1])
            DiseaseIncidenceDividing = int(ListOfPatients[i][3].split("/")[0]) * DiagnosisAccuracy
            DiseaseIncidenceDivisor = int(ListOfPatients[i][3].split("/")[1])
            Process = DiseaseIncidenceDividing / ((DiseaseIncidenceDivisor - DiseaseIncidenceDividing) * (1 - DiagnosisAccuracy) + DiseaseIncidenceDividing)
            TreatmentRisk = float(ListOfPatients[i][5])
            if TreatmentRisk <= Process: return writingoutputfile(f"System suggests {Inputs[0]} to have the treatment.\n")
            else: return writingoutputfile(f"System suggests {Inputs[0]} NOT to have the treatment.\n")
    return writingoutputfile(f"Recommendation for {Inputs[0]} cannot be calculated due to absence.\n")
```

Part of this function works the same way as the *probability()* function. By comparing the probability of the patient whose name is entered to actually have the disease and the treatment risk of the patient's disease, it is calculated whether the patient should be treated or not, and an output is given accordingly. If the patient whose name is entered is not in the list, it informs you and finishes the procedure.

def writingoutputfile(x)

```
def writingoutputfile(output):
    """Prints output from all processes to a new txt file."""
    WriteTXT.write(output)
```

This function is linked to other functions and is included in almost all of them. The purpose of this function is to print the feedbacks from each function to a file named "doctors_aid_outputs.txt".

CODE WRİTİNG PROCESS

At first, this assignment intimidated me because too many files were loaded into the system. The next few days were spent understanding these files. Later, when I started the code, it took four or five days. Actually, it could have ended sooner, but I also took the time to research some things since it was the first time I saw them. In addition, I encountered a lot of errors. I even once thought about my mistake for about an hour for not closing a parenthesis. The shortest part for me was writing the report. Because I knew what I had to do and I was thinking about it while writing my code.

REUSABİLİTY

If the necessary information is given in the necessary format, the probability of a person to actually have cancer can be calculated for any type of cancer.

USER CATALOGUE

Program's User Manual

You should write the operation you want to perform in the "doctors_aid_inputs.txt" file, which is in the same directory as the "Assignment2.py" file. Then, when you run the code, a file named "doctors_aid_outputs.txt" will be created in the same directory. If you have written the operations you want to do correctly, this file contains the information you want to reach.

Restrictions On The Program

What you write in the "doctors_aid_inputs.txt" file must be in a format that complies with the rule. These formats are: "create {patient name} {diagnosis accuracy} {disease name} {disease incidence} {treatment name} {treatment risk}", "remove {patient name}", "list", "probability {patient name}" , "recommendation {patient name}.

Sample Input

doctors_aid_inputs.txt

```
create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40
create Deniz, 0.9999, Lung Cancer, 40/100000, Radiotherapy, 0.50
create Ateş, 0.99, Thyroid Cancer, 16/100000, Chemotherapy, 0.02
probability Hayriye
recommendation Ateş
create Toprak, 0.98, Prostate Cancer, 21/100000, Hormonotherapy, 0.20
create Hypatia, 0.9975, Stomach Cancer, 15/100000, Immunotherapy, 0.04
recommendation Hypatia
create Pakiz, 0.9997, Colon Cancer, 14/100000, Targeted Therapy, 0.30
list
remove Ateş
probability Ateş
recommendation Su
create Su, 0.98, Breast Cancer, 50/100000, Chemotherapy, 0.20
recommendation Su
list
probability Deniz
```

EVALUATE YOURSELF / GUESS GRADİNG

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	…	0