Artificial Intelligence Engineering Department

Doctor’ s Aid Second Assignment Report

Course Name:
BBM103 – Introductıon to Programming Labortory

Report Prepared By:

Nurşah Satılmış
2210765010

November 24, 2022

1 INTRODUCTION

Clinical decision support systems (CDSS) are computer-based applications that analyze data inside EHRs to deliver prompts and reminders to health care practitioners in order to assist them in applying evidence-based clinical recommendations at the point of treatment. CDSS is a collection of tools designed to improve clinical decision-making. These tools include, among other things, automatic alerts and reminders to caregivers and patients, clinical recommendations, condition-specific order sets, focused patient data reports and summaries, documentation templates, diagnostic support, and contextually appropriate reference information. As a result, artificial intelligence can be employed in the field of medicine to make the tasks of physicians and nurses more easier. The goal of this project is to design a probability-based decision system for clinicians called Doctor's Aid.

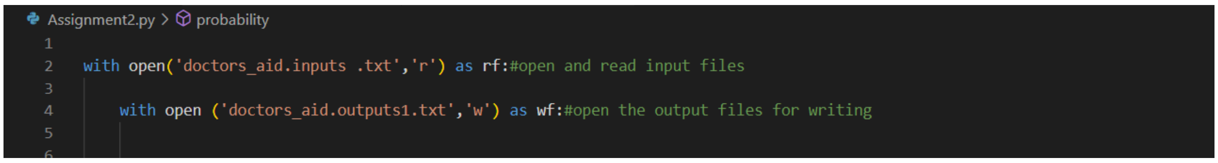
2 ANALYSİS

In 2020, around 19.3 million people were diagnosed with cancer, and 10 million people died as a result of cancer. Mammographic screening, for example, can help in breast cancer prevention, early detection, and therapy. However, some challenges remain during this process, such as overdiagnosis and overtreatment. The Doctor's Aid system was designed to overcome these challenges.

The Doctor's Aid system provides the doctor with the requested patient's name, accuracy of diagnosis, disease name, disease incidence, suggested treatment name, and treatment risk. The system also computes the patients' probability of having the assumed cancer type, and based on the patients' disease probability and the risk of the suggested treatment, it makes recommendations on whether or not the treatment should be taken by the patients.

3 Design

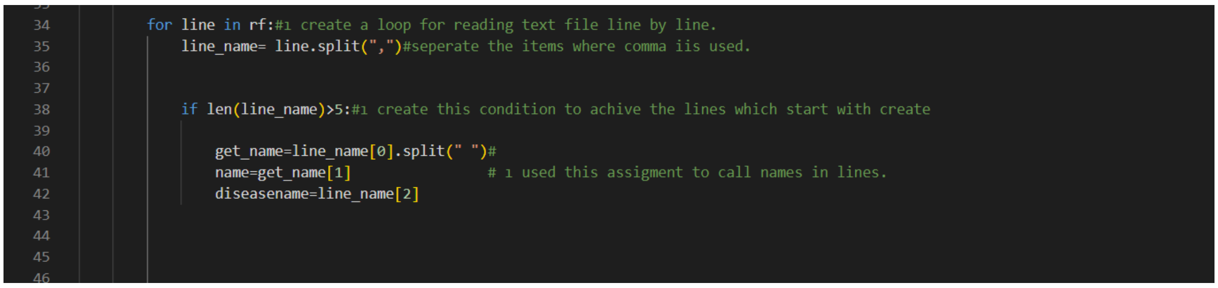
In the Doctor's Aid system, the data is supposed to be taken from and written to a .txt file.


Assignment2.py > probability

```

with open('doctors_aid.inputs .txt','r') as rf:#open and read input files

    with open ('doctors_aid.outputs1.txt','w') as wf:#open the output files for writing

```

In the Doctor's Aid system, the data is supposed to be taken from and written to a .txt file.

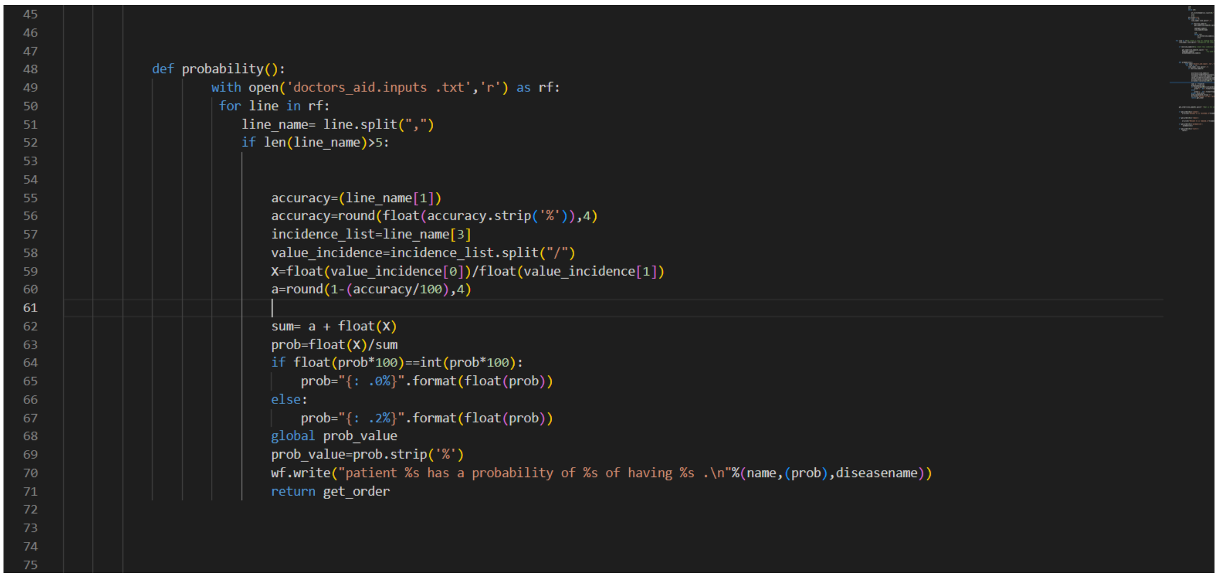
I open the output file for writing inside of reading function .İt make easier to writing outputs in output file..



```python
for line in rf:#ı create a loop for reading text file line by line.
    line_name= line.split(",")#seperate the items where comma iis used.


    if len(line_name)>5:#ı create this condition to achive the lines which start with create

        get_name=line_name[0].split(" ")#
        name=get_name[1]                # ı used this assigment to call names in lines.
        diseasename=line_name[2]
```

First for loop coded for reading the input file line by line. Line_name is the list of lines item.

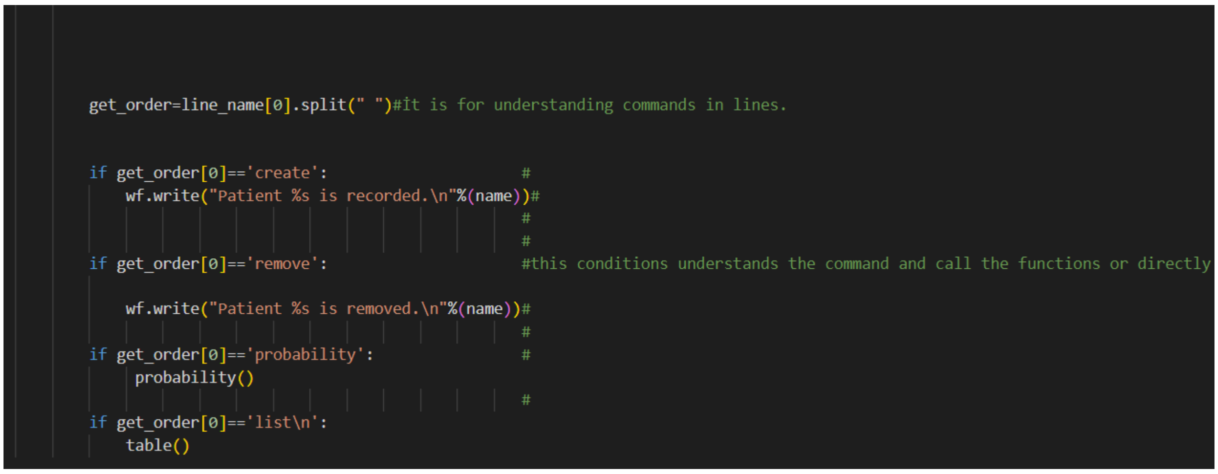
With first conditions, the lines which length is bigger than five filtered because these lines included patient information. With get_name list I access the patient name and disease name.



```python
def probability():
        with open('doctors_aid.inputs .txt','r') as rf:
         for line in rf:
            line_name= line.split(",")
            if len(line_name)>5:


                accuracy=(line_name[1])
                accuracy=round(float(accuracy.strip('%')),4)
                incidence_list=line_name[3]
                value_incidence=incidence_list.split("/")
                X=float(value_incidence[0])/float(value_incidence[1])
                a=round(1-(accuracy/100),4)

                sum= a + float(X)
                prob=float(X)/sum
                if float(prob*100)==int(prob*100):
                    prob="{: .0%}".format(float(prob))
                else:
                    prob="{: .2%}".format(float(prob))
                global prob_value
                prob_value=prob.strip('%')
                wf.write("patient %s has a probability of %s of having %s .\n"%(name,(prob),diseasename))
                return get_order
```

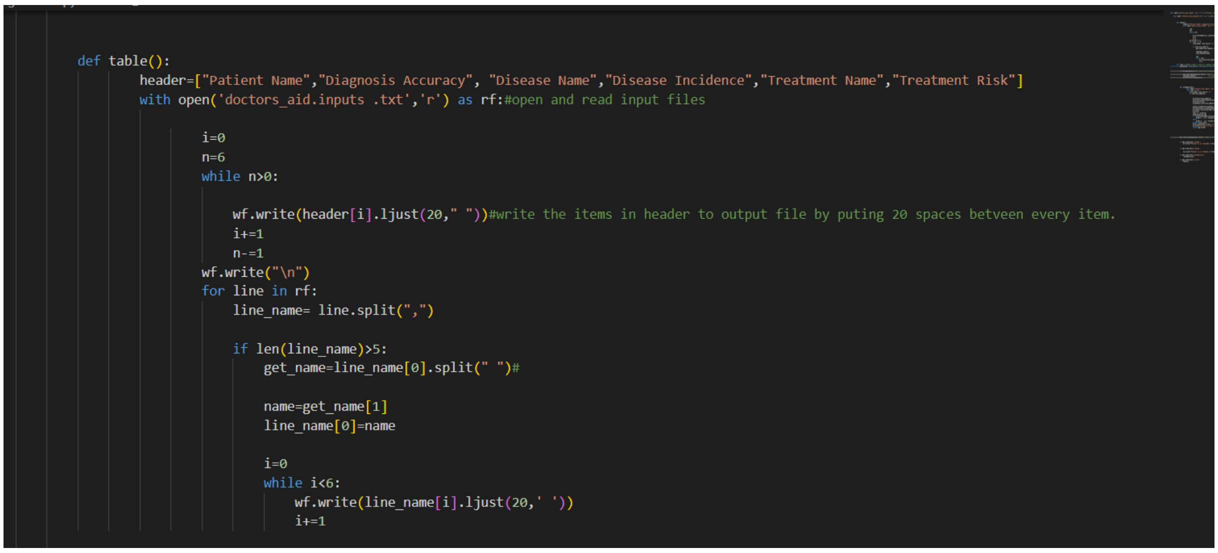
Here I called the same for loop again because I can’t achive the index of line_name.

With this function I try calculate the probability.The calculation done here is like so: Diagnosis accuracy is a percentage and I transform it into a float and subtract it from 1, disease incidence is a string and we transform it into float by dividing it, then we divide the disease incidence float by the sum of these to floats. This gives us the probability.



```
get_order=line_name[0].split(" ")#İt is for understanding commands in lines.


if get_order[0]=='create':                    #
    wf.write("Patient %s is recorded.\n"%(name))#
                                              #
                                              #
if get_order[0]=='remove':                    #this conditions understands the command and call the functions or directly

    wf.write("Patient %s is removed.\n"%(name))#
                                              #
if get_order[0]=='probability':               #
     probability()
                                              #
if get_order[0]=='list\n':
    table()
```

I splited the first index of line_name for get the command.And I used the get_order in conditions.In some conditions, I prefer writing outputs directly to he output files I think it is easier than creating function.



```
def table():
    header=["Patient Name","Diagnosis Accuracy", "Disease Name","Disease Incidence","Treatment Name","Treatment Risk"]
    with open('doctors_aid.inputs .txt','r') as rf:#open and read input files

        i=0
        n=6
        while n>0:

            wf.write(header[i].ljust(20," "))#write the items in header to output file by puting 20 spaces betveen every item.
            i+=1
            n-=1
        wf.write("\n")
        for line in rf:
            line_name= line.split(",")

            if len(line_name)>5:
                get_name=line_name[0].split(" ")#

                name=get_name[1]
                line_name[0]=name

                i=0
                while i<6:
                    wf.write(line_name[i].ljust(20,' '))
                    i+=1
```

With defining table function I try created a table which include every patient information.I create a header list. And, while loop wrote every item in the header list to output file with wf.write function. For created fixed length I used ljust function.

I used a same for loop again because when the program read “list\n” command it has passed the other lines so I solved this issue by this method.

With first conditions, the lines which length is bigger than five filtered because these lines included patient information. With get_name list I access the patient name and changed the first item in line_name list with name.

After all, with a while loop every item in line_name was written to output file.

Time Spent on This Assignment: During the analysis of this assignment, I read the pdf file and all the attachments thoroughly and this took approximately 1 hour.

Time spent on design and implementation: After understanding the problem, designing and implementing a solution was the part that took the most time. İt takes 5 day for me beause I couldn’t find the best approach for implementing, there was always error in some part of my code.

Time spent on testing and reporting: In this step, there were many errors I detected and fixing these with the whole testing part took around 2 day. The reporting part took about 3 hours.