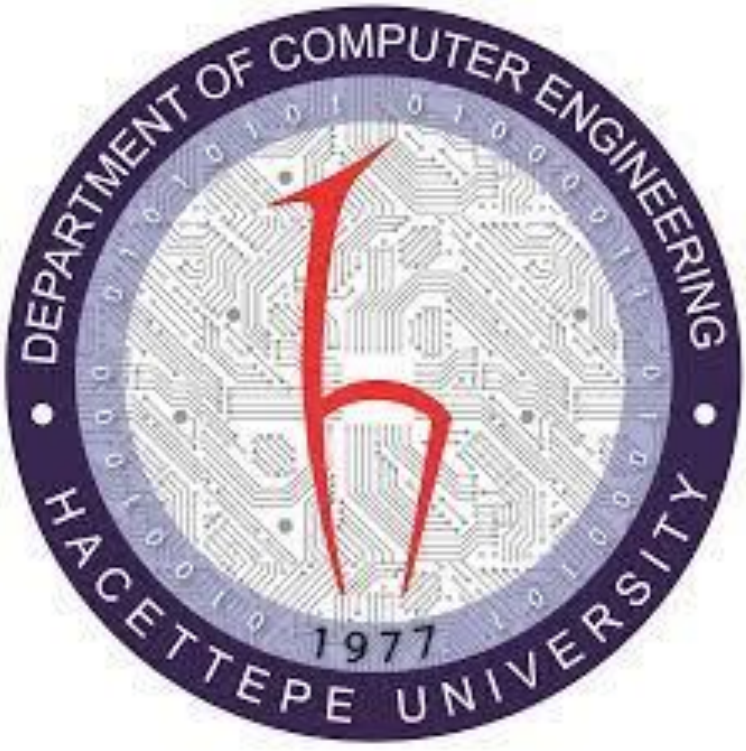
Hacettepe University Department of Computer Engineering

BBM 103 Assignment 2 Report

Ahmet Yavuz Mutlu -2210356014

21.11.2022



1. Analysis

Worldwide, cancer is very critical disease and it gets more dangerous day by the day but ıt is possible to successful treatment with early detection. However in the treatment process, doctors should be very sensitive. If they not, they can damage to the patients. So in this assignment, the thing that asked form us is to develop a program that helps to doctors with several ways. These are,

- *Inputs*

  We have to get user's inputs from the "doctors_aid_inputs.txt" and parse that input for understand what users want from us.

- *Create*

  When the user give an input like “create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40”, adding a new patient who name is Hayriye with his data’s in our system and giving a feedback to user.

- *Remove*

  If the input from user will be like "remove Hayriye", removing the patient Hayriye form the system and giving a feedback

- *List*

  Showing the all patients in our system in a table form when user write "list".

- *Probability*

  If the input from user will be like "probability Hayriye", calculating a Hayriye’s probability of being cancer according to the test results and showing that probability value.

- *Recommendation*

  Giving a recommendation that they should be take the treatment or not with using probability-based decision system to user for patient Hayriye, when user give an input like "recommendation Hayriye".

- *Outputs*

  At the end of the program we have to give feedback to user in a “doctors_aid_outputs.txt” file.

2. Design

❖ *Input Function:*

Firstly, I have to get the all inputs from "*doctors_aid_inputs.txt*". So I can use "*open*" and "*readlines*" function to get inputs line by line and I create a list to keep all this inputs for using this inputs later. Then, I parse this lines according to first space character to a function part and an data part and I keep this values in variables and the last part I call for the necessary function according to function part but I need to do this operation for the all lines, so I create a for loop. At the result, this input function get the inputs and understand it and do the necessary operations.

❖ *Create Function:*

When the function it calls, it gets the patient data and check this data, if there is not the same data in all patients list add this data's in all patients list and it calls the output function with parameter is *"Patient {patient name} is recorded."* but if is there the same data in all patients list, it calls the output function with parameter is *"Patient {patients name} cannot be recorded due to duplication."*

❖ *Remove Function:*

This function takes the patient's name and check it in all patient data's in all patients list with using a for loop and if it finds a patient that the same as patient's name then, it removes the whole data that this patient has and it calls the output function with parameter is *"Patient {Patients Name} is removed."* but if there is no match it calls the output function with parameter is "*Patient {Patients Name} cannot be removed due to absence."*

❖ *List Function:*

List function takes all patients list and writes it as a table form in "*doctors_all_outputs.txt*"

❖ *Patient Probability Function:*

Probability function gets patient name and search the name in all patients list and if it finds get some specific data's (disease accuracy, disease incidence and disease name) from this patient's dataset and calculate the patient's probability according to these specific data's and returns patient's probability and disease name but if is not find anything returns none.

❖ *Probability Function:*

This function calls patient's probability function with patient name and gets the patient's probability and patient's disease name. If the patient's probability different from none, it calls output function with parameter is *"Patient {patient name} has probability of {patient's probability}% of having {patients disease}"*. If is not it calls output function with parameter is *"Patient {patient name} probability cannot be calculated due to absence."*

❖ *Recommendation Function:*

Recommendation function calls patient's probability function with patient's name and gets treatment risk value from patient's data. If treatment risk value greater than patient's probability the function writes "*System suggests {patient name} NOT to have the treatment*" if treatment risk value less than patient's probability the function calls the output function with parameter is "*System suggests {patient name} to have the treatment*" if patient's probability value equal to none, it calls the output function with parameter is *"Recommendation for {patient name} cannot be calculated due to absence"*.

❖ *Output Function:*

When output function calls, it writes a text in doctors into the *"doctors_aid_outputs.txt"*.

3. Programmer's Catalogue

Analyzing: 1 hour Designing: 2 hours

İmplementing: 6-8 hours Testing: 2 hours Reporting: 3 hours

❖ ***input_function(input_file_name)**:*

parameters: inputs file name ("text_file.txt")

We can clearly say that the main function of this program is *input_function.* Because all of the inputs is made understandable in here. Let's look at how this this function works.

1. Firstly, function opens the input file with readable mode, reads all lines and creates a list and keep this inputs inside it.

```
def input_function(input_file_name):
    global patient_data, all_inputs
    all_inputs = open(input_file_name, "r")
    all_inputs = all_inputs.readlines()
```

2. After that, the input function parse a line from the first space character and the before part the first space character is function name and rest of is data's that we use in this function. But in some lines there is just function name, there is no first space character so our code gives an error. Because of that I used try/except to solve this problem.

```
try:
    first_space_chracter = line.index(" ")
    function = line[:first_space_chracter]
    patient_data = line[first_space_chracter+1:].rstrip("\n").split(", ")
except:
    function = line.rstrip("\n")
```

3. Then, our input function calls necessary function according to the function name.

```
if function_name == "create" : create()
elif function_name == "remove" : remove()
elif function_name == "list" : list()
elif function_name == "probability" : probability(patient_probability_function())
elif function_name == "recommendation" : recommendation(patient_probability_function())
```

4. But second operation and third operation above up, just for one line, so if we want to these operations for all lines we have to use a loop.

```
def input_function(input_file_name):
    global patient_data, all_inputs
    all_inputs = open(input_file_name, "r")
    all_inputs = all_inputs.readlines()
    for line in all_inputs:
        try:
            first_space_chracter = line.index(" ")
            function_name = line[:first_space_chracter]
            patient_data = line[first_space_chracter+1:].rstrip("\n").split(", ")
        except:
            function_name = line.rstrip("\n")
        if function_name == "create" : create()
        elif function_name == "remove" : remove()
        elif function_name == "list" : list()
        elif function_name == "probability" : probability(patient_probability_function())
        elif function_name == "recommendation" : recommendation(patient_probability_function())
```

5. Finally, our input function is completed.

❖ *create():*

parameters: no parameter

1. This function gets the patient data and check this data, if there is not the same data in patients list add this data's in patients list and write and output but if is not, the function just give feedback.

```
def create():
    if patient_data not in patient_list:
        patient_list.append(patient_data)
        output_function (f"Patient {patient_data[0]} is recorded. \n")
    else:
        output_function (f"Patient {patient_data[0]} cannot be recorded due to
                         duplication. \n")
```

❖ *remove():*

parameters: no parameter

1. In a for loop function checks the patient's name (patient_data[0]) in every patient's dataset and if it finds, remove the patient dataset from patient's list but and writes "Patient {patient name} is removed." if is not finds anything, it writes "Patient {patient name} cannot be removed due to absence."

```
def remove():
    global patient_list, patient_data
    for i in range(len(patient_list)):
        if patient_data[0] == patient_list[i][0]:
            patient_list.pop(i)
            output_function (f"Patient {patient_data[0]} is removed. \n")
            break
        elif i == (len(patient_list)-1):
            output_function (f"Patient {patient_data[0]} cannot be removed due
                             to absence. \n")
```

❖ *list():*

parameters: no parameters

1. List function takes all patients list and writes it as a table form. Table's has two main part, headers part and patient's data's part. Headers part is fixed so firstly I write this part

```
def list():
    global patient_list, patient_data
    output_function ("Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\t
       Treatment\n" +"Name\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n"+"-"*73+"\n")
```

```
Patient Diagnosis   Disease          Disease     Treatment        Treatment
Name    Accuracy    Name             Incidence   Name             Risk
---------------------------------------------------------------------------
```

2. After the headers part, I write patient's data's part with using nested two for loop. But between there is a special amount of character every two columns, so I defined a list for these (total_char). Firstly in the loop, the function calculates how many tab should be with using special character number. According to that, it writes the text into the output file.

```
total_char = [8,12,16,12,16,0]
for i in range(len(patient_list)):
    for j in range(6):
        # Amount of tab calculating in the bottom two lines.
        tab_num = (total_char[j] -len(patient_list[i][j]))//4 + 1
        if (total_char[j] -len(patient_list[i][j]))%4 == 0 : tab_num += -1
        if j == 1:
            # In the bottom three lines if the decimal part has one number after the
            roundation operation, we add "0", if it is not we don't add anything.
            if len(str(round(float(patient_list[i][j])*100,2))) <5 : zero ="0"
            else : zero =""
            text = f"{round(float(patient_list[i][j])*100,2)}{zero}%"
        elif j == 5: text = f"{round(float(patient_list[i][j])*100)}%"
        else: text = patient_list[i][j]
        output_function ((text+"\t"*tab_num))
    output_function ("\n")
```

```
Hayriye 99.90%      Breast Cancer   50/100000   Surgery          40%
Deniz   99.99%      Lung Cancer     40/100000   Radiotherapy     50%
Ateş    99.00%      Thyroid Cancer  16/100000   Chemotherapy     2%
Toprak  98.00%      Prostate Cancer 21/100000   Hormonotherapy   20%
Hypatia 99.75%      Stomach Cancer  15/100000   Immunotherapy    4%
Pakiz   99.97%      Colon Cancer    14/100000   Targeted Therapy30%
```

3. At the end of the process, our code and output like this,

```
def list():
    """This function, writes all patients datas
    as a table form in a output file"""
    global patient_list, patient_data
    output_function ("Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\n
       "+"Name\tAccuracy\tName\t\t\tIncidence\tName\t\t\tRisk\n"+"-"*73+"\n")
    total_char = [8,12,16,12,16,0]
    for i in range(len(patient_list)):
        for j in range(6):
            tab_num = (total_char[j] -len(patient_list[i][j]))//4 + 1
            if (total_char[j] -len(patient_list[i][j]))%4 == 0 : tab_num += -1
            if j == 1:
                # In the bottom three lines if the decimal part has one number after the
         roundation operation, we add "0", if it is not we don't add anything.
                if len(str(round(float(patient_list[i][j])*100,2))) <5 : zero ="0"
                else : zero =""
                text = f"{round(float(patient_list[i][j])*100,2)}{zero}%"
            elif j == 5: text = f"{round(float(patient_list[i][j])*100)}%"
            else: text = patient_list[i][j]
            output_function ((text+"\t"*tab_num))
        output_function ("\n")
```

```
Patient Diagnosis   Disease         Disease     Treatment       Treatment
Name    Accuracy    Name            Incidence   Name            Risk
-------------------------------------------------------------------------
Hayriye 99.90%      Breast Cancer   50/100000   Surgery         40%
Deniz   99.99%      Lung Cancer     40/100000   Radiotherapy    50%
Ateş    99.00%      Thyroid Cancer  16/100000   Chemotherapy    2%
Toprak  98.00%      Prostate Cancer 21/100000   Hormonotherapy  20%
Hypatia 99.75%      Stomach Cancer  15/100000   Immunotherapy   4%
Pakiz   99.97%      Colon Cancer    14/100000   Targeted Therapy30%
```

❖ *patient_probability_function():*

parameters : no parameter

This function calculates patient's probability of being cancer according to patient's data's and returns patient's cancer type and probability value.

1. Firstly function finds patient's data that user's want with using a loop.

```
def patient_probability_function():
    global patient_list, patient_data
    for i in range(len(patient_list)):
        if patient_data[0] in patient_list[i]:
```

2. When it finds, it takes some data's and calculates patient's probability of being cancer with using Bayer's Theorem.

```
diagnosis_accuracy = float(patient_list[i][1])
diev = patient_list[i][3].split("/") # diev : disease incedence elements values
diev[0], diev[1] = int(diev[0]), int(diev[1])
# With using Bayer's Theorem, function calculates patient's probability
probability_value =(diev[0]*diagnosis_accuracy) /
((diagnosis_accuracy*diev[0])+(diev[1]-diev[0])*(1-diagnosis_accuracy))
```

3. At the end, function returns that probability value and patient's cancer type.

```
def patient_probability_function():
    global patient_list, patient_data
    for i in range(len(patient_list)):
        if patient_data[0] in patient_list[i]:
            diagnosis_accuracy = float(patient_list[i][1])
            diev = patient_list[i][3].split("/") # diev : disease incidence elements values
            diev[0], diev[1] = int(diev[0]), int(diev[1])
            # With using Bayer's Theorem, function calculates patient's probability
            probability_value =(diev[0]*diagnosis_accuracy) /
              ((diagnosis_accuracy*diev[0])+(diev[1]-diev[0])*(1-diagnosis_accuracy))
            cancer_type = patient_list[i][2]
            return probability_value , cancer_type
```

❖ *probability(patient_data_values):*

Parameters :

patient_data_values = [patient_probability, cancer type]

1. Probability function takes 1 list parameters that includes patient's probability and cancer type. With using these, it writes some text into the output file. (Some information about the code is given inside the code)

```
def probability(patient_data_values):
    global patient_data
    if patient_data_values != None:
        # If the decimal part is zero, function does not shoes that part in text
        if round(patient_data_values[0]*10000)%100 == 0 :
              probability_persentage = round(patient_data_values[0]*100)
        else : probability_persentage = round(patient_data_values[0]*100,2)
       output_function(f"Patient {patient_data[0]} has a probability
                           of {probability_persentage}% of having
                           {patient_data_values[1].lower()}.\n")

    else: output_function(f"Probability for {patient_data[0]} cannot
                                  be calculated due to absence. \n")
```

2. If the parameters like [0.8000480076812466, Lung Cancer] the output will be like this,

```
Patient Deniz has a probability of 80% of having lung cancer.
```

Patient names (Deniz) came from global patient_data variable.

❖ *recommendation(patient_probability)*

parameters :

patient_probability = [patient_probability] (we just will use 0 index)

1. Recommendation function compares treatment risk and probability of being cancer that patient have. According to this result, it gives a suggestion.

```
def recommendation(patient_probability):
    global patient_data, patient_list
    for i in range(len(patient_list)):
        if patient_data[0] in patient_list[i]:
            patient_treatmet_risk = float(patient_list[i][5])
            if patient_treatmet_risk > patient_probability[0]:
                output_function (f"System suggests {patient_data[0]} NOT
                                 to have the treatment. \n")
                break
            else: output_function (f"System suggests {patient_data[0]} to
                                    have the treatment. \n")
                break
        elif i == (len(patient_list)-1):
           output_function (f"Recommendation for {patient_data[0]}
cannot be calculated due to absence \n")
```

❖ ***output_function(a_text):***

parameters : any texts

1. When this function calls, it writes the text into the output file *(“doctors_aid_outputs.txt”)*

```
def output_function (a_text):
    outputs_text_file.write(a_text)
```

4. Users Catalogue:

Create Command:

Input: create Hayriye, 0.999, Breast Cancer, 50/100000, Surgery, 0.40

If there is a patient that name is Hayriye:

Output: Patient Hayriye cannot be recorded due to duplication.

If there is not:

Output: Patient Hayriye is recorded.

Remove Command:

Input: remove Hayriye

If there is a patient that name is Hayriye:

Output: Patient Hayriye is remove.

If there is not:

Output: Patient Hayriye cannot be removed due to absence.

List Command:

***Input**:* list

Output:

Patient Name	Diagnosis Accuracy	Disease Name	Disease Incidence	Treatment Name	Treatment Risk
Hayriye	99.90%	Breast Cancer	50/100000	Surgery	40%
Deniz	99.99%	Lung Cancer	40/100000	Radiotherapy	50%
Ateş	99.00%	Thyroid Cancer	16/100000	Chemotherapy	2%
Toprak	98.00%	Prostate Cancer	21/100000	Hormonotherapy	20%
Hypatia	99.75%	Stomach Cancer	15/100000	Immunotherapy	4%
Pakiz	99.97%	Colon Cancer	14/100000	Targeted Therapy	30%

Probability Command:

Input: probability Hayriye

If there is a patient that name is Hayriye:

Output: Patient Hayriye has a probability of 33.32% of having breast cancer.

If there is not:

Output: Probability for Hayriye cannot be calculated due to absence.

Recommendation Command:

Input: recommendation Hayriye

If there is a patient that name is Hayriye:

Output: System suggests Hayriye to have the treatment.

or

System suggests Hayriye to NOT have the treatment

If there is not:

Output: Recommendation for Hayriye cannot be calculated due to absence.

Grading

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	…	…

Total = 100