Hacettepe University

Department Of Computer Engineering

BBM 103 Assignment 2 Report

Mert Can Köseoğlu – 2220356055

23.11.2022

CONTENTS

ANALYSIS

Cancer

Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. Cancer can star almost anywhere in the human body, which is made up of trillions of cells. There are more than 100 types of cancer. Types of cancer are usually named for the organs or tissues where the cancers form. For example, lung cancer starts in the lung, and throyid cancer starts in the throyid.

Cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths in 2020. The most common causes of cancer death in 2020 were: lung (1.80 million deaths), colon and rectum (916 000 deaths), liver (830 000 deaths), stomach (769 000 deaths) and breast (685 000 deaths).

Cancer mortality is reduced when cases are detected and treated early. When identified early, cancer is more likely to respond to treatment and can result in a greater probability of survival with less morbidity, as well as less expensive treatment.

CDSS

A clinical decision support system (CDSS) is intended to improve healthcare delivery by enhancing medical decisions with targeted clinical knowledge, patient information, and other health information. The main purpose of modern CDSS is to assist clinicians at the point of care. This means that clinicians interact with a CDSS to help to analyse and reach a diagnosis based on patient data for different diseases.

Problem

This assignment's problem is how to read and interpret data from the input file. Our purpose is making a list with patient's information, adding and deleting patients from list, evaluating the patient's probability of having the disease and suggesting to have the treatment or not.

Design and Programmer's Catalogue

Data Structures

Make a empty two lists named liste and patient_list. Liste includes commands like create, it is used for call the functions. patient_list does not have commands. It just inculudes arguments, it is used in functions.patient_list has patient informatin for every patient in the system thus it is a list of list. prob_list is used for recommendation function to compare probability of patient having disease and tratment risk.

```
liste = []           # main list with commands
patient_list = []    # current list with patient data
prob_list = []       #probability of patient list
```

Create Function

Make a create function to record patient. Make patient_list global to add the change in list from this function. Use name_found to check patient name is in the patient_list. (it means check index[r][0] in patient_list. r is used in for loop. One line in output file with create command is r as patient_list is list of list.)

```
def create(add_patient):
    global patient_list          # patient information list
    name_found = False           # for unique name
```

If patient_list is empty, add that line in the input file without create command to patient_list and write to output file which is doctors_aid_outputs.txt patient name (index[0] in add_patient which we get when we read input file below .)

```
if not patient_list:
    patient_list.append(add_patient)        # add patient information to
the list
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("Patient " + add_patient[0] + " is recorded\n")
```

. If patient_list is not empty check the patient name if it is not in the patient_list add it and write patient is recorded to the output file. If the name is in the patient_list write patient cannot be recorded due to duplication to output file.

```
else:
    for r in range(len(patient_list)):
        if patient_list[r][0] == add_patient[0]:        # check the patient
name
            with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
                output_file.write("Patient " + add_patient[0] + " cannot be
recorded due to duplication\n")
            name_found = True       # patient is recorded, end

    if not name_found:      # to record new patient
        patient_list.append(add_patient)
        with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
            output_file.write("Patient " + add_patient[0] + " is
recorded\n")
```

Remove Function

Make a remove function to delete patient to the system. We get remove_patient below when one line in input file command is remove then it includes one argument patient name it is remove_patient[0]. Found_name helps to check patient name.

```
def remove(remove_patient):
    global patient_list         # current patient list
    found_name = False          # check the name
```

If remove_patient[0] is equal to one name in patient_list, Remove patient from the patient list with its informations. (one index is deleted in the patient_list). Make found_name True to write patient is removed to output file.

```
for p in range(len(patient_list)):
    if patient_list[p][0] == remove_patient[0]:     # check the name
        found_name = True                           # go to "if found_name"
        patient_list.pop(p)                         # delete the patient
```

```
from the list
        break

if found_name:
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("Patient " + remove_patient[0] + " is removed\n")
```

If remove[0] cannot be found, found_name is stiil False and write patient cannot be removed due to absence to output file.

```
else:
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("Patient " + remove_patient[0] + " cannot be
removed due to absence\n")
```

Probability Function

Make probability function to calculate the probability of patient having disease. patient_list and prob_list is global now to use this lists inside and outside of the function. temp_list inculdes just one patient's name and probability. prob_list is list of list. (prob[0] is patient name which we get when we read input file below.)

```
def probability(prob):
    global patient_list
    global prob_list        # list of list name and probability.

    temp_list = []          # example: ['Hayriye','33.33']
```

If patient_list is empty, pass.

```
if not patient_list:
    pass
```

If patient_list is not empty. Check every index if one index has a name that is equal to prob[0]. temp_list append prob[0].

```
else:
    for z in range(len(patient_list)):
        if patient_list[z][0] == prob[0]:          # check  the name
            temp_list.append(prob[0])              # add name to temp_list
```

Evaluate probability with formula

disease incidence / ((1 + diagnosis accuracy) + disease incidence)

```
x = patient_list[z][4].split('/')        # separate index 4 by \  to get
float value
y = float(float(x[0]) / float(x[1]))     # disease incidence
f = float(patient_list[z][1])            # treatment risk

result = y / ((1 - f) + y)               # probability calculation
result = "%.2f" % (result * 100)         # two decimal numbers
```

add probabilty to temp_list then add temp_list to prob_list that list of list. write it to output file.

```
temp_list.append(result)                 # add probability value to
temp_list
prob_list.append(temp_list)              # make list of list

with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as output_file:
    output_file.write("Patient " + prob[0] + " has a probability of " +
str(result) + "% of having " +
                      patient_list[z][2] + " Cancer\n")
break
```

If prob[0] cannot be found in patient_list, write it to output file.

```
else:
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("Probability for " + prob[0] + " cannot be
calculated due to absence.\n")
```

List Function

List function is used to list patient information. It will help the clinician to check patient's data is correct or not.

```
def list_patient():
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:

output_file.write("Patient\tDiagnosis\tDisease\t\tDisease\t\tTreatment\tTre
atment\n")

output_file.write("Name\tAccuracy\tName\t\tIncidence\t\tName\t\tRisk\n")
        output_file.write("-----------------------------------------------
----------------\n")

        for patient in patient_list:
            output_file.write("\t".join(patient) + "\n")
```

Recommendation Function

Make recommendation function to suggest patient to have treatment or not. prob_list is used here to compare treatment risk and probability.

```
def recommendation(recommend):
    global patient_list
    global prob_list
    not_found = False
```

check patient_list and prob_list with for loop. If their patient name is equal to recommend[0] (that we get when we read input flie) compare probability (prob[b][-1]) and treatment risk in patient_list patient_list[a][-1].

```
for a in range(len(patient_list)):
    for b in range(len(prob_list)):
        not_found = False
        if patient_list[a][0] == prob_list[b][0]:
            if prob_list[b][0] == recommend[0]:     # check the patient
name
                x = float(patient_list[a][-1])      # x = treatment risk
```

If probability is bigger than treatment risk, system suggests to have the treatment.

```
if float(prob_list[b][-1]) > float("%.2f" % (x * 100)):     # for 40% > 33%
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("System suggests " + recommend[0] + " to have the
treatment.\n")
    break
```

If treatment risk is bigger, system suggests not to have treatment.

```
elif float(patient_list[a][-1]) < float(prob_list[b][1]):
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write("System suggests " + recommend[0] + " NOT to have
the treatment.\n")
    break
```

Other case it cannot be calculated.

```
else:
                not_found = True

if not_found:
    with open("doctors_aid_outputs.txt", 'a', encoding="utf-8") as
output_file:
        output_file.write(
            "Recommendation for " + recommend[0] + "  cannot be calculated
due to absence.\n")
```

Read input file

This is the most important part of the code and everything and every function is related here. The code starts here even it is below. First read line by line the input file that doctors_aid_inputs.txt then add every line to liste. Use space the index sprit and get rid of comma. line_list is just one line list. Liste is list of list. Liste helps to function input to check patient_list or _prob_list.

```
# read the input file line by line and append it to the list.
with open('doctors_aid_inputs.txt', encoding='utf-8') as input_file:
    for line in input_file:
        line_list = []
        line_split = line.split()       # separate by space
        for item in line_split:
            x = item.strip(',')         # get rid of comma
            line_list.append(x)
        liste.append(line_list)
```

Chek the first index in line list to choose which funtion to go.

```
for i in range(len(liste)):
    if liste[i][0] == 'create':         # command check
        liste[i].pop(0)
        create(liste[i])

    elif liste[i][0] == 'probability':
        liste[i].pop(0)                 # delete command in the list
        probability(liste[i])

    elif liste[i][0] == 'recommendation':
        liste[i].pop(0)
        recommendation(liste[i])        # send the list to function

    elif liste[i][0] == 'list':
        list_patient()                  # call function

    elif liste[i][0] == 'remove':
        liste[i].pop(0)
        remove(liste[i])
```

User Catalogue

Users have a input file which is doctors_aid_inputs.txt to write sysytem about patients data.

There are five commands to use

First is create command after write it you should write seven arguments (disease name is two words) patient name, diagnosis accuracy, disease name, disease incidence, treatment name, treatment risk.

Remove command to delete patient to system just one argument name.

List command does not have arguments it is used to list data.

Probability command has just one argument that patient name which patient you want to calculate its probability.

Recommendation command has just one argument that patient name which you want to suggest treatment or not. If you do not use probability fort hat patient you cannot get result.

Restriction

You must write two words with space for disease name and the others must be only one word.