EXPERT SYSTEM FOR THE DIAGNOSIS OF CARDIAC DISEASES

FINAL PROJECT

DSS: 4º IBM

Blanca Pueche, Natalia García, Jaime Aparicio and Andrea Martínez

Contenido

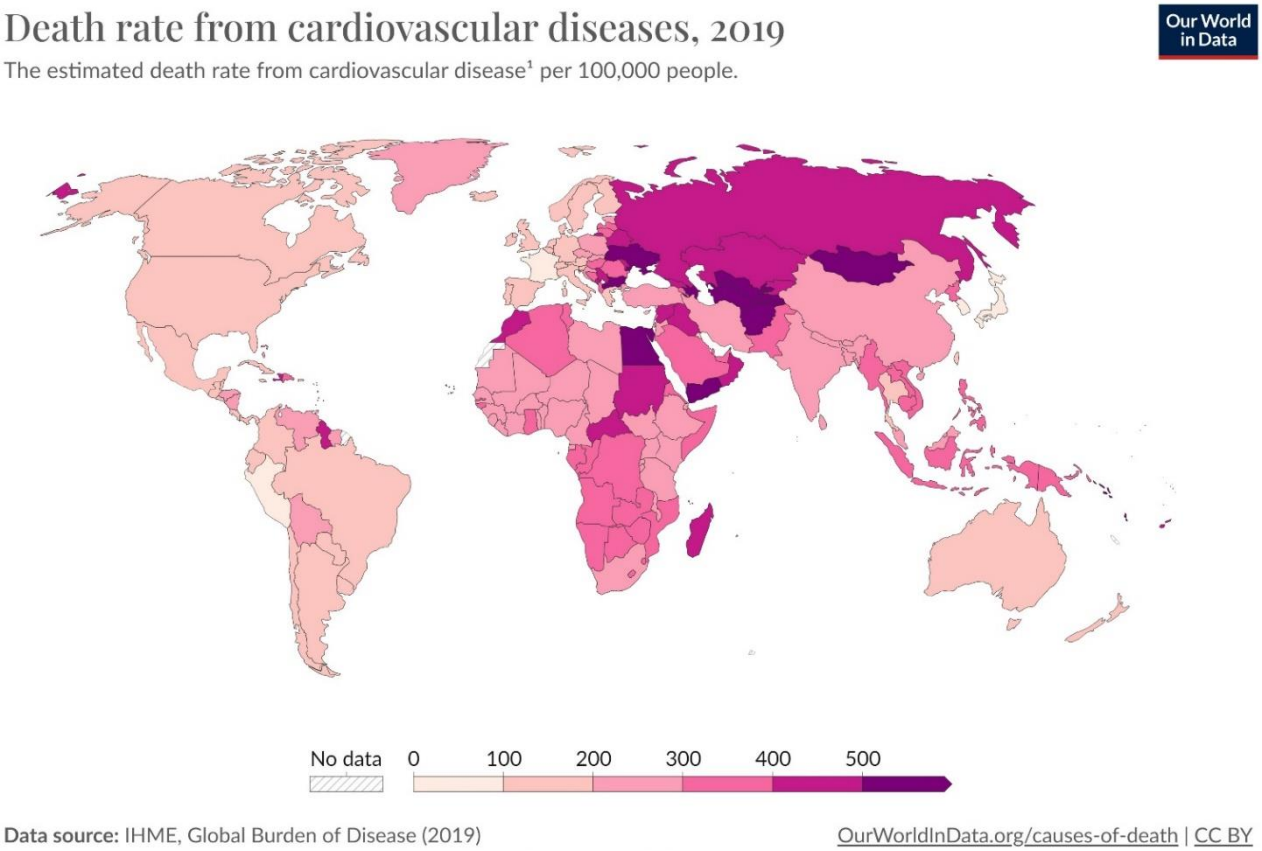
Introduction

Problem: Cardiac diseases

Cardiac diseases are a group of conditions that affect the heart and the blood vessels such as: coronary heart disease, cerebrovascular disease or rheumatic heart disease.[1] This type of diseases are, sometimes, hard to diagnose or to detect as they present a wide range of symptoms that can be overlooked by the patient but also can lead the doctor to misdiagnose the patient.

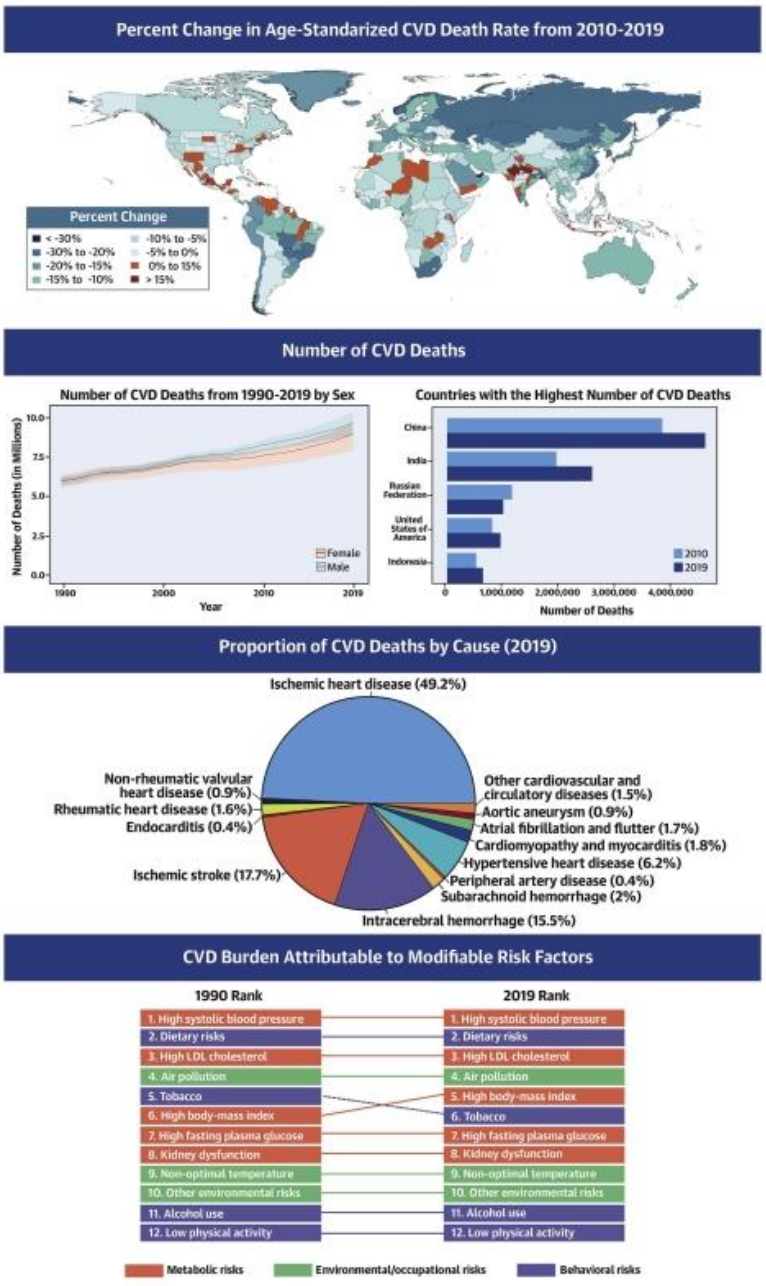
Cardiac diseases are, worldwide, the most common reason of death, it is estimated that these diseases are responsible for almost 18 million lives every year.[2] This number is expected to keep increasing as it is what it has been doing since 1990 when the number of deaths associated with cardiac diseases was 12.1 million.[3]

In order to prevent these deaths, it is crucial that the right diagnosis is made as soon as possible so counselling and medicine use can begin. This quick diagnosis is also essential to increase the chances of survival of the patients, prevent the disease from advancing and keep the aftereffects to a minimum.



[4]

CENTRAL ILLUSTRATION: Cardiovascular Disease Burden Across Time, Location, Cause, and Risk Factor



Roth, G.A. et al. J Am Coll Cardiol. 2020;76(25):2982-3021.

[5]

Why is a DSS needed?

A Decision Support System is useful technology when it comes to diagnosing a disease because it can help with the analysis and interpretation, risk assessment or early detection, as it recognizes abnormal patterns. Therefor a DSS would help the doctors make the diagnosis process shorter and more efficient, reducing the probability of the doctors making an unfounded decision that can lead to the wrong diagnosis.

A DSS can also be useful because it will have an unbiased point of view while helping the doctors make a personalized treatment plan. Furthermore, this type of systems are able to take up much more information than a human brain, this way the system can take into account a wider variety of cardiovascular diseases that due to their rarity may be discarded without conclusive proof.

In this case the DSS would be semi-structured because some aspects involve well-defined procedures or structured elements, like the results of an ECG (electrocardiogram) or of a blood analysis, as well as some unstructured elements, like the interpretation of symptoms, history of lifestyle or doctor interpretation [6].

There are already several DSSs existing for the diagnosis of heart diseases. Most of them focus on ischemic heart diseases, for example, there is the Artificial Neural Networks (ANN) that classifies myocardial infarction, the CART that classifies heart failure and the Support Vector Machine (SVM) that helps classify arrhythmia [7].

Our Expert System

Description

The Expert System we have developed in this project is aimed at providing doctors with an efficient, simple and useful tool for the diagnosis of cardiovascular diseases. This way the system will be able to reduce the misdiagnosis in this field but also make a diagnosis faster, which supposes a great benefit for the patient as it can be treated earlier, but also for the hospital as it can reduce costs.

The system is programmed to let the doctor introduce the information of the patient, such as name, last name and age, along with the symptoms he or she presents. Once the information requested by the program is fulfilled, the doctor can request the diagnosis of the patient to the system.

To establish the diagnosis, the system uses a table that associates 34 cardiac diseases with their respective symptoms. Therefore, the expert system will compare the symptoms of the patients with the symptoms of each disease to try to find a coincidence. The development of the table is based on the “Anatomical Chart Company. Atlas of Pathophysiology”[8].

The main benefit of using an expert system for the diagnosis of cardiovascular diseases is that the system has no cognitive limitations, therefore it can uptake information with no

limit. Another advantage is that, in contrast to doctors, the system does not get tired after hours of working and its effectiveness is kept the same.

Java Program

For our program we decided to create six classes and two enumerations. These files are divided in two folders, cardiac_diseases, which contains the main ones, and drools.config, which contains one class, DroolsBeanFactory.

- **Enumerations:** When designing our program, we chose to represent both diseases and symptoms with enumerations due to them being always the same ones, as our program specifies in cardiac diseases and doesn't let the user input new diseases or symptoms.
    - Disease: This enumeration contains 35 different types of cardiac diseases.
    - Symptom: This enumeration contains 157 different symptoms, all of which are present in one or more diseases of the Disease enumeration.
- **Classes:**
    - DroolsBeanFactory: This class oversees connecting with the drools part of the program.
      It has one attribute, kieServices (KieServices) and two functions, getKieSession and getDrlFromExcel.
      getKieSession: gets a Resource and returns a KieSession. The function creates all the necessary connections that lets us throw the rules from the drl file.
      getDrlFromExcel: gets a String and returns a String. The function gets the name of the excel file and returns the String containing the equivalent in a drl file. This proved to be useful when debugging the program.
    - Patient: The class patient represents the person that wants to be diagnosed. It has 5 different attributes: name (String), surname (String), age (int), disease (Disease) and symptoms (LinkedList<Symptom>). We decided that each patient has only one disease and one set of symptoms as one person is able to have many cardiac diseases at the same time, but other factors intervene in that, so to simplify the system we decided that in our program, one patient has only one disease and one set of symptoms.
      This class also has four constructors, the getters and setters, hashCode and the toString.
    - Hospital: This class represents only a list of patients, essentially, its mainly used to save patients correctly into the csv file and to save the patients while the program is running so the doctor can see all the patients, he/she is treating. It has two attributes: listOfPatients (LinkedList<Patient>) and name (String). It has two constructors, getters and setters, equals, hashCode and toString.
    - FileManager: This class only has one attribute, name (String). The whole functionality of this class is based on writing and reading csv files that contain the information of the patients. The class also has a constructor, and three classes: downloadCSV, symptomsToString and uploadCSV.

downloadCSV: gets a Hospital and returns a Boolean, true if the file is saved correctly and false if there's an error. The function creates a csv file, whose title will also be stored as the name of the hospital, in case it's not created, or updates the file that's been previously open. This file will contain a header and the data. It stores the hospital name and the patient's information.
symptomsToString: gets a LinkedList<Symptom> and returns a String that has the symptoms on the list separated by commas. It is used in the previous function to store the symptoms of the patients.
uploadCSV: this function doesn't receive any parameters and returns a Hospital. Its functionality is to read a csv file, whose title will be stored as the name of the hospital and generates a list of patients that will also be stored in the Hospital for the doctor to modify or see freely.
This class also has its correspondent getter and setter.

- Summary: This class oversees displaying and storing a summary, a couple of phrases, of each disease. The information is showed automatically if the patient has been diagnosed with a disease, so that the doctor can share the information with the patient to let them know exactly what they are being diagnosed with.
  It has one attribute, diseaseSummary (Map<Disease,String>), and one constructor.
  The class also has two classes, initialize and displaySummary, along with the correspondent getter.
  Initialize: is a function that creates the connection between each disease and their respective summaries. It doesn't receive anything and returns void.
  displaySummary: this function receives a disease, displays its summary and returns void.
- Main: This is the main class where the program runs. It has four attributes, hospital (Hospital), file (FileManager), sc (Scanner) and kSession (KieSession) along with the main function.
  The class also has the following functions:
  printMenu: only displays the main menu of the program. Doesn't receive anything and returns void.
  hospitalMenu: This displays the first menu that appears when the program starts. It gives the user the option to open a csv file that already exists or creating a new one. The function manages all the exceptions, receives nothing and returns void.
  createPatient: This function asks all the information required to create a new patient; it also calls other functions that let the doctor input the specific symptoms the patient may have. Doesn't receive anything and returns void.
  showAllSymptoms: This function displays all the sympwoms in the enum class along with a number that indicates the order they are in so, in other functions, the doctor can easily choose the symptoms number instead of writing it. Doesn't receive anything and returns void.
  selectSymptoms: This function is called in the createPatient and calls the showAllSymptoms. Here, the doctor will choose the symptoms the patient has. It doesn't receive anything and returns a LinkedList<Symptom>.

modifyPatient: This function lets the doctor choose and modify the patient's information by calling several different functions. It receives nothing and returns void.
checkYorN: This function only checks whether the doctor inputs y for yes or n for no when making changes in the program instead of writing another value that is not valid. It receives a String and returns a String.
modifyName: The function receives a Patient and changes their name if the doctor wants to do so. It's called from the changePatient function.
modifySurname: The function receives a Patient and changes their surname if the doctor wants to do so. It's called from the changePatient function.
modifyAge: The function receives a Patient and changes their age if the doctor wants to do so. It's called from the changePatient function.
modifySymptom: The function receives a Patient and changes their symptoms if the doctor wants to do so. It lets the doctor choose which symptoms are deleted and lets new symptoms be added to those the patient already had. It's called from the changePatient function. This function also has an option to let the doctor choose if they want to perform a new diagnosis based on the new set of symptoms without having to go back to the main menu.
makeNewDiagnosis: This function performs a diagnosis on a patient. It's specifically called from the modifySymptom function, receives a Patient and returns void.
makeDiagnosis: The functionality is the same as the previous one, only this one is called from the main menu and lets the doctor choose a Patient from all among the hospital first. It receives nothing and returns void.
removeSymptoms: This function is called when modifying a patient and lets the doctor choose wich symptoms are removes. Receives a Patient and returns LinkedList<Symptom> with the symptoms that haven't been removed.
addSymptoms: This function is called when modifying a patient and lets the doctor add new symptoms to those that the patient already has. It receives a Patient and returns a LinkedList<Symptoms> with the new symptoms added to the previous ones.
choosePatient: This function is called in several methods; it displays all the patients the hospital has and lets the doctor choose one of them. It doesn't receive anything and returns a Patient.
showPatientsInfo: This function displays all the information about the patients. It receives nothing and returns void.
setHospital: This function only sets the hospital to the one that's been passed. It's used in the tests.

In the program there is a folder called test where we can find the tests for all the use cases (see below): There are six different tests; insertPatient, modifyPatient, PatientTest, showInfo, uploadCSV and download CSV.

- InsertPatient: this test contains four different cases trying to create a new patient.
    1) Correct patient with name, surname, and age.
    2) Correct patient with name, surname, age, and symptoms.

3) Correct patient with name, surname, age, symptoms, and disease.
4) Correct patient whose disease we want to modify.

- ModifyPatient: this test makes different modifications on patients.
  1) Modify name.
  2) Modify surname.
  3) Modify symptoms.
  4) Modify disease.
  5) Doesn't let us modify because the hospital patients list is empty.
- PatientTest: this test uses fires rules.
  1) All symptoms match a disease.
  2) None of the symptoms match a disease.
  3) The patient doesn't have symptoms.
  4) All symptoms match except one.
  5) More symptoms than expected for a specific disease.
- ShowInfo: this test uses the showInfo function.
  1) No patients.
  2) One single patient.
  3) More than one patient.
  4) Null hospital.
  5) Null patient list.
- UploadCSV: this test uses the uploadCSV function.
  1) Valid CSV.
  2) Empty file.
  3) Empty hospital patient list.
  4) Patient with no disease.
  5) CSV file not found.
- DownloadCSV: this test uses the downloadCSV function.
  1) Valid hospital.
  2) Empty hospital (no patients).
  3) Null hospital.
  4) No name, but file saves correctly.
  5) Patient without disease.

Analysis of the user interface

For this system we have chosen to implement a user interface based on the console, as the goal of this project is to develop a simple and easy to use program. The user interface is divided into two submenus: the CSV menu and the general menu, both menus will show the options enumerated and to select one the number of the option must be typed, if a number that doesn´t match with any option is chosen, the program will ask for a new one. The CSV menu allows the user to upload an existing CSV file to use in the program, or to create a new one to store the new data.

After this the general menu will be shown. In this menu, there are four options: add a patient,

modify a patient, make a diagnosis, and show all the patients' information.

When a patient is added, the program asks for the name, last name and age of the patient, after these values are introduced an enumerated list of all the symptoms will be shown and the program will ask you to introduce the numbers of the symptoms that the patient has, if a number that isn´t in the list is chosen, that number will be eliminated and if the same number is chosen twice, it will add the symptom only once. The list of symptoms has to be introduced with spaces separating each number and the list will finish when enter is pressed. This design was chosen because if we show all the symptoms it´s harder for the user to overlook symptoms, it also makes the process of introducing the symptoms faster and we ensure that there are no typos that would make the program´s job harder.

The second option is to modify a patient. First, all the patients will be shown, then it will ask that a patient is chosen, in the case that the user choses a non-existing patient an exception will be launched, and afterwards it will ask the user, one by one, if he/she wants to modify any attribute, later it will ask for the new value of the attribute selected. In the case that there are no patients the program will say it and return to the menu.

Option number three is the most important one, make a diagnosis, and is where the expert system is implemented. First, a patient should be selected from the list of all the patients, then the program will compare the symptoms of that patient with the symptoms associated with the diseases and it will return a final diagnosis for the patient with a small description.

The last option is to show the information of all the patients saved in the program. The consol will show the name, surname, age, symptoms and, in case there is, the diagnosis of the disease associated with the symptoms.

The last option is to exit the program, this option is labeled with the number 7. This option downloads the data that has been created in the program in the CSV file selected at the beginning of the system.

Rules implemented in Drools

This expert system contemplates a total of 35 cardiac diseases and more than 100 symptoms. This is why, instead of implementing the rules one by one in an IF x THEN y format, we have opted for implementing the rules in an excel so the expert system will access this file to obtain the rules. Internally what the system does is convert the excel to a standard drl file containing all the rules and then work with that to get a diagnosis.

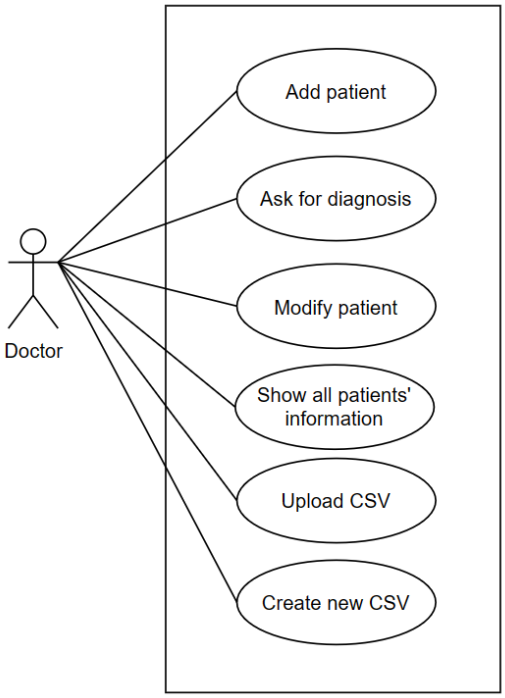
The rules use the symptoms introduced by the doctor as initial facts, and based on them, a different disease will be diagnosed. In order to set a disease as the diagnosis, all the symptoms associated with it must be present in the patient. If and only if all the symptoms match those from a specific disease, the doctor will be notified that the patient suffers from that disease, along with a small summary of it. If, on the other hand, there are no matches,

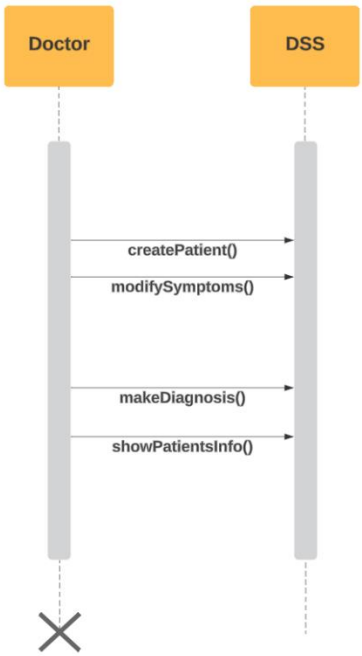
the doctor will be notified and advised to take the patient to another specialist for further testing.

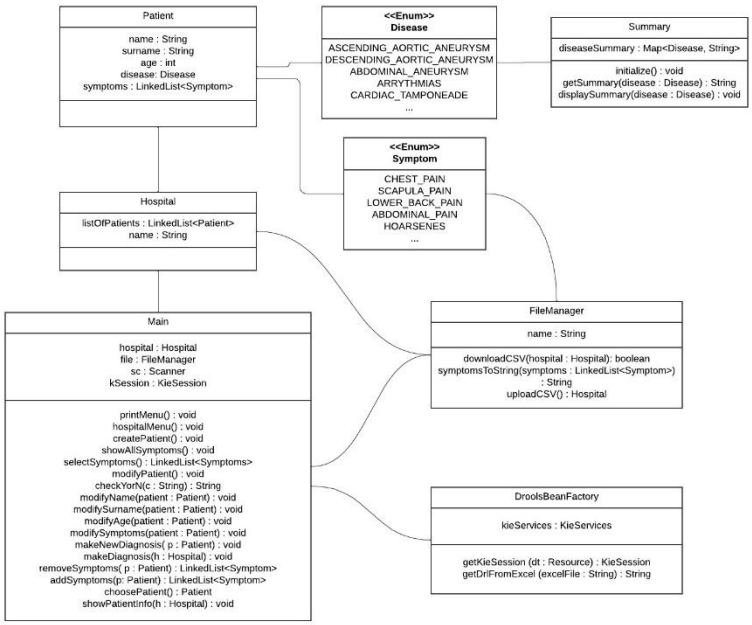
To view the drools excel click here.

The development of the rules is based on the “Anatomical Chart Company. Atlas of Pathophysiology” [6].

Diagrams







Github link:

To see the code and be able to clone the repository click here.
There's a folder called "documentation" where you can find everything related to the program.

Bibliography

[1] World Health Organization: WHO. (2021, 11 junio). *Cardiovascular diseases (CVDs)*. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[3] World Heart Report 2023: Confronting the World's Number One Killer. Geneva, Switzerland. World Heart Federation. 2023

[2] World Health Organization: WHO. (2019, 11 junio). *Cardiovascular diseases*. https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1

[4] Death rate from cardiovascular diseases. (s. f.-b). Our World In Data. https://ourworldindata.org/grapher/cardiovascular-disease-death-rates?country=~PRK

[5] J Am Coll Cardiol. 2020 Dec 22; 76(25): 2982–3021. Global Burden of Cardiovascular Diseases and Risk Factors, 1990–2019 - PMC (nih.gov)

[6] 2018 Nov 05. Maria Rashidi, Maryam Ghodrat, Bijan Samali and Masoud Mohammadi. Decision Support Systems. Decision Support Systems | IntechOpen

[7] Safdar, S., Zafar, S., Zafar, N., & Khan, N. F. (2018). Machine learning based decision support systems (DSS) for heart disease diagnosis: a review. Artificial Intelligence Review, 50(4), 597-623

[8] Stewart, J. (2017). *Anatomical chart company atlas of pathophysiology* (4a ed.). Lippincott Williams and Wilkins.