

**FOM - Hochschule für Oekonomie & Management**

**Hamburg**

**Master-Studiengang Big Data & Business Analytics**

**3. Semester**

**Development of a system to control and monitor blood pressure  
measurements to prevent cardiovascular disease**

Betreuer: Prof. Dr. Kai Brüssau

Autor: Jacqueline Franßen

Matrikel-Nr: 496804

3. Fachsemester

Hamburg, den 29.02.2020

# Contents

<b>1 Abstract</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
2.1 Problem statement . . . . .	3
2.2 Aim and scope of this work . . . . .	3
<b>3 Fundamentals</b>	<b>5</b>
3.1 Software Architecture: Best Practices . . . . .	5
3.1.1 Data mining in medical context . . . . .	5
3.1.2 Service-Oriented enterprise Architecture (SOA) for big data ap- plications in the cloud . . . . .	7
3.2 Medical Documentation Apps . . . . .	9
3.2.1 Overview: Smartphone apps to support self-management of hypertension . . . . .	9
3.2.2 Blood pressure monitoring in cardiovascular medicine and ther- apeutics . . . . .	10
3.2.3 Social web and use cases for medical apps . . . . .	11
3.3 Chatbots . . . . .	13
3.3.1 The potential of chatbots . . . . .	13
3.3.2 A deep learning question-answering specialized chatbot for med- ical students . . . . .	14
<b>4 Analysis and Development</b>	<b>16</b>
4.1 Experimental set-up . . . . .	16
4.2 Software architecture . . . . .	17
4.3 Setup of Angular Frontend . . . . .	17
4.4 Development of Watson Assistant Dialog . . . . .	18

4.4.1	Intent model . . . . .	18
4.4.2	Watson Assistant implementation . . . . .	18
4.5	Connect Watson Assistant and Angular frontend . . . . .	19
4.5.1	Data visualization: Development of a Python Script to show all measured values . . . . .	20
4.5.2	Development of email service to send reports every two weeks to the doctor . . . . .	20
4.6	Predictive Analytics: Creating a model to predict cardiovascular disease	20
4.6.1	Dataset . . . . .	20
4.6.2	Development of python script . . . . .	21
4.7	Results . . . . .	22
4.7.1	Challenges . . . . .	23
<b>5</b>	<b>Conclusion and Outlook</b>	<b>32</b>
5.1	Conclusion . . . . .	32
5.2	Outlook . . . . .	32
5.2.1	Connect Flask app, python script and Angular frontend . . . . .	33
<b>6</b>	<b>Abbreviations</b>	<b>35</b>
<b>Bibliography</b>		<b>37</b>
<b>7 Appendix A</b>		<b>39</b>
<b>8 Appendix B</b>		<b>40</b>

# List of Figures

3.1	Example software architecture cf.[Talukder et al. 2018] pp.260 . . . . .	6
3.2	Enterprise Software Architecture Reference Cube (ESARC) as an example for big data architecture cf.[Zimmermann et al. 2013] pp.133 . . . . .	7
3.3	ESARC business and information reference architecture cf.[Zimmermann et al. 2013] pp.134 . . . . .	8
3.4	Virtual patient software architecture cf.[Zini et al. 2019] p.3 . . . . .	15
4.1	Architecture diagram of developed system . . . . .	24
4.2	Angular Frontend (top page), chat UI and diagrams of measured values	25
4.3	Angular Frontend (bottom page), map to show the nearest doctors . . . . .	25
4.4	Dialog diagram: definition of hypertension . . . . .	26
4.5	Dialog diagram: causes of hypertension . . . . .	27
4.6	Dialog diagram: blood pressure measurement . . . . .	28
4.7	Dialog diagram: measurement tutorial . . . . .	29
4.8	Watson Assistant dialog . . . . .	30
4.9	Watson Assistant intents . . . . .	30
4.10	Watson Assistant entities . . . . .	31
5.1	Watson Assistant entities . . . . .	34

# List of Tables

# 1 Abstract

This scientific article focusses on the development of a chatbot and the analysis of blood pressure measurements. The purpose of this work can be divided into four top business cases. The first business case is that doctors can get an overview of the patient's blood pressure values. This makes them react more precisely to any special values or trends. Second, the measurements are taken more accurately since the patient is lead by a chatbot through a tutorial. Furthermore, the chatbot controls and analyses the measured blood pressure values. This improves the process of documentation. Third, doctors are informed in-time when there are outliers because a service sends a report regularly via email. This leads to the scenario that the doctor can interpret the values faster. Fourth, the patients get a recommendation of the nearest doctor which helps them to keep their appointments. Last, a neural network was developed to calculate the probability of suffering from cardiovascular disease of a patient. This information can be useful for the doctor as well as the patient to prevent the break out of the disease.

## 2 Introduction

### 2.1 Problem statement

According to organization Deutsche Hochdruckliga<sup>1</sup>, hypertension is one of the most common disease in our today's society. It can lead to cardiovascular disease and has many different causes, such as false nutrition, overweight or little movement. Another problem are the measurements, taken by patients at their home. Often, hypertension patients do not measure their blood pressure regularly or forget to write down their measured values. What is more, the treating doctors often get a large list of measured values (pulse, systolic and diastolic values) which they have to interpret in short-term. Often, the diagnosis or treatment and medication of these doctors depends on the average value or minimum and maximum value which was measured. All of these use cases can be automated and implemented by an application to prevent cardiovascular disease. The next section will explain the aims of this work and how they were implemented.

### 2.2 Aim and scope of this work

The first aim of this scientific work is to develop a solution to document blood pressure in order to react preventively against heart disease. To recommend an appropriate doctor in one's surrounding (approximately 5 kilometers of distance), an intuitive user interface with map is being shown. The application shall send every week/every two weeks a report (including a diagramm of all measured blood pressure values of the patient) to the doctor so that the doctor will be informed in real-time. In the diagrams/frontend, it is possible to select different scales, e.g. like the values of last

---

<sup>1</sup>cf.[Bluthochdruck vermeiden, behandeln und senken - Aktiv gegen Bluthochdruck 2019]

week/last month/last year. At the beginning of using the Chatbot, the user is being led through a tutorial which shows him how to measure correctly his blood pressure. One instruction is for example not to drink coffee before measuring your blood pressure or to sit for at least 5 minutes. After that, a routine for the measurement leads the patient through the measurement. The measured values are saved automatically into a Javascript Object Notation (JSON) file. Due to time limitations, there were no user tests driven. But to build up diagrams and to build up a neural network that calculates the probability of the cardiovascular disease, a sample dataset from Kaggle was used. The whole analytics process and implementation will be explained in section 4.6. What is important, the developed model is only a reference model and can be improved by changing the parameters and using real test data. Moreover, for diagnosis of any disease (in this case cardiovascular disease) depends on many factors, e.g. the appearance of the patient. Therefore, in case of a cardiovascular disease, the advice and treatment of a real doctor should be taken.

# 3 Fundamentals

## 3.1 Software Architecture: Best Practices

In this section, the 'best practices' of software architecture are described. To give an example, the architecture of a blockchain Peer to Peer (P2P) network which is backed with a distributed ledger system (see figure 3.1) will be explained. According to the authors, Talukder et al.<sup>1</sup>, the model is an appropriate solution for health applications because they support multiple stakeholders.

### 3.1.1 Data mining in medical context

As a common problem in every big data project there are multiple data sources and systems which provide relevant information for the particular use case. Generally, three types of data sources can be distinguished: For instance, the information can be handwritten human readable and human understandable medical notes. Besides, some information are computer readable and human understandable and the third 'generation' of information describes computer readable and understandable algorithms.

In order to provide an effective treatment of any disease, all health related data of a person on a spatial and temporal basis from birth is needed. For instance, all illness episodes, lab tests, pathological test results (which are outside the normal range), genomic data (to evaluate the genomic state of the individual), environmental and health events, lifestyle related data captured by Internet of Things (IOT), therapeutic data and outcome analysis results. This data is examined by a panel of experts to reach a consensus Proof of Disease (POC).

---

<sup>1</sup>cf.[Talukder et al. 2018] pp.258

According to Talukder et al.<sup>2</sup>, there are three different types of data mining:

- medical episode mining (MEM)
- health state mining (HSM)
- payment (financial/coin) mining

As can be seen in figure 3.1, medical systems need many resources from which all relevant medical data are loaded. As described by Talukder et al.<sup>3</sup>, all medical data is processed by Natural Language Processing (NLP) techniques, evidence based medicine as well as big data analytics (see figure 3.1). In most health applications, patients' participation increases when they have access to their health and lab records. In the solution provided by Talukder et al. genomic tests and non-communicable disease (NCD) data are stored in the blockchain as a transaction. Moreover, the blockchain technology is deployed in the cloud, as described in figure 3.1.

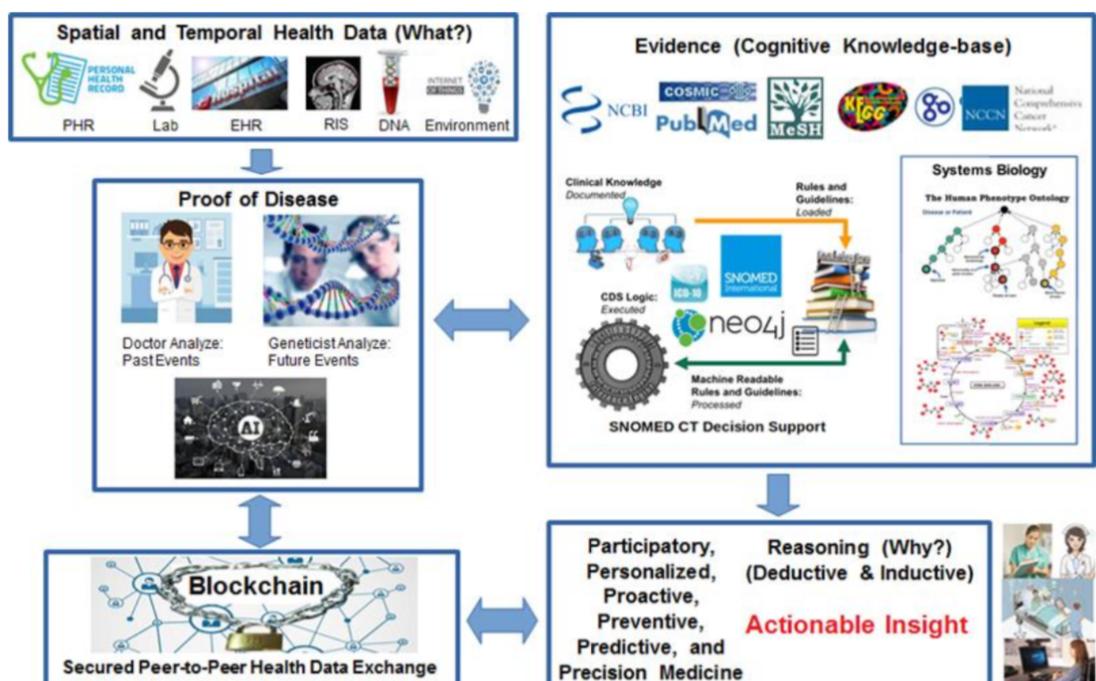


Figure 3.1: Example software architecture cf.[Talukder et al. 2018] pp.260

What is more, there is a medical miner which validates every transaction, then translates all clinical notes into structured International Classification of Disease (ICD) or Systemized Nomenclature of MEDicine Clinical Terms (SNOWMED CT) codes. After

<sup>2</sup>cf.[Talukder et al. 2018] pp.260

<sup>3</sup>cf.[Talukder et al. 2018] pp.259

that, all codes are stored into a smart contract. During that process, a medical expert validates whether the current onset matches any clinical pathway. Finally, medical experts discuss in a proper medical consensus if the data is useful for an accurate diagnosis and public health.

### 3.1.2 SOA for big data applications in the cloud

The state-of-the art architecture for any project is SOA and has many advantages, such as flexibility, agility, process orientation, time-to-market and innovation<sup>4</sup>. What is more, SOA is convenient for cloud computing since it is ready for extended service models. Figure 3.2 shows the architecture 'ESARC', developed by Zimmermann et al.<sup>5</sup>. It helps to cluster, classify, examine, compare, evaluate quality and optimize enterprise architectures. As depicted by figure 3.2, there is a link between enterprise and business information and design for supporting strategic initiatives. What is more, ESARC enables integration capacities for **IT!** (**IT!**) management, software engineering, service and operations management as well as process improvement initiatives.

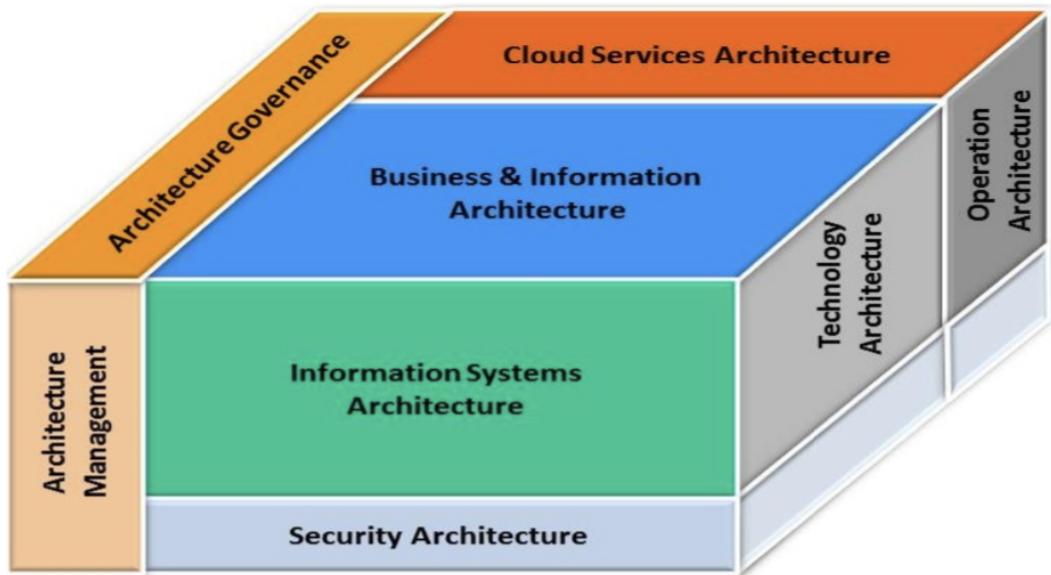


Figure 3.2: ESARC as an example for big data architecture cf.[Zimmermann et al. 2013] pp.133

As can be seen in figure 3.2, metamodels are used to define model elements in architectures. They relate architectural elements to ontologies which represent a common

<sup>4</sup>cf.[Zimmermann et al. 2013] pp.130

<sup>5</sup>cf.[Zimmermann et al. 2013] p.132

vocabulary for enterprise architectures. Zimmermann et al. recommend that operations of tasks and entity services should not have any knowledge about their process or interactive usage context<sup>6</sup>. Instead, task service operations should be independent from users and sessions and should only implement business functionality.



Figure 3.3: ESARC business and information reference architecture cf.[Zimmermann et al. 2013]  
pp.134

Figure 3.4 shows a more detailed view of 'ESARC', the procedural framework for architecture assessment processes and questionnaire design. On top of the graphic, with orange background, there are business vision, drivers, goals and objectives. To be more precise, architecture governance has the goal to manage activities such as plan, define, enable, measure, control and sets rules for architecture compliance to internal and external standards.

**Actors in cloud computing** According to Zimmermann et al., the main actors in cloud computing are cloud consumers, providers, auditors and broker<sup>7</sup>. In general, all SOA services are cloud services and follow a reference architecture: 'Jericho-Security-focused Service-oriented Reference architecture for cloud computing'. Thereby, management perspectives from Information Technology Infrastructure Library (ITIL)

<sup>6</sup>cf.[Zimmermann et al. 2013] pp.133

<sup>7</sup>cf.[Zimmermann et al. 2013] pp.133

and The Open Group Architecture Framework (TOGAF) standards are integrated.

## 3.2 Medical Documentation Apps

### 3.2.1 Overview: Smartphone apps to support self-management of hypertension

There exist many different self-management applications for patients who suffer from hypertension. Generally, these self-management programs are likely to be effective if they track the behaviour of their users<sup>8</sup>. This means that medical self-management applications should be supported by theory-based interventions, such as identifying the target behaviour and strategies of behavioural changes needed to achieve desirable health outcomes.

**Key functionalities** As depicted by Alessa et al., important functionalities of medical self-management applications are stress management, communication with Health Care Provider (HCP), self-monitoring abilities (e.g. portrayed in graphs or tables), reminders, automatic feedback and educational information.

**Behavioural Change Techniques (BCT)** Alessa et al. describe important functionalities during the development of medical self-management applications: BCT which form a theoretical domain framework. These recommendations include:

- behaviour regulation
- knowledge
- goals
- memory attention and decision process
- beliefs about consequences

In their study, Alessa et al. studied that many applications supporting self-management of hypertension have similar functionalities<sup>9</sup>. Besides these findings, Alessa et al. state that privacy and security is an important issue in many health applications and

---

<sup>8</sup>cf.[Alessa et al. 2019] p.1

<sup>9</sup>cf.[Alessa et al. 2019] p.2

that these are not available in 35% of all applications. Moreover, it should be ensured that users are able to make fully informed decisions by equipping the applications with skills and information necessary to scrutinize the privacy and security policies. This is due to the lack of knowledge and experience of many users in privacy concerns which can be seen in the social media<sup>10</sup>.

### **3.2.2 Blood pressure monitoring in cardiovascular medicine and therapeutics**

As mentioned above, many medical applications are developed for self-monitoring<sup>11</sup>. These bring many advantages and disadvantages. On the one hand, home blood pressure measurements are representative of natural environment and can show the response to antihypertensive medication. Furthermore, it is an easy and cost-effective way for obtaining a large number of readings. On the other hand, the measurement monitors might be too inaccurate and only a few devices have been subjected proper validation and failed tests. White et al. mention three different monitors for home measurements: arm, wrist and finger monitors. Moreover, multiple readings, e.g. two or three per day are recommended<sup>12</sup>

**Influence factors of hypertension** There are multiple factors which increase blood pressure, such as age, gender, environmental factors, smoking, alcohol, medication, caffeine, stress and talking. To be more precisely, e.g. women have lower blood pressure than men or age increases the blood pressure<sup>13</sup>. Environmental factors describe the influence of winter or summer terms on the blood pressure values. In winter, it is possible that blood pressure values increase up to 5 mmHg. Besides, the time of date can also influence measurements. For instance, it is recommended that patients take readings in the early morning and night. And there are differences between multiple systolic measurements whereas diastolic measurements stay nearly the same. But the most important fact, stated by White et al. is that summer and

---

<sup>10</sup> cf.[Alessa et al. 2019] p.3

<sup>11</sup> cf. p.4ff.[White 2007]

<sup>12</sup> cf. p.23ff.[White 2007]

<sup>13</sup> cf.[White 2007] pp.9

exercise decrease the blood pressure measurements<sup>14</sup>.

**Future trends in blood pressure measurements** As explained by White et al., it is very useful to have all readings available in an electronic form and to use these together with telemonitoring<sup>15</sup>. In detail, the readings could be transferred automatically to the health care provider. This can help to facilitate the communication between physician and patient in an easy way so that they could form virtual hypertension clinic.

### 3.2.3 Social web and use cases for medical apps

As stated by Lupton et al.<sup>16</sup> the current technical 'era' we are living in is the web 2.0 or social web. Social web includes sharing health and medical information with each other, e.g. patients and caregivers write about experiences and the individual health status. Often, the aim of these social webs is to control the health status by using online information and imaging. Conforming to Lupton et al.<sup>17</sup>, in healthcare projects, big data can be used to generate knowledge about healthcare, health behaviours and disease patterns. Such applications can assist in calculating diagnosis, identifying risks, facilitating health, fitness self-tracking as well as patient self-care regimes. As reported by a study which surveyed American doctors<sup>18</sup>, medication interaction apps are the first most-used and diagnosis apps the second most used category of apps. Moreover, pregnancy apps offer greater opportunities, such as that women can engage obsessive self-surveillance because of producing detailed data, such as heart rates, in real-time<sup>19</sup>. Pursuant to Lupton et al., the future potential of medical application lies in systems which enable lay people to access medical information (such as the electronic medical record) that was previously only available to healthcare practitioners or students.

In another article, Lupton et al. reported that the potential lies in automation of news or notifications which can be personalized or targeted so that doctors could contact

---

<sup>14</sup>cf.[White 2007] pp.9

<sup>15</sup>cf.[White 2007] pp.31

<sup>16</sup>cf.[Lupton 2014] p.607

<sup>17</sup>cf.[Lupton 2014] p.608

<sup>18</sup>cf.[Lupton 2014] p.610

<sup>19</sup>cf.[Lupton 2014] pp.614

patients directly to remind them to adhere to their treatment programs<sup>20</sup>. A further example of medical applications are 'smart pillboxes' for patients suffering from diabetes<sup>21</sup>. 'Smart pillboxes' are wireless devices that remind patients to take their medication and alert a doctor if the patient had failed to conform to their medication regimen. Continuing, medical health (M-HEALTH) technologies have a feedback, also called cybernetic mechanism in that they react with their users as opposed to passively provide information. To give an example, modern prosthesis or technological extensions of the body are a kind of cybernetic mechanism<sup>22</sup>. A big part of today's medical applications are surveillance systems in order to record and monitor cases of illnesses, such as obesity or infections<sup>23</sup>. These records might be useful to early detect epidemiological changes in the disease pattern. To give an example, 'individual medical encounters' which are conducted online enable doctors to flexibly practice personalized surveillance over each of their patients. At this point, another term occurs: 'surveillance knowledge' which refers to the digital data produced in the surveillance and can be useful for the individuate users.

**Blockchain solution for accurate medical decisions** As stated by Talukder et al. a significant amount of today's diagnosis in NCD is erroneous or unwanted<sup>24</sup>. The term NCD implicates disease caused by an unhealthy lifestyle, the proper environment or genomic causes over a long time and come up with confusing signs and symptoms. Talukder et al. mention 'P6'-Medicine which describes medicine using six adjectives starting with the letter 'p': medicine needs to be participatory, personalized, proactive, preventive, predictive, precision medicine. As a requirement list for health data, Talukder et al. describe important features as follows:

- secured (the anonymity, privacy, confidentiality of health data must be approved)
- systems which provide health data must have a zero down-time
- the integrity of the health data must be ensured
- the systems must be ubiquitous which implies an unlimited availability

---

<sup>20</sup>cf.[Lupton 2012] pp.229

<sup>21</sup>cf.[Lupton 2012] pp.232

<sup>22</sup>cf.[Lupton 2012] pp.234

<sup>23</sup>cf.[Lupton 2012] pp.235

<sup>24</sup>cf.[Talukder et al. 2018] p.263

- machine understandable (all health data should be conform to international standards and should be able to be distributed over multiple systems)
- health systems should be resistant against fraudulent hacking

## 3.3 Chatbots

### 3.3.1 The potential of chatbots

The main advantage of chatbots is to provide a 24-hour customer service with personalized interaction and no waiting time<sup>25</sup>. Besides, the data retrieval from conversations with chatbots provides many possibilities. To be more precise, modelling, profiling, analyzing and understanding users becomes increasingly important in many different industries and count as key to success in todays data driven world. Akhtar et al. analyzed chat conversations between customers and the chatbot of a telecommunication company in order to find out the user's topics of interest and how to satisfy users. As described by Akhtar et al., the tests of the chatbot were splitted into different activities, such as text mining techniques (feedback comments), event sequence analysis, frequent term extraction, analysis of bigrams/trigrams. During data preprocessing, Akthar et al. used the following methods:

1. corpus generation
2. eliminating extra white space
3. stopwords removal
4. tokenizing
5. stemming
6. creating term-document matrices

The main challenges during the data analysis process are data availability, the access to further user information (e.g. contract details or age in order to generate an user model) and the distinction between different user types and different personality structures.

---

<sup>25</sup>cf.[Akhtar, Neidhardt, and Werthner 2019] pp.297

**Question Answering Paradigms** There are several types of conversations which can be designed by building a chatbot<sup>26</sup>. Generally, there can be distinguished between two different paradigms: information-retrieval based Question and Answering and knowledge-based Question and Answering. The first type describes the mechanism to define short texts as answers to a user's intent. On the opposite, the second type describes how to in natural language. The answers are stored into a full-related database and the conversation works simply with a rule-based method.

**Types of dialog systems** In general, dialog systems can be divided into two kinds of systems. On the one hand, there are task-oriented systems which are appropriate for short conversations and built for a certain purpose. On the other hand, there are non-task-oriented systems which are built for longer and more complex interactions with the purpose of imitating human conversations<sup>27</sup>.

### 3.3.2 A deep learning question-answering specialized chatbot for medical students

During their studies, medical students have to take an exam which is called Objective Structured Clinical Examination (OSCE) where they interact with a 'standardized' patient played by an actor who simulates the symptoms and intents of the patient<sup>28</sup>. The aim of this exam is to test and assess the students' abilities and social interaction and diagnosis skills. Since in practice, there are not many actors who can play a patient's role, Zini et al. developed a virtual patient and chatbot system which works with NLP techniques.

Figure 3.4 shows the architecture of the developed system to create a virtual patient. Zini et al. used a Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) network in order to learn domain specific word embeddings, sentence embedding and answer selection models. The embeddings model which is outlined by a red rectangle in figure 3.4 is trained on a corpus of medical documents. In Figure 3.4, there is a NLP engine outlined by a red rectangle. By using a supervised learning

---

<sup>26</sup>cf.[Akhtar, Neidhardt, and Werthner 2019] p.398

<sup>27</sup>cf.[Akhtar, Neidhardt, and Werthner 2019] pp.399

<sup>28</sup>cf.[Zini et al. 2019] p.2

scheme to learn a mapping between question and answering pairs and judgement of correct match, this NLP engine should correctly answer questions based on a script.

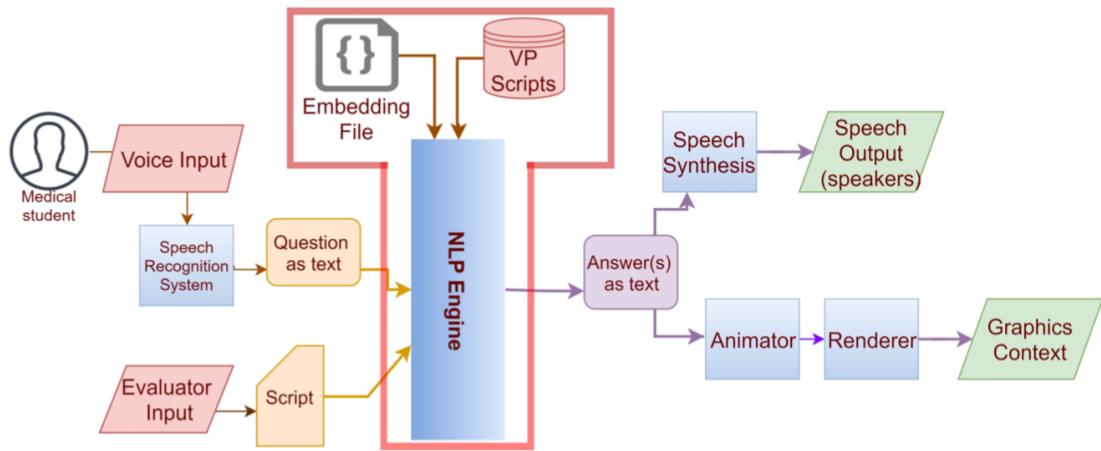


Figure 3.4: Virtual patient software architecture cf.[Zini et al. 2019] p.3

The aim of the developed system was to create a deep learning framework for answer selection in the medical domain and to create domain-specific word and sentence embedding models. Additionally, a question and answering corpus should be created for OSCEs.

**Question and Answering systems** According to Zini et al. there are two types of question and answering systems<sup>29</sup>. First of all, there is open domain question and answering which uses very specific terminology. Secondly there are restricted domain question and answering systems which are broader in their scope. These are for example insurance-related deep learning question and answering systems which make use of two baseline models: Bag Of Words (BOW) and Information Retrieval (IR) model.

<sup>29</sup>cf.[Zini et al. 2019] pp.4

# 4 Analysis and Development

## 4.1 Experimental set-up

First of all, the developed chatbot includes information about blood pressure and was built to remind patients of their measurements. Secondly, based on the measured data, analysis can be done in order to react earlier to outliers. Thirdly, a generated report is sent to the doctor via email so that he can get more insights about the blood pressure values of his patients and can improve the treatment. One of the main challenges during development was the process of providing information about the disease to the patient. Is it possible to include information about different types of blood pressure into the automated conversation with a chatbot? Or does it overwhelm the conversation's use case? Is it useful to let the user ask questions like: 'What are the different types of hypertension? Am i a high-risk patient?' Or should these information be provided as a video or a simple web page with long articles to read? Might the patient or user be aborred after a while of talking to a chatbot who only knows answering his questions in the same way? Of course, a chatbot can be developed in a more intelligentway to never provide the same answer and to answer more precisely to a users' intent. But this requires a lot of training and testing. For that reason, in the first version of this chatbot, five simple intents and dialogs have been designed and implemented with the focus of the instructions to measure correctly and regularly. In a second or third version, it is possible to focus more on the improvement of providing information about the disease (by not doing this in the style of question and answering).

## 4.2 Software architecture

Figure 4.1 gives an overview of all developed components. Quite above, there is the Watson Assistant instance, running in the IBM Cloud. Beneath Watson Assistant, a NodeJS server opens the session and sends messages from the client to the Assistant and backwards. The NodeJS server connects the cloud and the frontend by implementing Hypertext Transfer Protocol (HTTP) requests and responses. Finally, there is the AngularJS application running locally and creating a **JSON!** (**JSON!**) reporting file every few minutes. This reporting file includes all messages, with the user from whom it was send and a timestamp. It can be used to analyze the data and to create profiles of the patients.

## 4.3 Setup of Angular Frontend

In order to provide a comfortable way to chat and view all retrieved and measured data, a basic angular application was built and run locally (see figures 4.2 and 4.3). Three open source libraries, such as Nebular<sup>1</sup>, Apache Echarts<sup>2</sup> and Openlayers Maps<sup>3</sup> were included to the application (see figures 4.2 and 4.3). Nebular is a Javascript library that has certain themes (e.g. light, dark etc.) and a chat UI (which allows to send messages and different file types such as documents or images or videos). It also allows to send a location by defining longitude and latitude parameters. Generally, it is very handy and useful to fast built up an Angular webpage (see also figure 4.2).

Secondly, Apache Echarts is also a Javascript library which offers a huge variety of diagrams and maps, such as bar, line, pie and other special diagrams (including different animation modes, overlays and tooltips). For the first draft of the frontend, the basic line chart was used to display the weekly, monthly and yearly overview of measured pulse, systolic and diastolic values. For future use cases, it might be also useful to display other charts of Echarts or maybe tables to better understand trends. The third Javascript library that was used, is called Openlayers Maps and is

---

<sup>1</sup>cf.[Nebular - Customizable Angular UI Library, Auth and Security 2020]

<sup>2</sup>cf.[Apache ECharts (incubating) 2020]

<sup>3</sup>cf.[OpenLayers v6.1.1 API - Class: Map 2020]

an open source software. It is very famous and many projects rather use Openlayers API! (API!) than Google Places API! because their API! calls are very expensive. Openlayers Maps has many different modes and overlays. Besides, certain places can be marked by different icons. All in all, the three described libraries are very well documented and are easy and ready to use after installation, so that the focus could be spent more on developing the Watson Assistant dialog described in the next paragraph.

## 4.4 Development of Watson Assistant Dialog

### 4.4.1 Intent model

The chatbot was built according to the webpage of "Deutsche Hochdruckliga", a german organization for patients who suffer from hypertension<sup>4</sup>. To better understand the users and patients a basic intent model with four intents was developed. The four intents include

Intent	User input
Definition of hypertension	What is hypertension?
Curses of hypertension	What are the implications or curses of hypertension?
Blood pressure measurement	I am measuring my blood pressure.
Measurement tutorial	How should i measure my blood pressure?

These four intents were used to define and develop four typical dialogs, displayed in figure 4.4, 4.5, 4.6 and 4.7.

### 4.4.2 Watson Assistant implementation

In the following, the implemented dialog as well as all entities and intents are described. They have been developed according to the Watson Assistant documentation<sup>5</sup>. Basically, 16 entities (see figure 4.10) and 17 intents (see figure 4.9) were

<sup>4</sup>cf.[Bluthochdruck vermeiden, behandeln und senken - Aktiv gegen Bluthochdruck 2019]

<sup>5</sup>cf.[Übersicht über die Watson Assistant-API 2020]

defined. Entities give examples of an user's input whereas intents describe the 'wish' or what the user wants to know.

The dialog (see figure 4.8) includes a welcome node, a concierge node which asks the basic data (e.g. age, gender, height, weight etc.) and starts the tutorial, an additional information node (if the user wants to know more about hypertension), the measurement node and finally the recommendation of doctors node. The conversations are implemented as explained in section 4.4.1. In Watson Assistant, each node can be defined with a condition. If the value of the condition is true, the response of the node is returned to the user. If it is false, the next node (usually above the other node) is checked for its condition. In case of no appropriate node, there should be defined a 'anything-else' node which always return a response if the other nodes were not true. Generally, the response of the anything else node is something like 'Sorry, i did not understand. Could you try to rephrase your question, please?'. When it comes to the four intents, the first implementation included passing all nodes, one after another. But as mentioned in section 4.1, there are some special nodes which should be passed only once per user whereas other should be displayed regular intervals. This was not implemented in Watson Assistant (even though there are possibilities, such as defining 'slots' for certain IR routines) but in the frontend which is explained in section 4.3.

## 4.5 Connect Watson Assistant and Angular frontend

To be able to connect to the Watson Assistant instance on IBM cloud, the **API!** Version 2.0 had to be called. First of all, the current sessionId was requested to be able to interact with Watson Assistant. After that, a simple get request was executed to let the chatbot start the conversation. Everytime, the user sends a message to Watson Assistant, a post request is sent to the **API!**. The response is the answer of the current node in Watson Assistant<sup>6</sup>. In order to define the conversation's direction, the intent's probability is calculated (every intent has its own learning set which at least should include six sentences).

---

<sup>6</sup>cf.[Watson Assistant v2 - IBM Cloud API Docs 2020]

### 4.5.1 Data visualization: Development of a Python Script to show all measured values

Based on a given dataset from Kaggle<sup>7</sup>, a basic python script was developed. It compares the different analyzed data

### 4.5.2 Development of email service to send reports every two weeks to the doctor

Another challenge was the way to automatically ask the user for his measured data. Most chatbot only helps in certain situations including precise user intents, e.g. the question 'When should i measure my blood pressure?'. By implementing a python script for data analysis an alarm was created to remind users every five hours or once a day. In order to face this problem or use case, a routine for sending an email including a timer was implemented.

## 4.6 Predictive Analytics: Creating a model to predict cardiovascular disease

Generally, it is hard to let an algorithm diagnose cardiovascular disease if only looking at the dataset above. In practice, it needs the experience of a doctor and many different measured values in different situations (e.g. in stress situations or in relaxing situations). Nevertheless, for analytics it is very interesting to test if for example a neural network could learn on the given dataset and could predict the probability of a cardiovascular disease for a given patient.

### 4.6.1 Dataset

Since the setup of Watson Assistant API and the Angular Frontend took too much development time, a user test could not be executed. For that reason, the diagrams were calculated by using a dataset from Kaggle<sup>8</sup> which included 70000 entries of

---

<sup>7</sup>cf.[Cardiovascular Disease dataset 2020]

<sup>8</sup>cf.[Cardiovascular Disease dataset 2020]

different patients and 12 characteristics. These 12 characteristics included:

- age (in days)
- gender
- height
- weight
- systolic value
- diastolic value
- cholesterol (1: normal, 2: above normal, 3: well above normal)
- gluc (1: normal, 2: above normal, 3: well above normal)
- smoke (binary)
- alco (binary, consume of alcohol)
- active (binary, physical activity)
- presence or absence of cardiovascular disease (binary)

The first four characteristics (age, gender, height, weight) are objective features and give factual information. After that, systolic, diastolic, cholesterol and glucose values are examination features and include the results of medical examination. The last four features (smoke, alcohol, active and the presence of cardiovascular disease) are subjective information and given by the patient.

#### 4.6.2 Development of python script

Based on the given Kaggle Dataset, a python script using Keras library<sup>9</sup> was developed. This script builds up a neural network for an intelligent and fast way to find out whether a person with given health characteristics has a higher or lower risk to suffer from cardiovascular disease. Therefore, all given features were important for the analysis and none was eliminated. Moreover, the **bmi!** (**bmi!**) was added and calculated by the given height and weight values.

During the first step of analysis, the dataset had to be cleaned from null values. After that, a correlation matrix was built up and calculated to check whether all features are relevant for the analysis. This was implemented by using seaborn's (a python library for diagrams<sup>10</sup>)function 'heatmap'. The produced heatmap (see 8) shows the

---

<sup>9</sup>cf.[*Home - Keras Documentation 2020*]

<sup>10</sup>cf.[*seaborn: statistical data visualization — seaborn 0.10.0 documentation 2020*]

parameters of the dataset and how they correlated with the classification variable 'cardio' (which means that the person suffers from cardiovascular disease or not). The result of the correlation matrix shows that all features are relevant for the analysis.

After that, a model with 10 input variables and one target variable was created. The dataset was splitted into both a training and a test set. 30% of the dataset was declared as test set and the target variable was removed. The neural network was built up with three layers: two hidden and one output layer. The first two hidden layer contain 5 nodes each of them (since there are 10 input variables) and are activated by a simple 'relu' activation function. The third layer, the output layer was built up for the classification. Since it is binary classification problem, a sigmoid as the activation function was used. What is more, the Dense layer implements  $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ . The kernel is the weight matrix. The kernel initialization defines the way to set the initial random weights of Keras layers.

To optimize the neural network an Adam function was used<sup>11</sup>. Adam stands for Adaptive moment estimation and combines RMSProp (An optimization method closely related to Adam) and Momentum. Momentum takes the past gradients into account in order to smooth out the gradient descent. 'RMSProp with momentum generates its parameter updates using a momentum on the rescaled gradient, whereas Adam updates are directly estimated using a running average of first and second moment of the gradient. RMSProp also lacks a bias-correction term; this matters most in case of a small value  $\beta$  (required in case of sparse gradients), since in that case not correcting the bias leads to very large stepsizes and often divergence.'<sup>12</sup>

After the model building phase, the neural network was trained by setting 10 epochs and a batch size of 10. After the training, the model was evaluated and the result was an accuracy of 73 %.

## 4.7 Results

Referring to the above section 4.6.2, the neural network gets an accuracy of 73%. But this can be improved by changing the number of epochs and batches. Another possibility might be the change of the dataset. It might be useful to evaluate the data

---

<sup>11</sup>cf.[Kingma and Ba 2017], p.7

<sup>12</sup>cf.[Kingma and Ba 2017], p.5

retrieved from real users who chatted with the developed chatbot. This data could be more realistic than the data from Kaggle.

### 4.7.1 Challenges

During the development of the frontend and Watson Assistant, there were many challenges. One big challenge was the development and testing of the dialogs. Watson Assistant gives a great possibility to develop conversations in a decision tree but for testing, it does not offer any automated tests (which would be very helpful for large-scale analysis). Another challenge was the connection of Watson Assistant Application Programmers Interface (API) and the Angular app. The documentation of Watson Assistant is kept very basic and there are not many examples on Github. Finally, the connection was implemented by an easy HTTP get and post request. To give another example, the implementation of the OpenLayers Map was very challenging since the main configuration can only be done through the Hypertext Markup Language (HTML) template.

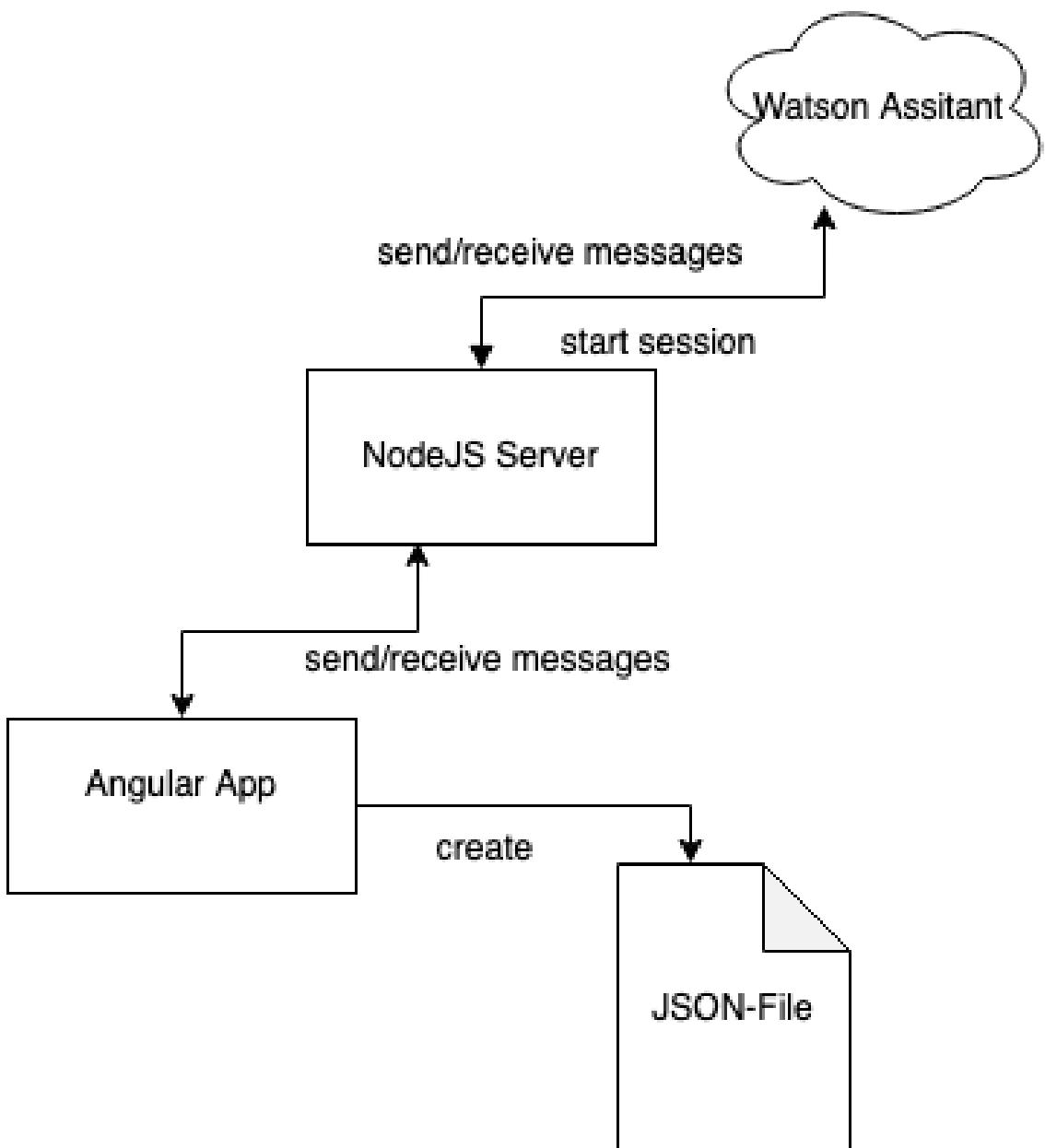


Figure 4.1: Architecture diagram of developed system

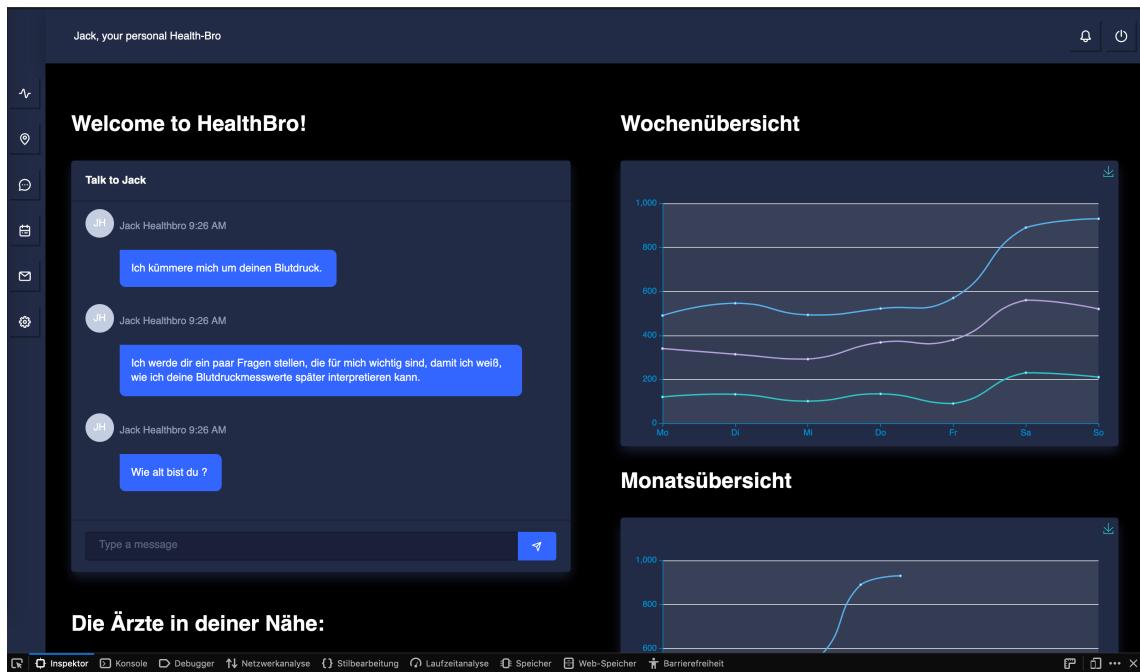


Figure 4.2: Angular Frontend (top page), chat UI and diagrams of measured values

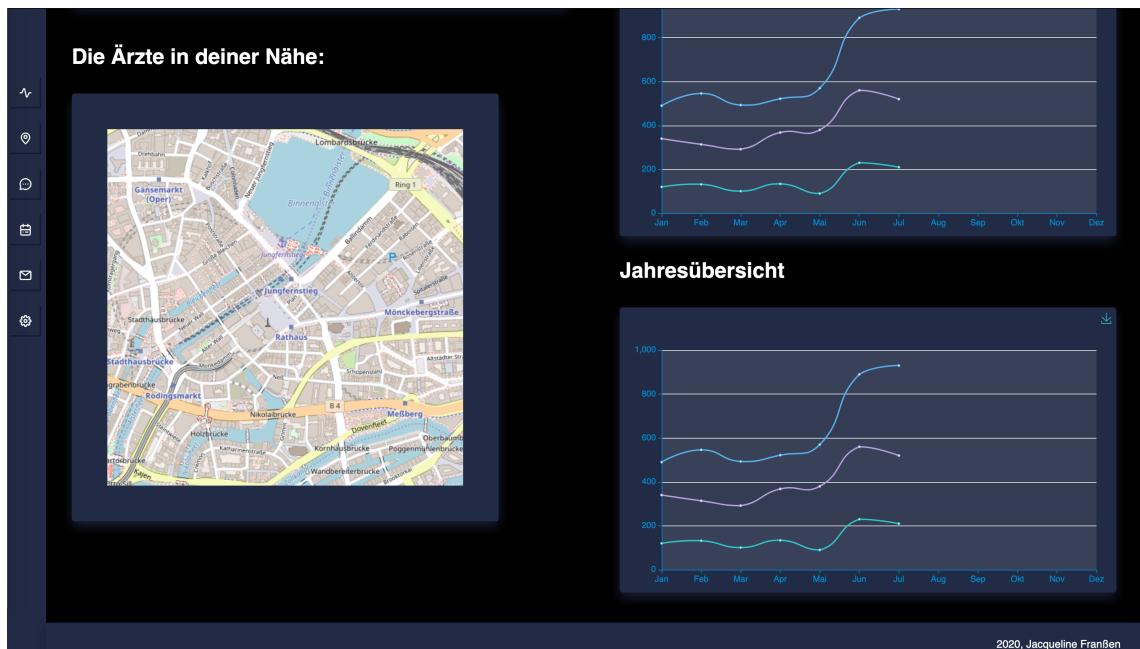


Figure 4.3: Angular Frontend (bottom page), map to show the nearest doctors

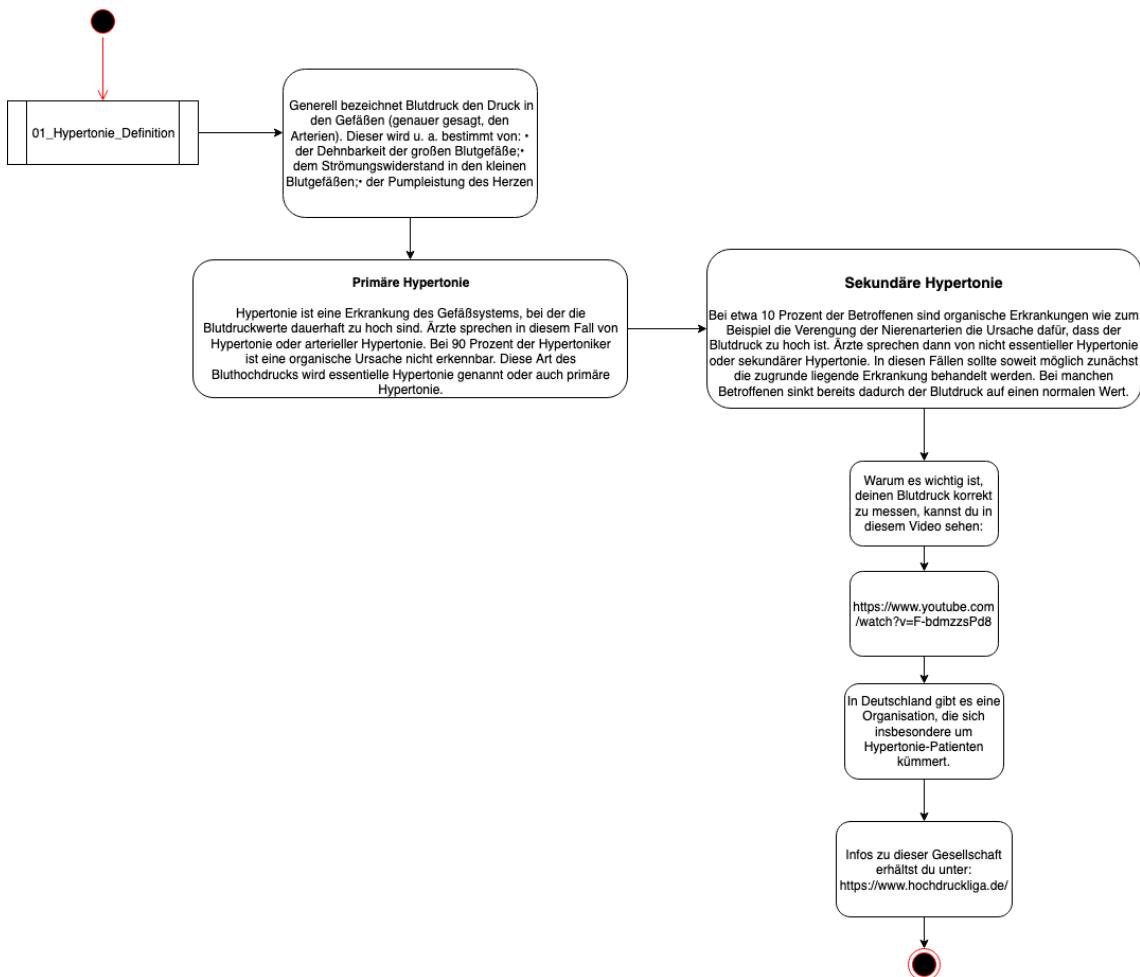


Figure 4.4: Dialog diagram: definition of hypertension

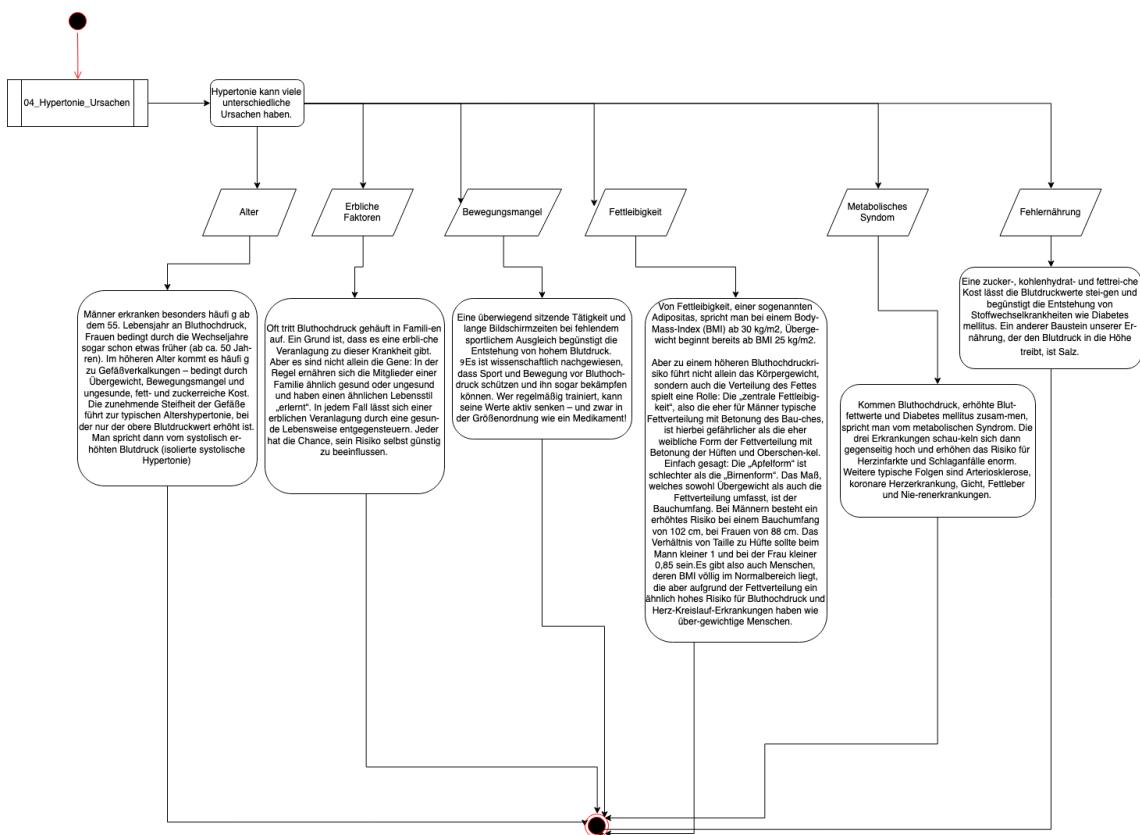


Figure 4.5: Dialog diagram: curses of hypertension

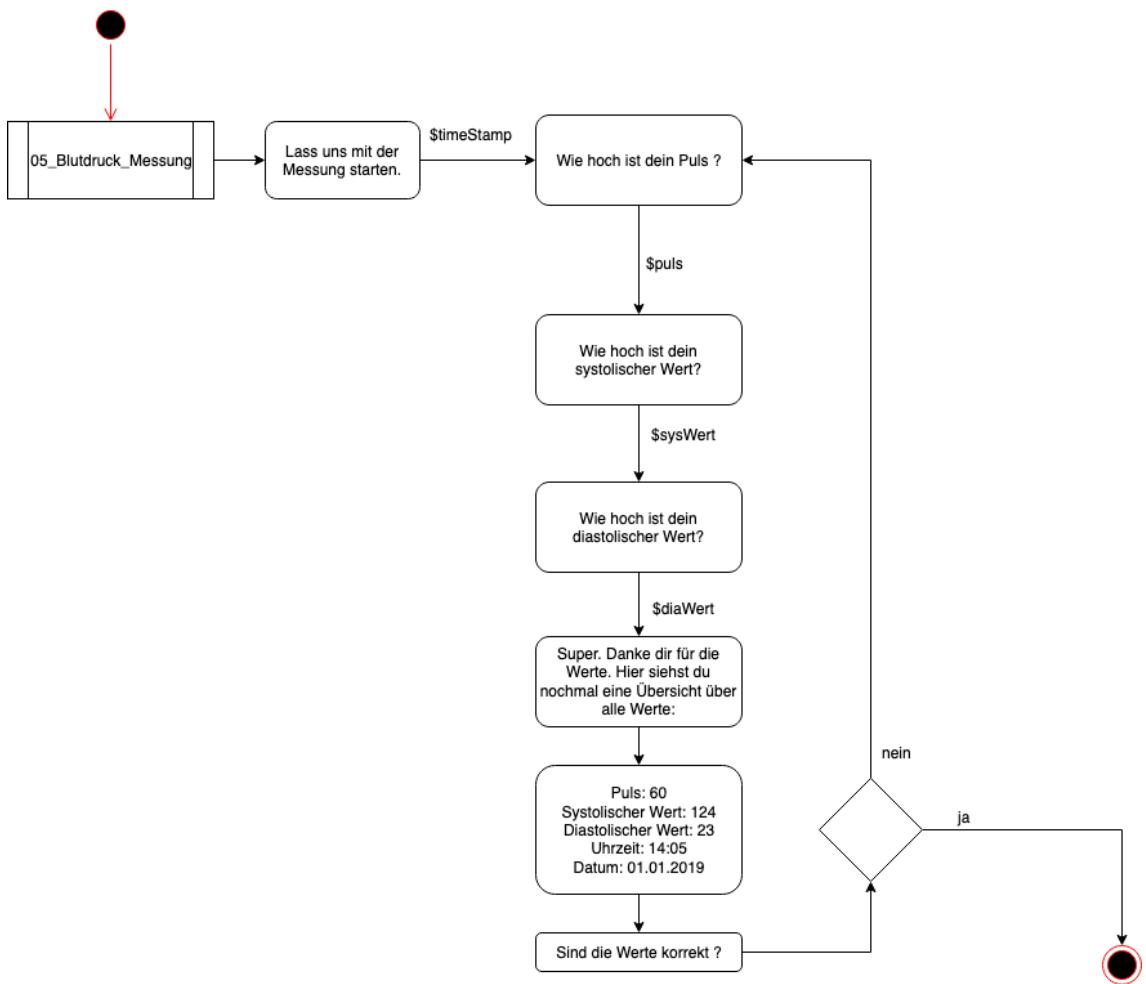


Figure 4.6: Dialog diagram: blood pressure measurement

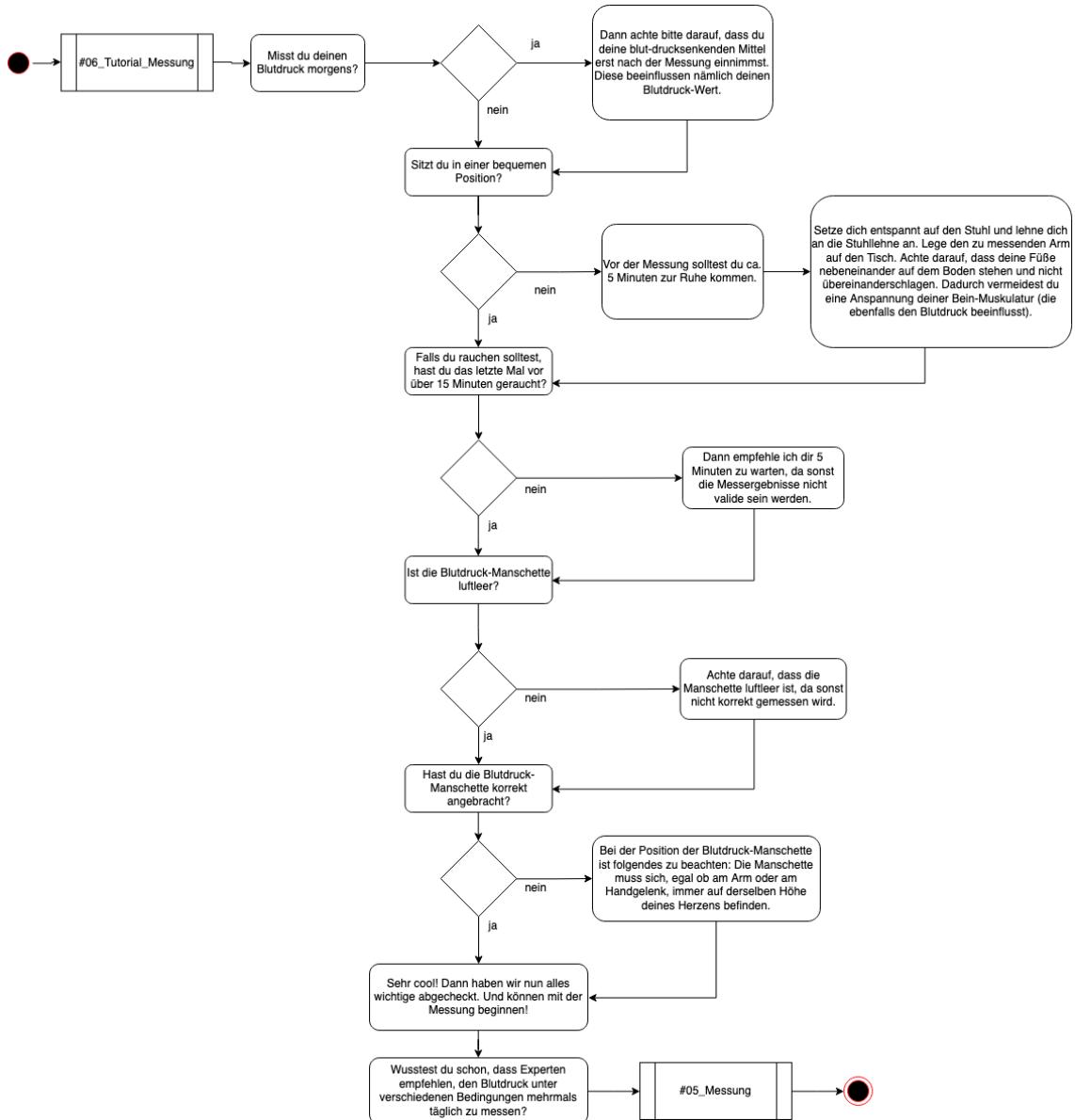


Figure 4.7: Dialog diagram: measurement tutorial

The screenshot shows the IBM Watson Assistant interface for a skill titled "My first skill". On the left, a sidebar lists navigation options: Intents, Entities, Dialog (which is selected), Options, Analytics, Versions, and Content Catalog. The main area displays a list of intents, each represented by a card:

- Willkommen**: welcome. 1 Responses / 0 Context Set / Jump to / Does not return.
- Concierge**. 0 Responses / 0 Context Set / Skip user input / Does not return.
- Zusatzinfos**: true. 0 Responses / 0 Context Set / Skip user input / Does not return.
- Blutdruck\_Messung**: true. 1 Responses / 5 Context Set / Skip user input / Does not return.
- Arzt\_Vorschlag**: #Arzt\_suche. 1 Responses / 0 Context Set / Does not return.
- Abschied**: true. 1 Responses / 0 Context Set / Does not return.
- Anything else**: anything\_else. 1 Responses / 0 Context Set / Does not return.

At the top right, there are buttons for "Save new version", "Try it", and a more options menu. At the bottom right, there is a search icon and a help icon.

Figure 4.8: Watson Assistant dialog

The screenshot shows the IBM Watson Assistant interface for a skill titled "My first skill". On the left, a sidebar lists navigation options: Intents (selected), Entities, Dialog, Options, Analytics, Versions, and Content Catalog. The main area displays a table of intents:

	Description	Modified	Examples
<input type="checkbox"/> #01_Hypertonie_Definition	a month ago	1	
<input type="checkbox"/> #02_Hypertonie_Schaeden	a month ago	1	
<input type="checkbox"/> #03_Hypertonie_Risiko	a month ago	2	
<input type="checkbox"/> #04_Hypertonie_Symptome	a month ago	0	
<input type="checkbox"/> #05_Blutdruck_Messung	a month ago	5	
<input type="checkbox"/> #06_Messung_Tutorial	a month ago	7	
<input type="checkbox"/> #Arztsuche	2 months ago	10	
<input type="checkbox"/> #Befinden_egal	3 months ago	8	
<input type="checkbox"/> #Befinden_negativ	3 months ago	8	
<input type="checkbox"/> #Befinden_positiv	3 months ago	8	
<input type="checkbox"/> #DiagrammEinsehen	2 months ago	6	
<input type="checkbox"/> #Feedback_negativ	3 months ago	6	
<input type="checkbox"/> #Feedback_positiv	3 months ago	6	
<input type="checkbox"/> #letzteMessung	2 months ago	9	

At the top right, there are buttons for "Save new version", "Try it", and a more options menu. At the bottom right, there is a search icon and a help icon.

Figure 4.9: Watson Assistant intents

The screenshot shows the IBM Watson Assistant interface for managing entities in a skill named "My first skill". The left sidebar includes links for Intents, Entities, My entities (which is selected), System entities, Dialog, Options, Analytics, Versions, and Content Catalog. The main area displays a table of entities with columns for Name, Value, and Modified date. There are 16 entries listed:

Name	Value	Modified
@Alter	Alter	21 days ago
@Alter_Patient	20	23 days ago
@Atemnot	Luftnot	2 months ago
@Atmen	Ausatmen, Luftholen	2 months ago
@Bewegungsmangel	Bewegungsmangel	21 days ago
@Blutdruck	Druck im Blut	2 months ago
@Erbliche_Faktoren	Erbliche Faktoren	21 days ago
@Fehlernahrung	Fehlernährung	21 days ago
@Fettleibigkeit	Fettleibigkeit	21 days ago
@Geschlecht	männlich, weiblich	2 months ago
@Herz	Mitte	2 months ago
@Herzrasen	Herzpochen	2 months ago
@Metabolisches_Syndrom	metabolisch, Metabolisches Syndrom	21 days ago
@Pochen	Schlagen	2 months ago

Figure 4.10: Watson Assistant entities

# 5 Conclusion and Outlook

## 5.1 Conclusion

To conclude, the developed system shows that through automatisation and human-like conversations, hypertonia patients can be guided through their measurements. What is more, the retrieved information can be used to run several analysis and to find out trends in the data. This enables doctors to react faster and to improve and personalize the treatment individually for each patient. Furthermore, the developed solution supports users to never forget to measure and doctors to better understand trends in the measured data. During development and research, there came up another issue: To inform patients precisely about their illness. Most patients go to doctors and describe their symptoms, the doctor makes some diagnosis and provides them some medication. But in most cases, doctors do not have enough time to answer all the questions of their patients. For that reason it might be useful to provide a 24/7 service for patients with chronic disease to both record their symptoms and measurements and to answer all their questions. This could be implemented by a chatbot or by adding analysis techniques, such as image recognition. By analyzing the images, an algorithm could check the state of the patient (e.g. if he or she looks like always or not).

## 5.2 Outlook

There are many things that can be added or improved, such as connecting the doctors' and patient's calendar to directly make an appointment through the chatbot. Another possibility is encryption. It is important to store all measured and personal data in a secure way. Asynchronous encryption is a good way to save these data and to

only allow access to the parties that should see the information, such as patients. To give an example, the application could save the data to the cloud and for this one action it uses a public key. If the patient wants to get to this information, he needs a private key to decrypt all information. As described in section 4.6, a neural network can calculate the probability of a patient to suffer cardiovascular disease. Moreover, another use case might be patients already suffering from cardiovascular disease which have to be treated and want to recover. In that situation the neural network could calculate the ideal weight and all other relevant factors and how to get to these ideal values. It could motivate patients through a personalized treatment plan by providing personalized nutrition plans. It could also forecast the term or maybe exact date when the patients will be healthy again. Besides, for doctors, it might be very useful to have a proper dashboard which shows a list of all patients and a 'profile' for each patient. This could contain several personal data as well as the last treatments and prescribed medication. Finally, an improved architecture which is described in the next section connects all components (analysis and frontend).

### 5.2.1 Connect Flask app, python script and Angular frontend

Flask is a python library which starts a web server to run the script in the browser<sup>1</sup>. Figure 5.1 shows the 'perfect' architecture. The idea is to connect the python script with the Angular web app by starting the flask server and connecting both of them. Another, easier way could be to include python directly into Angular. But this might be a little complicated since Angular is a typescript framework and the two languages might cause some conflicts. Nevertheless, the given architecture 5.1 shows the idea of saving the created data from the chat and directly using them for the analysis script. This offers the new possibility to evaluate and predict in real time.

---

<sup>1</sup>cf.[Flask 2020]

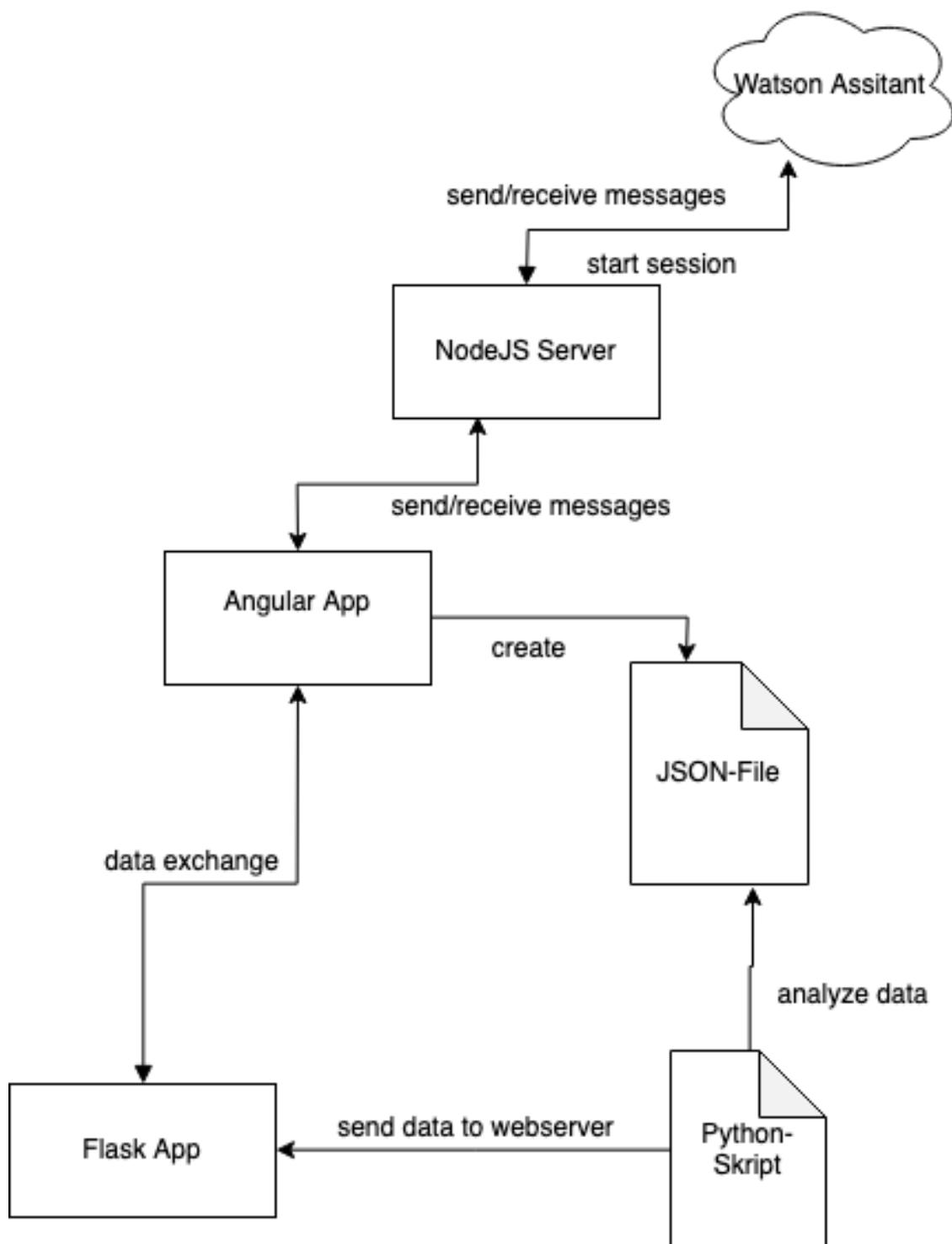


Figure 5.1: Watson Assistant entities

## 6 Abbreviations

<b>NLP</b>	Natural Language Processing
<b>ESARC</b>	Enterprise Software Architecture Reference Cube
<b>M-HEALTH</b>	medical health
<b>P2P</b>	Peer to Peer
<b>NCD</b>	non-communicable disease
<b>POC</b>	Proof of Disease
<b>IOT</b>	Internet of Things
<b>MEM</b>	medical episode mining
<b>HSM</b>	health state mining
<b>ICD</b>	International Classification of Disease
<b>SNOWMED CT</b>	Systemized NOmenclature of MEDicine Clinical Terms
<b>OSCE</b>	Objective Structured Clinical Examination
<b>CNN</b>	Convolutional Neural Network
<b>LSTM</b>	Long Short Term Memory
<b>BOW</b>	Bag Of Words
<b>IR</b>	Information Retrieval
<b>SOA</b>	Service-Oriented enterprise Architecture
<b>ITIL</b>	Information Technology Infrastructure Library

<b>HCP</b>	Health Care Provider
<b>TOGAF</b>	The Open Group Architecture Framework
<b>BCT</b>	Behavioural Change Techniques
<b>HTTP</b>	Hypertext Transfer Protocol
<b>HTML</b>	Hypertext Markup Language
<b>API</b>	Application Programmers Interface
<b>JSON</b>	Javascript Object Notation

# Bibliography

- Akhtar, Mubashra, Julia Neidhardt, and Hannes Werthner (July 2019). “The Potential of Chatbots: Analysis of Chatbot Conversations”. In: *2019 IEEE 21st Conference on Business Informatics (CBI)*. Vol. 01. ISSN: 2378-1963, pp. 397–404. doi: 10.1109/CBI.2019.00052.
- Alessa, Tourkiah et al. (May 2019). “Smartphone Apps to Support Self-Management of Hypertension: Review and Content Analysis”. In: *JMIR mHealth and uHealth* 7.5. ISSN: 2291-5222. doi: 10.2196/13645. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6658295/> (visited on 01/05/2020).
- Apache ECharts (incubating)* (2020). URL: <https://www.echartsjs.com/en/index.html> (visited on 02/02/2020).
- Bluthochdruck vermeiden, behandeln und senken - Aktiv gegen Bluthochdruck* (2019). de. URL: <https://www.hochdruckliga.de/> (visited on 11/23/2019).
- Cardiovascular Disease dataset* (2020). en. URL: <https://kaggle.com/sulianova/cardiovascular-disease-dataset> (visited on 01/19/2020).
- Flask* (2020). URL: <https://palletsprojects.com/p/flask/> (visited on 02/09/2020).
- Home - Keras Documentation* (2020). URL: <https://keras.io/> (visited on 02/09/2020).
- Kingma, Diederik P. and Jimmy Ba (Jan. 2017). “Adam: A Method for Stochastic Optimization”. en. In: *arXiv:1412.6980 [cs]*. arXiv: 1412.6980. (Visited on 02/09/2020).
- Lupton, Deborah (Aug. 2012). “M-health and health promotion: The digital cyborg and surveillance society”. en. In: *Social Theory & Health* 10.3, pp. 229–244. ISSN: 1477-8211, 1477-822X. doi: 10.1057/sth.2012.6. URL: <http://link.springer.com/10.1057/sth.2012.6> (visited on 12/27/2019).
- (Dec. 2014). “Apps as Artefacts: Towards a Critical Perspective on Mobile Health and Medical Apps”. en. In: *Societies* 4.4, pp. 606–622. doi: 10.3390/soc4040606. URL: <https://www.mdpi.com/2075-4698/4/4/606> (visited on 12/27/2019).

- Nebular - Customizable Angular UI Library, Auth and Security* (2020). URL: <https://akveo.github.io/nebular/> (visited on 02/02/2020).
- OpenLayers v6.1.1 API - Class: Map* (2020). URL: [https://openlayers.org/en/latest/apidoc/module-ol\\_Map.html](https://openlayers.org/en/latest/apidoc/module-ol_Map.html) (visited on 02/02/2020).
- seaborn: statistical data visualization — seaborn 0.10.0 documentation* (2020). URL: <https://seaborn.pydata.org/> (visited on 02/09/2020).
- Talukder, Asoke K et al. (Oct. 2018). "Proof of Disease: A Blockchain Consensus Protocol for Accurate Medical Decisions and Reducing the Disease Burden". In: *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. ISSN: null, pp. 257–262. doi: [10.1109/SmartWorld.2018.00079](https://doi.org/10.1109/SmartWorld.2018.00079).
- Übersicht über die Watson Assistant-API* (2020). URL: <https://cloud.ibm.com/docs/services/assistant?topic=assistant-api-overview> (visited on 01/10/2020).
- Watson Assistant v2 - IBM Cloud API Docs* (2020). URL: <https://cloud.ibm.com/apidocs/assistant/assistant-v2> (visited on 01/10/2020).
- White, William B (2007). *Blood pressure monitoring in cardiovascular medicine and therapeutics*. en. OCLC: 915957468. Totowa, N.J.: Humana Press. ISBN: 978-1-58829-512-5. URL: <http://www.springerlink.com/openurl.asp?genre=book&isbn=978-1-58829-512-5> (visited on 01/03/2020).
- Zimmermann, Alfred et al. (Sept. 2013). "Towards Service-Oriented Enterprise Architectures for Big Data Applications in the Cloud". In: *2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops*. ISSN: 2325-6605, pp. 130–135. doi: [10.1109/EDOCW.2013.21](https://doi.org/10.1109/EDOCW.2013.21).
- Zini, Julia El et al. (July 2019). "Towards A Deep Learning Question-Answering Specialized Chatbot for Objective Structured Clinical Examinations". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4393, pp. 1–9. doi: [10.1109/IJCNN.2019.8851729](https://doi.org/10.1109/IJCNN.2019.8851729).

## 7 Appendix A

### **Ehrenwörtliche Erklärung**

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde / Prüfungsstelle vorgelegen hat. Ich erkläre mich damit nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Ort, Datum (Vorname Nachname)

## **8 Appendix B**

# keras\_classifier07.02.20

February 7, 2020

## 0.1 Data preparation

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

## 0.2 import dataset

```
[2]: dataset = pd.read_csv("./cardio_train.csv", sep=';')
```

```
[3]: dataset.head(2)
```

```
[3]:   id      age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke \
0    0    18393       2     168     62.0     110      80            1      1      0
1    1    20228       1     156     85.0     140      90            3      1      0

      alco  active  cardio
0      0       1       0
1      0       1       1
```

```
[4]: #get all types of dataset
dataset.describe(include='all')
```

```
[4]:          id        age      gender      height      weight \
count  70000.000000  70000.000000  70000.000000  70000.000000  70000.000000
mean   49972.419900  19468.865814    1.349571   164.359229   74.205690
std    28851.302323  2467.251667    0.476838    8.210126   14.395757
min     0.000000  10798.000000    1.000000    55.000000   10.000000
25%   25006.750000  17664.000000    1.000000   159.000000   65.000000
50%   50001.500000  19703.000000    1.000000   165.000000   72.000000
75%   74889.250000  21327.000000    2.000000   170.000000   82.000000
max   99999.000000  23713.000000    2.000000   250.000000  200.000000

          ap_hi      ap_lo  cholesterol      gluc      smoke \
count  70000.000000  70000.000000  70000.000000  70000.000000  70000.000000
```

mean	128.817286	96.630414	1.366871	1.226457	0.088129
std	154.011419	188.472530	0.680250	0.572270	0.283484
min	-150.000000	-70.000000	1.000000	1.000000	0.000000
25%	120.000000	80.000000	1.000000	1.000000	0.000000
50%	120.000000	80.000000	1.000000	1.000000	0.000000
75%	140.000000	90.000000	2.000000	1.000000	0.000000
max	16020.000000	11000.000000	3.000000	3.000000	1.000000

	alco	active	cardio
count	70000.000000	70000.000000	70000.000000
mean	0.053771	0.803729	0.499700
std	0.225568	0.397179	0.500003
min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000

### 0.3 calculate bmi and years of age, delete unneeded columns

```
[31]: dataset['years'] = (dataset['age'] / 365).round().astype('int')
dataset['BMI'] = dataset['weight']/((dataset['height']/100)**2)
dataset.isnull().values.any()
dataset.drop(['id', 'age', 'weight', 'height'], axis=1)
```

	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	\
0	2	110	80		1	1	0	0	1	0
1	1	140	90		3	1	0	0	1	1
2	1	130	70		3	1	0	0	0	1
3	2	150	100		1	1	0	0	1	1
4	1	100	60		1	1	0	0	0	0
5	1	120	80		2	2	0	0	0	0
6	1	130	80		3	1	0	0	1	0
7	2	130	90		3	3	0	0	1	1
8	1	110	70		1	1	0	0	1	0
9	1	110	60		1	1	0	0	0	0
10	1	120	80		1	1	0	0	1	0
11	2	120	80		1	1	0	0	1	0
12	2	120	80		1	1	0	0	0	0
13	1	110	70		1	1	0	0	1	0
14	2	130	90		1	1	1	1	1	0
15	2	120	80		1	1	0	0	0	1
16	1	130	70		1	1	0	0	0	0
17	1	110	70		1	3	0	0	1	0
18	1	100	70		1	1	0	0	0	0
19	2	120	70		1	1	1	0	1	0
20	2	120	80		1	1	0	0	1	0

21	1	130	80	1	1	0	0	1	0
22	1	145	85	2	2	0	0	1	1
23	2	110	60	1	1	0	0	1	0
24	1	150	90	3	1	0	0	1	1
25	1	130	100	2	1	0	0	1	0
26	1	130	90	1	1	0	0	1	0
27	1	120	80	1	1	0	0	1	0
28	2	120	80	1	1	0	0	1	0
29	2	130	70	1	3	0	0	0	0
...	...	...	...	...	...	...	...	...	...
69970	2	140	80	3	1	1	1	0	1
69971	2	130	80	1	1	0	0	1	0
69972	1	140	90	1	1	0	0	1	1
69973	2	130	80	1	1	0	0	1	0
69974	1	120	80	1	1	0	0	1	0
69975	2	120	80	1	1	0	0	1	1
69976	1	120	80	2	2	0	0	1	0
69977	1	120	79	1	1	0	0	1	0
69978	1	90	60	1	1	0	0	1	1
69979	1	160	100	2	2	0	0	1	1
69980	2	110	80	1	1	0	1	0	0
69981	2	130	90	2	2	0	0	1	1
69982	1	130	90	1	2	0	0	1	1
69983	1	120	80	1	1	0	0	1	0
69984	2	120	80	1	1	0	0	1	1
69985	1	130	80	1	1	0	1	0	1
69986	2	120	80	1	1	0	0	1	0
69987	1	120	80	1	1	0	0	1	0
69988	1	110	70	1	1	0	0	1	0
69989	1	120	70	1	1	0	0	1	1
69990	1	110	70	1	1	0	0	1	1
69991	1	130	90	2	2	0	0	1	0
69992	1	170	90	1	1	0	0	1	1
69993	1	130	90	1	1	0	0	1	1
69994	1	150	80	1	1	0	0	1	1
69995	2	120	80	1	1	1	0	1	0
69996	1	140	90	2	2	0	0	1	1
69997	2	180	90	3	1	0	1	0	1
69998	1	135	80	1	2	0	0	0	1
69999	1	120	80	2	1	0	0	1	0

years	BMI
0	50 21.967120
1	55 34.927679
2	52 23.507805
3	48 28.710479
4	48 23.011177

5	60	29.384676
6	61	37.729725
7	62	29.983588
8	48	28.440955
9	54	25.282570
10	62	28.010224
11	52	20.047446
12	41	22.038567
13	54	31.244993
14	40	28.997894
15	46	37.858302
16	58	25.951557
17	46	20.829995
18	48	28.672626
19	60	21.338211
20	54	31.239414
21	59	27.993022
22	63	36.051915
23	64	18.491124
24	46	23.529412
25	40	27.767098
26	54	24.243918
27	50	30.853210
28	40	23.951227
29	58	25.909457
...	...	...
69970	62	34.414782
69971	55	25.535446
69972	47	27.915519
69973	61	23.510204
69974	50	26.573129
69975	58	30.189591
69976	59	24.464602
69977	46	26.573129
69978	52	29.357522
69979	61	27.852008
69980	49	24.740937
69981	48	33.208550
69982	52	36.738007
69983	54	26.446281
69984	49	28.344671
69985	50	41.913215
69986	50	24.074074
69987	52	21.490286
69988	60	23.046875
69989	58	33.672766
69990	41	25.510204

```
69991      56  28.479886
69992      51  21.604105
69993      54  23.661439
69994      58  29.384757
69995      53  26.927438
69996      62  50.472681
69997      52  31.353579
69998      61  27.099251
69999      56  24.913495
```

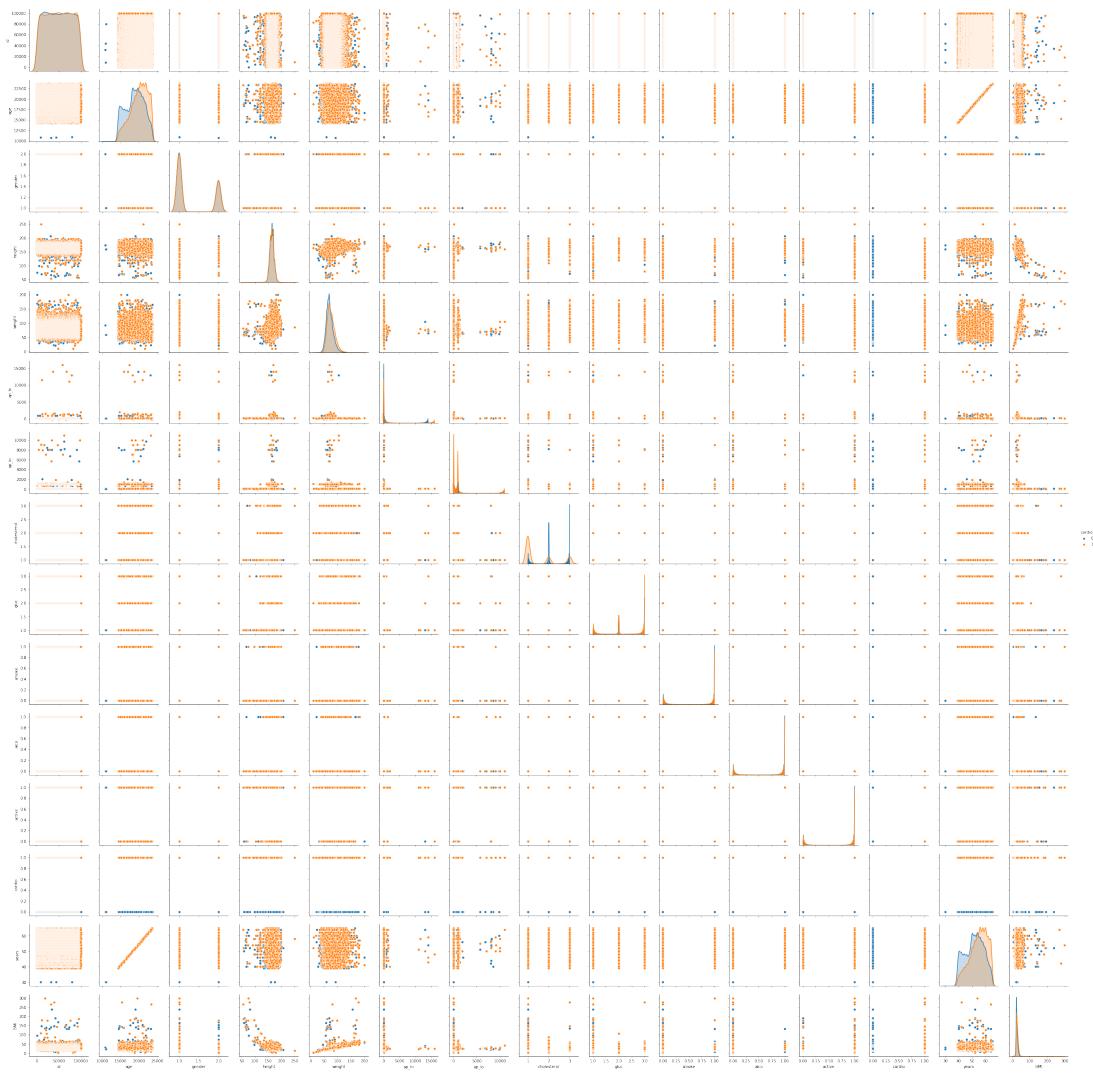
```
[70000 rows x 11 columns]
```

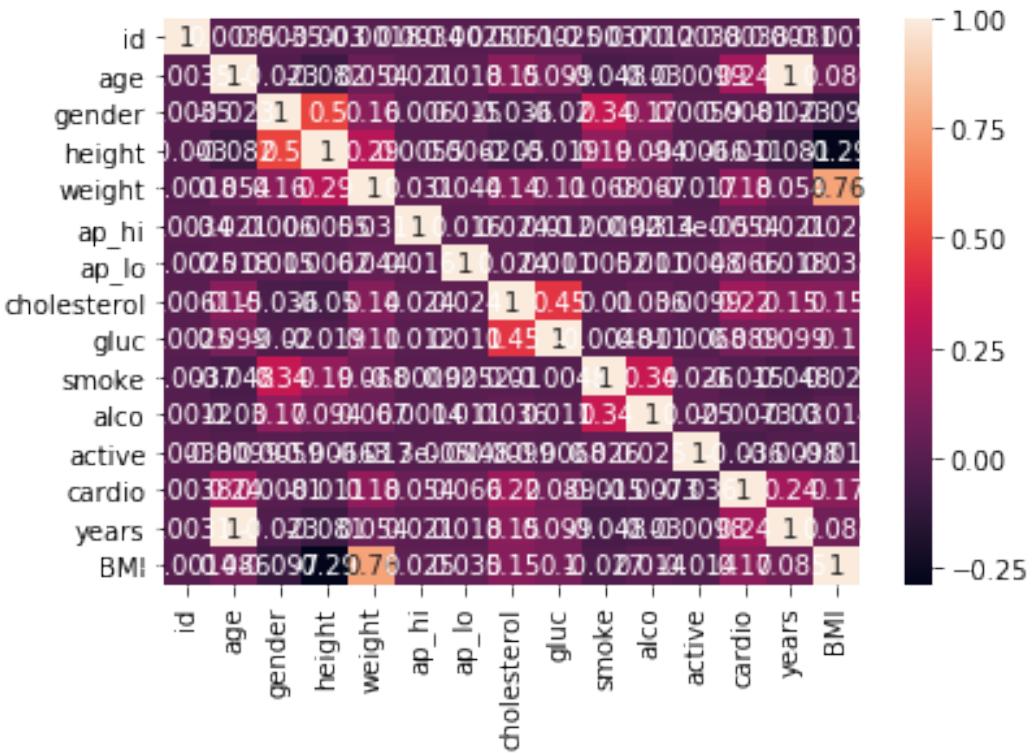
#### 0.4 plot data (pairplot and heatmap for correlation)

```
[32]: sns.pairplot(dataset, hue='cardio')
```

```
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:488:
RuntimeWarning: invalid value encountered in true_divide
    binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kdetools.py:34:
RuntimeWarning: invalid value encountered in double_scalars
    FAC1 = 2*(np.pi*bw/RANGE)**2
```

```
[32]: <seaborn.axisgrid.PairGrid at 0x1a1ba7d780>
```





## 0.5 create input features and target variables for neural network

```
[50]: # creating input features and target variables
X= dataset.drop(['cardio', 'id', 'age', 'height', 'weight'], axis=1)
y= dataset.
    ↪drop(['id', 'height', 'weight', 'age', 'gender', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'smoke', 'alco',
    ↪axis=1)
```

```
[51]: X.head(2)
```

```
[51]:   gender  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  years \
0        2     110      80            1      1      0      0        1      50
1        1     140      90            3      1      0      0        1      55

          BMI
0  21.967120
1  34.927679
```

## 0.6 normalization of input features

```
[56]: #standardizing the input feature
#Since our input features are at different scales we need to standardize the ↵
#input.

# (Normalization)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
X
```

```
[56]: array([[ 1.36405487, -0.12218198, -0.0882385 , ...,  0.49416711,
   -0.49350546, -0.91757729],
   [-0.73310834,  0.07261016, -0.03517999, ...,  0.49416711,
    0.24556599,  1.21008057],
   [-0.73310834,  0.00767945, -0.14129701, ..., -2.02360695,
   -0.19787688, -0.66465218],
   ...,
   [ 1.36405487,  0.33233302, -0.03517999, ..., -2.02360695,
   -0.19787688,  0.62334178],
   [-0.73310834,  0.04014481, -0.0882385 , ..., -2.02360695,
    1.13245175, -0.07506591],
   [-0.73310834, -0.05725127, -0.0882385 , ...,  0.49416711,
    0.39338029, -0.4338885 ]])
```

## 0.7 Model Building

```
[57]: # split the input features and target variables into training dataset and test ↵
#dataset.
# test dataset will be 30% of our entire dataset.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
[58]: from keras import Sequential
from keras.layers import Dense
```

```
[59]: classifier = Sequential()
#First Hidden Layer
classifier.add(Dense(5, activation='relu', kernel_initializer='random_normal', ↵
   input_dim=10))#Second Hidden Layer
classifier.add(Dense(5, activation='relu', ↵
   kernel_initializer='random_normal'))#Output Layer
classifier.add(Dense(1, activation='sigmoid', ↵
   kernel_initializer='random_normal'))
```

```
[60]: #Compiling the neural network  
classifier.compile(optimizer ='adam',loss='binary_crossentropy', metrics  
↳=['accuracy'])
```

## 0.8 Model training

```
[62]: #Fitting the data to the training dataset  
classifier.fit(X_train,y_train, batch_size=10, epochs=100)
```

```
Epoch 1/100  
49000/49000 [=====] - 4s 82us/step - loss: 0.5421 -  
acc: 0.7324  
Epoch 2/100  
49000/49000 [=====] - 4s 81us/step - loss: 0.5423 -  
acc: 0.7334  
Epoch 3/100  
49000/49000 [=====] - 4s 84us/step - loss: 0.5422 -  
acc: 0.7319  
Epoch 4/100  
49000/49000 [=====] - 4s 82us/step - loss: 0.5420 -  
acc: 0.7328  
Epoch 5/100  
49000/49000 [=====] - 4s 82us/step - loss: 0.5421 -  
acc: 0.7330  
Epoch 6/100  
49000/49000 [=====] - 4s 82us/step - loss: 0.5421 -  
acc: 0.7328  
Epoch 7/100  
49000/49000 [=====] - 4s 83us/step - loss: 0.5420 -  
acc: 0.7335  
Epoch 8/100  
49000/49000 [=====] - 4s 84us/step - loss: 0.5421 -  
acc: 0.7331  
Epoch 9/100  
49000/49000 [=====] - 4s 86us/step - loss: 0.5421 -  
acc: 0.7327  
Epoch 10/100  
49000/49000 [=====] - 4s 84us/step - loss: 0.5420 -  
acc: 0.7327  
Epoch 11/100  
49000/49000 [=====] - 4s 87us/step - loss: 0.5421 -  
acc: 0.7326  
Epoch 12/100  
49000/49000 [=====] - 4s 84us/step - loss: 0.5422 -  
acc: 0.7330  
Epoch 13/100  
49000/49000 [=====] - 4s 85us/step - loss: 0.5418 -
```

```
acc: 0.7328
Epoch 14/100
49000/49000 [=====] - 4s 84us/step - loss: 0.5421 -
acc: 0.7330
Epoch 15/100
49000/49000 [=====] - 4s 84us/step - loss: 0.5419 -
acc: 0.7336
Epoch 16/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5418 -
acc: 0.7321
Epoch 17/100
49000/49000 [=====] - 4s 85us/step - loss: 0.5418 -
acc: 0.7332
Epoch 18/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5420 -
acc: 0.7324
Epoch 19/100
49000/49000 [=====] - 4s 89us/step - loss: 0.5418 -
acc: 0.7336
Epoch 20/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5418 -
acc: 0.7336
Epoch 21/100
49000/49000 [=====] - 4s 84us/step - loss: 0.5416 -
acc: 0.7338
Epoch 22/100
49000/49000 [=====] - 4s 85us/step - loss: 0.5418 -
acc: 0.7330
Epoch 23/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5417 -
acc: 0.7340
Epoch 24/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5417 -
acc: 0.7336
Epoch 25/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5418 -
acc: 0.7324
Epoch 26/100
49000/49000 [=====] - 4s 87us/step - loss: 0.5416 -
acc: 0.7327
Epoch 27/100
49000/49000 [=====] - 4s 87us/step - loss: 0.5419 -
acc: 0.7329
Epoch 28/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5415 -
acc: 0.7337
Epoch 29/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5415 -
```

```
acc: 0.7324
Epoch 30/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5416 -
acc: 0.7325
Epoch 31/100
49000/49000 [=====] - 4s 89us/step - loss: 0.5418 -
acc: 0.7339
Epoch 32/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5418 -
acc: 0.7320
Epoch 33/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5418 -
acc: 0.7336
Epoch 34/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5416 -
acc: 0.7324
Epoch 35/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5417 -
acc: 0.7339
Epoch 36/100
49000/49000 [=====] - 4s 87us/step - loss: 0.5417 -
acc: 0.7324
Epoch 37/100
49000/49000 [=====] - 4s 90us/step - loss: 0.5416 -
acc: 0.7334
Epoch 38/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5418 -
acc: 0.7332
Epoch 39/100
49000/49000 [=====] - 4s 90us/step - loss: 0.5416 -
acc: 0.7340
Epoch 40/100
49000/49000 [=====] - 4s 89us/step - loss: 0.5414 -
acc: 0.7339
Epoch 41/100
49000/49000 [=====] - 4s 88us/step - loss: 0.5415 -
acc: 0.7331
Epoch 42/100
49000/49000 [=====] - 4s 90us/step - loss: 0.5415 -
acc: 0.7326
Epoch 43/100
49000/49000 [=====] - 4s 90us/step - loss: 0.5415 -
acc: 0.7328
Epoch 44/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5416 -
acc: 0.7334
Epoch 45/100
49000/49000 [=====] - 4s 90us/step - loss: 0.5414 -
```

```
acc: 0.7330
Epoch 46/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5416 -
acc: 0.7336
Epoch 47/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5414 -
acc: 0.7332
Epoch 48/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5416 -
acc: 0.7336
Epoch 49/100
49000/49000 [=====] - 5s 92us/step - loss: 0.5416 -
acc: 0.7337
Epoch 50/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5413 -
acc: 0.7331
Epoch 51/100
49000/49000 [=====] - 5s 95us/step - loss: 0.5415 -
acc: 0.7331
Epoch 52/100
49000/49000 [=====] - 5s 93us/step - loss: 0.5413 -
acc: 0.7332
Epoch 53/100
49000/49000 [=====] - 5s 93us/step - loss: 0.5414 -
acc: 0.7334
Epoch 54/100
49000/49000 [=====] - 5s 92us/step - loss: 0.5415 -
acc: 0.7333
Epoch 55/100
49000/49000 [=====] - 5s 93us/step - loss: 0.5410 -
acc: 0.7337
Epoch 56/100
49000/49000 [=====] - 5s 95us/step - loss: 0.5414 -
acc: 0.7332
Epoch 57/100
49000/49000 [=====] - 5s 103us/step - loss: 0.5414 -
acc: 0.7324
Epoch 58/100
49000/49000 [=====] - 5s 95us/step - loss: 0.5415 -
acc: 0.7326
Epoch 59/100
49000/49000 [=====] - 5s 103us/step - loss: 0.5413 -
acc: 0.7333
Epoch 60/100
49000/49000 [=====] - 5s 93us/step - loss: 0.5413 -
acc: 0.7337
Epoch 61/100
49000/49000 [=====] - 5s 99us/step - loss: 0.5415 -
```

```
acc: 0.7335
Epoch 62/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5414 -
acc: 0.7329
Epoch 63/100
49000/49000 [=====] - 4s 85us/step - loss: 0.5413 -
acc: 0.7323
Epoch 64/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5414 -
acc: 0.7336
Epoch 65/100
49000/49000 [=====] - 4s 81us/step - loss: 0.5414 -
acc: 0.7331
Epoch 66/100
49000/49000 [=====] - 4s 81us/step - loss: 0.5413 -
acc: 0.7326
Epoch 67/100
49000/49000 [=====] - 5s 94us/step - loss: 0.5412 -
acc: 0.7344
Epoch 68/100
49000/49000 [=====] - 5s 106us/step - loss: 0.5413 -
acc: 0.7330
Epoch 69/100
49000/49000 [=====] - 4s 84us/step - loss: 0.5410 -
acc: 0.7334
Epoch 70/100
49000/49000 [=====] - 5s 98us/step - loss: 0.5412 -
acc: 0.7336
Epoch 71/100
49000/49000 [=====] - 5s 95us/step - loss: 0.5412 -
acc: 0.7334
Epoch 72/100
49000/49000 [=====] - 5s 102us/step - loss: 0.5411 -
acc: 0.7346
Epoch 73/100
49000/49000 [=====] - 5s 102us/step - loss: 0.5410 -
acc: 0.7327
Epoch 74/100
49000/49000 [=====] - 5s 105us/step - loss: 0.5414 -
acc: 0.7328
Epoch 75/100
49000/49000 [=====] - 4s 86us/step - loss: 0.5411 -
acc: 0.7328
Epoch 76/100
49000/49000 [=====] - 5s 101us/step - loss: 0.5413 -
acc: 0.7333
Epoch 77/100
49000/49000 [=====] - 5s 96us/step - loss: 0.5412 -
```

```
acc: 0.7328
Epoch 78/100
49000/49000 [=====] - 5s 105us/step - loss: 0.5410 -
acc: 0.7332
Epoch 79/100
49000/49000 [=====] - 5s 92us/step - loss: 0.5410 -
acc: 0.7345
Epoch 80/100
49000/49000 [=====] - 4s 91us/step - loss: 0.5413 -
acc: 0.7329
Epoch 81/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5411 -
acc: 0.7325
Epoch 82/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5411 -
acc: 0.7333
Epoch 83/100
49000/49000 [=====] - 5s 104us/step - loss: 0.5412 -
acc: 0.7342
Epoch 84/100
49000/49000 [=====] - 4s 89us/step - loss: 0.5409 -
acc: 0.7341
Epoch 85/100
49000/49000 [=====] - 4s 89us/step - loss: 0.5413 -
acc: 0.7331
Epoch 86/100
49000/49000 [=====] - 6s 119us/step - loss: 0.5410 -
acc: 0.7343
Epoch 87/100
49000/49000 [=====] - 5s 106us/step - loss: 0.5409 -
acc: 0.7342
Epoch 88/100
49000/49000 [=====] - 6s 112us/step - loss: 0.5410 -
acc: 0.7341
Epoch 89/100
49000/49000 [=====] - 6s 112us/step - loss: 0.5413 -
acc: 0.7339
Epoch 90/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5412 -
acc: 0.7327
Epoch 91/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5411 -
acc: 0.7332
Epoch 92/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5411 -
acc: 0.7340
Epoch 93/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5410 -
```

```
acc: 0.7345
Epoch 94/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5410 -
acc: 0.7336
Epoch 95/100
49000/49000 [=====] - 4s 84us/step - loss: 0.5409 -
acc: 0.7333
Epoch 96/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5410 -
acc: 0.7331
Epoch 97/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5410 -
acc: 0.7339
Epoch 98/100
49000/49000 [=====] - 4s 83us/step - loss: 0.5410 -
acc: 0.7334
Epoch 99/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5410 -
acc: 0.7354
Epoch 100/100
49000/49000 [=====] - 4s 82us/step - loss: 0.5410 -
acc: 0.7344
```

[62]: <keras.callbacks.History at 0x1a879c7b70>

## 0.9 Model evaluation

```
[63]: eval_model=classifier.evaluate(X_train, y_train)
eval_model
```

```
49000/49000 [=====] - 0s 9us/step
```

[63]: [0.5403157654392476, 0.7336938775510204]

## 0.10 Predict cardiovascular disease

```
[64]: y_pred=classifier.predict(X_test)
y_pred =(y_pred>0.5)
```

```
[65]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[8056 2524]
 [3072 7348]]
```

0.11 total richtig/positiv falsch/negativ:  $8056 + 7348 = 15404$

0.12 insgesamt: 21000

0.13 accuracy:  $100 / 21000 * 15404 = 73,35 \%$

0.14 With the given inputs we can predict with a 73% accuracy if the person will suffer from cardiovascular disease or not