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

List of Figures

List of Tables

1 Abstract

Business Case: 1) With the developed solution, doctors are getting an overview of the patient's blood pressure values. This makes them react more precisely to any special values. 2) Since the patient is lead through a tutorial and the chatbot is 'controlling'/'checking' his measured blood pressure values, all measurements are taken more accurately. This improves the process of documentation. 3) With the service of sending a report via email every two weeks to the doctor. The doctor has always the current value and can interprete them faster. (now they are only getting a long list of all measured values of their patients which they have to interprete on their own and

4) Recommendation system of the nearest doctor helps the patient to directly go to his doctor Jack: 'Hey, i found these nearest doctors/specialists in your neighbourhood. Just select one of them and make an appointment with them.

2 Introduction

2.1 Problem statement

2.2 Aim and scope of this work

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 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.

3 Fundamentals

3.1 Reporting and Big Data

3.2 Software Architecture: Best Practices

To describe the 'best practices' of software architecture, in this section 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. As stated by Talukder et al.[1], this model is an appropriate solution for health applications because they support multiple stakeholders.

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. For instance, these information can be handwritten human readable and human understandable medical notes. 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. These data will be examined by a panel of experts to reach a consensus Proof of Disease (POC) and include 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.

According to Talukder et al.[2], there are three different types of mining:

[1] cf.[Talukder et al. 2018]

[2] cf.[Talukder et al. 2018]

- 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 iare loaded. As described by Talukder et al.[3], 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 (cf. figure 3.1 Ethereum).

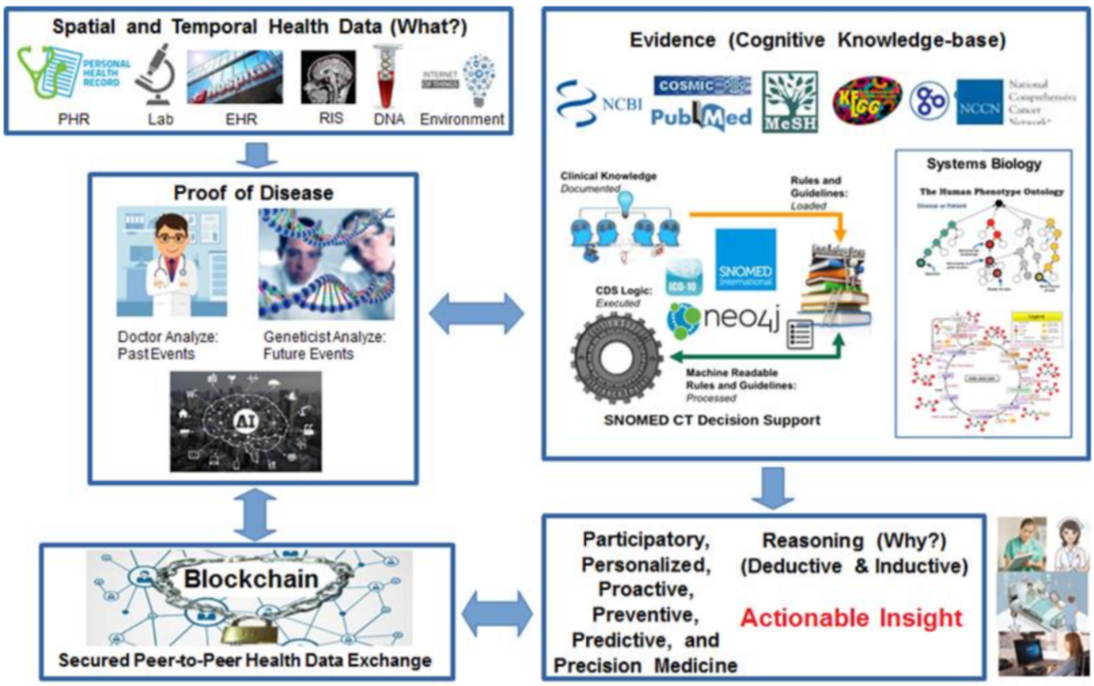


Figure 3.1: Example software architecture cf.[Talukder et al. 2018]

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 that, all codes are stored into a smart contract. During that process, a medical expert validates whether current the onset matches any clinical pathway. Finally, medical

[3] cf.[Talukder et al. 2018]

experts discuss in a proper medical consensus if the data is useful for an accurate diagnosis and public health.

3.2.1 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[4]. 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.[5]. 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.

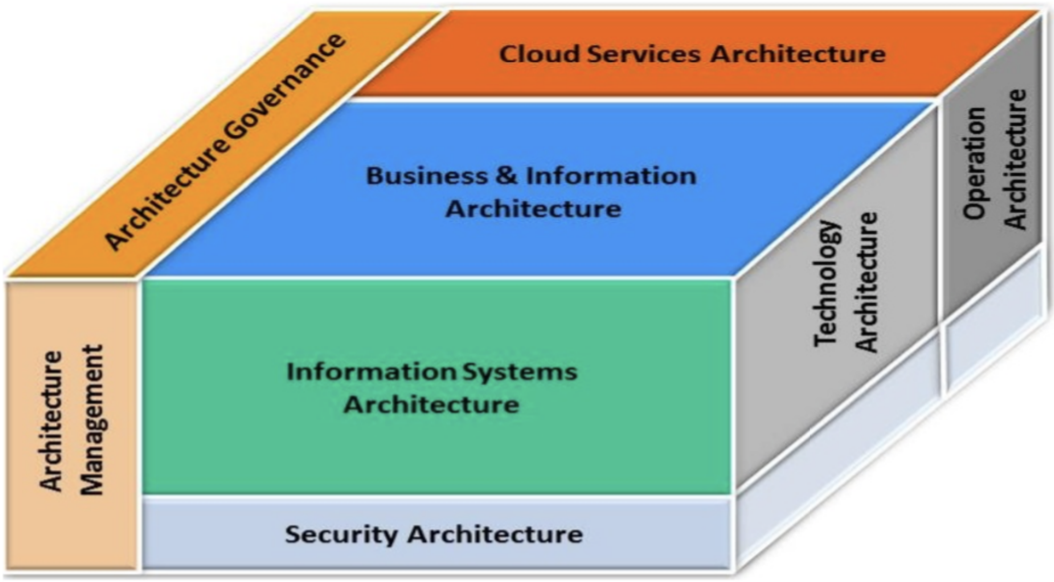


Figure 3.2: ESARC as an example for big data architecture cf.[Zimmermann et al. 2013]

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 vocabulary for enterprise architectures. Zimmermann et al. recommend that operations of tasks and entity services should not have any knowledge about their process

[4]cf.[Zimmermann et al. 2013]
[5]cf.[Zimmermann et al. 2013]

or interactive usage context [6]. 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]

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 precuse, architecture governance has the goal to manage activities such as plan, define, enable, measure, control and sets rules for architecture complicance 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 [7]. 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) and The Open Group Architecture Framework (TOGAF) standards are integrated.

[6] cf.[Zimmermann et al. 2013]
[7] cf.[Zimmermann et al. 2013]

3.3 Medical Documentation Apps

3.3.1 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[8]. This means that medical self-management applications should be supported by theory-based interventions which allow the identification of 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 a significant number of applications support self-management of hypertension with similar functionalities[9]. Besides these findings, Alessa et al. state that privacy and security is an important issue in many health applications and that these are not available in 35% of all applications. Moreover, it should be ensured that users are able to make fully informaed decisions by equipping the applications with skills and information necessary to scrutinize the privacy and

[8] cf.[Alessa et al. 2019]
[9] cf.[Alessa et al. 2019]

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[10].

3.3.2 Blood pressure monitoring in cardiovascular medicine and therapeutics

As mentioned above, many medical applications are developed for self-monitoring[11]. 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 [12]

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[13]. Environmental factors mean that the blood pressure values depend on winter or summer term. 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 exercice dicrease the blood pressure measurements[14].

[10] cf.[Alessa et al. 2019]
[11] cf. p.4ff.[White 2007]
[12] cf. p.23ff.[White 2007]
[13] cf. p.9ff.[White 2007]
[14] cf. p.9ff.[White 2007]

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 [15]. 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.

3.3.3 Social web and use cases for medical apps

As stated by Lupton et al.[16] 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.[17], 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[18] 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[19]. 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 patients directly to remind them to adhere to their tratment programs[20]. A further example of medical applications are 'smart pillboxes' for patients suffering from di-

[15]cf. p.31ff.[White 2007]
[16]cf.[Lupton 2014]
[17]cf.[Lupton 2014]
[18]cf.[Lupton 2014]
[19]cf.[Lupton 2014]
[20]cf.[Lupton 2012]

abetes[21]. '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[22]. 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[23]. 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 erroneos or unwanted[24]. The term NCD implicates disease caused by an unheallthy 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
- 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

[21] cf.[Lupton 2012]
[22] cf.[Lupton 2012]
[23] cf.[Lupton 2012]
[24] cf.[Talukder et al. 2018]

3.4 Chatbots

3.4.1 The potential of chatbots

Modelling, profiling, analyzing and understanding users becomes increasingly important in many different indrustries and count as key to success in todays data driven world. The main advantage of chatbots is to provide a 24-hour customer service with personalized interaction and no waiting time[25]. 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.

Question Answering Paradigms There are several types of conversations which can be designed by building a chatbot[26]. 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

[25] cf.[Akhtar, Neidhardt, and Werthner 2019]
[26] cf.[Akhtar, Neidhardt, and Werthner 2019]

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[27].

3.4.2 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[28]. 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 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.

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 e,bedding models. Additionally, a question and answering corpus should be created for OSCEs.

[27]cf.[Akhtar, Neidhardt, and Werthner 2019]
[28]cf.[Zini et al. 2019]

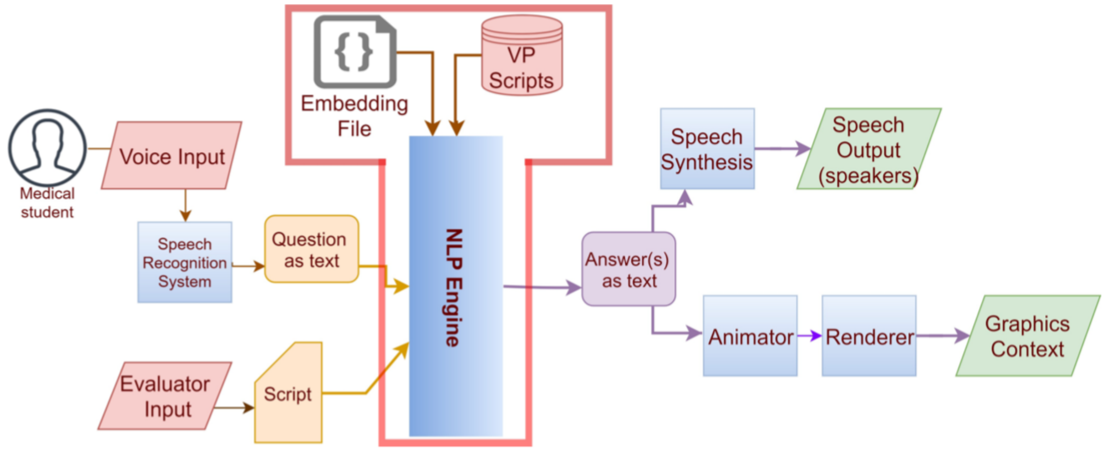


Figure 3.4: Virtual patient software architecture cf.[Zini et al. 2019]

Question and Answering systems According to Zini et al. there are two types of question and answering systems[29]. 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.

[29] cf.[Zini et al. 2019]

4 Analysis and Development

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 so that he can get more insights about the blood pressure values of his patients and 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 more intelligent 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).

4.1 Experimental set-up

4.1.1 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.

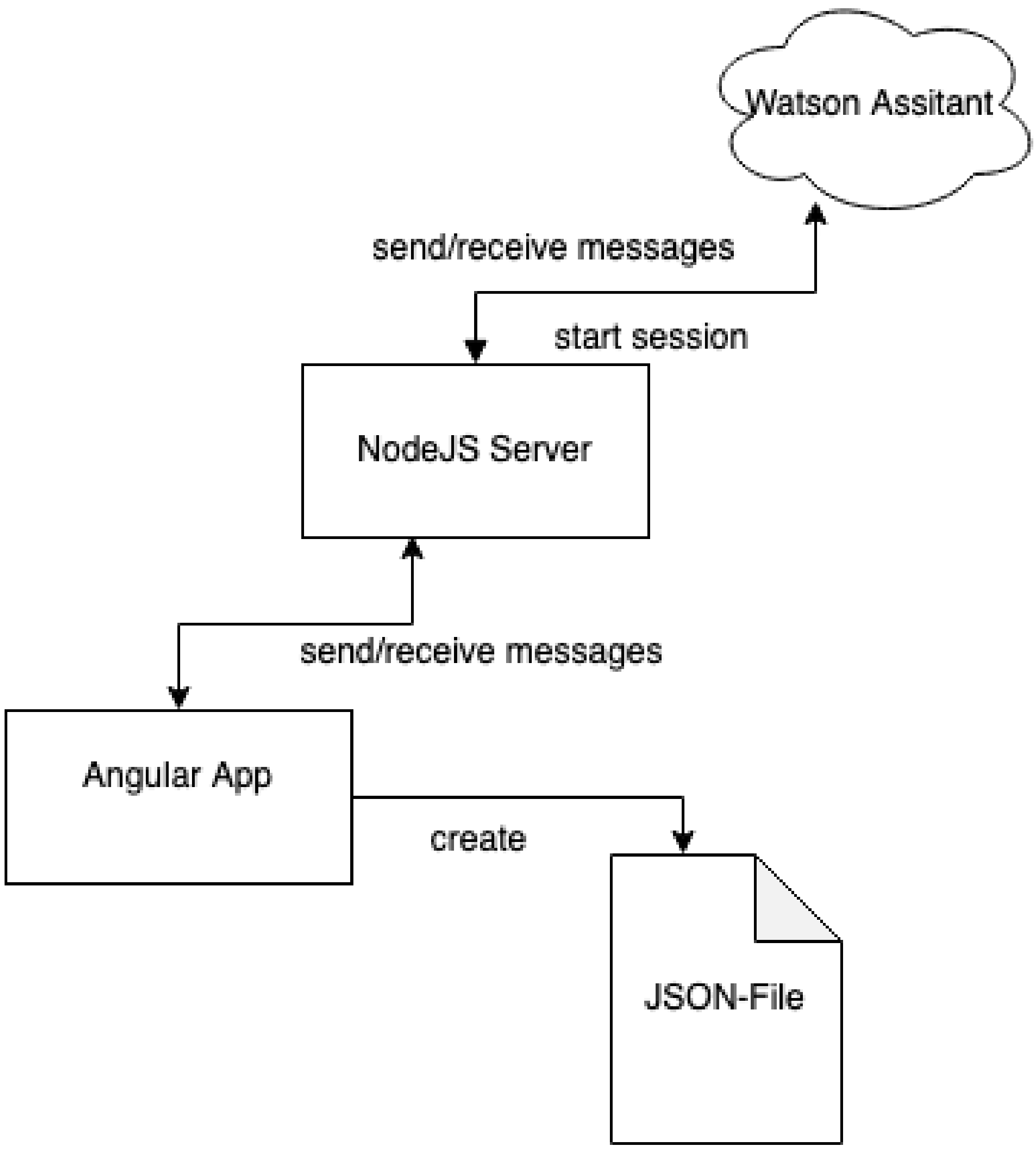


Figure 4.1: Architecture diagram of developed system

Development of Watson Assistant Dialog

Intent model The chatbot was built according to the webpage of "Deutsche Hochdruck-liga", a german organization for patients who suffer from hypertonia[1]. 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 hypertonia	What is hypertonia?
Curses of hypertonia	What are the implications or curses of hypertonia?
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 4.2, 4.3, 4.4 and 4.5.

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 [2].

Setup of AngularJS 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. Three open source ibraries, such as Nebular[3], Apache Echarts[4] and Openlayers Maps[5] were included to the application. 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. 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

[1] cf.[*Bluthochdruck vermeiden, behandeln und senken - Aktiv gegen Bluthochdruck* 2019]
[2] cf.[*Übersicht über die Watson Assistant-API* 2020]
[3] cf.[**nebular**]
[4] cf.[**echarts**]
[5] cf.[**openlayers**]

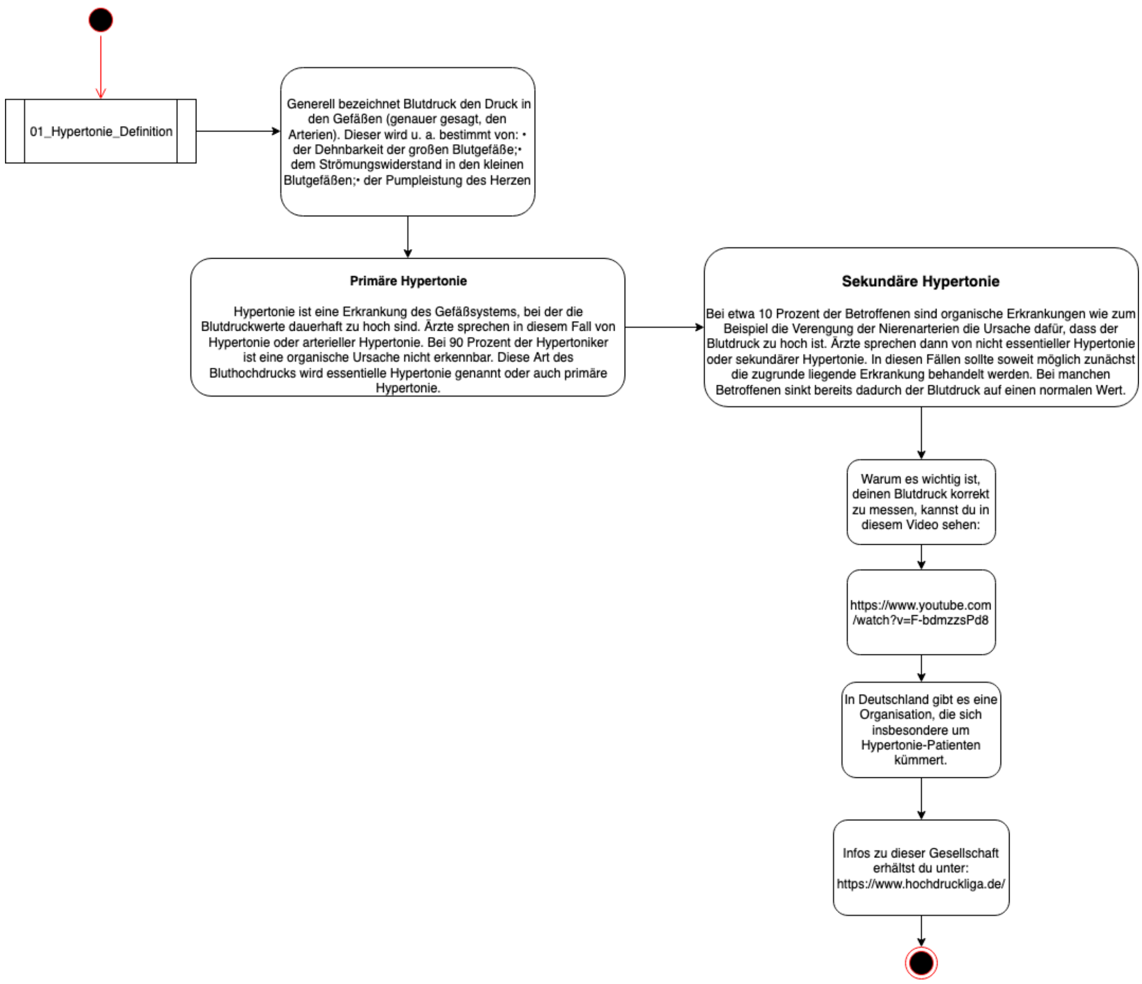


Figure 4.2: Dialog diagram: definition of hypertonia

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. But for future use cases, it might be also useful to display other charts of Echarts or maybe tables. The third Javascript library that was used, is called Openlayers Maps. It is very famous and many projects rather use Openlayers API than Google Places API because their API calls are very expensive. Openlayers Maps has many different modes and overlays. Certain places can be marked by different icons. All in all, the three described libraries were 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.

Connect Watson Assistant to Frontend difficul

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 has to be requested to be able

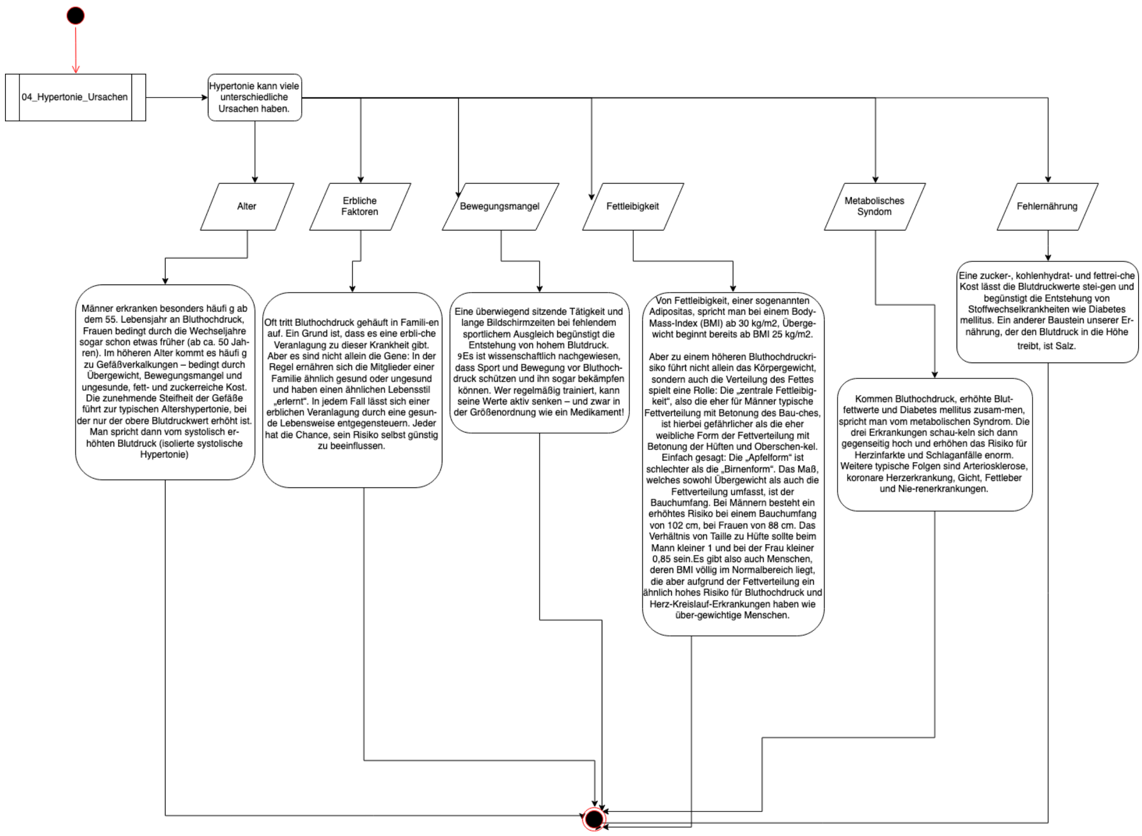


Figure 4.3: Dialog diagram: curses of hypertonia

to interact with Watson Assistant. After that, a simple get request is 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 and the response is the best fitting answer (with the highest confidence value) from Watson, which is calculated [6].

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

7 8

Development of recommendation of nearest doctors to patient outlook: maybe in future to connect to the doctors' calendar to directly make an appointment through the chatbot

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.

[6]cf.[*Watson Assistant v2 - IBM Cloud API Docs* 2020]
[7]cf.[*Cardiovascular Disease dataset* 2020]
[8]cf.[*Decision Tree Classification in Python* 2018]

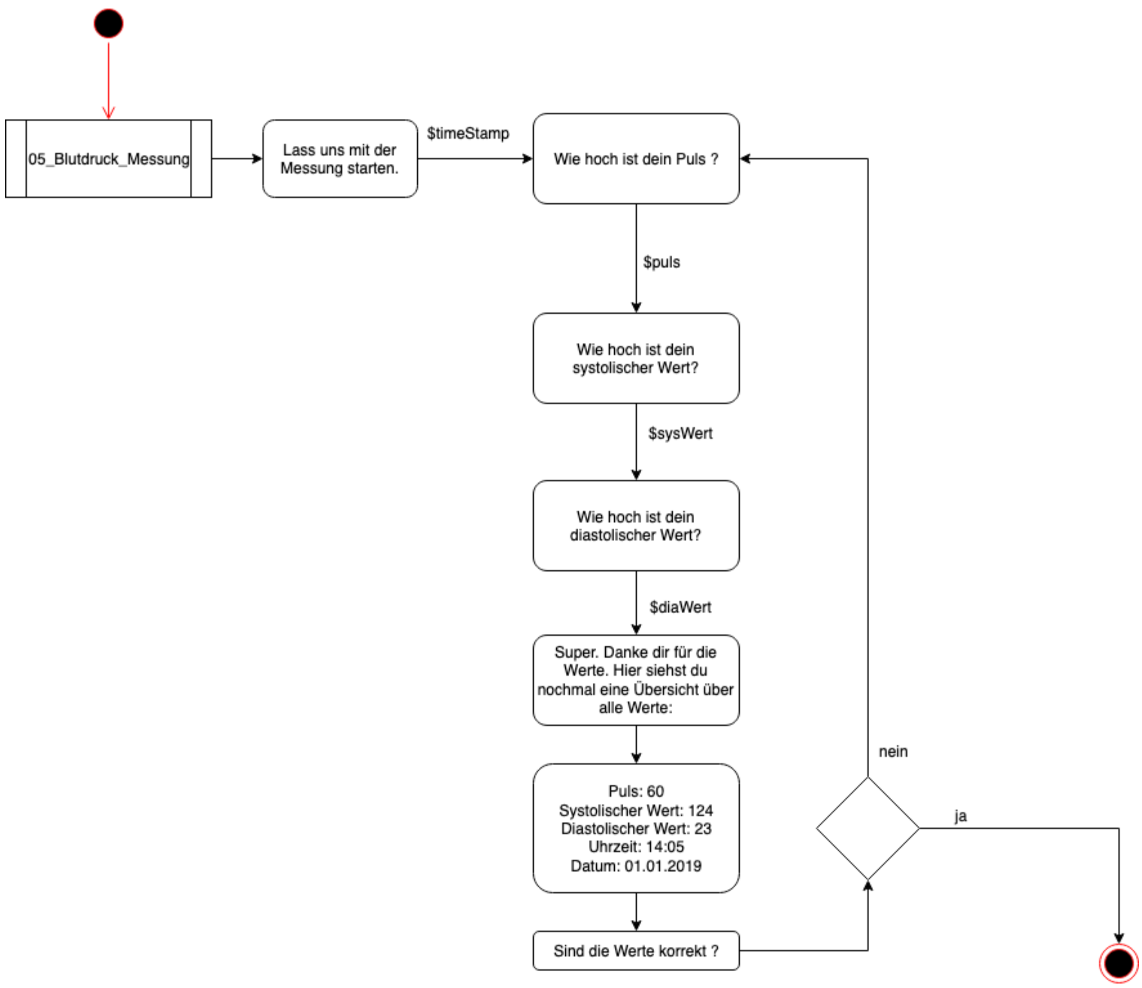


Figure 4.4: Dialog diagram: blood pressure measurement

A 'usual' chatbot only helps in certain situations including precise user intents, e.g. the question 'When should i measure my blood pressure?'. But they are not constructed to ping a user every five hours or once a day in order to retrieve his measured data, analyze these and send them to a doctor. In order to face this problem or use case, a routine including a timer had to be implemented. For future use cases, it might be very useful to have a proper dashboard for each doctor (which requires a user management and authentication mechanism)

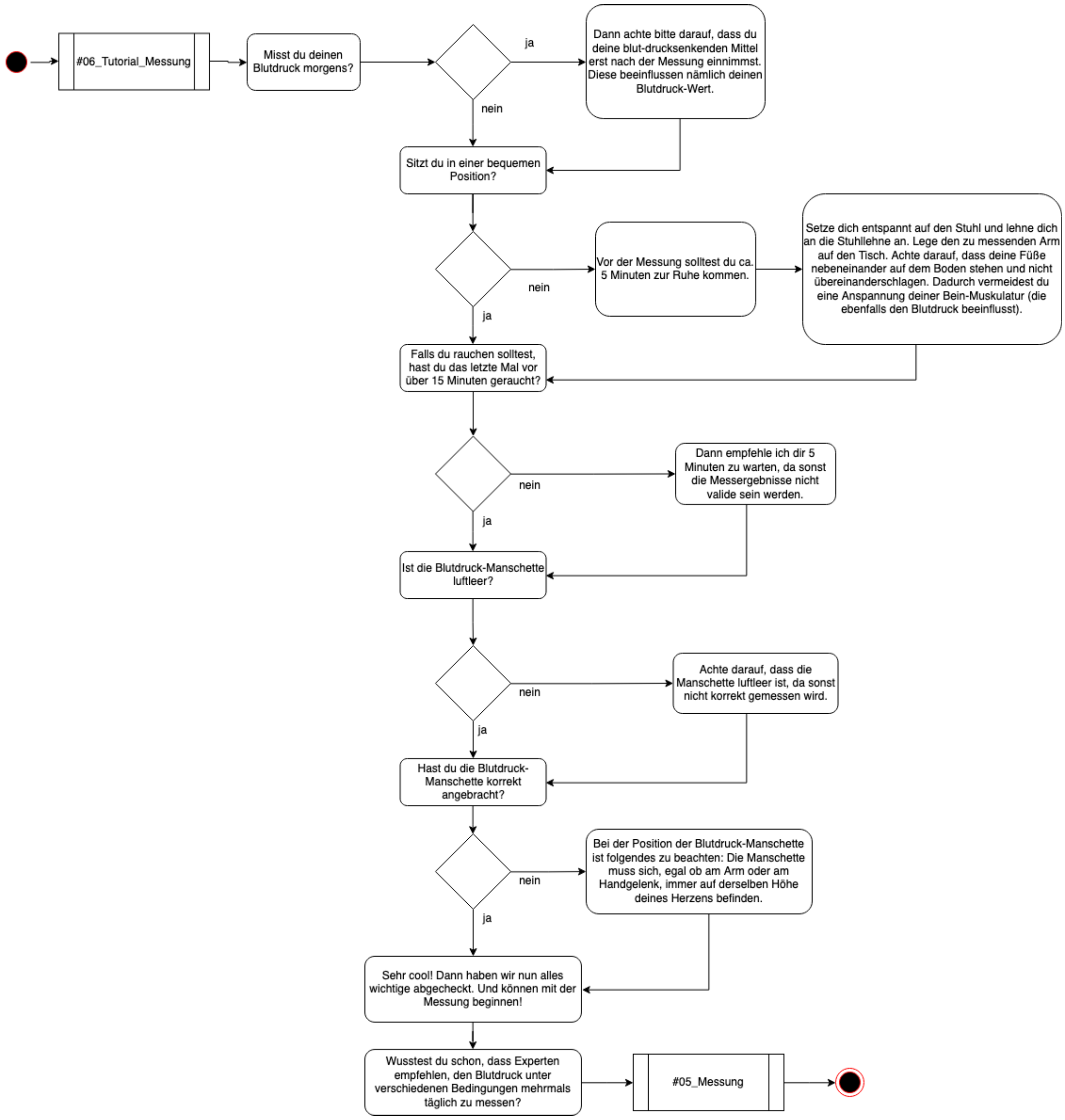


Figure 4.5: Dialog diagram: measurement tutorial

4.2 Problem solving

4.2.1 Tests

4.2.2 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[9] which included 70000 entries of different patients and 12 characteristics. These 12 characteristics included:

[9]cf.[*Cardiovascular Disease dataset* 2020]

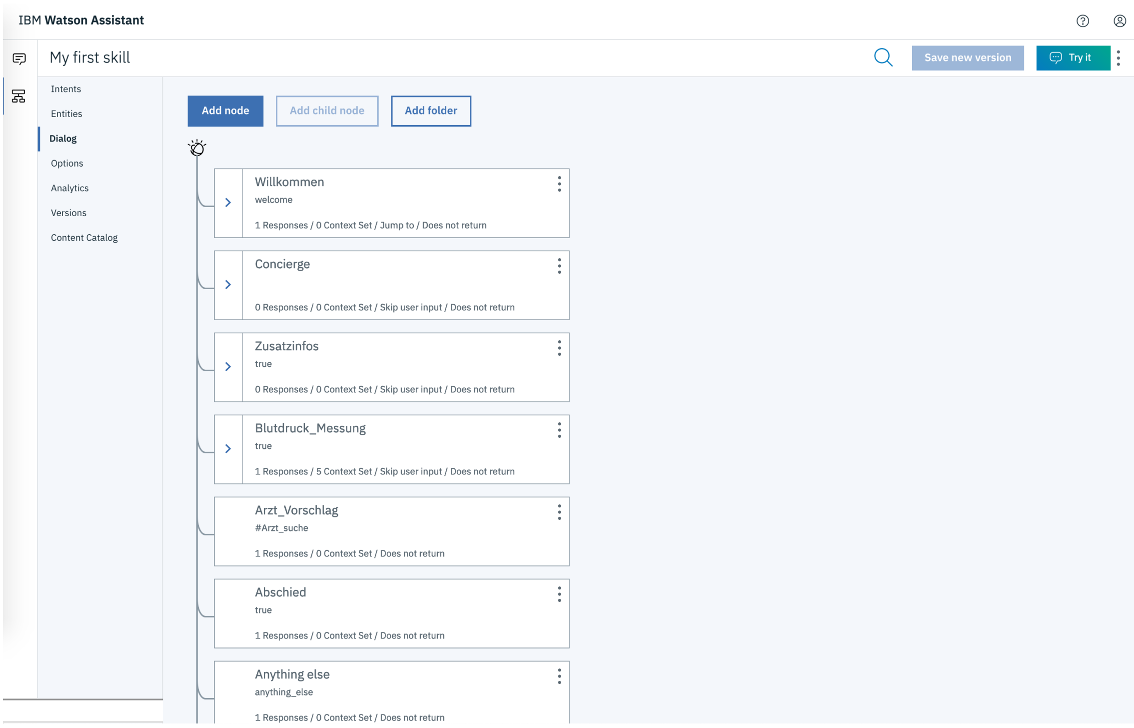


Figure 4.6: Watson Assistant dialog

- 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)
- active (binary)
- presence or absence of cardiovascular disease (binary)

4.3 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

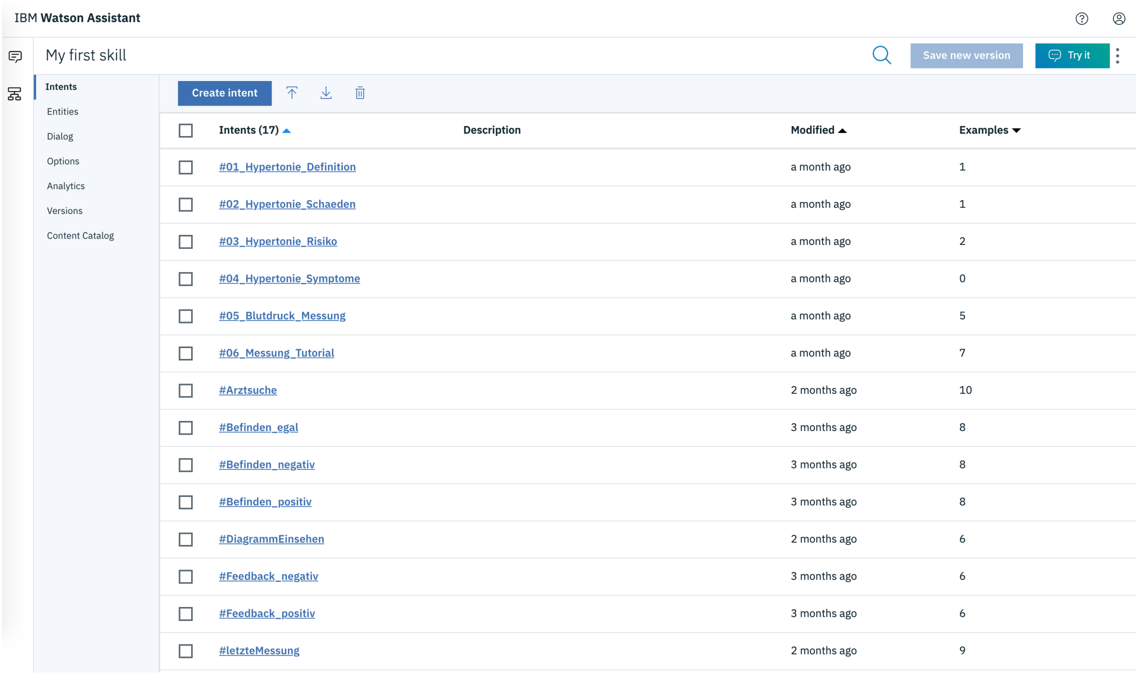


Figure 4.7: Watson Assistant intents

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.

4.3.1 Development of python script

Based on the given Kaggle Dataset, a python script 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 during the first step of analysis the dataset had to be cleaned from null values and only the factors which are relevant for correlation analysis were used. All other values were eliminated. This was implemented by using seaborn's function 'heatmap'. The produced heatmap showed the parameters of the dataset and how they correlated with the classification variable 'cardio' (which means that the person suffers from cardiovascular disease or not).

As this is a binary classification problem sigmoid as the activation function was used. Dense layer implements output = activation(dot(input, kernel) + bias). Kernel is the weight matrix. Kernel initialization defines the way to set the initial random weights

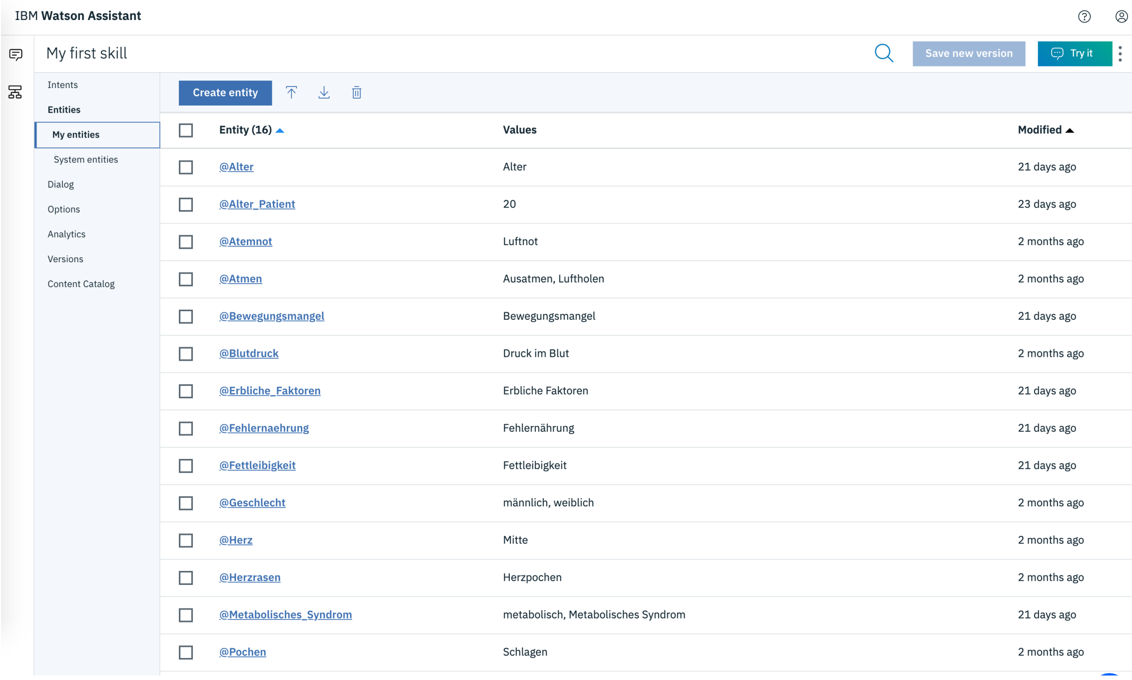


Figure 4.8: Watson Assistant entitities

of Keras layers. To optimize the neural network an Adam function was used. Adam stands for Adaptive moment estimation and combines RMSProp and Momentum. Momentum takes the past gradients into account in order to smooth out the gradient descent.

4.4 Results

5 Conclusion and Outlook

5.1 Conclusion

5.2 Outlook

5.2.1 Connect Flask app, python script and Angular frontend

Figure 5.1 shows the architecture which is aimed

Is a chatbot an appropriate solution for recording and reminding patients to measure their blood pressure ? In practice, most chatbots are created to solve and help multiple intents of their users and not to only 'retrieve' information. On the one hand, the retrieved information can be used to run several analysis and to find out trends in the data. But on the other hand, 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.

connection frontend (angularjs webpage) to mongodb through socket.io connection. Connection between Watson Assistant and Frontend through mongodb and via socket.io connection.

5.2.2 Behaviour change techniques

5.2.3 Therapy and Forecasting of healing

As described in section 4.3, 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, It could provide personlaized nutrition plans. It could also forecast the term or maybe exact date (if trained well) when the patients will be healthy again and

5.2.4 Encrypt and save patient data

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.

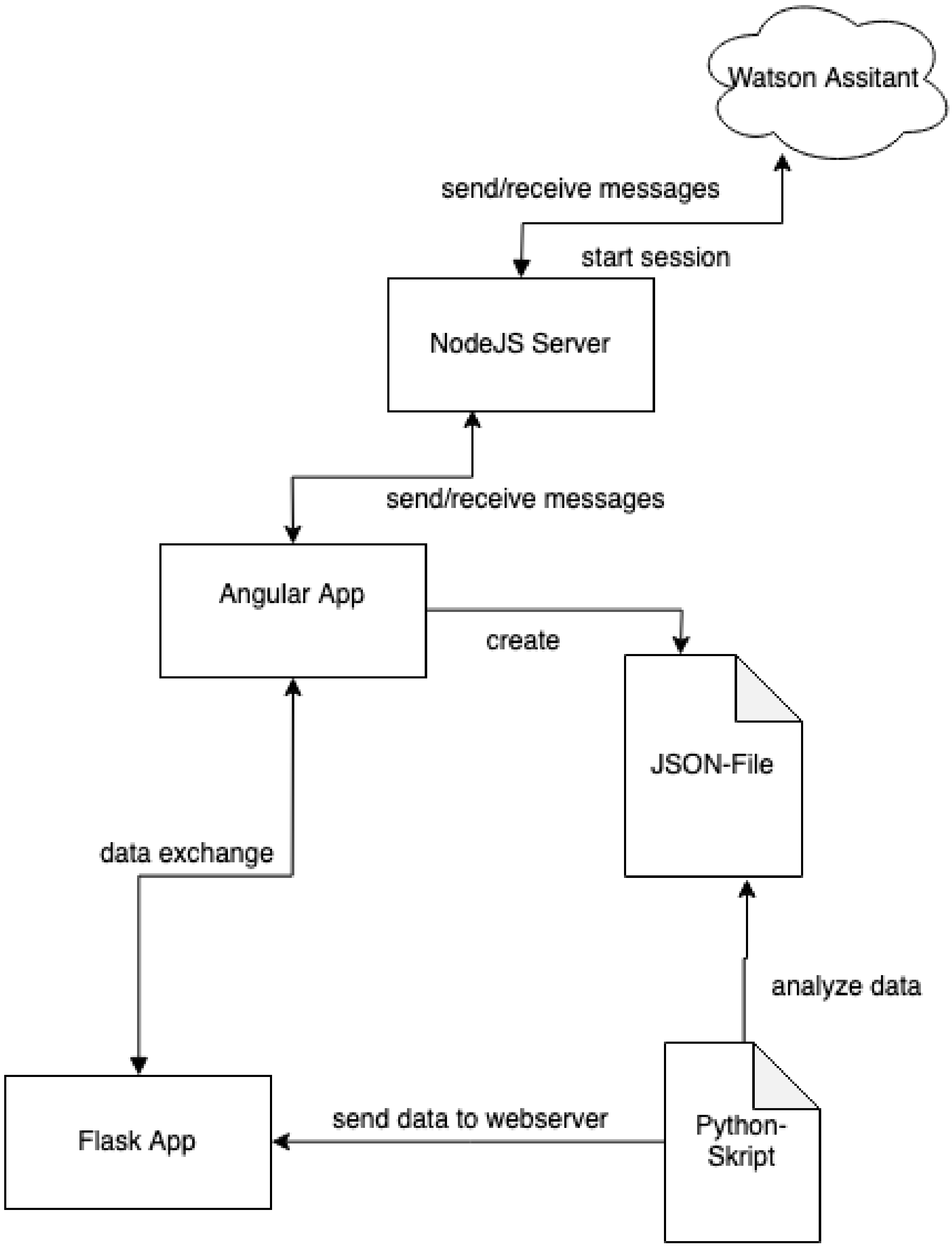


Figure 5.1: Watson Assistant entitities

6 Abbreviations

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

HCP	Health Care Provider
TOGAF	The Open Group Architecture Framework
BCT	Behavioural Change Techniques
HTTP	Hypertext Transfer Protocol

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).

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).

Decision Tree Classification in Python (Dec. 2018). URL: https://www.datacamp.com/community/tutorials/decision-tree-classification-python (visited on 01/19/2020).

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).

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.

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 cardiovsacular 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.

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.

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)
```

[31]:

	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

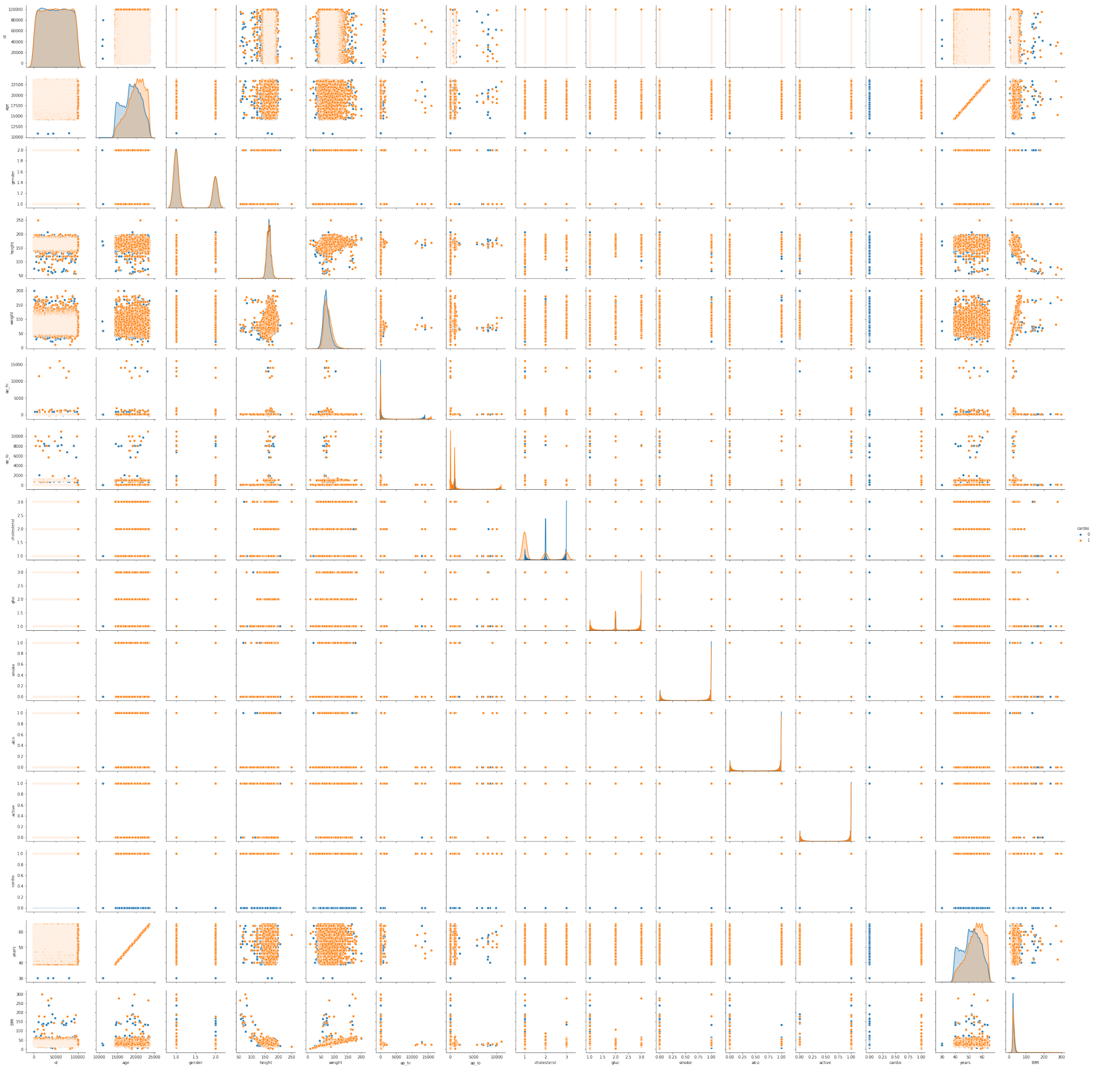
```
69991      56  28.479886
69992      51  21.604105
69993      54  23.661439
69994      58  29.384757
69995      53  26.927438
69996      62  50.472681
69997      52  31.353579
69998      61  27.099251
69999      56  24.913495

[70000 rows x 11 columns]
```

0.4 plot data (pairplot and heatmap for correlation)

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

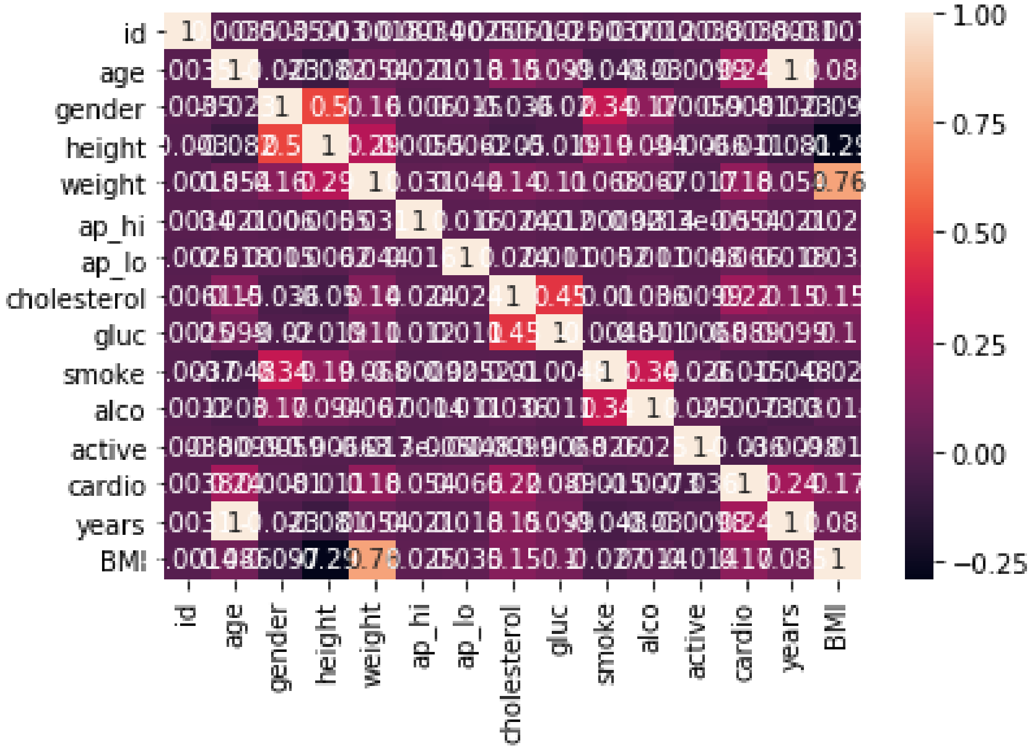
```
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:488:
RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kdetools.py:34:
RuntimeWarning: invalid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

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



[33]:
```
sns.heatmap(dataset.corr(), annot=True)
```

[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1a7f12f978>



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','a
↪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