dHealth 2022
G. Schreier et al. (Eds.)

doi:10.3233/SHTI220379

Development of an Architecture to Implement Machine Learning Based Risk Prediction in Clinical Routine: A Service-Oriented Approach

Michael SCHREMPF[a,b,1], Seda POLAT ERDENIZ[a], Diether KRAMER[a], Stefanie JAUK[a], Sai P. K. VEERANKI[a], Werner RIBITSCH[a,b], Werner LEODOLTER[a,b] and Peter P. RAINER[b]
[a] *Steiermärkische Krankenanstaltengesellschaft m.b.H., Graz, Austria*
[b] *Medical University of Graz, Graz, Austria*

Abstract. *Background*: Patients at risk of developing a disease have to be identified at an early stage to enable prevention. One way of early detection is the use of machine learning based prediction models trained on electronic health records. *Objectives*: The aim of this project was to develop a software solution to predict cardiovascular and nephrological events using machine learning models. In addition, a risk verification interface for health care professionals was established. *Methods*: In order to meet the requirements, different tools were analysed. Based on this, a software architecture was created, which was designed to be as modular as possible. *Results*: A software was realised that is able to automatically calculate and display risks using machine learning models. Furthermore, predictions can be verified via an interface adapted to the need of health care professionals, which shows data required for prediction. *Conclusion*: Due to the modularised software architecture and the status-based calculation process, different technologies could be applied. This facilitates the installation of the software at multiple health care providers, for which adjustments need to be carried out at one part of the software only.

Keywords. Public Health, Healthcare, prevention & control, Software Engineering, Computer Software, Health Risk Assessment, Machine Learning, Preventive Medicine, Preventive Health Services

1. Introduction

Many machine learning models have been published in the field of medicine over the last years [1]. However, few of these models have been implemented within a clinical setting to prove their benefit for health care [2], [3].

Within the scope of the project *Prevention Support Tool*[2], several machine learning models have been developed to predict severe cardiovascular events [4] and chronic kidney disease. The aim of the project is to reduce the burden of atherosclerotic

[1] Corresponding Author: Michael Schrempf, Steiermärkische Krankenanstaltengesellschaft m.b.H., Graz, Austria, E-Mail: michael.schrempf@kages.at

[2] Funded by the Styrian Health Fund (Gesundheitsfonds Steiermark)

cardiovascular diseases [5] and to avoid kidney transplantation or the need of dialysis, and thus to improve the quality of life of patients.

The predictive models have been trained on the electronic health records (EHR) of KAGes, the Styrian Hospitals Limited Holding (Steiermärkische Krankenanstaltengesellschaft m. b. H.). KAGes provides approximately 90% of all acute care beds besides of many ambulatory services in the province of Styria, Austria, and hosts EHRs of over two million patients.

One aim of the project was to develop the front-end and back-end for a software to be integrated in an hospital information system (HIS), of KAGes. The goal was to visualize the results of the machine learning models in a user friendly way and to facilitate the decision making of clinicians. When using the app, clinicians should be able to verify a risk prediction and receive more information about EHR data used for prediction.

2. Methods

The aim of this project was to develop a risk prediction software to support the prevention of clinical events. In order to achieve this objective, we wanted to develop a software which is able to communicate with a HIS. After gathering user requirements, an architecture for risk prediction, communication and visualisation was developed. In the case of using predictive models trained in different programming languages, the architecture had to be modular. Another challenge was portability, as the aim was to develop a software that can easily be installed at other health care providers and different HIS. To increase confidence into the predicted risks, a front-end for clinicians had to be designed and developed.

2.1. Architecture Design Process

For the architecture, the focus was on developing a system for automated risk calculation and a dashboard for risk verification.

Regarding the dashboard, we used a framework that was published for the development of a front-end. Such frameworks contain many functionalities that significantly accelerate development. Besides the development, the single-page concept also optimises the efficiency of the display and data transfer between client and server [6]. Based on the competencies, we chose AngularJS [7] instead of Vue.js [8] and ReactJS [9] to develop the dashboard.

In order to provide data for the user interface, a framework for creating an application programming interface (API) according to the OpenAPI Specification (OAS) [10] was selected. The OAS defines the structure of the API endpoints and responses. Common API frameworks are Django [11] and Spring [12]. To decide on an API framework, we defined several requirements. For example, the API should be able to communicate directly with various database systems without a need of additional implementation or changes in the backend software and to publish a RESTful web service. A RESTful web service means that the defined rules of the REST paradigm are followed by the service [13]. One of these rules is stateless transmission, which means that all information in the message is available for processing by the receiver. For the project, the API should follow these REST rules, be accessible via HTTP and exchange

data in the form of JSON [14] documents. Based on these requirements we have chosen Spring as an API framework.

However, another solution was required for retrieving data from the HIS, as data processing in Java [15] is more complicated and offers fewer possibilities compared to R [16]. That is why Plumber API [17] was introduced, which is a framework for developing an API service with R.

However, as the models were trained in Python [18] using Scikit-learn [19], a Python solution was needed as well. To keep the effort as low as possible, the framework Flask have been used. Flask is ideal for small applications because the HTTP endpoints can be created with little effort.

2.2. Data Model Development

In order to create a standardised data structure, an entity relationship (ER) diagram was created before the source code of the services was implemented. The Java-based model classes were created based on this ER diagram. For the ER diagram, FHIR [20] resources were consulted. For the implementation of a prediction system, the data models of the resources *observation* and *risk assessment* were adopted. FHIR was designed to facilitate the communication between different information systems.

2.3. Calculation Process

In previous projects, risk prediction was completely implemented in an R software solution. The advantage of this solution was that the entire business logic was combined in one project. As a result, the further development of the project could be easily arranged. Another advantage concerns the installation, as only R and the necessary packages have to be provided. However, there are some disadvantages as well. For example, adaptations regarding changes in the process can be difficult to implement. The integration of Python-based models needs also some effort. To overcome these disadvantages, status-based processing was planned. A status was defined for each step towards a risk prediction. If new steps are required due to changes in the process, new states have to be inserted.

2.4. Dashboard Development

Besides the automated risk calculation, another aim of the project was to develop a web-based interface to review the prediction. The interface should display the risk calculation and the underlying data in a comprehensive way. To ensure that the application is as user-friendly as possible, a design workshop was organised with different user groups. In this workshop, the functional requirements and a prototype were designed.

In addition, interviews were conducted with physicians in order to capture the requirements for the interface.

Based on the results of the design workshop and interviews, mock-ups were prepared, which were improved in cycles within the development team. The first mock-up also triggered the development of the front-end based on AngularJS.

3. Results

For providing risk predictions to support disease prevention, an architecture and calculation process were defined. In addition, a dashboard was developed for increasing the confidence of the predicted risks. The details of these developments are demonstrated in the following sections.

3.1. Architecture

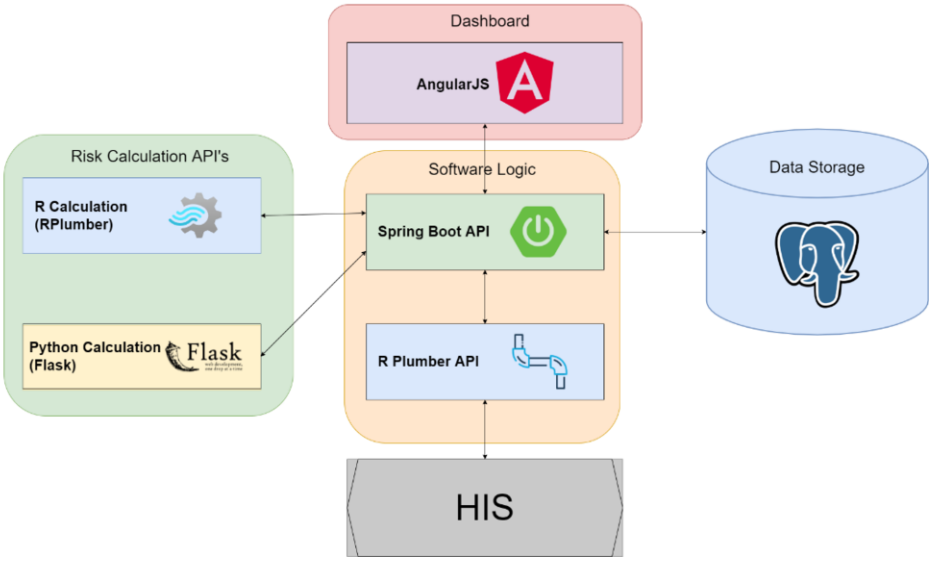

Figure 1: Software Architecture for the Project *Prevention Support Tool*

The different technologies described in section 2.1 were combined to fulfil the objectives of the software. Figure 1 shows the final structure and components.

On the top of the architecture, AngularJS was used to display the risk and electronic health records from the HIS to the users. The data is retrieved from the Spring API using RESTful http queries.

The core of the software is the Spring API, which is the interface for all components to retrieve data. In general, all data is managed with this API service. Since the calculation processing is status-based, data must be stored. These data are stored in the database, which can only be accessed through the Spring API. To retrieve data from the database by the Spring API, communication is carried out via a JDBC (Java Database Connectivity) connection.

To retrieve the data from the HIS, the R Plumber API is used as an adapter. The raw data is requested from the system and linked to the metadata of the software. In the end, the data is converted into the defined data structure and transferred to the Spring API.

The trained models for risk calculation were integrated into the Flask API. If a risk calculation is carried out, a request is made for it by Spring API. The request includes an identification number that the Python script executed by the Flask API is able to retrieve the EHR data from the Spring API. Afterwards the data is transferred to the

corresponding model for prediction. Upon completion of the prediction process, the results are returned using a structured JSON document.

Finally, the Spring API was connected to a PostgreSQL [21] database using Hibernate [22], which stores all the data of the application. A big advantage of the Hibernate ORM (Object/Relational Mapping) is that the database can be changed at any time with little effort.

For the machine-to-machine communication, RESTful http requests were used. Except for the HIS connection, JSON documents were used to transfer the relevant content. For the HIS, the requests were responded using XML documents.

3.2. Calculation Process

In order to facilitate the deployment in clinical routine, risk predictions need to be calculated as early as possible during a patient's stay, but at least some hours before discharge. However, determining the exact time of discharge is not always feasible based on data from the HIS. Therefore, two time points were defined for risk prediction: (1) admission time and (2) 8 o'clock in the morning.

At admission, an HL7 [23] file is sent to the application. With an incoming file, a calculation object for the patient is created. This calculation object has a status to check how far the risk prediction for the respective patient is.

After creating the calculation object, the patient's information is loaded from the HIS using http requests. Once the data is prepared for prediction, the status of the linked calculation object is set to *extraction*. The status *extraction* tells the system to request EHR data for the respective patient from the HIS, including demographic data, diagnoses, procedures, medication, laboratory values etc. from previous hospital stays. To initiate the risk prediction process, the status of the calculation object is set to *calculation*.

For risk prediction, the patient's data is loaded by a script in the Flask API application and passed to the machine learning models for prediction. In the end, the status of all calculation objects is set to *sending*.

The risk predictions are then communicated to the HIS. The predicted risk is shown within the user interface of the HIS using traffic light symbols. A red symbol presents a high risk, and a yellow symbol a moderate risk. No symbol is shown for low risk patients in order to avoid an overload of information in the interface.

Finally, the calculation for every patient is completed, if the status is set to *finished*.

3.3. Dashboard

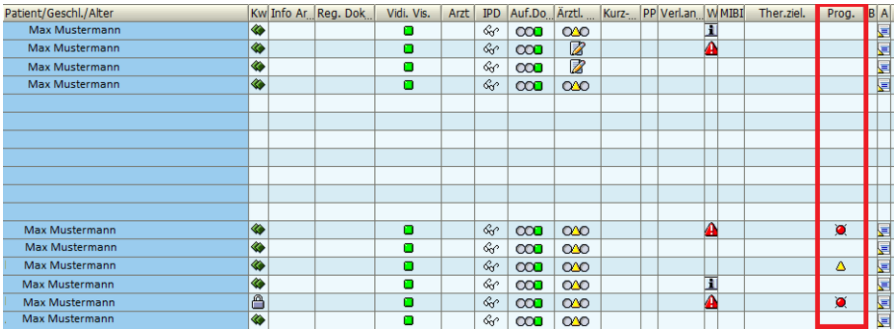

Figure 2: User interface for clinicians in the hospital information system of KAGes, including the column *Prog*. with the prediction results of the software.

As shown in Figure 2, red symbols in the column called "Prog." indicate patients with a high risk, and yellow symbols for moderate risk.

In order to verify the prediction, a click on the icon or the empty field opens the dashboard (Figure 3). Afterwards the overview page of the dashboard is displayed. This page shows coloured cards based on the traffic lights system. For each predicted clinical outcome, a card is shown for the patient.

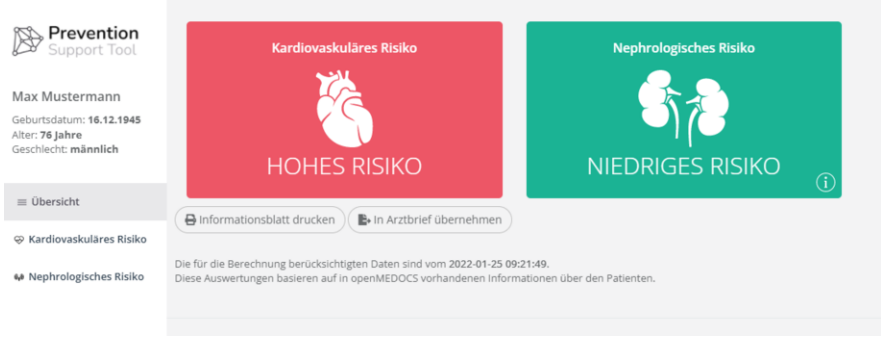

Figure 3: Dashboard of the software presenting an overview of the risk prediction.

In the example in Figure 3, the patient had a high risk for a major adverse cardiovascular disease within the following five years (e.g. myocardial infarction or stroke). However, the model predicted a low risk for a chronic kidney disease within the next year.

For more information, the user is able to click on one of these risk cards. By clicking, a risk bar and the most important predictors of the prediction are shown. Additionally, the patient history including diagnosis, laboratory values, medication and health care procedures with regard to the risk context is available within the dashboard.

For continuous improvement of the software tool, a feedback opportunity was added for the clinicians in order to report their clinical risk estimation or other comments.

4. Discussion

In this project, we developed a software to automatically predict clinical outcomes on the basis of EHR data. To enable health care professionals to verify the predictions, a dashboard was developed to display the data of risk prediction. This software solution aims to support the prevention of common diseases such as stroke, myocardial infarction or chronic kidney failure. With early detection of a patient at high risk for one of these diseases, targeted measures can be taken to prevent them [5].

For the prediction of cardiovascular diseases caused by atherosclerosis, several risk prediction tools are already available. However, for calculating risk scores like QRISK3 [24] or SCORE [25], data have to be entered manually via a web interface. Therefore, our aim was to create a solution, which avoids additional workload for the clinicians. Concerning the risk prediction, machine learning approaches have been used to enable better performances based on the available electronic health records [26].

Due to the use of multiple API frameworks, various strengths of the developed architecture can be pointed out. As the dashboard depends solely on the Spring API and the database, the other services can be started on demand, which could be useful if a containerised environment for managing resources is used.

The biggest advantage of this architecture is its modularisation. Changes can be implemented more flexibly. Side effects can also be better assessed and thus prevented.

Another advantage is that only the R Plumber API needs to be changed if the software tool will be deployed to other hospital providers. This leads to an easier way of further developments, because the services could be developed independently.

The main advantage of using the Spring API framework is the automated generation of HTTP endpoints and database schema when using model classes. Another advantage of the Spring API framework is the use of projections, which for example help to define the output of the JSON responses.

Another benefit of using a back-end solution like Spring API is that additional front-ends to manage data or to create software metric analysation dashboards can be developed in parallel to the software itself.

A future bottleneck within this architecture could be the R Plumber API service. R is a single-threaded language, which is leading to the problem that only one request can be treated by the Plumber application. To parallelise the requests, multiple Plumber back-ends have to be deployed. Containerised infrastructure or load balancing technologies like pm2 [26] can solve this issue. However, such technologies have not been implemented, as there was no lack of resources yet.

To ensure that the provided risk assessments are useful and reasonable, the physicians will be asked for their feedback to improve the quality of the models in a pilot evaluation study.

For further development, we plan to conduct usability tests to determine new requirements and user's satisfaction. This will ensure that the dashboard addresses the needs of clinicians. To avoid security issues, the used frameworks will be checked regularly for important updates. Additionally, a penetration test will be performed at least on a test system to check that the data managed by the *Prevention Support Tool* is secure.

A promising additional development of the software could be the integration of the ELGA framework. With the help of ELGA data, the predictions can be supplemented with missing data and the predictions can thus be improved. Furthermore, future research should evaluate whether the tool can also be offered to general practitioners in order to provide support outside of hospitals and further increase preventive actions for patients.

References

[1] D. Ben-Israel *et al.*, 'The impact of machine learning on patient care: A systematic review', *Artif. Intell. Med.*, vol. **103**, p. 101785, Mar. 2020, doi: 10.1016/j.artmed.2019.101785.

[2] T. C. Lee, N. U. Shah, A. Haack, and S. L. Baxter, 'Clinical Implementation of Predictive Models Embedded within Electronic Health Record Systems: A Systematic Review', *Informatics*, vol. **7**, no. 3, p. 25, Jul. 2020, doi: 10.3390/informatics7030025.

[3] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, and K. Zhang, 'The practical implementation of artificial intelligence technologies in medicine', *Nat. Med.*, vol. **25**, no. 1, pp. 30–36, Jan. 2019, doi: 10.1038/s41591-018-0307-0.

[4] M. Schrempf, D. Kramer, S. Jauk, S. P. Veeranki, W. Leodolter, and P. Rainer, 'Machine Learning Based Risk Prediction for Major Adverse Cardiovascular Events', in *Studies in health technology and informatics*, vol. **279**, 2021. doi: 10.3233/SHTI210100.

[5] M. F. Piepoli *et al.*, '2016 European Guidelines on cardiovascular disease prevention in clinical practice: The Sixth Joint Task Force of the European Society of Cardiology and Other Societies on Cardiovascular Disease Prevention in Clinical Practice (constituted by representatives of 10 societies and by invited experts)Developed with the special contribution of the European Association for Cardiovascular Prevention & Rehabilitation (EACPR)', *Eur. Heart J.*, vol. **37**, no. 29, pp. 2315–2381, Aug. 2016, doi: 10.1093/eurheartj/ehw106.

[6] M. A. Jadhav, B. R. Sawant, and A. Deshmukh, 'Single page application using angularjs', *Int. J. Comput. Sci. Inf. Technol.*, vol. **6**, no. 3, pp. 2876–2879, 2015.

[7] N. Jain, A. Bhansali, and D. Mehta, 'AngularJS: A modern MVC framework in JavaScript', *J. Glob. Res. Comput. Sci.*, vol. **5**, no. 12, pp. 17–23, 2014.

[8] E. You, Y. Ota, and N. Tepluhina, *Vue.js*. [Online]. Available: https://vuejs.org/

[9] 'React – A JavaScript library for building user interfaces'. Accessed: Jan. 26, 2022. [Online]. Available: https://reactjs.org/

[10] M. Darrel, J. Harmon, and J. Whitlock, 'The OpenAPI Specification'. Jan. 27, 2022. [Online]. Available: https://github.com/OAI/OpenAPI-Specification

[11] Django Software Foundation, *Django*. 2019. [Online]. Available: https://djangoproject.com

[12] 'Spring makes Java simple.' Accessed: Jan. 26, 2022. [Online]. Available: https://spring.io/

[13] L. Richardson and S. Ruby, *RESTful Web Services*. O'Reilly Media, 2008. [Online]. Available: https://books.google.at/books?id=XUaErakHsoAC

[14] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, 'Foundations of JSON schema', in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 263–273.

[15] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[16] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2017. [Online]. Available: https://www.R-project.org/

[17] B. Schloerke and J. Allen, *plumber: An API Generator for R*. 2022.

[18] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[19] F. Pedregosa *et al.*, 'Scikit-learn: Machine learning in Python', *J. Mach. Learn. Res.*, vol. **12**, no. Oct, pp. 2825–2830, 2011.

[20] D. Bender and K. Sartipi, 'HL7 FHIR: An Agile and RESTful approach to healthcare information exchange', in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, Porto, Portugal, Jun. 2013, pp. 326–331. doi: 10.1109/CBMS.2013.6627810.

[21] P. G. D. Group, 'PostgreSQL'. Jan. 2022. Accessed: Jan. 26, 2022. [Online]. Available: https://www.postgresql.org/

[22] Hibernate Community, *Hibernate*. [Online]. Available: https://hibernate.org/

[23] 'Health Level Seven International - Homepage \textbar HL7 International'. Accessed: Jan. 26, 2022. [Online]. Available: http://www.hl7.org/index.cfm

[24] J. Hippisley-Cox, C. Coupland, and P. Brindle, 'Development and validation of QRISK3 risk prediction algorithms to estimate future risk of cardiovascular disease: prospective cohort study', *BMJ*, p. j2099, May 2017, doi: 10.1136/bmj.j2099.

[25] R. Conroy, 'Estimation of ten-year risk of fatal cardiovascular disease in Europe: the SCORE project', *Eur. Heart J.*, vol. **24**, no. 11, pp. 987–1003, Jun. 2003, doi: 10.1016/S0195-668X(03)00114-3.

[26] S. F. Weng, J. Reps, J. Kai, J. M. Garibaldi, and N. Qureshi, 'Can machine-learning improve cardiovascular risk prediction using routine clinical data?', *PLOS ONE*, vol. **12**, no. 4, p. e0174944, Apr. 2017, doi: 10.1371/journal.pone.0174944.

[27] 'PM2 - Home'. Accessed: Jan. 26, 2022. [Online]. Available: https://pm2.io/