Data Driven Risk Management For Fire Services

—

Chimney Fire Prediction Application

Ömer Esas

DATA DRIVEN RISK MANAGEMENT FOR FIRE SERVICES

—

CHIMNEY FIRE PREDICTION APPLICATION

E n g D T h e s i s

to obtain the degree of
Engineering Doctorate (EngD) at the University of Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board,
to be defended
on Thursday, September 19, 2024 at 10.00

by

Ömer Esas
born on December 15, 1993
in Izmir, Turkey

This EngD Thesis has been approved by:

Thesis Supervisors: Prof. Dr. R.J. Boucherie

Co-supervisors: Prof. Dr. M.N.M van Lieshout
Dr. ir. M. de Graaf
P.J. Visscher

UNIVERSITY OF TWENTE.

Acknowledgements

I am deeply indebted to Prof. Dr. M.N.M van Lieshout for her exceptional mentorship. Her insights, patience and encouragement have been the cornerstone of this work, and words cannot express my gratitude for her unwavering support throughout my engineering doctorate journey.

I would like to express my deepest appreciation to Dr. ir. M. de Graaf for his invaluable guidance and constructive feedback. His expertise has significantly enhanced the quality and depth of this project. Additionally, I would like to extend my heartfelt appreciation to Prof. Dr. R.J. Boucherie for his ongoing support and advice during the entire course of my programme.

I am extremely grateful to the team at Twente Fire Brigade, especially Paul, Marijn, Emiel and Niels. Their collaboration and provision of real-world context have been instrumental in grounding our research in practical application.

Special thanks to my colleague Changqing for providing the foundational work upon which my project builds. Our collaborative efforts and exchange of ideas have contributed significantly to the success of our endeavor.

I would like to extend my sincere thanks to the Netherlands Organisation for Scientific Research (NWO) for funding this project. Their support has been crucial in bringing this work to fruition.

I am also thankful to the user committee for their insightful feedback and industry perspective, which helped ensure the practical relevance of this work.

Lastly, I would like to thank my family and friends. Their understanding and encouragement have been a constant source of strength throughout this process.

I would be remiss if I did not mention all those who have contributed to this journey in ways both big and small. This thesis is the result of collective effort and support, for which I am profoundly grateful.

Contents

Introduction

1

In this engineering doctorate thesis, we present the development of an innovative chimney fire prediction application for the Twente Fire Brigade (TFB), by utilizing the mathematical predictive model defined in the research paper by Lu et al.[22]. By transforming the mathematical model into a practical, web-based tool, this project aims to empower the TFB with data-driven insights about chimney fire incidents for improving capacity planning, enhancing fire safety and emergency response in the Twente region.

1.1 Background

Since 2013, the Twente Fire Brigade (TFB) has been collaborating with researchers from the University of Twente on several preliminary studies to enhance their operations using Business Intelligence. These studies, often carried out by students, have explored various aspects of fire incidents, such as identifying the source of car fires [4], determining suitable locations for new fire stations [20], and using decision theory to schedule the relocation of fire engines during major incidents [15].

Two recent studies by Wendels [42] and School [37] have motivated the current project. In 2017, Wendels, a bachelor's student in applied mathematics, attempted to model fire-related emergency calls in the Twente region using an inhomogeneous Poisson point process [18]. However, due to the consideration of too many different types of incidents, the relatively simple model could not fully explain the data. Wendels recommended focusing on a single type of fire, limiting the number of explanatory variables and accounting for omitted variables and latent influences using a Cox process.

Building upon Wendels' recommendations, School, a master's student, collaborated with the TFB to develop a dashboard using PowerBI that specifically focused on modeling chimney fires, one of the most common types of residual building fires in the Twente region. By narrowing the scope to a single fire type, School developed a more accurate and reliable prediction model using two statistical approaches: an inhomogeneous Poisson process and a log Gaussian Cox process [26]. While the inhomogeneous Poisson process provided a basic understanding of factors influencing chimney fires, the log Gaussian Cox process enhanced the predictive capability by accounting for spatial correlations. The result was a predictive tool, implemented in R and PowerBI, which helped the TFB to create awareness for the residents in Twente.

Lastly, the recent study by Lu et al.[22] serves as the foundation for our project, as the authors developed a comprehensive data-driven modeling process that

combines machine learning techniques for variable selection and point process tools for model development. Their approach resulted in a fully data-driven and interpretable model that accurately predicts chimney fire risk in the Twente region. Our project aims to transform the research output by Lu et al. into a web-based chimney fire prediction tool that will remain useful, maintainable and easily integratable with the BI systems of the Twente Fire Brigade.

1.2 Outline of the EngD Thesis

In the following chapters of this engineering doctorate thesis, we examine the various aspects of our endeavor as outlined below:

Chapter 2 presents the key concepts and relevant information about the project goals and structure. It introduces the ArchiMate modeling framework, which is used extensively throughout the thesis to illustrate both the technical design and non-technical aspects of the project. The chapter also delves into the underlying motivations driving the project, highlighting how the research objectives align with the Twente Fire Brigade's strategic goals of enhancing public safety through data-driven risk management. Finally, it examines the organizational structure and roles of key stakeholders involved in the realization of this innovative application.

Chapter 3 investigates the problem domain, providing a thorough understanding of the underlying mathematical model developed by Lu et al.[22] and the IT systems used by the TFB. It examines the data-driven modeling process, which combines machine learning techniques for variable selection and point process tools for model development. The chapter also explores the Twente Fire Brigade's existing IT landscape, identifying potential integration points and data compatibility considerations essential for designing a seamlessly integrated software solution. Furthermore, it defines the problem we aim to solve by establishing high-level principles that guide the design of our chimney fire prediction application.

Chapter 4 discusses the methodological aspects of developing the chimney fire prediction application. It begins by explaining the agile software development approach adopted for the project, emphasizing the importance of iterative and collaborative development. The chapter then presents a list of key requirements for the application, which were established and refined through collaboration with the TFB and the research team. Next, the chapter outlines the architectural design of the application across four distinct milestones, providing detailed insights into the goals, features, and considerations associated with each milestone. It also explores the technology choices made throughout the development process, discussing the rationale behind selecting specific programming languages and technologies such as R, JavaScript and Docker.

Chapter 5 explains the technical implementation details of the chimney fire prediction application, following the milestone-based approach outlined in the previous chapter. It explores the progressive development of the application's features and functionalities, from the initial prototype in Milestone 1 to the final solution in Milestone 4. The chapter again utilizes ArchiMate modeling language to illustrate the API and discusses the development of the front-end application, which enhances the usability of the tool for the risk management team.

Chapter 7 summarizes the key findings and contributions of the project. It discusses the application's potential impact on enhancing the organization's risk management capabilities and supporting their mission to protect the community from the effects of chimney fires. The chapter also identifies areas for future

research and development, such as integrating findings from similar studies in other regions and incorporating advanced features to provide a more comprehensive and flexible risk assessment tool.

The appendices provide additional technical details and resources, including the API endpoints (Appendix A), example API responses (Appendix B.1 and B.2), the structure of the Excel files used for model refitting (Appendix C), and screenshots of the model refit page of the front-end application (Appendix D).

Preliminaries

2

In this chapter, we present the necessary key concepts and provide the relevant information about our goals and structure of our endeavor. We begin by introducing the ArchiMate modeling framework, as ArchiMate views are extensively used to illustrate both the technical design and non-technical aspects of our project. Having a basic understanding of ArchiMate will be essential in interpreting these diagrams, grasping the inner workings and connections between software components, and the impact of our application. Next, we delve into the underlying motivations that drive this project, highlighting how the research objectives align with the Twente Fire Brigade's strategic goals of enhancing public safety through data-driven risk management. This section underscores the significance of our project and its potential impact on the organization's operations. Finally, we examine the organizational structure and the roles of key stakeholders involved in the realization of this innovative application. By discussing the diverse backgrounds and contributions of the collaborators, we emphasize the importance of interdisciplinary cooperation in tackling complex, real-world challenges at the intersection of academia and public safety.

2.1 ArchiMate Modeling Language

ArchiMate is a standardized modeling language developed by The Open Group to support enterprise architecture and the design, analysis and visualization of an organization's business processes and IT infrastructure [39]. ArchiMate provides a comprehensive and consistent framework for describing, analyzing and visualizing complex systems, enabling stakeholders to understand and manage their organization's architecture effectively.

ArchiMate modeling relies on a well-defined set of concepts and relationships, organized into aspects and layers, as illustrated in Figure 2.1. Aspects represent different perspectives or views of the enterprise architecture, providing specialized viewpoints that focus on specific aspects of the architecture:

- **Active Structure**: This aspect focuses on subjects that can perform behavior, i.e., the "subjects" of technical and non-technical activities, and can be the organization's structure, components and properties. It includes concepts such as actors, business roles, application components and infrastructure elements.
- **Behavior**: This aspect captures the dynamic aspects of the architecture, describing the behavior and interactions of the architectural elements over time. It includes concepts such as processes, functions, services and interfaces.

- **Passive Structure**: This aspect deals with the representation and management of data and information within the architecture. These are elements on which behavior is performed. It includes concepts such as data objects, business objects and artifacts.
- **Motivation**: This aspect focuses on modeling the drivers, goals, principles, requirements and stakeholders that influence and shape the architecture. They provide the context of an enterprise's architecture.

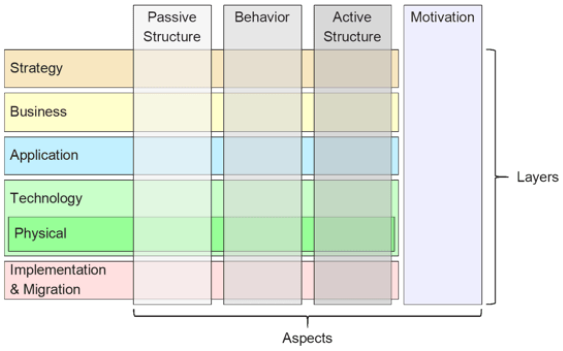


Figure 2.1 Layers and Aspects in ArchiMate Modeling Language

Layers, on the other hand, represent different levels of abstraction within the architecture. They define a hierarchical structure that helps to organize and separate different domains within the enterprise architecture. ArchiMate defines these primary layers:

- **Strategy Layer**: Represents the highest-level layer in the ArchiMate framework and focuses on capturing strategic aspects of an organization's enterprise architecture. It includes elements such as capabilities, business goals, drivers, principles and policies.
- **Business Layer**: Offers products and services to internal users and external customers, which are realized in the organization by business processes performed by business actors and roles.
- **Application Layer**: Provides a bridge between the business layer and the technology layer. It supports the business layer with application services which are realized by software components.
- **Technology Layer**: Offers infrastructural services such as processing, storage and communication services needed to run applications, realized by computer and communication hardware and system software.
- **Physical Layer**: Expands the technology layer by incorporating physical infrastructure elements, such as servers, storage devices and facilities.
- **Implementation and Migration Layer**: Supports the modeling of transformation and change within the architecture, including project portfolios, work packages and migration paths.

The layering approach in ArchiMate allows for a structured representation of the architecture, enabling a clear separation and understanding of different domains and their relationships. Each layer encapsulates specific elements and relationships

that are relevant to that particular domain. Together, aspects and layers provide a comprehensive framework for modeling and analyzing the enterprise architecture from different perspectives and levels of abstraction.

Utilizing ArchiMate modeling in the development of the chimney fire risk prediction application for the Twente Fire Brigade ensures alignment with the organization's IT landscape and business processes. The ArchiMate views provided in this thesis effectively illustrate the interconnections between various application components and the existing systems and processes of the TFB. This comprehensive visualization facilitates a clearer understanding of the application's role and its integration within Twente Fire Brigade's operations. Additionally, ArchiMate views serve as an invaluable communication tool, fostering a shared understanding among stakeholders regarding the system's objectives, functionalities and organizational impact, which in turn enhances collaboration and supports informed decision-making throughout the project's lifecycle.

2.2 Motivation

In this section, the driving forces behind the need for our project and its alignment with the strategic goals of Twente Fire Brigade are explained. The collaborating parties, their goals and overarching drivers are illustrated in Figure 2.2, through the motivation elements of ArchiMate modeling framework.

The initial group of stakeholders comprises three members from the Stochastic Operations Research (SOR) group at the University of Twente and a co-writer of the proposal for our project. The co-writer's contract with UT SOR group ended in the early stages of our work and even though he started working full-time as innovation program manager at Thales, he continued to supervise our team and guide in the right direction. The team is led by a professor who also serves as a senior researcher specializing in spatial stochastics and acts as the project's supervisor on the university's behalf. Under her guidance are a PhD candidate and an EngD candidate, who focus on data science and software development, respectively. Collectively, our research team are eager to apply their research toward directly enhancing public safety. Their responsibilities include conducting fire risk prediction research and developing its application for the TFB, which will be elaborated upon in Section 2.3.

From the industry side, the research conducted by the SOR group could help enhance the operational planning of TFB. The risk management and business intelligence (BI) teams of TFB are committed to leveraging their BI capabilities to gain deeper insights into chimney fire risks on a neighborhood level. The BI coordinator integrates these data-driven insights into the brigade's strategic decision-making processes. The BI team translates complex data into clear reports and dashboards that highlight fire risk patterns. Meanwhile, the risk management team uses these insights to develop strategies and plans to mitigate fire risks. This enhanced understanding will enable them to more effectively allocate fire crews and assets, strategically plan future resources based on the assessed risks and increase public awareness about fire safety and prevention.

The motivations illustrated in Figure 2.2 reflect Twente Fire Brigade's commitment to their core mission: protecting life, property, and the environment from the devastating effects of fires and other emergencies. Eager to enhance their capabilities in minimizing fire damage and optimizing response strategies, TFB aims to strengthen public safety and resilience. This collaboration with the SOR group of the University of Twente enables them to combine advanced business

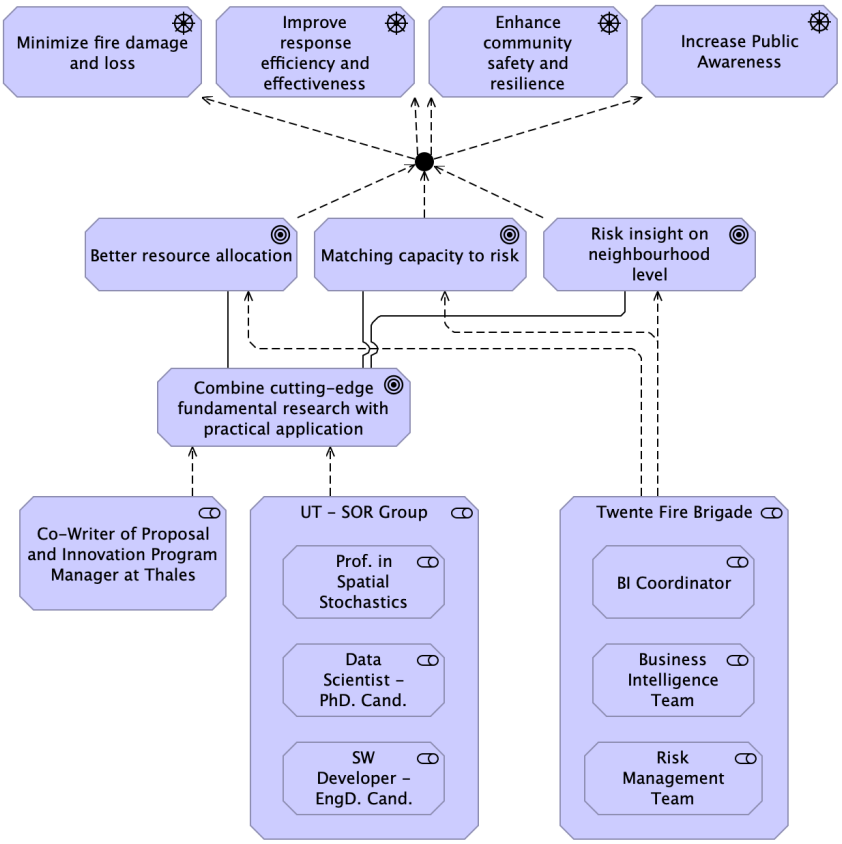


Figure 2.2 ArchiMate Motivation View of the Chimney Fire Prediction Application

intelligence and academic research, thereby achieving their strategic objectives, improving their operational effectiveness and increasing public awareness.

2.3 Organization

The development of our sophisticated application demanded a coordinated effort involving participants of diverse backgrounds, each contributing to this endeavour from their own perspective. A detailed diagram that depicts the collaborators and their contribution is presented in Figure 2.3.

At the core of our project is the foundational research conducted by Lu et. al. [22], which was crucial for facilitating the subsequent design and implementation of our chimney fire prediction application. Under the mentorship of the professor specializing in spatial stochastics, the PhD candidate developed a mathematical model that is able to predict the chimney fire risk intensity, by integrating machine learning and spatial statistics methods. The predictive model was trained on datasets comprising past chimney fire incident records and various spatio-temporal data, provided by the business intelligence staff of Twente Fire Brigade.

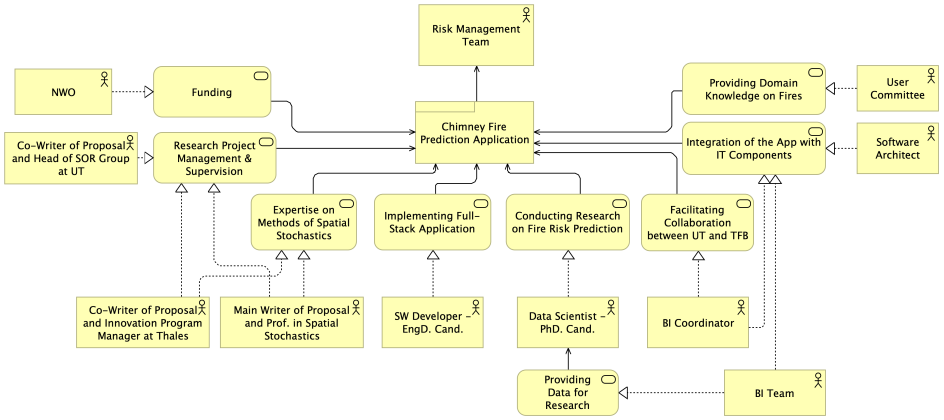


Figure 2.3 ArchiMate Business Collaboration View of the Chimney Fire Prediction Application

The successful output of this research paved the way for transitioning into the implementation phase of our application.

Continuing from the research output, I, as the EngD candidate, took the lead in the application development, focusing on translating the predictive models into a practical software solution for the TFB. This responsibility involved the design and implementation of a robust software architecture capable of providing accurate predictions of chimney fire risks to the risk management team.

The three supervisors wrote the proposal for funding for our project from NWO [28], and with their extensive experience in managing research projects, oversaw the whole project, making sure that all activities adhered to high standards and the objectives of the project are met. In addition, during our bi-annual meetings and discussions outside, representatives from Twente and other fire brigades in the Netherlands provided domain knowledge about fire incidents and general feedback on our progress. Meanwhile, the BI coordinator played a pivotal role in fostering effective collaboration between the academic staff, the business intelligence and the risk management teams of TFB. Furthermore, between the development iterations, the software architect facilitated the deployment and integration of our application. Last but not least, the research team and TFB were inspired to dedicate time and effort to this subject, thanks in large part to the promising results obtained by bachelor and master degree students who had previously collaborated with Twente Fire Brigade. Although not directly involved in our project, their work showcased the value of applying mathematical models to predict chimney fires.

Together, these efforts from different actors highlight the importance of collaboration and interdisciplinary skills in tackling complex projects. This project not only helps the risk management team for operational planning tasks but also serves as a model for future collaborations between academia and public safety organizations.

Problem Investigation 3

Developing an accurate and reliable chimney fire prediction application requires a thorough understanding of the underlying mathematical model and the IT systems used by the Twente Fire Brigade. In this chapter, we delve into the research conducted by Lu et al.[22] on predicting chimney fire risk in the Twente region using a Poisson point process model. We examine the data-driven modeling process, which combines machine learning techniques for variable selection and point process tools for model development. Then, we explain the mathematical model that predicts chimney fires, which will be the set of equations that will be implemented in our application.

Furthermore, we investigate the IT products and infrastructure utilized by the TFB to gain a better understanding of their data sources, data warehouse and the tools they employ for data integration, analysis and reporting. By exploring their existing IT landscape, we can identify potential integration points and data compatibility considerations. This knowledge is essential for designing a software solution that seamlessly integrates with their systems, leverages existing tools when appropriate and introduces custom functionality when necessary.

3.1 Mathematical Model

The research paper "Data-driven chimney fire risk prediction using machine learning and point process tools" by C. Lu et al.[22] addresses the critical need for accurate fire risk predictions to support the Twente Fire Brigade in their mission to protect and serve the community. The authors focus on chimney fires, a frequent occurrence that significantly impacts daily life and is heavily influenced by environmental factors. In collaboration with the TFB, the study aims to develop a comprehensive risk prediction model for chimney fires in the Twente region.

The authors begin by examining the existing literature on fire risk prediction, categorizing the approaches into machine learning-based and statistical methods. Machine learning techniques, such as logistic regression, decision trees and neural networks, can automatically identify relationships between fire risk and environmental variables. However, these methods often require discretizing continuous data and may not yield easily interpretable results. In contrast, statistical approaches, like point process models, can directly learn from spatio-temporal fire patterns and provide interpretable results with confidence intervals. Various studies have utilized statistical methods, including the K-function [34], Poisson processes [18] and log Gaussian Cox processes [26], to analyze and predict fire occurrences. Nevertheless, these studies often determine the model structure first

and then select and fit explanatory variables, potentially not fully exploiting the available information.

To overcome these limitations and align with the Twente Fire Brigade's needs, the authors propose a two-step data-driven modeling process that combines the strengths of machine learning and point process tools. The first step involves using random forests and permutation importance techniques to non-parametrically select the most informative explanatory variables. In the second step, they construct a spatio-temporal Poisson point process model based on the selected variables and estimate the model parameters using logistic regression. This hybrid approach enables a fully data-driven and interpretable model for chimney fire risk prediction.

3.1.1 Chimney Fire and Environmental Data

With regards to the data used in the research, the authors first collected data on 1,759 chimney fire incidents occurring in the Twente region between January 1, 2004 and December 31, 2020. Each incident is recorded with an ID number, location in Dutch RD coordinate system [9], time in days, and a brief description of the fire and rescue processes. The RD (Rijksdriehoek) coordinate system is a commonly used geodetic reference system in the Netherlands, designed to map the country with high precision. It is based on a grid that divides the country into coordinates in meters, originating from the zero point at Amersfoort. The RD system simplifies spatial data handling by using a planar coordinate grid, thus ensures accurate mapping and analysis of spatial data, making it especially useful and easier to work with for regional studies like ours. The spatial and temporal projections of the incidents reveal heterogeneous patterns, with most fires occurring in urban areas and during winter months.

To capture the factors influencing chimney fire risk, the authors collaborated with experts from the TFB to identify 27 putative explanatory variables. These variables include spatial factors such as building types, population density and composition, and urbanity degrees, as well as temporal factors like season and weather conditions. Building information, from $V_{\sigma,1}$ to $V_{\sigma,11}$, such as the total number of houses, their types, ages and functions, is sourced from the Netherlands Institute for Public Safety (NIPV) [8]. Population data, from $V_{\sigma,12}$ to $V_{\sigma,22}$, including the number of residents, their ages and gender distribution, as well as urbanity data, are provided by the Statistics Netherlands (CBS) [38] and recorded over 6,291 pre-defined 500m $\times$ 500m area boxes in Twente. Daily weather observations, from $V_{\tau,1}$ to $V_{\tau,5}$, including wind speed, temperature, wind chill, sunshine duration and visibility, are collected from the Royal Netherlands Meteorological Institute (KNMI) [35], specifically from the Twente airport weather station, with additional data from two neighboring stations to assess the influence of small weather variations across the region. The list of 27 putative variables is presented in Table 3.1, where the following notation is used for environmental variables:

- $V_{\sigma,i}(u)$: the smoothed value of the i-th spatial variable at location u,
- $V_{\tau,i}(t)$: the assigned value of the i-th temporal variable at time t,

where (u,t) denotes any location and time combination in the spatio-temporal domain.

3.1.2 Selection of Most Informative Variables

Following data acquisition and pre-processsing, the authors employ a non-parametric variable importance analysis using random forests and permutation importance

Variable	Description
$V_{\sigma,1}$	The total number of houses
$V_{\sigma,2}$	The number of houses with an industrial function
$V_{\sigma,3}$	The number of houses with a hotel function
$V_{\sigma,4}$	The number of houses with a residential function
$V_{\sigma,5}$	The number of houses constructed before 1920
$V_{\sigma,6}$	The number of houses constructed between 1920 and 1945
$V_{\sigma,7}$	The number of houses constructed between 1945 and 1970
$V_{\sigma,8}$	The number of houses constructed between 1970 and 1980
$V_{\sigma,9}$	The number of houses constructed between 1980 and 1990
$V_{\sigma,10}$	The number of houses constructed after 1990
$V_{\sigma,11}$	The number of free standing (detached or semi-detached) houses
$V_{\sigma,12}$	The total number of residents
$V_{\sigma,13}$	The number of residents with an age in the range of 0 till 14
$V_{\sigma,14}$	The number of residents with an age in the range of 15 till 24
$V_{\sigma,15}$	The number of residents with an age in the range of 25 till 44
$V_{\sigma,16}$	The number of residents with an age in the range of 45 till 64
$V_{\sigma,17}$	The number of residents with an age of 65 or higher
$V_{\sigma,18}$	The number of male residents
$V_{\sigma,19}$	The number of female residents
$V_{\sigma,20}$	The number of addresses in the neighbourhood
$V_{\sigma,21}$	The urbanity of the neighbourhood
$V_{\sigma,22}$	Boolean variable indicating the presence of a town
$V_{\tau,1}$	Daily mean wind speed (km/h)
$V_{\tau,2}$	Daily mean temperature (°C)
$V_{\tau,3}$	Daily mean wind chill (°C) (calculated from $V_{\tau,1}$, $V_{\tau,2}$)
$V_{\tau,4}$	Daily sunshine duration (h)
$V_{\tau,5}$	Daily minimum visibility (h)

Table 3.1 Description of Initial Putative Variables

techniques to select the most informative variables for their chimney fire risk prediction model. Random forests are robust classification and regression methods widely used in risk prediction and data mining applications. They consist of a collection of decision trees, each trained on a sampled subset of the data using repeated bagging (bootstrap sampling with replacement). The final output is a combined result over all trees. Additionally, random forests can assess the importance of a variable by measuring the mean increase in prediction error when the variable's values are randomly permuted across observations. The authors use random forests to determine the importance of spatial and temporal variables separately, while accounting for correlations between variables.

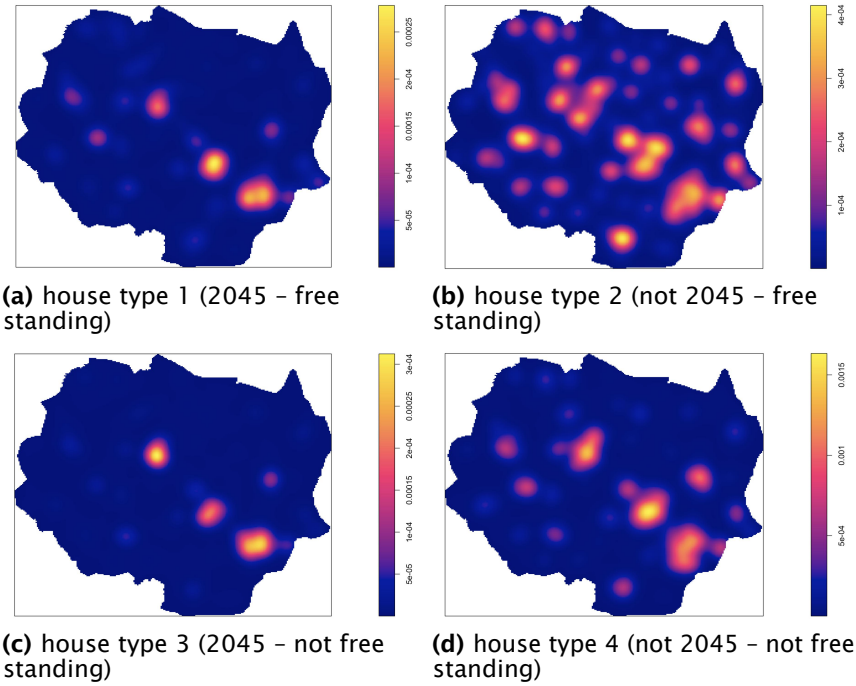
The variable importance results under conditional permutation reveal that the most important spatial variables are the number of buildings constructed between 1920 and 1945 ($V_{\sigma,6}$) and the number of free-standing (detached or semi-detached) houses ($V_{\sigma,11}$). For temporal variables, wind speed ($V_{\tau,1}$) and wind chill ($V_{\tau,3}$) are identified as the most influential factors.

3.1.3 Poisson Point Process Model

The authors first categorize the houses into four types of houses, based on construction age (i.e., whether they have been constructed between 1920 and 1945 or not) and whether they are free-standing (detached or semi-detached) or not. Then, they assume that each house catches a chimney fire independently at a type-dependent rate that varies in time. Under this assumption, the fire prediction model is a spatio-temporal Poisson point process defined by the intensity function

$$\Lambda(u,t) = \sum_k \lambda_k(u,t) = \sum_k h_k(u)\phi_k(t) \tag{3.1}$$

$\Lambda(u,t)$ indicates the overall fire risk at location u at time t and $\lambda_k(u,t)$ indicates the risk at location u at time t for houses of type k. To calculate the overall chimney fire intensity $\Lambda(u,t)$, the individual chimney fire intensities for house type k, $\lambda_k(u,t)$, are summed over values of k, k ranging from 1 to 4. $h_k(u)$ represents the spatial density of houses of type k at location u, which is derived from the corresponding building information by smoothing using Gaussian kernels. To illustrate, the spatial density of four types of houses is presented in Figure 3.1.



(a) house type 1 (2045 – free standing)

(b) house type 2 (not 2045 – free standing)

(c) house type 3 (2045 – not free standing)

(d) house type 4 (not 2045 – not free standing)

Figure 3.1 Spatial Density Maps of Four House Types

The term $\phi_k(t)$ represents the temporal risk intensity for a house of type k at time t, which is modeled using a log-linear combination of functions as shown in Equation 3.2. The first component of the power term is a harmonic function with order $o_{k,1}$, employed to capture seasonal variations in chimney fire risk. The second and third components are polynomial functions with orders $o_{k,2}$ and $o_{k,3}$, respectively, used to model the effects of wind speed ($V_{\tau,1}(t)$) and wind chill

($V_{\tau,3}(t)$) on the temporal risk intensity. Additionally, the authors include a fourth component, which is a polynomial function of the interaction between wind speed and wind chill with order $o_{k,4}$.

$$\begin{aligned}\phi_k(t) = \exp(&\text{Harmonic}(t; o_{k,1}) + \text{Polynom}(V_{\tau,1}(t); o_{k,2}) + \\ &\text{Polynom}(V_{\tau,3}(t); o_{k,3}) + \text{Polynom}(V_{\tau,1}(t)V_{\tau,3}(t); o_{k,4}))\end{aligned} \tag{3.2}$$

To fit the Poisson point process model, the authors use logistic regression estimation. They estimate the model parameters for each house type separately using a three-step implementation involving the specification of a dummy point process, generation of realizations from the dummy point process and estimation of model parameters using logistic regression. The temporal intensity functions obtained for four house types as a result of model fitting process are presented in Equations 3.3, 3.4, 3.5 and 3.6.

$$\begin{aligned}\phi_1(t;\theta_1) = \exp\Bigg(&\theta_{1,1} + \theta_{1,2}\cos\left(\frac{2\pi}{365}t\right) + \theta_{1,3}\sin\left(\frac{2\pi}{365}t\right) \\ &+ \theta_{1,4}\cos\left(\frac{4\pi}{365}t\right) + \theta_{1,5}\sin\left(\frac{4\pi}{365}t\right) \\ &+ \theta_{1,6}\cos\left(\frac{6\pi}{365}t\right) + \theta_{1,7}\sin\left(\frac{6\pi}{365}t\right) \\ &+ \theta_{1,8}\cos\left(\frac{7\pi}{365}t\right) + \theta_{1,9}\sin\left(\frac{8\pi}{365}t\right) \\ &+ \theta_{1,10}V_{\tau,3}(t) + \theta_{1,11}V_{\tau,3}^2(t)\Bigg),\end{aligned} \tag{3.3}$$

$$\begin{aligned}\phi_2(t;\theta_2) = \exp\Bigg(&\theta_{2,1} + \theta_{2,2}\cos\left(\frac{2\pi}{365}t\right) + \theta_{2,3}\sin\left(\frac{2\pi}{365}t\right) \\ &+ \theta_{2,4}\cos\left(\frac{4\pi}{365}t\right) + \theta_{2,5}\sin\left(\frac{4\pi}{365}t\right) \\ &+ \theta_{2,6}\cos\left(\frac{6\pi}{365}t\right) + \theta_{2,7}\sin\left(\frac{6\pi}{365}t\right) \\ &+ \theta_{2,8}V_{\tau,3}(t) + \theta_{2,9}V_{\tau,3}^2(t) + \theta_{2,10}V_{\tau,3}^3(t) \\ &+ \theta_{2,11}V_{\tau,3}^4(t)\Bigg),\end{aligned} \tag{3.4}$$

$$\begin{aligned}\phi_3(t;\theta_3) = \exp\Bigg(&\theta_{3,1} + \theta_{3,2}\cos\left(\frac{2\pi}{365}t\right) + \theta_{3,3}\sin\left(\frac{2\pi}{365}t\right) \\ &+ \theta_{3,4}\cos\left(\frac{4\pi}{365}t\right) + \theta_{3,5}\sin\left(\frac{4\pi}{365}t\right) \\ &+ \theta_{3,6}\cos\left(\frac{6\pi}{365}t\right) + \theta_{3,7}\sin\left(\frac{6\pi}{365}t\right) \\ &+ \theta_{3,8}V_{\tau,3}(t)\Bigg),\end{aligned} \tag{3.5}$$

$$\begin{aligned}\phi_4(t;\theta_4) = \exp\Bigg(&\theta_{4,1} + \theta_{4,2}\cos\left(\frac{2\pi}{365}t\right) + \theta_{4,3}\sin\left(\frac{2\pi}{365}t\right)\\ &+ \theta_{4,4}\cos\left(\frac{4\pi}{365}t\right) + \theta_{4,5}\sin\left(\frac{4\pi}{365}t\right)\\ &+ \theta_{4,6}\cos\left(\frac{6\pi}{365}t\right) + \theta_{4,7}\sin\left(\frac{6\pi}{365}t\right)\\ &+ \theta_{4,8}\cos\left(\frac{7\pi}{365}t\right) + \theta_{4,9}\sin\left(\frac{8\pi}{365}t\right)\\ &+ \theta_{4,10}V_{\tau,3}(t) + \theta_{4,11}V_{\tau,3}^2(t) + \theta_{4,12}V_{\tau,3}^3(t)\\ &+ \theta_{4,13}V_{\tau,1}(t)V_{\tau,3}(t)\Bigg).\end{aligned} \tag{3.6}$$

The estimates of the parameters of the temporal intensity functions are provided in Table 3.2.

Parameter	Estimate (CI)	Parameter	Estimate (CI)
$\theta_{1,1}$	-1.22e1(±3.82e-1)	$\theta_{1,2}$	-2.78e-1(±4.81e-1)
$\theta_{1,3}$	-1.44e-1(±2.84e-1)	$\theta_{1,4}$	-1.63e1(±2.64e-1)
$\theta_{1,5}$	7.99e-2(±3.04e-1)	$\theta_{1,6}$	-9.21e-3(±2.49e-1)
$\theta_{1,7}$	-2.33e-2(±2.62e-1)	$\theta_{1,8}$	3.05e-1(±2.19e-1)
$\theta_{1,9}$	1.59e-2(±2.20e-1)	$\theta_{1,10}$	-7.42e-2(±3.56e-2)
$\theta_{1,11}$	-6.12e-3(±3.31e-3)	$\theta_{2,1}$	-1.30e1(±3.31e-1)
$\theta_{2,2}$	6.40e-2(±2.74e-1)	$\theta_{2,3}$	1.42e-2(±1.56e-1)
$\theta_{2,4}$	-3.14e-2(±1.64e-1)	$\theta_{2,5}$	1.69e-1(±1.60e-1)
$\theta_{2,6}$	1.48e-1(±1.24e-1)	$\theta_{2,7}$	-4.54e-2(±1.20e-1)
$\theta_{2,8}$	-6.81e-2(±2.64e-2)	$\theta_{2,9}$	2.33e-3(±3.65e-3)
$\theta_{2,10}$	-8.50e-6(±2.47e-4)	$\theta_{2,11}$	-2.76e-5(±1.93e-5)
$\theta_{3,1}$	-1.43e1(±9.48e-2)	$\theta_{3,2}$	1.57e0(±1.60e0)
$\theta_{3,3}$	2.25e-1(±6.00e-1)	$\theta_{3,4}$	-1.19e0(±1.09e0)
$\theta_{3,5}$	-3.19e-1(±7.20e-1)	$\theta_{3,6}$	6.32e-1(±6.06e-1)
$\theta_{3,8}$	-1.12e-1(±5.25e-2)	$\theta_{3,9}$	6.32e-2(±1.76e-1)
$\theta_{4,1}$	-1.39e1(±2.41e-1)	$\theta_{4,2}$	1.49e-1(±3.05e-1)
$\theta_{4,3}$	2.77e-2(±1.61e-1)	$\theta_{4,4}$	-2.06e-1(±1.83e-1)
$\theta_{4,5}$	9.00e-2(±1.95e-1)	$\theta_{4,6}$	2.77e-2(±1.61e-1)
$\theta_{4,7}$	7.11e-2(±1.72e-1)	$\theta_{4,8}$	1.82e-1(±1.36e-1)
$\theta_{4,9}$	-4.84e-2(±1.38e-1)	$\theta_{4,10}$	-9.38e-2(±4.06e-2)
$\theta_{4,11}$	-7.92e-4(±1.99e-3)	$\theta_{4,12}$	-2.35e-4(±1.62e-4)
$\theta_{4,13}$	2.99e-3(±2.10e-3)		

Table 3.2 Parameter Estimates of the Temporal Intensity Functions and Their 95% Confidence Intervals

3.2 IT Systems of Twente Fire Brigade

Following the overview of the mathematical model, we examine the IT products and infrastructure utilized by the Twente Fire Brigade through two ArchiMate views. In these views, we use different colors to represent elements from various layers of the ArchiMate framework. The blue elements represent components in the application layer, specifically the software applications and data objects used by the TFB. The green elements denote physical layer components, such as the actual servers or databases where these applications and data reside. Finally, the yellow elements signify business layer elements, representing the business processes and functions supported by the IT infrastructure. It is important to note that the views do not seek to model the entire IT architecture or business processes of the TFB, which also involves dispatch mechanisms of the fire crew and other operational systems, in addition to the aspects that are present in most organizations, such as human resources, personnel management or finances. In other words, in the following two views, only the elements and relationships that are closely related to our project have been included.

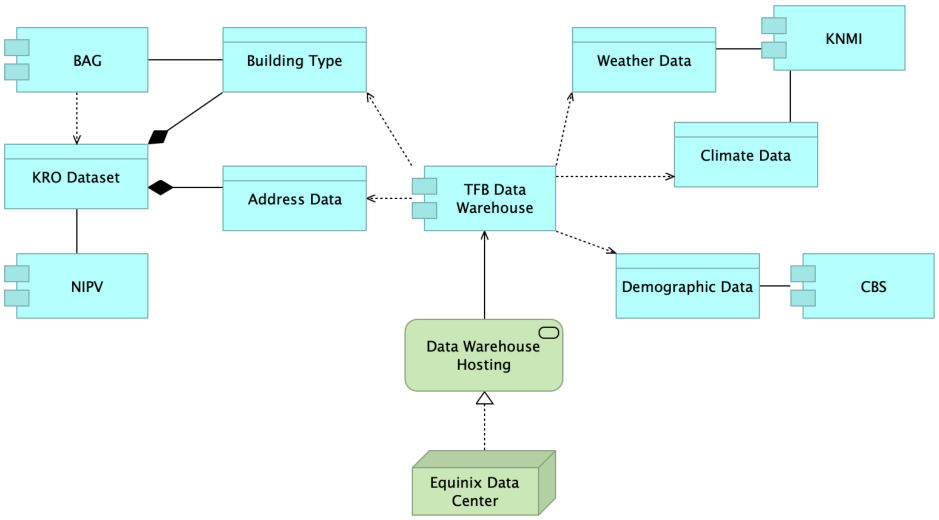


Figure 3.2 Archimate Application Usage View of the Data Warehouse and Data Sources

In Figure 3.2, the data warehouse of the TFB is placed at the center of the figure as it is the source of the data used to train the mathematical model. The data warehouse is hosted by an internet service company, namely Equinix [10], in an on-premise data center in Enschede. Pointing out from the application component “TFB Data Warehouse”, various kinds of data used for training the model are presented, accessed by connecting to numerous APIs, databases or data warehouses of other organizations. For example, the past weather and climate data is collected from the data source of KNMI. Similarly, Twente Fire Brigade utilizes the data about the characteristics of buildings and areas, and official address designations from the ”KRO” dataset. The ”KRO” dataset is a collection of various kinds of data from different kinds of public organizations in the Netherlands and is provided by

NIPV. Regarding the building data, "Basic Registration of Addresses and Buildings" (BAG)[16] contributes the data about buildings to the KRO dataset. Last but not least, CBS provides demographic data, such as population and age segments.

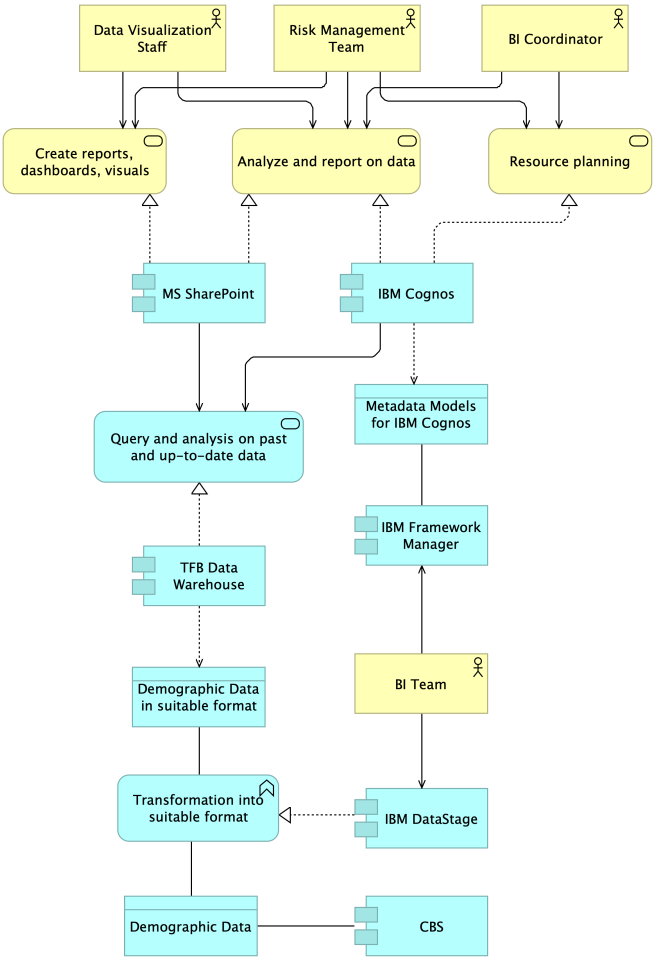


Figure 3.3 ArchiMate Application Usage View of Business Intelligence Processes at Twente Fire Brigade

The next view, presented in Figure 3.3, illustrates how the data from various data sources are used for the business intelligence operations at TFB, using demographic data and CBS as examples of data and data source. Before saving the data of interest, which is represented as a demographic data in Figure 3.3, the original data needs to be transformed into a format that is suitable for writing it to the data warehouse. For this task, the BI team of Twente Fire Brigade uses IBM DataStage tool, which is a data integration tool that enables users to extract, transform and load data from various sources and prepare it for use in BI and analytical operations. After the data is loaded on the data warehouse, the data then becomes available for query and analysis purposes. For such tasks, one of the applications the BI team at TFB uses is Microsoft SharePoint, which is a web-based collaboration and centralised

content management platform for organizations to share information and manage documents, with features such as versioning and document approval workflows. In addition to MS SharePoint, the BI and the risk management teams also utilize IBM Cognos software, which provides a set of tools to gather and analyze data and create interactive and dynamic reports. In order to analyse, understand and make sense of the vast amount of data that are obtained from numerous data sources and saved on the data warehouse, the data in the data warehouse need to be organised and related to each other, which is achieved by creating metadata models via IBM Framework Manager. In other words, the metadata models define the structure and relationships of data, without needing to understand the underlying data structures in the data warehouse. In short, IBM Framework Manager helps to create a common business vocabulary for data that is consistent across the organization, which makes it easier for the BI staff to analyze data and derive insights, while IBM Cognos uses these metadata models to generate reports, dashboards and various kinds of visualisations.

In conclusion, understanding the IT landscape of TFB empowers us to design a solution that aligns with their needs, without introducing unnecessary complexity or redundancy. By carefully considering the frequency of use cases, the effort needed to integrate and the potential benefits, we can make informed decisions about when to leverage existing tools and when to implement custom functionality through code. By finding the right balance between utilizing existing systems and developing custom components, we can deliver a powerful tool that enhances Twente Fire Brigade's decision-making processes and supports their mission to protect and serve the community.

3.3 Problem Definition

As a result of exploring the mathematical model and IT systems of Twente Fire Brigade, we establish the following high level principles that will allow us to design our system in more detail in Section 4.3.

1. **Input Specification**: The application shall accept a specific region within the Twente area as an input parameter. The granularity and format of the input region will be determined in collaboration with the Twente Fire Brigade to align with their operational needs.
2. **Output Requirements**: The primary output of the application shall be the predicted number of chimney fires for the specified region, accompanied by a confidence interval. The confidence interval will provide a measure of the uncertainty associated with the prediction, enabling better-informed decision-making.
3. **Weather Data Integration**: To enhance the accuracy and relevance of the predictions, the application shall incorporate up-to-date weather data. By leveraging real-time or forecasted weather information, the application can capture the influence of weather conditions on the likelihood of chimney fires.
4. **Integration with Existing Infrastructure**: The chimney fire prediction application shall be designed to integrate with the Twente Fire Brigade's day-to-day operational IT infrastructure. This includes considering the compatibility with existing data formats, systems and workflows to ensure smooth adoption and utilization of the application.

5. **Utilization of Mathematical Model**: The application shall be based on the mathematical model developed by Lu et al.[22], specifically the formula (3.1).

By adhering to these guiding principles, the chimney fire prediction application will be developed as a targeted and effective tool that addresses the specific needs of the Twente Fire Brigade. The subsequent sections of this thesis will delve into the architectural design and implementation details, demonstrating how these principles are realized in practice.

Methodology

4

In this chapter, we delve into the methodological aspects of developing the chimney fire prediction application for the Twente Fire Brigade. We begin by discussing our design approach, which is rooted in agile software development principles. By adopting an iterative and collaborative approach, we aim to deliver a high-quality application that effectively meets the evolving needs of the TFB.

The chapter then proceeds to outline the key requirements of the chimney fire prediction application, and presents the architectural design of the application in four distinct milestones. Each milestone represents a significant step forward in the development process, incrementally enhancing the application's capabilities and addressing key requirements. We provide detailed insights into the goals, features and considerations associated with each milestone, showcasing how the application evolves from a basic prototype to a sophisticated and adaptable tool.

Furthermore, we explore the technology choices made throughout the development process. We discuss the rationale behind selecting specific technologies, such as R, Javascript and Docker, highlighting their suitability for the project's requirements. By examining these technological decisions, we aim to provide a comprehensive understanding of the technical foundation upon which the chimney fire prediction application is built.

4.1 Design Approach

In developing the chimney fire prediction application for the Twente Fire Brigade, we adopted an agile software development methodology with iterations, as opposed to relying on extensive upfront documentation. This decision aligns with the principles outlined in the Agile Manifesto, which emphasizes the value of individuals and interactions, working software, customer collaboration, and responding to change [3].

Agile methodologies are well-suited for projects with evolving requirements and the need for rapid results. They promote adaptability, improve productivity, and enhance software quality [27]. Given the innovative nature of this project, which combines advanced mathematical modeling with practical software engineering, a flexible approach was crucial to accommodate insights and improvements identified during the development process. By working in iterations, we could continuously refine the application based on feedback from the TFB and the research team, ensuring that the final product effectively met the needs of the Twente Fire Bridage.

The close collaboration between the development team and stakeholders, a core principle of agile methodologies, was particularly valuable in this project. Regular interactions with the TFB allowed us to develop a better understanding of their

requirements and constraints. This collaborative approach fostered a shared sense of ownership and ensured that the application was tailored to their specific needs, increasing the likelihood of successful adoption and long-term value.

In conclusion, adopting an agile software development methodology with iterations was a strategic choice that aligned with the project's objectives, the collaborative nature of the partnership, and the need for a flexible and responsive approach. By prioritizing individuals and interactions, working software, and customer collaboration, we could deliver a high-quality chimney fire prediction application that met the Twente Fire Brigade's needs effectively and efficiently.

4.2 List of Key Requirements

The following list outlines the key requirements for the chimney fire prediction application. These requirements were established and refined through collaboration with the Twente Fire Brigade and the research team, taking into account the specific needs and constraints of the project. The requirements focus on the functionality, usability, and maintainability of the application, ensuring that it effectively supports the TFB in their risk management processes.

- R1 The back-end application shall compute the chimney fire prediction risk for a specific area and a specific date provided as inputs, in accordance with the algorithm defined by Lu et al.[22]. The area shall be one of the following: gemeente (municipality), wijk (district), buurt (neighbourhood), or a 500 by 500 meter square box defined by the Statistics Netherlands (CBS) in the Twente region.
- R2 The back-end application shall incorporate up-to-date weather forecast data, including air temperature and wind speed, to have the forecasted wind speed and wind chill values to compute the chimney fire risk predictions. The weather data shall be obtained from a source that is able to make reliable weather forecasts for the Twente region.
- R3 The back-end API shall provide confidence intervals alongside the predicted number of chimney fires in the API response, offering additional information about the reliability and precision of the predictions.
- R4 The back-end application shall make the chimney fire prediction data available through an application programming interface (API), accepting HTTP requests with specific region and date as input parameters.
- R5 The back-end API shall use communication protocols and data formats, such as JSON, that are suitable for integration with the IT products used by the business intelligence team of Twente Fire Brigade. The data format shall be well-documented and adhere to industry standards to ensure smooth integration and interoperability.
- R6 The back-end application shall include the capability to refit the model parameters, specifically the coefficients of the temporal term of the mathematical model, using an R script that utilizes appropriate spatial statistics and linear model libraries.
- R7 The back-end application shall expose API endpoints that allow users to trigger the model refitting process remotely.
- R8 The back-end application shall be capable of receiving and processing new input data for training the predictive model, ensuring compatibility with the existing data structures and model requirements.

R9 The front-end application shall provide a user-friendly and intuitive interface for users to interact with the API endpoints defined in the back-end application. Users shall be able to visualize the chimney fire predictions on an interactive map and easily upload new datasets to update the model. The front-end application shall prioritize usability and clarity to ensure a smooth user experience.

R10 Upon completion of the project, the developed artifacts, including the back-end API and front-end application, shall be accompanied by comprehensive documentation. The documentation shall enable the Twente Fire Brigade to effectively maintain, update, and extend the application as needed. It shall include, but not be limited to, deployment instructions, API references, user guides and troubleshooting procedures.

4.3 Architectural Design of Milestones

The architectural design of the chimney fire prediction application was divided into four distinct milestones, each focusing on specific aspects of the system's functionality and performance. These milestones were carefully planned to ensure a progressive development process, allowing for iterative improvements and the incorporation of feedback from the Twente Fire Brigade. In the following sections, we will explore the architectural design of each milestone in detail and their key goals and features.

4.3.1 Milestone 1

In the design phase of Milestone 1, our goal was to quickly establish a functioning software tool that relied on historical data for its predictions. By the conclusion of this milestone, we aimed to have a back-end API capable of forecasting chimney fire risks across various areas of the Twente region using the past weather data. Even though employing historical data compromises the accuracy of our predictions, it simplifies the implementation and allows us to quickly integrate and test the IT systems, ensuring all components operate smoothly and validate the system's overall functionality from the outset. This ensures all components are able to function as expected right from the start and establishes a reliable basis for the features planned in subsequent milestones. With this mindset, we designed the architecture of Milestone 1 to consist of a simple API and the necessary IT systems from TFB for visualization purposes, as seen in Figure 4.1.

The back-end API is designed to serve as a source for generating predictions based on historical data. To fetch the chimney fire prediction data, TFB is to develop an Extract-Transform-Load (ETL) script that makes HTTP GET requests to the API and saves the prediction data to the data warehouse of TFB. To display the predictions on an interactive map, a Geo-Information-System (GIS) tool, which is already utilised for visualization tasks at TFB as can be seen in Figure 4.2, can connect to the data warehouse and the geo database to obtain the chimney fire prediction data and geographical information of the areas in the Twente region, respectively. Last but not least, to realise this and subsequent architectures, the application is to be deployed to Azure cloud platform in a Docker container, using the Azure account of TFB.

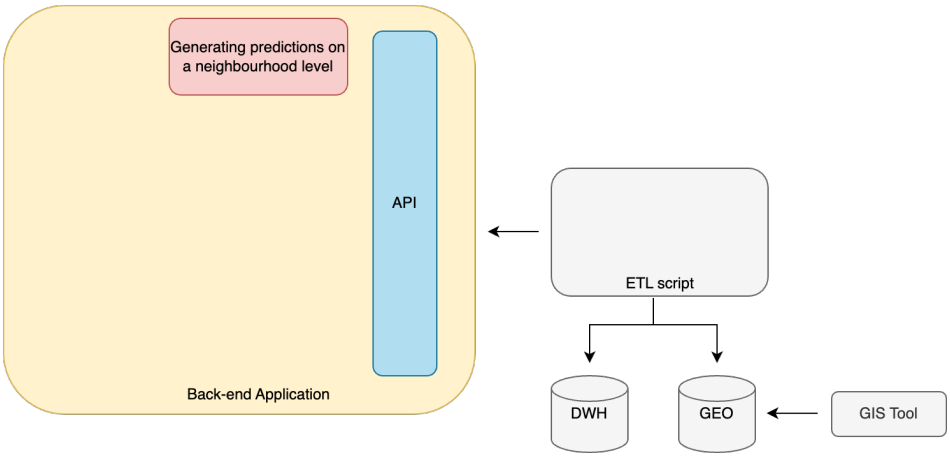


Figure 4.1 Architectural Diagram of Milestone 1

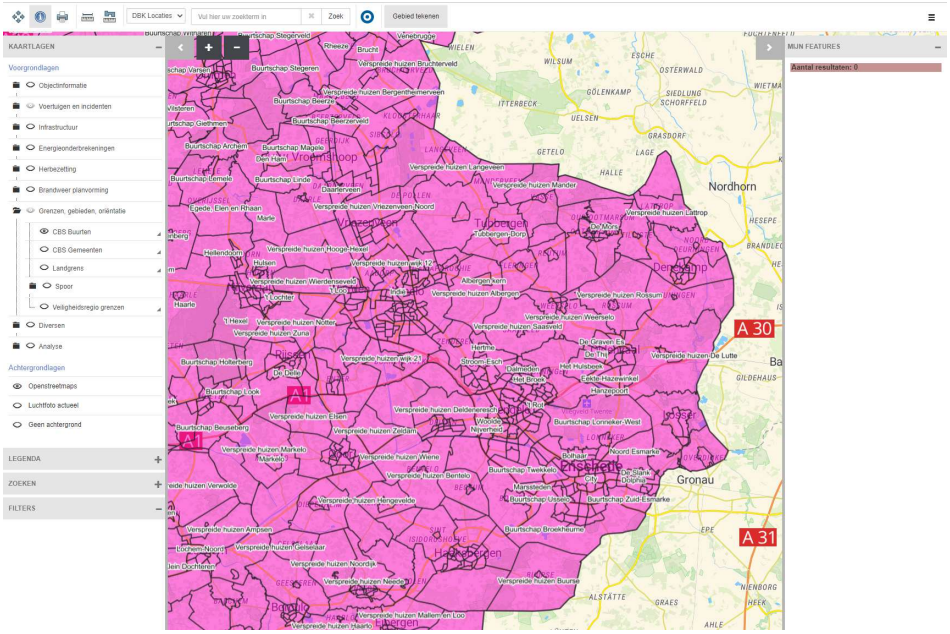


Figure 4.2 Geo-Information System tool used by the TFB

4.3.2 Milestone 2

In this iteration, we integrate the capability to receive and process the up-to-date weather forecast data, which allows the back-end application to generate more accurate and timely predictions about the chimney fire risks in the Twente region. The architectural diagram of Milestone 2 is illustrated in Figure 4.3.

Specifically, the back-end API in Milestone 2 connects to an external weather forecasting service to obtain the up-to-date weather predictions. These forecasts

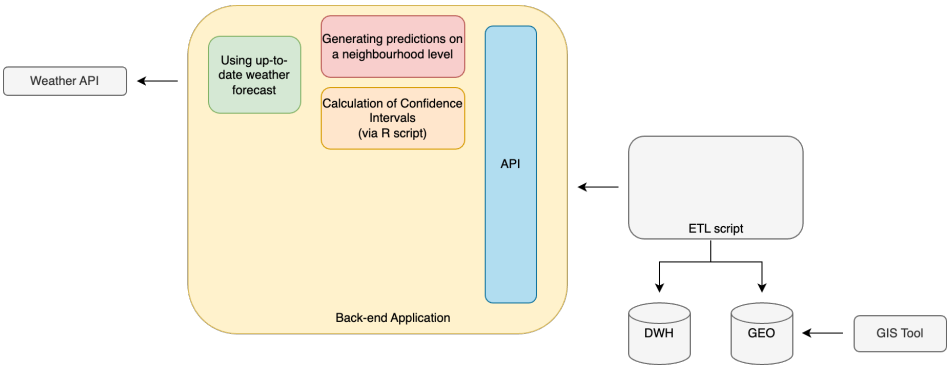


Figure 4.3 Architectural Diagram of Milestone 2

include critical weather parameters that significantly influence chimney fire occurrences, such as air temperature, wind speed and the wind chill data which is derived from these two values [19]. By incorporating current weather data, the API can adjust its risk assessments on a daily basis to reflect changing environmental conditions, thereby providing predictions that are not only based on historical weather patterns but also on the present weather scenario.

Another significant enhancement in Milestone 2 is the introduction of confidence intervals in the prediction outputs. Confidence intervals provide a statistical range that is likely to contain the true value of the number of chimney fires expected, offering an additional layer of information about the reliability and precision of the predictions. This feature is particularly valuable as it helps the decision-makers at the TFB understand the uncertainty associated with the predictions, aiding them in risk assessment. As a side note, during the design phase of milestones, we expected that the calculation of confidence intervals might require the use of spatial statistics libraries in R, necessitating the running of some R processes in the back-end application. However, this anticipation turned out to be unnecessary, as we realised in the implementation phase of Milestone 2 that Javascript is sufficient to implement the necessary mathematical algorithm, which we explained in more detail in Section 5.1.2.

In short, Milestone 2 moves from a prototype that provided basic predictions based on historical data to a more sophisticated tool that utilizes up-to-date weather data and confidence intervals to allow the users to gauge the prediction accuracy.

4.3.3 Milestone 3

In Milestone 3, we aimed to implement the application's back-end logic that recalculates the parameters of the mathematical model, in addition to developing a user-friendly front-end application. The architectural diagram of Milestone 3 is presented in Figure 4.4.

The first feature of Milestone 3 is to realise the capability of refitting the model parameters, specifically, the coefficients of the temporal term of the mathematical model, by means of executing an R script that includes appropriate spatial statistics and linear model libraries. To ease the implementation efforts, our goal in Milestone 3 was to achieve the correct application logic, using the same dataset that is initially used to fit the model parameters. Even though employing the same dataset in

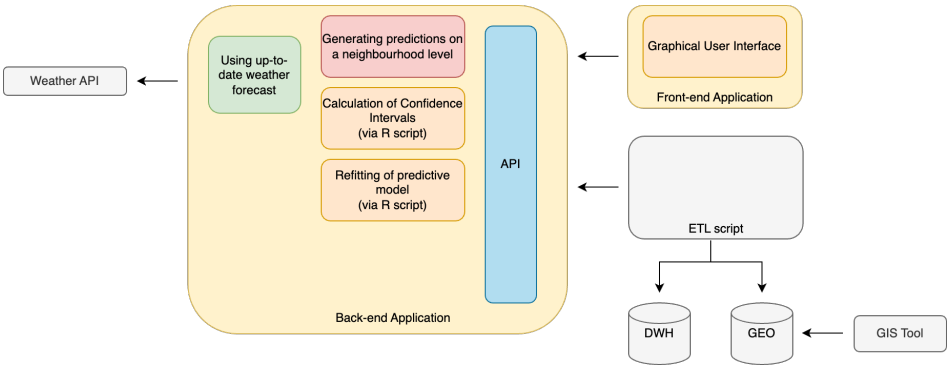


Figure 4.4 Architectural Diagram of Milestone 3

the model training process would result in very similar coefficients compared to existing model parameters, this would ensure the effectiveness and the correct behaviour of the model refitting mechanism and act as a precursor to using new training data in Milestone 4.

In addition to the efforts in making sure the application would stay relevant, we also aimed to develop a front-end application that serves as a graphical user interface for the users at TFB. This would enhance the user interaction with the application, enabling the users to trigger the model fitting process, without the need to have the IT skills to interact with the back-end application via manual HTTP requests.

In summary, our goals in Milestone 3 were to lay the groundwork for increased adaptability and the user experience of our chimney fire risk prediction tool. By implementing the back-end logic to allow for recalibration of model parameters, we aimed to realise an essential part in providing a software tool to TFB that would remain relevant in the coming years. Furthermore, the introduction of the front-end application was designed to provide user-friendly access to more sophisticated features not available through the GIS tool at the TFB, such as model refitting.

4.3.4 Milestone 4

In Milestone 4, we continued to build on the groundwork laid in Milestone 3, with a particular emphasis on finalizing the model refitting feature. By implementing a mechanism that allows users to input new datasets into the application, we aimed to fully harness the adaptability of the mathematical model. This enhancement ensures that the Twente Fire Brigade can utilize the most current and relevant data, thereby maximizing the benefits offered by our chimney fire prediction application. The architectural diagram designed to facilitate this functionality is presented in Figure 4.5.

The key feature of Milestone 4 is the capability of the back-end application to receive and process new input data for training the predictive model. This process can be initiated on-demand via the front-end application, enabling users at the TFB to trigger the retraining process as needed. The retraining involves updating the model's coefficients based on the latest data, which not only refines its predictions but also adapts to any changes in spatio-temporal factors affecting chimney fire risks.

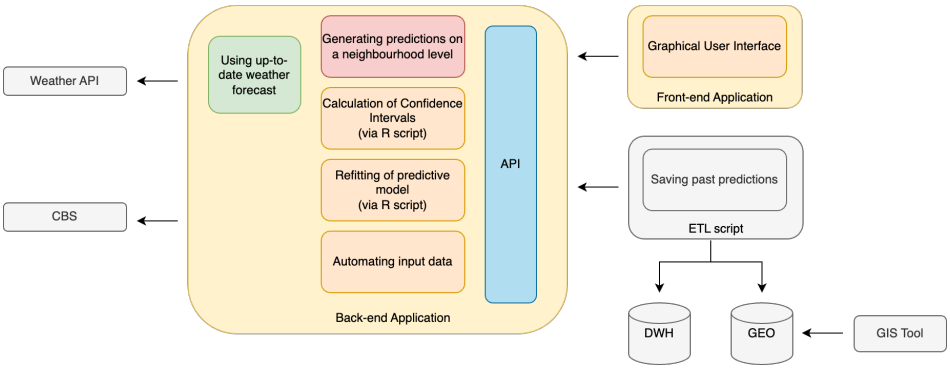


Figure 4.5 Architectural Diagram of Milestone 4

During the design phase of Milestone 4, as illustrated in Figure 4.5, we designed our architecture under the assumption that we would be able to identify and integrate an appropriate data source, such as CBS, into our application. However, not all necessary input data might be readily accessible via APIs. The retrieval, reformatting and integration of these new datasets might require thorough investigation and the development of new ETL (Extract, Transform, Load) processes within our back-end application. This might involve complex data cleansing and preparation steps to ensure that the data fed into the model is clean, relevant and structured appropriately. In the implementation phase of Milestone 4, we changed the mechanism of providing up-to-date spatial data to a more manual procedure, which we explained in Section 5.1.4.

Additionally, as part of Milestone 4, we wanted to illustrate that TFB can explore further enhancements and discover new ways to benefit from our chimney fire prediction application. For example, they can implement changes to their ETL processes to save these predictions after they receive the daily chimney fire risk data. This stored data can then be used to compare the predicted risk levels with the actual occurrences of chimney fire incidents. Such comparisons not only validate the accuracy of the predictions but also provide critical feedback that can be used to refine the predictive model. This process enhances its effectiveness and reliability over time, offering substantial support to the risk management team by demonstrating the reliability of the predictions, thereby helping them gauge how much they can confidently rely on these insights for making informed decisions.

In conclusion, Milestone 4 represents the completion of our project, where we have built upon and refined the capabilities established in earlier milestones. This final stage focused on enhancing the adaptability of our predictive model through the integration of new and varied datasets in a user friendly way, enabling on-demand retraining processes that adjust the model's parameters to reflect the latest spatio-temporal data. During the design phase of Milestone 4, the specific mechanism for inputting new data was not fully defined, and we continued under the assumption that it will become clearer and easier to assess the feasibility of various options as we progress through the implementation of the previous milestones. This iterative approach allows us to refine our strategies and ensure the most effective integration of new data into the system. All in all, these enhancements guarantee that the TFB can access the most up-to-date risk data, thus optimizing the utility and efficacy of the chimney fire prediction application, allowing them

to benefit from its predictions in risk management operations for many years to come.

4.3.5 Mapping of Requirements to Milestones

To ensure a structured and incremental development process, the key requirements of the chimney fire prediction application were implemented across four distinct milestones. Table 4.1 illustrates the mapping of these requirements to the respective milestones in which they were addressed.

Requirement	MS1	MS2	MS3	MS4
R1	✓	✓	✓	✓
R2		✓	✓	✓
R3		✓	✓	✓
R4	✓	✓	✓	✓
R5	✓	✓	✓	✓
R6			✓	✓
R7			✓	✓
R8				✓
R9				✓
R10				✓

Table 4.1 Mapping of Requirements to Milestones

4.4 Technology Choices

The development of the chimney fire prediction application required a careful selection of technologies to ensure a robust, efficient, and maintainable solution. For the back-end and front-end applications, we chose to use JavaScript, as it can be used to implement both server-side and client-side functionality. Additionally, I had prior experience with JavaScript, which is beneficial because it allowed me to leverage my existing knowledge and skills, resulting in faster development, improved code quality and reduced learning curve.

For the back-end application, we specifically utilized the Node.js runtime [30]. Node.js provides a scalable and high-performance environment for building server-side applications, making it an ideal choice for implementing the API that serves the prediction data. As for the server framework, we chose Express.js [29], a minimalist and widely-used framework that is known for its simplicity, extensive middleware support and popularity. Its extensive documentation also makes it a solid choice for building a maintainable web application for Twente Fire Brigade. Furthermore, for providing the prediction data, we opted for JSON [6] due to its widespread use and compatibility across a wide range of products. JSON integrates seamlessly with JavaScript, allowing for effortless conversion of JavaScript objects into JSON format. This ease of use, combined with JSON's lightweight data-interchange format, ensures efficient data handling and parsing within the Express.js framework.

To handle the spatial statistics computations necessary for updating the spatial and temporal terms of the predictive model, we utilized R scripts [40]. R is a powerful programming language and environment for statistical computing and

graphics, offering a wide range of libraries and packages specifically designed for spatial data analysis. Integrating R scripts into the back-end application allowed us to leverage the extensive spatial statistics capabilities of R while maintaining the flexibility and performance benefits of Node.js. Furthermore, the research by Lu et al.[22] was conducted using R, and the libraries and functions required for refitting the model, such as "spatstat" [2], "sf" [32] and "raster" [13], were readily available, making R a natural choice for our implementation.

To combine the dependencies of both Node.js and R, we employed Docker [7], a containerization platform that enables the packaging of an application along with its dependencies into a standardized unit. By containerizing the back-end application using Docker, we ensured a consistent runtime environment that includes the Node.js runtime, R libraries, and the necessary low-level C++ libraries required by the R libraries. This containerization approach simplifies deployment and ensures portability across different systems. Moreover, the TFB uses the Azure cloud platform, which provides easy ways of deploying Docker images as applications through Azure App Service [25], making Docker an ideal choice for seamless deployment and integration with their existing infrastructure.

For the front-end application, we selected React [12], a popular JavaScript library for building user interfaces. React's component-based architecture and declarative syntax make it easy to create reusable and modular UI components. Having prior experience with React was advantageous, as it allowed me to focus on solving the specific problems of the chimney fire prediction application rather than spending time learning a new framework. Additionally, React's extensive ecosystem provides access to a wide range of open-source libraries and components, such as data visualization tools and complex components like calendar widgets, which can be easily integrated into the application.

To efficiently fetch data from the back-end API and manage asynchronous state, we utilized Tanstack Query [21] (also known as React Query). Tanstack Query simplifies data fetching, caching, and synchronization, ensuring a smooth and responsive user experience. It also provides powerful features for monitoring the status of running R processes on the back-end, enabling real-time updates and progress tracking in the front-end application.

For navigation between different pages of the front-end application, we employed React Router [33]. React Router is a standard routing library for React applications, allowing for smooth navigation and a cohesive user experience.

To display the prediction data on a map of Twente region, we incorporated Leaflet [1] for the interactive map component. Leaflet is a lightweight and feature-rich JavaScript library for creating interactive maps, enabling users to explore and interact with the predicted chimney fire risk data in a geospatial context. In addition, the availability of open-source examples on GitHub and "developers first" approach also simplify learning and development, as opposed to other more complex map libraries.

Finally, we utilized Tailwind CSS [41] for styling the front-end application. Tailwind CSS is a utility-first CSS framework that provides a set of pre-defined classes for rapid and consistent UI development. Its flexible and customizable nature allowed us to create a visually appealing and responsive user interface without the need for extensive custom CSS.

In short, the combination of these technologies enabled us to build a solid and user-friendly chimney fire prediction application. Each technology was chosen based on its strengths, suitability for the specific requirements of the project and my prior experience in using these technologies to create web applications, ensuring a cohesive and efficient development process.

Implementation

5

In this chapter, we explain the technical implementation details of the chimney fire prediction application, following the milestone-based approach outlined in Section 4.3. We explore the progressive development of the application's features and functionalities, from the initial prototype in Milestone 1 to the final solution in Milestone 4. Throughout the chapter, we utilize the ArchiMate modeling language to illustrate our API and how an HTTP request from the ETL script of TFB is processed and served with the chimney fire prediction data. The blue elements in the ArchiMate views represent components in the application layer, specifically the functions executed by different software modules in sequence and the prominent data objects used as part of the chimney fire prediction algorithm. Similarly, the green elements denote those in the physical layer, in other words, the actual files where these functions and data are located. For simplicity, unless otherwise noted by green elements, the functions are implemented using JavaScript and as middlewares in the Express.js web framemork. By providing insights into the application logic, API design and the integration of R scripts for model refitting, we showcase how the application evolved to meet the needs of the TFB. Furthermore, we discuss the development of the front-end application, which enhances the usability and accessibility of the chimney fire prediction tool for the risk management team. In short, this chapter aims to provide a comprehensive understanding of the technical aspects that bring the chimney fire prediction application to life.

5.1 Implementation of Milestones

5.1.1 Milestone 1

As the first step in the implementation phase, to start producing the predictions and providing them to the TFB, the application logic implemented is illustrated in Figure 5.1. To fetch the prediction data, the application is sent an API request by the ETL script, which involves the data about the day and the area of interest as query parameters. The application then validates if the query parameters are valid, in other words, if the date is a date string in proper format, such as in "2023-01-25", and if the area code is a valid CBS area code, such as "GM0153" for the municipality of "Enschede" or "WK015303" for the district of "Twekkelerveld" in Enschede. If both query parameters are valid, then the application calculates the predicted number of chimney fires for that area on that day. An example API GET request and response in Milestone 1 is presented in Appendix B.1.

To avoid complexity in the first milestone, we decided to simplify the prediction generation process, which consists of calculating spatial and temporal terms and

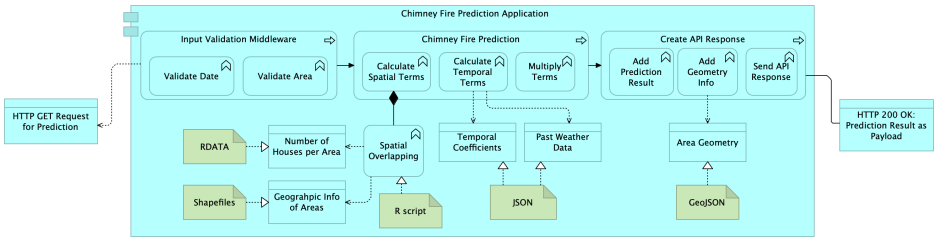


Figure 5.1 ArchiMate Application Logic View of Milestone 1

multiplying them to have the actual prediction result. To have the spatial terms, we took a straightforward approach and implemented this in a similar way to the code implemented during the research phase by Lu et. al. [22]. More specifically, we used the density information of houses of different types in the format of ”shapefile” and ”.RData” objects. Shapefiles are a popular geospatial vector data format for storing geometric location and associated attribute information, while .RData files are a binary file format used for storing R objects, such as data frames or spatial data [11]. Then, we made use of the “spatial overlapping” functions in R to have the spatial terms for the area given as a query parameter. As for the temporal terms for that date, we utilised only the historical weather data, namely, the wind speed and wind chill values for the Twente region in 2020, which we stored as JSON files alongside the temporal coefficients used to calculate the temporal terms for the date given as a query parameter in the API request. Multiplying the calculated spatial terms for the area of interest and temporal terms for the date of interest for each house type and summing them up, as shown in Equation (3.1) , we have the predicted number of chimney fires for that area on that date.

The course of the API request ends with creating and sending the API response in JSON format. Finally, the geometry information of the area is included in the response, in case the API request made by the ETL script of the TFB required this geometry information in GeoJSON format so that the GIS tool is able to draw the borders of the area of interest on the map of the Twente. In short, at the end of Milestone 1, we implemented the prototype which can process the API requests and respond with a prediction result, although only using historical weather data instead of up-to-date weather forecast, hence not responding with the most accurate predictions.

5.1.2 Milestone 2

In Milestone 2, we aimed to equip the application with the capability of having access to up-to-date weather forecasts, so that it uses current weather info in making predictions about the number of chimney fires in areas of interest. To this end, the application logic presented in Figure 5.2 is implemented.

The first major improvement in this version of the API is that it uses weather forecast data from a weather API to calculate the temporal terms of the predictions to be made. In our implementation, we identified three potential weather API providers: Buienradar [5], OpenWeatherMap [31], and Meteoserver [23]. Buienradar API only provided forecasts for the next five days and lacked clear documentation, which would make it challenging to integrate and utilize the API effectively. OpenWeatherMap offered daily forecasts for the next sixteen days, which was

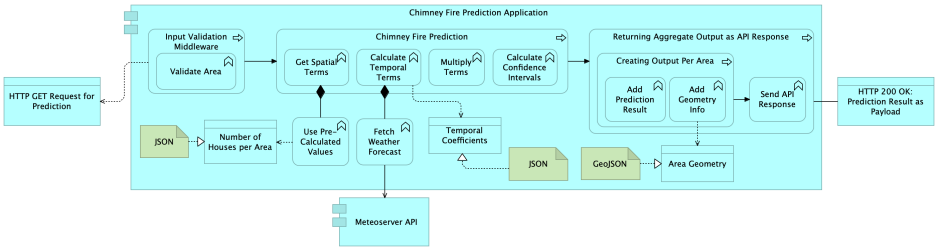


Figure 5.2 ArchiMate Application Logic View of Milestone 2

advantageous; however, it did not include the average wind speed information, instead providing the maximum wind speed expected for each day. Ultimately, we selected Meteoserver as our weather forecast provider due to its clear documentation, inclusion of the necessary data (mean temperature and mean wind speed for each day in the forecast), and provision of weather forecasts for the next ten days. Moreover, Meteoserver utilizes data from KNMI, which is sourced from the weather station at Twente Airport, aligning with the data source used in the research by Lu et al.[22]. The weather forecast data from the Meteoserver API is received in JSON format, which is the native data format for JavaScript-based applications, making it straightforward to read and process the weather data in our application. Moreover, we configured our application to fetch the weather data at the specific times of 0:35, 7:35, 12:35, and 18:35 every day, ensuring that the forecast data used in making predictions is always up-to-date. Since the Meteoserver API provides the weather data for the next ten days, our application's prediction capability is aligned with this time frame, allowing us to generate chimney fire risk forecasts for the upcoming ten day period.

The second major improvement is related to the spatial terms of the prediction calculation process. Instead of using the shapefiles, R-specific objects and scripts to calculate the spatial terms each time an API request is received, we calculated the spatial terms, in other words, the number of houses of the four types defined in the mathematical model, for all municipalities, districts, neighbourhoods and 500 by 500 meter boxes in Twente and stored the result in JSON format. Even though every few years there could be minor changes regarding the borders of neighbourhoods, because the number of areas in the Twente region will be more or less the same in the coming years, this approach of pre-calculating and storing spatial term values for each area was reasonable and practical. Furthermore, this optimisation cut the time spent for processing API requests from approximately thirty seconds to only one second, since the compute-heavy R scripts are replaced by simple look-up operations from a JSON file via Javascript code.

The third most important change in the second milestone is the capability of returning prediction results not only on a per-area basis, but also in an aggregate way. For example, if the GIS tool or ETL script of TFB would like to receive predictions for all municipality in Twente region, they can make an API request to the relevant endpoint, namely to "prediction/gemeente", instead of making fourteen separate requests for fourteen different municipalities in Twente, which would be API requests to "prediction/gemeente/GM0153", "prediction/gemeente/GM0158" and so on. In other words, in addition to the previous endpoints which return predictions per area, the second milestone version also implements four additional API endpoints: "prediction/gemeente", "prediction/wijk", "prediction/buurt" and

"prediction/box", which return the predictions for the areas of that type for the next ten days. Furthermore, in case the TFB does not need the geospatial information of the areas of interest and would like to receive a response smaller in size, we added a query parameter "excludeGeoInfo", which can be set to "true" during the API request to omit the GeoJSON part from the response. The last change we made in the API structure in Milestone 2 is to add the "lastWeatherFetchTimestamp" property as part of API responses, which informs TFB of the time the application fetched the weather forecast from the Meteoserver API. To describe these changes in the API more clearly, an example API GET request and response in Milestone 2 is illustrated in Appendix B.2.

During the implementation of the confidence interval calculation feature in Milestone 2, we initially considered using R scripts to manage these computations. However, upon further development, it became clear that these calculations could be effectively handled using JavaScript code exclusively. The algorithm starts with calculating the covariate vectors $C_k(t)$ of house type k for each day:

$$C_1(t) = \left[\begin{array}{l} 1, \cos\left(\frac{2\pi}{365}t\right), \sin\left(\frac{2\pi}{365}t\right), \cos\left(\frac{4\pi}{365}t\right), \sin\left(\frac{4\pi}{365}t\right), \\ \cos\left(\frac{6\pi}{365}t\right), \sin\left(\frac{6\pi}{365}t\right), \cos\left(\frac{8\pi}{365}t\right), \sin\left(\frac{8\pi}{365}t\right), V_{\tau,3}(t), V_{\tau,3}^2(t) \end{array} \right] \tag{5.1}$$

$$C_2(t) = \left[\begin{array}{l} 1, \cos\left(\frac{2\pi}{365}t\right), \sin\left(\frac{2\pi}{365}t\right), \cos\left(\frac{4\pi}{365}t\right), \sin\left(\frac{4\pi}{365}t\right), \\ \cos\left(\frac{6\pi}{365}t\right), \sin\left(\frac{6\pi}{365}t\right), V_{\tau,3}(t), V_{\tau,3}^2(t), V_{\tau,3}^3(t), V_{\tau,3}^4(t) \end{array} \right] \tag{5.2}$$

$$C_3(t) = \left[\begin{array}{l} 1, \cos\left(\frac{2\pi}{365}t\right), \sin\left(\frac{2\pi}{365}t\right), \cos\left(\frac{4\pi}{365}t\right), \sin\left(\frac{4\pi}{365}t\right), \\ \cos\left(\frac{6\pi}{365}t\right), \sin\left(\frac{6\pi}{365}t\right), V_{\tau,3}(t) \end{array} \right] \tag{5.3}$$

$$C_4(t) = \left[\begin{array}{l} 1, \cos\left(\frac{2\pi}{365}t\right), \sin\left(\frac{2\pi}{365}t\right), \cos\left(\frac{4\pi}{365}t\right), \sin\left(\frac{4\pi}{365}t\right), \\ \cos\left(\frac{6\pi}{365}t\right), \sin\left(\frac{6\pi}{365}t\right), \cos\left(\frac{8\pi}{365}t\right), \sin\left(\frac{8\pi}{365}t\right), \\ V_{\tau,3}(t), V_{\tau,3}^2(t), V_{\tau,3}^3(t), V_{\tau,1}(t)V_{\tau,3}(t) \end{array} \right] \tag{5.4}$$

Then, the standard deviation $\sigma_k(t)$ is calculated for house type k for each day using the inverse of G (Godambe) matrix, which is a predefined matrix that transforms the covariate data into a measure of variance:

$$\sigma_k(t) = C_k(t)\mathbf{G}^{-1}C_k(t)^T \tag{5.5}$$

Next, we compute the confidence intervals by scaling the temporal terms $\phi_k(t)$ for house type k as follows:

$$\left(\phi_k(t)\left(1 - \sigma_k(t)\xi_{1-\alpha/2}\right), \phi_k(t)\left(1 + \sigma_k(t)\xi_{1-\alpha/2}\right)\right) \tag{5.6}$$

To determine the appropriate quantile $\xi_{1-\alpha/2}$ from the standard normal distribution, we consider the confidence level α. Commonly, we set α to 5%, which corresponds to a 95% confidence interval. In this case, the quantile $\xi_{0.975}$ is used, which has a value of 1.96. This means that for a 95% confidence interval, the multiplier for the standard deviation is 1.96, ensuring that the true value of the temporal term $\phi_k(t)$ falls within the calculated confidence interval 95% of the time. In Figure 5.3, we illustrate these steps with an ArchiMate application logic view.

In conclusion, the second milestone of the chimney fire prediction application represents a significant advancement in our ability to provide timely and accurate predictions to the TFB. By integrating real-time weather data, enhancing the efficiency of spatial data processing and improving the API's response capabilities, this milestone greatly increases the operational competence of the chimney fire

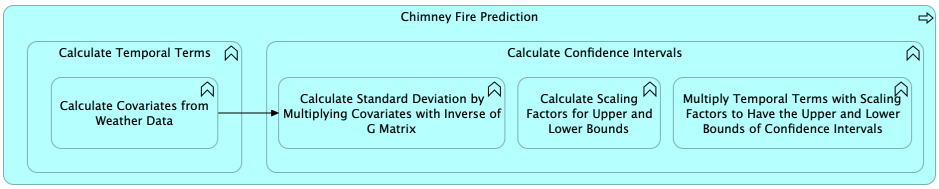


Figure 5.3 ArchiMate Application Logic View of Confidence Interval Calculation Algorithm

risk prediction system. The ability to incorporate the confidence intervals further strengthens the reliability of the predictions, providing the Fire Brigade with crucial data to make informed decisions. From this milestone and onwards, the TFB is able to truly reap the benefits of our chimney fire prediction tool, making it a valuable part of their risk management and operational planning processes.

5.1.3 Milestone 3

In our subsequent iteration, aimed at ensuring our chimney fire prediction application remains useful in the coming years, we introduced a critical functionality: enabling the application to process new data and update the parameters of the underlying mathematical model. To this end, in Milestone 3, our focus was primarily on implementing the logic for updating the model correctly, as opposed to facilitating the user interaction to use this feature of the application, which we scheduled for Milestone 4. By executing two R scripts that handle the model training datasets, we update the spatial and temporal parts of the prediction model. To provide a clearer understanding, we delve into the specifics of the implementation of these R scripts and then discuss how we enabled the capability to remotely execute these scripts within the application.

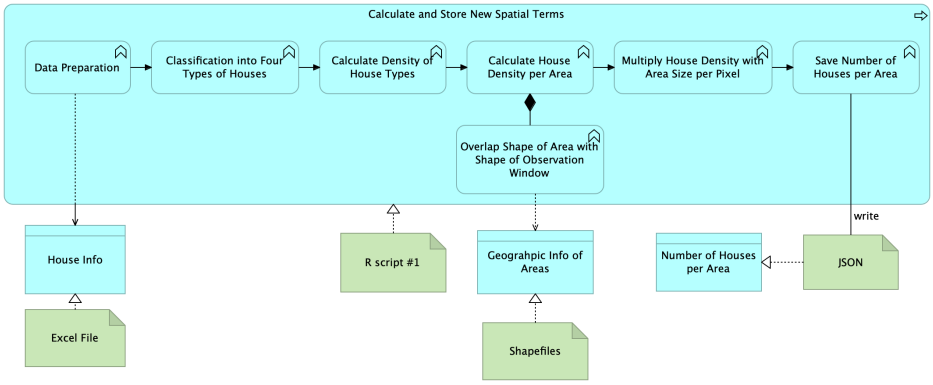


Figure 5.4 ArchiMate Application Logic View of Spatial Term Updating Process

In order to update the spatial part of the prediction model, we implemented the application logic illustrated in Figure 5.4. The process begins with the extraction of information about houses from an Excel file, which includes essential data for classifying houses based on two key variables: the construction year (before or

after 1945) and the architectural style (whether the house is free-standing or not). Following classification, the density of each house type within each geographic area is calculated. This step involves spatial analysis where the geographical shape of each area is overlaid with the original observation window, which is the shape of the whole Twente region, to have the house density metric that belonged to a specific area. These densities are then multiplied by the area size represented per pixel, essentially calculating the number of houses of different types per area. Finally, the calculated number of houses per area is saved in JSON format, overwriting the previous JSON file that contained the house count data for the same areas. In short, with this R script, we realised that the house count information, in other words, the spatial terms of our model, are continuously updated, establishing the first half of model updating capability of our application.

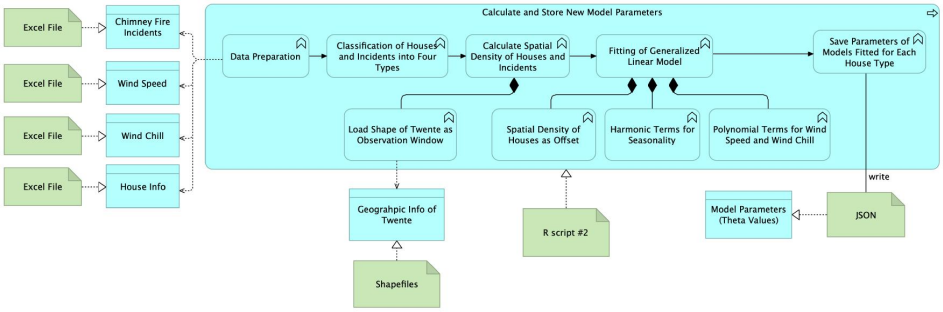


Figure 5.5 ArchiMate Application Logic View of Model Parameters Updating Process

To complement the spatial term updating process, we implemented a second R script that focuses on recalculating the model parameters of the prediction model, which are the coefficients of the temporal terms. As illustrated in Figure 5.5, the script begins by loading the four types of datasets, which are house information, chimney fire incidents, wind speed and wind chill, as R data frames. The script proceeds to prepare the data by ensuring consistency between wind speed and wind chill data, calculating the number of years and days based on the available data and loading the observation window object representing the Twente borders. It then filters the building data to include only the buildings that are in use and defines house classes representing freely standing or semi-detached houses.

Next, the script divides the building dataset into four house types based on construction year and architectural style, and the incident dataset into four subsets for each house type. Then, the script creates point patterns for each house type and each incident subset and calculates the spatial densities for both values using a Gaussian kernel bandwidth of 1000 meters. A generalized linear model is then fit with the model formula including harmonic terms for seasonality and polynomial terms for wind chill and wind speed. The offset term in the generalized linear model adjusts for the spatial density of the house type. After fitting the models for all four house types, the script combines the coefficients into a single list and writes the combined coefficients to a JSON file that contained the previous theta values. In concise terms, the second R script enables the application to read the model training datasets and update the coefficients of the temporal part of the predictions.

After implementing the two R scripts that update the spatial and temporal terms

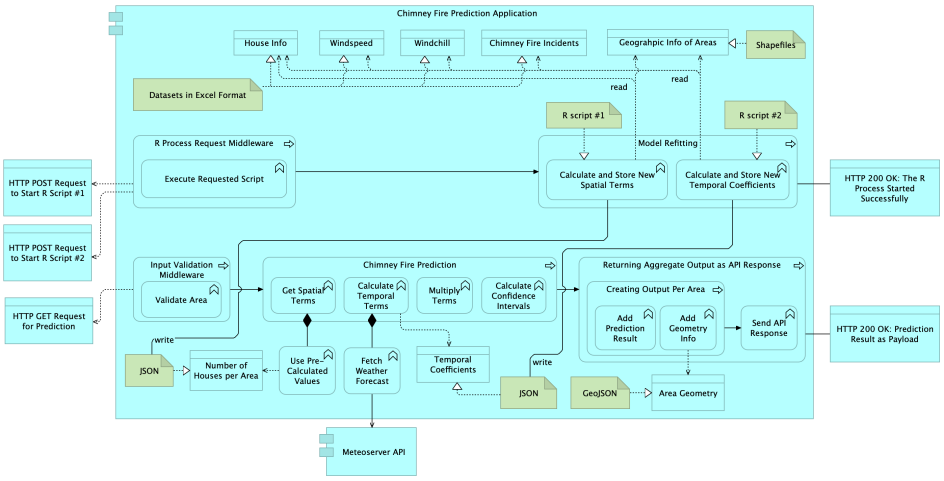


Figure 5.6 ArchiMate Application Logic View of Milestone 3

of the prediction model, we created two new API endpoints that allow the user to trigger the execution of these scripts. To start the house count and model refitting processes, HTTP POST requests to "/api/model/count" and "api/model/refit" are sent to the application. The API endpoint handler middleware then starts the requested R process, as a child process via "node:child_process" module, then replies with a response that informs the user that the requested R process has started. Upon completion of the R process, the contents of the respective JSON file is overwritten by the result of the R process, which allows the application to use the updated spatial or temporal coefficients for the chimney fire prediction process.

5.1.4 Milestone 4

In the last iteration, our primary focus was on implementing the functionality to upload new data files, which are crucial for refitting the prediction model with the latest information. Additionally, we made several small changes that finalized our back-end application. The application logic of the final version of the back-end application is illustrated in Figure 5.7.

To facilitate file uploading, we introduced a new API endpoint "/api/model/upload", which expects an HTTP PUT request with the Excel files that are intended to be uploaded to the application. When a PUT request is made to "/api/model/upload", the application expects a multipart form-data request body with the key "excelFiles", which should include one or more of the four Excel files named "kro.xlsx", "incident.xlsx", "windchill.xlsx" and "windspeed.xlsx", which contain the data on buildings, chimney fire incidents, wind chill and wind speed, respectively. Because the R scripts rely on these specific identifiers, it is essential to maintain the same names for the files and the columns within these Excel files. The contents of these four Excel files are detailed in the Appendix C.

Furthermore, to improve the reliability and performance of the application, we introduced a constraint that ensures only one R process can be run at a time. Given that R processes are memory-intensive, running multiple R processes simultan-

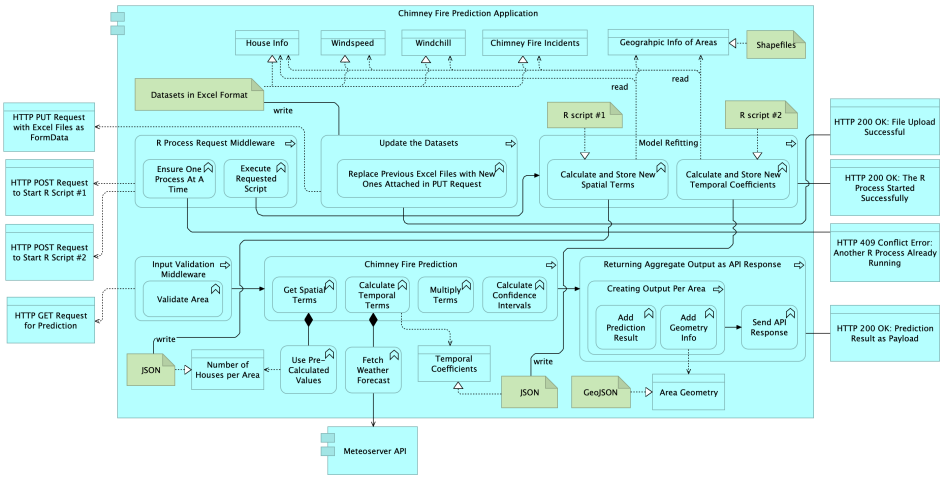


Figure 5.7 ArchiMate Application Logic View of Milestone 4

eously could significantly increase the RAM usage, potentially slowing down the application and causing delays. This could lead to confusion among users regarding the status and results of these processes. Therefore, if an R process is already running, any subsequent request to start another R process will receive an HTTP 409 Conflict error. This modification helps to manage system resources inside the Docker container more effectively and ensures that each process completes in a timely manner without overloading the server.

The third and final change we made to the back-end application is to modify the API endpoints that return the chimney fire prediction risks. Specifically, we changed the endpoint from "/prediction/:areaType" to "/api/prediction/:areaType" to achieve consistent naming with the endpoints that serve requests related to model refitting. The final structure of API endpoints of the back-end application is provided in Appendix A.

In summary, in Milestone 4 we successfully implemented the final piece of the model updating functionality, making our chimney fire prediction application robust and adaptive to new data inputs. By allowing easy file uploads, we ensured that the application could integrate the latest data to its model updating processes without requiring extensive manual intervention. Additionally, by improving and refining API endpoints and application logic, we finalised the application logic of our chimney fire prediction application, completing the features outlined during the planning of the milestones.

5.2 Front-End Application

Upon completing the planned functionalities, our chimney fire prediction application was ready for deployment by the risk management team of the Twente Fire Brigade. The application could provide the predictions to be displayed on an external mapping tool and the users could update model parameters via HTTP requests. However, interacting with the application through its API required famili-

arity with web technology and APIs. To improve the overall user experience and accessibility, we developed a front-end application with two main pages.

The first page we implemented is an interactive map that eliminates the need for additional data visualization tools. Users can select an area type and a specific date using the overlay located in the top right corner of the viewport. Once the selections are made, the map updates to display chimney fire risks for the chosen areas and date in the form of a heatmap. The colors on the heatmap change gradually to indicate higher risk areas, providing a clear visual representation of chimney fire risks. Upon clicking an area, a pop-up with the chimney fire risk information is displayed to the user. To implement this map, we utilized the Leaflet.js mapping library for interactive maps, React Query to efficiently fetch data from the back-end and the React framework to render all components seamlessly. With this interactive map, the application provides suitable data visualization tool out-of-the-box, removing the necessity of maintaining an external component to take advantage of chimney fire prediction information. A screenshot of the interactive map is provided in Figure 5.8.

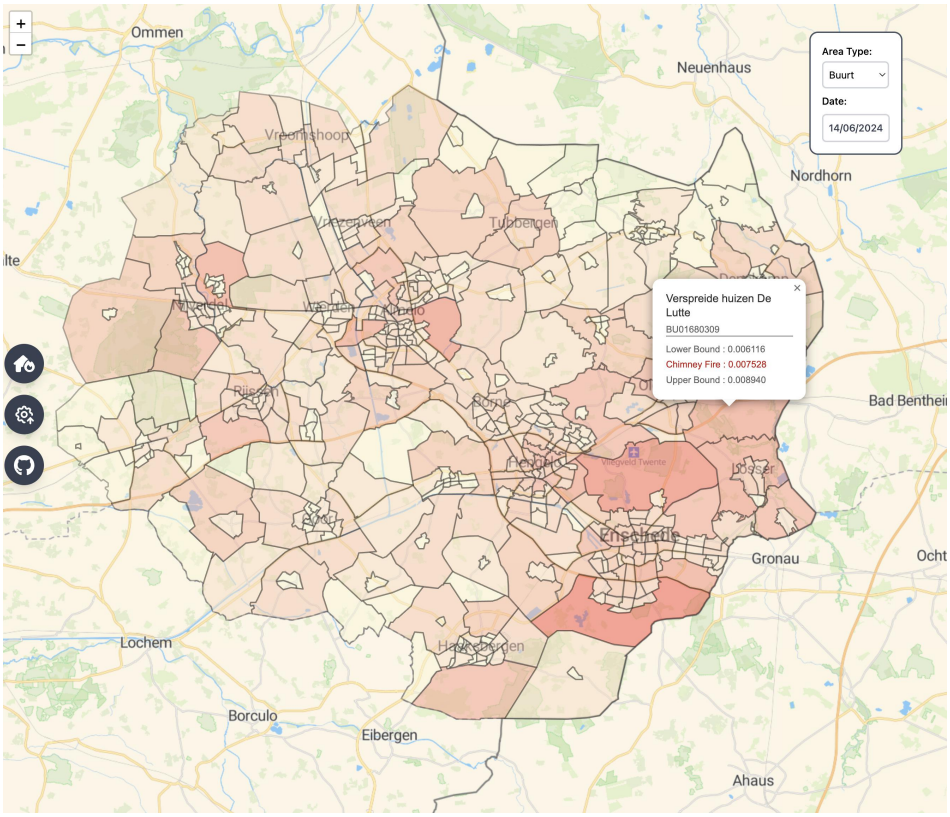


Figure 5.8 The Interactive Risk Map of the Front-End Application

The second page we created is the model refitting page, designed to guide the user through the necessary steps to update the model parameters. Initially, we provide a list of datasets as Excel files stored within the application, allowing users to download the desired file for modifications. To implement the file download

functionality, we added the endpoint "/api/model/files/:filename" to the back-end application, which returns the Excel file upon an HTTP GET request. After acquiring the Excel files, users can view the details of each file in a modal component. When modifying the Excel files, such as adding rows to "incident.xlsx" to include new chimney fire incident data, users are reminded not to change the file names or column names, as the data extraction process from Excel files to R data frames relies on these specific hardcoded names in the R scripts.

Next, we provide a "drag and drop" interface for users to upload the modified Excel files back to the application, utilizing the "/api/model/upload" API endpoint to replace the existing files. Finally, we created two buttons for users to execute R scripts via the "/api/model/count" and "/api/model/refit" API endpoints in the back-end application, which in turn read the modified Excel files and update the parameters of the predictive model.

To ensure transparency and provide feedback during these operations, we implemented a mechanism to monitor the server state. Using React Query, the front-end periodically sends requests to the "/api/model/state" endpoint, which returns information about the server state. This includes whether an R process is currently running on the server or if there was an error during the last R process execution. The server state is then communicated to the user through notifications, leveraging the React-Toastify library for real-time updates. This setup ensures that users are aware of the ongoing processes and any issues that may arise, enhancing the overall user experience. Screenshots illustrating these steps are provided in Appendix D.

Overall, with the addition of these front-end pages, the chimney fire prediction application has evolved from a back-end API into a comprehensive, web-based standalone product. This transformation ensures that users can easily access, visualize and update chimney fire risk data without requiring extensive technical knowledge. By integrating the interactive map and model refitting features into a single, cohesive interface, the application significantly enhances usability. The guided user interface eliminates the need for additional user manuals, making it straightforward for users to understand and utilize the tool effectively. This consolidation of features into one platform greatly simplifies the user experience, making the application a valuable asset for the risk management team of the TFB.

Deployment & Testing

6

In this chapter, we explain the deployment and testing tasks we conducted at the end of milestone implementation phases. Deploying our application involved preparing the application for end-users through containerization and cloud services, using a modern cloud platform such as Azure. On the other hand, testing was conducted iteratively to validate the implemented features, ensuring the desired functionality works as expected. In the following sections, we provide an in-depth exploration of these crucial activities.

6.1 Deployment

First, we delve into an often overlooked aspect of the application development process: how we deploy the application and make it available to end users. A simplified yet comprehensive ArchiMate view that captures this phase is illustrated in Figure 6.1.

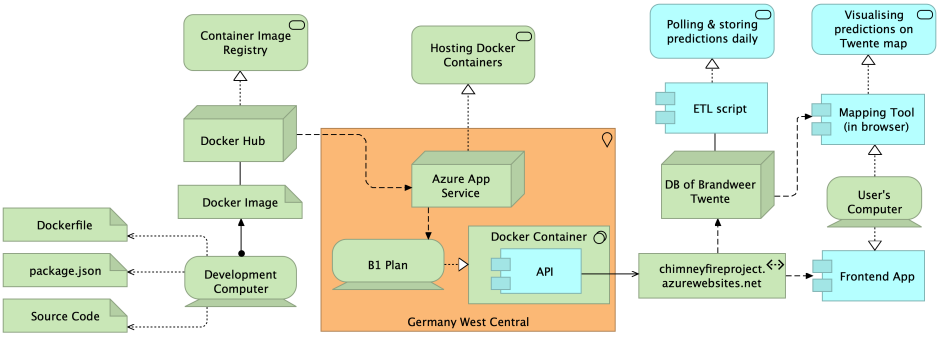


Figure 6.1 ArchiMate Deployment View

Throughout the implementation phase, we produce three key artifacts: the actual source code of the application features in JavaScript, the package.json file that specifies the application's dependencies, and a Dockerfile that contains Docker commands to create a Docker image on our local development computer. This Docker image serves as a blueprint for creating a Docker container, a lightweight and executable software package that provides the environment for our application to run.

Once the Docker image is created, it is uploaded to Docker Hub, a central repository for Docker images. For deployment, we create an Azure App Service on the Azure cloud platform, effectively reserving computing resources from an Azure data center located in Germany West Central. We initially opted for the basic B1 plan, offering 1.75 GB of memory, which is sufficient for daily tasks such as fetching weather data and performing chimney fire predictions using simple mathematical operations. However, during memory-intensive model refitting processes that execute R scripts, the users can scale up to a B2 or B3 plan, providing 3.50 GB and 7 GB of memory, respectively [24].

With control over our computing resources, we instantiate our Docker container, making the API accessible at an address such as "chimneyfireproject.azurewebsites.net". An ETL script subsequently makes the necessary API requests to this address to retrieve daily predictions and store them in the database of the TFB. For displaying the predictions to the risk management team, a GIS tool accesses this database and visualizes the predicted number of chimney fires per area on an interactive map. Additionally, the interactive map implemented as part of the front-end application can be utilized for the same task of interactive data visualization.

This deployment architecture ensures that the application is consistently and reliably delivered to end users. By leveraging containerization and cloud services, we provide a manageable environment that meets the operational needs of the Twente Fire Brigade.

6.2 Testing

As we explained in Section 4.1, our aim was to embrace the agile mindset and test the implementation of features at the end of each milestone before starting to implement the next set of features. This iterative approach ensured that any issues were identified and addressed promptly, maintaining the quality and reliability of the application throughout its development. In the following subsections, we detail the testing processes for each milestone, highlighting the specific features and functionalities verified at each stage. Additionally, we discuss the implementation of automated snapshot tests, which provide a robust mechanism for detecting unintended changes in the application's behavior and ensuring consistent output across different scenarios.

6.2.1 Testing Milestone 1

At the end of Milestone 1, we tested the application locally to ensure that it could generate predictions based on the historical weather data for the Twente region. This milestone involved validating the query parameters and calculating the predicted number of chimney fires using spatial and temporal terms derived from historical data. We verified that the API could handle requests with valid date and area code parameters, return predictions in JSON format and include the necessary GeoJSON geometry information for visualization. By conducting extensive local testing, we confirmed that the prototype functioned correctly, providing the expected prediction results based on historical weather data.

6.2.2 Testing Milestone 2

For Milestone 2, we focused on testing the application's capability to use up-to-date weather forecasts and enhance the efficiency of spatial data processing. Using the Azure account of the Twente Fire Brigade, we deployed the updated application and verified its functionality in a real-world environment. The key features tested included fetching weather data from the Meteoserver API, calculating temporal terms and processing API requests in under one second due to the pre-calculated spatial terms stored in JSON format. The TFB successfully fetched prediction data via the ETL script, saved them to their databases and visualized the predictions using their mapping tool, as shown in Figure 6.2 and Figure 6.3 for the neighbourhoods and CBS boxes. This testing phase validated that the application could provide accurate and timely predictions based on current weather data, significantly improving its prediction accuracy and operational readiness.

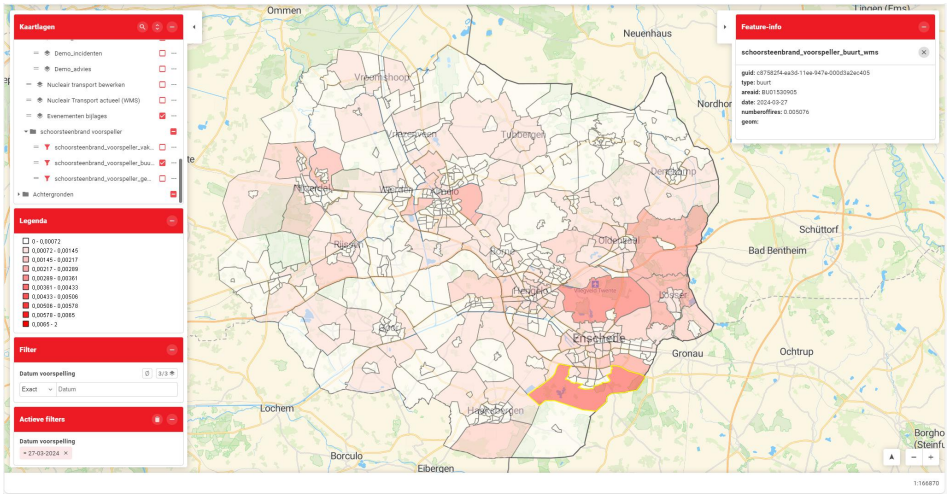


Figure 6.2 GIS tool with Prediction Data for Neighbourhoods

6.2.3 Testing Milestone 3

The primary focus of Milestone 3 was to implement and test the model refitting processes. This milestone involved updating the spatial and temporal terms of the prediction model using the existing dataset present in the application, inside the Docker container. We tested the new API endpoints designed to trigger these R scripts, ensuring they executed correctly and handled the model training datasets as intended. A crucial aspect of this testing phase was to confirm that the Docker container was correctly configured with the necessary low-level libraries required by the spatial data processing libraries present in the R scripts. By thoroughly testing the execution of these scripts within the Docker environment, we ensured that the container included all dependencies and R scripts functioned correctly inside this containerised environment. The successful implementation and testing of these processes demonstrated the robustness of the Docker setup and the application's ability to handle parameter updates when prompted by the user.

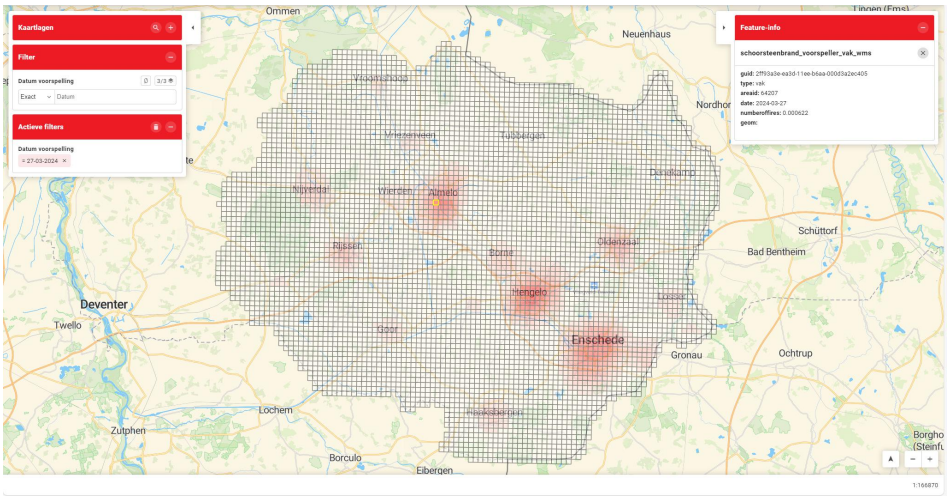


Figure 6.3 GIS tool with Prediction Data for CBS Boxes

6.2.4 Testing Milestone 4

In the final milestone, Milestone 4, we conducted comprehensive testing to verify the full functionality of the application, including the new feature to upload and replace Excel data files. We performed manual API testing to ensure that the new endpoints for file uploads worked as expected. To verify that the Excel files were correctly replaced, we accessed the Docker container and checked the modification dates of the uploaded files. Additionally, we tested the front-end application, focusing on the file upload process and the subsequent model refitting operations. By using the front-end interface, we confirmed that users could successfully upload new data files, initiate the model refitting process, and receive real-time updates on the server state. This testing phase ensured that all features were integrated seamlessly and that the application provided a user-friendly and reliable interface for managing and updating prediction data.

Through these iterative testing phases, we ensured that each milestone was thoroughly validated before proceeding to the next, maintaining the quality and robustness of the chimney fire prediction application. The successful completion of these tests demonstrated the application's readiness for deployment and use by the TFB, providing them with a valuable tool for risk management and operational planning.

6.2.5 Automated Snapshot Tests

After implementing and testing Milestone 4, we introduced automated snapshot tests using the Japa testing framework [14] to ensure the reliability and correctness of our API endpoints. These tests treat our application as a black box, providing controlled inputs, such as weather data, house count and model coefficients, and comparing the actual outputs with expected outputs, without concern for the internal workings of the application.

This testing method involves comparing the output of a function against a previously saved "snapshot" to detect unintended changes in behavior. For our ap-

plication to consistently produce the same chimney fire predictions during testing, we needed to control the weather input data. We achieved this by employing the dependency injection pattern to modify the application's weather fetching behavior during testing. While the production environment uses the Meteoserver API for real-time weather forecasts, our testing environment uses two predefined JSON files containing simulated weather data. These files represent typical cold weather (five days in January) and hot weather (five days in June), ensuring consistent weather inputs across different test runs.

Our tests cover both aggregate and single prediction functionalities. For aggregate prediction tests, we verify that each endpoint returns the correct number of areas for city, district, neighbourhood and box types. For single prediction tests, we examine individual area predictions for different area types and multiple specific area IDs, verifying the structure and content of the API response.

For each simulated weather forecast, we generated predictions for five specific areas. We then saved the API responses as JSON files, effectively creating ten snapshots, one for each combination of weather type and a specific area. During test execution, our tests compare the actual API responses against these pre-saved snapshots. This approach ensures that the prediction algorithm consistently produces the expected results across weather forecast scenarios and for different areas.

These snapshot tests also serve as regression tests, allowing IT team members at the Twente Fire Brigade to quickly identify any unintended changes in the prediction algorithm's output when modifying the application in the future. By running these tests after introducing changes to the implementation, they can ensure that the core functionality and the structure of the API responses remain consistent.

Conclusion and Future Work

7

7.1 Conclusion

In this thesis, we presented the successful development and deployment of an innovative chimney fire prediction application for the Twente Fire Brigade. By leveraging advanced mathematical modeling techniques and modern software engineering practices, we created a powerful tool that significantly enhances the Twente Fire Brigade's risk management capabilities and supports their mission to protect the community from the effects of chimney fires.

The application's development was driven by the strategic objectives outlined in Section 2.2. By combining the expertise of the Stochastic Operations Research group at the University of Twente with the operational needs of the TFB, we aimed to create a data-driven solution that would provide actionable insights for effective resource allocation and public awareness campaigns. The close collaboration between academia and the fire brigade throughout the project ensured that the application was tailored to their specific requirements and constraints.

The mathematical model developed by Lu et al.[22] formed the foundation of our application. By integrating machine learning techniques for variable selection and point process tools for model development, their research provided a fully data-driven and interpretable model for predicting chimney fire risk. Our work focused on transforming this research output into a practical and maintainable software solution that could be deployed on and integrated into the Twente Fire Brigade's existing IT infrastructure.

Through an iterative development process, we progressively enhanced the application's capabilities across four distinct milestones. Starting with a prototype that relied on historical data, we gradually incorporated real-time weather forecasts, optimized spatial data processing and introduced confidence intervals to improve the accuracy and reliability of the predictions. The final version of the application allows users to update the model parameters using the latest data, ensuring its continued relevance and effectiveness over time.

The development of a user-friendly front-end application further enhanced the accessibility and usability of the chimney fire prediction tool. By providing an interactive map and a guided interface for model refitting, we empowered the risk management team to easily visualize and interact with the prediction data, eliminating the need for extensive technical knowledge.

The successful deployment of the application on the Azure cloud platform, using containerization technologies like Docker, ensures its maintainability, monitoring and integration with the Twente Fire Brigade's existing systems. The application's use of industry standards and comprehensive documentation facilitate future updates and extensions, allowing the TFB to adapt to evolving requirements.

In conclusion, the chimney fire prediction application developed in this thesis represents a significant advancement in data-driven risk management for fire services. By utilizing the power of mathematical modeling, software engineering and close collaboration between academia and public safety organizations, we have created a valuable tool that supports the TFB in their mission to protect lives, property and the environment. The application's successful implementation demonstrates the potential of such interdisciplinary projects to drive innovation and enhance public safety.

7.2 Future Work

The successful research and implementation of the chimney fire prediction application for the Twente Fire Brigade has opened up several opportunities for future research and enhancements. A recent study by Van Kasteren [17] applied similar techniques to predict chimney fire risk in the IJsselland region, yielding comparable results to those found in the research by Lu et. al. [22] in Twente. The findings and insights from this new research could be integrated into our application, enabling it to generate chimney fire risk predictions for the IJsselland region as well. This extension would provide valuable support to the IJsselland Fire Brigade in their risk management efforts and demonstrate the adaptability of our approach to different geographical contexts.

Furthermore, the Safety Region North and East Gelderland (VNOG)[36] has expressed interest in applying the chimney fire prediction model to their region and potentially to the other two Gelderland safety regions. Collaboration between the researchers and data scientists from VNOG could facilitate the execution of variable selection and model adaptation using data from these regions. This expansion would not only benefit the fire brigade in Gelderland but also validate the generalizability of our approach to diverse geographical areas and fire risk patterns.

The application could be enhanced with more advanced features to provide a more comprehensive and flexible risk assessment tool. One such feature is the ability to draw custom shapes on the interactive map or input shapefiles representing specific areas of interest. By allowing users to define custom areas that may not necessarily align with predefined administrative boundaries such as municipalities, districts, neighborhoods or 500 by 500 meter boxes, the application could provide more targeted risk assessments. The system could then calculate and display the predicted number of chimney fire incidents within the drawn or provided area. While this functionality was not implemented in the current version due to time constraints, it presents a valuable path for future development, enabling the TFB to analyze risk at even more granular and flexible level and support decision-making in specific localities.

By expanding the application's geographical scope and incorporating advanced features, increasingly powerful and versatile tools for chimney fire risk management can be created. Through ongoing research and development, this application has the potential to become an indispensable asset for fire brigades across the Netherlands and beyond, supporting their efforts to protect communities and save lives.

Bibliography

[1] V. Agafonkin. Leaflet - an open-source JavaScript library for mobile-friendly interactive maps. https://leafletjs.com/. Retrieved July 8, 2024.

[2] A. Baddeley, E. Rubak, and R. Turner. *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press, London, 2015.

[3] K. Beck, M. Fowler, D. Thomas, R.C. Martin, et al. Manifesto for Agile software development. https://agilemanifesto.org/, 2001.

[4] M. Borchers, T. Gellenbeck, G. Maduro, M. Visser, and S. Visser. Op zoek naar het operatiecentrum van autobrandstichters in Twente. Datafiltering en lokalisatie door middel van Rossmo's model. BSc thesis, University of Twente, 2013.

[5] Buienradar Weather Forecast API. https://www.buienradar.nl/overbuienradar/gratis-weerdata. Retrieved July 5, 2024.

[6] D. Crockford. Json - JavaScript Object Notation. https://www.json.org/json-en.html. Retrieved July 5, 2024.

[7] Docker. https://www.docker.com/. Retrieved July 5, 2024.

[8] Dutch Institute of Public Safety. https://nipv.nl/over-nipv/. Retrieved July 5, 2024.

[9] EPSG.io. Amersfoort RD new coordinate system. https://epsg.io/28992. Retrieved July 5, 2024.

[10] Equinix. https://www.equinix.com/about. Retrieved July 5, 2024.

[11] ESRI. Shapefiles. https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm. Retrieved July 5, 2024.

[12] Facebook Open Source. React – a JavaScript library for building user interfaces. https://reactjs.org/. Retrieved July 8, 2024.

[13] R.J. Hijmans, J. van Etten, M. Sumner, et al. raster: Geographic data analysis and modeling. https://cran.r-project.org/web/packages/raster/index.html. Retrieved July 5, 2024.

[14] Japa Testing Framework. https://japa.dev/docs/introduction. Retrieved July 23, 2024.

[15] M. de Jongh. Inzet brandweerauto's bij calamiteiten. BSc thesis, University of Twente, 2018.

[16] Kadaster. Basic Registration of Addresses and Buildings. https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag. Retrieved July 5, 2024.

[17] L.N. van Kasteren. Chimney fire prediction in IJsselland. BSc thesis, University of Twente, 2024.

[18] J.F.C. Kingman. *Poisson processes*, volume 3. Clarendon Press, 1992.

[19] R. Kozlowski. How to calculate wind chill factor. https://sciencing.com/calculate-wind-chill-factor-5981683.html, 2020. Retrieved July 5, 2024.

[20] Y. Kraakman. Optimale plaatsing van een nieuwe brandweerkazerne in regio Tubbergen. BSc thesis, University of Twente, 2018.

[21] T. Linsley. Tanstack Query - hooks for fetching, caching and updating asynchronous data in React. https://tanstack.com/query/v4. Retrieved July 8, 2024.

[22] C. Lu, M.N.M. van Lieshout, M. de Graaf, and P. Visscher. Data-driven chimney fire risk prediction using machine learning and point process tools. *The Annals of Applied Statistics*, 17(4):3088 – 3111, 2023.

[23] MeteoServer Weather Forecast API. https://meteoserver.nl/weersverwachting-API.php. Retrieved June 6, 2024.

[24] Microsoft Azure. App Service Pricing. https://azure.microsoft.com/en-us/pricing/details/app-service/windows/. Retrieved July 5, 2024.

[25] Microsoft Azure. Azure App Service - create enterprise-ready web apps in the cloud. https://azure.microsoft.com/en-us/services/app-service/. Retrieved July 5, 2024.

[26] J. Møller, A. R. Syversveen, and R. P. Waagepetersen. Log Gaussian Cox processes. *Scandinavian journal of statistics*, 25(3):451–482, 1998.

[27] A.B.M. Moniruzzaman and S.A. Hossain. Comparative study on Agile software development methodologies. *CoRR*, abs/1307.3356, 2013.

[28] Netherlands Organisation for Scientific Research. https://www.nwo.nl/en. Retrieved July 9, 2024.

[29] OpenJS Foundation. Express.js - minimalist web framework for Node.js. https://expressjs.com/. Retrieved July 5, 2024.

[30] OpenJS Foundation. Node.js - run JavaScript everywhere. https://nodejs.org/. Retrieved July 5, 2024.

[31] OpenWeatherMap Weather Forecast API. https://openweathermap.org/forecast16. Retrieved July 5, 2024.

[32] E. Pebesma and R. Bivand. *Spatial Data Science: With applications in R*. Chapman and Hall/CRC, 2023.

[33] Remix. React Router - declarative routing for React. https://reactrouter.com/. Retrieved July 8, 2024.

[34] B.D. Ripley. Modelling spatial patterns. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):172–192, 1977.

[35] Royal Dutch Meteorological Institute. https://www.knmi.nl/over-het-knmi/about. Retrieved July 5, 2024.

[36] Safety Region North and East Gelderland. https://www.vnog.nl. Retrieved July 8, 2024.

[37] M.L. School. A log-Gaussian Cox process for predicting chimney fires at fire department Twente. MSc thesis, University of Twente, 2018.

[38] Statistics Netherlands. https://www.cbs.nl/en-gb/about-us. Retrieved July 5, 2024.

[39] The Open Group. Archimate® 3.1 specification. https://publications.opengroup.org/standards/archimate, 2019. Retrieved July 8, 2024.

[40] The R Foundation. The R project for statistical computing. https://www.r-project.org/, 2024. Retrieved July 8, 2024.

[41] A. Wathan. Tailwind CSS - rapidly build modern websites without ever leaving your HTML. https://tailwindcss.com/. Retrieved July 8, 2024.

[42] M. Wendels. A spatio-temporal point process model for firemen demand in Twente. BSc thesis, University of Twente, 2017.

API Endpoints

A

A.1 Aggregate Area Prediction Endpoints

- **GET /api/prediction/gemeente**
  - Returns fire predictions for all municipalities (gemeente).
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/wijk**
  - Returns fire predictions for all neighborhoods (wijk).
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/buurt**
  - Returns fire predictions for all blocks (buurt).
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/box**
  - Returns fire predictions for all 500x500 meter boxes.
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.

A.2 Individual Area Prediction Endpoints

- **GET /api/prediction/gemeente/:gemeenteId**
  - Returns fire prediction for a specific municipality (gemeente) identified by CBS code.
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/wijk/:wijkId**
  - Returns fire prediction for a specific neighborhood (wijk) identified by CBS code.
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/buurt/:buurtId**

  - Returns fire prediction for a specific block (buurt) identified by CBS code.
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.
- **GET /api/prediction/box/:boxId**
  - Returns fire prediction for a specific 500x500 meter box identified by CBS code.
  - Optional query parameter: `includeGeoInfo=true` to include geographic information.

A.3 Model Refitting Endpoints

- **PUT /api/model/upload**
  - Uploads the Excel files used for model refitting.
  - Request body: `multipart/form-data` with key `excelFiles` containing the four Excel files.
- **POST /api/model/count**
  - Triggers the application to read the new building data from `kro.xlsx`.
- **POST /api/model/refit**
  - Triggers the application to read all the four Excel files and re-calculate the model parameters.
- **GET /api/model/files**
  - Returns the metadata about the Excel files stored in the application.
- **GET /api/model/files/:filename**
  - Returns the requested Excel file to be downloaded by the browser.
- **GET /api/model/state**
  - Returns the server state in terms of R processes running/ran previously.

A.4 Front-End Application Routes

- **GET /**
  - Serves the HTML, CSS and JavaScript code of the Single Page Application (SPA) using React.
  - Shows the interactive chimney fire risk map.
- **GET /model-refit**
  - Handled by React Router in a SPA style, providing navigation without a full page reload.
  - Shows the steps related to model refitting within the front-end application.

Example API Responses

B

B.1 Milestone 1

A typical example API request and response in Milestone 1 version of the chimney fire prediction application are given in Listings B.1 and B.2. Note that the array of coordinates in the ”geometry” property of the API response is kept short for demonstration purposes.

Listing B.1 Example API Request in Milestone 1

```
https://chimneyfireproject.azurewebsites.net/prediction?areaCode=
    GM0164&date=2023-01-01
```

Listing B.2 Example API Response in Milestone 1

```
{
  "areaCode": "GM0164",
  "date": "2023-01-01T00:00:00.000Z",
  "predictedFires": 0.07389275859749127,
  "geoInfo": {
    "type": "Feature",
    "crs": {
      "type": "name",
      "properties": {
        "name": "urn:ogc:def:crs:EPSG::28992"
      }
    },
    "properties": {
      "id": 114,
      "fid": 114,
      "gemeenteco": "GM0164",
      "gemeentena": "Hengelo",
      "jaarstatco": "2021GM0164",
      "jaar": 2021
    },
    "geometry": {
      "type": "MultiPolygon",
      "coordinates": [
        [
          [
            [251978.591, 481220.258],
```

```
              [251979.382, 481218.495],
              [251983.707, 481220.19]
            ]
          ]
        ]
      }
    }
}
```

B.2 Milestone 2

A typical example API request and response in Milestone 2 version of the chimney fire prediction application are given in Listings B.3 and B.4.

Listing B.3 Example API Request in Milestone 2

```
https://chimneyfireproject.azurewebsites.net/prediction/gemeente/
    GM0153
```

Listing B.4 Example API Response in Milestone 2

```
{
    "lastWeatherFetchTimestamp": "17-04-2024 15:15:13",
    "predictions": {
        "areaId": "GM0153",
        "prediction": [
            {
                "date": "17-04-2024",
                "numberOfFires": "0.064677",
                "lowerBoundOfFires": "0.061271",
                "upperBoundOfFires": "0.068083"
            },
            {
                "date": "18-04-2024",
                "numberOfFires": "0.063505",
                "lowerBoundOfFires": "0.060078",
                "upperBoundOfFires": "0.066931"
            },
            {
                "date": "19-04-2024",
                "numberOfFires": "0.069518",
                "lowerBoundOfFires": "0.065613",
                "upperBoundOfFires": "0.073422"
            },
            {
                "date": "20-04-2024",
                "numberOfFires": "0.081339",
                "lowerBoundOfFires": "0.076496",
                "upperBoundOfFires": "0.086181"
            },
            {
```

```
                "date": "21-04-2024",
                "numberOfFires": "0.074268",
                "lowerBoundOfFires": "0.069827",
                "upperBoundOfFires": "0.078709"
            },
            {
                "date": "22-04-2024",
                "numberOfFires": "0.086056",
                "lowerBoundOfFires": "0.080469",
                "upperBoundOfFires": "0.091644"
            },
            {
                "date": "23-04-2024",
                "numberOfFires": "0.071638",
                "lowerBoundOfFires": "0.067111",
                "upperBoundOfFires": "0.076164"
            },
            {
                "date": "24-04-2024",
                "numberOfFires": "0.058141",
                "lowerBoundOfFires": "0.054491",
                "upperBoundOfFires": "0.061790"
            },
            {
                "date": "25-04-2024",
                "numberOfFires": "0.057063",
                "lowerBoundOfFires": "0.053378",
                "upperBoundOfFires": "0.060747"
            },
            {
                "date": "26-04-2024",
                "numberOfFires": "0.067903",
                "lowerBoundOfFires": "0.063232",
                "upperBoundOfFires": "0.072574"
            },
            {
                "date": "27-04-2024",
                "numberOfFires": "0.097660",
                "lowerBoundOfFires": "0.086680",
                "upperBoundOfFires": "0.108641"
            }
        ],
        "geoInfo": {
            "type": "Feature",
            "crs": {
                "type": "name",
                "properties": {
                    "name": "urn:ogc:def:crs:EPSG::28992"
                }
            },
            "properties": {
                "id": 110,
```

```
                "fid": 110,
                "gemeenteco": "GM0153",
                "gemeentena": "Enschede",
                "jaarstatco": "2021GM0153",
                "jaar": 2021
            },
            "geometry": {
                "type": "MultiPolygon",
                "coordinates": [
                    [
                        [
                            [
                                259057.888,
                                478571.251
                            ],
                            [
                                259074.141,
                                478513.545
                            ],
                            [
                                259097.161,
                                478429.967
                            ]
                        ]
                    ]
                ]
            }
        }
    }
}
```

Structure of the Data Used for Model Refitting

C

The application relies on four specific Excel files for refitting the prediction model. These files must have exact names and contain particular columns with specified values to ensure proper functionality. Below are the details of each file:

C.1 Building Data

The file "kro.xlsx" in the source code of the application contains building data. The necessary columns are:

- **bouwjaar**: The year the building was built.
- **status**: The status of the building, indicating whether it is currently in use. The application uses rows where the status is one of the following values:
  - Pand in gebruik
  - Pand in gebruik (niet ingemeten)
- **gebrklasse**: The usage class of the building. The application differentiates between free-standing or semi-detached houses and other types. The relevant values are:
  - 2 onder 1 kap doelgroepwoning
  - 2 onder 1 kap recreatiewoning
  - 2 onder 1 kap woning
  - Vrijstaande doelgroepwoning
  - Vrijstaande recreatiewoning
  - Vrijstaande woning
- **x**: The x-coordinate of the building in the RD Amersfoort coordinate system.
- **y**: The y-coordinate of the building in the RD Amersfoort coordinate system.

C.2 Chimney Fire Incident Data

The file "incident.xlsx" in the source code of the application contains data on chimney fire incidents. The necessary columns are:

- **Year**: The year of the incident.

- **Day**: The day of the incident, from 1 to 365.
- **Xcoordinate**: The x-coordinate of the incident location in the RD Amersfoort coordinate system.
- **Ycoordinate**: The y-coordinate of the incident location in the RD Amersfoort coordinate system.
- **bouwjaar**: The year the building was built.
- **gebrklasse**: The usage class of the building. The relevant values are the same as those mentioned in `kro.xlsx`.

C.3 Wind Chill Data

The file "windchill.xlsx" in the source code of the application contains wind chill data. The necessary structure is:

- The first row should contain the column names (e.g., `WC2004`, `WC2005`).
- Each column, from the second row to the 366th row, should contain the average wind chill value for the corresponding day of the year.
- Each cell should use a decimal comma as the decimal separator and be calculated using the formula:

$$\text{Wind chill} = 13.12 + 0.6215T - 11.37V^{0.16} + 0.3965TV^{0.16}$$

  where:
  - **T**: The temperature in Celsius on that day.
  - **V**: The wind speed in km/h on that day.

C.4 Wind Speed Data

The file "windspeed.xlsx" in the source code of the application contains wind speed data. The necessary structure is:

- The first row should contain the column names (e.g., `FG2004`, `FG2005`).
- Each column, from the second row to the 366th row, should contain the average wind speed value for the corresponding day of the year.
- Each cell should use a decimal comma as the decimal separator, and the wind speed should be in km/h.

Screenshots of Model Refit Page of the Front-End Application

D

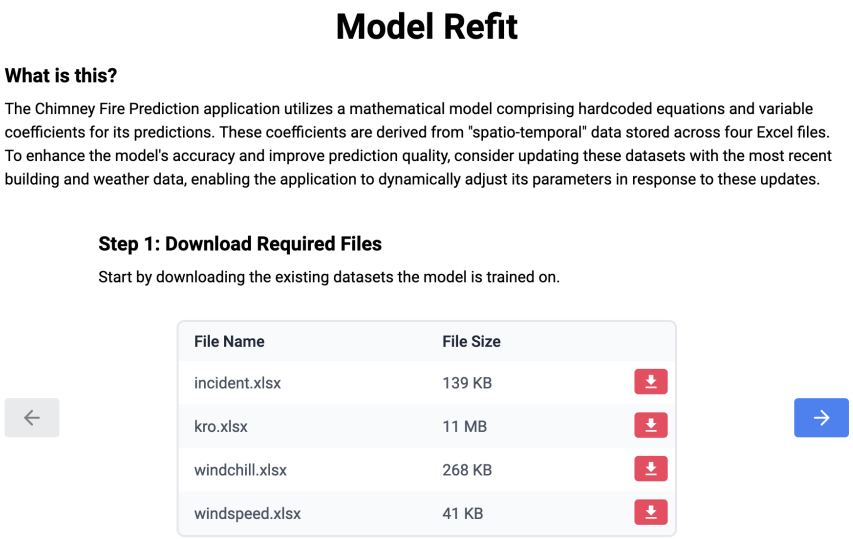


Figure D.1 Step 1 of Model Refit Page

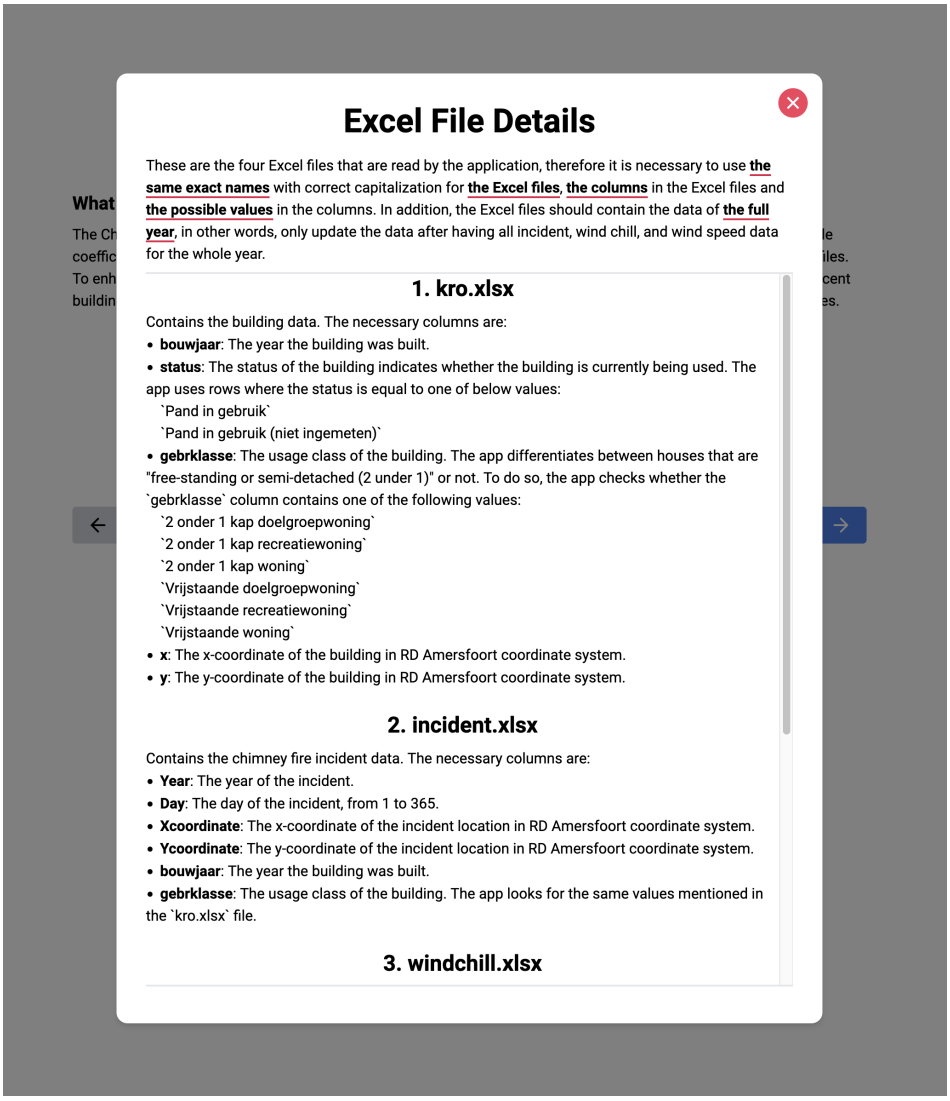


Figure D.2 Step 2 of Model Refit Page

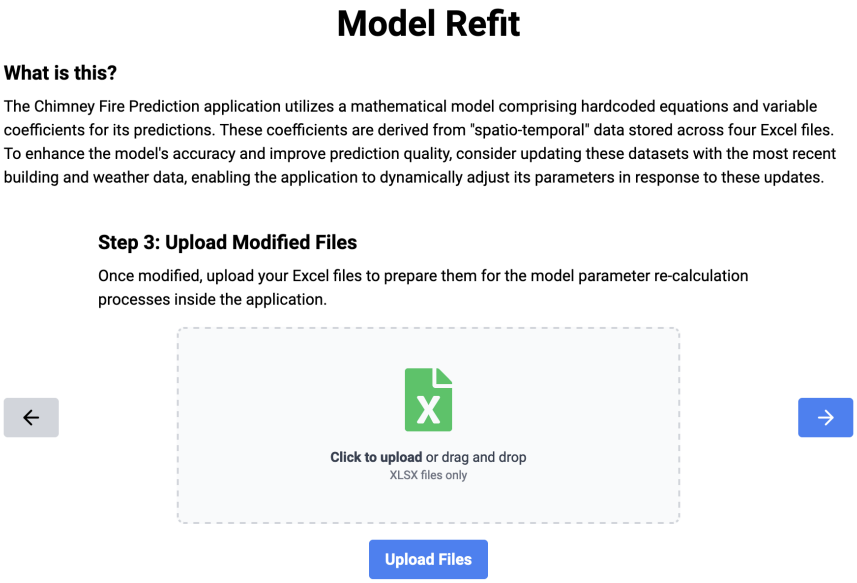


Figure D.3 Step 3 of Model Refit Page

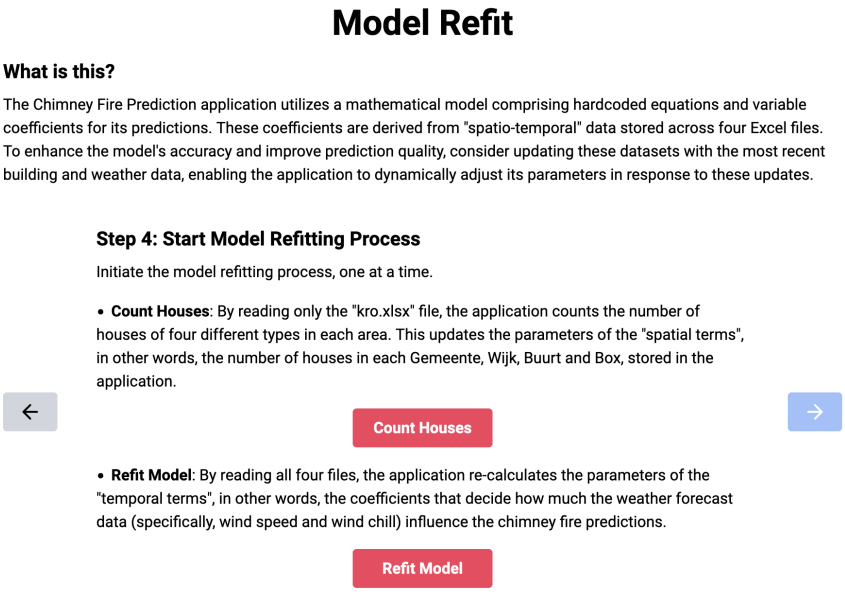


Figure D.4 Step 4 of Model Refit Page

List of Figures

E

List of Acronyms

F

API	Application Programming Interface
BAG	Basisregistratie Adressen en Gebouwen (Basic Registration of Addresses and Buildings)
BI	Business Intelligence
CBS	Centraal Bureau voor de Statistiek (Statistics Netherlands)
CI	Confidence Interval
CSS	Cascading Style Sheets
EngD	Engineering Doctorate
ETL	Extract, Transform, Load
GIS	Geographic Information System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IT	Information Technology
JSON	JavaScript Object Notation
KNMI	Koninklijk Nederlands Meteorologisch Instituut (Royal Netherlands Meteorological Institute)
KRO	Kernregistratie Objecten (Key Register Objects)
NIPV	Nederlands Instituut Publieke Veiligheid (Netherlands Institute for Public Safety)
NWO	Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Netherlands Organisation for Scientific Research)
RAM	Random Access Memory
RD	Rijksdriehoek (Dutch National Triangulation System)
TFB	Twente Fire Brigade
VNOG	Veiligheidsregio Noord- en Oost-Gelderland (Safety Region North and East Gelderland)