Predicting the function recovery time for railway incidents

Merel Wemelsfelder

merel@wemel.nl

May 2019

Bachelor thesis

Bsc Artificial Intelligence, Utrecht University

15 ECTS

Supervisors

Dr. T.B. Klos · UU

Dr. R.M. van Lambalgen · UU

W.L. Tielman · ProRail

Abstract

When an incident is reported at ProRail, many decisions have to be made. All those decisions influence the function recovery time (FHT), which is the time span between the moment an incident is reported and the moment everything is restored. When the FHT can be accurately predicted, better decisions will be made with regard to incident handling, and delayed train passengers can be accurately informed. The aim of this research is to build a supervised machine learning model to predict the FHT. Currently, a Decision Tree is implemented at ProRail to produce an initial prognosis. Ideally, our model will be suitable for both generating an initial prognosis, and updating this prognosis during the recovery process. Historical incident data of ProRail will be used to train the model on. This data is extended by adding a few other data sets, to increase the number of relevant variables. The resulting data set is divided by incident type, because the variables that correlate with the FHT are very different between incident types. For each type, a different set of features is selected, and a different model is trained. The models have to meet three requirements. First, all of our variables are or can be discrete, so the model should be able to work with categorical data. Second, when the initial prognosis is made, usually the values of only a few variables are known, so the model should be able to handle this, and also be able to make a better prediction based on more values later in the process. Third, the model will be used to support humans in making choices, so the output of the model should be easily interpretable for a human. As a result, Bayesian Networks and the k-Nearest-Neighbors algorithm are chosen to be implemented. These algorithms are tested on data sets for rolling stock incidents, section TOBS incidents and collision/hindrance incidents. The mean distance between the predicted FHTs and the actual FHTs is similar for both algorithms; between 45 minutes and one hour for all three data sets. This performance is compared with the predictions of the currently implemented Decision Tree, using a small test set. On this set, our models do perform similarly or better.

0 Contents

1 Introduction

On an average working day, more than a million people in the Netherlands travel by train [26]. In the media, ProRail even claims that the number of train passengers will increase by 40% in the next ten years [44]. The occurrence of an incident on the rail tracks can cause a lot of annoyance among passengers. A customer survey showed that passengers already experience heavy nuisance from a 15 minute delay [32]. ProRail, the infrastructure managing company of the Dutch railways, and the train companies of the Netherlands (e.g. NS and Arriva) are always working on both the prevention of incidents and the improvement of incident handling. The latter can be seen as consisting of two parts: reducing passenger hindrance by minimizing delays, and accurately informing the delayed passengers so they can keep control over their own journey.

The moment an incident is reported at the Railway Alarm Room (MKS), a decision has to be made when to start the process of recovery. This can either be done immediately, at a later time that day (after rush hour) or at night. This choice can have a huge impact on the delays that are caused by the incident, and is therefore an important factor in minimizing delays. Among other things, this choice depends on the diagnosis (what needs to be fixed), priority (how urgently does it need to be fixed), time of day (is it during rush hour) and the prognosis (estimate of the time needed for recovery) [3].

Before and during the recovery process of an incident, a lot of trains have to be rescheduled. Some will be cancelled, others will be re-routed, and others might still use the same tracks while riding very slowly. When the problem is solved and the train table can get back to normal, the train traffic controller (TRDL), who is responsible for a specific part of the rail infrastructure, has to notify the involved train company. At that point, the train company has to arrange the needed rolling stock and train personnel, and reschedule the train table. Doing this in such a way that hindrance is the lowest, takes at least 30 minutes [3]. This means that, if the train company gets a notification at the moment the incident is already recovered, about 30 minutes of unnecessary delays are caused. Therefore, if the TRDL can use the prognosis to notify the train company at least 30 minutes before the moment of recovery, this contributes a lot to the minimization of delays.

Customer experience is not only influenced by minimizing passenger delays, but also by the accuracy of information that is communicated to the delayed passengers. This is why ProRail is concerned with the improvement of information provision for delayed passengers, as is mentioned in the management plans of ProRail for the years 2016 and 2018 [32, 33]. For both minimizing delays and accurately informing delayed passengers, it is necessary to make an accurate prediction of the function recovery time (FHT). As is shown in figure 1.1, this is the time span between the moment an incident is reported, and

function recovery, the moment everything is restored. These moments differ from 'start incident' and 'end incident', in that the former is the moment the incident actually takes place, before it is reported to the MKS, and the latter is the moment that the train table *is* back to normal, which takes time after this is theoretically possible. The FHT can be split up into latency time and repair time, which are respectively the time span between the moment an incident is reported and the arrival of the contractor, and between arrival of the contractor and the function recovery moment.

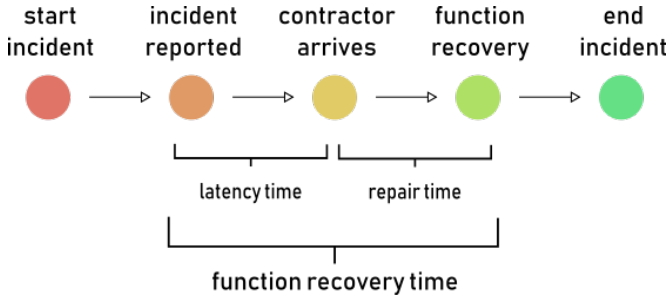

Figure 1.1: Function recovery time

Humans seem quite capable of making predictions, based on their experience with past incidents. However, in the case of decisions from experience, people tend to rely on small samples of the information and tend to overweight recently sampled information [16]. Besides, our ability to reflect on how we perform is very bad [5], so the risk of unknowingly being influenced by irrelevant factors is very high. To prevent this from happening, the assistance of artificial predictive models comes into play, to make predictions more stable and based on years of equally weighted experience. This model will, at least in the near future, not be able to make decisions based on these predictions, so this will remain a human's job. Therefore, the output of the predictive model should be easy to interpret by the human, to be of effective support.

At this moment, a Decision Tree is used for this purpose. This model is designed by the consultancy company CQM [9]. For each type of incident (e.g. train defect, signal failures, person on the rail tracks, ...), a different Decision Tree is applied. In each node of the Decision Tree, a choice is made between two or more categories, e.g. 'during rush hour' and 'not during rush hour'. This choice causes an alteration of the prognosis, based on the new information. Finally, in the leaves of the tree, this results in the initial prognosis. This model is currently applied just once: at the moment a disruption is reported to the MKS. During the recovery, an AL (General Leader) is present at the location of the incident, so when more information becomes available (e.g. failure cause), the AL communicates this to ICB (Incident Control), where the prognosis is updated. The Decision Tree of CQM is not suited for adding new information, because its nodes are fixed. When the value of a node is not known, the decision between the children of that node can not be made, so only the few nodes of which the value is known during the initial prognosis are included in the model. Therefore, the prognosis update, based on the information that is gained later in the recovery process, can not be done by the algorithm. It is done manually, using human experience and expertise.

Generating an accurate initial prognosis is very useful for deciding when to start recovery and for giving passengers a first indication of delays. However, it is very difficult to make an initial prognosis that is sufficiently accurate to communicate to train traffic controllers, to accordingly reschedule trains and minimize unnecessary delays. This is simply because some very relevant information, such as failure cause, is not yet known when the incident is reported to the MKS. This means that, in most cases, only updated prognoses can be accurate enough to be fundamental to train rescheduling. Considering that the Decision Tree is static due to its fixed nodes, a more dynamic model is needed to solve this problem.

This problem leads us to the main question of this thesis: How can we accurately predict the recovery time for various types of railway incidents, using a dynamic model that updates with newly gained information? This question can be divided into two subquestions. First, what information is needed to make accurate predictions of recovery time for various railway incidents? Second, how can we create a dynamic model that is able to update its prediction based on new information?

Machine learning models can be classified in three categories: supervised learning, reinforcement learning and unsupervised learning [1]. Supervised learning can be applied when the training data contains explicit examples of what the correct output should be for given inputs. In reinforcement learning, a training example does not contain a target output, but instead contains some possible output together with a measure of how good that output is. In contrast to supervised learning, where the training examples are of the form [input, correct output], the examples in reinforcement learning are of the form [input, some output, grade for this output]. In the unsupervised setting, the training data does not contain any output information at all. We can get similar clusters to those in supervised learning, but without the labels.

Our data does contain examples of what the correct output should be, the FHT. I therefore aim to build a supervised learning model, which will generate a predicted FHT based in relations between the FHT and other information (features) that we have on incidents. The setup of this research is inspired by the process of supervised machine learning, proposed by Kotsiantis et al. [22]. This process consists of seven steps, where steps 2-4 contribute to answering the first subquestion, and steps 5-7 to answering the second subquestion.

1. Defining the problem.
2. Identification of the required data. This is necessary because not all data that is owned by ProRail is relevant for our problem, and most probably not all information that is relevant is directly available.
3. Data pre-processing, which includes cleaning the data from missing values.
4. Defining the training set. It is very important to divide the data into a training set and a test set, to prevent the model from overfitting.
5. Algorithm selection, because there are various algorithms that all have their specific strengths and weaknesses.

6. Training the model, using the previously chosen training set.
7. Evaluating the algorithm, using the previously chosen test set.

Step 1, problem definition, is mostly discussed in this chapter, but will be further explained in chapter 2. Chapter 3 contains some information about the resources I have used, which is not included in the process of supervised learning, but seemed reasonable to mention somewhere. Step 2 is introduced in chapter 2 and fully discussed in chapter 4. Step 3 is spread out over chapters 5 and 6, where the first chapter is about reshaping the data to make it usable, and the second is about selecting the relevant features for our problem. Step 4 is also included in chapter 5. Chapter 7 is dedicated to step 5. In this chapter, six commonly used algorithms are discussed, and compared by their suitability for our data and our problem. Chapter 8 contains both step 6 and 7, because testing and evaluating the model goes hand in hand with the training procedure, which is continuously adjusted based on testing results. Chapter 9, the conclusion of this thesis, summarizes the algorithms and features found to perform best. In chapter 10, I discuss what I could have done differently and why, so future research can build on this.

2 Background and previous research

Figure 2.1: Switches [44]

2.1 About ProRail

ProRail is responsible for the Dutch railway network, which includes construction, maintenance, traffic management and safety. ProRail distributes the capacity of the tracks, manages all train traffic, and builds and manages stations, rail tracks, switches, signals and crossings. The operational maintenance of the rail tracks is not done by ProRail itself. For the execution of maintenance activities, ProRail has contracted companies like BAM Rail, Strukton Rail and Volker Rail. All contractors take care of the maintenance of a certain part of the railways. Train companies, like NS and Arriva "buy" railway capacity from ProRail to transport people and goods. Then passengers can in

turn pay the train company for using their trains as transport. ProRail B.V. is a private company, of which the Dutch government is 100% shareholder via Railinfratrust B.V.

ProRail has about 4300 employees (2017) [31]. The company manages 7021 km of rail tracks, 2589 crossings, 7071 switches, 12036 signals and 404 stations. The railways are connected by 725 viaducts, 455 bridges, 56 movable bridges and 15 tunnels. The management is done by 13 traffic control posts. The head office of ProRail is located in Utrecht and there are 4 regional offices, located in Amsterdam, Eindhoven, Rotterdam en Zwolle [30].

ProRail consists of many departments and hires people for many different functions. Because some of these department and function terms will be casually used throughout this thesis, the most important ones are shortly explained in Appendix A. Besides, the jargon at ProRail does not only consist of words one does not understand by common sense, but of many abbreviations as well. Both in the conversations people have, and in the data sets I will use for FHT prediction, those abbreviations are abundantly used. In this thesis, I hold on to the Dutch abbreviations, since they also occur in the variable names of our data sets. In Appendix A, I have also listed some useful ones with both their Dutch and English meaning.

2.1.1 Data on incidents

Considering the objective of this research, the main thing we are interested in is ProRail's database on incidents. At ProRail, incidents are recorded in various ways. The systems that contain the most relevant information for this research are Spoorweb, SAP and ISVL.

ISVL ISVL is a data set containing information that was recorded during the handling of incidents. It has a relatively free form, focusing on transferring useful information about the incident between concerned parties, rather than recording the incidents for future use. The ISVL data set that I used contains recordings of incidents from 2013/05/08 to 2017/06/25. In total, this captures 50,574 incidents and 41 variables.

Spoorweb Spoorweb is the newer version of ISVL. It is also used for exchanging information between parties, during the handling of an incident. It has improved relative to ISVL in that its variables are more structured. Besides, tasks are assigned more specifically to people, and it is possible to record who has done which task. The Spoorweb data set I used contains recordings of incidents from 2017/06/18 to 2018/12/12. In total, this captures 28,018 incidents and 153 variables.

SAP SAP is the recordings database of Asset Management (AM). This database is created and maintained with the purpose of creating historical data for later use, as opposed to the real-time use of ISVL and Spoorweb. The SAP data set contains recordings of incidents from 2006/01/01 to 2018/12/03. In total, this captures 95,761 incidents and 150 variables.

SAPX This database did not have a clear name, but its variable labels are much like those in SAP, as is part of its content. Therefore, I chose to name this data set SAPX. It contains recordings of incidents from 2005/01/01 to 2018/06/21. In total, this captures 124,363 incidents and 123 variables.

2.2 Previous research on feature recovery time

ProRail has currently outsourced the research on prediction of FHT (Function Recovery Time, figure 1.1) to the consultancy company CQM [9]. The model they created, and which they are still improving, is used to assist the MKS (Railway Alarm Room) at establishing an initial prognosis. The fact that CQM is still improving its model seems to conflict with the purpose of this research. However, the improvements of their model concern researching new features and adding them to the model. The use and shape of the model will stay the same, meaning that it will stay a static model which is not suitable for generating an updated prognosis, while this research focuses on being able to update prognoses whenever new information becomes available. In the past, a few other parties have also researched FHT prediction at ProRail. De Wit [46] and Bergsma [3] have written their master theses on this topic, respectively in 2016 and 2018. In 2017, Zilko [50] has written his doctoral thesis, also on the same topic.

2.2.1 Used data

Both De Wit, Zilko and Bergsma have chosen to use only one of the available data sets with incident recordings. Bergsma uses the recordings of Spoorweb, while De Wit and Zilko preferred to use SAP. CQM was the first to merge ISVL and SAP to expand the set of usable features. Moreover, all of these researchers made some distinction between failure types and created a different model per type. CQM is the only one of these parties that is able to return a prognosis for all types of failures. De Wit only includes technical disruptions in his research, so for example, incidents caused by obstruction are excluded. Zilko specifically focuses on incidents concerning switch and track circuit failures, and Bergsma stays with incidents related to switches.

Bergsma does not explain why he only focuses on switch failures. He does mention that his model is a proof of concept, in which he explores the use of a multi-agent system (MAS) for predicting FHT. He seems to imply that, if the MAS shows useful for predicting FHT for switch failures, it might as well be useful for predicting other kinds of failures.

Both De Wit and Zilko make a distinction between technical and non-technical problems, and choose only to include technical problems in the scope of their research. Zilko argues that incidents caused by suicide (non-technical) generally have a more or less certain disruption length of 120 minutes, which is why he has chosen not to construct a disruption-length model on such incidents. The other non-technical incidents, like hindrance by animals, are excluded because there is too little available data about them. Moreover, Zilko chooses to only create models for switch and track circuit failures. He argues that models for other technical incidents can be constructed in a similar fashion, by following the presented model construction procedure. De Wit decided not to include

catenary failures ("bovenleiding"). According to him, these failures are usually repaired at night, since they take a lot of time to repair.

2.2.2 Prediction methods

In Appendix B, visualizations are shown of the predictive models that the four previous researchers have used. Bergsma uses a multi-agent system to predict FHT. This model is dynamic, in the sense that extensions (e.g. switch type and weather) are added during the process of creating a prognosis. This means that new information about an incident can be added to the model when it comes available, to update the prognosis. The model of Zilko is also suitable for adding information later in the process, because it is a Bayesian Network. This model takes dependencies between variables into account, which implies that values that are not known initially can be inferred from known values, and added later when the real value is known.

The models of CQM, De Wit and Zilko are a bit alike: they are all based on probability distributions, and selecting a certain part of that distribution to be the prognosis. The models of CQM and De Wit are not as suitable for adding information as those of Bergsma and Zilko. Both are very static and made for a one time use (the initial prognosis), other than being updated with new information. The Decision Tree of CQM could be adjusted to do be suitable for this, by recording the number of elements in the training set that go down each branch and use the most popular branch if a value is missing [47]. This solution is, however, not implemented by CQM.

2.2.3 Conclusions from previous research

It can be concluded that none of the four models fully satisfies the current needs of ProRail, concerning FHT prediction. In the ideal situation, not only the initial prognosis could be generated by a model, also updates of the prognosis could be made by the model by adding newly obtained information. To do this, the model used for prognoses should not be as static as the currently used Decision Tree. Of the other three existing models, two already are dynamic. However, of all four models, only CQM covers all types of incidents. Both the models of De Wit and Zilko could presumably be extended to cover all technical incidents, but the problem remains that it will be difficult to extend them to non-technical incidents. This is because the relevant variables for non-technical problems are different and their parameters are unknown. Thus, the three theses are not fully applicable for predicting FHT for all incidents. It can be concluded that a new model should be created, which implements the dynamic form and takes all kinds of incidents into account.

3 Resources

3.1 Specifications

The laptop that I used to conduct this research is property of ProRail. Its specifications are the following:

Laptop: HP EliteBook 850 G3

Processor: Intel(R) Core(TM) i5-6300U

RAM: 8GB

Edition: Windows 10 Pro

3.2 Methods

The code that I have written during the process, was either in the language Python or in R. During the first, exploratory phase, I made use of Python Jupyter Notebooks to write my code. In this phase, mainly the Python packages Pandas and sci-kit learn were very useful. The first makes it possible to work with large amounts of data, and the second contains various machine learning techniques to explore the way that different algorithms react to the data.

During the second phase, the implementation of machine learning algorithms, I switched to using the language R with the program RStudio. In R, there are many available packages for machine learning algorithms. I used bnlearn for implementing Bayesian Networks, and class for the k-Nearest-Neighbors algorithm.

4 Data selection

In this chapter, I discuss the second step of the process of supervised machine learning by Kotsiantis et al. [22]. This step holds identification of the required data. I will discuss both ProRail's data on incidents, and other data sets that may be used to increase the number of features.

4.1 Used incident data

In section 2.1.1, the data sets of ProRail that contain incident recordings were presented. At first, we tried to merge all four data sets, to combine as much information as possible. This turned out to be very difficult, due to differences between the incidents that are recorded in each database, what incident id they are given and what time stamp is assigned. Besides, there is a large part of each data set that does not overlap with the other data sets. Eventually, only SAP and SAPX could be merged. This means I had to make a choice which data set to use for which model.

The merge of SAP and SAPX contains by far the most variables, as well as the most relevant variables. Moreover, it is much more suitable to merge with most of the additional data sets mentioned in section 4.2 than Spoorweb and ISVL are. The only disadvantage of this data set is that it does not contain rolling stock failures: incidents caused by failure of a train. As will be clarified in section 4.4, I will treat rolling stock failure apart from other kinds of incidents anyway. Therefore, I chose to use the SAP/SAPX merge (from now on referred to as just SAP) for all models except for the rolling stock failures. Recordings of the latter are present in both the ISVL and Spoorweb databases. I chose ISVL to use for rolling stock failures, because this data set is much more suitable to merge with the data set discussed in section 4.2.1.

Underlying the decision to use SAP, and not ISVL, for all other types of incidents, are two assumptions. The first is that SAP does contain at least as many data points than ISVL, so we would not lose a lot of information by using SAP. I use the selection of switch failures to check whether this is the case. To do this, I first cut off part of both data sets, because the time span of incidents recorded in both sets is not equal. This results in the test shown in code 4.1, from which it can be concluded that SAP contains more than twice as many data points on switch failures than ISVL does.

```
>>> isvl["start_incident"].min()
2015-01-01 14:47:00

>>> sap["start_incident"].min()
2015-01-01 10:05:09
```

```
>>> isvl["start_incident"].max()
2017-06-24 14:59:00

>>> sap["start_incident"].max()
2017-06-24 07:56:00

>>> isvl["incident_type"].isin(["wissel"]).shape
(2159, 41)

>>> sap["incident_type"].isin(["wissel"]).shape
(5051, 62)
```

Code 4.1: Python code to check whether the SAP data base contains more or equally as many data points as ISVL on switch data. The first four lines of output show that the time span of the two data sets is made equal. The last two lines return the shapes of the two data sets, where the first number represents the number of data points and the second number represents the number of variables.

The second assumption is that the time stamps of the incidents that are in both databases are more or less the same. This is important because the data points in ISVL are of the real-time format that will eventually be used to make prognoses. The box below resulted from testing a sample of incidents that were recorded in both SAP and ISVL. In this comparison, the variables on the left (SAP) are similar to the variables on their right (ISVL).

SAP		ISVL
begin_incident_dt	$\approx$	T_voorval
dt_aangemaakt	$\approx$	T_aangemaakt
dt_functieherstel	$\approx$	T_afsluiten

4.2 Other used data

Besides the general incident data sets discussed in section 2.1.1, there is a variety of other data sets available to add extra features to the data. In this section, I present the data sets that I merged with the incident data, together with their relevant variables. Only a short description and the name of the variables is given. A full description can be found in appendix C. A more detailed description of the files and the location that they can be found is omitted, since all files except for `KNMI_20181219.txt` are confidential property of ProRail.

4.2.1 Rolling stock defects

File name: `DefectMaterieel.csv`
This file contains records of rolling stock defects. Its variables are relevant for the rolling stock data set. The variable `Datum`, the date of the incident, is used to merge the data set with ISVL. The relevant features are rolling stock type (`Mat`), the driving characteristic

(Rijk), the involved train company (Vervoerder), whether the train table point ("dienstregelpunt") that the train is registered at is also a shunting point (RangeerDrp), and what the activity of the train was at this timetable point (Act).

4.2.2 Animals

File name: dieren.csv
This file contains incident recordings of incidents that involve animals. This can be a collision with an animal that was standing on or crossing the tracks, or hindrance by an animal being too close to the tracks to let the train drive at its usual speed. The information in this data set could be relevant for the collision/hindrance model. The variable inc_prorail_incident_id, the incident id number, is used to merge this data set with SAP. Currently, none of its features are actually implemented. Features that may be relevant are whether the animal was already dead, if it's hit or almost hit (Melding), the animal species (Soort) and the animal category this species belongs to (Categorie_bepaald).

4.2.3 Contract type

File name: dim_am_locatie.csv
This file contains a couple of location variables that also appear in the SAP data, of which I used aml_contractgebied_nr, the contract area, to merge the two data sets. The reason to use this data set are two variables. The first is aml_contractgebied_contractsoort, which I renamed to contract_soort for practical purposes. This variable represents the contract type, which is either PGO or OPC. This variable can be used for all technical incident groups. The second is aml_onderhoudsregio_naam, which is a location variable that divides the Netherlands in four regions and represents each one as a value.

4.2.4 Technical section information

File name: equipments_post21.csv
This file contains information about sections, the parts that the Dutch railway network is divided in, and is therefore useful for the section TOBS data set. The main relevant variables of this data set are the date of placement (Plaatsingsdatum) and the planned date of replacement (Theoretisch_vervangingsjaar) of each section.

4.2.5 TOBS data

File name: train_201805.csv
This file contains analysis data of all sections, to possibly add to the section TOBS model. The two variables sectie and datum, respectively representing the section number and the date, can be used to merge with the incident data. The relevant variables it contains are the number of TOBSs that day (n_tobs), the number of TOBSs of today and yesterday combined (n_tobs_2d), the number of TOBSs today of less and of more than 1 minute (technisch_falen_0s) and (technisch_falen_60s), the difference between the number of TOBSs of yesterday and today (n_tobs_delta), and the difference between the number of TOBSs of today and the day before yesterday (n_tobs_2d_delta). These features were

not implemented yet, because their use is very complex. This is because the variables are not very informative on their own, but they could make it possible to let the algorithm use information from other data points. TOBS incidents namely tend to repeat themselves short after a first incident has happened.

4.2.6 Weather

File name: `KNMI_20181219.txt`
This file is obtained from the KNMI, the Dutch institute for meteorology [21]. This institute provides daily data covering many aspects of the weather. Some aspects of the weather may influence FHT, for multiple types of incidents. A few weather variables that are potentially relevant (all of them covering one day) are the maximum temperature (`temp_max`), the amount of precipitation measured in millimeters (`neersl_mm`), the maximum distance of sight (`zicht_max`), the wind speed (`wind_speed`), the direction of the wind in degrees (`wind_dir`), and the duration of sunshine (`sun_dur`).

4.3 Irrelevant data

When reported, all incidents get a priority code, which can be either 1, 2, 4, 5, 8 or 9. These codes are grouped in categories UO ('Urgente Onregelmatigheid": urgent irregularity), DOT ("Dringende Onregelmatigheid met Tijdsafspraak": urgent irregularity with time appointment) and NUO ("Niet-Urgente Onregelmatigheid": non-urgent irregularity) [46].

UO This category contains priority 1 and 2. Priority code 1 almost never occurs, and is assigned when an incident is a major disaster and needs to be solved as soon as possible. Priority 2 is the most often occurring code. It means that the incident is classified as urgent, and needs to be repaired immediately instead of postponing the recovery to the night.

DOT This category contains priority 5. These are urgent incidents that need to be recovered, but it does not have to happen immediately. Depending on the nature of the incident and the current time, the contractor can start recovery when rush hour is over, or recovery is postponed to nighttime. The latter often means that time stamps of the recovery are not accurately recorded, because as long as recovery happens between 2 and 6 o'clock, it does not affect any trains.

NUO This category contains priority 4 and 9. Priority 4 is similar to priority 5, only it always implies postponing recovery to nighttime. Priority 9 is an administrative notification, which can be anything worth notifying. Most of the time, these are not even communicated to the contractor.

Priority 8 is a preventive notification. It means that the contractor should take a look at the concerned part at night, to verify that it works correctly. Incidents classified

as priority 1 or 2 are the only incidents that an accurate FHT prognosis is relevant for. Besides, these incidents are the only ones that are convenient to train a model on, since all other classified incidents have a higher risk of not being recorded accurately, because of being postponed to the night. For this reason, I chose to discard all incidents with a priority code (`prioriteit_code`) that is not 1 or 2 from SAP. ISVL does not have a documentation of priority codes, so I will leave this data set as it is.

4.4 Partitioning data per incident type

The next chapter will be about selecting features from the data set, to use for training and testing our model. In section 2.2, we saw that all previous studies on the topic of predicting FHT for railway incidents have split up the data by incident type. This is a very reasonable choice, because for example, factors influencing the FHT of a collision with a person are completely different from those influencing the FHT of a switch that is not controllable. Therefore, I also decided to partition the data in such a way that it is possible to make a selection of relevant features, specifically focused on a certain incident category.

Apart from incident type, it also seems reasonable to split up the data by incident cause. The resulting groups would also require different variables, which would presumably contribute to a better model. However, when the first prognosis of the FHT is made for a certain incident, the exact cause is usually not yet known. The incident type usually is. For training a model, this difference is not of any influence, but for the eventual implementation, it will be. For this reason, I chose to categorize by incident type. The resulting categories are shown in figure 4.1, including the number of data points per group.

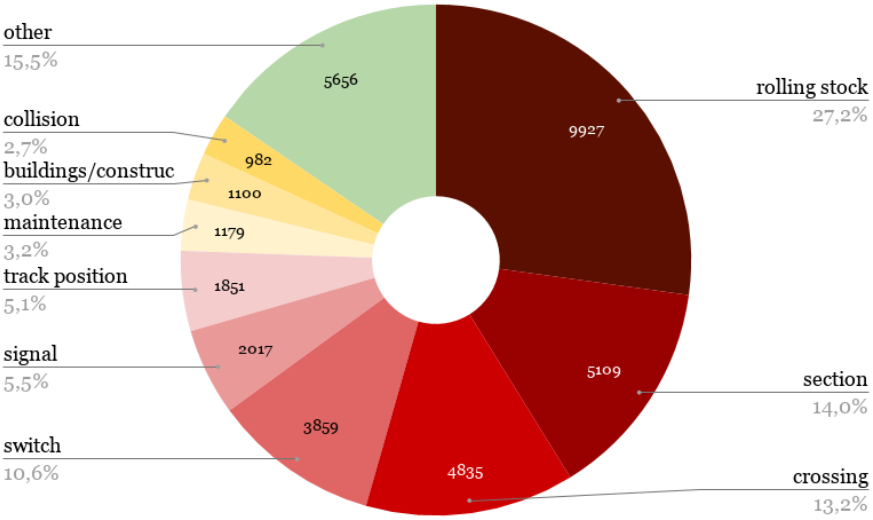

Figure 4.1: Number of data points per incident type

5 Data pre-processing

Now the data sets have been chosen, a little preparation should be done. This corresponding to the third step in the process of supervised learning of Kotsiantis et al. [22], discussed in chapter 1. The main reason for pre-processing the data is that models tend to perform very badly when working with raw data. This is because data usually contains a lot of missing values, or because the model prefers to work with other classes of data, so the variables have to be transformed. In this chapter, I will make the data as easy to work with as possible for our future models.

5.1 Missing values

5.1.1 Missing target values

The first thing we need to be present in the data is the function recovery time. After all, if we want to apply supervised learning, the target values should be available [1]. The variables on the left in the box below are present in SAP. On the right is their explanation.

timestamps	
dt_aangemaakt	date and time incident reported
aanntpl_ddt	date and time contractor arrives
dt_functieherstel	date and time function recovery
durations	
reactie_duur	latency time
arbeid	repair time
functiehersteld_duur	function recovery time

Of these variables, functiehersteld_duur, the FHT, is our main target variable. The variables reactie_duur and arbeid, latency time and repair time, are optional target variables. Knowing their values may contribute to making our prediction more accurate and specific. In SAP all of these six variables are partly filled with values, and partly empty. However, if a certain data point contains some of these values but not all, the missing values can often be calculated from the known ones, which is what I did for as many data points as possible. After that, there were a still lot of cases left where the FHT was present, but the latency time and repair time were not. Considering the exploratory nature of this research, I prefer not throwing away lots of data over the exactness of the repair and latency time, as long as the FHT is correct. I chose to fill in these cases using the median ratio of the two, and taking these parts of the FHT as latency and repair time. The ratio between the two appears to be 1.42, meaning that the latency time is usually 1.42 times longer than the repair time.

5.1.2 Few occurrences target values

Apart from values that are not recorded at all, it can also be tricky to work with values that almost never occur. When a model is trained on few examples of certain cases, it can become very biased by the few data points it has seen, and it will even be hard to tell whether a data point is an outlier[1]. Figure 5.1 shows how many data points have a certain FHT. This graph contains only the data of SAP, but the graph resulting from ISVL looks similar. The FHTs are divided in bins of 0.5 hours. From the figure can be concluded that there is quite some data on incidents with an FHT of less than about 6 hours, but incidents that take longer than that are quite rare. Note that there is a peak at 24 hours. This is caused by putting all incidents with an FHT that is 24 hours or more in the 24 hour bin, so the dot represents the total number of data points with an FHT of 24 hours or more.

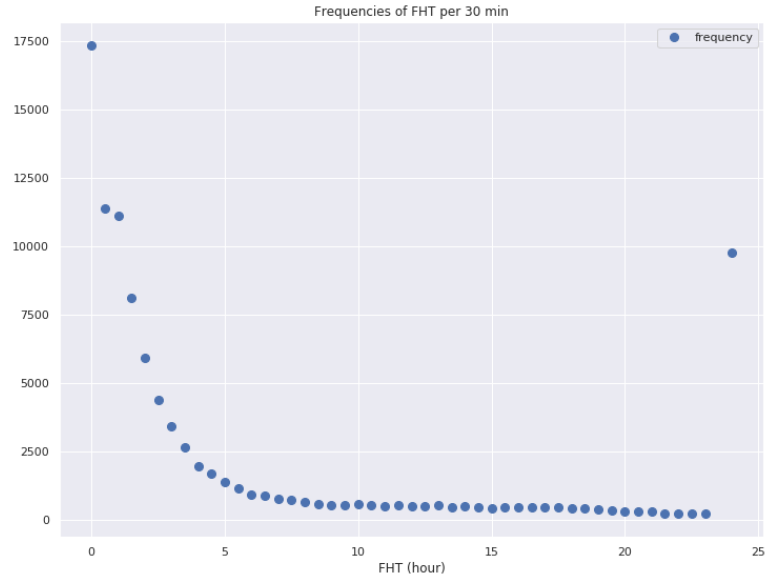

Figure 5.1: Frequencies of FHT (SAP)

Besides the fact that there are only few data points with an FHT of more than 6 hours, the duration of incidents with a large FHT is likely to be influenced by other factors than recorded in the data. This can be, for example, a decision to wait with the recovery process until nighttime, or a realization that the needed materials are not present, so extra time is needed to get those. Taking both the scarcity and the presumed abnormality of the incidents into account, I discarded the data points with an incident duration longer than 6 hours. Together with discarding data where the FHT is 0 or missing, this leaves 58,937 data points in SAP and 40,045 in ISVL.

[1] An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism [13].

5.1.3 Other missing values

When a variable contains missing values, there are two ways to handle this: deleting the variable or using imputation to fill in the missing values. Imputation techniques are based on the idea that any subject in a study sample can be replaced by a new randomly chosen subject from the same source population. Imputation of missing data on a variable, is replacing that missing value by a value that is drawn from an estimate of the distribution of this variable [11].

Features with many missing values were directly deleted in the beginning of the process. This is because when many values are missing, it is very likely that a lot of them will be filled in wrongly when using imputation, since there are relatively few known values to base their imputation on. There are methods that are able to work with missing values, like Bayesian Networks, but they also need to see a lot of examples from known values before they can predict what a value will be. Therefore, features with relatively few known values were deleted from the data set immediately.

For variables that contained only a few missing values, I did use imputation. Missing values for a variable $\mathbf{x}_i$ are missing completely at random if the probability of missing a value for $\mathbf{x}_i$ is unrelated to the values of $\mathbf{x}_i$ itself or to any other variables in the data set. If data are not missing at random, but as a function of some other variable, a complete treatment of missing data would have to include a model that accounts for missing data [29]. Unfortunately, for most of our variables, it is hard to tell whether there is a consistent reason for data not being recorded, or if values are randomly missing.

For this research, I assumed that all missing values are missing at random, because to me this seemed like a better solution than possibly assuming wrong dependencies as a basis for the rest of the process. Based on this assumption, I used three different ways of imputation. If a variable already contained a frequently appearing value like 'none' ore 'other', I chose to assign the missing values to this category too. For binary values, where `1` represents the presence of some attribute and `0` represents the absence, I did something similar by filling the missing values with `0`s. The remaining variables were filled with their median or mode, depending on whether the variable is numerical or categorical.

5.2 Variable discretization

Besides the many originally discrete variables, the data set contains a few continuous variables. Models usually prefer to work with either discrete or continuous variables, not with both. The few variables that are continuous, and relevant enough to possibly use in the model, are weather variables. Since I want to have the possibility of choosing a model that prefers to work with discrete variables, I discretized these variables. The option remains to use the continuous versions, when deciding to implement a model that prefers numerical variables. All weather variables presented in section 4.2.6 have a certain, more or less strict, threshold of severeness. I chose to use these thresholds as a boundary to make them binary, except for the wind direction, which I discretized to contain four values:

- warm: 1 if temp_max > 25, 0 otherwise
- veel_neerslag: 1 if neersl_mm > 8, 0 otherwise
- weinig_zicht: 1 if zicht_max < 70, 0 otherwise
- harde_wind: 1 if wind_speed > 20, 0 otherwise
- wind_compass: 0/90/180/270, for all 4 compass points in wind_dir

The preferred class of our target variable FHT depends on the model. Since the optimal model for our problem is not completely clear, I will now discuss how to discretize FHT, which is originally a continuous variable containing time in minutes. If a final model has been chosen, the decision can be made to use either the continuous or discrete version of the FHT variable.

It should be noted that discretization of continuous data can be problematic because information is lost, power is reduced and relationships may be obscured or changed [28]. This means that a considerate choice should be made where to place the boundaries of each category. According to Altman [2], there is no generally accepted method for doing this. Using groups of equal size is a reasonable approach, which means either an equal distance between the boundaries of each group, or an equal number of data points in each group. These strategies are not dependent on examination of the data. Sigurdsson et al. prefer another approach: "For optimal categorization of continuous covariates, univariate comparison was performed with chi-square values, with various cutoff levels" [41]. They compare the prognostic value of different covariates and adopted a p-value of 0.15 as the limit for the inclusion of a covariate. This led to a critical response of Altman. He states that it should not be the objective of the discretization to maximize the fit of the sample data, but that the whole point of analyzing a sample of data is to make inferences about all current and future data points. Thus, the method of analysis will overestimate the relevance of certain variables and invalidate the p-values obtained, which raises the risk of detecting a significant effect of a variable that is in reality not prognostic [2].

I was convinced by the argument of Altman, that although some way of discretizing FHT may result in a better fit on the training data, this does not mean that this it is the best way to represent its behavior in the real world. For this reason, I decided to stay with the proposal of Altman for discretization, using groups of equal size. Now, a choice still has to be made whether to make groups of equal space between group boundaries, or groups of an equal number of data points. I chose to do this based on the prediction errors produced by both methods. Although this seems to conflict with the argument of Altman, I would argue that his argument does not hold for this choice, since both options are very general divisions, that are not based on the meaning of FHT or its relation to other variables.

In table 5.1, a comparison is made between the root mean square error (RMSE) of FHT predictions on the section TOBS (left) and collision/hindrance (right) data sets. These predictions were made using a Naive Bayes Classifier. As explained in section 7.1.6, this method is very simple, assumes independence of all features, and is remarkably successful in practice. I chose to use this method for the test because of its simplicity, which reduces

the chance of some unknown influence interfering with the test. In both data sets, the FHT was discretized, but in different ways. The data sets used for the top part of the table contain an FHT that was discretized by creating bins with an equal number of minutes. The FHTs of the bottom part consists of bins with an equal number of data points. The results in the table suggest that dividing FHT in equally sized bins with respect to the time span results in a smaller error than putting an equal number of data points in each bin. Therefore, I chose to discretize FHT by making groups with an equal time span.

equal number of minutes in each group

# groups	section TOBS	collision/hindrance
6	96.67648	67.39478
12	95.0261	71.08472
18	94.71225	74.82672
24	94.07444	57.43896

equal number of data points in each group

# groups	section TOBS	collision/hindrance
6	133.933	153.4553
12	116.9134	129.6132
18	108.9696	98.61567
24	110.8834	106.4399

Table 5.1: Root mean square error between actual FHT and predicted FHT, predicted using a Naive Bayes Classifier. This overview compares the scores with the use of different FHT bins, either containing an equal number of minutes or containing an equal number of data points.

5.3 Overfitting

The principle of parsimony calls for using models and procedures that contain all that is necessary for the modeling but nothing more. Overfitting is the use of models or procedures that violate parsimony, that is, that include more terms than are necessary or use more complicated approaches than necessary. Adding predictors to a model that perform no useful function not only wastes resources, it can also make predictions worse, because the coefficients fitted to them add random variation to the subsequent predictions [14].

Overfitting usually happens when the data that a model is trained on has some property that does not exist 'in the real world'. In other words, the model is fitted too specifically on the training data and therefore not extendable to the data points it has to classify. The out-of-sample error E_{out} measures how well a trained model can be generalized to data that it has not seen before. Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training [1]. Consequently, it is necessary to divide the data in two parts: a training set and a test set. According to Cassio de Campos, a professor at Utrecht Uni-

versity, it is important to divide the data in such a way that the target variable is equally distributed over the two sets. This means that the density of our FHT variable should be equal in both sets. This is achievable by using the `createDataPartition` function from the caret package in R. I chose to take 90% of the data as training data, and save 10% for testing the model. The code for this division is shown in code 5.1.

```
target_var = "fht"

train_ind <- createDataPartition(data[,target_var], p = .9)
data_train <- data[train_ind,]
data_test  <- data[-train_ind,]

x_cols = colnames(data)[colnames(data) != target_var]
y_cols = c(target_var)
xTrain = data_train[,x_cols]
yTrain = data_train[,y_cols]
xTest = data_test[,x_cols]
yTest = data_test[,y_cols]
```

Code 5.1: Dividing the data in a training set and a test set, with an equal distribution of FHT in both sets, using the caret package in R [23].

6 Feature selection

The pre-processing of data, step three in the supervised learning process of Kotsiantis et al. [22], is not only about formatting the data in a way that a model will be able to work with it. It is also important to select the right features to give to the model as input. I will do this per incident type, as discussed in section 4.4. The available data sets contain a large number of features, many of which are not useful for our predictive model at all. Therefore, it is necessary to make a selection of features that are relevant for predicting incident duration.

6.1 Purpose of feature selection

Feature selection is about selecting a subset of features that are in some way relevant to the prediction target. Making this selection reduces the dimensionality of the feature space [15], and sometimes improves prediction accuracy [20]. Mainly reducing dimensionality is of importance for our data set, because we have a large number of possible features compared to a relatively small number of data points. This comes with the risk of a model complexity that is much too high.

According to Chandrashekar and Sahin, an important part of removing redundant features is eliminating dependent variables: those that are correlated with other variables [7]. However, this may be the case for e.g. linear models, because adding an extra dimension to the feature space that is almost the same as another dimension most likely does not contribute to the performance of the model. To Bayesian Networks, on the contrary, correlations can be very useful, since these models often deal with missing information. The correlations can then be used to compensate for the absence of information. In my opinion, this difference makes it difficult to bluntly say that deleting variables with a strong correlation contributes to the model's performance. Therefore, I will base the selection of features mainly on testing to what extent a feature correlates with the target variable FHT, and thus likely contributes to its accurate prediction.

6.2 Features per incident type

In this section, I will discuss the relevant features for three incident types of the ones distinguished in figure 4.1: rolling stock failures, section TOBS and collision/hindrance incidents. These groups are visualized in figure 4.1. The reason that I have not applied feature selection to the data for all incident types is that this was not achievable within the time span of this project. I chose to focus on these three types specifically because I think they best represent all types of incidents. Therefore, both feature selection and the

model building that follows in chapter 7 will be easily extendable to all data points, to make it possible to predict FHT for all kinds of incidents.

For each type, I will go over the most important steps that I took to select relevant features. These steps mainly consist of considering features used in previous research and adding new features that I found to contribute to accurately predicting FHT. To do this, I have alternately used common sense, statistical tests and testing of arc strength by the bnlearn package in R. At the end of the section for each type, the chosen features are summed up in a box, together with their variable name in the data set. In appendix C, all features included in those boxes are explained. Besides, in this section I will particularly often refer to Bergsma [3], CQM [9], De Wit [46] and Zilko [50]. This is why I chose to leave annotation for those papers out during this section, for the purpose of readability.

6.2.1 Rolling stock failure

An incident is classified as a rolling stock failure if the incident is caused by failure of a train, because some part of the train is broken. CQM is the only one of the four previous studies that built a model to predict the FHT of rolling stock failures. The features they use are whether the incident takes place at the `HSL/Betuweroute/other`, whether it is in the `Randstad`, the **`driving characteristic`** and **`rolling stock type`** of the failed train, if it is a **`freight train`**, and if it is **`day or night`**. Other features that CQM has considered, but which they did not use, are the **`type of section`** and **`time`** of day.

Whether the incident takes place at the **`HSL or Betuweroute`** can be extracted from the variable `Melder`, by applying a regular expression. The result is saved in the variable `HSL_betuwe`. Whether the incident takes place in the `Randstad` is also recorded in `Melder` in ISVL, but of the selected data points that represent rolling stock failures, only one data point with the value `Randstad` remains. Therefore I will not use this feature in the model. The **`driving characteristic`** and **`rolling stock type`** of the failed train are represented by the variables `Rijk` and `Mat`. Because the `Mat` is quite chaotic, holding information for all train units separately in each cell, I filtered its values to contain just one value for each data point and renamed the variable to `mat_type`. Whether the specific failed train is a freight train is represented in the data as a value `GO` for 'goederentrein' in the variable `Rijk`. I will place this in a separate, binary variable `goederentrein`. If it is **`day or night`** can be derived from the variable `tijd`, which represents the hour that the incident is reported. The value of this variable `night` will be `1` if the time is between 0AM and 6AM, and `0` otherwise.

As a feature selection technique, I created a Bayesian Network for rolling stock incidents using hill climbing, using the bnlearn package in R. When I calculated arc strengths, the following features having strong arcs to either FHT or other important variables, which led me to selecting them as features. The first is the `TIS` scenario, which is a number that is assigned to an incident to indicate its severity. This feature is recorded in ISVL as `Scenario`, but I renamed it to `tis` for clarity. Next, the variable `vsm_aangepast` is added to the model. Vsm is the abbreviation for 'versperringsmaatregel', which stands for the adjustment of the train schedule due to the rail blockage at the incident loca-

tion. This variable indicates whether the vsm is adjusted, which in turn presumably indicates a complex incident scenario. The **train company** that is delivering the ride is also included as a feature, named Vervoerder. RangeerDrp is a binary variable that contains whether the involved service control point also acts as a **shunting point**. Last, the train's **activity** at the specified location is added, containing the values passing (D), arriving (A) and departing (V), and represented as the variable Act.

feature	variable
HSL/Betuwe	HSL_betuwe
driving characteristic	Rijk
rolling stock type	mat_type
freight train	goederentrein
day/night	night
TIS	tis
train table adjusted	vsm_aangepast
train company	Vervoerder
shunting point	RangeerDrp
activity	Act

6.2.2 Section TOBS

TOBS stands for "Ten Onrechte Bezet Spoor", which means 'falsely occupied track'. The network of rail tracks is divided in sections. Between the edges of two sections, an insulating weld is placed to separate them. When a train drives over this weld, it short circuits an electricity loop, which is registered as an occupation of the concerning track. Sometimes, a track occupation is registered, despite there being no train present, which is called a 'section failure' or 'section TOBS'. To predict the FHT of a TOBS, CQM uses the features whether the failure is at the **HSL/Betuweroute/other**, whether it is **day or night**, whether it takes place in the **Randstad** and the **type of section** (vrijebaan/emplacement). Zilko uses the features **contract type**, distance to the **nearest level crossing**, distance to the **nearest working station**, whether it is '**warm**' outside, whether the incident takes place **during contractual working hours**, the presence of an **overlapping incident** and whether it is **rush hour** to predict the FHT of a TOBS. De Wit implements the features whether the incident is caused by a **grinding train**, which **VL−post** belongs to the concerned area, the **contract type** and whether it is **rush hour**. When I went to his office to talk about his research, he also mentioned that it is very important whether the incident happens **at/near a station** or far from it.

Both whether the incident is **during contractual working hours**, whether it is during **day or night** and whether it is during **rush hour** can be derived from the feature **time** and placed in a binary variable. The first, named contr_working_hours, will be 1 if the reporting time of the incident is between 7AM and 4PM, and 0 otherwise. The second, named night, will be 1 if the time stamp is between 0AM and 6AM, and 0 otherwise. The latter, named rush_hour, is 1 if the time is between 6:30AM and 9AM or between 4PM and 6:30PM. Whether the failure occurred in the **Randstad** is represented in SAP as a value in aml_contractgebied_naam. I recorded this as a variable Randstad, with value 1

when the incident occurs in the Randstad, and 0 otherwise. The feature **contract type** is directly represented by the variable contract_soort. Whether the weather is **warm** can be derived from the temp_max, as already mentioned in section 5.2. According to Zilko, the optimal threshold for this feature is 25°C, meaning that if it is warmer than that, the section systems have a much bigger chance to fail, which can affect the latency time due to an increase of overlapping incidents of the same sort. I created, like Zilko, a binary variable warm, which is 1 when the maximum temperature of that day is higher than 25°C, and 0 otherwise. The feature that contains whether there is an **overlapping incident** occurring is represented by the binary variable overlapping_inc. Its value is determined by checking whether there is another incident in the data set that happened in the same contract area (contractgeb), within the last four hours before the current incident. All of the features in this paragraph can be implemented, but the question remains whether this is useful. Therefore, I applied a one-way ANOVA test using the aov function in R, to test whether these variables correlate with FHT. From these tests, I concluded that the variables contr_working_hours, rush_hour, Randstad, contract_soort, warm and overlapping_inc all correlate significantly with either latency time, repair time or both. For the variable night, this is not the case, but this variable does significantly correlate with oorzaak_groep and overlapping_inc, which can be of added value when implementing a Bayesian Network.

I discarded the features **grinding train** and **HSL/Betuweroute/other**. Both were binary variables, respectively representing whether a grinding train was the incident cause and whether the incident occurred at the HSL, Betuweroute or somewhere else. However, the former contained two occurrences, the latter contained only others, so both features are not used in this model. Besides, the features **type of section**, **nearest level crossing**, **nearest working station** and **nearest station** are not added to the model, simply because they are very hard to find and to add to the data. The feature **VL−post** is also not used, because its variable vl_post is categorical with many values. Models usually prefer either continuous variables or categorical variables with few values, especially when the number of data points is low.

Besides considering the features that were implemented by previous researchers, I added a few other features to the model, which I found to also correlate with FHT. The **theoretical year of replacement** contains information about the maintenance of the considered part of track. To convert this feature to a usable variable, I divided the dates and years in three groups. This resulted in the categorical variable Theoretisch_vervangingsjaar. Next, a location variable standplaats was added, containing a contractor-related location, with five possible values. I also added which **contractor** is responsible for recovery of the incident, represented by the variable aannemer. Furthermore, a feature is added which indicates whether the train table is expected to be influenced by the incident. This is recorded in the variable tao_ind, in which **tao** stands for 'trein aantastende onregelmatigheid', or 'train affecting irregularity'. The variable ozi_oorzaak_code contains codes that stand for a specific cause of the incident. These codes can be grouped to make the variable more compact, resulting in a categorical variable oorzaak_groep with nine possible values.

Last, I tried adding three weather variables: wind direction, the amount of rainfall and the distance of view, which can be limited by mist or heavy rain. By testing correlations between those features and the other implemented variables, I concluded none of them correlate (almost) significantly with FHT. The latter two also barely have correlations with other variables. The direction of the wind, represented by the categorical variable wind_compass does significantly correlate with warm and Randstad. Because such correlations may be helpful for the construction of Bayesian Networks, I only added wind_compass to the model.

feature	variable
day/night	night
contractual working hours	contr_working_hours
Randstad	Randstad
contract type	contract_soort
warm	warm
overlapping incidents	overlapping_inc
rush hour	rush_hour
year of replacement	Theoretisch_vervangingsjaar
location of base	standplaats
contractor	aannemer
tao indicator	tao_ind
wind direction	wind_compass
cause	oorzaak_groep

6.2.3 Collision / hindrance

Sometimes, nothing is broken and nothing has to be repaired, but an incident is reported because a train is (almost) involved in an accident. This can, for example, be because there is hindrance of a sheep standing on the tracks, because of a suicide attempt, or because the train has (almost) collided with a vehicle that was driving on a crossing. CQM and De Wit discuss this kind of incident. CQM uses the features whether the incident takes place in the Randstad, if it takes place during working hours (7AM to 6PM), if the involved train is a Sprinter and if it is day or night. De Wit uses the features contract type and whether there have been recent maintenance activities.

The feature Randstad is similar to the variable aml_onderhoudsregio_naam. I changed the variable to one named Randstad, being 1 when the value is Randstad Noord or Randstad Zuid, and 0 otherwise. Both the features whether the incident is during working hours, and whether it is during day or night can be abstracted from the feature time. Both variables will be binary. The former, named working_hours, will be 1 if the reporting time of the incident is between 7AM and 6PM, and 0 otherwise. The latter, named night, will be 1 if the time stamp is between 0AM and 6AM, and 0 otherwise. The feature whether the involved train is a Sprinter is seldomly recorded. Because I think using this feature comes with a high risk of using wrongly imputed data, I will leave this feature out. The feature contract type is recorded in the variable contract_soort, which can immediately be included in the data set. Whether there has

been `recent maintenance` activities is, at the moment, hard to include in the data set, because the processes and data sets related to incidents and those related to maintenance are very different. It would take a lot of time to merge the information, and will probably result in the loss of many data points, which is very unfortunate for the already small set of 982 data points on collision/hindrance incidents.

As a feature selection technique, I created a Bayesian Network for collision/hindrance incidents using hill climbing, using the bnlearn package in R. When I calculated the arc strengths, this resulted in no arcs from or to `contract_soort` other nodes (including FHT) with a cogent strength. This does make a lot of sense, because for most incidents in this category, no repairing work is needed. Although the type of contract does influence the duration of a repair, this is not the case for a non-technical problem like hindrance or a collision, since they require a completely different type of work.

Apart from using some features mentioned in previous research, I added some other features to this model. First of all, it is both relevant and very easy to know soon after the incident has happened, what type of `thing the train has collided with` or is hampered by. I divided this `counterpart` in four categories: animals (`dier`), persons (`persoon`), vehicles (`wegvoertuig`) and objects (`object`). I also added an 'other' category (`overig`) for all data points that can not be placed into one of those categories. Next, I thought it would be important to make a distinction based on the `nature of the incident`, between a collision and hindrance. The value for this feature is very easy to fill in during an actual incident, and for the training set it can be abstracted from the cause codes (`ozi_oorzaak_code`). I called this variable `type_hinder`.

Last, the location variable `standplaats` and the hindrance indicator `tao_ind` are added. The former indicates where the repair team has to come from when the incident takes place during working hours. The abbreviation TAO stands for "Trein Aantastende Onregelmatigheid", which means 'train affecting irregularity'. Thus, the latter variable indicates whether the train table is influenced by the incident. An estimate of this variable can also be an indication for the repair time.

feature	variable
Randstad	Randstad
working hours	working_hours
day/night	night
thing train collided with	counterpart
nature of the incident	type_hinder
location of base	standplaats
train table affected	tao_ind

7 Model selection

The fifth step of the supervised machine learning process of Kotsiantis et al., discussed in chapter 1, is algorithm selection [22]. There exists a large variety of supervised learning algorithms, that all have their specific strengths and weaknesses. Because we will use supervised learning, I will only discuss supervised learning algorithms in this chapter. In the process of selecting a model to predict FHT, we should take a look at the data first.

Something very important to notice about the data, is that the historical data that we will train our model on may be complete, with no missing values after the pre-processing, but the real data that will be used in practice for predicting FHT will contain a lot of missing values. This is due to the absence of (diagnostic) information about the incident when the first prognosis is made. Thus, we have to choose a model that is able to make predictions for data points that contain missing values. The second important property of the data is the scale of its variables. As can be seen in appendix C, all features are categorical (either nominal, dichotomous or ordinal). The scale of the target variable, FHT, can be chosen dependent on our preferred output and predicting method. It could be kept a ratio variable, as the duration in minutes originally is of ratio scale. The variable can also be discretized in various ways, as explained in chapter 5.

Apart from the properties of our data, we also must take into account the way that the model will be used. Maybe, in the very far future, a model will independently be able to predict FHT, and decide what approach to handle the incident is best, based on this prediction. However, at least in the near future, algorithms will only act as assistants, intended to give humans more insight in the situation, to help them making decisions. This means that a property of a good algorithm should be its interpretability. After all, if a model outputs only a single number, e.g. predicted minutes FHT, the human can barely do more than choosing to agree or to disagree with the prediction. On the contrary, when this human has any idea what the prediction is based on, he or she can decide what aspects might have influenced the decision. The interpretability of the model could contribute to a better understanding of the situation and its influence on FHT, which in turn might improve the humans decision making. In the next section, several algorithms are discussed, paying attention to the handling of missing values, the classes of our variables and the interpretability of the models.

7.1 Supervised learning techniques

There exists a large variety of supervised learning techniques, some of which are extensions of others. I want to take enough, but not too many techniques into consideration. Therefore, I chose to use four books about Machine Learning as a starting point, and

to elaborate on the main techniques that are discussed in these books. These books are "Learning from data" by Abu-Moustafa et al. [1], "Pattern recognition and machine learning" by Bishop [4], "The elements of statistical learning" by Hastie et al. [12], and "Data mining" by Witten et al. [47].

7.1.1 Linear models

When all variables, both the target variable and the features, are numeric, *Linear Regression* (LR) is a natural technique to consider. The idea is to express the value of the target variable as a linear combination of the attributes, with predetermined weights [47]. LR is based on minimizing the squared error between the predicted value $h(\mathbf{x})$ and the real value y of the target variable. Here $\mathbf{x}$ is a vector representing a data point, with values $(x_0, \ldots, x_k)$ representing all its feature variables. $h(\mathbf{x})$ takes the form of a linear combination of the components of $\mathbf{x}$, that is,

$$h(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^T \mathbf{x},$$

where $x_0 = 1$, and $\mathbf{x} \in \{1\} \times \mathbb{R}^d$, and $\mathbf{w} \in \mathbb{R}^{d+1}$, where d is the dimensionality of the input space [1].

A similar model to Linear Regression, is *Linear Classification* (LC), where $h(\mathbf{x})$ takes the form

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x}).$$

Figure 7.1 shows an example of a two-dimensional linear model, where the black line represents $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ from Linear Regression [12]. This line can be directly used to predict a numeric value, or complemented by the sign function to predict a binary class value. In the example of figure 7.1, these classes are orange and blue, either assigned to one side of the $h(\mathbf{x})$-line.

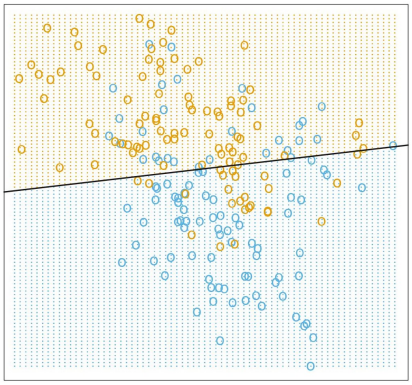

Figure 7.1: Example of a two-dimensional linear model, with classes orange and blue. Both axes represent a feature [12].

A third variation of the linear model is *Logistic Regression* (LogR), which is something between LC and LR, and can be applied when the target variable can not be approximated accurately using a linear function. The output can be interpreted as a probability for a binary event. The difference from Linear Classification is that in Logistic Regression, the chosen class is allowed to be uncertain, with intermediate values between 0 and 1 reflecting this uncertainty. Instead of approximating the 0 and 1 values directly, LogR replaces the original target $h(\mathbf{x})$ by $log(h(\mathbf{x}))$ and smoothly restricts the output to the probability range [0,1]. Here, $h(\mathbf{x})$ takes the form

$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}),$$

where θ is the so-called logistic function $\theta(s) = \frac{e^s}{1+e^s}$ whose output is between 0 and 1 [1]. An example of LogR is shown in figure 7.2.

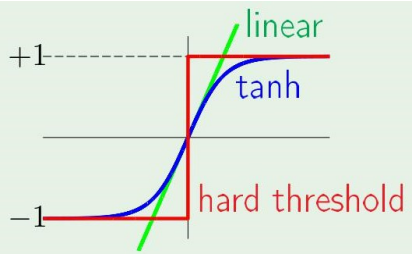

Figure 7.2: A visual comparison of the outputs given by Linear Regression. Linear Classification and Logistic Regression. Here *tanh* is used for LogR instead of $\theta(s)$, which moves the boundaries to -1 and 1 instead of 0 and 1. However, the relative shapes of the three outputs stay the same. The output of LR (green) is a real value that is not bounded. LogR (blue) is bounded like in LC (red), but also outputs real values [1].

Both LR and LC can only effectively work with numerical values, otherwise $\mathbf{w}^T\mathbf{x}$ can not be computed. There is, however, done research on the topic of discrete linear classification, but finding a discrete linear function that minimizes the cumulative hinge loss[1] of a data sample is NP-hard [8]. For LogR, however, it does not matter whether the used variables are continuous or discrete.

Linear models are not able to directly deal with missing values. This can be illustrated by the visualized model in figure 7.1. In this two-dimensional space, both axes represent a feature. In case we want to predict the class for a new point, we have to know the value of both features, otherwise it is not possible to position this data point in the model space.

7.1.2 Support Vector Machines

In figure 7.3, two classification problems are shown. Intuitively, a large margin between the decision boundary and the closest data points contributes to the performance of a

[1] A loss function $V(w, y)$ represents the price we pay by predicting $h(\mathbf{x})$ in place of y. Examples of loss functions are the square loss $V(w, y) = (w - y)^2$ and the hinge loss $V(w, y) = max\{1 - xy, 0\}$ [36].

linear classifier. This is because then, the classifier will still classify a data point correct when new data points are a little bit shifted with respect to the training data, or if the values of the training data contain some noise [37]. In Support Vector Machines, the decision boundary is chosen to be the one for which the margin is maximized [4]. The data points closest to the margin are called support vectors. There is always at least one support vector for each class, and often there are more. The set of support vectors uniquely defines the maximum margin hyperplane for the learning problem; all other training instances can be deleted without changing the position and orientation of the hyperplane [47].

Support vector machines can also be used for classification problems where the data is not linearly separable. This can be done either by transforming the feature space using a kernel (discussed later), or by allowing data points in the margin to a certain extent, parameterized by ξ [37]. This case is shown in the right panel of figure 7.3.

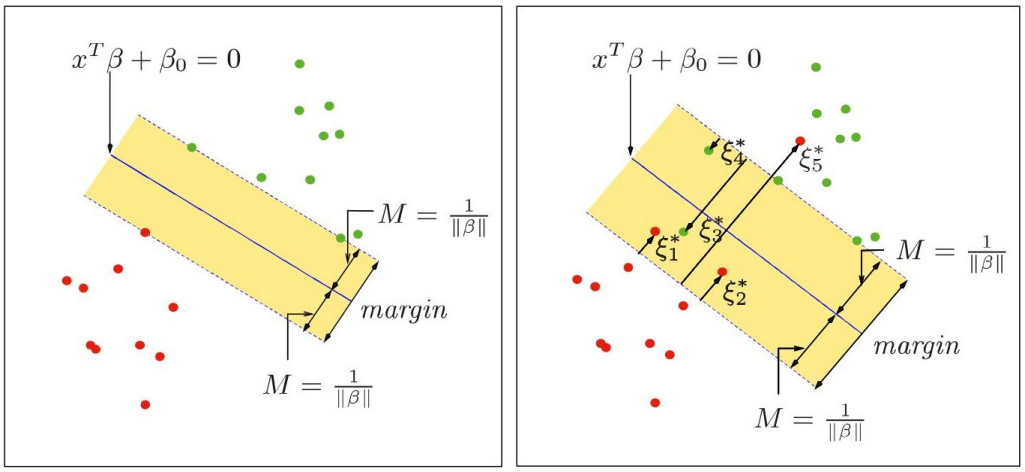

Figure 7.3: Support vector classifiers. The left panel shows the case where the two classes can be linearly separated, where the solid line represents the decision boundary, and the dotted lines bound the maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the non-separable case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin. [12].

Finding the optimal separating hyperplane is an optimization problem. For the case where the data is linearly separable, this problem is defined as [37]:

$$\begin{aligned} \underset{b,\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \text{ for } n = 1, \ldots, N \end{aligned}$$

where b represents the bias, which was denoted as w_0 for the linear models. For the case where the data is not linearly separable, like in the right panel of figure 7.3, the optimization problem is defined as:

$$\begin{aligned} \underset{b,\mathbf{w},\xi}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n \\ \text{subject to} \quad & y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \\ & \xi_n \geq 0 \text{ for } n = 1, \ldots, N \end{aligned}$$

where ξ_n is the 'soft' error on $(\mathbf{x}_n, y_n)$, and C is a regularization parameter. If C is small, the margin has to be big, even at the expense of the in-sample error. If C is big, the in-sample error is minimized despite of creating a small margin [37]. Unfortunately, this parameter must be chosen by the user, and the best setting can only be determined by experimentation [47].

Another way to use SVMs to classify non-linearly separable data, is by transforming the feature space, using a kernel function. Kernel functions represent a dot product in the feature space created by Φ, which is a function that maps an instance into a (potentially high-dimensional) feature space [47]. In fact, we do not have to specify the transformation $h(x)$ at all, but require only knowledge of the kernel function $K(x, x') = \langle h(x), h(x') \rangle$ that computes dot products in the transformed space [12].

SVMs can also be extended to multi-class problems, although essentially by solving many two-class problems. A classifier is built for each pair of classes, and the final classifier is the one that dominates the most [12]. They can also be developed for prediction of numeric variables, despite the fact that the maximum margin hyperplane only applies to classification [47]. Support Vector Machines are only able to work with numerical features. There are ways to work with discrete variables, but only by transforming them in some way to numerical variables, otherwise it will not be possible to compute $\mathbf{w}^T\mathbf{x}_n$.

7.1.3 Neural Networks

The central idea of Neural Networks is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features [12]. The most prominent type of a Neural Network (NN) is the multilayer perceptron [47], which in fact is a misnomer, because the model comprises multiple layers of Logistic Regression models rather than multiple perceptrons [4].

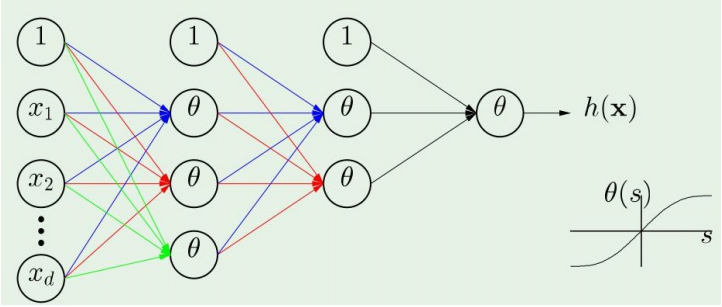

Figure 7.4: A fully connected multilayer perceptron, where the left layer is the input $\mathbf{x}$, the middle two layers are the hidden layers $1 \leq \ell < L$, and the right layer is the output layer $\ell = L$ [10].

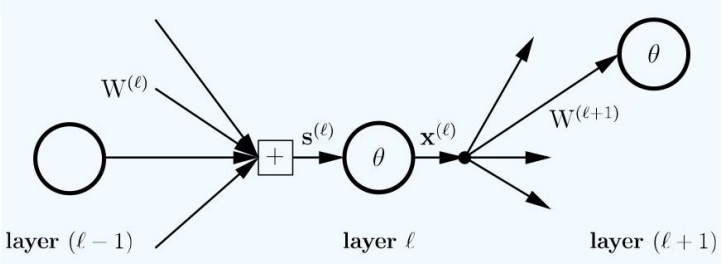

Figure 7.5: One layer ℓ of a Neural Network [10].

Figure 7.4 shows an example of a Neural Network. In each 'neuron' in layers $1 \leq \ell \leq L$, a non-linear function θ is applied to the input s before passing it to the output x, as is illustrated in figure 7.5:

$$x_j^{(\ell)} = \theta(s_j^{(\ell)}) = \theta(\sum_{i=0}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)}).$$

Each layer $\ell \in \{1, \ldots, L\}$ has a matrix of weights $W^{(\ell)}$. The (i,j)th entry in $W^{(\ell)}$ is $w_{ij}^{(\ell)}$, going from node i in the previous layer to node j in layer ℓ. $\mathbf{w} = w_{ij}^{(\ell)}$. After we fix the weights in all layers, we have a Neural Network hypothesis $h(\mathbf{x}; \mathbf{w})$ [10]. The generic approach to minimizing the prediction error is by gradient descent, called back-propagation in this setting. This can be computed by a forward and backward sweep over the network. In the forward pass, the current weights are fixed and the predicted values are computed. In the backward pass, the errors are computed, which are used to compute the gradients for the updates [12].

A serious disadvantage of multilayer perceptrons is that they contain hidden units, which makes them opaque. It is also unclear whether they offer any advantages over standard rule learners that induce rule sets directly from data, especially considering that this can generally be done much more quickly than learning a multilayer perceptron

[47]. Besides, Neural Networks require complete records to do their work [22]. The variable classes that the model is able to work with are usually the same as for LogR, since the network is a combination of multiple layers of LogR models. This means that the algorithm is able to work with both discrete and continuous variables, but its output is a classification. It is also possible to predict a numerical variable using a Neural Network, by replacing the LogR functions in its neurons by LR functions [42].

7.1.4 K-Nearest-Neighbors

K-Nearest-Neighbors (kNN) classifiers are memory-based classifiers, they do not require a model to be fit. Given a query point $\mathbf{x}_0$, we find the k training points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ closest in distance to $\mathbf{x}_0$. The predicted value of $\mathbf{x}_0$ is then determined by taking the majority vote among the values of these neighbors [12]. This can also be written as:

$$h(\mathbf{x}_0) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x}_0)} y_i,$$

where $N_k(\mathbf{x}_0)$ is the neighborhood of $\mathbf{x}_0$ defined by the k closest points $\mathbf{x}_i$ in the training sample. An example of the kNN algorithm is shown in figure 7.6.

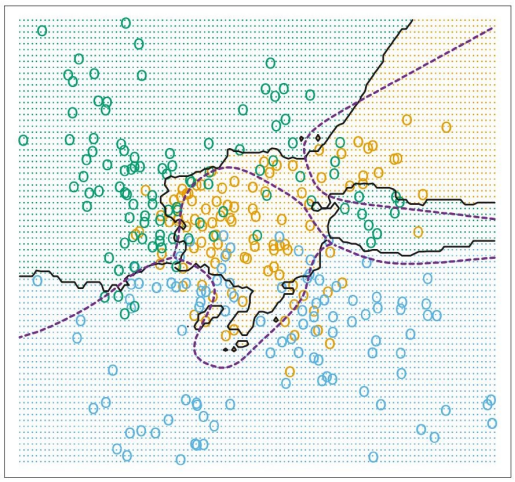

Figure 7.6: Example of a 15-Nearest-Neighbors algorithm in a two-dimensional feature space, used to assign the data points to three classes [12].

Closeness implies a metric, a measure of how close a data point is. Although there are other possible choices, most instance-based learners use Euclidean distance [47]. The Euclidean distance between an instance $\mathbf{x}^{(1)}$ with values $x_1^{(1)}, x_2^{(1)}, \ldots, x_k^{(1)}$, and an instance $\mathbf{x}^{(2)}$ with values $x_1^{(2)}, x_2^{(2)}, \ldots, x_k^{(2)}$ is defined as

$$\sqrt{(x_1^{(1)} - x_1^{(2)})^2 + (x_2^{(1)} - x_2^{(2)})^2 + \ldots + (x_k^{(1)} - x_k^{(2)})^2}.$$

To use this distance measure, all variables first have to be normalized, so variables with large numbers as values will not influence the prediction more than variables with small values. Besides, this distance measure can only be used for data points with exclusively numerical variables. A similar measure for categorical variables has not been defined yet. Generally, when the kNN algorithm is used on data with categorical variables, these are converted to a number of binary 'dummy' variables. Buttery proposes another way to handle categorical variables, by replacing each category by a real number in an 'optimal' way [6], but this still implies the need for making the data numerical, because the kNN-algorithm is not able to work directly with categorical data.

The k-Nearest-Neighbors algorithm is mostly successful in problems where the decision boundaries are very irregular [12], so the data points are far from linearly separable. However, it is best when the feature space is not too high-dimensional. When the algorithm is carried out in a feature space that is too complex, the nearest neighbors of a point can easily be very far away, causing bias and degrading the performance of the rule [12].

The algorithm is very bad at handling missing values of points in the data set, because this means the data point can not adequately be placed in the feature space. When the data point that has to be classified is incomplete, however, it could be a possibility to shrink the feature space to include only the features that the data point does contain values for. This would make it possible to find its nearest neighbors, and make a prediction, based on only a small amount of available information.

7.1.5 Decision Trees

Decision Trees are graphs that classify instances by sorting them based on feature values. Each node in a Decision Tree (DT) represents a feature in an instance to be classified, each branch represents a value that the node can adopt, and each leaf is a class name of the respective instance. In order to classify an object, we start at the root of the tree and take the branch with the appropriate value. This is repeated at every encountered node, until a leaf is reached, and the object is classified as the class named by the leaf [22, 34]. Figure 7.7 shows an example of a DT. This tree represents the choice whether to play tennis or not, based on the weather; it classifies objects (days) either as class P (suitable to play tennis) or N (not suitable to play tennis), based on three discrete variables.

When it comes to predicting numeric quantities, the same kind of tree can be used, but each leaf would contain a numeric value, acquired by taking a specific part of the distribution of all the training set values to which the leaf applies. This means that for both the discretized version of our target variable, and for the numerical version, a Decision Tree can be used. If one of the features is numeric, the node usually determines whether its value is greater or less than a predetermined constant, giving a two-way split [47].

It should be noted that Decision Trees suffer from the fragmentation problem: after multiple splits based on the node-tests, there can be very little data on which to base decisions [19]. Besides, missing values pose a problem, because it is not clear which

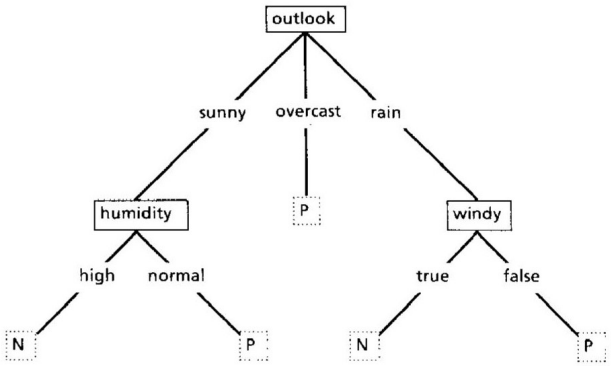

Figure 7.7: Example of a simple Decision Tree [34]

branch should be taken when a node tests a variable whose value is missing. Sometimes, a missing value can be treated as an attribute value in its own right. If this is not the case, a simple solution is to record the number of elements in the training set that go down each branch and to use the most popular branch if the value for a test instance is missing [47].

7.1.6 Bayesian Network

As Scutari explains it [38, 39], a Bayesian Network (BN) is a graphical model, where nodes represent variables and arrows represent the probabilistic dependencies between them. The graphical structure $\mathcal{G} = (\mathbf{V}, A)$ of a BN is a directed acyclic graph (DAG), in which each node $v_i \in \mathbf{V}$ corresponds to a random variable x_i. The global probability distribution $P(\mathbf{x})$, with parameters Θ, can be factorized into smaller local probability distributions according to the arcs $a_{ij} \in A$. The form of the factorization is given by the Markov property of Bayesian Networks, which states that every random variable x_i directly depends only on its parents Π_{x_i}. If a node has no parents, the probability distribution is unconditional [45]. The main role of the network structure is to express the conditional independence relationships among the variables in the model [38]:

$$P(\mathbf{x}) = \prod_{i=1}^{N} P(x_i \mid \Pi_{x_i}; \Theta_{x_i}) \qquad \text{for discrete variables}$$

$$f(\mathbf{x}) = \prod_{i=1}^{N} f(x_i \mid \Pi_{x_i}; \Theta_{x_i}) \qquad \text{for continuous variables}$$

The Markov Blanket of a node x_i, denoted by $\mathrm{MB}(x_i)$, consists of its parents, its children, and its children's parents [43], as is shown in figure 7.8. This is the minimal set of attributes conditioned on which all other attributes are probabilistically independent of the target x_i. Knowing the values of the $\mathrm{MB}(x_i)$ is enough to determine the probability distribution of x_i [48].

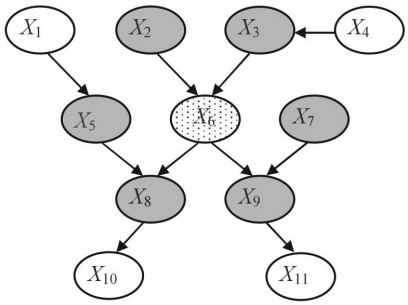

Figure 7.8: Example of a Bayesian Network, where the grey nodes represent the Markov Blanket of node X_6 [43].

Since a Bayesian Network is focused on modeling the (in)dependencies between all variables, there is no difference between the target variable and the other variables, from the perspective of the network. The values of its variables can be either known or unknown, and if a value is unknown, it is predicted by the network using the values of the variables it is dependent on. This means that these algorithms can deal very well with issues of uncertainty and noise [18], and missing values are not a problem for the model.

The classes of the variables are a limitation of Bayesian Networks. The algorithm is able to work with discrete, continuous or both types of variables. However, all continuous variables must have conditional linear Gaussian (or 'normal') distributions. Various ways have been proposed to transform other distributions to a Gaussian, to make them usable for BNs. When the parents of a node are discrete, the value of their child node is calculated using a probability table. When the number of possible values of a discrete variable is large, the resulting probability table will also become very large, which reduces the prediction accuracy. This makes it important for the discrete variables not to have too many possible values. Besides, a limitation of the model is that discrete nodes can not have continuous parents [40].

Unfortunately, it has been proved that learning an optimal Bayesian Network is NP-hard [17]. Learning a BN is this complex because there may be many random variables in a network, and each variable may take many values. Also a single random variable can have many parents, and finding the conditional distribution conditioned on all those parents increases the complexity [27]. In order to avoid this complexity for learning Bayesian Networks, Naive Bayes has attracted much attention from researchers [17].

Naive Bayes

The Naive Bayes Classifier (NBC) is the simplest form of a Bayesian Network. The method goes by this name because it naively assumes independence of all attributes, based on the Naive Bayes assumption [25]:

$$P(\mathbf{x}|y = C) = \prod_{i=1}^{D} P(x_i|y = C);$$

it is only valid to multiply probabilities when the events are independent [47]. Despite this unrealistic assumption, the resulting classifier is remarkably successful in practice, often competing with much more sophisticated techniques [35]. Figure 7.9 shows an example of an NBC.

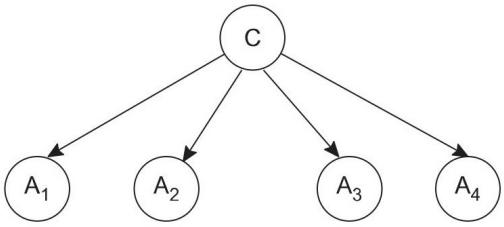

Figure 7.9: Example of a Naive Bayes Classifier, in which the predictive attributes $(A_1, \ldots, A_4)$ are conditionally independent given the class attribute (C) [17].

Naive Bayes is especially appropriate when the dimension of the feature space is high [12]. This method is also naturally robust to missing values since these are simply ignored in computing probabilities, and hence have no impact on the final decision [22]. However, Naive Bayes fails at predicting the target variable for a data point if the specific combination of its feature values does not occur in the training set [47].

Tree-Augmented Naive Bayes

The NB-assumption that all attributes of an instance are independent of each other, given the class of that instance, is very simple, but unrealistic [27]. Tree-Augmented Naive Bayes (TAN) improves on the Naive Bayes model by adding one more level of interaction among attributes of the system [27]. It relaxes the Naive Bayes attribute independence assumption by employing a tree structure, in which each attribute only depends on the class variable and one other attribute [49], as is shown in figure 7.10. The TAN can therefore be seen as a hybrid of a Decision Tree and a Naive Bayes Classifier. Designed to allow accuracy to scale up with increasingly large training datasets, the TAN model is a DT of nodes and branches with NBCs on the leaf nodes [45].

7.2 Model selection

In this section, I evaluate the models discussed in the previous section, with regard to our data set. Our data set has two main properties to take into account. First, all features are or can be discrete, and the target variable can be both discrete (ordinal) or continuous (ratio). Second, when the first prediction is made, usually the values of only a few variables are known, so the model should be able to handle this, and also be able to make a better prediction based on more values later in the process. Besides the properties of the data, the use of the model is also important for selecting a model. The model will be used to support humans in making choices, so the output of the model should be easily interpretable for a human.

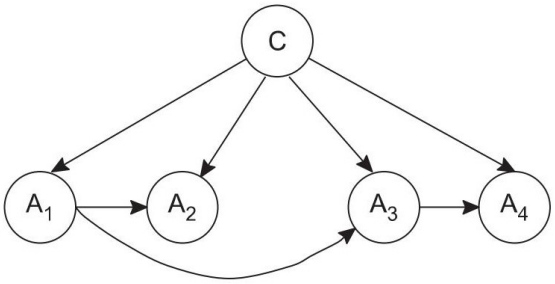

Figure 7.10: Example of an Tree-Augmented Naive Bayes Classifier, in which $(A_1, \ldots, A_4)$ are the predictive attributes and C is the class attribute [17]. Each attribute only depends on the class and at most one other attribute.

7.2.1 Variable classes

For Linear Regression (LR), Linear Classification (LC), Support Vector Machines (SVM) and k-Nearest-Neighbors (kNN), all feature variables have to be numeric. They all have ways of dealing with discrete variables, but this is only by making them numerical. The feature values of Logistic Regression (LogR), Neural Networks (NN), Decision Trees (DT) and Bayesian Networks (BN) can be both discrete and continuous. DTs and BNs do have a preference for few-valued discrete or normally distributed continuous variables. Concerning the discrete nature of our feature values, using LogR, NN, a DTs or BNs would be preferred.

For LR, the target variable has to be numerical. For SVMs, NNs, kNN, DTs and BNs, the target variable can be either discrete or continuous. An SVM is at it's best, however, when predicting a dichotomous variable. DTs and BNs prefer not to deal wit a not-normally distributed continuous target variable. For LogR and LC, the target variable has to be discrete, preferably dichotomous. Although the output of LogR is continuous, it is still a classification, presented as a continuous probability for such a class. Based on the target variable, we thus have the possibility to use all of the discussed techniques, but using LR, NNs or kNN would best fit our target variable FHT.

7.2.2 Missing values

LR, LC, LogR, SVMs and NNs are not able to work with missing values. Of course, it is a possibility for all of these techniques to use an imputation method before using the model, but this is not an optimal solution. The kNN algorithm is also not directly able to handle missing values, but the variables of which the value is missing could be simply left out of the feature space. DTs also don't have a natural way of predicting the value for a data point with missing values, but there are ways to work around this limitation, for example by taking the branch with the highest probability when the value of a certain node is missing. BNs are easily able to work with missing values, since they are equally as much built for predicting the values of feature variables as well as of the target variable. The main problem remaining is that Naive Bayes Classifiers fail when a particular attribute

value does not occur in the training set in conjunction with every class value. We can conclude that, with respect to the problem of missing values, it would be best to use a kNN algorithm, a Decision Tree or a Bayesian Network.

7.2.3 Interpretability

The model that is by far the least interpretable is a Neural Network. This model usually has hidden layers, which makes the algorithm by nature opaque. Linear models, in our selection those are LR, LC, LogR and SVMs, are conceptually very easy to understand. When put into practice, however, the models are hard to visualize when having more than three dimensions, which is equal to the number of features. Then, only a mathematical equation remains, which for most people is interpretable to some extent, but only with quite some effort. The kNN algorithm suffers from the same difficulty to visualize the feature space when dealing with more than three features, but for this method there is another way to make it more interpretable. This algorithm bases its decision on other data points: the k points closest to the one that is predicted. If the algorithm is programmed to not only show the human the predicted duration, but also what k data points that the decision is based on, this would contain a lot of useful information. Now only DTs and BNs remain. Both are inherently very interpretable, since they are by nature visualizations of the influence that certain features have on the target variable. There are still mathematical calculations behind the nodes and their branches, yet even these could be visualized by, for example, making the lines for strong connections or frequently taken paths thicker.

7.2.4 Overview

Table 7.1 contains a summary of the qualities and limitations of the discussed algorithms, with respect to our data set and problem. It may be concluded from this table that Decision Trees and Bayesian Networks are most suitable, and k-Nearest-Neighbors follows close behind. The Decision Tree algorithm is already implemented, and still being researched, by CQM, as is discussed in section 2.2. Therefore, I will not examine DTs in the next chapter.

Based on the table, it would seem reasonable to continue by only training and testing Bayesian Networks. However, it is hard to tell whether a BN or a kNN algorithm will perform better in practice, based only on this theoretical comparison, because their theoretical performance is very similar. It is a possibility that I did not take some factors into account that, without my knowledge, are of importance. It is also very probable that new features will become available in the future, which may be of another shape than our current features. I will therefore continue discussing the kNN and BN algorithms in the next chapter.

	LR	**LC**	**LogR**	**SVM**	**NN**	**kNN**	**DT**	**BN**
Discrete features	−	−	+	−	+	−	+	+
Target variable class	+	±	±	−	+	+	±	±
Missing values	−	−	−	−	−	±	±	+
Interpretability	±	±	±	±	−	+	+	+

Table 7.1: An overview of model properties with respect to our data set and problem. Compared algorithms: Linear Regression (LR), Linear Classification (LC), Logistic Regression (LogR), Support Vector Machine (SVM), Neural Network (NN), k-Nearest-Neighbors (kNN), Decision Tree (DT) and Bayesian Network (BN).

8 Results

This chapter contains both the implementation of our models and the evaluation of their outcome, respectively corresponding to step six and seven in the supervised learning process of Kotsiantis et al. [22].

8.1 Model implementation

In this section, I discuss what was needed to implement the BN and kNN algorithms. This is complemented by the R-code that I used to implement the algorithms. The original code also contains lines for e.g. loading the data and transforming the variables, which I did not find relevant to show in this thesis.

8.1.1 Bayesian Network

As described in section 7.1.6, a BN is a graphical model, where nodes represent variables and arrows represent the probabilistic dependencies between them. If the algorithm works with continuous variables, they have to be normally distributed for the BN to work optimally. Because our target variable FHT is not normally distributed, I chose to use the discretized version of it. All other variables are kept discrete as they were. Discrete variables should not contain too many values, because this will make the resulting probability tables enormous. For this reason I chose to implement the Tree-Augmented Naive Bayes (TAN), one of the variations of BNs, because this implementation is more suitable to work with the large number of possible FHT-values. Code 8.1 shows the lines that are used to build a TAN, and use this to predict FHT for the selected test set. Figures 8.1 and 8.2 show respectively a BN and a TAN of the collision/hindrance data set, generated by the `graphviz.plot` function in the code.

```
# create, show and fit a BN
bn = tree.bayes(data_train, "fht")
graphviz.plot(x=bn, layout='dot')
fitted = bn.fit(bn, data_train, method='bayes')

# use the fitted BN to predict FHT for test data
data_test$pred = predict(object=fitted, data=data_test, node="fht")
```

Code 8.1: Code for the Tree-Augmented Naive Bayes model to predict FHT. The package that is needed for these functions is bnlearn. The train and test objects are generated by code 5.1.

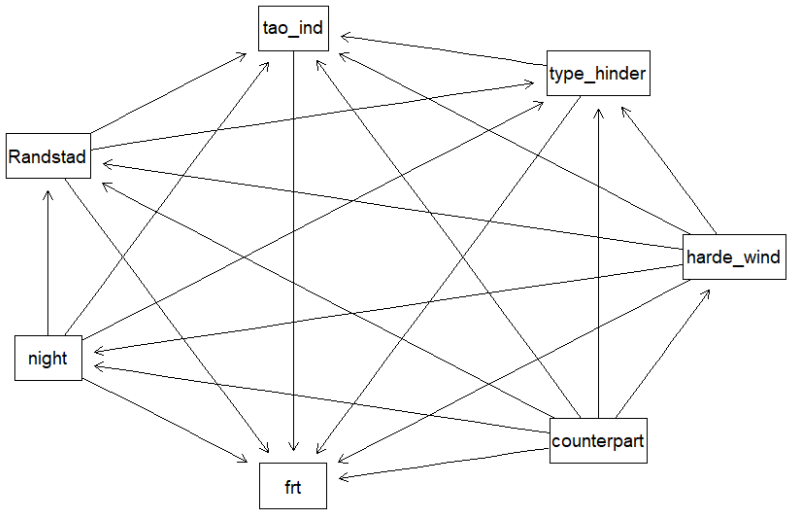

Figure 8.1: The Bayesian Network for collision/hindrance incidents, learned with hill-climbing.

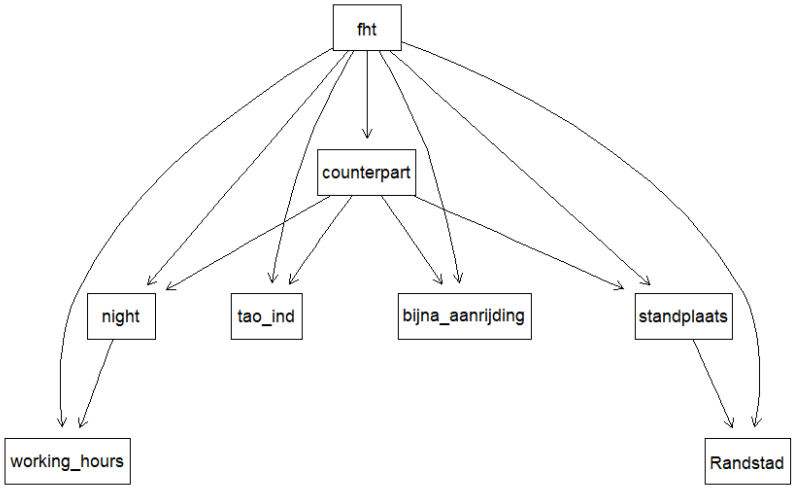

Figure 8.2: The Tree-Augmented Naive Bayes Classifier for collision/hindrance incidents.

8.1.2 k-Nearest-Neighbors

As discussed in section 7.1.4, the kNN algorithm is only able to work with numerical features. Unfortunately, most of our features are categorical. They can of course be treated as numerical values. The binary variables `working_hours` and `night` are discretized versions of the continuous variable `tijd_begin`, the time that the incident is reported.

Therefore this original time variable is used instead of the edited binary variables. The FHT variable has to be the discretized version, since kNN prefers to do classification. Code 8.2 shows the one line of code that is needed to build a kNN model in R.

```
# use kNN algorithm to predict FHT for test data
data_test$pred = knn(xTrain, xTest, yTrain, k=10, l=1)
```

Code 8.2: Tiny code for creating a kNN model to predict FHT. The package that is needed for this function is class. The train and test objects are generated by code 5.1.

In section 7.2.3, I have argued that the kNN algorithm scores quite highly on interpretability, because showing the set of nearest neighbors that the prediction is based on could give a human more insight in the reason for the predicted number. Code 8.3 shows a way of implementing this, applied to the collision/hindrance data set. The output shows the 10 data points of the training set that are closest to a given point in the test set, which have been used for the majority vote for the FHT of this test element.

```
# For every data point in xTest, kNN contains the indices of the
# 10 nearest points in xTrain.
> kNN = get.knnx(data=xTrain, query=xTest, k=10)

# Of only the first test element, find the data points in data_train
# that correspond to the indices in kNN.
> x_test = kNN[["nn.index"]][1,]
> data_train[x_test,]
index counterpart working_hours night Randstad standplaats tao_ind bijna_aanrijding fht
312             1             1     0        1           2       0                 0  15
22              1             1     0        1           2       1                 0 135
121             1             1     0        0           2       0                 0  60
975             1             1     0        0           2       0                 0  30
364             1             1     0        0           2       0                 0 180
859             1             1     0        0           2       0                 0  60
470             1             1     0        0           2       0                 0  60
796             1             1     0        0           2       0                 0  60
448             1             1     0        0           2       0                 0  30
616             1             1     0        0           2       0                 0  45

# The corresponding test element, to which the above 10 points are the nearest.
> data_test[1,]
index counterpart working_hours night Randstad standplaats tao_ind bijna_aanrijding fht pred
2               1             1     0        1           2       0                 0  30   60
```

Code 8.3: This R-code outputs the first element of the collision/hindrance test data, and the 10 data points of the training data that are nearest to this test element, according to the kNN algorithm. This example shows that the predicted FHT for the test element is 60, because this is the result of a majority vote between the 10 nearest training elements.

8.2 Model evaluation

8.2.1 Prediction errors

As explained in section 5.3, I have divided the data in a training set and a test set. This is done for the purpose of estimating performance of the prediction models. In this section, both the predictions of the TAN and the kNN algorithm are compared to the real FHT-values of the test set. The `table` function was used to output a prediction table, where the rows represent the predicted values and the columns represent real values of all elements in the test set. As an error measure, I chose to calculate both the root mean square error (RMSE) and the mean distance between the predicted and real values. The RMSE is a very popular error measure in literature, so I chose to calculate this for consistency. The mean distance, however, is much easier to understand, since it represents a time span. Because both error measures have a very different advantage over the other, I chose to use both. The RMSE is calculated by using `rmse` from the Metrics package in R. The mean distance is calculated by taking the absolute value of the difference between the predicted and real value, for each data point in the test set, and taking the mean of all differences.

The complete outputs of these tests are shown in appendix D. It should be noted that every time that the code is run, the output differs a little, because the data is divided in a different training set and test set. The outputs that I chose to show are the ones I found most representative, with a result that is more or less the average of multiple runs. Thus, this should be seen as an demonstrative example, not as the only true output of the model. For all tests, first the output is shown for prediction using all feature variables as evidence. Then the output is shown for the prediction made with only the variables that are most probably known when the initial prognosis is made, because in case the model would be implemented, it should be able to output both the initial and updated prognosis.

Table 8.1 shows an overview of the prediction errors, produced by both the TAN and kNN algorithms, on the test sets of our three incident type models. The performance of the TAN and kNN algorithm are similar for all data sets, as is the accuracy of predictions using all variables as evidence or only the initially known. The main difference in performance can be observed between incident types. It is shown that all mean distances are between 45 minutes and more or less one hour, which means that the difference between a generated prediction and the actual value for that data point will be on average 45 to 60 minutes.

8.2.2 Comparison to current prognosis

In the Spoorweb data set, the column `PrognoseINIduur` contains the initial prognosis generated by the model of CQM. For both section TOBS incidents and collision/hindrance incidents, I randomly selected 10 data points occurring in both Spoorweb and my SAP-based data set. Appendix D.4 contains the prediction tables and error rates for these 20 data points, using both the TAN and kNN algorithm. The predictions of CQM for the same data points are also known, and presented in the same appendix. This way it is possible to compare the performance of my algorithms with those currently implemented

		All variables		Initial variables	
		RMSE	*mean dist.*	*RMSE*	*mean dist.*
Rolling stock	*TAN*	67.39	48.83	70.17	50.08
	kNN	65.62	49.36	71.43	53.22
Section TOBS	*TAN*	87.36	63.35	89.05	60.95
	kNN	87.99	61.57	86.39	61.49
Collision/hindrance	*TAN*	63.68	47.19	63.20	45.63
	kNN	62.28	47.03	62.99	47.66

Table 8.1: An overview of error measures, of the results produced by the Tree-Augmented Naive Bayes (TAN) algorithm and the k-Nearest-Neighbors (kNN) algorithm. The root mean square error (RMSE) and the mean distance are both used as an error measure. The complete outputs are shown in appendix D.

at ProRail, although we have to keep in mind that this is a very small and maybe not completely representative sample.

Since the CQM model is only used for the initial prognosis, when not all possible information is yet available, it is not fair to compare its prognosis with my models using variables that are not initially known. Therefore, I applied every algorithm to the data twice: once with all selected features, and once with only the features that are known when the initial prognosis is made. For the section TOBS data, this meant deleting `contract_soort`, `overlapping_inc`, `Theoretisch_vervangingsjaar`, `standplaats`, `tao_ind`, `wind_compass`, and `oorzaak_groep`. For the collision/hindrance data, only `tao_ind` and `counterpart` were removed.

To make a final note, the CQM model has only been implemented since February 2018. All 20 data points that I selected are therefore from past this implementation. This is also the reason why I was not able to do this comparison for rolling stock failures. Spoorweb and ISVL do have a little time of overlapping data points, but this was before the CQM model was implemented, so there are no CQM predictions for the data that my model for rolling stock failures is able to classify.

Table 8.2 shows an overview of the prediction errors, produced by the TAN and kNN algorithms, and by the Decision Tree of CQM.

8.2.3 Optimistic and pessimistic prediction

In appendix B.2, an example of a Decision Tree built by CQM is shown. Each node of the tree contains a probability distribution of FHT. When a leaf is reached, the 65th percentile of this distribution is taken as the prognosis. Taking the median or mode of the distribution seems like a more sensible choice to reduce the prediction error. However, the reason for using the 65th percentile is that a pessimistic prognosis is much more practical than a prognosis that is too optimistic. If ProRail communicates to the train companies that the incident will be resolved at 2PM, but this happens to be at 2:30PM (optimistic prognosis), there is half an hour of time where the train companies already

		All variables		Initial variables	
		RMSE	*mean dist.*	*RMSE*	*mean dist.*
Comparison set TOBS	*TAN*	106.70	78.00	106.91	78.00
	kNN	104.14	81.00	99.84	73.5
	CQM	-	-	84.72	66.90
Comparison set col./hindr.	TAN	58.48	45.00	36.43	25.50
	kNN	67.92	55.50	58.67	43.50
	CQM	-	-	196.83	143.30

Table 8.2: An overview of error measures, of the results produced by both my algorithms and the one of CQM, which is currently implemented at ProRail. These predictions are made for 10 randomly selected data points of the section TOBS data set, and 10 points of the collision/hindrance data set. The root mean square error (RMSE) and the mean distance are both used as an error measure. The complete outputs are shown in appendix D.5.4.

have their trains ready to drive while this is not actually possible. They may have even informed passengers about specific trains that will start driving again, which will lead to a disappointment. However, when ProRail communicates to the train companies that the incident will be resolved at 2:30PM, but this is actually already the case at 2PM (pessimistic prognosis), the only effect is that the train table could have gotten back to normal half an hour earlier, than it would based on the prognosis. This is a much better outcome for all parties involved. Therefore, a pessimistic prediction is, for this problem, better than an optimistic prognosis.

Figure 8.3 shows ratios of over-estimation (pessimistic), under-estimation (optimistic) and correct predictions. These are all means taken over multiple predictions. The *CQM* bar represents the predictions of CQM on the two small test sets of 10 data points. A note should be made that the CQM-bar shows that 0% of the test data was correctly classified, while the other bars show a higher percentage in the 'correct' section. This is because the CQM model outputs the FHT in minutes, while my models work with bins of 15 minutes. Thus, in this comparison, the CQM model is much less likely to correctly classify data points than the other models. In figure 8.3, the *pred* bar stands for the overall predictions of both the TAN and kNN algorithms, applied to the data sets for our three incident types, presented in appendices D.1, D.2, and D.3. The *test* bar stands for the TAN and kNN predictions on the two small test sets of 10 data points. The results for these tests are presented in appendix D.4.

For both cases, there is also a *+30* bar added to the figure. These represent the same sets, but with all FHT predictions increased by 30 minutes. The reason for increasing the FHT by 30 minutes is to test the effect of predicting a higher FHT on both the under-estimation and the performance of the models. Both this approach and the amount of time are arbitrarily chosen, only to get an impression of what the effect would be of increasing the FHT. A more sophisticated approach would be to alter the models in such a way that a higher FHT is given as output, instead of modifying the output afterwards. The outputs of my provisional approach are shown in appendix D.5. Subsequently, figure 8.3 shows that increasing the FHT positively influences the ratio of over-estimated

and under-estimated FHTs.

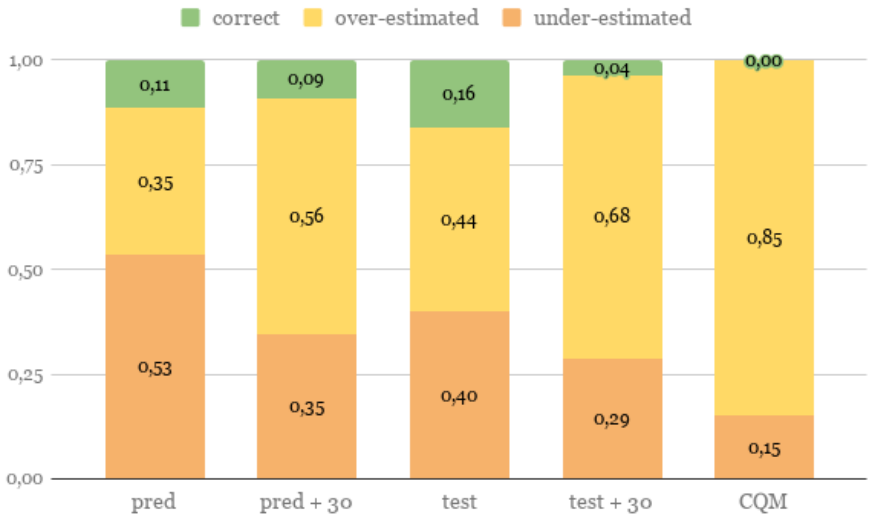

Figure 8.3: Ratio of over-estimation, under-estimation, and correct predictions for all predicted sets. Here *pred* stands for our three data sets for rolling stock incidents, section TOBSs and collision/hindrance incidents, *test* for both comparison selections of 10 data points, and *CQM* for the prognoses of CQM. For the sets with *+30* put behind the name, 30 minutes are added to each predicted FHT, to decrease under-estimation.

Increasing the predicted FHT by 30 minutes positively influences the ratio of under-estimations, but this comes with the question whether this does not have a huge negative impact on the prediction errors, which would make it less of an effective approach. Table 8.3 shows the prediction errors of all predictions where the FHT is increased with 30 minutes. If we compare these errors with the errors shown in tables 8.1 and 8.2, we can conclude the following. The error rates of predictions on the three incident-type models stay more or less the same when predicting with all variables as evidence. When using only the initially known variables as evidence, the error for rolling stock failures and section TOBSs even decreases, and increases only by a very small amount for collision/hindrance incidents. The errors for the comparison test set of 10 data points of the section TOBS set also decrease, and even become lower than the errors of CQM for the prognosis using only initially known variables. For the comparison set of 10 collision/hindrance data points, the error increases with about 20 minutes of mean distance, but stays far below the error of CQM. Thus, generally, increasing the predicted FHT by 30 minutes barely affects the prediction errors, but it does positively influence the ratio of over-estimated and under-estimated FHTs.

		All variables		Initial variables	
		RMSE	*mean dist.*	*RMSE*	*mean dist.*
Rolling stock	*TAN*	69.58	52.20	64.70	49.20
	kNN	65.34	52.31	70.12	54.89
Section TOBS	*TAN*	61.16	66.78	75.92	54.88
	kNN	77.36	58.02	78.52	59.13
Collision/hindrance	*TAN*	65.54	51.91	62.80	47.06
	kNN	62.69	49.66	64.46	50.68
Comparison set TOBS	*TAN*	92.95	75.00	90.50	69.00
	kNN	105.53	85.50	71.78	55.50
	CQM	-	-	84.72	66.90
Comparison set col./hindr.	TAN	77.07	69.00	49.97	46.5
	kNN	89.75	72.00	68.41	54.00
	CQM	-	-	196.83	143.30

Table 8.3: An overview of error measures of the prognosis made by both my algorithms and the one of CQM, which is currently implemented at ProRail. In this table, both the general predictions on test sets of all three incident types are shown, as well as predictions on the 20 randomly selected data points of the section TOBS and collision/hindrance sets. For all predictions that these errors apply to, the FHT was increased by 30 minutes, in an attempt to decrease under-estimation. The root mean square error (RMSE) and the mean distance are both used as an error measure.

9 Conclusion

The main question of this thesis, as posed in chapter 1, was: How can we accurately predict the recovery time for various types of railway incidents, using a dynamic model that updates with newly gained information? This question was divided into two subquestions. First: what information is needed to make accurate predictions of recovery time for various railway incidents? And second: how can we create a dynamic model that is able to update its prediction based on new information?

The first subquestion is mainly discussed in chapters 4, 5, and 6. The first aspect of the information that is needed for making predictions is the used data set. I have used the ISVL data as a basis for rolling stock failures, and the merge of SAP and SAPX for all other incident types. I can conclude that this was a good choice for exploratory research purposes, but it has to be noted that Spoorweb is used for the incident recordings during the incident. Thus, the variables in Spoorweb are the ones available to use when the model is brought into practice.

There are some relevant features that are only in SAP or ISVL, and not in Spoorweb, or in another data set that is only possible to merge with SAP/ISVL and not with Spoorweb. For these variables, I would advise to look for a replacement when the model would be implemented. These features are: `Randstad`, `contract type`, `contractor`, `Theoretisch_vervangingsjaar`, `standplaats`, `Mat`, `Rijk`, `Vervoerder`, `Drp`, `RangeerDrp`, `Act`, and `goederentrein`. However, there are also a few features that are only contained by Spoorweb, and not by SAP or ISVL, which may be relevant for prediction. These are `ProRailBedieningGebiedSoort` (type of section), `SchadeIndicatie` (whether there is damage), `LogistiekebeperkingType` (degree of train traffic restriction) and `Infrabeperkingstatus` (degree of infra restriction). These are features that could be used when working with Spoorweb, but which I have not tested. In addition, I would advise to keep looking for more potentially relevant data.

The answer to the second subquestion can be found in chapters 7 and 8. I can conclude that it is important that the model that is used is suitable for discrete feature variables, is able to handle missing values when the first prognosis is made, and is interpretable for a human. The conclusion that emerged from my comparison is that there are three suitable algorithms for our problem and data set: k-Nearest-Neighbors, Bayesian Networks and Decision Trees. Since DTs are already implemented by CQM, I continued by only further researching kNN and BNs, but I did not find a significant difference in performance between the two. Both algorithms are, as far as I can conclude, equally suitable to accurately predict FHT.

I chose to apply both the section TOBS model and the collision/hindrance model to 10 randomly selected data points, and compared the results to the prognoses of CQM for the same data points. For the 10 section TOBS predictions, the predictions of CQM had a lower error rate than mine, but for the 10 collision/hindrance predictions, the error rate of CQM was much higher than mine. However, in both cases this was only a sample of 10 data points, so the main thing to conclude here is that both models are not very accurate yet and quite unstable in their predictions. To sufficiently compare the models, it would be better to use a bigger data set for this. For me this was complicated, since there is very little overlap between the SAP data I used and the Spoorweb data that the CQM model is based on. For future research, this could be done by applying both models to their own data set and comparing their accuracy for different predictions, but first a couple of tests should be done to check whether it is valid to compare these results, which I have not done due to a limitation of time.

One important thing that the predictions by CQM do stand out in, is that they are usually over-estimating FHT instead of under-estimating, while this ratio is about 50/50 for the predictions generated by my models. Over-estimating is much more beneficial for ProRail and involved train companies than the under-estimating, which is why I increased all predicted FHTs by 30 minutes. This appeared to barely affect prediction errors, or even decrease them a little, while having a very positive influence on the ratio of over-estimated and under-estimated FHTs. From this, I would conclude that it may not be unreasonable to make prognoses by first using the models proposed in this thesis, and then increasing the predicted FHT by a certain amount to decrease the probability of under-estimating.

In my opinion, this thesis reflects a property of the current status of AI in society. In chapter 7, I stated interpretability as one of three important aspects of the models to choose from. The reason that this is important, is that when the model is implemented, it will operate as an assistant for humans, to help them make the final decisions. Pessimists tend to worry that AI machines will become smarter than us, and that they will make all our decisions for us. They are also concerned with social discontent as the amount of work available for people will diminish [24]. Although many jobs may be suitable to be executed by an AI, other tasks within these same jobs do not fit the criteria well [5]. As I believe it, and as 'doubters' state it according to Makridakis, it is wrong to believe that once computers have been provided with sufficiently advanced algorithms, they will be able to improve and then replicate the way our mind works. According to them, computers will not be able to achieve the human ability of being creative, as doing so requires breaking the rules and being anti-algorithmic. This would mean that all tasks requiring creativity, including innovative breakthroughs, strategic thinking, entrepreneurship, risk taking and similar ones could never, or at least not in the foreseeable future, be done algorithmically. Maybe, in a far future, a model will independently be able to accurately predict FHT, and consequently decide what approach is best to handle the incident. But there is no plausible way that this could happen in the near future, just as it does not seem very plausible that AI will soon be able to make all decisions for us and take all our jobs.

10 Discussion

In this chapter, I conclude my thesis by discussing the things I would advise to do different in future research. By this, I hope that future researchers will be able to efficiently build on the process of this thesis.

The first thing that could be improved is the imputation of missing values, discussed in chapter 5. Because I found it very difficult to tell whether there is a consistent reason for data not being recorded, I assumed that all missing values were missing completely at random. If this is not the case, if some of them are missing as a function of some other variable, this should be taken into account when imputing values. I would advise to future researchers to take this in consideration.

The next possible improvement lies in the choice to use either a discretized or numerical FHT. Bayesian Networks prefer working with either a discrete or a normally distributed numerical target variable. Since the numerical version of FHT, in minutes, is not normally distributed, I chose to use the discretized version for this algorithm. As Zilko demonstrates, it is also possible to use copulas, to transform the real FHT-distribution to a normal one, and make it suitable to work with for a BN or other models with a similar preference. For the purpose of loosing as little information as possible by applying transformations like discretization, working with copulas may be a proper approach to investigate more.

Moreover, I am convinced that I have made a valid choice by using SAP as a basis for the models of most incident types. For the collision/hindrance data set on the other hand, I should have used Spoorweb instead of SAP. This is because, as I discovered too late, Spoorweb contains more than three times as many data points on this type of incidents as SAP does, over a time span that is about three times as short. SAP appears to be focused mainly on technical incidents, while Spoorweb seems to leave those types of incidents out more often and holds more records on non-technical incidents.

Something else that should be taken into account when in the future using the algorithm for prediction, is the difference between the historical data that the model is trained on, and the current situation when the predictions are made. While we are working on algorithms to predict FHT, other departments at ProRail (or other companies) are working at improving procedures to decrease FHT. Therefore, in the time between the recordings of the historical data and the time the prediction is made, ProRail has possibly decreased the time that is needed to resolve incidents. These changes presumably take a lot of time, so the impact will also take a lot of time to be noticeable, but it might as well be kept in mind.

As a last notice, I would like to propose a complementary research. One of the reasons that FHT prediction is relevant is to determine the optimal recovery moment. Another important factor that has to be estimated before this can be done is passenger hindrance: the degree to which train passengers are delayed as a consequence of an occurred incident. Thus, researching ways to accurately estimate passenger hindrance would contribute to choosing the optimal recovery moment, and thereby to the relevance of research on FHT prediction.

11 Bibliography

[1] Y.S. Abu-Moustafa, M. Magdon-Ismail, and H.T. Lin. *Learning from data*, volume 4. AMLBook, New York, USA, 2012.

[2] D.G. Altman. Categorizing continuous variables. *British journal of cancer*, 64(5): 975, 1991.

[3] T.N. Bergsma. MASSIEF: Modelling disruption management as a multi-agent system to improve the prediction of the function recovery time. Master's thesis, Utrecht University, 2018.

[4] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[5] E. Brynjolfsson and T. Mitchell. What can machine learning do? Workforce implications. *Science Magazine*, 358(6370):1530–1534, 2017.

[6] S.E. Buttrey. Nearest-neighbor classification with categorical variables. *Computational Statistics & Data Analysis*, 28(2):157–169, February 1998.

[7] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[8] Y. Chevaleyre, F. Koriche, and J.D. Zucker. Rounding methods for discrete linear classification. In *Proceedings of the 30 th International Conference on Machine Learning*, pages 651–659, February 2013.

[9] CQM: Consultants in Quantitative Methods. Aanscherping prognose herstelduur incidenten. Slideshow presentation at ProRail, October 2017.

[10] T. Deoskar. Lecture s09: Stochastic Gradient Descent, Neural Networks. Slide show, September 2018. Machine Learning course at Utrecht University.

[11] A.R.T. Donders, G.J. Van Der Heijden, T. Stijnen, and K.G. Moons. A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59 (10):1087–1091, January 2006.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume Second Edition. Springer, 2017.

[13] D.M. Hawkins. *Identification of outliers*, volume 11. Chapman and Hall, London, 1980.

[14] D.M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[15] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *Advances in neural information processing systems*, pages 507–514, 2006.

[16] R. Hertwig, G. Barron, E.U. Weber, and I. Erev. Decisions from experience and the effect of rare events in risky choice. *American Psychological Society*, 15(8):534–539, 2004.

[17] L. Jiang, Z. Cai, D. Wang, and H. Zhang. Improving Tree Augmented Naive Bayes for class probability estimation. *Knowledge-Based Systems*, 26:239–245, August 2011.

[18] G.H. John and P. Langley. Estimating continuous distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., August 1995.

[19] R. Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A Decision-Tree hybrid. *Kdd*, 96:202–207, August 1996.

[20] D. Koller and M. Sahami. Toward optimal feature selection. *Stanford InfoLab*, pages 1–14, 1996.

[21] KNMI: Koninklijk Nederlands Meteorologisch Instituut. Data ophalen vanuit een script. '`https://www.knmi.nl/kennis-en-datacentrum/achtergrond/data-ophalen-vanuit-een-script`. Accessed: 2018/12/19.

[22] S.B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[23] M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, and B. Kenkel. *Package 'caret': Classification and Regression Training*, 6.0-84 edition, April 2019. `https://cran.r-project.org/web/packages/caret/caret.pdf`.

[24] S. Makridakis. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures*, 90:46–60, April 2017.

[25] K.P. Murphy. Naive Bayes Classifiers. *University of British Columbia*, October 2006.

[26] NS: Nederlandse Spoorwegen. Reizigersvervoer. '`https://www.ns.nl/over-ns/activiteiten/reizigersvervoer.html`. Accessed: 2019/01/23.

[27] H. Padmanaban. Comparative analysis of Naive Bayes and Tree Augmented Naive Bayes models. Master's thesis and graduate research, San Jose State University, May 2016.

[28] J.L. Peacock, O. Sauzet, S.M. Ewings, and S.M. Kerry. Dichotomising continuous data while retaining statistical power using a distributional approach. *Statistics in medicine*, 31(26):3089–3103, January 2012.

[29] K. Pelckmans, J. De Brabanter, J. A. Suykens, and B. De Moor. Handling missing values in Support Vector Machine Classifiers. *Neural Networks*, 18(5-6):684–692, 2005.

[30] ProRail. Prorail in cijfers. 'https://www.prorail.nl/over-prorail/wat-doet-prorail/prorail-in-cijfers, . Accessed: 2019/01/26.

[31] ProRail. Jaarverslag 2017. https://www.jaarverslagprorail.nl, . Accessed: 2019/01/26.

[32] ProRail. Hoofdlijnen beheerplan 2016. Beheerplan, ProRail B.V., May 2015.

[33] ProRail. Hoofdlijnen beheerplan 2018. Beheerplan, ProRail B.V., May 2017.

[34] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[35] I. Rish. An empirical study of the Naive Bayes Classifier. *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, 3(22):41–46, August 2001.

[36] L. Rosasco, E.D. Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 5(16):1063–1076, 2004.

[37] H. Schnack. Lecture s11: NN, crossvalidation, SVMs. Slide show, October 2018. Machine Learning course at Utrecht University.

[38] M. Scutari. Learning Bayesian Networks with the bnlearn R package. *Journal of Statistical Software*, 35(3), 2010.

[39] M. Scutari. Understanding Bayesian Networks; with examples in R. Slide show, January 2017. University of Oxford.

[40] P.P. Shenoy. Inference in hybrid Bayesian Networks using mixtures of Gaussians. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 428–436. AUAI Press, July 2006.

[41] H. Sigurdsson, B. Baldetorp, Å. Borg, M. Dalberg, M. Fernö, D. Killander, and H. Olsson. Indicators of prognosis in node-negative breast cancer. *New England Journal of Medicine*, 322(15):1045–1053, 1990.

[42] S.I.V. Sousa, F.G. Martins, M.C.M. Alvim-Ferraz, and M.C. Pereira. Multiple Linear Regression and artificial Neural Networks based on principal components to predict ozone concentrations. *Environmental Modelling & Software*, 22(1):97–103, 2007.

[43] S. Visweswaran and G.F. Cooper. Learning instance-specific predictive models. *Journal of Machine Learning Research*, 11:3333–3369, December 2010.

[44] De Volkskrant. Prorail: intercity kan iedere 7,5 minuut rijden. https://www.volkskrant.nl/nieuws-achtergrond/prorail-intercity-kan-iedere-7-5-minuut-rijden~b0bb53b6/, November 2018. Accessed: 2019/02/05.

[45] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Computer Communication Review*, 36(5):5–16, October 2006.

[46] N. de Wit. Development of a reliable prediction method for urgent infra-failure recovery times at ProRail B.V. Master's thesis, University of Twente, February 2016.

[47] I.H. Witten, E. Frank, M.A. Hall, and C.J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2017.

[48] Y. Zhang, Z. Zhang, K. Liu, and G. Qian. An improved IAMB algorithm for Markov Blanket discovery. *Journal of Computers*, 5(11):1755–1761, November 2010.

[49] F. Zheng and G.I. Webb. *Tree Augmented Naive Bayes*, pages 990–991. Springer US, Boston, MA, 2010.

[50] A.A. Zilko. *Mixed Discrete-Continuous Railway Disruption-Length Models with Copulas.* Doctoral thesis, TU Delft, 2017.

Appendices

A Abbreviations and concepts

A.1 Abbreviations

departments & people

AL: “Algemeen Leider” (General Leader)

AM: “Asset Management”

ICB: “Incidentenbestrijding” (Incident Control)

MKS: “Meldkamer Spoor” (Railway Alarm Room)

TRDL: “Treindienstleider” (Train Traffic Controller)

VL: “Verkeersleiding” (Traffic Control)

failures

UO: “Urgente Onregelmatigheid” (Urgent Irregularity, priority level 1 or 2)

DOT: “Dringende Onregelmatigheid met Tijdsafspraak” (Urgent Irregularity with Time Appointment, priority level 5)

NUO: “Niet Urgente Onregelmatigheid” (Not Urgent Irregularity, priority level 4 or 9)

NIC: “Niet In Controle” (Not In Control)

TAO: “Trein Aantastende Onregelmatigheid” (Train Affecting Irregularity)

TIS: “Trein Incident Scenario” (Train Incident Scenario)

TOBS: “Ten Onrechte Bezet Spoor” (Falsely Occupied Tracks)

FHT: “Functiehersteltijd” (Function Recovery Time)

GOB: “Gestoord Object” (Disturbed Object)

TVA: “Te Verklaren treinAfwijkingen” (Explainable Train Deficiencies)

other

DET: “Detectie” (Detection)

MON: “Monitoring”

PPLG: “Projectleidingsgebied” (Project Managing Area)

A.2 Concepts

departments & people

AL / General Leader: Supervisor who is present at the location of an incident.

AM / Asset Management: ProRail department responsible for managing all infrastructure assets.

ICB / Incident Control: ProRail department responsible for handling the practical aspects of incident recovery.

MKS / Railway Alarm Room: ProRail department that receives reports of incidents and coordinates the handling of the incident.

TRDL / Train Traffic Controller: Person who has his/her own area of the rail infrastructure to work in. This person determines where trains should and shouldn't drive, (s)he is capable of controlling switches and signs, and responsible for arranging a safe work place for people that have to enter the rail track.

VL / Traffic Control: ProRail department responsible for traffic control, deciding where trains are allowed to ride.

failures

ISVL: A data set containing information that was recorded during the handling of an incident. It has a relatively free form, focusing on transferring useful information about the incident between concerned parties, rather than recording the incidents for future use.

SpoorWeb: The newer version of ISVL. It is also used for exchanging information between parties, during the handling of an incident. It has improved relative to ISVL in that its variables are more structured. Besides, tasks are assigned more specifically to people, and it is possible to record who has done which task.

SAP: The recordings database of Asset Management. This database is created and maintained with the purpose of creating historical data for later use, opposed to the real-time use of ISVL and Spoorweb.

TIS / Train Incident Scenario: A number corresponding to certain characteristics of an incident, that is assigned to the incident to indicate its severity.

TOBS / Falsely Occupied Tracks: TOBS stands for "Ten Onrechte Bezet Spoor", which means 'falsely occupied tracks'. The network of rail tracks is divided in sections. Between the edges of two sections, an insulating weld is placed to separate them. When a train drives over this weld, it short circuits an electricity loop, which is registered as an occupation of the concerning track. Sometimes, a track occupation is registered despite there is no train present, which is called a 'section failure' or 'section TOBS'.

FHT / Function Recovery Time: The time span between the moment an incident is reported and the moment everything is restored, so the train table can get back to normal.

B Previous research: models for FHT prediction

B.1 Extended multi-agent system

This extended MAS (Multi-Agent System) approach for predicting FHT is designed by Bergsma [3]. In figure B.1, an example is shown of the prediction of FHT for a specific switch failure. The low blue dots represent the best performing extended MAS, measured by RMSE. The black dots represent extended prognosis. The green line is the optimal line. The orange crosses are the original prognosis. The numbers correspond to the extensions that influenced the adjustment of the prognosis. 1: Initial prognosis. 2: Bayesian Network. 3: Switch type. 4: Extended RVO. 5: No extension, just prognosis 1 confirmed. 6: extended tasks GL. 7: Weather. 8: Predicted failure cause. 9: Prognosis 2. 10: Prognosis 3.

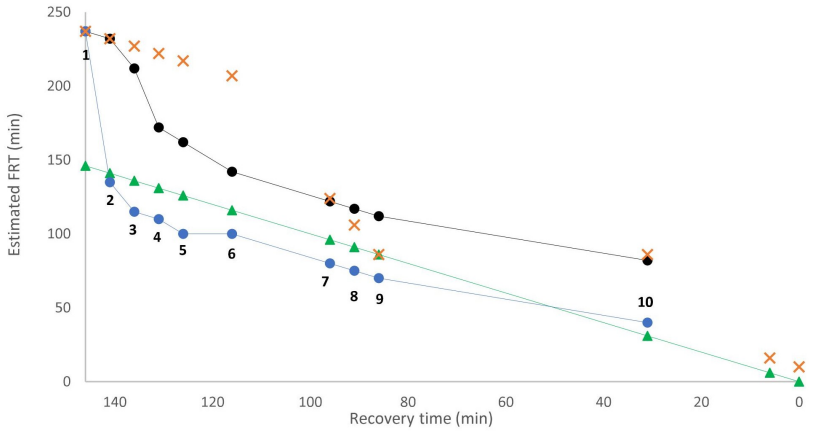

Figure B.1: Visualization of the multi-agent system approach for predicting FHT of a specific switch failure by Bergsma

B.2 Decision tree with probability distributions

This approach is designed by CQM [9], and is a combination of a Decision Tree and probability distributions that change in each node. In the leaves of the tree, the 65th

percentile of the FHT-distribution is taken as the prognosis. In figure B.2, the Decision Tree for rolling stock incidents is shown as an example.

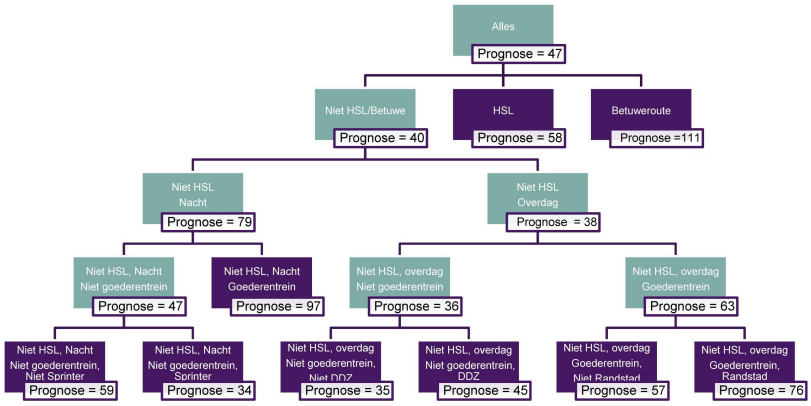

Figure B.2: Visualisation of the Decision Tree approach for predicting FHT for rolling stock incidents, created by CQM.

B.3 Probability distributions

This approach is designed by De Wit [46], and is based on probability distributions for each type of incident, influenced by a couple of variables. In figure B.3, prognoses are shown for five incident cases, taken as an example.

	50% Repaired	80% Repaired	90% Repaired	< 25 Minutes	<50 Minutes	< 75 Minutes
Case 1 – Model	51	138	205	37%	51%	62%
Case 1 – Reality	49	118	223	32%	54%	71%
Case 1 - Difference	+2	+20	-18	+5%	-3%	-9%
Case 2 – Model	29	84	129	50%	66%	77%
Case 2 – Reality	23	45	81	53%	80%	84%
Case 2 - Difference	+6	+39	+48	-3%	-14%	-7%
Case 3 – Model	29	54	120	50%	77%	92%
Case 3 – Reality	22	49	88	60%	80%	88%
Case 3 - Difference	+7	+5	+32	-10%	-3%	+4%
Case 4 – Model	64	113	192	22%	42%	60%
Case 4 – Reality	60	116	212	9%	44%	56%
Case 4 - Difference	+4	-3	-20	+11%	-2%	+4%
Case 5 – Model	34	62	130	44%	71%	87%
Case 5 – Reality	28	67	133	40%	75%	80%
Case 5 - Difference	+6	-5	-3	+4%	-4%	+7%
Average Difference	5	14	24	6.6%	5.2%	6.2%

Figure B.3: Visualisation of prognoses generated by the probability distribution approach for predicting FHT by De Wit

B.4 Mixed discrete-continuous Bayesian Network with copulas

This approach is designed by Zilko [50], and has the shape of a mixed discrete-continuous Bayesian Network. This model takes dependencies between variables into account, using copulas. In figure B.4, the model for Track Circuit (section) disruptions is shown, including marginal distributions for all variables, based on all disruptions in the database.

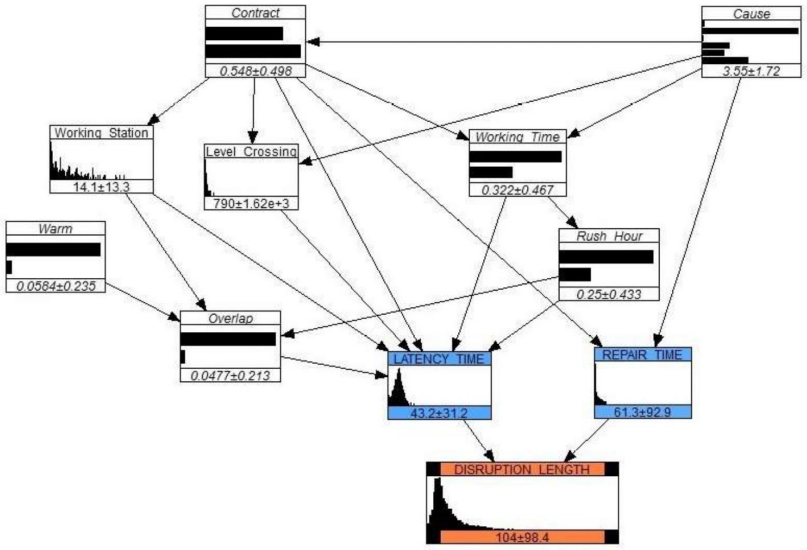

Figure B.4: Visualisation of the mixed discrete-continuous Bayesian Network with copulas for Track Circuit disruptions, by Zilko

C Feature explanation

C.1 Nominal variables

Nominal variables are a classification of groups that can not be placed in a specific order. An example of this is the categorization of biological species. Many of our relevant variables are nominal variables.

Contractor As explained in section 2.1, ProRail outsources track maintenance activities to contractors. Each contractor has its own way of working and is located somewhere else, which can cause differences in both latency time and repair time. The variable `aannemer` contains which contractor was responsible for recovery.

Contract type A couple of years ago, ProRail has introduced PGO-contracts ("prestatiegericht onderhoud"), which holds that the contractor gets paid based on the quality and efficiency of the performed work. Before this type of contract was introduced, all contracts were OPC ("output-processcontract"), which holds getting paid for the performed activities. The average FHT has proven to be lower with PGO-contracts than with OPC-contracts [50]. ProRail is slowly converting all contracts to PGO, but there are still many contractors with an OPC-contract. The variable `contract_soort` contains values `PGO` and `OPC` or `Overig` (other).

HSL or Betuwe route The HSL (high speed line) and the Betuwe route are two special parts of the railway system. The variable `HSL_betuwe` holds whether the incident occurs at the `HSL`, at the `Betuwe` route or somewhere else (`Overig`).

Location of base The variable `standplaats` holds information about the locations of the contractor's bases, represented by four numerical values.

Wind direction This feature stems from the imported weather data from KNMI [21]. Originally, the variable contains continuous, numerical values, representing the compass degree for the direction that the wind is coming from. I discretized the variable by assigning each number to the closest compass point, and using these four points as values for the variable `wind_compass`.

Driving characteristic The driving characteristic of a train is the way it is used in the infrastructure. Examples of this are the Intercity (`IC`), Sprinter (`SPR`) or international trains (`INT`). There are seven driving characteristics represented as values in the variable, and one `OVERIG` (other). The variable is named `Rijk`, for "Rijkarakteristiek".

Rolling stock type The rolling stock type, `mat_type` for 'materieeltype', represents inherent characteristics of a train or part of a train. The rolling stock failure can, for example, concern a `FLIRT`, which is a specific type of train that is mostly used as a Sprinter, for which the most occurring incident is that its doors refuse to close. There are eight different rolling stock types contained in the variable, and one `OVERIG` (other).

Train company While ProRail is responsible for the railway network, the trains that drive on it are property of other companies, like NS and Arriva. The variable `Vervoerder` contains separate values for the three largest companies, one value for all freight train companies and one for all others.

Activity All over the railway network, timetable control points are assigned to specific points on the tracks. These points are used to locate trains, by monitoring their movement from one timetable control point to another. Some of these points are train stations or shunting points, where a train is likely to arrive or depart. The variable `Act` contains the action of the concerned train at the timetable control point that it was last registered at, which can be either passing (`D`), arriving (`A`), or departing (`V`).

Cause group The variable `oorzaak_groep` contains the cause of the incident, categorized in nine groups. Weather and maintenance activities are examples of these groups. This cause is usually not known yet when the initial prognosis is made, but only after a diagnosis period.

Thing that the train collided with For the incidents belonging to the collision/hindrance incident type, there is usually some specific thing that the train has collided with or is hindered by. Whether this is an animal, person, object, vehicle or something else is recorded in the variable `counterpart`.

C.2 Dichotomous variables

Dichotomous variables are also, like nominal variables, a classification of groups, only they contain exactly two values. This class is equivalent to binary variables.

Time The starting time of an incident could be used as a value on it's own, but more information may be captured by creating other meaningful variables from this feature. I inferred the following binary features:

`night`: Whether the incident occurs at night, being `1` when the time is between 0AM and 6AM, and `0` otherwise.

`working_hours`: Whether the incident occurs during common working hours, being `1` when the time is between 7AM and 6PM, and `0` otherwise.

`contr_working_hours`: Whether the incident occurs during contractual working hours of the ProRail repair teams, being `1` when the time is between 7AM and 4PM, and `0` otherwise.

rush_hour: Whether the incident occurs during rush hour, being 1 when the time is between 6:30AM and 9AM, or between 4PM and 6:30PM, and 0 otherwise.

Randstad The 'randstad' is an area of the Netherlands that consists of Amsterdam, Rotterdam, Den Haag and Utrecht and the smaller cities between them. This variable holds whether the incident takes place in the randstad (1) or not (0).

Nature of the incident For collision/hindrance incidents, there is a great difference between a train being hindered by something, or actually colliding with it. For example, the consequences of running into a vehicle are very different from those of tree branches hanging too low and possibly causing damage to equipment. The variable type_hinder is either Aanrijding when the train has collided with something, or Hinder otherwise.

Shunting point All over the railway network, timetable control points are assigned to specific points on the tracks. These points are used to locate trains, by monitoring their movement from one timetable control point to another. Some of those points are also shunting points, which is a place where trains can be stored and picked up, when they are scheduled into or out of the timetable. Whether the timetable control point (Drp) that the concerned train is located at is also a shunting point, is recorded in the binary variable RangeerDrp.

Presence of overlapping incident Because there is a limited number of mechanics available in each contracted area, the presence of an overlapping incident in this area does affect the latency time. This feature is represented by the binary variable overlapping_inc. It's value is 1 when there is another incident in the data set that happened in the same contract area (contractgeb), within the last four hours before the current incident, and 0 otherwise.

Weather A lot of information about the weather can be obtained from KNMI. A couple of relevant features are selected and made binary by splitting the value at a critical point. I created the following binary features:

warm: Zilko found that high temperatures can trigger track sections to fail more or less simultaneously, which increases the probability of an overlapping incident [50]. The variable warm is 1 if temp_max > 25, and 0 otherwise.

veel_neerslag: This variable holds whether there is a lot of rainfall at the specific day. Its value is 1 if neersl_mm > 8 (more than 8 millimeters of rainfall), and 0 otherwise.

weinig_zicht: This variable is 1 if zicht_max < 70, which means that the distance of sight is limited to less than 70 meters, and 0 otherwise.

harde_wind: This variable is 1 if the wind speed is more than 20 kilometers per hour (wind_speed > 20), and 0 otherwise.

Freight train The type of train influences both the type of failures that occur and the impact that such a failure has. A type of train that is very different from most other types is the freight train, which transports goods instead of people. Whether the involved train is a freight train is contained in the binary variable `goederentrein`.

Train table adjusted When an incident occurs, it often happens that the part of the rail tracks where the incident took place has to be blocked, to be able to repair or clean something. When this happens, a 'versperringsmaatregel' (vsm) is applied, which means that other trains are temporarily not allowed to ride on a specific part of tracks. The variable `vsm_aangepast` indicates whether the vsm is adjusted, which in turn presumably indicates a complex incident scenario.

TAO indicator TAO stands for 'trein aantastende onregelmatigheid', or 'train affecting irregularity'. The variable `tao_ind` holds whether the train table is expected to be influenced by the incident.

C.3 Ordinal variables

Ordinal variables contain groups that can be placed in a specific order, but without a specified degree of difference between them. An example of this is judgment about something, where 'good' is higher than 'bad', but it is not possible to express the exact difference between them.

TIS Some incidents are assigned a TIS-scenario, which is a number ranging from 1 to 5. This number indicates the severity of the situation, of which a 1 means not severe (e.g. some trains have to be rescheduled) and a 5 means very severe (e.g. gas leak in a tunnel). This feature corresponds to the variable `tis`, with values varying from `1.1` to `5.3`.

C.4 Interval variables

The elements of interval variables are in a specific order, and there exists a degree of difference between them. It is nevertheless not possible to say that one element is 'two times bigger' then the other, e.g. there is no ratio between them. An example of this is temperature.

Year of replacement For the section TOBS incidents, our data set contains the variable `Theoretisch_vervangingsjaar`, which represents the expected year that the concerned section will be replaced. Originally, this variable is of the interval class, representing a year. For using it as input for the Bayesian Networks, I categorized these years in three groups, which makes it an ordinal variable.

C.5 Ratio variables

Ratio variables do allow for a ratio between the ranked elements. They also posses a meaningful and non-arbitrary zero value. In our original data there are some ratio variables, like the length of a certain section, but none of the selected features are of the ratio class.

D Results

D.1 Rolling stock failure

Both the TAN and kNN algorithm are applied to the rolling stock failure data twice: once with all features presented in section 6.2.1, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting `tis`, `vsm_aangepast`, `RangeerDrp`, and `Act`.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   0  0  0  0  0  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
30   0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
45   0  0  2  0  1  3   0   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0
60   5  3  5  8 11  9   4   5   2   3   2   1   0   1   0   0   0   0   1   0   0   1   0
75   1  8 12 16 15 12  12   7  12   7   2   3   3   1   0   0   1   1   0   0   0   0   0
90   0  2  4  7 13 11   5   5   3   3   5   2   1   1   3   0   2   0   0   0   0   0   0
105  1  1  2  4  5  9   6   4   2   1   3   2   0   1   0   1   0   0   0   0   0   0   0
120  0  0  4  4  0  2   4   5   2   0   2   1   3   2   2   3   0   0   2   1   1   0   0
135  0  0  0  2  0  0   2   1   0   0   1   2   0   1   0   0   0   0   0   0   0   0   0
150  0  0  0  0  1  0   2   0   2   0   1   1   0   1   0   0   0   0   0   0   0   0   0
165  0  0  0  2  3  0   2   2   2   5   1   1   2   0   0   1   1   1   0   1   0   0   0
180  0  0  0  0  0  1   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0
195  0  0  0  0  0  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
210  0  0  0  0  1  1   1   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0
225  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
240  0  0  0  0  1  0   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
255  0  0  0  0  0  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
270  0  0  0  1  0  0   1   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
285  0  0  0  0  0  0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1
300  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
315  0  0  0  0  1  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
330  0  0  0  0  0  0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 67.39048
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 48.82576
```

```
# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   0  0  0  0  1  0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
60   1  1  9  7  7  9   5   4   3   2   1   0   1   1   1   0   0   1   0   1   0   0   0
75   3  7 12 21 18 23  18  16  13  15   4   4   6   4   1   3   3   1   1   1   0   0   0
90   1  5  5 10 16 10   9   7   5   1   6   6   0   1   2   1   1   0   2   0   1   0   1
105  0  0  1  4  3  3   0   0   2   0   1   0   0   1   0   0   0   0   0   0   0   0   0
120  1  0  2  0  2  1   1   3   1   0   0   0   2   0   0   0   0   0   0   0   0   0   0
135  1  0  0  1  1  1   4   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0
150  0  0  0  0  0  0   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0
165  0  1  0  2  2  0   2   1   2   3   3   3   1   1   0   0   0   0   0   0   0   1   0
180  0  0  1  0  1  1   0   0   1   0   0   0   0   0   1   0   0   0   0   0   0   0   0
240  0  0  0  0  1  0   0   0   1   0   0   0   0   1   1   0   0   0   0   0   0   0   0
255  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
285  0  0  0  0  0  0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0
330  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
345  0  0  0  0  0  0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> rmse(data_test$pred, data_test$fht)
[1] 70.17025
> mean(data_test$pred_error)
[1] 50.07576
```

Code D.1: Output of the Tree-Augmented Naive Bayes algorithm applied to the rolling stock failure data set, predicted with FHT bins of 15 minutes.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.1, the time-variable `night` is replaced by a continuous time-variable `tijd_begin`, representing the reporting time of the incident.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
30   0  0  1  0  2  1   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
45   0  3  4  5  4  2   3   2   3   1   1   1   0   0   0   0   0   0   0   0   0   0   0
60   1  0  6 10  5  6   3   1   3   0   3   1   0   0   0   0   0   0   0   0   0   0   0
75   1  7  9 13 15  9   3  11   7   4   5   1   1   0   1   0   2   0   1   0   0   0   1
90   1  2  1  6 10  3   8   4   8   6   0   3   4   3   1   0   0   0   0   1   0   0   0
105  1  0  3  3  1  8   5   4   1   4   0   0   1   0   1   0   0   0   0   1   1   0   0
120  0  0  3  1  4  5   4   2   0   2   1   3   0   0   0   1   0   0   0   0   0   0   0
135  1  0  0  3  4  5   1   1   3   2   5   1   0   3   0   2   0   1   1   0   0   0   0
150  0  1  0  2  2  2   2   1   1   2   0   2   2   1   1   1   0   1   0   0   0   0   0
165  1  0  0  0  4  2   3   3   0   0   3   2   0   0   0   0   1   1   0   0   0   0   0
180  1  1  1  1  0  1   2   1   0   1   0   0   1   1   0   1   0   0   0   0   0   0   0
195  0  0  2  0  1  4   0   1   1   1   0   0   1   1   1   0   0   0   0   0   0   0   0
210  0  0  0  0  0  0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
225   0  0  0  0  0  0   1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
240   0  0  0  0  1  0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   0
255   0  0  0  0  0  0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
270   0  0  0  0  0  0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
330   0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 65.62202
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 49.35606

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   0  1  1  0  1  1   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
30   0  0  0  1  1  1   0   1   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0
45   1  0  4  7  7  2   2   2   1   2   1   0   1   1   0   0   0   0   0   0   0   0   0
60   3  2  5  5  9  9   8   4   4   0   1   2   2   2   1   1   0   0   0   1   0   0   0
75   1  6  3 12  8 11   7   5   7   7   2   3   2   2   1   0   3   1   1   0   0   0   1
90   1  2  9  7  7  6  12   9   3   5   2   1   1   1   2   1   0   0   1   1   0   0   0
105  0  1  3  3  8  8   1   4   6   1   3   2   0   0   0   0   0   0   0   0   1   1   0
120  1  0  1  3  3  1   6   0   1   2   2   2   1   1   0   1   1   0   0   0   0   0   0
135  0  0  0  6  5  2   0   2   3   1   2   3   1   0   1   1   0   0   1   0   0   0   0
150  0  0  1  0  2  2   1   1   1   0   2   1   0   1   1   0   0   0   0   0   0   0   0
165  0  1  1  0  1  2   1   1   1   1   2   0   1   0   0   1   0   1   0   0   0   0   0
180  0  0  0  1  0  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
195  0  1  1  0  0  2   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
210  0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
225  0  0  0  0  0  1   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0
240  0  0  0  0  1  0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
255  0  0  0  0  0  0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 71.42622
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 53.2197
```

Code D.2: Output of the kNN algorithm applied to the rolling stock data set. Predicted with FHT bins of 15 minutes, $k = 10$, and $l = 0$.

D.2 section TOBS

Both the TAN and kNN algorithm are applied to the section TOBS twice: once with all features presented in section 6.2.2, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting `contract_soort`, `overlapping_inc`, `Theoretisch_vervangingsjaar`, `standplaats`, `tao_ind`, `wind_compass`, and `oorzaak_groep`.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   2  4  6  1  1  0   3   4   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0
30   4 10  9  5  3  3   1   0   1   1   0   1   2   0   1   1   0   0   0   0   0   0   0
45   2  9 14 17 11  7   3   4   3   2   1   1   0   1   1   2   2   1   1   1   1   0   0
60   5  5  7  7  4  8   5   5   3   3   3   2   0   1   0   1   0   0   1   0   0   0   0
75   1  7  4  3  2  4   3   3   4   2   2   0   1   1   1   0   0   0   0   1   0   0   0
90   1  0  2  1  4  1   1   0   1   2   1   2   2   0   0   0   0   0   0   0   0   0   0
105  0  0  0  0  0  0   2   0   1   0   1   1   2   0   0   0   0   0   0   0   0   1   0
120  1  0  1  4  4  0   0   1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
135  1  0  2  0  0  1   1   1   0   0   0   1   0   0   0   0   1   0   0   1   0   0   0
150  2  2  1  1  4  2   1   0   0   1   1   0   0   1   0   0   0   2   1   0   1   0   0
165  0  0  0  1  0  1   0   0   0   0   0   2   0   1   0   0   1   0   0   0   1   0   0
180  1  1  1  0  0  0   2   1   1   0   2   0   0   1   1   0   0   0   0   0   0   0   0
195  0  0  0  1  1  2   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
210  2  1  0  1  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
225  0  0  0  1  0  0   0   1   0   2   0   0   0   0   0   0   0   0   0   0   0   1   0
240  0  0  1  0  0  0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
255  0  0  1  1  0  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
285  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
300  0  0  0  1  0  0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
315  1  0  0  0  0  1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
330  0  0  0  0  0  0   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 87.36146
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 63.34711
```

```
# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   0  1  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
30   5  5  8  7  5  3   2   3   2   1   1   3   2   2   1   0   0   2   0   1   0   0   0
45  14 25 18 24 19 17  13  10   9   7   6   6   6   4   4   2   2   1   3   1   3   2   0
60   3  7 15 11  9  8   6   7   3   5   2   0   0   1   0   1   1   1   0   1   0   0   0
75   0  1  4  1  1  1   0   0   0   0   0   1   0   0   0   1   1   0   0   0   0   0   0
90   0  0  2  1  0  1   3   1   0   0   0   0   0   0   0   0   0   0   0   1   0   1   0
105  1  0  1  1  1  0   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   0
120  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
150  0  0  1  0  0  0   0   1   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 89.048
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 60.95041
```

Code D.3: Output of the Tree-Augmented Naive Bayes algorithm applied to the section TOBS set, predicted with FHT bins of 15 minutes.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.2, the time-variables `night`, `contr_working_hours` and `rush_hour` are replaced by a continuous time-variable `tijd_begin`, representing the reporting time of the incident. The `warm`-variable is brought back to the continuous `temp_max` temperature-variable where it was originally discretized from. The variable `Theoretisch_vervangingsjaar`, that was discretized according to section 6.2.2, is taken as a numerical variable representing years of (re)placement.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   1  0  1  3  1  2   0   2   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0
30   5  9  7  9  5  5   1   2   2   1   1   1   1   1   1   0   2   1   1   0   0   1   0
45   2  7 13 12 10  6   4   7   6   0   5   2   3   1   2   3   1   1   0   0   1   0   0
60   7 11 11  6  9  4  11   4   2   6   5   4   2   3   0   0   0   0   0   1   0   0   1
75   2  4  6  8  2  4   3   3   0   1   0   2   2   0   1   0   0   0   0   2   1   0   0
90   3  1  1  3  4  5   0   3   2   1   0   0   0   1   0   0   0   0   0   0   0   0   0
105  2  4  6  2  4  3   3   0   0   2   0   0   0   0   0   1   0   0   1   1   0   1   0
120  1  1  1  1  0  0   1   1   1   1   0   1   0   1   1   0   0   2   0   0   0   0   0
150  0  0  0  0  0  0   0   0   2   1   0   0   0   1   0   0   0   0   0   0   0   0   0
165  0  1  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
180  0  0  0  0  0  1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
210  0  1  1  1  0  0   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   0
270  0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
285  0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
> rmse(data_test$pred, data_test$fht)
[1] 87.98713
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 61.57025

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
15   2  1  3  0  6  1   0   2   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
30   3  7  9  3  4  3   4   3   2   2   1   2   0   0   1   1   0   0   0   0   0   1   0
45   8 12 14 22  4 11  11   4   6   5   1   3   3   4   1   1   1   1   1   1   1   0   1
60   6 13  5  6  7  7   5   6   3   3   3   3   2   2   1   1   1   1   0   1   0   0   0
75   1  2  6  2  7  5   1   4   3   0   2   0   2   0   0   1   1   0   1   0   0   0   0
90   0  4  4  1  2  1   1   0   0   1   1   0   0   2   0   0   0   0   0   0   1   1   0
105  1  0  3  6  3  1   0   0   0   0   1   0   1   0   0   0   1   1   0   2   0   1   0
120  0  0  2  2  2  0   1   3   0   0   2   2   0   0   0   0   0   0   0   0   0   0   0
135  0  0  0  1  0  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
150  1  0  0  0  0  0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0   0
165  1  0  1  1  0  0   0   0   1   1   0   0   0   0   1   0   0   0   0   0   0   0   0
180  0  0  0  0  0  0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
210  0  0  0  1  0  1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
255  0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
285  0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 86.38756
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 61.4876
```

Code D.4: Output of the kNN algorithm applied to the section TOBS data set. Predicted with FHT bins of 15 minutes, $k = 20$, and $l = 0$.

D.3 Collision/hindrance incidents

Both the TAN and kNN algorithm are applied to the section TOBS twice: once with all features presented in section 6.2.3, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting `tao_ind` and `counterpart`.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
15   1  0  0  1  1  0   0   0   0   1   0   0   0   0   0   0
30   2  2  0  3  1  1   1   1   0   0   1   0   0   0   0   0
45   0  0  2  2  1  1   1   1   0   0   0   0   0   0   0   0
60   3  4  3  6  4  2   2   2   1   2   0   0   2   1   0   0
75   0  1  1  0  0  0   0   0   0   0   0   0   0   0   1   1
90   0  2  5  1  2  2   1   1   2   0   0   0   0   0   0   0
105  0  0  0  0  1  1   0   0   1   0   1   1   0   0   0   0
120  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0
135  1  2  0  0  0  1   2   1   0   1   0   1   0   0   0   0
165  0  0  1  1  0  0   0   0   0   0   1   0   0   0   0   0
210  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0
240  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 63.67643
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 47.1875

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
30   2  0  0  1  0  1   0   1   0   1   0   0   0   0   0   0
45   1  6  4  4  4  2   2   1   2   1   1   2   0   0   1   0
60   3  4  7  9  4  7   3   4   2   1   2   0   2   1   0   0
90   0  0  1  1  1  0   1   0   0   1   0   0   0   0   0   1
120  1  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0
195  0  1  0  0  0  0   1   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 63.19612
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 45.625
```

Code D.5: Output of the Tree-Augmented Naive Bayes algorithm applied to the collision/hindrance data set, predicted with FHT bins of 15 minutes.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.3, the time-variables `night` and `working_hours` are replaced by a continuous time-variable `tijd_begin`, representing the reporting time of the incident.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
15   1  0  0  0  0  1   1   0   1   0   0   0   0   1   0   0
30   1  1  2  3  4  2   0   0   0   1   0   0   0   0   1   0
45   1  3  2  1  3  1   1   0   0   1   0   0   0   0   0   0
60   1  2  3  4  2  1   3   2   2   0   0   0   1   0   0   0
75   1  2  1  3  0  2   1   2   0   0   0   1   0   0   0   1
90   1  3  2  1  1  0   0   0   0   0   0   0   0   0   0   0
105  0  0  1  0  0  0   0   0   0   0   1   0   0   0   0   0
120  0  0  0  1  0  0   0   1   0   1   0   0   1   0   0   0
135  1  1  0  1  0  0   0   1   1   0   0   1   0   0   0   0
150  0  0  1  0  0  0   0   0   0   0   1   0   0   0   0   0
165  0  0  0  0  1  2   1   0   0   0   1   0   0   0   0   0
180  0  0  0  0  0  0   0   0   0   1   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 62.28087
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 47.03125

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
15   0  0  1  0  1  2   0   0   1   0   1   0   0   0   0   0
30   0  0  1  1  3  1   1   0   0   0   0   0   0   0   0   0
45   2  2  2  2  1  1   0   0   2   2   1   0   1   0   0   0
60   0  4  5  4  1  0   1   0   0   0   0   0   1   1   0   0
75   0  1  1  1  3  2   3   3   0   0   0   0   0   0   0   0
90   2  3  2  3  1  1   1   2   1   1   1   0   0   0   1   1
105  0  1  0  1  0  2   0   0   0   0   0   1   0   0   0   0
120  2  1  0  2  1  0   0   1   0   0   0   0   0   0   0   0
150  0  0  0  0  0  0   1   0   0   1   0   1   0   0   0   0
180  1  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 62.99181
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 47.65625
```

Code D.6: Output of the kNN algorithm applied to the collision/hindrance data set. Predicted with FHT bins of 15 minutes, $k = 10$, and $l = 0$.

D.4 Test sets for comparison

For both section TOBS incidents and collision/hindrance incidents, I randomly selected 10 data points occurring in both Spoorweb and my SAP-based data set. The outputs of the code in this section contain error rates of my predictions of these 20 data points, using both the TAN and kNN algorithm, and the predictions of CQM for the same incidents.

Tree-Augmented Naive Bayes

```
# Section TOBS data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
30   1  0  0   0   0   1   0   0
45   0  0  0   1   0   0   0   1
60   0  1  0   0   0   0   1   0
75   1  0  0   0   1   0   0   0
120  0  0  1   0   0   0   0   0
135  0  0  0   1   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 106.7005
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 78

# Section TOBS data, only initially known variables
> table(data_test$pred, data_test$fht)
   30 45 60 105 120 165 195 300
30  0  0  0   0   1   0   1   0
45  1  1  1   1   0   1   0   0
60  1  0  0   1   0   0   0   1
> rmse(data_test$pred, data_test$fht)
[1] 106.9112
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 78

# Collision/hindrance data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
45   1  1  0  0   0
60   1  0  1  0   1
75   1  0  1  0   0
90   1  0  0  0   0
135  1  0  0  0   0
165  0  0  0  1   0
> rmse(data_test$pred, data_test$fht)
[1] 58.48077
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 45
```

```
# Collision/hindrance data, only initially known variables
> table(data_test$pred, data_test$fht)
   30 45 60 75 150
45  1  1  0  0   0
60  3  0  2  1   1
75  1  0  0  0   0
> rmse(data_test$pred, data_test$fht)
[1] 36.43487
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 25.5
```

Code D.7: Prognoses made by the Tree-Augmented Naive Bayes algorithm, for 10 data points of the section TOBS data and 10 points of collision/hindrance data, to be compared with the CQM prognoses.

k-Nearest-Neighbors

```
# Section TOBS data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
15   0  0  0   0   1   0   0   0
30   0  0  0   0   0   0   1   0
45   0  1  0   1   0   1   0   0
75   1  0  1   0   0   0   0   0
90   0  0  0   1   0   0   0   1
105  1  0  0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 104.1393
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 81

# Section TOBS data, only initially known variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
30   0  0  0   0   1   0   1   0
45   1  1  0   1   0   0   0   0
60   0  0  1   0   0   0   0   0
75   0  0  0   0   0   1   0   0
90   0  0  0   1   0   0   0   1
120  1  0  0   0   0   0   0   0
> rmse(data_test$pred, data_test$fht)
[1] 99.83737
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 73.5
```

```
# Collision/hindrance data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
30   0  1  0  0   0
45   1  0  0  0   0
60   0  0  0  1   1
75   1  0  1  0   0
105  2  0  0  0   0
135  0  0  1  0   0
165  1  0  0  0   0
> rmse(data_test$pred, data_test$fht)
[1] 67.91539
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 55.5

# Collision/hindrance data, only initially known variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
30   1  0  0  0   0
45   1  0  0  1   0
60   0  0  2  0   0
75   0  0  0  0   1
90   0  1  0  0   0
105  2  0  0  0   0
150  1  0  0  0   0
> rmse(data_test$pred, data_test$fht)
[1] 58.67282
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 43.5
```

Code D.8: Prognoses made by the k-Nearest-Neighbors algorithm, for 10 data points of the section TOBS data and 10 points of collision/hindrance data, to be compared with the CQM prognoses.

Prognosis CQM

```
# CQM prognosis for section TOBS data
> table(data_test$prog_cqm, data_test$fht)
    30 45 60 105 120 165 195 300
124  1  0  0   1   1   0   0   0
132  0  1  1   1   0   1   0   1
168  1  0  0   0   0   0   1   0
> rmse(data_test$prog_cqm, data_test$fht)
[1] 84.72367
> data_test$cqm_error = abs(data_test$prog_cqm - data_test$fht)
> mean(data_test$cqm_error)
[1] 66.9
```

```
# CQM prognosis for collision/hindrance data
> table(data_test$prog_cqm, data_test$fht)
    30 45 60 75 150
49   1  0  0  0   0
134  1  0  0  1   0
174  1  0  0  0   1
177  2  0  0  0   0
186  0  1  1  0   0
582  0  0  1  0   0
> sqrt(mse(data_test$prog_cqm, data_test$fht))
[1] 196.8322
> data_test$cqm_error = abs(data_test$prog_cqm - data_test$fht)
> mean(data_test$cqm_error)
[1] 143.3
```

Code D.9: Prognoses made by the Decision Tree model of CQM, for 10 randomly selected data points of the section TOBS data and 10 points of collision/hindrance data.

D.5 Pessimistic prediction

As explained in section 8.2.3, it is better to predict FHT pessimistically than optimistically. To the output shown in this section, 30 minutes is added to every prediction. This barely increases the prediction errors, but it does decrease the number of under-estimations.

D.5.1 Rolling stock failure

Both the TAN and kNN algorithm are applied to the rolling stock failure data twice: once with all features presented in section 6.2.1, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting `tis`, `vsm_aangepast`, `RangeerDrp`, and `Act`. To decrease under-estimation, 30 minutes are added to all predicted FHTs.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   0  0  0  0  0  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
60   0  0  1  0  0  0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0
75   0  0  0  3  2  3   1   2   1   2   0   0   1   1   0   0   0   0   0   0   0   0   0
90   2  2  9  8 13  7  10   3   2   2   0   0   0   1   0   0   0   0   0   0   0   0   0
105  2  8 13 18 16 21  11   6   8   3   6   3   1   1   0   1   0   0   1   0   0   0   1
120  2  2  4  5  7  4   5   9   4   6   3   5   2   1   1   0   0   2   0   1   0   0   0
135  1  2  0  2  4  1   2   3   4   3   2   3   1   1   0   1   1   0   0   0   0   0   0
150  0  0  2  1  3  1   1   2   2   0   0   0   0   1   1   1   0   0   0   0   0   0   0
```

```
165  0  0  0  4  0  0   2   2   1   1   0   2   2   1   0   1   0   0   0   0   1   0   0
180  0  0  0  0  1  3   1   1   1   0   1   0   1   0   0   0   1   0   0   0   0   1   0
195  0  0  1  0  3  6   4   1   2   4   3   1   1   0   2   1   0   0   1   1   0   0   0
210  0  0  0  0  0  0   0   0   1   1   0   0   1   0   0   0   0   0   1   0   0   0   0
225  0  0  0  1  0  1   0   1   0   0   1   0   0   1   2   0   2   1   0   0   0   0   0
240  0  0  0  1  1  1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
255  0  0  0  1  0  0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
270  0  0  0  0  0  0   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0   0   0
285  0  0  0  1  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
330  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
345  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
360  0  0  0  0  0  0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 238, under-estimation: 122, correct: 36
> rmse(data_test$pred, data_test$fht)
[1] 69.57664
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 52.19697

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
60   0  0  0  0  0  0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0
75   0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
90   0  2  3  4  5  3   4   1   0   0   0   1   1   0   1   0   0   0   0   0   1   0   1
105  4  6 17 34 29 30  22  18  16  11   8   6   6   6   3   3   1   1   1   1   0   0   0
120  0  4  8  6 14  9   9   5   4   9   4   2   2   2   2   0   1   0   0   0   0   0   0
135  0  0  0  0  0  1   0   0   0   0   0   0   0   1   0   1   0   0   1   0   0   0   0
150  1  1  2  0  5  2   1   3   3   0   0   1   2   0   0   0   0   0   0   1   0   0   0
165  1  1  0  0  0  0   2   0   0   1   1   1   0   0   0   0   1   0   0   0   0   0   0
195  0  0  0  1  0  1   1   2   1   1   5   2   0   0   0   1   1   2   1   0   0   1   0
210  0  0  0  0  0  0   0   2   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0
270  0  0  0  0  0  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
285  1  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
360  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
375  0  0  0  0  0  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 230, under-estimation: 135, correct: 31
> rmse(data_test$pred, data_test$fht)
[1] 64.69772
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 49.20455
```

Code D.10: Output of the Tree-Augmented Naive Bayes algorithm applied to the rolling stock failure data set, predicted with FHT bins of 15 minutes. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.1, the time-variable **night** is replaced by a continuous time-variable **tijd_begin**, representing the reporting time of the incident.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
60   0  0  0  3  2  1   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
75   0  1  3  3  6  2   3   1   0   1   3   0   1   1   0   0   0   1   0   0   0   0   0
90   0  6  5 11 12  7   9   8   2   2   2   2   2   0   1   0   0   0   0   0   0   0   0
105  4  2  9  7 10 15   4   3   7   4   1   3   2   2   1   1   0   0   0   0   1   0   0
120  1  2  6  7  6  9   1   4   5   4   2   3   2   0   2   1   2   1   0   0   0   0   0
135  0  2  2  5  5  6   9   3   2   1   5   2   1   0   0   2   1   0   0   0   0   0   0
150  0  1  1  2  3  3   4   1   2   3   0   1   0   3   0   1   0   0   1   0   0   1   0
165  1  0  3  3  5  1   2   3   4   1   3   0   0   0   0   0   0   0   0   0   0   0   0
180  1  0  1  2  1  2   5   2   1   3   0   0   3   1   1   0   1   0   0   1   0   0   1
195  0  0  0  1  1  1   1   1   1   0   0   1   0   1   0   0   0   1   1   1   0   0   0
210  0  0  0  0  0  0   0   1   3   1   0   1   0   0   1   0   0   0   1   0   0   0   0
225  0  0  0  0  0  1   1   2   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0
240  0  0  0  1  1  0   0   1   1   2   1   0   0   1   0   0   0   0   0   0   0   0   0
# over-estimation: 237, under-estimation: 136, correct: 32
> rmse(data_test$pred, data_test$fht)
[1] 65.34437
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 52.31061

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   0  0  1  0  0  2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
60   1  0  0  1  5  1   1   0   1   2   0   0   0   0   0   0   0   0   0   0   0   0   0
75   0  0  2  5  6  2   2   2   3   2   1   1   2   2   1   0   0   0   0   0   0   0   0
90   1  5  4  4 14  6   7   9   4   1   0   3   2   2   0   0   1   0   0   0   1   0   0
105  1  3 11  8 10 11   6   4   7   5   8   1   2   1   0   2   0   0   0   1   0   0   0
120  1  1  4  8  7  9   6   4   6   4   0   3   0   1   1   1   0   2   1   0   0   0   0
135  1  1  2  6  3  3   3   4   3   2   3   0   0   0   1   1   1   0   0   0   0   0   0
150  2  1  0  6  1  3   3   4   2   1   3   0   0   0   1   0   0   0   0   0   0   1   0
165  0  1  3  2  4  4   1   1   0   1   0   3   1   1   1   0   0   0   1   0   0   0   1
180  0  1  1  1  1  1   3   0   1   0   1   2   2   1   0   0   1   0   0   0   0   0   0
195  0  1  1  3  0  4   4   0   1   4   0   1   0   0   0   1   0   1   1   1   0   0   0
210  0  0  0  1  1  1   2   1   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0
225  0  0  0  0  0  0   1   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0
240  0  0  0  0  0  1   0   1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
255  0  0  0  0  0  0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
270  0  0  1  0  0  0   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
300  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
315  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
# over-estimation: 227, under-estimation: 139, correct: 31
```

```
> rmse(data_test$pred, data_test$fht)
[1] 70.1176
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 54.88636
```

Code D.11: Output of the kNN algorithm applied to the rolling stock data set. Predicted with FHT bins of 15 minutes, $k = 10$, and $l = 0$. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

D.5.2 section TOBS

Both the TAN and kNN algorithm are applied to the section TOBS twice: once with all features presented in section 6.2.2, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting contract_soort, overlapping_inc, Theoretisch_vervangingsjaar, standplaats, tao_ind, wind_compass, and oorzaak_groep. To decrease under-estimation, 30 minutes are added to all predicted FHTs.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   0  3  2  2  1  1   1   1   0   1   0   0   0   1   0   0   1   0   0   0   0   0   1
60   3  6  8  7  6  1   4   3   2   1   0   1   0   0   1   0   0   0   0   0   1   0   0
75   7 11  9 15  5  8   5   4   2   0   3   0   2   1   0   1   0   0   0   1   1   1   0
90   2  9 12  7 10  5   5   4   4   4   1   2   2   2   2   1   0   2   0   1   0   1   0
105  1  3  6  7  4  2   3   3   1   2   2   0   0   0   0   1   0   0   1   0   0   0   0
120  1  0  4  1  0  1   3   1   1   2   0   0   1   0   1   0   1   1   1   1   0   0   0
135  0  1  0  1  2  1   0   0   1   0   0   3   1   2   0   0   0   0   0   0   0   1   0
150  0  1  1  1  2  0   0   1   1   0   3   0   1   0   0   1   0   0   0   1   0   0   0
165  0  1  2  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
180  4  0  1  1  0  3   1   0   0   2   0   0   0   1   0   0   0   0   1   0   1   0   0
195  2  1  1  1  2  2   0   1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
210  0  2  0  1  1  0   1   1   1   0   0   1   1   1   0   0   1   0   0   0   0   0   0
225  0  0  1  0  0  2   1   1   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
240  3  0  1  0  1  1   0   1   0   1   0   1   0   0   0   0   0   1   0   0   0   0   0
255  0  0  0  0  0  0   0   0   0   0   1   0   0   0   1   0   1   0   0   0   0   0   0
285  0  0  0  0  0  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
300  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
315  0  0  1  0  1  1   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
330  0  1  0  1  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
345  0  0  0  0  0  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 210, under-estimation: 127, correct: 26
```

```
> rmse(data_test$pred, data_test$fht)
[1] 91.15648
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 66.77686

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   0  0  1  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
60   1  2  4  9  3  5   2   1   1   0   1   1   2   0   0   0   1   0   1   0   0   0   0
75  16 24 25 19 17 13  15  12   8   6   7   2   4   3   3   2   0   1   0   2   2   3   0
90   5  6 14  8 12  6   4   6   2   3   2   1   0   0   2   1   1   3   1   2   0   0   0
105  0  3  1  6  2  4   2   2   2   1   1   3   1   3   0   0   2   0   0   0   1   0   1
120  0  2  1  0  0  1   1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
135  1  2  2  0  0  0   0   1   1   1   0   2   1   2   0   0   0   0   0   0   0   0   0
150  0  0  0  1  1  0   0   0   0   1   0   1   0   0   0   0   0   0   1   0   0   0   0
195  0  0  1  2  0  1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
285  0  0  0  0  0  0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 171, under-estimation: 155, correct: 37
> rmse(data_test$pred, data_test$fht)
[1] 75.92406
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 54.87603
```

Code D.12: Output of the Tree-Augmented Naive Bayes algorithm applied to the section TOBS set, predicted with FHT bins of 15 minutes. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.2, the time-variables night, contr_working_hours and rush_hour are replaced by a continuous time-variable tijd_begin, representing the reporting time of the incident. The warm-variable is brought back to the continuous temp_max temperature-variable where it was originally discretized from. The variable Theoretisch_vervangingsjaar, that was discretized according to section 6.2.2, is taken as a numerical variable representing years of (re)placement.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   3  1  3  1  1  0   0   1   2   1   1   0   0   0   0   0   0   0   0   1   1   0   0
60   3  8  8  6  6  6   5   3   2   2   3   1   1   0   0   0   0   0   0   0   1   0   0
75   4  9 10 15  7  7   8   3   2   3   1   3   0   2   1   1   1   0   2   0   0   0   0
90   5  8 12 12 12  4   5   6   6   2   3   3   1   4   0   1   0   2   0   0   1   2   0
105  2  3  4  5  3  5   4   5   0   2   2   0   2   0   0   0   1   1   0   0   0   1   0
120  3  4  6  2  3  1   1   2   0   0   0   1   1   2   1   1   0   0   0   1   0   0   0
```

```
135  0  2  4  1  1  4  1  0  1  1  1  1  1  0  2  1  1  0  1  2  0  0  0
150  0  1  1  2  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0  0  0  0  1
165  2  3  0  0  1  0  0  0  0  1  0  1  0  0  1  0  1  1  0  0  0  0  0
180  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
195  0  0  1  1  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
210  0  0  0  0  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
225  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
240  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
# over-estimation: 187, under-estimation: 148, correct: 28
> rmse(data_test$pred, data_test$fht)
[1] 77.35557
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 58.01653

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345
45   1  2  2  2  2  0   0   0   2   3   1   2   2   0   0   0   0   0   0   0   1   0   0
60   6  6 10  7  3  1   4   3   2   2   1   0   2   0   2   1   0   0   0   0   0   1   0
75   9 12 18 19  4  7   9   7   3   2   5   2   3   0   1   0   2   0   1   1   0   0   0
90   1  8  8  8  8  8   4   9   2   3   1   1   0   4   1   2   0   0   0   2   0   1   0
105  3  3  4  3  6  2   3   1   3   0   1   1   0   0   0   0   1   2   0   1   1   1   1
120  2  4  2  1  1  2   2   1   2   1   1   2   1   3   0   1   0   0   0   0   0   0   0
135  1  1  4  0  5  3   0   1   1   0   1   1   0   0   1   0   1   2   0   0   0   0   0
150  0  2  0  2  4  5   0   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0
165  0  0  0  1  0  0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
180  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
195  0  0  0  1  2  1   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
210  0  1  1  0  0  0   1   0   0   1   0   0   0   0   0   0   0   0   1   0   0   0   0
225  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
285  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 193, under-estimation: 146, correct: 26
> rmse(data_test$pred, data_test$fht)
[1] 78.52067
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 59.13223
```

Code D.13: Output of the kNN algorithm applied to the section TOBS data set. Predicted with FHT bins of 15 minutes, $k = 20$, and $l = 0$. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

D.5.3 Collision/hindrance incidents

Both the TAN and kNN algorithm are applied to the section TOBS twice: once with all features presented in section 6.2.3, and once with only the features that are known when the initial prognosis is made. For the second case, this meant deleting `tao_ind` and `counterpart`. To decrease under-estimation, 30 minutes are added to all predicted FHTs.

Tree-Augmented Naive Bayes

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225
45   0  0  1  1  0  0   0   0   0   0   0   0   0   0   0
60   1  1  0  3  2  1   3   1   0   0   0   1   0   0   0
75   1  1  2  0  1  1   0   0   0   0   0   0   0   1   0
90   2  5  5  9  3  5   0   3   1   1   1   0   0   0   0
105  2  2  0  0  1  2   0   0   1   2   1   0   1   0   1
120  1  1  3  1  3  1   0   2   0   1   0   0   0   0   0
135  0  0  0  0  0  1   0   0   0   0   0   0   1   0   0
150  0  1  1  0  0  0   1   0   0   0   1   1   0   0   0
165  0  1  1  2  0  0   0   1   0   1   0   0   0   0   0
225  0  0  0  0  1  0   1   0   0   0   0   0   0   0   0
255  0  0  0  0  0  0   1   0   0   0   0   0   0   0   0
270  0  0  0  0  0  0   1   0   2   0   0   0   0   0   0
# over-estimation: 63, under-estimation: 28, correct: 12
> rmse(data_test$pred, data_test$fht)
[1] 65.53513
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 51.91176

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225
45   0  1  1  0  0  0   0   0   0   0   0   0   1   0   0
60   1  1  3  2  3  0   1   3   1   2   2   0   0   0   0
75   1  1  1  2  2  2   1   0   1   0   0   1   0   0   0
90   3  6  6 11  1  5   2   3   1   2   1   1   1   1   0
105  1  1  0  0  3  2   1   0   0   0   0   0   0   0   1
120  0  0  0  0  0  0   1   0   1   0   0   0   0   0   0
135  1  1  2  1  2  2   0   0   0   1   0   0   0   0   0
150  0  0  0  0  0  0   1   1   0   0   0   0   0   0   0
330  0  1  0  0  0  0   0   0   0   0   0   0   0   0   0
# over-estimation: 58, under-estimation: 33, correct: 11
> rmse(data_test$pred, data_test$fht)
[1] 62.80221
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 47.05882
```

Code D.14: Output of the Tree-Augmented Naive Bayes algorithm applied to the collision/hindrance data set, predicted with FHT bins of 15 minutes. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

k-Nearest-Neighbors

Of the features mentioned in section 6.2.3, the time-variables night and working_hours are replaced by a continuous time-variable tijd_begin, representing the reporting time of the incident.

```
# All variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
45   0  0  0  0  0  0   0   1   0   0   0   0   0   0   0   0
60   1  1  2  2  0  1   1   0   0   1   1   0   1   0   0   0
75   1  2  0  1  4  1   1   2   0   0   0   0   0   1   0   0
90   2  5  4  6  3  3   4   0   0   1   1   0   1   0   0   0
105  2  1  2  1  2  1   1   2   0   1   0   3   0   0   1   0
120  0  1  2  5  2  1   0   2   1   0   0   0   1   0   0   0
135  1  0  1  0  0  1   0   0   1   0   0   0   0   0   0   0
150  0  1  1  0  0  0   0   0   1   1   0   0   0   0   0   1
165  0  1  0  0  0  1   0   0   1   0   0   0   0   0   0   0
180  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0
195  0  0  0  0  0  0   0   0   0   0   1   0   0   0   0   0
225  0  0  0  1  1  0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 61, under-estimation: 28, correct: 14
> rmse(data_test$pred, data_test$fht)
[1] 62.68855
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 49.66019

# Only initially known variables used as evidence
> table(data_test$pred, data_test$fht)
    15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 255
45   0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0
60   0  3  0  2  2  0   2   1   0   1   0   0   0   0   0   0
75   1  2  3  1  1  3   1   0   0   0   1   0   1   0   0   1
90   2  1  7  7  2  5   4   3   1   1   0   1   0   1   0   0
105  2  4  2  0  2  1   0   1   2   2   1   1   1   0   0   0
120  0  1  0  2  2  0   0   0   0   0   0   0   1   0   1   0
135  1  0  0  0  2  1   0   0   0   0   1   1   0   0   0   0
150  0  1  0  1  0  0   0   0   1   0   0   0   0   0   0   0
165  1  0  0  0  0  0   0   1   0   0   0   0   0   0   0   0
180  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0
195  0  0  0  0  0  0   0   1   0   0   0   0   0   0   0   0
210  0  0  0  1  0  0   0   0   0   0   0   0   0   0   0   0
300  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0   0
# over-estimation: 58, under-estimation: 37, correct: 8
```

```
> rmse(data_test$pred, data_test$fht)
```

```
[1] 64.45816
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 50.67961
```

Code D.15: Output of the kNN algorithm applied to the collision/hindrance data set. Predicted with FHT bins of 15 minutes, $k = 10$, and $l = 0$. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

D.5.4 Test sets for comparison

For both section TOBS incidents and collision/hindrance incidents, I randomly selected 10 data points occurring in both Spoorweb and my SAP-based data set. The outputs of the code in this section contain error rates of my predictions of these 20 data points, using both the TAN and kNN algorithm, and the predictions of CQM for the same incidents. To decrease under-estimation, 30 minutes are added to all predicted FHTs.

Tree-Augmented Naive Bayes

```
# Section TOBS data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
60   1  0  0   0   0   1   0   0
75   0  1  0   1   0   0   0   0
90   0  0  0   0   0   0   1   1
105  1  0  0   0   1   0   0   0
150  0  0  1   0   0   0   0   0
165  0  0  0   1   0   0   0   0
# over-estimation: 5, under-estimation: 5, correct: 0
> rmse(data_test$pred, data_test$fht)
[1] 92.9516
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 75

# Section TOBS data, only initially known variables
> table(data_test$pred, data_test$fht)
   30 45 60 105 120 165 195 300
60  0  0  0   0   1   0   1   0
75  1  1  1   1   0   1   0   0
90  1  0  0   1   0   0   0   1
# over-estimation: 4, under-estimation: 6, correct: 0
> rmse(data_test$pred, data_test$fht)
[1] 90.49862
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 69
```

```
# Collision hindrance data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
75   1  1  0  0   0
90   1  0  1  0   1
105  1  0  1  0   0
120  1  0  0  0   0
165  1  0  0  0   0
195  0  0  0  1   0
# over-estimation: 9, under-estimation: 1, correct: 0
> rmse(data_test$pred, data_test$fht)
[1] 77.0714
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 69

# Collision hindrance data, only initially known variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
75   1  1  0  0   0
90   3  0  2  1   1
105  1  0  0  0   0
# over-estimation: 9, under-estimation: 1, correct: 0
> rmse(data_test$pred, data_test$fht)
[1] 49.97499
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 46.5
```

Code D.16: Prognoses made by the Tree-Augmented Naive Bayes algorithm, for 10 data points of the section TOBS data and 10 points of collision/hindrance data. To all predicted FHTs, 30 minutes are added to decrease under-estimation.

k-Nearest-Neighbors

```
# Section TOBS data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
75   1  1  0   1   0   1   0   0
90   1  0  0   0   0   0   0   0
105  0  0  0   0   1   0   1   0
120  0  0  0   0   0   0   0   1
165  0  0  1   0   0   0   0   0
315  0  0  0   1   0   0   0   0
# over-estimation: 5, under-estimation: 5, correct: 0
```

```
> rmse(data_test$pred, data_test$fht)
[1] 105.5344
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 85.5

# Section TOBS data, only initially known variables
> table(data_test$pred, data_test$fht)
    30 45 60 105 120 165 195 300
45   0  0  0   1   0   0   0   0
75   1  1  0   0   0   0   0   0
90   1  0  0   0   0   0   0   0
120  0  0  0   1   1   0   1   0
135  0  0  0   0   0   0   0   1
150  0  0  1   0   0   1   0   0
# over-estimation: 5, under-estimation: 4, correct: 1
> rmse(data_test$pred, data_test$fht)
[1] 71.78092
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 55.5

# Collision/hindrance data, all variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
45   0  1  1  0   0
75   1  0  0  0   0
90   0  0  1  0   0
105  1  0  0  0   0
135  1  0  0  0   0
165  1  0  0  0   0
180  0  0  0  0   1
195  1  0  0  1   0
# over-estimation: 8, under-estimation: 1, correct: 1
> rmse(data_test$pred, data_test$fht)
[1] 89.74965
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 72

# Collision/hindrance data, only initially known variables
> table(data_test$pred, data_test$fht)
    30 45 60 75 150
60   1  0  0  0   0
75   1  1  0  1   0
90   0  0  2  0   0
135  2  0  0  0   0
165  1  0  0  0   0
180  0  0  0  0   1
```

```
# over-estimation: 9, under-estimation: 0, correct: 1
> rmse(data_test$pred, data_test$fht)
[1] 68.41053
> data_test$pred_error = abs(data_test$pred - data_test$fht)
> mean(data_test$pred_error)
[1] 54
```

Code D.17: Prognoses made by the k-Nearest-Neighbors algorithm, for 10 data points of the section TOBS data and 10 points of collision/hindrance data. To all predicted FHTs, 30 minutes are added to decrease under-estimation.